Digital Forensics in Small Devices: RFID Tag Investigation

Volume 1

AR KAR KYAW
B.Eng.Tech. (Massey University, NEW ZEALAND), B.E. (Mandalay Technological University, BURMA)

A thesis submitted to the Graduate Faculty of Design and Creative Technologies
AUT University
in partial fulfilment of the
requirements for the Degree of
Master of Forensic Information Technology

School of Computing and Mathematical Sciences

Auckland, New Zealand
2010

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person, nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning, except where due acknowledgement is made in the acknowledgements.

.....

Signature

Acknowledgements

This thesis was completed at the Faculty of Design and Creative Technologies of the AUT University in New Zealand. While conducting the research project I received support from many people in one way or another, without whose support, this thesis would not have been completed in its present form. It is my pleasure to take this opportunity to thank all of you, without the intention or possibility to be complete. I would like to apologize to those who I have not mentioned by name here; however, I highly value your kind support.

Firstly, I would like to deeply thank my two promoters, Dr. Brian Cusack and Dr. Dave Parry. Dr. Cusack provided me with the freedom to explore research directions and to choose the routes that I wanted to investigate. Dr. Cusack's encouragement, excellent guidance, creative suggestions, and critical comments have greatly contributed to this thesis. Dr. Cusack, I would like to thank you very much for your daily supervision. I enjoyed our discussions and had learned a great deal from you. I also would like to thank Dr Parry, who has kindly agreed to be my secondary supervisor. Dr. Parry has taught me many valuable lessons during the first year of my MFIT study. I strongly believe that what I have learned during the MFIT study period will be infinitely profitable for the rest of my life. For this I am eternally grateful. Thank you, from depth of my heart!

I would also like to acknowledge the friendship, mentorship and support of my fellow MFIT students. Similarly, I would like to thank my colleagues at Whitireia New Zealand for their support. Without their support and encouragement, my study life at AUT would have been more difficult. I would like to thank all of you. In addition, I would like to express my deep appreciation to all of the lecturers who gave me knowledge as well as taught me the concepts during the MFIT lectures.

Next, I would like to extend my appreciation to Mr. Dean Yamagata (Sales Executive ANZ, Tripwire Inc.,) for providing me with the evaluation versions of the Tripwire Software (Tripwire for Servers and Tripwire for Manager) and Mr. Grant Pugh (Tracient Technologies Ltd., New Zealand) for the RFID (Padl-R UF Reader) product support.

Finally, I would also like to express my sincere thanks to my late father, (U Kyaw Myint), and my mother, Daw Tin Tin Win, whose continual support, encouragement, love, praying for my progress and for teaching me the values in life that brought me where I am today. I greatly appreciate my wife, Sunkeum Ji, for also providing continual support and taking care of our four-year-old son Andrew Kyaw during my MFIT study. I greatly appreciate my younger brother and two sisters in Burma for fulfilling my duty of taking care of our mum while I pursued this MFIT degree.

Abstract

Read/Write Radio Frequency Identification (R/W RFID) chips are commonly used to tag stock in retail shops. The security risk of RFID has been well established in the literature and hence there is potential for fraudulent use of RFID networks in commercial settings. This study proposes the identification of all possible data storage locations in a RFID system, a method for forensic extraction of the data, preservation, analysis and best practice recommendations for digital forensic investigators working on RFID systems. The research shows that it is possible to identify theft from a RFID business system (RIFD BS) after a tag poisoning attack.

In order to conduct the proposed research, a trial system was set up in the lab to simulate a commercial retail situation where theft occurred. The normal operation of the trial system was documented as the trusted operation of a stable RFID retail system. The simulation context was the retail environment of clothing and electronic goods, as in such environments Stock Items (SI) could vary in price from a few dollars to tens of thousands of dollars. Hence, the stabilized BS was stressed by using a malicious poisoning attack to change the value of the stock item from the backend SQL Server. Then the entities of BS such as SI, point of sale (POS), business information system (BIS) were investigated in order to locate the potential evidence for the theft of the SI.

The methodology used in a simulated environment was based on descriptive methods in which the case scenario of the replicated SQL poisoning attack through a R/W RFID Tag was initiated. To investigate the presence of digital evidence after the theft of a SI, a customized RFID middleware and *ReaderLogExtraction* tool (to acquire bit-to-bit evidence from RFID reader's memory) were developed based on Software Development Kit (SDK) of the RFID reader's manufacturing company. Live forensic investigation was performed by using customized incident response toolkits (Helix_RFID_IR and *dcfldd* toolkits) and a hardware write-blocker. The descriptive methodology allowed the elaboration of precise details relevant to the research question.

The purpose of the main research question was to perform the complete and accurate forensic examination of a compromised RFID stock management system in the retail sector. As a result, the digital forensic investigation (DFI) procedures such as acquisition, extraction, preservation and analysis of potential evidence data were carried out in a forensically sound manner. Hence, such DFI procedures were the important phases of this descriptive research. During the forensic investigation, the traces of evidence after the poisoning attack were able to be identified, acquired, preserved, analysed and presented according to DFI principles and guidelines. The significant findings were found in each entity of the simulated RFID BS. For instance, changes to the original values of SIs were found in the backend database and evidence of the malicious poisoning code was found in the transaction log of the backend server. Moreover the evidential traces of the fake tag ID, date and timestamp were also found in the memories of RFID reader and POS host station.

In the simulated research experiment, the theft of a SI through an orchestrated test scenario of hardware, software and social engineering illustrates the potential vulnerability of RFID based retail systems. To conclude, the research conducted confirms that the DFI procedures used were able to obtain viable digital evidence from a compromised RFID stock management system in the retail sector. The learning from conducting this research provided not only additional knowledge in DFI of RFID BS, but also best practices for digital forensic researchers and practitioners. Consequently, providing assurance of the DFI process to present trustworthy digital evidence could prove the theft of stock items in a RFID retail system.

Table of Contents (Volume 1)

Chapter 1 – Introduction

Chapter 2 – Literature Review

Chapter 3 – Research Methodology

Chapter 4 – Research Findings

Chapter 5 – Research Discussion

Chapter 6 – Conclusion

List of Tables

List of Figures

Abbreviations

AC	Alternate Current
ALE	Application Level Event
BIS	Business Information System
BS	Business System
CBV	Core Business Vocabulary
CRC	Cyclic Redundancy Code
DC	Direct Current
DCI	Discovery Configuration and Initialization
DFRWS	Digital Forensic Research Workshop
DNS	Domain Name System
DoS	Denial of Service
EMAP	Efficient Mutual Authentication Protocol
EEPROM	Electrically Erasable Programmable Read-Only Memory
EPC	Electronic Product Code
EPCglobal	Electronic Product Code Global
EPCIS	Electronic Product Code Information Services
ERP	Enterprise Resource Planning
FRAM	Ferroelectric Random Access Memory
Helix_RFID_IR	Helix_Radio Frequency Identification_Incident Response
HF	High Frequency
IC	Integrated Circuit
ID	Identification
IEC	International Electro-technical Commission
IS	Information System
ISO	International Organization for Standardization
IT	Information Technology
ITF	Interrogator-talks-first
JTC 1	Joint Technical Committee 1
KHz	Kilo-Hertz

LF	Low Frequency
LLRP	Low Level Reader Protocol
MAC	Media Access Control
MHz	Mega-Hertz
NIJ	National Institute of Justice
ONS	Object Name Service
POS	Point of Sale
RFID	Radio Frequency Identification
RFID BS	Radio Frequency Identification Business System
RAM	Random Access Memory
RM	Reader Management
RO	Read Only
RP	Reader Protocol
RW	Read/Write
R/W RFID	Read/Write Radio Frequency Identification
SAC	Surface Area Configuration
SC 31	Sub-Committee 31
SG 1	Sub-Group 1
SI	Stock Item
SID	Session Identification
SRAM	Static Random Access Memory
TDS	Tag Data Standard
TDT	Tag Data Translation
TM	Tripwire for Manager
TS	Tripwire for Servers
TPS	Transaction Processing System
UHF	Ultra High Frequency
TID	Tag Identification
WFT	Windows Forensic Toolchest
WGs	Working Groups
WORM	Write Once/Read-Only Memory

Chapter 1 -

Introduction

1.0 INTRODUCTION

The goal of ubiquitous computing, as described by Mark Weiser (1991 & 1993, p. 75), is "*making many computers available throughout the physical environment, while making them effectively invisible to the user".* However, the border line between the virtual and physical world has become faded as the technologies in the age of ubiquitous computing (for examples; sensor networks, Global Positioning Systems, Radio Frequency Identification Systems and the like) enable those ubiquitous devices to connect with each other and to the Internet. Crime and criminal activity has also flourished as a result of the new technology opportunities. The exposing or theft of personal information, location and context information are the challenges related to security (Juels, 2005; Bhargava, 2006; Albrecht, 2006; Ding & Bo, 2008) and privacy in ubiquitous devices and technologies (Garfinkel, Juels, & Pappu, 2005; Anthony, Kotz, & Henderson, 2007).

Radio Frequency Identification (RFID) is one of the emerging and promising ubiquitous computing technologies (Roberts, 2006) applied in an extensive range of applications such as ticketing (Nath, Reynolds, & Want, 2006), healthcare (Borriello, 2005), payment systems like *Speedpass* (Haines, 2006a), stock management systems in retail environments like *Future Store* in Germany (Haines, 2006b) and supply-chain management in *Wal-Mart* (Sheng, Li, & Zeadally, 2008) and so on. RFID, which uses wireless communications technology, is convenient for automatic identification of items or objects by tagging them with RFID tags (Borriello, 2005) and is analogous to a bar code system (Roberts, 2006). Hence, the Read/Write RFID tags are used in the retail environment to tag stock with an ID that provides two utility values for the retailer, namely theft protection (for example, the tag is removed at the point of sale so the exit sensors do not alarm) and also mapping into the retail database for price and stock control – this includes the point of sale (POS) process and inventory control

process (see Chalasani, Boppana, & Sounderpandian, 2005). As a result of the low cost and improved capabilities of RFID tags, the tags are employed in many applications. However, the employment of the tags does not only bring benefits (such as ease of stock management, and the anti-counterfeiting of medicines) but also give the significant challenges and threats (see Section 2.4 in Chapter 2) to the businesses or organizations. As a consequence, the objective of this research is to investigate the presence of digital evidence after the theft of a RFID tagged stock item (SI) in a forensically sound manner.

Hence, the purpose of this chapter is to present an overview of the problem or a gap in the current literature (Section 1.1), the motivation of the research (Section 1.2), a brief presentation of main research findings (Section 1.3) and the structure of thesis (Section 1.4).

1.1 THE PROBLEM OR GAP IN CURRENT LITERATURE

The potential for fraudulent intervention into a RFID system is established in the literature (Chapter 2; Section 2.4). Some researchers (for example: Ngai, Moon, Riggins, &Yi, 2008) provide a comprehensive 1995 – 2005 review relevant to RFID research. Other researchers (Jones, Hoare, Dontharaju, Shenchih, Fazekas, Cain, & Mickle, 2006a) describe the evolution of RFID tag design and process time costs. In the manufacture of chips, the access to writing has become easier in the interest of letting retailers code their own chips. The benefit has however brought with it easier access for criminals to write and rewrite over chips and hence change the values (including the null value) in a RFID system. One of the concerns for the RFID based retail environment is that as tags are made more user friendly, then the risk of attack grows. Hence, most of the previous literature reported the security risk concerned with RFID systems. Consequently, it is exposed that the requirement of forensic readiness is necessary as there is potential for the fraudulent uses of RFID networks in commercial settings.

On the other hand, the digital forensic research into small devices is a current growth area of knowledge (Harrill & Mislan, 2007). In the Digital Forensic Research Workshop (DFRWS) road map for Digital Forensic research, small devices are again prioritised as key areas for research and growth of investigator knowledge (Palmer, 2001). Similarly, some researchers (Xiao, Yu,

Wu, Ni, Janecek, & Nordstad, 2007) raise a number of research issues associated with RFID investigations.

Nevertheless, each of these references has guidance for research and specific details of small-device forensic techniques. The literature reviewed (Chapter 2) shows that research is leading towards knowledge of better system architectures and to improve best practice guidance knowledge for digital investigators (see Chapter 5; Section 5.3). The National Institute of Justice (NIJ) Report (2005) also continues the theme by specifying best forensic investigator practice for small devices – but does not report RFID chip practice. Therefore, this appears to be a gap in the current professional literature.

1.2 MOTIVATION OF THE RESEARCH

RFID tags are used in the commercial retail sector for stock management (Chalasani, Boppana, & Sounderpandian, 2005). A tag is attached to a SI so that the identity of the SI is accessible by digital scanning. The cost advantage is apparent in various stock management processes including audit, transaction, and entry (for example in a book shop the contents of a carton may be individually scanned and data matched without opening a carton). However, the risk of fraudulent exploitation of RFID stock management systems escalates when more expensive tags are used (usually on more expensive SI). The higher risk tags are read/write and/or active. The appetite for risk also escalates with higher valued SI and the parallel increase in opportunity to crack a system. The utility value to the retailer of RFID tags is transaction efficiency and inventory control but the trade-off is trust in the system. The violation of trust may occur in many ways and be demonstrated by educated theft of property. The residual risk of system violation requires a forensic readiness capability that can inform the system security module on how best to treat a risk. In addition, the evidence requires identification, preservation, and analysis so that perpetrators may be prosecuted. The retailer has balanced a cost-benefit analysis to invest in such a stock management system but may not realized the forecasted benefits in the event of negative control risks materializing (Altschaffel, Kiltz, & Dittmann, 2009).

The research interest in this project is to scope the risk of violation in a RFID stock management system (the Business System), to document the location

of potential evidence after an event, to analyse the collected potential evidence and to recommend best practice for both system security and forensic investigation based on the findings of the experiment. A laboratory simulation is set up to replicate previously published security violations of RFID systems, and then the Business System is forensically investigated in order to locate potential evidence. The simulation context is the retail environments of clothing and electronic goods. In both environments, SI can vary in price from a few dollars to tens of thousands of dollars. The environments are high intensity with large numbers of SI, high transaction rates, many entry level sales assistants, high staff turnover, and generic brand systems architectures. The Read/Write RFID tag is often deployed to SI in these environments by press clip attachment (with optional band extension) (Jones, Hoare, Dontharaju, Shenchih, Sprang, Fazekas, Cain, & Mickle, 2006). The attachments are released at the POS with a design tool for future re-writing and re-attachment to another SI. The business process is cost effective and the Business System integrity is theoretically maintained (Jeng, Chang, & Wei, 2009).

The thesis is structured to first define a RFID tag Business System (RFID BS) and then to elaborate on the specific security risks associated with this Business System (Rotter, 2008). The Business System is defined (in Table 2.5 in Chapter 2) and the potential for system violation is identified between the Business System entities, within the entities (for instance: tag cloning), and through social engineering (Figure 2.14 in Chapter 2). It is assumed the Business System is closed so that the event of an item being thrown out of a window undetected or breaking and entering are excluded from the study. The case of a Business System is violated by SQL injection attack through the Read/Write RFID Tag is replicated (Haines, 2006b; Rieback, Crispo, & Tanenbaum, 2006b). The RFID BS is investigated for evidence remaining after the attack and suggestions made for where to look for evidence, how to extract evidence, how to preserve and analyse the evidence (Michael, & McCathie, 2005; Masters, & Turner, 2007; Harrill, & Mislan, 2007; Khannaa, Mikkilinenia, Martonea, Alia, Chiub, Allebacha, & Delpa, 2006; Martone, Mikkilineni, & Delp, 2006). Therefore the relevant research question is: *What should the forensic investigator do in order to perform the complete and accurate forensic examination of a compromised RFID stock management system in the retail sector?*

1.3 FINDINGS OF THE RESEARCH

The research summarizes findings relevant to the digital forensic investigation of a compromised RFID based stock management system. The findings from the investigation prove that the theft of SI has occurred in a RFID based retail system. Hence, the perpetrator could be identified and later be prosecuted according to the court of law if more evidence (which are the limitations of this research; see Section 3.4 in Chapter 3) such as images from the Closed Circuit Television (CCTV) monitoring system and the interview evidence from human participants are taken into consideration during the investigation.

In order to investigate the presence of digital evidence after theft of SI, a prototype of stabilized commercial retail environment using RFID stock management system is constructed in the laboratory according to the proposed system design (Section 3.3.1). As part of the proposed system design, a customized RFID middleware software (source code can be seen in Appendix 4) is developed based on the Software Development Kit (SDK) of RFID reader's manufacturing company. The developed programme allows users to configure the RFID reader, to collect the tag data, and write the tag data to the backend database. Basically, it enables reading and writing the tag data from the RFID scanner to the BIS SQL 2005 Server (and vice versa). By design, one real and one poisoned RFID Read/Write tags are also applied in a stabilized system. The Tripwire for Servers (Version 4.8) and Tripwire Managers (Version 4.8) are utilised to establish the baseline of a stabilized/trusted system in operation. Hence, the previous studies such as compromising the backend database by using a RFID tag as an attack vector (Section 3.1.2), and the acquisition of backend SQL server artefacts methods (Sections 3.1.3 and 3.1.4) are replicated in this simulated research. Subsequently, SQL poisoning attack (Section 3.3.2) is initiated via a fake tag to compromise the backend database of the stock management system and then each entity in the RFID BS (see Table 2.5 in Chapter 2) such as the tag, POS, scanner, and the backend SQL Server are investigated for evidence of the SI theft. Thus, the evidence is collected from the tag, the scanner, the POS and the backend SQL server. However, the *ReaderLogExtraction* tool (see Appendix 3) is developed based on SDK of RFID reader's manufacturing company in order to acquire bit-to-bit evidence data from the RFID reader's memory during the

investigation. Likewise, all the required software forensic tools for data collection (such as *ReaderLogExtraction* tool, WinEn, extended WFT, and the like) are integrated into the customized Helix_RFID_IncidentResonse (Helix_RFID_IR DVD) toolkit (see Appendix 2 for steps in creating Helix_RFID_IR toolkit) and the live data acquisition is performed by placing RIFD_IR into the compromised machine's DVD drive in order to avoid any modifications or affect minimum impact to the original evidence during the acquisition phase (Jones, Bejtlich, & Rose, 2006b; Jones, 2007; Fowler, 2007, 2009). Hence, the details of the tools and techniques used to collect the digital evidence, findings and analysis of collected data are presented in (Chapter 4).

During the simulation experiment, the traces of evidence after the attack are able to be identified, acquired, preserved and analysed according to the digital forensic principles and guidelines for handling the digital evidence. Hence, the evidence of values of the original stock items are changed in the backend database during the investigation (see Figure 4.37 in Section 4.2.3.4.4). Likewise, the evidence of malicious transaction traces including the tag ID, date and timestamp are found in logs of the RFID reader's memory (Section 4.3.1) and in the memory of the POS station (POS RAM) (Section 4.3.2). Similarly, significant evidence is found in the acquired current table data of the stock management database (Figure 4.37) from the backend SQL Server. Moreover, the acquisition result of current table data of transaction log file - RFID_test_log.ldf (see Figure 4.41) shows the significant evidence of the attack in which the malicious SQL poisoning code is found and the SI (Tag IDs starting with E004) are updated to the values $600 at 06:39:48pm on the 12 October 2010. Hence, a fake tag is also found and preserved as proof of the theft of SI, in the simulation experiment.

1.4 CONCLUSION (STRUCTURE OF THE THESIS)

In conclusion, Chapter 1 introduces the motivational factors in connection with the study of the digital forensic investigation in a compromised RFID tag based stock management system in the retail environment. It also reviews the motivation of the research and emphasizes the gap in the digital forensic research area.

Chapter 2 includes a literature review of relevant research. For instance, the three-tier model of a RFID system such as the material layer (Section 2.1), the

IS layer (Section 2.2) and the business process layer (Section 2.3) is reviewed to understand and identify the processes, model, and concepts that could be applied to the proposed research. Likewise, RFID security threats and challenges (Section 2.4) are reviewed and then the RFID BS security risks model (Figure 2.15 in Section 2.5) is proposed, based on the requirement of the research after reviewing the previous literature.

Chapter 3 outlines the comprehensive review of similar published works (Section 3.1) that leads to the adopted descriptive research methodology used in the experiment research. Similarly, the review of problem areas in RFID stock management systems concerned with the proposed security risk model (Section 3.2.2) is explained. Moreover, the research questions (Section 3.2.6), tested hypotheses (Section 3.2.7), the data requirements (Section 3.3), the limitations (Section 3.4) and the expected outcomes of the research (Section 3.5) are discussed in Chapter 3. Hence, after running the pilot tests; the stabilized RFID BS is compromised with SQL poisoning attack through a fake RFID tag to change the value of the SI and then each entity in the BS (Table 2.5 in Section 2.3) is investigated for evidence of SI theft. The collected evidence will be preserved, analysed and presented to validate the case of SI theft in Chapter 4.

Chapter 4 reports the findings from the simulation of a RFID BS that is designed to simulate the use of RFID tags in a retail shop. The variations encountered or changes made during the experiment are acknowledged and explained in Section 4.1. Likewise, the report of the collected digital evidence is presented (Section 4.2) and the analyses of the acquired evidence from the different entities of RFID BS, such as SI, POS, and Business Information System (BIS) are presented and discussed (Section 4.3).

Chapter 5 discusses the findings of the research, presented in Chapter 4, in relation to the research question, sub-questions and hypotheses in order to evaluate the importance of the results. Hence, the developed research questions (Section 3.2.6) are answered and discussed in relation to the asserted hypotheses in Section 5.1. The findings of the simulated experiment are explained in Section 5.2 to comprehensively evaluate the results of the research findings and experiment implementation. This chapter also discusses the acquired knowledge during the simulated experiment as the recommendations for digital forensic best

practices concerned with the forensic investigation of a compromised RFID stock management system.

Finally, Chapter 6 summarizes the research findings and the answers to the research questions. In addition, the areas of potential future research in RFID based BS are discussed. The conclusion of the thesis is then followed by a list of references and the appendices.

Chapter 2 -

Literature Review

2.0 INTRODUCTION

The focus of this research is on the criminal risk of the Read/Write radio frequency identification (RFID) chip in the retail environment. These chips are used to tag stock with an identification (ID) that provides two utility values for the retailer; namely theft protection (ie. The tag is removed at the point of sale so the exit sensors do not alarm). Similarly, these chips are mapping into the retail database for price and stock control – this includes the point of sale process and inventory control process (Chalasani et al., 2005). RFID chips are divided into passive and active and each type has particular risks. Passive chips are open to "kill" hits and hence the retail exit security can be negated (Chawla & Ha, 2007; El-Said & Woodring, 2009). Active chips with the read/write feature can be hacked in order to change the ID and in practice swap an ID linked to a higher price with an ID linked to a lower price (Haines, 2006a, 2006b; Jeng, Chang, & Wei, 2009).

One of the concerns for retail is that as RFID chips are made more user friendly then the risk of attack grows. Jones et al., (2006a) describes the evolution of RFID chip design and process time costs. In the manufacture of chips the access to writing has become easier in the interest of letting retailers code their own chips. The benefit has however brought with it easier access for criminals to write and rewrite over chips and hence change the values (including the null value) in a RFID system. System solutions are provided by Solanas et al., (2007) and others. Digital forensic research into small devices is a current growth area of knowledge (Harrill & Mislan, 2007). In the Digital Forensic Research Workshop (DFRWS) road map for Digital Forensic research, small devices were again prioritised as key areas for research and growth of investigator knowledge (Palmer, 2001). Xiao et al., (2007) raises a number of research issues associated with RFID investigations and suggests that application research has to be evaluated. Rieback et al., (2006a) explicitly

defines different types of malware that are available for RFID attack and advises middleware developers to proceed in a considerate fashion. Each of these references has guidance for research and specific details of small device forensic techniques.

The literature reviewed shows that research is leading towards knowledge of better system architectures and to better best practice guidance knowledge for digital investigators. For example, Chalasani et al., (2005) discusses possible improvements in network designs and Chalasani et al., (2007) investigates improving the data architectures of the RFID transaction system. The National Institute of Justice (NIJ) Report (2005) also continues the theme by specifying best forensic investigator practice for small devices – but does not report RFID chip practice. This appears to be a gap in the current professional literature. Jeong and Kim (2005) provide guidance for RFID architectures in ubiquitous environments. Much of the knowledge of RFID systems comes from papers reporting security problems and architectures for RFID systems. The security analysts have interrogated RFID systems to understand the risks and vulnerabilities of the systems. As a consequence, the technical definition and scope are made explicit. For example Li et al., (2008) proposes a cross-layer architecture to protect the system and user privacy. Others have taken up the challenge to ask the forensic question of RFID systems. For example, Khanna et al., (2006) spelled out a general method of inquiry for physical devices, and Kim et al., (2007) for RFID systems in particular. Martone et al., (2006) go a step further and examine current forensic identification techniques for data verification in RFID systems.

The proposed study will look at the identification of all possible data storage locations in a RFID system. Also recommendations for digital forensic investigators working on RFID systems for best practice will be made. Thus, the literature review chapter is firstly organized into a three-tier model of the RFID system in the retail sector including a material level (Section 2.1). Different types of tags, readers, the hosts and information system (Section 2.2) are reviewed. For example, where the data on tag is stored, the way in which the data is captured by the reader and transferred to the backend database servers or enterprise applications and business processes of a RFID system (Section 2.3). Then, the

security features and challenges (Section 2.4) will be followed by the RFID stock management security risk (Section 2.5) and conclusion (Section 2.6).

2.1 RFID SYSTEM

RFID (radio frequency identification) is designed to identify the objects or people by using wireless radio communication (Hunt et al., 2007). As every system has its own essential component in order to operate successfully, a typical RFID system is composed of three main components as shown in Figure 2.1. These components are an RFID tag device, which is sometimes referred to as a transponder, RFID reader (or transceiver) and a host or controller which is connected to an enterprise system (Roberts, 2006; Xiao et al., 2007; Hunt et al., 2007).

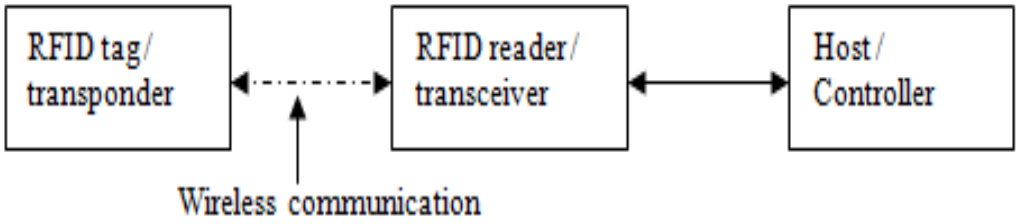


Figure 2.1: Block diagram of a typical RFID system (author)

In general, the first stage of how an RFID system works is that the RFID reader transmits a signal via a radio frequency which is captured by the antenna of the tag. After the tag transmits its stored data, it is received and decoded by the RFID reader. Then, the tag's data is transferred to the host or controller (Xiao et al., 2007). To visualize how an RFID system works, it can be seen in the following Figure 2.2.

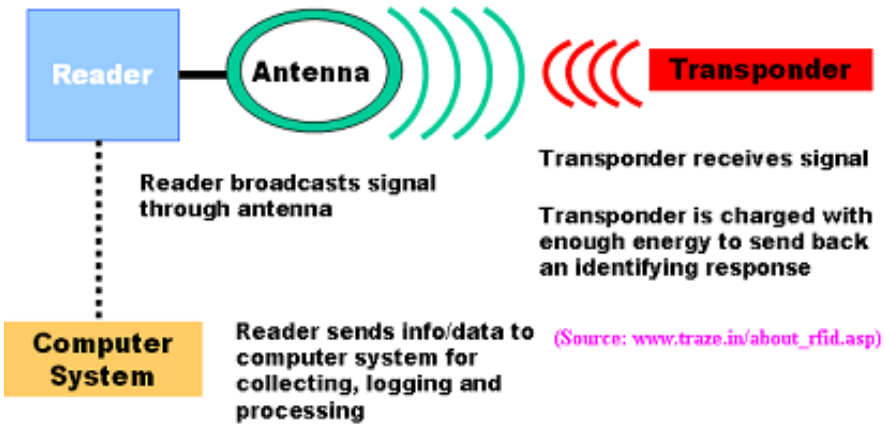


Figure 2.2: How does RFID work? (Source: www.traze.in/about_rfid.asp, p. 1)

2.1.1 RFID Transponders or Tags

The transponders or tags are small items similar to the small portions of paper with a metal printed pattern, as shown in Figure 2.3 below (Xiao et al., 2007).

Figure 2.3: RFID Tags (adapted from: www.uktelematicsonline.co.uk; www.electronics-lab.com; www.associatedcontent.com; www.media.photobucket.com; www.prisms.cs.umass.edu)

Even though the tags come in different types (Figure 2.3), different shapes and different classes; their main functions are not only to store the unique information of an object or a person, but also to respond to the request for information coming from the RFID reader (Hunt et al., 2007; Sanghera et al., 2007).

A tag is basically made up of an electronic chip, antenna and substrate (see Figure 2.4).

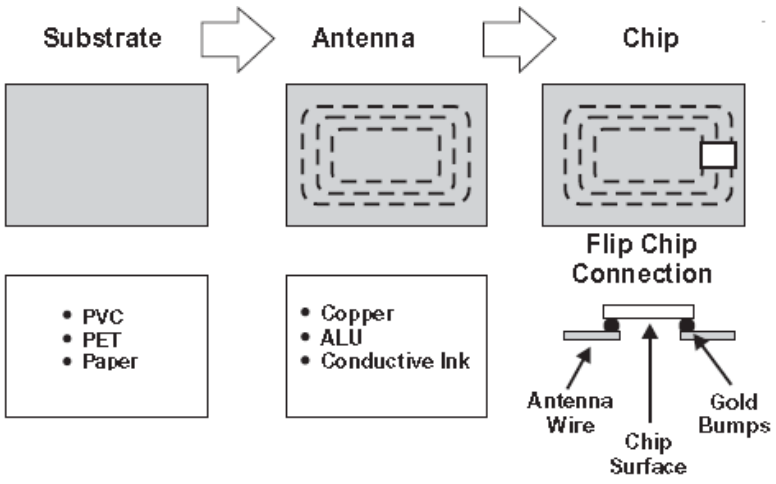


Figure 2.4: Basic Components of RFID Tag (adapted from LARAN RFID; cited in Hunt et al., 2007, p. 7)

An electronic chip connects to the antenna in order to receive or transmit the signal. According to Sanghera et al., (2007, p. 53), the tag consists of "a *logical unit* which is used for implementing protocol for tag-reader communication", "*memory* which is used for storing data", "a *modulator* which is used for modulating the outgoing signal and demodulating the incoming signal", and "*power controller* which is used for converting alternate current (AC) power from the incoming signal to direct current (DC) power supply and supplying power to the components of the chip."

The purpose of the antenna on the chip is to receive a signal from the antenna of the reader and vice visa. It is important to note that the position and design of the antenna determines the range or coverage communication zone between the tag and the RFID reader (Ngai et al., 2008). The substrate for RFID tags accommodates the antenna and the chip.

Depending on whether or not has an on-tag power source (that is a battery), RFID tags are classified into passive, semi-passive and active types. Generally, passive tags which do not have an on-tag power source are read-only (RO) while semi-passive and active tags that have an on-tag power source are Read/Write or Smart tags (Hunt et al., 2007; Xiao et al., 2007; Sanghera et al., 2007). As the tags come in different types and classes; the way they operate, their size, cost and characteristics are also dissimilar. According to the use of radio frequencies, the RFID tag technologies are classified into microwave, ultra high frequency (UHF), high frequency (HF) and low frequency (LF) RFID tags. Similarly, based on the functionality of the tag; there are six tag classes (starting from Electronic Product Code: EPC Class 0 to Class 5) proposed by Auto-ID Centre, Massachusetts Institute of Technology. Sanghera et al., (2007, pp. 61-65) describe the different classes of RFID tags as follow:

"The Class 0 tags are passive and RO tags which normally programme with a unique number during manufacturing. Likewise, the Class 1 and Class 2 tags are passive and write once/read many (WORM) memory tags, and have 128-bit and 65KB of memory space respectively. However, the Class 3 tags are re-writable (can be programmed and reprogrammed by the user many times), semi-passive tags as they have got their own power sources on-tag for supporting a longer read range and have 56KB of memory space. But, they are not capable of initiating communication with the reader.

Table 2.1: Characteristics and Classes of RFID Tags (adapted from: Sanghera et al., 2007, p. 60 & 65; Roberts, 2006, p. 23; Chawathe et al., 2004, p. 1191; Hunt et al., 2007, pp. 12-13)

Tags / Characteristics	Passive	Semi-Passive	Active
Power Source	No power on-tag and receive the power from RFID reader's signal.	Has on-tag power source.	Has on-tag power source.
Communication	Initiated by the RFID reader.	Initiated by the RFID reader.	Can respond to the reader's signal and also initiate the communication.
Size	Small as no need to give space for on-tag power source.	Medium	Largest
Read range	Short (2mm); few meters depending on the operating frequency.	Maximum 100 meters.	Large (maximum 1 kilo-meter); some limitation apply according to the standard and regulations.
EPC Tag classes	0, 1, and 2	3	4* and 5
Life span	unlimited	(5 – 10) years	(1 – 5) years
Memory design	Read only (RO), Write once/read many (WORM), Read/write (RW) for Class 2	Read only (RO), Write once/read many (WORM), Read/write (RW)	Read only (RO), Write once/read many (WORM), Read/write (RW)
Memory capacity	Mostly up to 128 bits, but some tags can have memory up to 64KB	Can be greater than that of a passive tag	Up to 8 MB
Operation frequency	Low frequency (125-134 KHz), High frequency (13.56 MHz)	---	Ultra-high frequency (860-960 MHz), Microwave (2.5 GHz and above)
Cost	Inexpensive	Intermediate	Expensive

***Class 4 tags are re-writable and active tags.**

These Class 4 tags are capable of not only initiating communication with the reader, but also they can communicate with other tags. Class 5 tags, referred to as the reader tag, have the same functionalities as that of Class 4, except they can initiate communication with all classes of tags.

Hence, the main characteristics of different types and classes of RFID tags are summarized as shown in Table 2.1.

2.1.2 RFID Transceiver or Reader

An RFID transceiver or reader (see Figure 2.5), sometimes referred to as an interrogator, is a communication bridge between the tag and the host or controller. A RFID transceiver consists of a transmitter and receiver. There is no need for a line of sight scanning between the tag and the receiver in a RFID system; in the same way the bar code system does (Hunt et al., 2007). The reader queries a tag and received the data stored in the tag. The other parts of the reader include "*antenna* (either integrated or a separate one), *RS-232 serial port or Ethernet jack*, *cryptographic encoding and decoding circuit*, *a power supply or battery* and *communication control circuits*" (Thornton et al., 2006, p. 16).

Figure 2.5: Handheld RFID reader (Source: http://www.prlog.org/10138875-handheld-rfid-reader-dl8033-in-1356mhz.jpg)

Typically, RFID readers have the following basic functions, which are stated by Hunt et al., (2007, p. 9):

> "the readers read the contents of the data, write the data on the tag if the tags are writable, transmit data to and from the host or controller, and give power to the tags if the tags are passive".

Moreover, depending on the complexity of the readers, they can carry out *data protection* for integrity of the data on tag (encryption and decryption), tag *authentication* for preventing unauthorized access, and implementation of *anti-collision algorithms* to enable simultaneous communication with many tags (Hunt et al., 2007).

Furthermore, energizing, demodulating, and decoding are the three main functions of the reader. The reader uses a tuned circuit to emit a frequency field in order to power up the tag (example: energizing the passive tag). The data sent by the tag must be demodulated. The encoded data is decoded by the reader's on-board microcontroller (Kou et al., 2007).

Based on the reader's technology and application, writing to the tags can be performed by some interrogators. There are many types of RFID readers such as handheld, mobile mounted, and the like. However, the basic RFID reader structure includes antenna or coupling unit, memory, supply voltage, radio frequency interface, as shown in Figure 2.6 below (Kou et al., 2007).

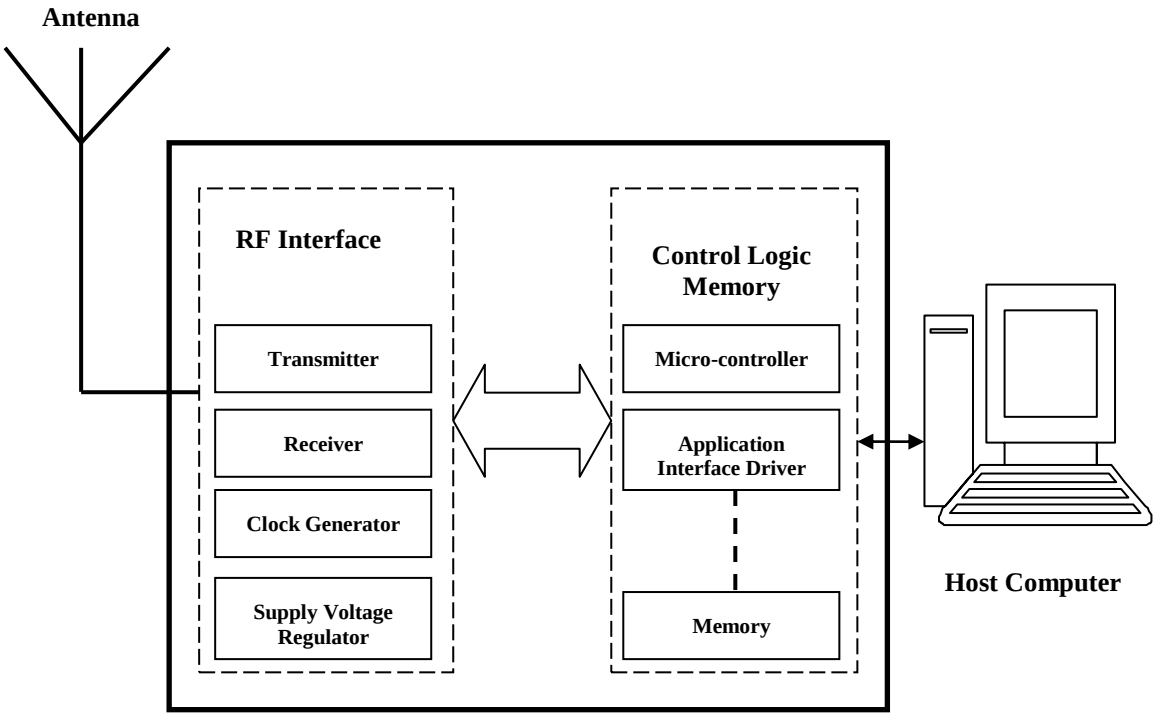


Figure 2.6: RFID reader structure (adapted from: Kou et al., 2007, p. 96)

2.1.3 RFID Host or Controller

Another component of a basic RFID system is a host or controller. RFID host is often a personal computer which is running a database or RFID application

software (middleware) and it centrally processes information coming from multiple readers that are networked together. The data or information is obtained by readers and then passes down to the controller. Hunt et al., (2007) explain some of the functions of the RFID controller are while obtaining information from multiple readers, the host can keep an updated inventory list of a retail system, the host is able to verify, identity and grant authorization in keyless entry systems, and the host can track the movements of the objects throughout the system in a manufacturing application.

2.1.4 RFID Standards

RFID Standards are developed or defined by the International Organization for Standardization (ISO) in order to achieve interoperability of the different RFID devices manufactured by different companies. There are different RFID standards including "*tag–reader air interface specifications, reader–host command specifications, reader network standards, and data formats*" (Su et al., 2008, p. 41).

The RFID air interface operation, is defined by the ISO 18000 Series of standards, comprises physical layer electronic characteristics and data link layer for data communication. This ISO 18000 includes seven parts such as "*Part 1 defines general parameters, Part 2 specifies the parameters for frequency under 125KHz (Low Frequency), Part 3 specifies parameters for 13.56MHz (High Frequency), Part 3 and Part 4 work on 2.45 GHz and 5.8 GHz, respectively, Part 6 works on frequency between 860 MHz to 930 MHz (Ultra High Frequency), and Part 7 works on 433 MHz (used for active RFID)*" (Su et al., 2008, p. 42).

Similarly, ISO 14223 is for the air interface standard and is used for the "*identification of animals in agricultural applications*", whereas the standards ISO14443 and ISO 15693 are defined for "*the air interface and communication protocols for the proximity system and vicinity card*" (Su et al., 2008, p. 42).

Moreover, Su et al., (2008) indicates that there are other standards in addition to air interface standards. For instance, "*the concept, data syntax, business process, and information exchange for item level management*" are defined by ISO 15961-15963; and "*the different types of packaging processes for the logistic and supply chain*" are defined by ISO 17363-17368 (Su et al., 2008, p. 42).

However, according to EPCglobal (Electronic Product Code Global [EPCglobal], 2010), there are two common standards such as data and tag-reader air interface standards (see Figure 2.7).

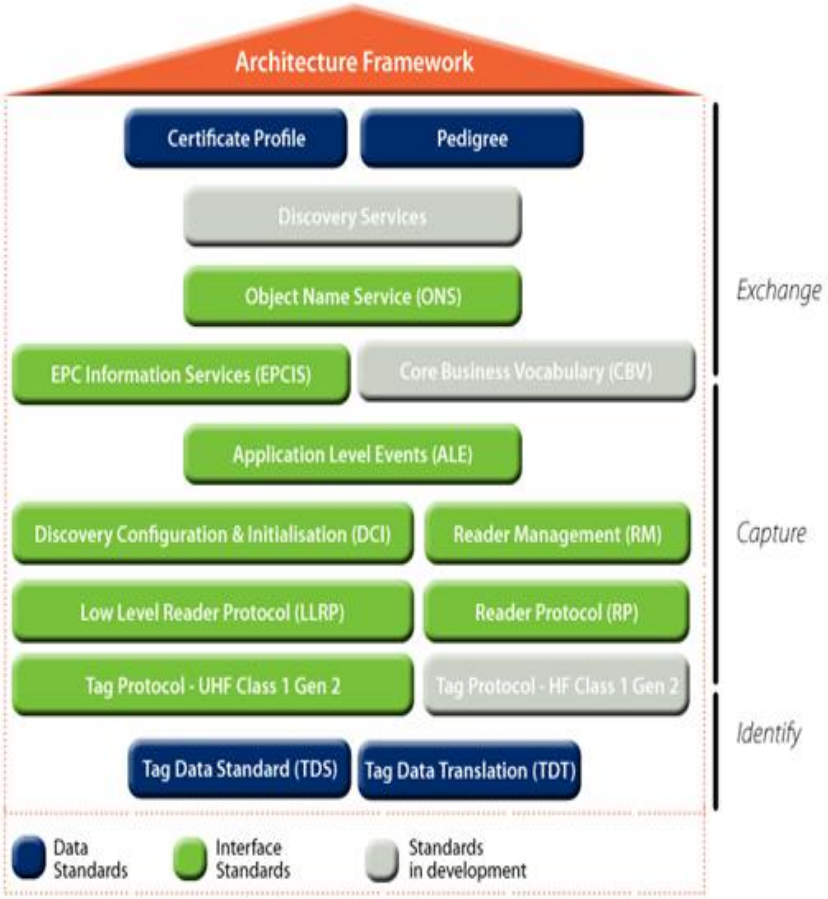


Figure 2.7: EPCglobal Standards Overview (Source: http://www.gs1.org/gsmp/kc/epcglobal)

The data standard includes tag data standard (TDS) that defines "*how data is encoded on the tag and how data is encoded for use in the information systems layers of the EPC Systems Network*" (EPCglobal, 2010) and tag data translation (TDT) specification that explains "*how to interpret the machine-readable version of the EPC tag data standards specification*" (EPCglobal, 2010).

The interface standards are made up of Class 1 Generation 2 UHF air interface protocol standard (known as "Gen 2" standard), reader protocol (RP), low level protocol (LLP), application level events (ALEs), reader management (RM), and the like. These EPC interface standards are summarized in the Table 2.2 below.

Table 2.2: EPC Interface Standards and Their Specifications

EPC Interface Standards	Specifications
Class 1 Generation 2 Ultra High Frequency (UHF) Air Interface Protocol Standard "Gen 2"	"*Defines the physical and logical requirements for a passive-backscatter, Interrogator-talks-first (ITF), RFID system operating in the (860 – 960) MHz frequency range. The system comprises interrogators (Readers), and tags (labels)*" (EPCglobal Inc., 2008, p. 10). Latest version for item level tagging is v1.2.0.
Low Level Reader Protocol Standard (LLRP)	"*Specifies the interface between the RFID readers and clients; providing control of RFID air protocol operation timing and access to air protocol command parameters*" (EPCglobal Inc., 2007, p. 2).
Reader Protocol Standard (RP)	This interface standard "*specifies the interactions between a device capable of reading/writing tags and application software*" (EPCglobal Inc., 2006, p. 17).
Reader Management Standard (RM)	"*The current RM Standard Version 1.0.1 of the wire protocol used by management software to monitor the operating status and health of EPCglobal compliant RFID Readers*" (EPCglobal Inc., 2007, p. 2).
Discovery, Configuration and Initialization (DCI) Standard for Reader Operations	"*Specifies the interface between RFID readers, access controllers, and the network on which they operate* and *identify the necessary and optional operations of a reader and client that allow them to utilize the network to which they are connected to communicate with other devices, exchange configuration, and initialize the operation of each reader, so that the reader operations protocols can be used to control the operation of the readers to provide tag and other information to the client*" (EPCglobal Inc., 2009, p. 5).
Application Level Events (ALE) Standard	"*Specifies a software interface through which client applications may interact with filtered, consolidated EPC data and related data from a variety of sources. In particular, ALE provides a convenient way for applications to read and write RFID tags, interacting with one or more RFID reader devices*" (EPCglobal Inc., 2008, p. 1).
EPC Information Services (EPCIS)	"*Specifies a standard for sharing EPC related information between trading partners. EPCIS provides important new capabilities to improve efficiency, security, and visibility in the global supply chain, and complements lower level EPCglobal tag, reader, and middleware standards*" (Asher et al., 2007, p. 4).
Object Name Service (ONS) Standard	"*Specifies how the domain name system (DNS) is used to locate authoritative metadata and services associated with a given electronic product code*" (EPCglobal Inc., 2008, p. 6).

According to Harmon (2006, pp. 45-47), the Subcommittee 31 (SC 31) of the Joint Technical Committee 1 (JTC 1) between ISO and the International Electro-technical Commission (IEC) addresses the standards of automatic identification and data capture techniques. There are five working groups (WGs) organized by "*ISO/IEC JTC 1/SC 31*" and WG 4 is responsible for standards relating to RFIDs. Specifically, the Sub-Group 1 (SG 1) of "*ISO/IEC JTC 1/SC 31/WG 4/SG 1*" is responsible for RFID data protocols and defines the standards for access to user memory on a tag (Harmon, 2006).

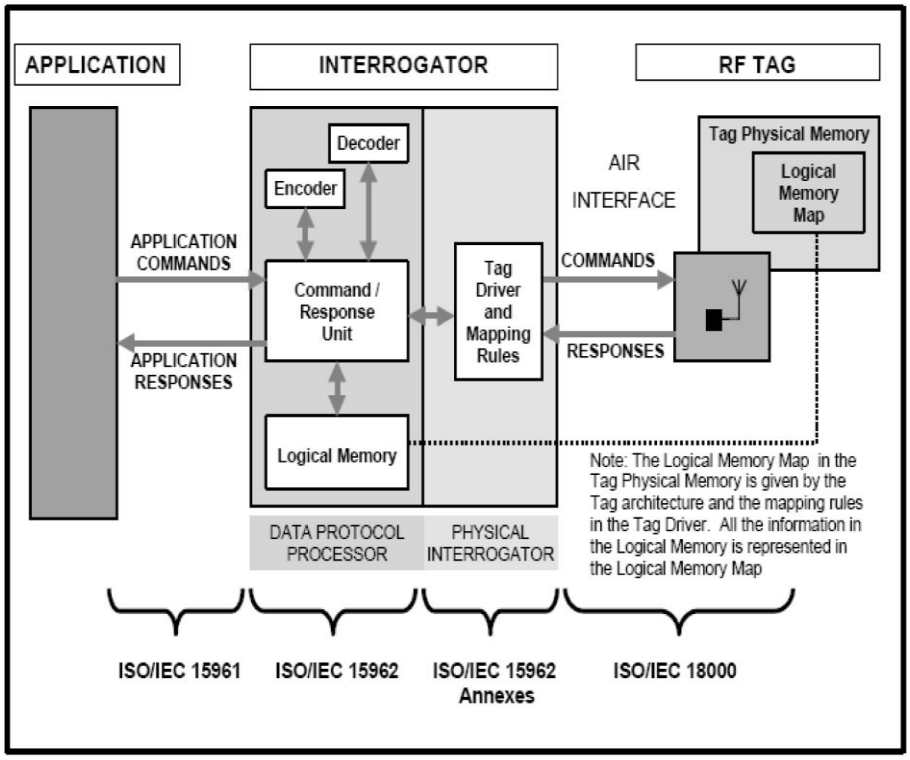


Figure 2.8: The Data Protocol standards of ISO/IEC 15961, ISO/IEC 15962 and the Air Interface standards of ISO/IEC 18000 (Source: Harmon, 2006, p. 46)

As shown in the above (Figure 2.8), the data protocol standards defined by "*ISO/IEC JTC 1/SC 31/WG 4/SG 1*" includes: "*ISO/IEC 15961-1, Information technology – RFID for item management – Data protocol – Part 1: Application interface; ISO/IEC 15961-2, Information technology – RFID for item management – Data protocol – Part 2: Registration of RFID data constructs; ISO/IEC 15961-3, Information technology – RFID for item management – Data protocol – Part 3: RFID data constructs; and ISO/IEC 15962, Information technology – RFID for item management – Data protocol: Data encoding rules and logical memory functions*" (Harmon, 2006, p. 46).

Conversely, ISO/IEC 15962 standard denotes "*how data is encoded, compacted and formatted on the tag and how this data is retrieved from the tag to be meaningful to the application*" (Harmon, 2006, p. 46).

Furthermore, Knospe and Pohl (2004) describe the comprehensive information about RFID standards relating to their operating frequencies. In addition, Knospe and Pohl (2004) also explain RFID applications such as item management, contactless smart card, animal identification and the like (as shown in the Figure 2.9).

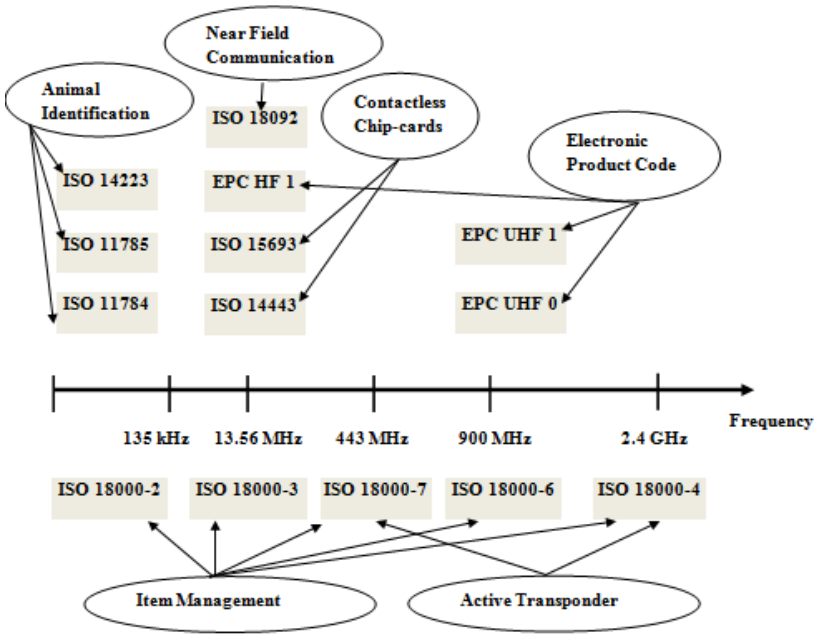


Figure 2.9: RFID technology standards and frequency bands (adapted from: Knospe & Pohl, 2004, p. 43)

2.2 THE INFORMATION SYSTEM

The information system of the RFID architecture in a retail environment will be focusing on where the data on the tag is stored, the way in which the data is captured by the reader and the way in which the data is transferred to the backend database servers or enterprise applications. Hence, the three-layer model of a RFID information system is explained in this literature review section including the tag's memory (logical layer), the data transfer between the tag and reader (material layer), and the data transfer between the reader and the backend (enterprise sub-system layer).

2.2.1 Memory Technologies for RFID Tag (Logical Layer)

Every RFID tag is basically designed with two essential components such as an integrated circuit and antenna, and one optional component which is the memory (Banks et al., 2007). The tag antenna is for communications with the readers. The integrated circuit (IC) contains a microprocessor that "*processes the information coming from the reader and accesses the memory to provide the unique identifier for the tag*" (Banks et al., 2007; p. 8), memory, and a transponder.

Table 2.3: Comparison of Memory Types (adapted from: Finkenzeller, 2003, p. 388; http://www.fujitsu.com/global/services/microelectronics/technical/fram/index.html)

Memory Technology	SRAM	EEPROM	FRAM
Type	Volatile back-up	Non-volatile	Non-volatile
Size	1 byte – 512 Kbyte	16 byte – 32 Kbyte	16 byte – 32 Kbyte
Data transmission	Inductive coupling, Backscattering	Inductive coupling	Inductive coupling
Power supply	Battery	Inductive coupling	Inductive coupling
Typical number of write cycles	Unlimited	$100{,}000 – 1\times10^{6}$	1×10^{10}
Read cycle (ns)	12	200	110
Internal write voltage (V)	3.3	20 (supply voltage 3.3V)	3.3
Write cycle (ns)	12	3	110
Data write	Overwrite	Erase + Write	Overwrite
Data erase	Un-necessary	Byte (64 byte page)	Unnecessary
Typical temperature range	0 ℃ - 60 ℃	-20 ℃ to 120 ℃	-20 ℃ to 120 ℃
Applications	With any number of reprogramming operations. Use in normal industrial temperature range.	Applications with a limited number of reprogramming operations. Applications with expanded temperature range.	Applications with a limited number of reprogramming operations. Applications with expanded temperature range.

However, the memory is used for storing the information sent by the reader and it is where the tag's data is written to and read from. According to Finkenzeller (2003), there are different memory technologies for the tags such as random access memory (*RAM*), electrically erasable programmable RO memory

(*EEPROM*), ferroelectric random access memory (*FRAM*) which is a non-volatile memory that uses ferroelectric film as a capacitor for storing data, and static random access memory (*SRAM*).

In RFID tags, *RAM* is used for the temporary storage of data during the interrogation zone of the readers. The stored data in *RAM* in passive tags will be lost when the power supply is off. Unlike the passive tags, *RAM* in the active tags with on-tag power supply can store the data for a longer period. Similarly, *EEPROM* which uses inductive coupling for data transmission (such as to read data from and write data to the tag) can store the data for several years without a power supply. But, the tags using *SRAM* technology, in which the data transmission method is ether inductive coupling or backscattering, require a constant power to preserve the stored data (Finkenzeller, 2003). The comparison of different memory types for RFID tags can been referred in the Table 2.3.

Roussos and Kostakos (2009) discuss the memory layout of EPC Class 1 Gen 2 as an example of the modern RFID tag's memory (as shown in Figure 2.10 below). The chip's non-volatile memory is partitioned into four areas including reserved, EPC, tag identification (TID) and the user memory bank areas.

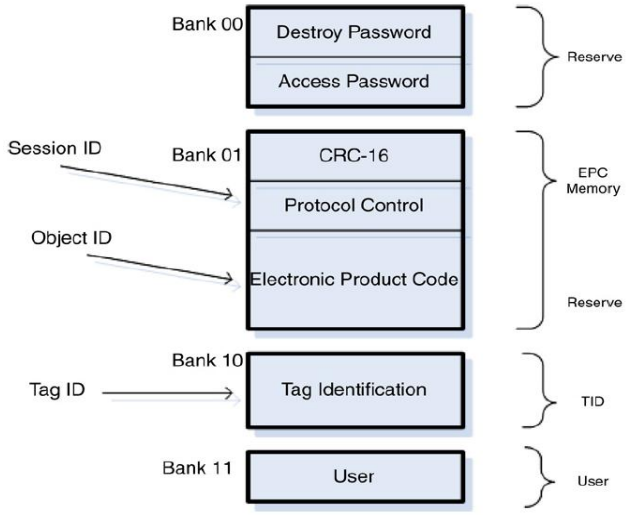


Figure 2.10: Memory layout of EPC Gen 2 Tag (Source: Roussos & Kostakos, 2009, p. 115)

The 64-bit *reserved memory bank* contains an access password to gain access to the tag's content and destroy or kill-password to disable the tag permanently. The *EPC memory bank* is composed of a session identification (SID) or protocol control (PC) which is similar to the media access control (MAC) address and used

by the reader as the tag's address during communication. Electronic Product Code (sometimes referred to as Object Identification) that is assigned to the tagged entity, and the cyclic redundancy code (CRC) which is a checksum algorithm used for detecting transmission errors (Peris-Lopez et al., 2009; Roussos & Kostakos, 2009). The *tag identification (TID) bank* includes the detail information about a unique serial number, manufacturer, and type of tag; while the user bank is optional and used in applications (Roussos & Kotakos, 2009).

2.2.2 Data Transfer between Tag and Reader (Material Layer)

The transfer of data or information one or other between RFID tags and readers in a RFID system is accomplished by transmitting or receiving the energy, in which the data is coded, from an antenna as radio frequency waves.

According to Sanghera et al., (2007), communication through inductive or magnetic *coupling* (near field) and *backscattering* (far field) techniques are mainly used to communicate for conveying data between RFID tags and readers (as shown in Figure 2.11).

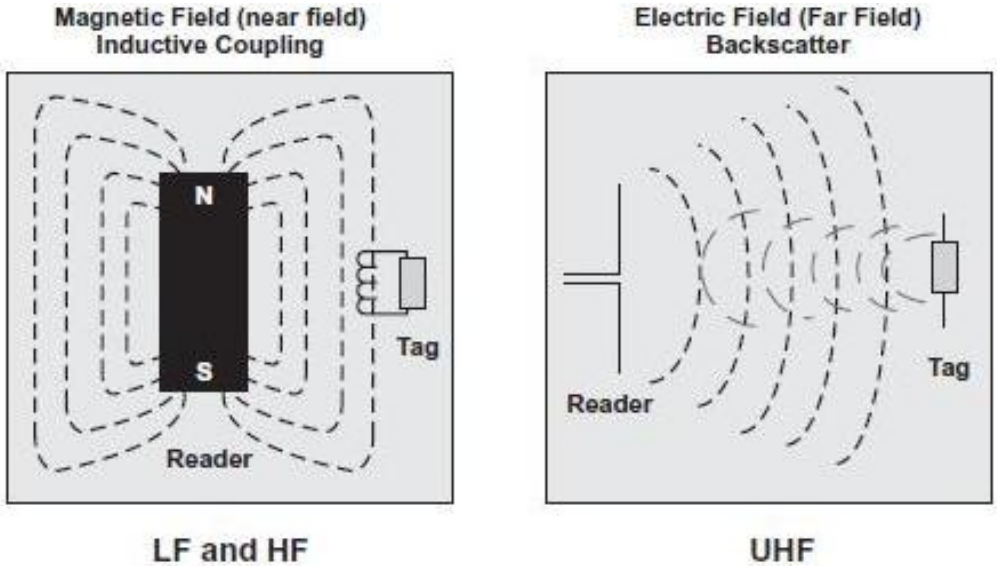


Figure 2.11: Data transfer techniques between RFID Tag and Reader

(Source: LARAN RFID cited in Hunt et al., 2007, p. 14)

2.2.2.1 Near Field or Inductive Coupling Communication

In RFID systems, the near field communication is used by low frequency (LF) or high frequency (HF) RFID systems (Sanghera et al., 2007; Kuo et al., 2007; Hunt et al., 2007).

Near field communication techniques deploy the inductive coupling of the RFID tag to reactive energy flowing around the RFID reader antenna and it is only reliable in the applications such as access control, item level tagging, animal tracking and the like, where the read range or distance is not an issue (Flores et al., 2005; Kuo et al., 2007).

2.2.2.2 Far Field or Backscattering Communication

However, backscattering or far field communication uses *"the process of collecting an inbound signal (energy), changing the signal (the data it carries), and reflecting it back to where it came from"* (Sanghera et al., 2007, p. 34).

Table 2.4: Data transfer techniques between RFID tags and readers (adapted from Kuo et al., 2007, p. 294)

Technique	Operation Frequency	Example Applications
Inductive Coupling (Near Field)	LF: (f < 135 KHz) HF: (13.56 MHz)	Access control, animal tracking, etc., Smart cart, item level tagging/tracking, etc.,
Backscattering (Far Field)	UHF: (443.92 MHz) or (860-930 MHz) Microwave:(2.45 GHz) or (5.8 GHz)	Pallet level tracking, container tracking, item level tracking, etc., Item level tracking, baggage handling, etc.,

Unlike, the near field RFID tag antenna which is mainly a coil (its physical size slightly depends on the radio frequency used); Flores et al., (2005) explains that the far field tag antenna is typically a dipole which is half a

wavelength long and depends on the frequency used (for instance; the higher the frequency, the smaller the far field tag antenna).

The backscattering or far field communication technique is commonly used in the long-range RFID systems that are operated in UHF or microwave frequencies and its applications include pallet level tracking, container tracking, item level tracking and the like (Kuo et al., 2007; Sanghera et al., 2007).

2.2.3 Data Transfer between the Reader and Backend (Enterprise Sub-system Layer)

In order to create the information useful to a RFID supported business process, the *enterprise subsystem* links the RFID readers to computers operating software which *"can store, process and analyze data acquired from RF subsystem transactions"* (Karygiannis et al., 2007, pp. 2-14). For instance, a RFID system deployed in an electronic retail shop has a *RFID subsystem* that is able to read any identifier related to each RFID tagged electronic product. Then, the responsibility of the *enterprise subsystem* is to match the identifier to the electronic product record in a database at the backend server in order to find out its price. Even though some RFID systems may be made up of only an *RF subsystem*, Karygiannis et al., (2007) mentions that most of the systems consist of an *enterprise subsystem* and a *RF subsystem* (see Figure 2.12). Furthermore, there are three main elements in an *enterprise subsystem* that includes "*RFID middleware, analytic systems and network infrastructure*" (Karygiannis et al., 2007, pp. 2-14).

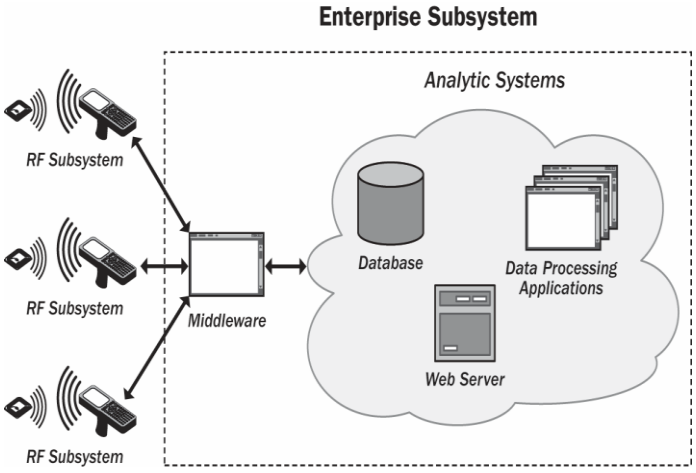


Figure 2.12: Enterprise sub-system in RFID system architecture

(Source: Karygiannis et al., 2007, pp. 2-15)

2.2.3.1 RFID Middleware

The responsibility of a middleware is to prepare the data collected from the RFID readers in the *RF subsystem* for the AS that sustain business processes. In addition, RFID middleware does the filtering of *"duplicate, incomplete, and erroneous information"* receiving from RFID reader (Karygiannis et al., 2007, pp. 2-15). Hence, the RM filtering is useful and important in an environment where many RFID tags are in very close locations. After filtering, the non-duplicated and accurate data will be instantly transported to the AS by RFID middleware. The stored data in AS can easily be retrieved later if it is needed.

To be brief, as shown in the Figure 2.13, the collected and filtered real-time data from the RFID readers is sent to the backend *Enterprise Resource Planning (ERP)* system by the RFID middleware (Bhargava, 2006, cited in Karygiannis et al., 2007).

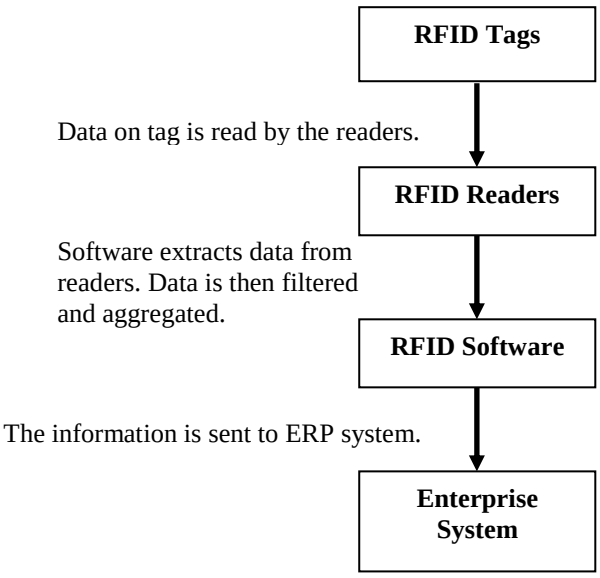


Figure 2.13: Information Functions of RFID Middleware Software (author)

2.2.3.2 Analytic Systems (AS)

AS typically consists of databases, data processing applications and Web servers for processing of the information forwarded by RM according to the instructions of the users and business requirements (Karygiannis et al., 2007).

In the AS of a RFID system, the customized business logic is enclosed in order to support each business process. According to

Karygiannis et al., (2007, pp. 2-15), the supporting customized business logistics perhaps consists of *"rules for automated inventory management, procurement, shipping, receiving and billing"*. Hence, the AS is mostly based on enterprise database applications to support a variety of data coming from multiple sources such as customers, suppliers, logistics service providers and the like as the AS is intended to exist with or replaced the bar code technology (Karygiannis et al., 2007). Thus, the AS deals with the RFID tag data that complies with EPCglobal standards and is sometimes referred to as *EPC Information Services (EPCIS)*.

2.2.3.3 Network Infrastructure (NI)

The communication between the *RF subsystem* and *enterprise subsystem* is facilitated by the network infrastructure which is made up of the communication protocols, logical and physical topology of the RFID network (Karygiannis et al., 2007).

2.3 BUSINESS SYSTEM OF STOCK MANAGEMENT SYSTEM

The Business System (BS) architecture of a RFID stock management system consists of three entities, namely the Stock Item (SI), the Point of Sale (POS) and the Business Information System (BIS). Each entity has sub-systems and services that are required by the other entities. For example the SI requires RFID Tags, a scanner, and services from POS and BIS. The POS requires a Transaction Processing System (TPS), scanners (for cards, Tags and Chips), a Tag attach/detach service and the services of SI and BIS.

Table 2.5: The Business System (author)

Entity	Stock Items (SI)	Point of Sale (POS)	Business Information System (BIS)
Sub-Systems	Tag Scan	Scan TPS	Data Storage IS Management
Services	ID Authentication	De/Attach Authorization	Refresh Audit

The BIS requires software for data base/warehouse and the relevant build for information management. In addition the services of SI are required for audit and POS for updating the relevant stock values. The Business System consequently

presents a closed system of sub-systems, entities and services (Zhang, Li, Wang, Li, & Xia, 2007).

A BS mixes and mingles elements of the real world with informatic abstractions and the people who interact within the system. The Scan sub-systems (often termed 'Readers') illustrate the environment complexity where hardware, software, data, information and humans interact. Hence, each of the instances of interaction elements of the BS creates a nexus of intent that delivers the business value. Similarly non-beneficial interactions may occur that create drag cost and potential legacy costs of unintended disclosures. The risk of disclosure and subsequent compromise of the Business System is eventual in every process (Ohkubo, Suzuki, & Kinoshita, 2005; Rieback, Crispo, & Tanenbaum, 2006).

2.4 RFID SECURITY AND CHALLENGES

The potential for fraudulent intervention into the RFID system is established in the literature (Weis, 2003a; Juels, 2005; Shih et al., 2005; Bhargava, 2006; Bolan, 2007; Ding & Bo, 2008; Haines, 2006a, 2006b; Roberts, 2006; Rieback et al., 2006b; Kim et al., 2007; Rotter, 2008; and so on).

Table 2.6: Threats and their relationship to vulnerabilities in RFID system components (Source: Rotter, 2008, p. 72)

	Tag	Air interface	Reader	Network	Back end
Eavesdropping	•	•		•	
Relay attack		•			
Unauthorized tag reading	•	•	•		
Tag cloning	•	•			
People tracking	•	•			
Replay attack	•	•			
Tag content changes	•				
Malware	•		•		•
RFID system breakdown				•	•
Tag destruction	•				
Blocking		•			
Jamming		•			
Back-end attacks				•	•

Like all other information systems, RFID systems are susceptible to malicious attacks (Rotter, 2008). However, there are many threats that can compromise a RFID BS. In general, the problem with security threats can be related to five different system components of RFID systems (see Table 2.6). For instance, the

attacker can perform eavesdropping (Weis, 2003a; Weis et al., 2003b) to get the information related to a RFID tagged item. Likewise, the RFID system can be cheated (changing the tag data) with a malicious or counterfeit tag after the attacker performing the spoofing (Weis, 2003a; Roberts, 2006; Rieback et al., 2006b; Rotter, 2008; and so on) and cloning (Juels, A. 2005; Masters & Turner, 2007; Ding et al., 2008; El-Said et al., 2009; Abawajy, 2009) of the RFID tag.

Hence, some of the security problems concerned with a RFID BS are briefly described in the following pages (pp. 30-31).

2.4.1 Unauthorized Tag Reading

In an unauthorized tag reading (Rotter, 2008), the tag information can be read by the attacker who uses a fake reader after breaking the tag-reader authentication.

2.4.2 Eavesdropping

It is the way in which the attacker passively and secretly observes the data communication between RFID tag and reader through the air interface (Weis, 2003a; Weis et al., 2003b; Rotter, 2008 and the like). If there is no data encryption method applied in the RFID BS, the eavesdropping attack can easily be performed by the attacker.

2.4.3 Traceability or People Tracking

Rotter (2008) mentions that the attacker can deploy different techniques (for instance; locating illegitimate readers or other eavesdropping equipments/tools near genuine readers) in order to find out the movements of tagged items in the shop or people who carried the RFID tagged items. The RFID threats concerned with traceability are raising the problems related to the location privacy nowadays.

2.4.4 Tag Cloning

RFID tag cloning (Juels, 2005; Masters & Turner, 2007; Ding et al., 2008; El-Said et al., 2009, Abawajy, 2009) is generally a method in which the tag identification is acquired by the attacker. Likewise, Ding et al., (2008, p. 766) describe that tag cloning as *"a matter of determining the signal that the tag transmits and building a device that mimics that signal"*. The procedure of cloning or duplicating the RFID tag can be found in the previous literature (Halamka et al., 2006). Hence,

tag cloning is one of the most serious RFID threats as the cloned tag can easily compromise the backend server of the RFID BS.

2.4.5 Spoofing

RFID tags are vulnerable to a spoofing attack (Weis, 2003a; Roberts, 2006; Rieback et al., 2006b; Bolan, 2009; and so on). A spoofing attack is the way in which the attacker gains profit or compromises the RFID system by using a fake or counterfeit tag. For instance, the researchers from *RSA Security and Johns Hopkins University* used a cloned RFID tag as a legitimate tag in order to buy gasoline (Ding et al., 2008). Thus, the theft of a product by replacing the original tag on an expensive product with a malicious R/W RFID tag (in which cheaper value is written) and counterfeiting the products are two main categories of the RFID spoofing threat.

2.4.6 Denial of Service (DoS)

The DoS threat (Juels et al., 2003; Weis et al., 2003a) is another one of the RFID BS threats, in which the attacker initiates a blocking attack in order to flood the BS with RF energy to interrupt the communication between the tags and readers (Roberts, 2006; Rotter, 2008).

2.4.7 RFID Virus or Malware

The researchers, Rieback et al., (2006b), firstly discussed about a RFID virus or malware in 2006. RFID BS can be totally down if the virus spreads after affecting the backend server that is attacked by a R/W RFID tag injected with a malicious SQL injection code.

2.4.8 Tag Content Changes

The content of a RFID tag can be changed (Rotter, 2008) if the tag is rewritable. In the case where the authentication between the RFID tag and readers is breached, the attacker can simply insert the malicious SQL injection code in the tag data to compromise the RFID BS further.

2.5 RFID STOCK MANAGEMENT SECURITY RISK

As stated above (Section 2.4), trust in RFID based Tagging of stock inventories is unfounded (Rieback, et al., 2006b). The potential for fraudulent intervention into the RFID system is established in the literature (for example: Ding & Bow, 2008; El-Said, & Woodring, 2009). Other authors classify the security risks into layers that represent the different architecture designs a RFID stock management system may have (Haines, 2006a, 2006b; Mitrokotsa et al., 2009). For instance, Garfinkel et al., (2005) classifies RFID security threats into corporate data security and personal privacy threats from the privacy perspectives. Likewise, Karygiannis et al., (2006; cited in Ding et al., 2008) categorizes a RFID risk model into three classes such as *"network-based risks, business process risks, and business intelligence risks"*. Moreover, Ding et al., (2008) proposes the taxonomy model of RFID security threats in three layers such as *physical layer threats* (eavesdropping, jamming and tag cloning), *communication layer threats* (collision), and *application layer threats* (spoofing and virus attacks). However, Mitrokotsa et al., (2010) classifies the comprehensive RFID attacks into layers where the attack could take place (see Figure 2.14).

Consequently, one of the concerns for retail commerce is the progression that as RFID Tags/chips are made more user-friendly then the risk of attack grows. Jones, et al. (2006a) described the evolution of RFID chip design and process time costs. In the manufacture of tags, the access to writing has become easier and less costly in the interest of letting retailers code their own tags. The benefit has however brought with it easier access for criminals to write and rewrite over tags and hence changing the values (including the null value) in a RFID stock management system (Chalasani, & Boppana, 2007). Thus, a RFID BS security risk model is proposed related to the research as shown in the Figure 2.15.

The proposed security risk model of RFID enabled stock management system of a BS comprises three simple layers such as SI entity security risk, POS entity and BIS security risks.

The risks related to SI entity includes replay attack, tracking, eavesdropping and the like, while POS entity security risk consists of unauthorized tag reading, theft or incorrect handling of R/W Tags at the POS,

intentionally or unintentionally attachment and detachment of Tags to SI (for instance, intentional attachment of tag with a cheaper value to an expensive SI).

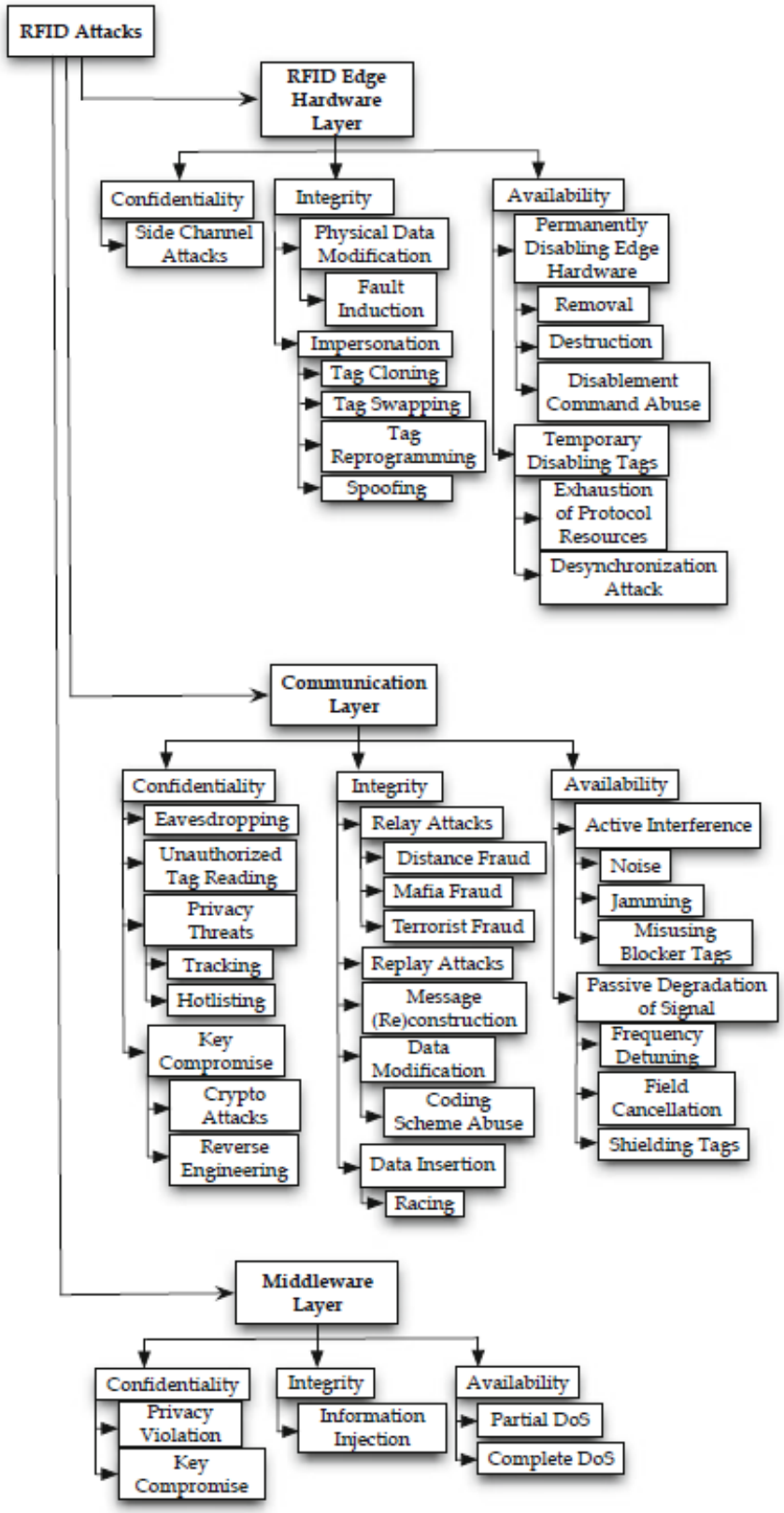


Figure 2.14: Classification of RFID Threats (Mitrokotsa et al., 2010, p. 41)

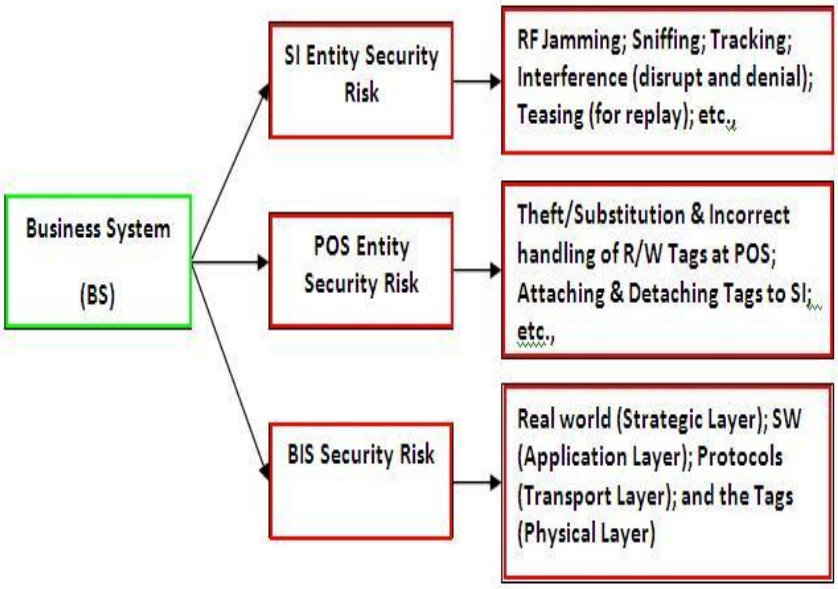


Figure 2.15: RFID Business System Security Risks (author)

However, the Business Information System (BIS) consists of the database/warehouse application, the human participants, the Information System (IS) build, and the Information Technology (IT) that supports the IS. As a sub-system of the BIS, middleware is found on the scanners and in the POS entity. The explanation given by other researchers (Bhargava, 2006) suggests that the division of the BIS is categorized into layers such as the real world (strategic layer), the software (application layer), the protocols (the transport layer) and the Tags (the physical layer). Such a categorization maps onto the archive of known security breaches in the Internet world and provides structure for security risk assessment of BIS entity.

2.6 CONCLUSION

In this chapter, the three-tier model of a RFID system such as material layer (Section 2.1), IS layer (Section 2.2) and business process layer (Section 2.3) were reviewed in order to understand and identify the processes, model, and concepts that could be applied to the proposed research. As stated in the material layer, the characteristics of different classes of RFID Tags, readers, host or controller, RFID standards were reviewed. Likewise, the review of the IS layer explained where the data was stored and the way in which the tag data was transferred to the backend servers or enterprise applications. Moreover, the BS of a RFID stock management

system was explained by using three entities such as SI, POS and BIS as business process layers.

Like any other IS, RFID enabled systems are susceptible to malicious attacks. Thus, RFID security threats and challenges (Section 2.4) were reviewed. However, most of the previous literature reported the security risk concerned with RFID systems. Consequently, it was exposed that the requirement of forensic readiness might be necessary as there were potential for fraudulent uses of RFID networks in commercial settings. Hence, the RFID BS security risks model (Figure 2.15 in Section 2.5) was proposed based on the requirement of the research after reviewing the previous literature.

In order to perform the investigation on the presence of digital evidence after the theft of a SI, a prototype of a commercial retail environment using a RFID stock management system will be constructed in the laboratory. The stabilized system will be compromised with SQL poisoning attack through a RFID tag in order to change the value of the SI and then each entity in the BS (Table 2.5 in Section 2.3) will be investigated for evidence of theft. The collected evidence will be preserved, analysed and presented to validate the case of SI theft. Therefore, in the following chapter (Chapter 3); the research methodology will be discussed in detail including the review of similar published work, the research design, data collection, data analysis, expected outcomes, and limitation of the research.

Chapter 3 -

Research Methodology

3.0 INTRODUCTION

Chapter 2 reviewed the literature relating to a RFID system, the information system of RFID architecture, the business system (BS), security challenges, and the proposed RFID BS security risks. This chapter will derive a research methodology and identify the main research question, sub questions and hypotheses. The specification for data collection, processing and analysis will also be given.

It is proposed that the RFID security risks identified in Chapter 2 will be exploited to crack a trial business system in laboratory conditions. The system is to be stressed by running transaction scenarios. Hence, the whole system (chip, network, & database) can be searched for sources of digital evidence (stored data). Finally, the process steps are to be analyzed and written up as a guide for best practice for the digital forensic investigators.

The research methodology is to be based on descriptive methods for using case scenarios. In Section 3.1 previous similar studies are reviewed to ground the method in others published papers. Hence, the methodology chapter is firstly organized into the review of similar published works (Section 3.1); the research design, research question and hypotheses (Section 3.2); and the data requirements such as how data is to be collected, processed, analyzed and presented (Section 3.3). Then, the limitations and expected outcomes of the research (Section 3.4) will be followed by the conclusion (Section 3.5).

3.1 REVIEW OF SIMILAR PUBLISHED WORKS

In order to develop the methodology, the reviews of similar works have been identified and analysed. The analysis of these studies had lead to the methodology for this research project. The previous similar published works have guided the ways in

which the implementation of the methodology adopted as described in this methodology chapter. Section 3.1.1 discusses the study conducted by Cerrudo (2003), in which the attacker's manipulations of data in the databases and other applications on the network were by using SQL injection attack. Section 3.1.2 confers the warning to the middleware designers that the RFID tag data could be used to exploit back-end database servers and software systems (Rieback et al., 2006b). Sections 3.1.3 (Fowler, 2009) and 3.1.4 (Fowler, 2007) contain the reviews of SQL Server forensics and forensic analysis of a database SQL Server 2005, respectively. Then, Section 3.1.5 (Wang et al., 2009) discuses a model for physical memory analysis of a computer by using live forensic methods. Afterwards, research conducted by Ceasar and Shaul (2009) is reviewed in Section 3.1.6 followed by the discussion of "*digital forensics investigation framework that incorporates legal issues* (Ieong, 2006)" in Section 3.1.7.

3.1.1 Manipulating Microsoft SQL Server Using SQL Injection

The study conducted by Cerrudo (2003) did focus on the techniques of retrieving the content of the database by using a SQL injection attack on the backend Microsoft SQL Server behind the firewall. The purpose of the research was to inform security professionals about the potential SQL injection attack which could have a destructive effect on an organization. However, Cerrudo (2003) did mention that the SQL injection attack was not only a problem for Microsoft SQL Servers, but also a problem for other database vendors such as Oracle or IBM DB2.

Even though web applications were more and more secure due to the awareness of SQL injection attacks, a single mistake could compromise the entire system in large and complex applications. Cerrudo (2003) stated that the methods of execution of SQL injection code on the server and retrieving the results from the server were required in order to exploit an application. These methods are possible due to the *"OPENROWSET"* and *"OPENDATASOURCE"* functions of a SQL server which are used to open an Object Linking and Embedding, Database (OLEDB). The functions of *"OPENROWSET"* and *"OPENDATASOURCE"* are also commonly used to push data to a remote SQL Server and pull data into a manipulated SQL Server. Hence, the researcher gave examples of SQL injection statements on how to

retrieve data to an external data source, how to redirect remote data sources to any server, how to load logins and password hashes if given the appropriate privileges. By acquiring the password hashes, the brute-force attacks on the passwords could be performed.

According to Cerrudo (2003), there are many techniques that the attacker can use to find a way of avoiding the firewall even if the outbound connections of the SQL Server are blocked. Furthermore, the attacker could gain full administrative privileges by escalating an exposed non-privileged login. After gaining the adequate privileges, the attacker could upload a binary file such as an executable password dump file (*pwdump.exe*) by using either SQL statements on the victim SQL Server or by writing Visual Basic Script or Java Script files to execute the binary file. The researcher explained that the malicious person could access the internal network by using linked and remote servers, which allowed the attacker to execute distributed queries. The attacker could execute stored procedures, even if the linked and remote servers were not configured to run random queries to access data, to move around from one database server to another in order to go deeper inside the network. Then, the attacker could upload files into the servers after gaining enough privilege access to the remote or linked server.

Finally, Cerrudo (2003) concluded the paper by presenting his recommendations to prevent SQL injection. For instance, disallowing ad hoc queries through OLEDB from the SQL Server, updating the latest security fixes promptly, configuring firewall filters in order to block all unnecessary outbound traffic and the like, could help to prevent the attacker's manipulations of data in the databases and other applications on the network.

3.1.2 Is Your Cat Infected with a Computer Virus?

This paper written by Rieback et al., (2006b) was presented at the *"Fourth Annual IEEE International Conference on Pervasive Computing and Communications"*. The purpose of the research was to warn the middleware designers that RFID tag data could be used to exploit back-end database servers and software systems.

Rieback et al., (2006b) presented the first self-replicating RFID virus. The researchers used the tag as an attack vector and compromised the backend RFID middleware systems by the use of a SQL injection. A number of weaknesses in RFID systems that malware could exploit were also discussed. The exploits such as buffer overflows and SQL code insertion attacks could be performed by the use of even less than 1Kbits of the data on the RFID tags. The researchers did also mention about RFID worms and viruses, and how these viruses could affect the real world scenarios.

Furthermore, Rieback et al., (2006b) had demonstrated Oracle-specific viral functionality. The researchers used a Windows machine on which Oracle 10g backend database along with a Philips I.Code/Mifare RFID reader and I.Code SLI tags were installed. For an application scenario, the supermarket distribution centre scenario was chosen as a target. The implementation of the testing architecture used by the researchers is as shown in the Figure 3.1 below.

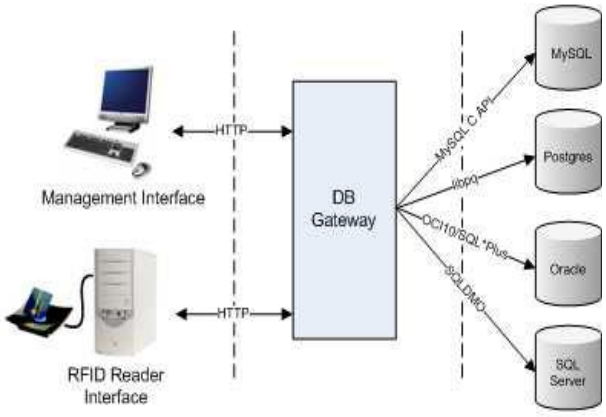


Figure 3.1: RFID Malware Test Platform (Rieback et al., 2006, p. 175)

In their experiment, the virus (Oracle/Server-Side Include virus), which was a self-replication virus with malicious payload leveraging with SQL and script injection attacks, was only 127 characters and written on an I.Cod SLI tag. When the tag was read by the RFID reader, the injection attack queries were sent to the Oracle database through the Oracle Call Interface (OCI) library and exploited the backend database. Hence, the contents of the database were changed by the virus.

To sum up, the researchers (Rieback et al., 2006b) invented a RFID virus that replicates itself and it only needs an initiator tag as an attack vector to infect the whole system. In fact, the tag injected with a malicious SQL injection code will

firstly affect the back-end RFID middleware system and then the virus will later be spreading within the corporate RFID system.

Thus, the writers or designers of the RFID middleware must carefully design and build the middleware by performing *appropriate checks* (*bounds checking*, *special character filtering* and the like) in order to avoid *"RFID middleware from suffering all of the well-known vulnerabilities experienced by the Internet"* (Rieback et al., 2006b, p. 169).

3.1.3 SQL Server Forensics

In this book chapter, Fowler (2009) discuses what SQL Server forensics is, and how to investigate the incidents relating to SQL Server data. The author also mentions the differences between the traditional forensics and SQL Server forensics. Furthermore, the method used in SQL Server forensics is used to identify whether the security of a database is breached or supplementary investigation is needed to confirm a breach.

Firstly, Fowler (2009) explains that the digital forensics field is very new and it started getting industry attention in 2001. Since 2001 until now, the digital forensics field has rapidly been growing due to groups such as the Digital Forensic Research Workshop (DFRWS), and a number of conferences such as the Black Hat USA conferences and the like. Then, he states that the problems of security breaches of data are not an uncommon theme in current news. As a result of companies trying to cut costs by consolidating databases onto a smaller amount of database servers, it leads single sources of confidential information to become the primary targets for malicious attackers. Hence, *"75% of cyber attacks are application based and many involve the theft of personal or financial information stored within a database, according to the Gartner Group"* (Fowler, 2009, p. 48). He gives the examples of recent data breaches such as the *CardSystems* which involved the theft of 200,000 credit card details and *TJ Maxx* which entails the leak of 45.7 million credit and debit account details. Moreover, in order to help the protection of sensitive data, Fowler (2009) points to the legislation and regulations dealing with data breach issues. For instance, the *Payment Card Industry (PCI)* necessitates organizations to employ control actions towards sensitive data and *Senate Bill 1386 (SB – 1386, commonly known as "The California Security Breach Information Act")* forces companies to

have to notify their customers affected by a breach of data security (Fowler, 2009, p. 48).

Secondly, Fowler (2009) mentions a forensic investigator can use SQL Server forensic techniques to investigate data security breaches, in order to confirm whether the digital invasion has taken place. Hence, when a digital invasion has occurred, the process of carrying out SQL Server forensics is able to help a forensic investigator to determine whether the breach of data is affecting the data protected by legislation and perhaps prohibiting the organizations from erroneously unveiling the protected data due to an unauthorized digital intrusion. In fact, SQL Server forensics which focuses precisely on the *"identification, preservation, and analysis of the database data suitable for presentation in a court of law"* facilitates a forensic investigator to succeed, determine, and investigate the digital intrusions relating to SQL Server data (Fowler, 2009, p. 49). Furthermore, the author indicates the initiation of SQL Server forensics during a data security breach can accomplish the objectives of proving or disproving the event and scope of database intrusion, identifying the data transactions before and after an intrusion event, retracing *Data Manipulation Language (DML)* and *Data Definition Language (DDL)* operations, and recovering deleted database information. Even though, the database logging or DML/DDL tracing applications can give the results comparable to that of SQL Server forensics, the utilization of those applications can cost the organizations money, storage requirements and the like. Thus, most of the organizations in the industry rarely invest in database logging applications. As a result, the author mentions that the forensic analysis of a SQL Server is a suitable method for extraction of *"the server data from published and unpublished data stores, both native within SQL Server and scattered throughout the operating system"* (Fowler, 2009, p. 49).

Thirdly, the author explains the differences between traditional forensics and SQL Server forensics by using two fictitious case scenarios (see Figure 3.2 and Figure 3.3).

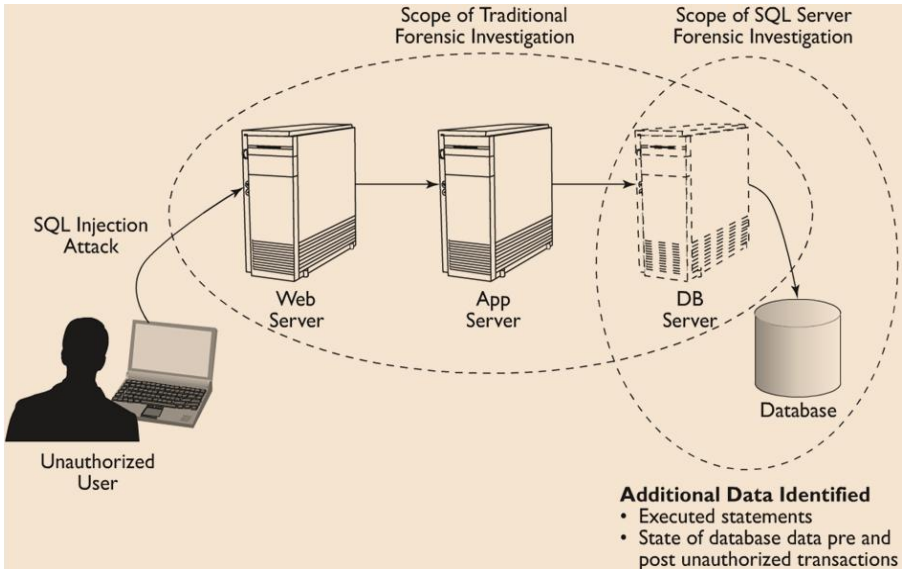


Figure 3.2: Scenario 1 illustration (Fowler, 2009, p. 52)

In the first scenario, the incident response team was notified by a system administrator. In fact, a system administrator discovered attempted SQL injection attacks after analyzing a number of peculiar entries found in the log file of the Web Server (Fowler, 2009). Then, a forensic investigation was initiated on the *three-tier Web application* by acquiring the system memory, event log and Web Server log files. In addition, the investigator not only examined the system files, but also checked *"the registry of each system for any file or registry during the time frame of the attacks"* as the extent of the attempted intrusion was unidentified (Fowler, 2009, p. 51). During the analysis of the collected artifacts, the investigator found part of the traces of SQL injection attack code in the system memory from the Web and Application Servers, which are mapped to the Web log files. Hence, the forensic investigator presumed that the SQL injection attack was launched to target the SQL Server by exploiting a Web-based application by an unauthorized user. However, the investigator could not yet identify whether the attack was successful, and which SQL Server data was compromised or disclosed even if the attack was successful. Thus, the investigator can relate *"the attempted SQL injection attacks identified within Web server logs and memory dumps against actual statements executed within SQL Server to verify whether the attack code was corrected, tunneled and executed by the database"* (Fowler, 2009, p. 51). Furthermore, the investigator can repeat a Transact-

SQL statement from the SQL injection attack code in order to verify which specific Server data was compromised. To conclude, the forensic investigator could be able to verify that any SQL Server data was undisclosed by utilizing *SQL Server forensics*.

In the second case scenario, Fowler (2009) gives an example of a malicious employee who made unauthorized changes in a company's database (500 gigabytes capacity). The changes are considered to be done within the online sales database, especially on *"the billing amounts of the customer's order"* and hence *"the traditional forensic investigation is performed"* (Fowler, 2009, p. 52). After investigation, the forensic examiner concludes that the unhappy employee logs on to the database server after office hours for a few nights and the transactions done during that time seems to be abnormal. But, the investigator cannot verify whether the modification of data by a suspicious employee is authorized and what the extent of the incident is. Thus, Fowler (2009) mentions that the forensic investigator who initiates SQL Server forensics can verify that a breach of security incident has happend, prove the malicious activity causing database changes without authorized permission, and confirm the user who carried out those changes. However, the investigator has to apply *"the data reduction principles to reduce the amount of data to be acquired and analysed"* due to the large capacity of the database (Fowler, 2009, p. 52).

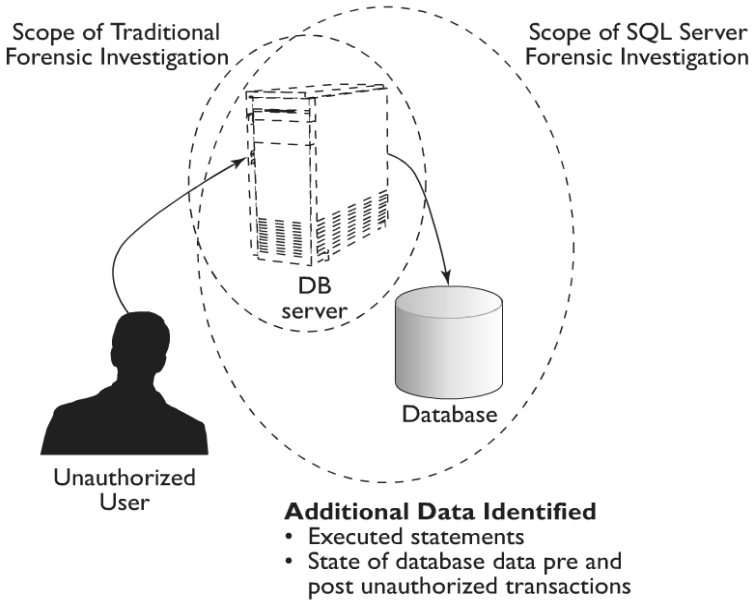


Figure 3.3: Scenario 2 illustration (Fowler, 2009, p. 53)

Unlike SQL Server forensics, the traditional Windows digital forensic investigation generally targets on "*the volatile and non-volatile operating system, and selected application data"*, and often ignores the database (Fowler, 2009, p. 50). As the database is overlooked, the decision on whether a database is compromised or not during an incident is very hard for a forensic investigator. Nevertheless, the two case scenarios explained in this book chapter demonstrate that the application of SQL Server forensics can discover additional important data beyond the scope of a traditional Windows forensic investigation (Fowler, 2009).

Fourthly, the author explains three different acquisition methods that can be performed during forensic investigations such as live acquisition, dead acquisition, and hybrid acquisition. All SQL Server data, volatile and non-volatile, can be identified and acquired by using the live acquisition method. On the other hand, the live acquisition method can solve the problem of taking a SQL Server offline for an investigation as the business companies may not allow the forensic investigator to take the server offline (Fowler, 2009). Furthermore, the author points out some benefits of live acquisition. For instance, the forensic investigator can perform *"investigation on identified data repositories without needing to forensically duplicate the entire logical or physical disk drives"* as live forensics is more and more realistic due to the ever growing storage capacity of the computers (Fowler, 2009, p. 54). In fact, the investigator can save time and easily focus on the database artifacts that are mainly related to the investigation by using the live acquisition method. Then, the author mentions two connection methods, interactive and remote connections, to a SQL Server to perform live acquisition. In an interactive connection, the investigator can either access the SQL server onsite or use a remote desktop protocol (RDP) to log onto the SQL Server and utilize the tools from an incident response-compact disc (IR-CD) to acquire SQL artifacts. Then, the acquired output can be transferred to a reliable storage location by using tools such as *Netcat* or *Cryptcat*. Instead, a universal serial bus (USB) flash or external hard drive can be connected to the target SQL Server to store the acquired output. Unlike an interaction connection, Fowler (2009) mentions the investigator can employ a computer that can be connected to a compromised network in order to remotely connect a target live

server and use IR tools to collect the artifacts. But, these connections depend on the *"system and database libraries, and dynamic management objects (DMOs) on the SQL Server"* (Fowler, 2009, p. 57).

The dead acquisition method is the way in which the power connection of the SQL Server is forcefully shutdown. However, the potential evidence from the volatile data will be lost when the server's power is down even though the analysis results can easily be repeated for later verification. But, the actual SQL data alterations have normally happened in the memory and are recorded in the database transaction log. Likewise, on-disk data pages are only updated *"during the start-up and shutdown of the SQL Server service",* and when *"the regularly scheduled checkpoint process runs which then flushes the modified data pages to disk"* (Fowler, 2009, pp. 58-59). Thus, the author does not recommend dead acquisition as the SQL Server utilizes as much system memory as possible.

In hybrid acquisition, the live acquisition of volatile data from a target system is initially performed and then followed by dead acquisition. Hence, the hybrid acquisition allows the forensic investigator to manage the relative proportions of live and dead acquisitions according to his or her needs (Fowler, 2009).

Finally, the author explains a methodology of SQL Server forensics (see Figure 3.4).

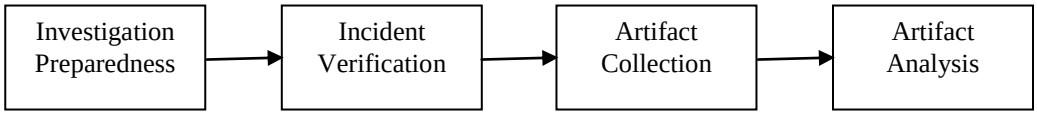


Figure 3.4: SQL Server forensic methodology (Fowler, 2009, p. 60)

Despite the fact that there are many different forensic methodologies which can be found in literature, every phase of any methodology must be performed correctly by the forensic investigator. Otherwise, the results of the investigation as evidence may not be possibly accepted by the court (Fowler, 2009). For instance, the integrity of the digital evidence must be maintained by producing digital hashes on all the acquired artifacts during the investigation.

For *Investigation Preparedness* (IP), the first phase of the SQL Server forensic methodology, the author mentions that the IR toolkit must be created, a

forensic workstation must be prepared, and previously-written SQL IR scripts must be collected before the actual investigation is initiated (Fowler, 2009). As a result, the forensic investigator can lessen the response time and improve the probability of achieving success in an investigation.

The second phase of the SQL Server forensic methodology is *Incident Verification* (IV). During IV phase, an incident can be verified by either a system administrator or a third party prior to carrying out a comprehensive forensic investigation on a SQL Server. However, Fowler (2009) states that the forensic investigator can skip the IV phase depending upon the circumstances of the case.

The third phase of the SQL Server forensic methodology is *Artifact Collection* (AC) involving the acquisition and preservation of artifacts. During the AC phase, it is significant that any acquisition process performed must be followed in a forensically sound manner and the collected data must also be preserved by using Message-Digest-5 (MD5) or Secure-Hash-Algorithm-1 (SHA-1) cryptographic hashing functions to maintain the integrity of the collected data (Fowler, 2009). Otherwise, the evidence may not be acceptable to the court of law. The final phase of the SQL Server forensic methodology is Artifact Analysis (AA) of the collected data. In the AA phase, the author mentions that important events (for instance; failed and successful login attempts, and irregular database activities) must be identified and included in the timeline of investigation (Fowler, 2009).

The significant sections in this book chapter are how to investigate the SQL Server related incidents in a forensically sound manner, and what phases of forensic methodology can be used in SQL Server forensics to identify whether the security of a database has been breached or whether a supplementary investigation is needed to confirm a breach.

3.1.4 A Real World Scenario of a SQL Server 2005 Database Forensic Investigation

Fowler (2007) had written the paper by using a real world security incident scenario, in which a company's backend database server was compromised by an unauthorized person resulting in wrong product shipments and financial loss.

In the verification stage, a live analysis was carried out in order to acquire volatile and non-volatile data. Fowler (2007) used a customized configuration file with a Windows Forensic Tool (WFT Version 1.0.0.3) to begin the incident verification. At exactly 10:02AM of server time, Fowler (2007) logged into *PRODSQL05* SQL Server by using an administrator account. Then, at 10:03AM, he started the command of the console in order to begin the investigation.

The trusted command shell was firstly launched by using "*D:\FResponse\cmd.exe"* from the inserted *Forensic Response CD* to avoid the corruption or tampering of data on the target system. Likewise, the outputs during the investigation were saved on a trusted forensic workstation. To ensure the integrity of the data, the acquisition file was hidden and protected with a password. Then, the *"D:\FResponse\wft.exe –dst E:\"* command was used to gather volatile database and operating system data from the target system and to store them on the forensic workstation. As a SQL Server reserves #50 sessions for internal server processes, Session #51 (belonging to the unknown user login as *EASYACCESS*) and Session #52 (the instance of WFT executing under local administrator account) were identified as currently active on the SQL Server. Fowler (2007) had mentioned that observing the history of connected users could lead to the suspicious transaction, as the server always maintains the record of the last SQL statement executed in a given session. The review of the error log, which is stored in "c:\ProgramFiles\Microsoft SQL Server\MSSQL.1\MSSQL\LOG", identified several failed login attempts against the "**sa**" account followed by the successful login. Hence, the activity in the error log was evidence of a successful brute force attack against the SQL server.

To further the investigation, Fowler (2007) used a SQLCMD utility (from the response CD) that allowed the submission of ad-hoc SQL scripts to a Microsoft SQL Server. In order to open a connection to the server, the *"D:\FResponse\Sqlcmd –S PRODSQL05 –e –s"* command was launched. The "-e" and "-s" switches were used to echo input SQL statements into the result files and to ensure the outputs were comma delimited accordingly. Once the server was connected, the output files, including the initial connection to the server were sent to the trusted location and MD5 hashes were created on all the output files for integrity of the collected data.

Then, the researcher verified the SQL Server configuration settings in order to check the login authentication mode (either mixed or Windows mode) and importantly the server audit level such as *"successful and failed login attempts"* were logged (Fowler, 2007). Afterwards, a query was issued to find out whether the *EASYACCESS* account had permission to access the OnlineSales (target) database as two stages of authorization were needed to access the SQL Server instance and various databases (Fowler, 2007). Furthermore, the configuration of Microsoft extended procedure *"xp_cmdshell"* was verified since it should be disabled in a Microsoft SQL Server 2005 by default. In fact, database users could *"execute Disk Operating System (DOS) level command within the underlying host Operating System (OS) using their SQL clients"* if the *"xp_cmdshell"* was enabled (Fowler, 2007, p. 9). The target system information was collected not only during the verification stage, but also from the client.

After presenting the initial findings at 10:45AM, the investigator was granted permission to perform a comprehensive forensic investigation (at 11:01AM) by the client who confirmed that *EASYACCESS* was a glitch (Fowler, 2007). Then the server was isolated from the production network in order to prevent additional changes by *EASYACCESS*.

As part of the evidence acquisition, the author gave the priority to all the data sources by using the formula, *"[10 – (significance rating) + (volatility rating) = priority]"*, according to the volatility and significance to reinforce the investigation (Fowler, 2007, p. 12). By giving priority to the relevant data (including SQL Server connections, sessions, transaction logs, system event logs, SQL Server logs and database files) that has to be acquired, the data acquisition cost can be minimized as the databases may store huge amounts of data. Fowler (2007) captured server connection and session data by using the customized WFT tool during the verification stage, whereas the OnlineSales database file information was acquired by using the query with the trusted SQLCMD before the transaction log files were acquired. Then, the transaction log that recorded all the statements within the database was saved into the trusted location. However, the researcher (Fowler, 2007, pp. 13-14) mentioned that the logical allocation status of the physical transaction log was also necessary to

be acquired by using the command *"dbcc loginf"*. In fact, each physical log file is split into *4-16 Virtual Log Files (VLFs)* according to Delaney (2007, cited in Fowler, 2007, p. 13) and the critical relevant data may be located in those reusable VLFs.

Then, the SQL Server was shutdown in order to acquire a bit-to-bit copy of the physical log files and the *OnlineSale* database file by using one of the disk imaging tools (dcfldd tool) that gave MD5 hash values of acquired file during acquisition. In addition, Fowler (2007) restarted the server services after collecting the server trace files and error logs by using *dcfldd tool* as the potential evidence could also be found in those logs.

In order to establish the scope of investigation, the researcher constructed timeline by reviewing server error logs, failed login attempts and associated server process identifiers (SPIDs), and the trace files. Afterwards, the researcher (Fowler, 2007, p. 19) acquired the different data types (either Unicode or non-Unicode; or both) used by the SQL Server as the non-Unicode data could be lost if it was examined by a forensic workstation *"using a code page which did not cover a range of characters used within the collation setting of the database"* (Microsoft, 2007). Similarly, the transaction log information was observed after the log file was imported to Microsoft Excel on the forensic machine using the same code page acquired from the target server. However, only target columns relevant to the investigation were examined as there were more than 100 columns in a SQL Server 2005 transaction log (Fowler, 2007).

Furthermore, the researcher determined the raw data pages relating to the transaction IDs after attaching the acquired OnlineSale database within SQL Server Management Studio (SSMS) on the forensic workstation. Subsequently, the data type within previously acquired row offset was verified by using the table schema that was also acquired before examining raw data pages. Likewise, the value conversions of hexadecimal to decimal from *Rowlog0* (on disk value prior to transaction) and *Rowlog1* (committed transaction value) of the transaction log were performed. Then, the transaction updates were determined according to the affected data pages and slot IDs. Hence, Fowler (2007, p. 30) mentioned "*the row offset and page ID values obtained from the transaction log were used to identify the specific value updated*

within the affected records". In addition, the data type within row offset could be verified by the previously acquired table schema and the use of string searches on the inactive parts of the transaction log can give the forensic investigator information *"to identify the rows for reconstruction"* (Fowler, 2007, p. 41).

To conclude, the researcher reported that an unauthorized user from Internet Protocol (IP) address *192.168.1.20* and gained access to the *PRODSQL05 server* by conducting a brute force attack. As soon as the attacker got access to the database, *"a backdoor account named EASYACCESS"* was created by a connection *"using the Microsoft OSQL client"* (Fowler, p. 42). Then, the attacker performed 7 SQL statements (including 5 updates, 1 insert, and 1 delete statements associated with *SPID 51*) in order to get *"an XBOX 360 with the incorrect price of $4.00 that was billed to the Visa card of an existing customer within the database"* and to order "*Volcano 62 inch Plasma TV VC2332 with a price of $3.50 instead of $3,500.00"* with the mailing address totally different from that of the compromised customer (Fowler, pp. 17-42).

3.1.5 A Model of Computer Live Forensics Based on Physical Memory Analysis

The article written by Wang et al., (2009) proposes a model for physical memory analysis of a computer by using live forensic methods due to the deficiency of current methods in live forensics. Live memory forensics is the way in which the potential evidence from the physical memory of running a workstation or server is collected. The live forensic method is very significant for forensic investigators, as possible volatile evidence such as *"running processes, logged-in users, current network connections, users' sessions, drivers, open files, etc.,"* can be lost if the target computer or server is down (Wang et al., 2009, p. 4647).

The researchers mention that the integrity and the chain of custody of the collected evidence are difficult to maintain due to the acquisition and analysis of potential evidence data are normally carried out simultaneously in current live forensic approaches. Thus, Wang et al., (2009, p. 4647) proposes *"a model computer live forensics based on recent achievements of analysis techniques of physical memory image"* to overcome the shortcomings of the traditional live forensic methods.

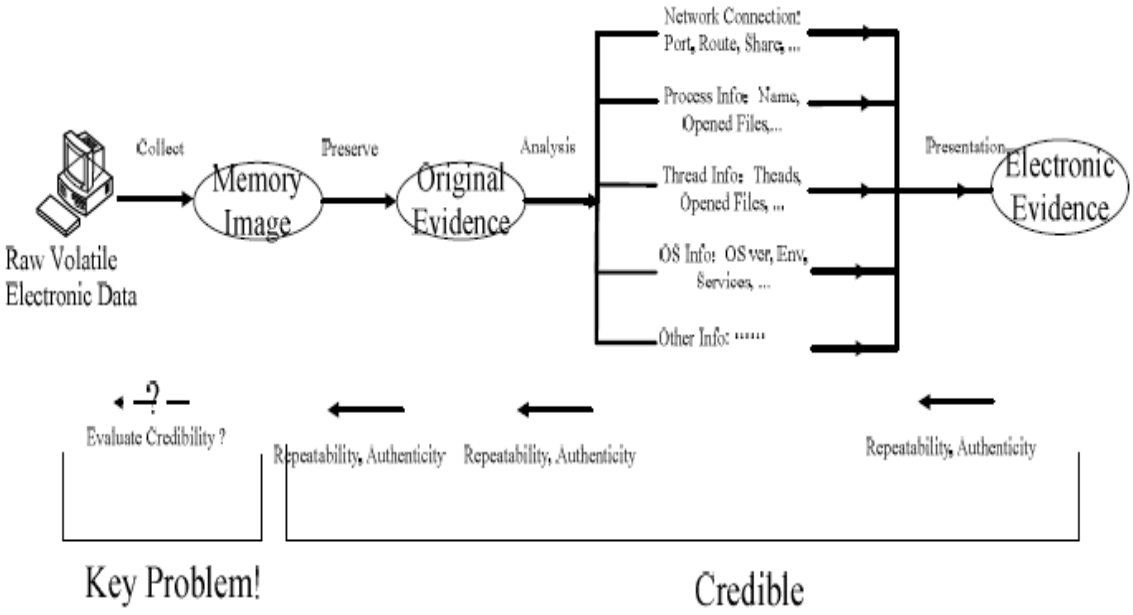


Figure 3.5: Model of Computer Live Forensics Based on Physical Memory Analysis (Wang et al., 2009, p. 4648)

Unlike the traditional live forensic methods, the forensic procedures in their proposed model (see Figure 3.5) can be performed and evaluated individually.

The researchers also describe current achievements in memory analysis, by referring to the previous literature (Wang et al., 2009, p. 4648), which makes their proposed model possible. Subsequently, the benefits of the proposed model such as the evidence credibility can be calculated, evidence can be validated, and the tool impact on the target system can be minimized are briefly explained.

Eventually, Wang et al., (2009, pp. 4648-4649) discuss the issues concerned with the credibility of live forensics such as evidence integrity, repeatability, the extent to which the raw evidence from the physical memory will be affected by running the tool on the target system, and so forth.

3.1.6 Review of SQL Server Anti-Forensics: Techniques and Countermeasures

In this article, Cerrudo (2009) has stated that the SQL server logs can be found in the locations such as the SQL error log, Windows application log, default trace, and transaction log by default. These logs are mostly located in data files, either flat file or database, stored on the hard disk. Moreover, the footprints of the recent attacker

can also be found not only in the database's cache and procedure cache, but also can be traced in the memory. Referring to the Microsoft SQL Server Book online,

> *"SQL Server logs certain system events and user-defined events to the SQL Server error log and the Microsoft Windows application log. Both logs automatically timestamp all recorded events. Use the information in the SQL Server error log to troubleshoot problems related to SQL Server.*
>
> *The Windows application log provides an overall picture of events that occur on the Windows operating system, as well as events in the SQL Server and SQL Server Agent...."* (http://msdn.microsoft.com/en-us/library/ms191202.aspx; 2010, p. 1)

The researchers mention that the valuable information to the forensic investigator can be found in the SQL Server error logs and the Windows application logs, although the duplication of log data exist in those two locations. However, the log in the SQL Server can easily be erased by the users who have got the administrative rights or privileges. Even though there are events such as failed login attempts, some database console commands (DBCC) captured by the SQL Server error and Windows application logs, more evidence is required for the forensic investigation that has to be found somewhere in order to identify the structure of the database, the changes that are made to the data and the like. In addition to error and application logs, the SQL default trace and transaction logs are critical components of finding evidence in a potential data breach, as data modifications made by every transaction are recorded with timestamps. But Cerrudo (2009) also states that the extended stored procedure executions, SELECT and DBCC statements cannot be found in the transaction log. But in some cases, the critical evidence can also be found in the contents of the SQL server memory (data and procedure caches).

Finally, Cerrudo (2009) discusses *"SQL server anti-forensic techniques"*, the protection of an audit trail and evidence when the system is compromised.

3.1.7 FORZA – Digital Forensics Investigation Framework that Incorporate Legal Issues

Digital Forensics is defined in many different ways and there is no standard definition (Politt, 2004, cited in Ieong, 2006). Likewise, there are several different digital forensic investigation (DFI) procedures mentioned in the previous literature. Some papers discuss the technical aspects in potential evidence data acquisition cases while others focus on the analysis of the acquired data (Brill and Pollitt, 2006, cited in Ieong, 2006).

However, different DFI procedures are being developed in order to challenge various technologies used in the compromised systems or the devices under investigation (Ieong, 2006). As a result of the complicated technical procedures, the author (Ieong, 2006, p. S29) mentions that a technical-independent framework is required to thwart the technical gap between legal practitioners, investigators and information technologists.

In this paper, *"FORZA (FORensics ZAchman framework)"*; Ieong (2006) firstly emphasizes the three basic principles of DFI such as *Reconnaissance*, *Reliability*, and *Relevancy* (3R; see Figure 3. 6).

Figure 3.6: Digital Forensics Investigation Fundamentals (Ieong, 2006, p. S31)

Reconnaissance is one of the principles of DFI, in which a forensic investigator has to use different methods, practices and tools in order *"to collect, recover, decode, discover, extract, analyse, and convert data that are kept on different storage media to readable evidence"*, whereas *Reliability* is concerned with the digital evidence that could be non-repudiated and acceptable to a court of law (Ieong, 2006, p. S30).

Likewise, *Relevancy* is related to the values of the potential evidence for the case under investigation.

Table 3.1: A high-level view of the FORZA framework (Ieong, 2006, p. S33)

Role	Why (motivation)	What (data)	How (function)	Where (network)	Who (people)	When (time)
Case leader (contextual investigation layer)	Investigation objectives	Event nature	Requested initial investigation	Investigation geography	Initial participants	Investigation timeline
System owner (if any) (contextual layer)	Business objectives	Business and event nature	Business and system process model	Business geography	Organization and participants relationship	Business and incident timeline
Legal advisor (legal advisory layer)	Legal objectives	Legal background and preliminary issues	Legal procedures for further investigation	Legal geography	Legal entities and participants	Legal timeframe
Security/system architect/auditor (conceptual security layer)	System/Security control objective	System information and security control model	Security mechanisms	Security domain and network infrastructure	Users and security entity model	Security timing and sequencing
Digital forensics specialists (technical preparation layer)	Forensics investigation strategy objectives	Forensics data model	Forensics strategy design	Forensic data geography	Forensics entity model	Hypothetical forensics event timeline
Forensics investigators / system administrator/operator (data acquisition layer)	Forensics acquisition objectives	Onsite forensics data observation	Forensics acquisition / seizure procedures	Site network forensics data acquisition	Participants interviewing and hearing	Forensics acquisition timeline
Forensics investigators / forensics analysts (data analysis layer)	Forensics examination objectives	Event data reconstruction	Forensics analysis procedures	Network address extraction and analysis	Entity and evidence relationship analysis	Event timeline reconstruction
Legal prosecutor (legal presentation layer)	Legal presentation objectives	Legal presentation attributes	Legal presentation procedures	Legal jurisdiction location	Entities in litigation procedures	Timeline of the entire event for presentation

Then, the author (Ieong, 2006, p. S29) summarizes *"eight different roles and their responsibilities in a digital investigation"* after re-examining the tasks need to be performed during an investigation (see Table 3.1).

In addition, the eight different DFI roles are classified into different layers that are *"interconnected to each other through sets of six categories of questions namely:* ***What*** *(the data attribute),* ***Why*** *(the motivation),* ***How*** *(the procedures),* ***Who*** *(the people),* ***Where*** *(the location), and* ***When*** *(the time)"* (Ieong, 2006, p. S32).

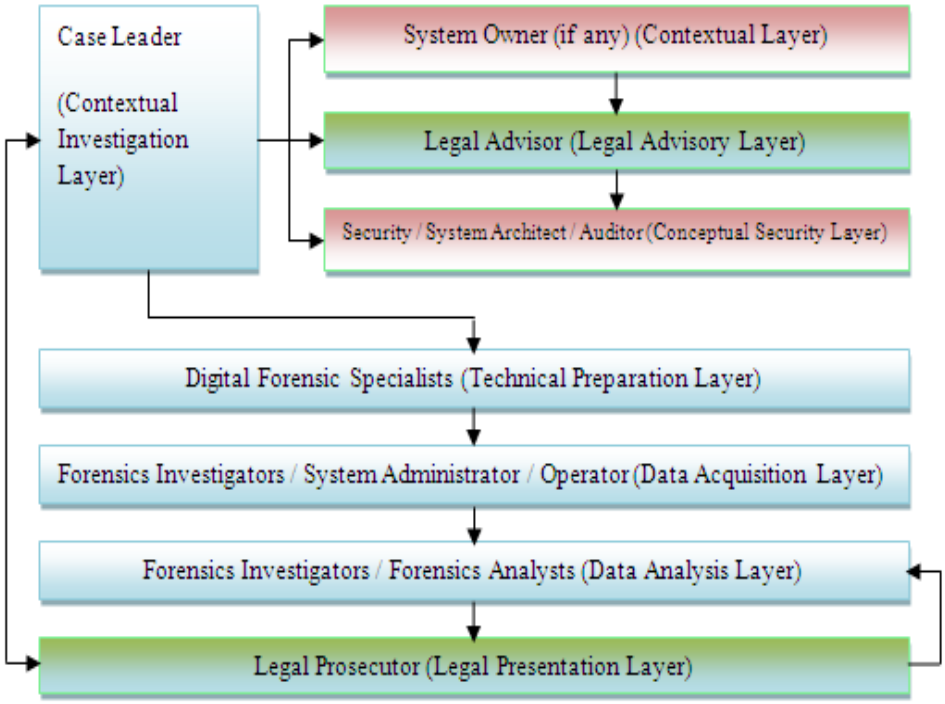


Figure 3.7: Process flow between the roles in digital forensics investigation (Ieong, 2006, p. S32)

Even though there are eight different roles for DFI processes, the author (Ieong, 2006) also mentions that some of the roles may be managed by the same person (for instance; legal advisor and prosecutor roles, or forensics investigator and forensics analyst roles). Furthermore, the flow of procedures among different roles in DFI (as shown in Figure 3.7) is also discussed in detail (Ieong, 2006, p. S31-S33).

Afterwards, the author illustrates the FORZA framework by applying it to an assumed case in which a corporate web system is hacked. In conclusion, Ieong (2006, p. S36) explains how "*the FORZA framework will be formulated as a semi-automatic investigation toolbox*" that would allow the forensics investigators to carry out *"fast and zero-knowledge data acquisition"*, as a future work.

3.2 THE RESEARCH DESIGN

In the previous Section 3.1, the reviews of seven similar works have been identified and analysed in order to develop the methods that will be used in the research.

After the reviews of the similar published works (Section 3.2.1), a review of the problem areas in a RFID business system (Section 2.5) will be discussed. Hence, the main objective of the research is to investigate the presence of digital evidence after the theft of a Stock Item (SI) in a prototype of commercial retail environment using a RFID stock management system. Thus, a prototype of a commercial retail situation of RFID system will be constructed in the laboratory by using a single RFID (read/write) tag. The normal operation of this prototype will be documented as the trusted operation of a stable RFID retail system. Afterwards, the system will be stressed by running transaction scenarios through it. For instance, those scenarios will be hacking the chip and backend database of the system in order to adjust the price on the tag. The unknown situation or unstable running condition of the system will then be documented.

The following sections (Section 3.2.1 and Section 3.2.2) will help formulate the research question in Section 3.2.6, which is followed by the hypotheses (Section 3.2.7). In the later sections, brief explanations of what the case study is (Section 3.2.3), what the descriptive method is (Section 3.2.4) and an overview of the research (Section 3.2.5) that comprises five main research phases will be discussed prior to the research question (Section 3.2.6). Afterwards, the feasible statements of the classified hypotheses (Section 3.2.7) will be explained as the elements of the research. Then, the data mapping of the main research question (Section 3.2.8) relating to the research sub-questions, hypotheses, phases of research, and data collections will be presented as a data map (see Figure: 3.9). The collected data will be analysed and the analysis results will be linked to the hypotheses that are mapped to the research sub-questions. In addition, the answers to these research sub-questions will give the solutions to the main research question.

3.2.1 Review of Similar Works Leading towards the Research Methodology

Seven reviews of similar works were described in Section 3.1. The study conducted by Cerrudo (2003) did was not only to make the security professionals aware that a potential SQL injection attack could have a critical effect on a business organization, but also to give recommendations on how to prevent the malicious attacker's manipulations of data in the backend databases and other applications on the network. Likewise, the research performed by Rieback et al., (2006b) was to warn the middleware designers that a RFID tag could be used as an attack vector to exploit backend database servers in order to change the contents of the databases.

Hence, the articles written by Cerrudo (2003) in Section 3.1.1 and Rieback et al., (2006b) in Section 3.1.2 give an idea of the type of data manipulation attack by using RFID tag towards the backend database server of a RFID stock management system (Section 2.3) of a Business System (BS). Therefore, the case of a BS being violated by a SQL injection attack through the Read/Write RFID Tag (Section 3.3.2) will be replicated (Rieback et al., 2006b). Even though the attack case scenario of the research will be replicated, the data requirement in this attack scenario (Section 3.3) will be completely different from previous literature. For instance, the system design (Section 3.3.1) of the research is not the same as that of the system mentioned in previous literature and the middleware software has to be developed (according to the research requirement) by using Software Development Kit (SDK) provided by the company that manufactures the RFID scanner. Furthermore, it is important to acknowledge that none of the researchers (Cerrudo, 2003; Rieback et al., 2006b) discussed that the potential for a breach of asset protection could be high and hence would increase the requirement for forensic readiness in any RFID BS. Most importantly, there was no forensic analysis of the RFID BS in the previous literature.

One of the objectives of the research phases (Figure 3.8 in Section 3.2.5) is to create a prototype of a commercial retail environment using a RFID stock management system within the laboratory (see Section 3.3.1, in which a Microsoft SQL Server 2005 will be utilized as a backend database server). Subsequently, the BS will be stressed by a SQL injection attack in order to change/manipulate the data from the backend server and then each entity in the BS (Table 2.5 in Section 2.3) will

be investigated for evidence of the theft. Hence, the forensic analysis of the SQL Server related to previous works (Sections 3.1.3 and 3.1.4) should be revised. As stated in Section 3.1.3, Fowler (2009) discussed the differences among traditional forensics and the SQL Server forensics by giving two fictitious scenarios in which the attack of the first scenario was originated via the Web and the latter was caused by an unhappy employee. Then, Fowler (2009) explained the ways in which the investigation should be performed on the SQL Server related incidents in a forensically sound manner. In addition, the ways in which the phases of forensic methodology performed in the SQL Server forensics could be utilized to identify a database security breach. On the other hand in Section 3.1.4, the research conducted by Fowler (2007) described how to perform a forensic analysis of a Microsoft SQL Server 2005 by using a real world security incident scenario. The values of the articles (Fowler, 2007; Fowler 2009) were to highlight malicious attacks on SQL Servers could come from not only the Web, but also from a malicious employee. In addition, it is important to note that a live forensic analysis has to be performed on a SQL Server in order to acquire all volatile and non-volatile SQL Server artefacts (see Appendix 1). With the exception of the ways in which a forensics investigation of a SQL Server incident is conducted, none of the papers written by Fowler (2007, 2009) were related to the forensic investigation of a compromised RFID BS as a whole and the malicious attack compromised the SQL Server from the web. However, the idea of creating IRCD (Incident Response CD) and the application of SQL forensic methodology would be replicated for developing a customized **Helix_RFID_IncidentResponse (Helix_RFID_IR) toolkit** (see Appendix 2) and investigating the backend database server respectively.

As stated in Section 3.1.5, Wang et al., (2009) proposed a live physical memory forensic method of a target or compromised host (workstation or server). In fact, important potential evidence can be acquired from the physical memory of a running workstation or server. Otherwise, the evidence of a compromised RFID BS will not be complete and the volatile evidence from the memory will be lost when the power supply of the target host is turned off. Additionally, most business companies may not permit a forensic investigator to take the server offline (Fowler, 2009). Thus,

the forensics investigation in the research will be performed as a live forensic investigation.

In addition, Wang et al., (2009) discussed the issues of live memory forensics. Apparently, issues such as evidence integrity, repeatability, the extent to which the raw evidence from the physical memory will be affected by running the forensic tool on a target system should be noted. In research's system design (Section 3.3.1), there is a single UHF RFID scanner/reader which will be attached to the test-station (acting as a point of sale terminal). Hence, the forensics analysis of the memories of RFID scanner and the test-station will be performed by using the model proposed by Wang et al., (2009). But, the tools used in evidence collection of the research (Section 3.3.4) will be perfectly different. For instance, there is no such tool to acquire raw (bits to bits) evidence data from the memory of RFID reader in the industry according to the previous literature (during the time the author was conducting the research). For this reason; a customized memory extraction tool, **ReaderLogExtractionTool**, for acquiring bit-to-bit transaction logs from the RFID reader's memory needs to be developed by using C# programming language (source code can be observed in Appendix 3).

In the previous Section 3.1.6, Cerrudo (2009) mentioned that the forensic investigator can find valuable information not only in the SQL Server error logs and application logs, but also in the memory. Moreover, SQL default traces and transaction logs are significant in finding evidence as data modifications made by every transaction are recorded with timestamps.

Even though there are several DFI procedures in the previous literature, the *"eight different roles and their responsibilities in a digital investigation"* and the process flow among them as discussed by Ieong (2006, p. S29) in Section 3.1.7 are very important facts to understand in order to narrow down the technical gap between legal practitioners, investigators and information technologists. However, it is unlikely for the investigators to carry out data acquisition in the near future by using a semi-automated investigation toolbox in a forensically sound manner without having any knowledge of forensic data acquisition methods.

3.2.2 Review of Problem Areas in RFID Stock Management System

As mentioned in the previous chapter (Chapter 2; Section 2.4), there are so many threats and challenges in RFID enabled BS. Different researchers in recent literature proposed different RFID security risk models (Garfinkel et al., 2005; Karygiannis et al., 2006; Ding et al., 2008; Mitrokotsa et al., 2010) while other researchers discussed the classifications of RFID attacker behaviors (for example: Mirowski, Hartnett & Williams, 2009). Nonetheless, the problems concerned with the proposed security risk model related to this research (see Section 2.5, Figure 2.15) will be reviewed and discussed as follows:

3.2.2.1 The SI Entity Security Risk

The architecture and engineering of a RFID Tag (Section 2.1.1) invites a suite of obvious attacks that may exploit any radio frequency device, for example, as stated in Sections 2.4 and 2.5, radio frequency jamming, sniffing, teasing (for replay), tracking, and interference (disrupt and denial) (Bolan, 2007; Rao, et al., 2005). Some of the less obvious vulnerabilities are in the perceived appetite for risk. The false sense of security engendered by the innovation and the 'off-line' context reduce awareness of the potential threats and yet the high value of the stock inventories managed by RFID tags heightens the criminal's motivation to take a risk. The perception that RFID tags (Section 2.1.1) are small and cannot be protected overlooks the growing capacity of Tags and the strengthening of encryption algorithms. The scanners (Section 2.1.2) and the relatively large number of lines of source code use to read and write on tags provide a celebrity challenge and backdoor for criminal activity.

According to the literature review in Section 2.1.1, RFID tags are passive and active by nature of their design and each type has particular risks. The type of tag can determine the attack vector and the proximity an attacker must gain (Mirowski et al., 2009). For example a tag with a range of one meter or less would require the presence of an attacker onsite and possible social engineering for a radio attack to be effective. Passive tags are open to kill hits and hence the retail exit security can be negated (Tu & Piramuthu, 2008; El-Said & Woodring, 2009). Active tags with the read/write feature can be

hacked in order to change the ID to cheat pricing scales (Haines, 2006a, 2006b). Furthermore, both types of tags are vulnerable to worms and viruses. In the real world context, practices of physically swapping a high value tag with a lower value tag are being reduced by customized attachment designs and detachment tools.

3.2.2.2 The POS Entity Security Risk

The POS Entity (Section 2.3) has many potential security risks that fall beyond the interest of this research and are published elsewhere, for example card skimming (Heydt-Benjamin et al., 2007; Rieback, et al., 2006b). The principle risk to the RFID stock management system (Section 2.5) is the substitution and incorrect handling of the Read/Write tags at the POS. The Business System processes of attaching and detaching tags to SI occurs at the POS. These services require the input from the BIS and the SI entities to assure the release from and the capture into the Business System. It is at this point that the value is realized and extracted from the business process and hence vulnerabilities exist.

Tags that are released from the system (using a design tool) require physical protection from theft or substitution. To this end a secure receptacle is required within the release tool to assure all tags are captured and no substitutes are deposited. Similarly policies are required for the management of released tags and the rewriting of these tags. An audit is to be maintained of the number of tags engaged in the different processes of the Business System and every released tag is to be hashed and re-written (i.e. a tag from one SI may not be reattached to a similar SI until due process is complied). Similarly the attachment of a tag is to be tested to assure the Tag cannot be easily removed and stolen for analysis / disclosure (Chawla & Ha, 2007).

3.2.2.3 The BIS Security Risk

The threats to the Business Information System (BIS) mentioned in Section 2.5 are related to that of the database/warehouse application, the human

participants, the Information System (IS) build, and the Information Technology (IT) that supports the IS.

Most importantly, the most effective way to mitigate the risk of theft (which is the main interest of this research) is to implement protective policies and to put in place physical barriers to shield wrongful transactions. In a compliant security environment only the residual risk remains. The residual represents a percentage chance of an occurrence and therefore forensic readiness is a requirement for post-event assurance. Intrusion detection systems that mine the BIS transaction logs may provide alert and also regular physical stock taking (made easy by reader scanning of tags). To successfully breach a security compliant RFID stock management system, an attacker must co-ordinate an orchestrated script of social engineering, tag cloning and the physical SI release (Kim, Shin, & Park, 2007; Landit, 2005; Li, Xu, & Yu, 2008).

3.2.2.4 Human Actors

In addition to the entities of SI, POS and BIS security risks, the investigation of a crime scene considers all matters evidential. In the previous studies, reviewed attention has been paid to IT and IT Systems technical detail. The scope of inquiry is however wider than isolated technical details and considers integration scenarios and multidisciplinary complexities. The humans who interact with the technologies (the actors) trap and convey evidence at the scene that requires extraction. In different instances, the actors are perpetrators and victims (both conscious and unconscious) (Gonzalez, Sarriegi, & Gurrutxaga, 2006). Social engineering concerns the deception of people in order to have them disclose systems sensitive information that can be used to compromise integrity and to defraud the system of resources. The manipulation of people, in addition to the errors and inadvertent mistakes they make, are significant security risks (Workman, 2008). The potential for disclosure of information that leads to the compromise of passwords, encryption keys, SI codes, SI prices, and other mission critical information is high. The Business System security risks described in Section 2.5.3 are

increased when social engineering or human factor is added into the mix (Qin & Burgoon, 2007).

The social engineer is able to extract information that may provide precision in an attack and also leave less digital evidence. The perpetrator also requires less technical knowledge and can execute a social control strategy (Samani, 2010). The common approaches of impersonating roles, such as friend, colleague, technician, authority figure, and so on, are less effective than an employee compromising the Business System. In this way the exact and sensitive information regarding the SI Tags, the POS processes and the BIS architecture may be disclosed. The technical security measures such as release mechanisms and encryption are generally reliable, robust and effective controls. However, the people who specify, build, use, and manage the Business System can be persuaded into overriding the control system. Social engineering is a powerful technique for gaining unauthorized access to confidential proprietary or personal information. The risk to the Business System is escalated by drip feed gains where an attacker starts with publicly available information and then leverages the knowledge for social relationships and escalating gains of sensitive information. The approach includes coercion (eg. blackmail) and different sequenced win-loss situations for a targeted internal customer (current or former). The soft edge of profiling trivial includes casual gossip and rumors, and the frame internal procedures, roles and responsibilities.

The system impact of successful social engineering has consequences over a range of business performance indicators. In our study we are principally concerned with theft and hence the additional risk social engineering has for Business System violation. Loss of integrity, trust, system utility and other collateral damages caused by successful social engineering fall outside of the interest of this thesis. The interest focuses on criminal activity and the potential to collect robust digital evidence.

3.2.3 Case Study

The researcher intends to look at a single network system so that specific investigative practices may be recorded. A case study is to be constructed from literature, the build and scenario testing. Case study can be defined as the intensive study of a single case where the purpose of the study is to be able to generalize a theory to a population of cases (Gerring, 2007). According to Yin (2003, cited in Dul and Hak, 2008, p. 4), *"a case study is an empirical enquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between object of study and context are not clearly evident"*. Similarly, Collis and Hussey (2003) define case study to be an extensive study of a single instance of a phenomenon of interest. In addition, Collis and Hussey also refer case study as exploratory research. Dul and Hak (2008) simplifies the definition of case study to be a study of a single case or multiple cases in its real life situation and analyze data that are obtained from these cases qualitatively. Yin (1994, cited in Collin and Hussey, 2003) identifies the following characteristics of case study research:

1) to explore certain phenomena and understand them within a particular context;
2) to conduct the research without a preset of questions and notions about the limits within which the study will take place; and
3) to use multiple methods for collecting data which can be both qualitative and quantitative.

3.2.4 Descriptive Method

The descriptive method is useful for implementing IT testing because every system is different and the integration of different software and hardware is not a science. The technique will be employed in the set up of the RFID system, initial testing and scenario testing.

One of the key methods in systems development is the descriptive documentation. As the build proceeds a diary is to be kept that describes each occurrence (the good and the bad) in the process. The diary later becomes the target

for analysis so that the best practice can be identified. Also designs are sketched there and network relational maps.

3.2.5 Research Overview

The research involves five different phases including literature review as mentioned in the above introduction (Section 3.0) and the pictorial overview of these research phases can be seen in the following figure (3.8).

In the first stage of the research the comprehensive search of publications from the Institute of Electrical and Electronics Engineers (IEEE), the Association for Computing Machinery (ACM), and the like was conducted for the past 10 years. The literature was analyzed and the learning was compounded. The testing environment will be created within the laboratory to simulate a RFID tag retail stock inventory system (see Section 2.3) with stock item (SI), point-of-sale (POS), and business information system entities (BIS).

Then the system will be stressed by different types of attacks such as SQL injection attack (will be adopted from Section 3.1.2) and the like. After the completion of the attacks; each entity, sub-system and service will be interrogated to identify evidence left from the attacks. The final phase of the research consists of an evaluation of the learning in the form of data analysis and presentation.

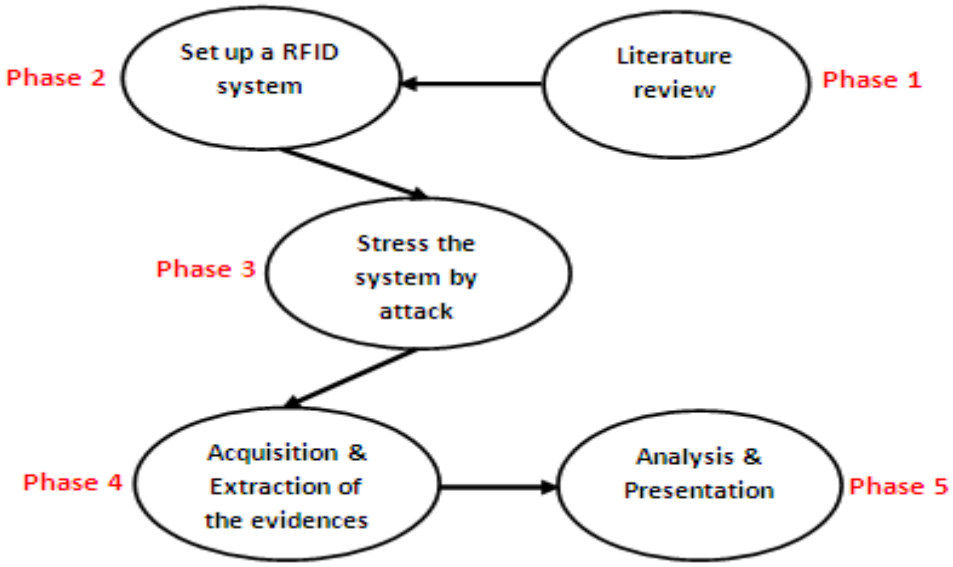


Figure 3.8: Research Phases (author)

3.2.6 Research Question

As stated in the preceding sections (Sections 2.4, 2.5 and 3.2.2), there are so many problem areas and challenges in RFID system. The literature review in the previous chapter (Chapter 2) presents the knowledge in which how RFID works (Section 2.1), where the data is stored (Section 2.2) and how the data can be compromised in the RFID retail system (Section 3.1.2). However, the extraction of the evidence or data in a forensically sound manner is important to the investigators when a RFID retail system is compromised. Hence, the main research question in this study is:

> What should the forensic investigator do in order to perform the complete and accurate forensic examination of a compromised RFID stock management system in the retail sector?

Based on the above mentioned main research question, the sub-questions can be derived down into the following:

> What are the locations where the evidence can be found?
>
> What evidence can be extracted from a compromised RFID based retail system?
>
> What is the best way to preserve the acquired evidence?
>
> What are the methods to analyze the acquired evidence?
>
> What are the capabilities to determine the attack event?

3.2.7 Hypotheses

Even though there are variety of different attacks, as described by Mitrokotsa et al., (2009, 2010), in Section 2.5, at different layers such as physical layer (for instance: permanently disabling tags, relay attacks), network transport layer where tag, reader and network protocol attacks can happen, application layer where the middleware attacks (buffer overflow, malicious code injection and the like) can be occurred, strategic layer (social engineering attack, privacy threats and the like) and multilayer attacks (such as denial of service, traffic analysis, crypto and replay attacks); the

hypotheses of the research will be based on how the tag data can be compromised by using a few case scenarios.

Hypothesis 1 (H1): Backend database server can be compromised by a malicious RFID R/W tag.

Hypothesis 2 (H2): The value of a SI tagged with R/W RFID tag can be changed.

Hypothesis 3 (H3): There will be transaction traces in the RFID reader memory after the attack.

Hypothesis 4 (H4): There will be transaction traces in the POS/Server's memory after the attack.

Hypothesis 5 (H5): There will be transaction traces in the SQL Server transaction logs after the attack.

Hypothesis 6 (H6): There will be transaction traces in the SQL Server error logs after the attack.

Hypothesis 7 (H7): The malicious tag can be found at the crime scene.

Hypothesis 8 (H8): The significant evidence can be extracted by analyzing collected data with EnCase forensic software, Windows Forensic Toolchest, and a hardware write blocker (*Tableau Forensic USB Bridge*).

Hypothesis 9 (H9): The MD5 hash values of the collected evidence before and after analysis are the same. Hence the collected evidence is credible, reliable, repeatable, and acceptable to the court of law.

3.2.8 Data Map

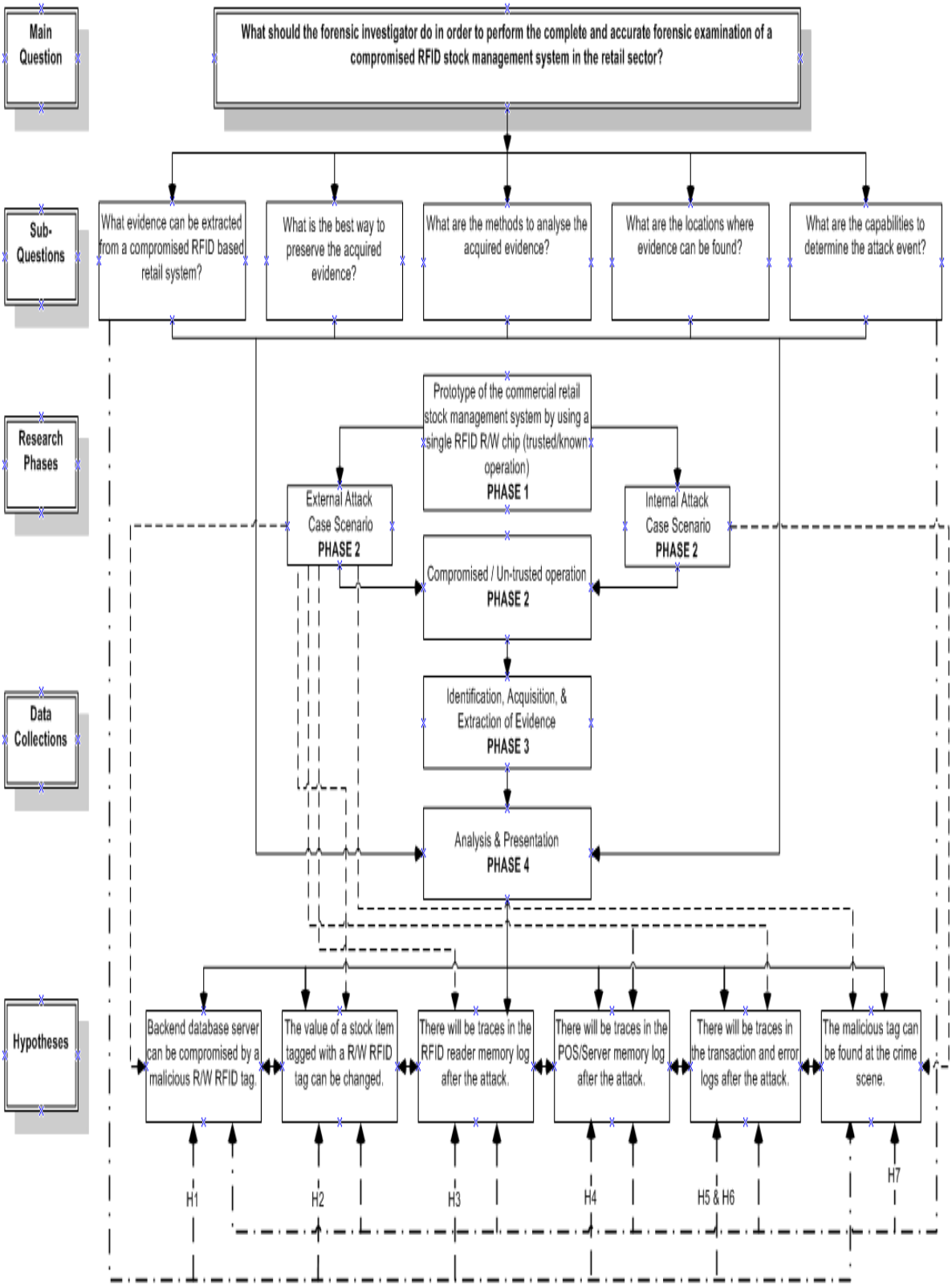


Figure 3.9: Data Map (author)

3.3 DATA REQUIREMENTS

The data requirements for the research project include several different sources, according the Business System (BS) architecture of a RFID stock management system (see Table 2.4). The BS consists of three entities namely the Stock Item (SI), the point of sale (POS), and the Business Information System.

BS = SI + POS + BIS

Figure 3.10: The Business System Entity Composition (author)

As mentioned in the previous Chapter 2 (Section 2.3), each entity has sub-systems and services that are required by the other entities. For instance, the SI requires RFID Tags, a scanner, and services from POS and BIS. The POS requires a Transaction Processing System (TPS), scanners (for cards, Tags and Chips), a Tag attach/detach service and the services of SI and BIS.

Hence, to investigate the presence of digital evidence after the theft of a SI; a prototype of commercial retail environment using a RFID stock management system will firstly be constructed in the laboratory. Secondly, the SQL poisoning attack will be launched through RFID tag. Then, each entity in the BS will be investigated for evidence of the theft. The extraction of the evidence will take place from the tag, the scanner, the POS, and the SQL server. In addition, the closed-circuit television (CCTV) and interview evidence are considered relevant to the investigation.

3.3.1 System Design

In order to conduct the experimental research, a simple RFID stock management system (see Figures 3.10 and 3.11) will be set up by using a single test-station running Windows XP Service Pack 2 (WinXP SP2) as a POS terminal. The RFID reader driver (Tracient TraceConnect Software Developer Kit) and enterprise version of Microsoft SQL Server 2005 for backend database, in order to store data/information relevant to SI, will be installed on the test-station. Then, a single UHF RFID scanner/reader (Tracient RFID reader) will be attached to the test-station via Universal Serial Bus (USB) port. Furthermore, the customized RFID middleware

software (see Appendix 4) that can read and write the tag data from the RFID scanner to the BIS SQL 2005 Server (vice versa) will be developed, so as the Tripwire for Servers (version 4.8), and the Tripwire Manager (version 4.8.0.246) that allows to manage Tripwire for Servers from a central location in graphical user interface (GUI) mode will also be installed on the POS test-station (see Appendix 5 for Tripwire setup, and Appendix 9 for TEST-STATION information in detail).

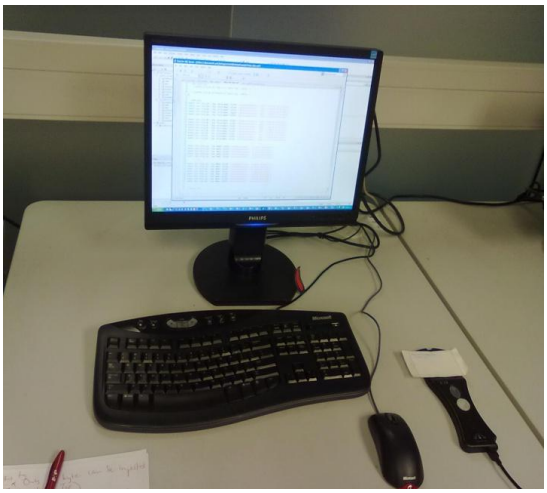


Figure 3.11: Test-Station Setup (Fake tag is read by the RFID reader)

According to the experimental requirement, only two (one real and one poisoned) RFID Read/Write tags will be applied in a stabilized BS. These RFID tags are EPC Gen 2/ 18000-6B standards and operate in the frequency range of 860MHz to 960MHz (see Table 2.2 in Section 2.1.4). The size of the user memory is 1728 bit (216 bytes) and 64 bits ID size, according to the information sheet provided by the RFID tag manufacturer, SkyeTek.

In addition, the two databases will be created in the backend database SQL Server. The primary data file named RFID_test.mdf is used for storing stock item (SI) values according the RFID tags. Likewise, the transaction log file named RFID_test_log.ldf is used for storing the reader logs to the backend SQL Server.

Then, all the fictitious product values and tag IDs will be pre-keyed into the database files (see Appendix 7). For instance, the insertion of the fictitious values into the primary data and log files are as follows.

```
insert into rfid_db (Tag, Value, Date) VALUES ('E0040000E90A4301', '1000','17:19:51 02/07/2010');

insert into rfid_log (Tag, Date) VALUES ('E0040000E90A4301', '17:19:51 02/07/2010');
```

Subsequently, the real tag (ID: E004000074251502**)** with value of $1500 will also be inserted to the backend database on the Microsoft SQL 2005 Server. Afterwards, the integrity check of the database files on the backend server will be performed, in order to set up the baseline, with Tripwire Manager (TM) before initiating the attack case scenario as TM can give the hash values in different types such as MD5, SHA, and so forth. However, only MD5 hash values are the interest of this research. Once, the integrity check report is captured and the database of TM will be updated. Hence, as mentioned in the Section 3.2.8, the data map, the prototype of the commercial retail stock management system setup will be completed in a trusted/known operation (see Appendix 7 for the steps taken before the initial attack case scenario).

3.3.2 Data Generation or Attack Case Scenario

In order to deal with the problems relating to tag data leakage, traceability, tag spoofing, cloning and the like in RFID technology of the retail BS environment (see Sections 2.4 and 2.5), it is important to implement the tag-reader mutual authentication and data encryption techniques (Kamoun, 2009; Li et al., 2009; Li & Deng, 2007). Consequently, many researchers and organizations have previously proposed variety of different mutual authentication protocols such as "*lightweight symmetric-key authentication protocol"* proposed by Juels and Weis (2005), "*EMAP"* (An Efficient Mutual Authentication Protocol) proposed by Peris-Lopez et al., (2006), "*One-Way-Reader-To-Tag"* scheme proposed by EPCglobal (2010, p. 1), "*Tag-Reader-Mutual-Authentication-Scheme"* proposed by Konidala et al., (2007), just to name a few.

However, as a result of the generally low cost with limited resources; the encryption methods used in security solution for usual computing environments cannot be performed by the RFID tags (Li & Deng, 2007). Hence, the mutual authentication protocol can be violated by the malicious attackers. For instance, Li

and Deng (2007) illustrated two effective attacks such as a *de-synchronization attack* and a *full-disclosure attack* on EMAP protocol which claimed to secure against the man-in-the-middle, replay and forgery attacks.

Thus, based on the following two assumptions, namely the breach of the authentication protocol and the successful cloning of a fake tag by the malicious attacker before entering an electronic retail shop ("**Digital House**" – a fictitious name); the attack scenario for the research experiment will be initiated.

In order to initiate the attack, as stated previously, the attacker firstly cloned a fake tag in which the malicious code is written into the user memory of the fake RFID tag (as shown in the Figure 3.11) in order to change the values from the backend server. The inserted malicious code is 56 bytes and the command is:

```
');update rfid_db set Value='600' where Tag > 'E004%' –
```

Hence, the injected command will change the values 600 to all the tag IDs starting with E004 in the databases of RFID BS. With the help of malicious employee from the shop, the attack will be initiated by replacing a fake tag (ID: E0040000E90A4302 which is injected with a malicious code in which the value $600) to a SI (on which genuine one is tagged) in order to compromise the values of SI in the backend server.

Afterwards, the integrity check on the backend server databases will be performed by using TM again (see Appendix 7).

3.3.3 Investigation Scenario

It is assumed that the IT system administrator of the retail shop is informed by email about the tripwire report every morning (see Appendix 7). However, the system administrator does not yet know whether any changes have been done by the authorized users or not. Hence, the hash values created by TM before and after the attack case scenario are reviewed to check the integrity of the database files by the system administrator. If MD5 values before and after the attack is not the same, the administrator has to perform further examination on whether the changes made are authorized or not. In other words, the RFID BS is compromised if the changes made are not authorized.

After analyzing the report on the 12th of October 2010, the administrator finds out the product database has been changed, according to the changes in MD5 hashes of the databases and all the values of SI are changed to $600.00. Thus, the changes on the values of SI are unusual as different products in the shop are tagged with different prices (see Section 3.3.1 and Appendix 8).

So, the administrator escalates the event's priority and reports it to the incident response team. Then, the report is reviewed and the forensic investigation is initiated by the first responder or forensic investigator.

3.3.4 Data Collection

Generally, the main digital evidence can be collected from the target system of interest as the digital evidence can be found in several different areas of target system (Zhang & Lin, 2010). The very first step needs to be performed by the first responder or forensic investigator is to identify and acquire the evidence. There are various procedures that need to be followed. The procedures are followed in such a way that the evidence is acceptable to court of law. For instance, the potential evidence must be identified and located. Similarly, the crime scene must be secured. So, the potential evidence of the compromised RFID BS will not be altered. Likewise, Britz (2009) stated that the preservation is also a part of seizing the evidence such as obtaining the warrant, planning the seizure, securing the crime scene, identifying, locating, documentation and transportation of the evidence, in the real world.

As stated in Section 3.2.1, the live forensic acquisition method will be used to collect all the potential evidence from the compromised RFID BS of the retail shop. Hence, the potential evidence data will be identified and acquired from several different locations such as the services, entities and sub-systems of the Business System architecture (see Table 2.5 in Section 2.3) of a compromised RFID stock management system.

Therefore, after the identification of the evidence at the crime scene during the investigation; the investigator will acquire the log files including reader's memory logs, server transaction logs, volatile and non-volatile artifacts of the SQL Server data, random access memory (RAM) of POS and the like on the live target operation

system of the target machine (Jones, Bejlich, & Rose, 2006b; Jones, 2007; Fowler, 2007, 2009; Brobler & Von Solms, 2009).

The different data collection methods during live forensic investigation include using the variety of hardware and software tools. For instances; Guidance Software's WinEn, RAM imaging tool will be used to acquire RAM of POS while the proposed **ReaderLogExtractionTool** (Section 3.2.1) will be used to acquire bit-to-bit data acquisition of transaction logs from the RFID reader's memory. Similarly, WFT tool (Section 3.1.4) will be applied in order to perform the automated evidence collection of the volatile and non-volatile SQL Server artefacts (Fowler, 2007) whereas the imaging tool like dcfldd (which is an extended version of the imaging tool "dd" and developed by the United States Department of Defense Computer Forensics Lab – DCFL) will be employed for the ac hoc acquisition of the physical SQL Server database files from the compromised RFID BS (Fowler, 2009). Moreover, the hardware write blocker like *Tableau Forensic USB Bridge* will also be deployed in order to avoid the alternation of any original evidence data and preserve them during the forensic investigation. Likewise, the hardware forensic disk imaging tool, *Disk Jokey Pro*, will be used if necessary in order to collect and preserve the bit-to-bit image of the physical hard disk of the target server/POS workstation.

However, all the required software forensic tools for data collection will be integrated into the customized **Helix_RFID_IncidentResonse (Helix_RFID_IR CD/DVD) toolkit** (Section 3.2.1) and the live data acquisition will be performed by placing it into the compromised machine's CD/DVD drive in order to avoid any modifications or affect minimum impact to the original evidence during the acquisition phase (Jones et al., 2006b; Jones, 2007; Fowler, 2007, 2009; etc.,).

For the purpose of preservation; all the collected artefacts will be digitally hashed with trusted *Message Digest Algorithm 5* (MD5), which is a 128 bit-cryptographic hashing algorithm, in order to maintain the integrity of the artefacts. MD5 hashing will be performed by using dcfldd and md5deep tools during the data collection. An example of hashing the collected data using the trusted md5deep can be found in the following (Figure 3.12). The outputs of the collected data will also be saved into a sterilized USB flash drive, which is forensically wiped with Guidance

Software's EnCase Forensics Training (**Version 6.16.1)** software on the forensic test-station. By doing so, the integrity of the original evidence will not be compromised due to the residual data from the flash drive. Hence the preservation of the artefacts will be maintained during the data collection phase.

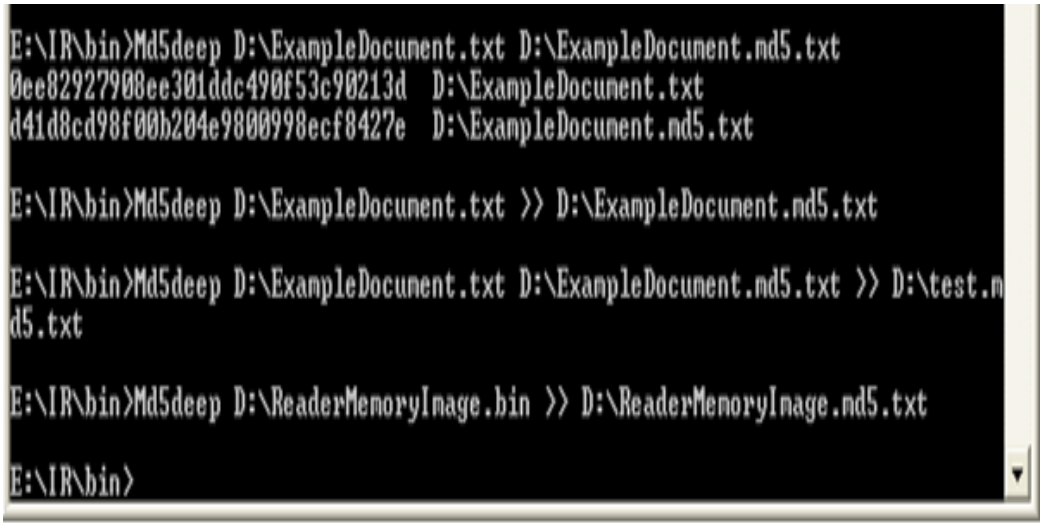


Figure 3.12: An Example of Using the Trusted md5deep Hashing Algorithm for Preservation

3.3.5 Data Processing

After the collection of potential evidence data from the entities of compromised RFID BS, the investigator has to transport and stored the acquired evidence in a secured location in order to maintain the integrity, reliability, and creditability (Ieong, 2006; Zhang & Lin, 2010).

Similarly, as part of the data processing; all the acquired data (Section 3.3.4) will be copied in a forensically sound manner by using the hardware write blocker (*Tableau Forensic USB Bridge*) that will be attached to the forensic workstation in order to preserve the integrity of the collected evidence data during the forensic investigation. In fact, the investigator will only use a forensic copy of the acquired digital evidence during analysis phase of the investigation. Only the way in which performing analysis on a forensic copy of acquired digital evidence data can maintain the integrity and security of the evidence.

Furthermore, the format conversion of the original collected evidence files will be done if required. For instance, as stated in Section 3.1.4; the transaction log files and

other collected evidence data by using WFT (Fowler, 2007, 2009) will be imported to Microsoft Excel on the forensic workstation using the same code page acquired from the target compromised SQL Server. Likewise, the value conversions of hexadecimal to decimal from *Rowlog0* (on disk value prior to transaction) and *Rowlog1* (committed transaction value) of the transaction log will be performed if necessary during the analysis of the collected data (Section 3.1.4). However, only target columns relevant to the investigation will be examined as there are more than 100 columns in a SQL Server 2005 transaction log (Fowler, 2007).

Then, the MD5 hash values of all the collected artefacts before and after analysis will have to be compared to check the integrity of the evidence in order to confirm there is no alternation on the digital evidence during the analysis phase (Vacca, 2005). Hence, the evidence can be reliable, repeatable and acceptable to the court of law (Britz, 2009; Jone et al., 2006b).

3.3.6 Data Analysis

The forensic data analysis phase of the research will mainly focus on each artifact extracted from the three entities of the simulated RFID based BS (see Figure 3.10 in Section 3.3).

Reith, Carr and Gunsch (2002, pp. 6-7) stated the data analysis phase is to *"determine significance, reconstruct fragments of data and draw conclusions based on evidence found"*. As a result, in the data analysis phase of the research; all the forensic copies of the collected evidence copies (Section 3.3.5) will be combined and analyzed on the forensic workstation by using different analysis techniques and forensic tools. For instance, the analysis of collected volatile memory artefacts of POS/Server and RFID reader will be performed by using EnCase forensic software in order to extract the valid evidence. Likewise, as stated in Section 3.1.3, the collected data concerned with volatile and non-volatile backend SQL Server data will also be analyzed in order to find the notable events irregular database activities and other potential evidence to proof the theft of SI. Then, the important evidence will be extracted and included into the timeline of investigation (Fowler, 2009). In addition to using commercial forensic tool like EnCase, the manual search of the notable evidence on collected artifact (example: the SQL Server artifact collected by using

WFT tool and ad-hoc acquisition method) may be involved in the data analysis phase. However, only the collected data in the range of investigation timeline will be analyzed and evidence recovered or analysis results will be stored on the forensic workstation.

Moreover, as stated in Section 3.3.5; the comparison of the hash values before and after the analysis will also be performed for evidence integrity checking. One of the critical parts of data analysis is documentation, as Britz (2009) mentioned, which is very significant in any forensic investigation although computer crime investigations may not be the same. Hence, all the process of analyzing collected data and evidence recovered will be documented and those documented analysis events will later be useful for integrity checking and presenting the evidence related to the theft of SI in the court. Consequently, the evidence recovered will be used to reconstruct the timeline when the attack occurs, to determine how the backend server data is manipulated, and so forth in order to answer the research question mentioned in Section 3.2.6. Thus, the forensic analysis of digital evidence is an important phase of digital forensic investigation.

3.3.7 Data Presentation

After analyzing and documenting the evidence related to the theft of SI in a simulated RFID BS, the evidence recovered will be presenting in the table or diagram formats as part of the data presentation phase in the forensic investigation.

The data presentation phase will involve the results of analysis of collected artefacts, the reasons why the tools and procedures are chosen to perform the investigation, hashing comparison of the evidence files before and after analysis for evidence reliability and integrity (as stated in Section 3.3.5), and the like. Furthermore, the correlation of the evidence examined will also be presented in order to reconstruct the malicious activity to proof the theft of SI. For instance, the table or diagram of volatile data in relation to the reconstruction of malicious attack and non-volatile data in relation to the reconstruction of the malicious attack will be presented. Likewise, the grouping the analysis results of collected artefacts towards the purpose of investigation will be presented. For example, the contributions of the analysis results of database users, authentication and authorization settings of non-volatile

SQL Server artefacts (for the purpose of making decision on whether the attack comes from authorized or unauthorized person) towards the proof of SI theft will be visualized in the presentation phase. In doing so, *"a specific time and special state of the collected digital evidence can be confirmed by the authority or a third party and can provide scientific evidence of behavior, standardized proof solidification for the judicial investigation of digital evidence"* (Zhang & Lin, 2010, p. 652).

The final part of the presentation phase will include the recommendations for the forensic investigators and business owners based on the presented results mentioned above.

3.4 LIMITATIONS

Even though there are varieties of malicious RFID attacks and the threats of RFID stock management system (Sections 2.4 and 2.5 respectively) the attack scenario mentioned in this research is using RFID R/W tag as the attack vector to compromise the backend DB server in a closed RFID BS environment. It does mean that the proposed system design (Section 3.3.1) will not be connected to the Internet. On the other hand, the Web Server applications such as Microsoft Internet Information Server (IIS) are normally used as the front end in order for clients to access backend database server from external network. These web server applications maintain log files which can contain the attack events occurred on the backend server during the timeline of investigation. However, the acquisitions of web server logs will not be covered in this project experiment. Thus, the evidence from the Web Server will not be presented.

Likewise, there are limitations in finding evidence in POS areas. POS evidence such as CCTV and interview evidence from human participants will not be available. On the other hand, the additional risks such as loss of integrity, trust, system utility and other collateral damages to BS caused by successful social engineering will not be assessed. Furthermore, the Shelf-Inventory-Read transactions will not be encountered. Moreover, Fowler (2009) mentioned that the acquisitions of the physical event logs can be done by using dcfldd utility in case of these collected files are corrupted. However, the repairing of system event log files and how to

extract the physical log files by using dcfldd utility will not be covered in this research.

Similarly, there are also some the limitations in the areas concerned with external security controls. External security controls including firewalls, intrusion detection systems (IDS), and antivirus client (AVC) can store the information about the database data and SQL client traffic flowing through them. However, the testing station in this research experiment will not be installed with IDS. Thus, the acquisition of the IDS logs will not be discussed even though the information provided by IDS logs can give critical data including the IP address of the attacker and the timestamp of the database attack to the forensic investigator. Moreover, the Windows firewall logs can provide the useful information for investigation such as the clients that are connected to backend SQL Server within the time frame of the database attack in addition to the AVC may record failed exploitation attempts by the attackers (even though the successful attempts will not be recorded). Therefore, the information contains in the failed attempts can help the investigator to create the approximate timestamp and originating source of the attack (Fowler, 2009). Nevertheless, the acquisition of evidence from external security controls will be covered in this research.

In the previous literature, some of the researchers have discussed the RFID risks concerned with privacy. For instance, Karygiannis et al., (2007) presented RFID risks into three different categories such as *business process risks*, *business intelligence risk*, and the like (Section 2.5) along with the privacy risk. However, the discussion concerned with the RFID privacy risk will not be covered in this research. Likewise, the information from Window Registry as a source of digital evidence is explained in previous literature (Dolan-Gavitt, 2008; etc.,) and it will not be covered in this research.

3.5 EXPECTED OUTCOMES

The expected outcomes are the anticipated results of a specific research project. As stated in the Section 3.2, the main objective of the research is to investigate the

presence of digital evidence after the theft of a Stock Item (SI) a prototype of commercial retail environment using a RFID stock management system.

In the proposed research experiment (see Sections 3.3.1, 3.3.2, and 3.3.3), the forensic investigation will focus on each entity of the RFID BS and the sub-systems identified in Table 2.5 (Section 2.3). Thus, based on the literature reviewed (Section 3.1); the expected outcomes of the simulated research can be established on the evidence extracted from the three entities of RFID BS as follow.

The expected outcome includes the evidence of the SI theft can be found in the log of the reader memory after the attack. Likewise, the traces of evidence such as the timestamps and malicious SQL injection code that compromised RFID BS can also be found in the logs of the backend SQL Server. Furthermore, the evidence can also be found in RAM of the POS/Server test-station.

3.6 CONCLUSION

In this methodology chapter, the comprehensive review of similar published works was described in Section 3.1. The explanation of the ways in which these previous published works leading towards the adopted research methodology was discussed in Section 3.2.1 followed by the problem areas in RFID stock management system. Likewise, Sections 3.2.3 and 3.2.4 briefly explained about what the case study and descriptive methodology were, while the overview of the research was described in Section 3.2.5.

Moreover, the research questions and hypotheses of the research project were explained in Section 3.2.6 and Section 3.2.7, respectively. The data map (Figure 3.9) presented in Section 3.2.8 was a visualization of the main research question relating the hypotheses, research phases, data collection and the like. The data requirements in Section 3.3 included a proposed RFID stock management system design for simulation experiment (Section 3.3.1), an attack case scenario for data generation (Section 3.3.2), an investigation scenario (Section 3.3.3), data collection (Section 3.3.4) and so forth. Similarly, the ways in which the collected digital artefacts would be processed, analyzed and presented were also discussed in Sections 3.3.5, 3.3.6 and

3.3.7 accordingly, followed by the limitations (Section 3.4) and expected outcomes of the research (Section 3.5).

After performing the experiment simulation according to the adopted methodology mentioned in this chapter, the following chapter is related to the findings of the research experiment. Hence, the following chapter will be explained in details including the variations encountered during the research experiment, the report of collected data, analysis of collected data and presentation of the findings in this research project.

Chapter 4 -

Research Findings

4.0 INTRODUCTION

In the previous chapter (Chapter 3), the derived research methodology was explained in detail after analyzing the previous similar published works. Likewise, the main research question and sub-questions (Section 3.2.6) that were also derived from the literature (Chapter 2) were described in Chapter 3. The research methodology offered the ways in which the implementation of the experimental RFID system design (Section 3.3.1), the experimental data creation phase (Section 3.3.2), the data acquisition (Section 3.3.4), data processing (Section 3.3.5), analysis, and presentation methods (Section 3.3.6 and Section 3.3.7 respectively).

In order to investigate the presence of digital evidence after the theft of a Stock Item (SI), a prototype of commercial retail environment using a simple RFID stock management system was constructed in the laboratory according to the proposed system design (Section 3.3.1). By design, one real and one poisoned RFID Read/Write tags were applied in a stabilized Business System. Then, the Tripwire for Servers (TS) and Tripwire for Manager (TM) were utilized to establish the baseline of a trusted/know system in operation before the SQL poisoning attack. The previous studies such as compromising the backend database by using RFID tag as an attack vector (Section 3.1.2), and the acquisition of the backend SQL server artefacts methods (Sections 3.1.3 and 3.1.4) were replicated. Thus, the SQL poisoning attack was launched through the malicious RFID tag (Section 3.3.2) and then each entity in the Business System (Table 2.5 in Section 2.3) was investigated for evidence of the theft. Evidence extraction occurred from the tag, the scanner, the POS and the backend SQL server. However, the acquisitions of evidence from the log of RFID scanner's memory and POS station's memory were performed by using developed

ReaderLogExtractionTool and WinEn respectively from the customised Helix_RFID_IR toolkit (see Appendix 2).

Hence, this chapter will report the variations encountered in the testing during the experiment (Section 4.1), what the field findings are (Section 4.2), and what the analysis results of the extracted evidence are (Section 4.3). Then, the presentation of the findings (Section 4.4) will be followed by the conclusion (Section 4.5).

4.1 VARIATIONS ENCOUNTERED IN EXPERIMENT

Even though most of steps in the research experiment were done according to the adopted methodology explained in Section 3.3, changes were made in some areas during the experiment. Pilot experiments were also performed repeatedly before the final experiment was initiated. Hence, there were problems or errors during the pilot experiments. Therefore, some changes had to be made.

The changes made during the experiment, referred to as variations encountered in the experiment, are presented in the following sections including system design (Section 4.1.1), data generation or attack case scenario (Section 4.1.2), investigation scenario (Section 4.1.3), the creation of customized Helix_RFID_IncidentResonse (Helix_RFID_IR) toolkit for evidence data collection (Section 4.1.4), the acquired data processing (Section 4.1.5), analysis of acquired evidence (Section 4.1.6), and presentation of the evidence (Section 4.1.7).

4.1.1 System Design

During the pilot testing to get the stabilized system design, there were a few problems encountered.

The problems such as the backend SQL Server was not able to establish a connection when RFID tag was read by the reader (Figure 4.1) due to error locating the server, the initial policy profile was not able to create during the Tripwire for Servers installation (Figure 4.2) due to incorrect number of parameters on the command line, (system baseline integrity checking software (Tripwire) could not open the Tripwire Manger authentication key during the initial set up (Figure 4.3), and so forth.

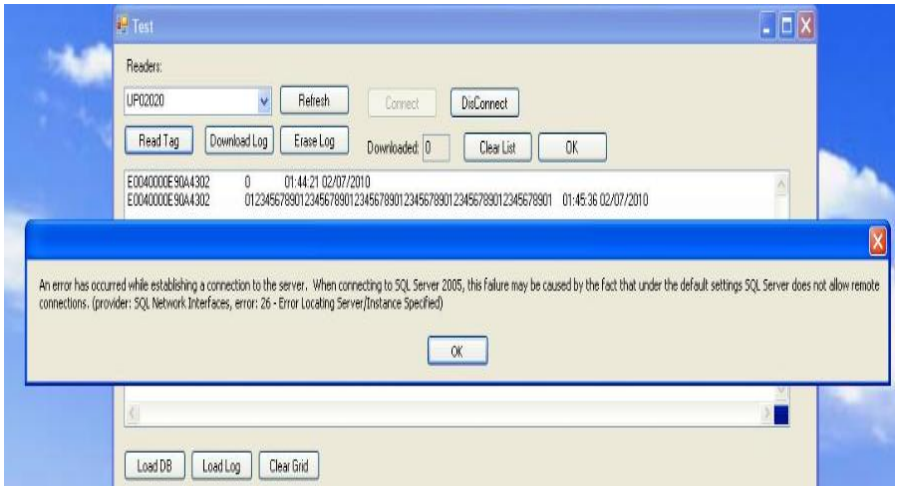


Figure 4.1: Error encountered during the tag was read by RFID scanner

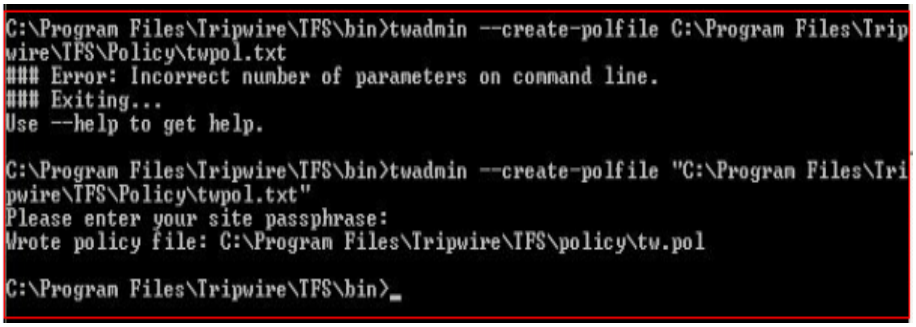


Figure 4.2: Error encountered during Tripwire for Servers set up

Figure 4.3: Error encountered during Tripwire baseline integrity checking software

4.1.2 Data Generation or Attack Case Scenario

There were a few errors that were fixed during the data generation in the pilot tests of the research experiment. For instance; reading the RFID tag injected with the malicious code by using RFID scanner was unsuccessful in a pilot test as only 62 Bytes of the tag data (see Figure 4.4) could be collected by the customized RFID middleware. Hence, the size of inserted malicious code was adjusted to 56 bytes as stated in Section 3.3.2.

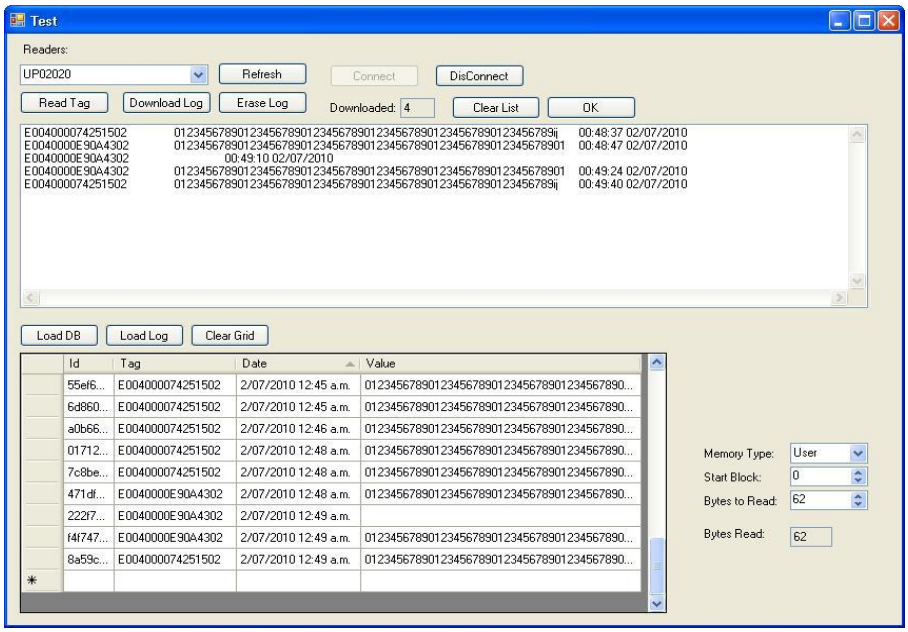


Figure 4.4: Only 62 Bytes of on-tag data can be read by the scanner

4.1.3 Investigation Case Scenario

The investigation scenario was the same, as stated in Section 3.3.3.

4.1.4 Data Collection

A few changes were made during the data collection stage. For instance, the incorrect syntax error was returned when the investigator recorded the details information regarding connection session to the backend database SQL Server instance before shutting down the Server instance. The correct syntax for recording the connection session details on the compromised system can be seen in the Section 4.2.3.5.

Furthermore, a few problems were encountered during the preparation stages before collecting evidence data such as forensically wiping the USB flash drive for the storage of collected data by EnCase (Figure 4.5), initial connection to the backend

server (Figure 4.6) due to the user account did not have full administrative rights, were initially unsuccessful.

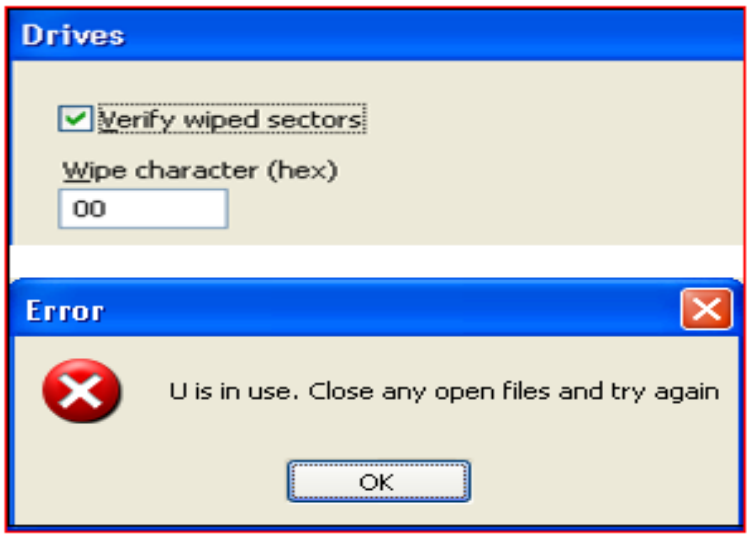


Figure 4.5: Error during wiping the USB flash drive

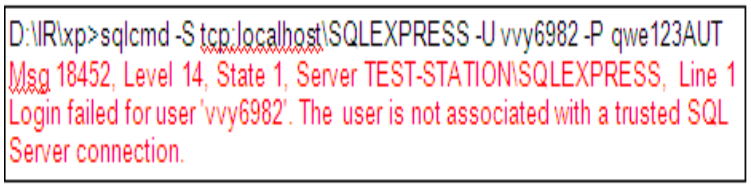


Figure 4.6: Error during wiping the USB flash drive

However, these errors were fixed before the final experiment was initiated. For example, the error of connection to backend server was fixed by keying the correct command as shown in the figure below.

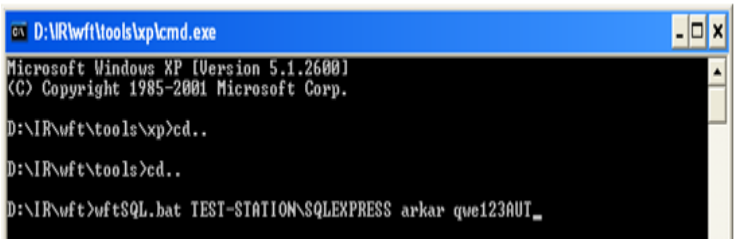


Figure 4.7: Connection to the backend server for data collection

During the collection of backend server data in the pilot tests, a few more errors (see Figures 4.8 and 4.9) were also encountered due to the failure to update the configuration file (wftsql.cfg) hashes when preparing Windows Forensic Toolchest (WFT) in Helix_RFID_IR toolkit.

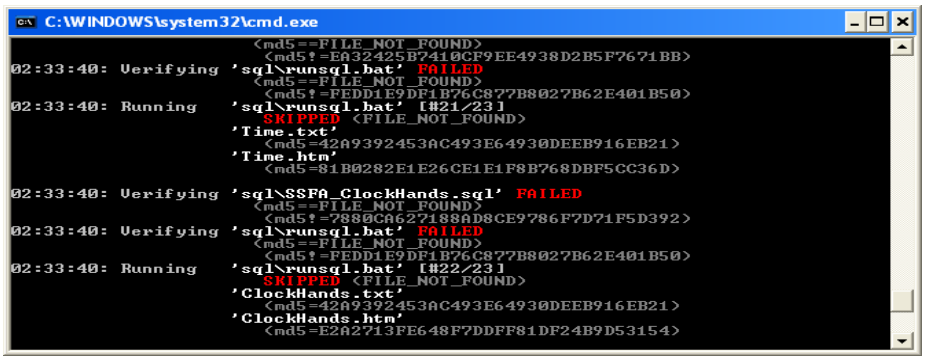


Figure 4.8: Problem running WFT from Helix_RFID_IR toolkit during pilot test

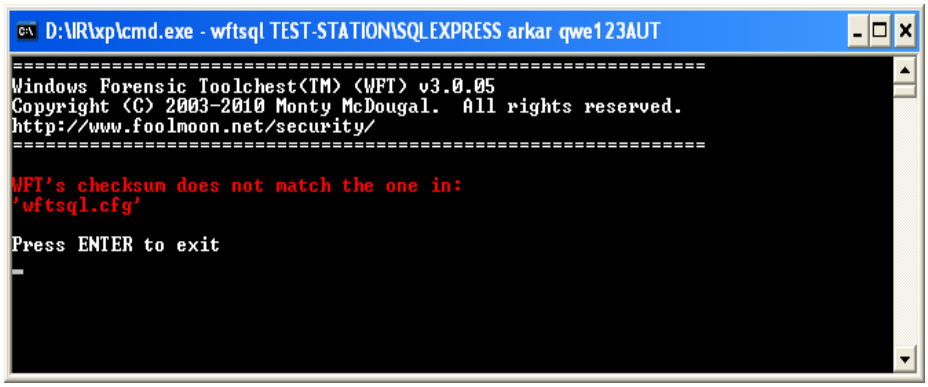


Figure 4.9: Pilot data acquisition was unsuccessful due to unmatched hash values in wftsql.cfg file

Likewise, there was an error when WFT batch file was run during a pilot acquisition of the backend SQL Server data (Figure 4.10).

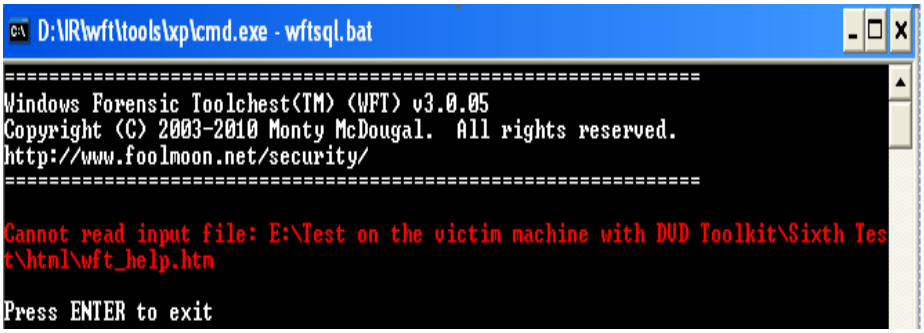


Figure 4.10: Error when running WFT batch file

4.1.5 Data Processing

There were no changes in data processing that was explained in Section 3.3.5. However, it is important to acknowledge that the findings presented in the following

Section 4.2 include the snippets of the acquired data as some of the information in the acquired data is too large to fit into the presentation format.

4.1.6 Data Analysis

The investigation scenario was the same, as stated in Section 3.3.6.

4.1.7 Data Presentation

There were no changes in the data presentation, as stated in Section 3.3.7.

4.2 REPORT OF THE COLLECTED DATA

The digital evidence data of the compromised RFID of the retail system was acquired by the forensic investigator using the forensic acquisition tools of extended Windows Forensic Toolchest (WFT Version 3.0.03), *winen.exe* and from customized Helix_RFID_IR tool.

As explained in the previous chapter 3 (Section 3.3.4), the live forensic acquisition method was used to collect all the evidence artefacts. For instance, the WFT was used to perform the automated evidence collection of the volatile and non-volatile SQL Server artefacts. Similarly the imaging tool such as *dcfldd*, which is an extended version of imaging tool dd and developed by the U.S. Department of Defense Computer Forensics Lab (DCFL), was used for the *ac hoc* acquisition of the physical SQL Server database files. All the collected digital evidence data were mostly hashed with MD5 on the fly during the acquisition in order to preserve the integrity of the collected artefacts. Likewise, the FTK Imager.exe (Figure 4.11) from AccessData Corporation was deployed to not only perform the live acquisition of physical hard drive image of the POS Test-Station computer where the backend SQL Server was installed, and but also to calculate the MD5 hash values.

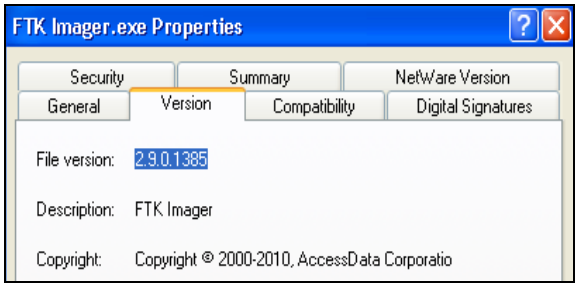


Figure 4.11: The version of the FTK Imager

The live acquisitions of all collected evidence data were performed due to the volatilities of the potential evidence data. The outputs of the collected evidence data were also saved into a sterilized USB flash drive, which was forensically wiped. Hence, the forensic investigator placed the customized Helix_RFID_IR tool into the victim's DVD drive to commence the artefact acquisition.

Figure 4.12: Helix_RFID_IR tool in action on the compromised machine

4.2.1 Live Memory (Random Access Memory) Acquisition of POS Host Station

For the purpose of the random access memory (RAM) evidence collection of the target POS host station, *winen.exe* memory acquisition tool provided by Guidance Software was used. The *winen.exe* was located in the customized Helix_RFID_IR tool and run from the trusted command prompt, D:\IR\xp\cmd.exe (as shown in Figure 4.13). Hence, the POS RAM was successfully collected and saved on the evidence collection flash drive (E:\).

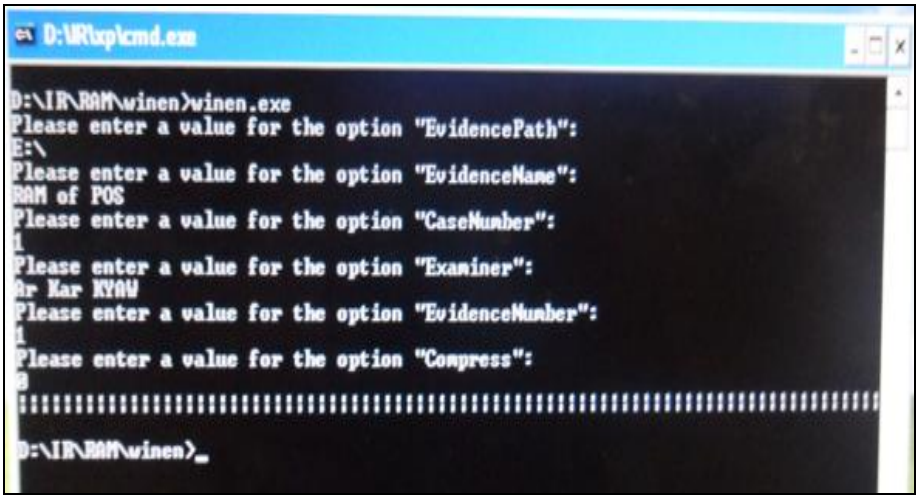


Figure 4.13: Live memory acquisition of RAM in action

4.2.2 Live Extraction of Binaries Reader's Memory Log by Using Log Extraction Tool (LET)

After the POS RAM acquisition, the extraction of logs from the RFID Reader was performed by using the developed Log Extraction Tool (LET) from Helix_RFID_IR tool. The LET (LogExtraction.exe) Tool was run by using the trusted cmd.exe from the Helix_RFID_IR tool as shown in below.

```
D:\IR\ReaderLogExtractionTool\bin\Debug>Log Extraction.exe
```

The acquired image file was saved in the evidence collection drive (Figure 4.15) after following the instructions from the user interface of the LET tool and was hashed with the md5deep hashing tool soon after the acquisition of the logs from RFID Reader (Reader's ID: UP02020) by using the syntax below (Figure 4.14).

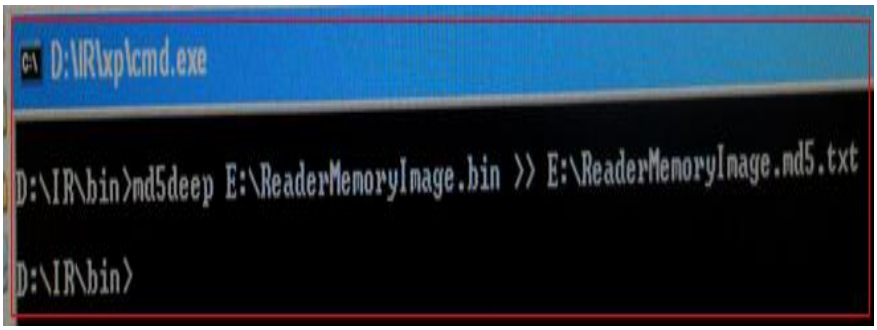


Figure 4.14: Hashing reader's memory log by using md5deep

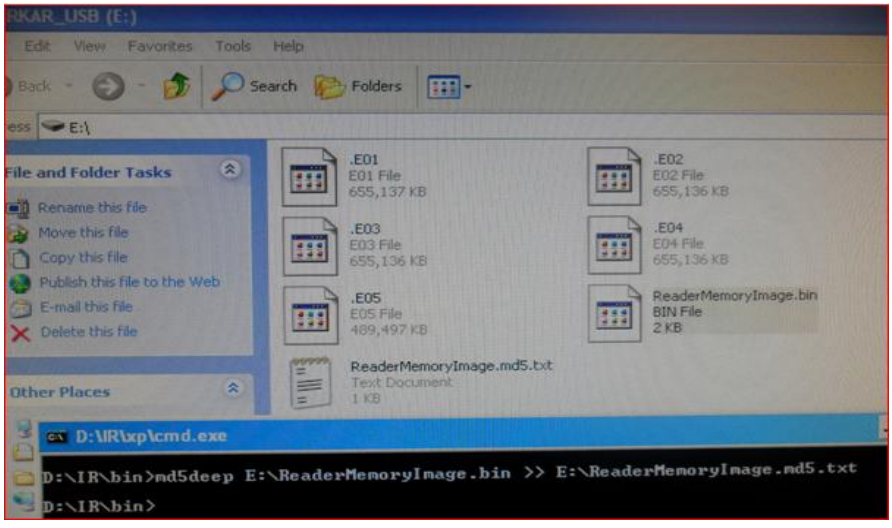


Figure 4.15: Memory log from RFID reader was saved on the collection drive

4.2.3 SQL 2005 Server Data

The SQL Server data related to the scope of investigation were collected in a forensically sound manner. The artefacts collection methods involved using the customized Windows Forensic Toolchest (WFT) from Helix_RFID_IR, the executions of ad hoc SQL commands, and the like. Hence, the trusted SQLCMD from Helix_RFID_IR was also used for connection to the victim server to collect the SQL Server related data. For example, the connection to the SQL Server instance was completed by using the following syntax (Figure 4.16).

```
SQLCMD –S.\TEST-STATION\SQLEXPRESS –s"|" –e
:out E:\sessions.txt
SELECT * from sys.dm_exec_sessions
GO
(# :out is variable is used to switch output files between statements!)
```

Figure 4.16: Connection to the SQL Server instance

Once the connection was established, the output returned a list of active sessions which were stored within the file named E:\sessions.txt. Then, the md5deep utility was used to create MD5 hash value of the file.

The hashing techniques used in the acquisitions of all SQL Server artefacts during the forensic investigation were summarized the tables of Appendix 1. As a result of the acquired artefacts have their own MD5 hash values, these hash values can later be used for integrity checking of the collected artefacts in the analysis section. Hence, the preservation of the artefacts is maintained during the forensic investigation.

Furthermore, as stated in the data processing (Section 3.3.5), the acquired data presented in this section were formatted in the customized style in which some of the irrelevant columns and rows from the acquired artefacts are excluded due to the limitations of the page layout.

4.2.3.1 Automated Artefact Collection by Using Extended Windows Forensic Toolchest (WFT)

In order to run the extended WFT which was integrated in Helix_RFID_IR tool, the SQL Server instance name of the victim was required. The server instance name was obtained from the on-site administrator of the retail shop. Once the forensic investigator knew the victim's server instance name (TEST-STATION\SQLEXPRESS), the investigator effectively collected and preserved SQL artefacts in a forensically sound manner by executing the SQL Server IR scripts from extended/customized WFT. The following syntax (Figure 4.17) was used to run WFTSQL (to run the SQL Server IR scripts) from the trusted command prompt from Helix_RFID_IR tool.

After running the SQL Server IR script, the forensic investigator initially identified the SQL Server version used on the victim's system. The server version specific information was essential to gather as it helps forensic investigator the way in which he/she should be interacting with the SQL Server by using version specific ad hoc query commands when collecting the evidence or artefacts from the backend database SQL Server of the compromised retail system.

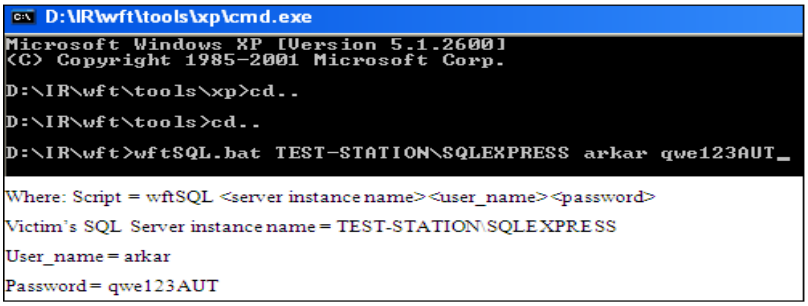


Figure 4.17: Running the SQL Server IR scripts from the Extended WFT

According to the execution result of SSFA_DbSrvInfo.sql from the SQL Server IR scripts (located in the automated WFT from Helix_RFID_IR DVD toolkit), the victim's system was running the enterprise version of SQL Server 2005 (http://support.microsoft.com/kb/321185), as shown in the Figure 4.18 below.

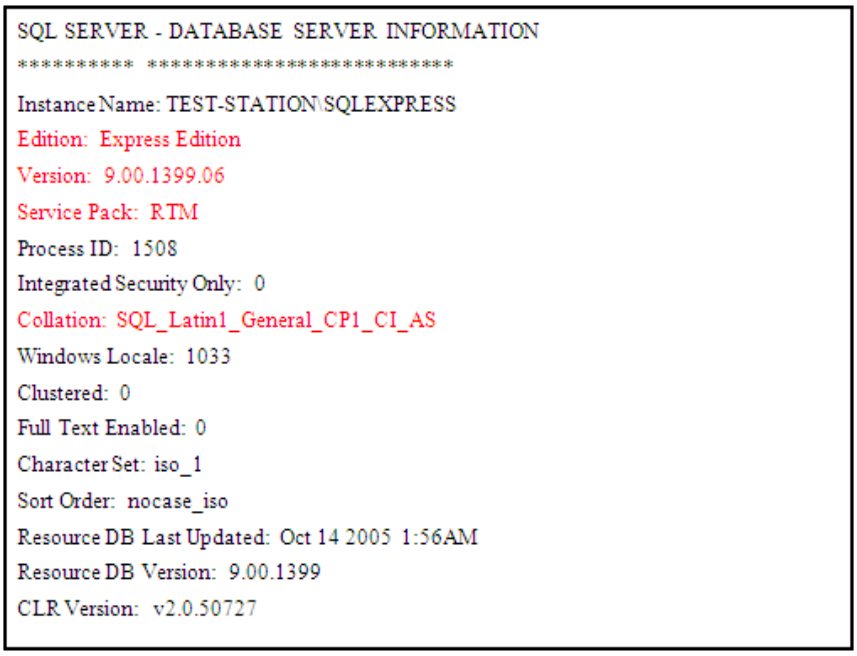


Figure 4.18: SQL Server Version of the Victim's System from the Result of SSFA_DbSrvInfo.sql

Likewise, the configuration settings of the database server could give the forensic investigator the information on how the attacker gained access to the backend database server and whether the attacker changed some configurations during the attack in order to exploit the backend server further (Fowler, 2009). For instance, the attacker could change configuration of allowing updates to the system tables by setting the configuration (ID 102: allow updates) value to 1, which could allow the attacker to perform the system tables updates directly. Hence, the acquisition method used by the automated WFT tool gave the following notable backend server's configuration setting information (Figure 4.19).

configuration_id	name	value	minimum	maximum	value_in_use
102	allow updates	0	0	1	0
518	show advanced options	0	0	1	0
544	c2 audit mode	0	0	1	0
1568	default trace enabled	1	0	1	1
1576	remote admin connections	0	0	1	0
16390	xp_cmdshell	0	0	1	0
16391	Ad Hoc Distributed Queries	0	0	1	0

Figure 4.19: Acquired Evidence of Victim's Server Configurations

4.2.3.2 Ad Hoc Artefact Collection by using Trusted SQLCMD

In order to perform interactive ad hoc artefact collection, the trusted SQLCMD binary for SQL Server 2005 from HELIX_RFID_IR tool was used. However, the connection establishment to victim's SQL server was firstly done by using the following syntax.

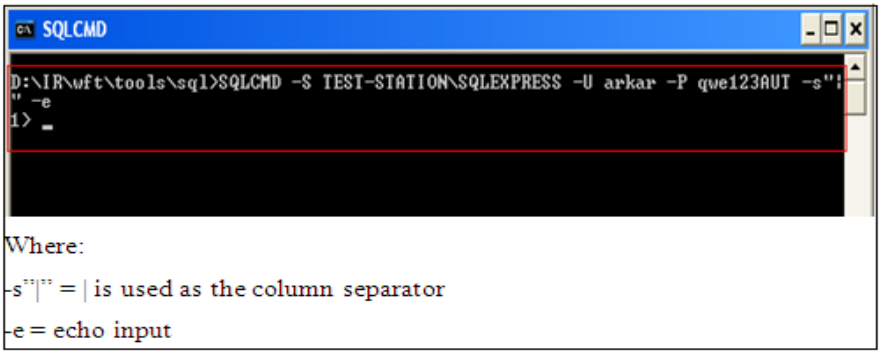


Figure 4.20: Connection to the Victim's Server for ad hoc data acquisition

Then, all the actions performed on the victim's system by using SQLCMD binary was directed to the output file by using the following command.

```
:out E:\InitialConnection.txt
```

However, the results of executed Database Console Commands (DBCC) were required to redirect to the above mentioned output file from the SQL Server log (Fowler, 2009). The following syntax, in which the trace flag 3604 was enabled, was used to direct DBCC results.

```
DBCC TRACEON(3604)
GO
```

(Fowler, 2009, p. 182)

After enabling the connection session, the session would remain effective even the investigator switch database contexts during the investigation (Fowler, 2009).

4.2.3.3 Volatile SQL Server Evidence

The potential evidence could be presented in the volatile SQL Server artefacts. The important part of the investigation was to acquire the volatile SQL Server data as early as possible due to the volatile data from the on-disk server data

file, such as the transaction log file, could be lost when the server instance was shut down or restarted (Fowler, 2009). Hence, the volatile SQL Server artefacts were acquired and preserved by using different collection methods for different purposes in order to help the investigation (see Table A1. 1 in Appendix 1).

Even though, most of the volatile server evidence was acquired by running the automated WFT, which was stated in the previous section; hence, the active virtual log files (VLFs) and ring buffer data were collected by ad hoc artefacts collection method.

The acquisition of VLFs containing the crucial volatile information such as Data Manipulation Language (DML) and Data Definition Language (DL) statements were performed early in the stage of volatile artefacts acquisition.

However, the current state of physical database transaction log files was determined by executing the following syntax through SQLCMD within the RFID_test database before performing the active VLF data (Fowler, 2009).

```
:out E:\DBSE_LGNF.txt
DBCC loginfo
GO
```

According to the result output (Figure 4.21); the only physical database transaction log was associated with RFID_test database (*FileId* = 2). The values of *StartOffset* for active (*Status* = 2) and inactive (*Status* = 0) VLFs could later be useful in order to carve inactive VLF data from the transaction log (Folwer, 2009).

1	DBCC loginfo						
2							
3	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
4	2	253952	8192	202	0	128	0
5	2	262144	262144	203	2	128	0
6							
7	(2 rows affected)						
8	DBCC execution completed. If DBCC printed error messages, contact your system administrator.						

Figure 4.21: Database Console Commands (DBCC) loginfo command results

4.2.3.3.1 Active VLF Data

After gathering the information about the current state of RFID_test database transaction log files, the extraction of active VLF data from transaction log was performed by using the following syntax run against the targeted SQL Server database under the native (sqlcmd).

```
:out E:\TransactionLog.txt
DBCC LOG (RFID_test, 3)
GO
```

Where;

Option = 0, 1, 2, 3, 4, -1 (possible values) which specifies the level of detail within the transaction log output. Value 3 can provide the detailed information about each transaction.

The extraction result was sent to the output file,TransactionLog.txt (Figure 4.22).

1	DBCC LOG (RFID_test, 3)						
2							
3	Current LSN	Operation	Context	Transaction ID	Tag Bits	Log Record Fixed Length	Log Record Length
4	----------	----------	----------	----------	----------	----------	----------
5	00000014:00000130:0001	LOP_BEGIN_CKPT	LCX_NULL	0000:00000000	0x0000	96	96
6	00000014:00000131:0001	LOP_END_CKPT	LCX_NULL	0000:00000000	0x0000	136	136
7	00000014:00000132:0001	LOP_BEGIN_XACT	LCX_NULL	0000:00000310	0x0000	48	92
8	00000014:00000132:0002	LOP_MODIFY_ROW	LCX_BOOT_PAGE	0000:00000310	0x0000	62	80
9	00000014:00000133:0001	LOP_COMMIT_XACT	LCX_NULL	0000:00000310	0x0000	48	52
10	00000014:00000134:0001	LOP_BEGIN_XACT	LCX_NULL	0000:00000311	0x0000	48	96
11	00000014:00000134:0002	LOP_DELETE_ROWS	LCX_MARK_AS_GHOST	0000:00000311	0x0000	62	152
12	00000014:00000134:0003	LOP_SET_BITS	LCX_PFS	0000:00000000	0x0000	54	56
13	00000014:00000134:0004	LOP_DELETE_ROWS	LCX_MARK_AS_GHOST	0000:00000311	0x0000	62	152
14	00000014:00000134:0005	LOP_DELETE_ROWS	LCX_MARK_AS_GHOST	0000:00000311	0x0000	62	152

Figure 4.22: Fragment of active VLF data from transaction log

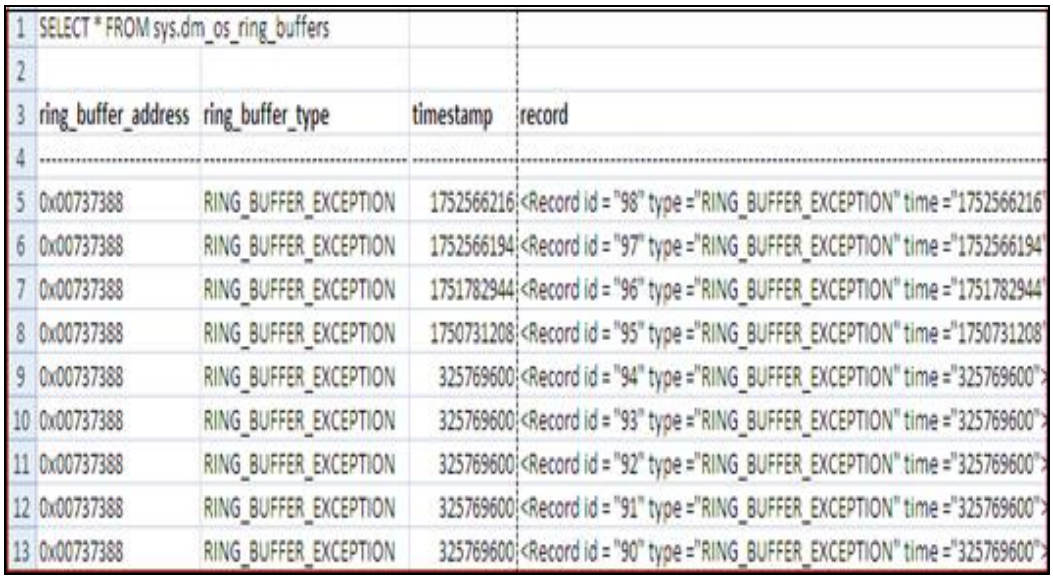
4.2.3.3.2 Ring Buffer Data

According to Fowler (2009, p. 189), the various SQL Server events information is stored in ring buffers which are *memory based SQL Server 2005 and SQL Server 2008 logs*. Acquiring, preserving and analysing of *ring buffers* can lead the investigator to get significant information to an investigation as the past

SQL Server events can be identified. Hence, the following syntax was used to acquire the *ring buffer* data.

```
:out E:\RingBuffer.txt
SELECT * FROM sys.dm_os_ring_buffers
GO
```

The fragment of acquired *ring buffer* data out was as shown in the Figure 4.23.



1	SELECT * FROM sys.dm_os_ring_buffers			
2				
3	ring_buffer_address	ring_buffer_type	timestamp	record
4	----------	----------	----------	----------
5	0x00737388	RING_BUFFER_EXCEPTION	1752566216	<Record id = "98" type = "RING_BUFFER_EXCEPTION" time = "1752566216"
6	0x00737388	RING_BUFFER_EXCEPTION	1752566194	<Record id = "97" type = "RING_BUFFER_EXCEPTION" time = "1752566194"
7	0x00737388	RING_BUFFER_EXCEPTION	1751782944	<Record id = "96" type = "RING_BUFFER_EXCEPTION" time = "1751782944"
8	0x00737388	RING_BUFFER_EXCEPTION	1750731208	<Record id = "95" type = "RING_BUFFER_EXCEPTION" time = "1750731208"
9	0x00737388	RING_BUFFER_EXCEPTION	325769600	<Record id = "94" type = "RING_BUFFER_EXCEPTION" time = "325769600">
10	0x00737388	RING_BUFFER_EXCEPTION	325769600	<Record id = "93" type = "RING_BUFFER_EXCEPTION" time = "325769600">
11	0x00737388	RING_BUFFER_EXCEPTION	325769600	<Record id = "92" type = "RING_BUFFER_EXCEPTION" time = "325769600">
12	0x00737388	RING_BUFFER_EXCEPTION	325769600	<Record id = "91" type = "RING_BUFFER_EXCEPTION" time = "325769600">
13	0x00737388	RING_BUFFER_EXCEPTION	325769600	<Record id = "90" type = "RING_BUFFER_EXCEPTION" time = "325769600">

Figure 4.23: Fragment of Ring Buffer data results from sys.dm_os_ring_buffers

The notable results included not only the *timestamp* of each buffer entry, but also the *record* of the SQL Server events.

Furthermore, the acquisition of the *ring buffer* security error was also essential as in order to determine login failures and events related to the security. Thus, the following syntax was used to acquire the security related errors.

```
:out E:\RingBuffers.txt
SELECT * FROM sys.dm_os_ring_buffers WHERE ring_buffer_type =
'RING_BUFFER_SECURITY_ERROR'
GO
```

The acquisition result of *ring buffer* security error was as shown in the figure below.

```
1  SELECT * FROM sys.dm_os_ring_buffers WHERE ring_buffer_type = 'RING_BUFFER_SECURITY_ERROR'
2
3  ring_buffer_address  ring_buffer_type  timestamp  record
4  -------------------- ---------------- ---------- ----------
5
6  (0 rows affected)
```

Figure 4.24: Ring Buffer security error results from sys.dm_os_ring_buffers

After acquiring all the volatile SQL Server evidence, the forensic investigator must perform the acquisition of non-volatile server evidence.

4.2.3.4 Non-volatile SQL Server Evidence

As stated in Section 3.1.3, the traditional digital forensic investigation approach of acquiring the evidence is firstly to collect the volatile data while the system is running and then unplug or disconnect the power cable to collect non-volatile data. However, Fowler (2009) mentions that the traditional investigation approach cannot be applied to the acquisition of non-volatile SQL Server artefact as the artefacts such as server logins, database users and the like cannot be collected when the Server is down. Hence, the non-volatile SQL Server artefacts were also acquired and preserved by using different collection methods for different purposes in order to help the investigation (see Table A1. 2 in Appendix 1). For instance, the different non-volatile server collection methods included using the extended WFT tool, which was integrated in **Helix_RFID_IR toolkit** (see Appendix 2), and ad hoc acquisition method. The findings of these artefacts are presented in the following.

4.2.3.4.1 Authentication Settings of the Backend Database Server

Brainard et al., (2006) has mentioned that the user authentication in a computing system can be accomplished through three factors such as what you do know (e.g., a password), what you do have (e.g., an ID badge) and who you really are (e.g., a fingerprint). Hence, the SQL Server authentication is a process in which the server verifies and manages login requests and the authentication mode (e.g., Windows, or SQL Server and Windows

authentication mode) can be identified by using the following *xp_loginconfig* extended procedure (Fowler, 2009).

```
:out E:\LoginConfig.txt
Master..xp_loginconfig
GO
```

After running the above mentioned code, the result was saved into the output file named "LoginConfig.txt". According to the result (Figure 4.25), the authentication mode used by the backend database, SQL Server 2005, was identified.

1	Master..xp_loginconfig	
2		
3	**name**	**config_value**
4	----------	----------
5	login mode	Mixed
6	default login	guest
7	default domain	WORKGROUP
8	audit level	failure
9	set hostname	false
10	map _	domain separator
11	map $	NULL
12	map #	-
13		
14	(8 rows affected)	

Figure 4.25: Authentication mode used by the backend database server

Hence, the configured login authentication mode was "Mixed" in which both SQL Server 2005 and Windows user accounts were allowed to gain access to the backend server.

4.2.3.4.2 Backend Database Server's Authorization Catalogs of SQL Server

After acquiring the authentication settings, the collection of authorization catalogs was performed in order to identify the access levels the user logins had within the SQL Server.

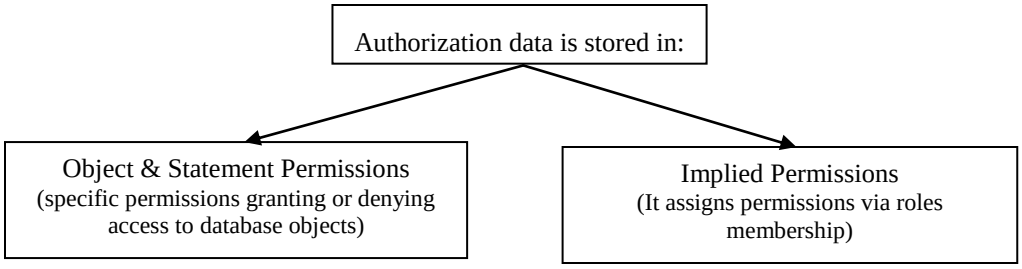


Figure 4.26: Locations of the authorization data

(adapted from Fowler, 2009, p. 196)

Depending on the version of the victim's SQL Server; the method of acquiring authorization data from the server would be different. Since the server-level authorization data of the SQL Server 2005 was stored in two different areas (see Figure 4.15), the forensic investigator had to acquire the evidence with respect to the "*Object and Statement Permissions*" and "*Implied Permissions*".

According to Fowler (2009), the server-level authorization data of the SQL Server 2005 could be acquired by viewing *SYS.SERVER_PERMISSIONS* and *SYS.SERVER.PRINCIPALS*. In order to gather server login permissions, the following syntax was used.

```
:out E:\ServerPermission.txt
SELECT * from sys.Server_Permissions
GO
```

As shown in the Figure 4.27, the output result of the preceding syntax comprised of *grantee_principal_id* (i.e., the login that received the referenced permission), *grantor_principal_id* (i.e., the identity of the login who assigned the permission), and *permission_name* (i.e., the type of permission assigned by the grantor to grantee).

	A	B	C	D	E	F	G	H	I	J
1	SELECT * FROM sys.Server_Permissions									
2										
3	class	class_desc	major_id	minor_id	grantee_principal_id	grantor_principal_id	type	permission_name	state	state_desc
4										
5	100	SERVER	0	0	1	1	COSQ	CONNECT SQL	G	GRANT
6	100	SERVER	0	0	2	1	VWDB	VIEW ANY DATABASE	G	GRANT
7	100	SERVER	0	0	101	1	VWAD	VIEW ANY DEFINITION	G	GRANT
8	100	SERVER	0	0	102	1	AUTH	AUTHENTICATE SERVER	G	GRANT
9	100	SERVER	0	0	102	1	VWAD	VIEW ANY DEFINITION	G	GRANT
10	100	SERVER	0	0	102	1	VWSS	VIEW SERVER STATE	G	GRANT
11	100	SERVER	0	0	103	1	AUTH	AUTHENTICATE SERVER	G	GRANT
12	100	SERVER	0	0	256	1	COSQ	CONNECT SQL	G	GRANT
13	100	SERVER	0	0	257	1	COSQ	CONNECT SQL	G	GRANT
14	100	SERVER	0	0	258	1	COSQ	CONNECT SQL	G	GRANT
15	100	SERVER	0	0	259	1	COSQ	CONNECT SQL	G	GRANT
16	100	SERVER	0	0	260	1	COSQ	CONNECT SQL	G	GRANT
17	100	SERVER	0	0	261	1	COSQ	CONNECT SQL	G	GRANT
18	100	SERVER	0	0	262	1	COSQ	CONNECT SQL	G	GRANT
19	105	ENDPOINT	2	0	2	1	CO	CONNECT	G	GRANT
20	105	ENDPOINT	3	0	2	1	CO	CONNECT	G	GRANT
21	105	ENDPOINT	4	0	2	1	CO	CONNECT	G	GRANT
22	105	ENDPOINT	5	0	2	1	CO	CONNECT	G	GRANT
23										
24	(18 rows affected)									

Figure 4.27: Acquired evidence of server level authorization data by using sys.server_permissions

However, the *server principal data* (a record of all SQL Server-level logins, groups and roles) was needed to acquire in order to interpret the values of *grantee_principal_id* and *grantor_principal_id* (Fowler, 2009).

```
:out E:\ServerPrincipalData.txt
SELECT * from sys.Server_Principals
GO
```

After running the above mentioned syntax, the result of server principal data information (see Figure 4.28) was saved in the specified output file of the evidence collection drive (E:\).

	A	B	C	D	E	F	G	H
1	SELECT * FROM sys.Server_Principals							
2								
3	**name**	**principal_id**	**sid**	**type**	**type_desc**	**is_disabled**	**create_date**	**modify_date**
4								
5	sa	1	0x01	S	SQL_LOGIN	0	2003-04-08 09:10:35.460	2010-10-01 22:37:39.530
6	public	2	0x02	R	SERVER_ROLE	0	2005-10-14 01:36:06.923	2005-10-14 01:36:06.923
7	sysadmin	3	0x03	R	SERVER_ROLE	0	2005-10-14 01:36:06.923	2005-10-14 01:36:06.923
8	securityadmin	4	0x04	R	SERVER_ROLE	0	2005-10-14 01:36:06.923	2005-10-14 01:36:06.923
9	serveradmin	5	0x05	R	SERVER_ROLE	0	2005-10-14 01:36:06.923	2005-10-14 01:36:06.923
10	setupadmin	6	0x06	R	SERVER_ROLE	0	2005-10-14 01:36:06.923	2005-10-14 01:36:06.923
11	processadmin	7	0x07	R	SERVER_ROLE	0	2005-10-14 01:36:06.923	2005-10-14 01:36:06.923
12	diskadmin	8	0x08	R	SERVER_ROLE	0	2005-10-14 01:36:06.923	2005-10-14 01:36:06.923
13	dbcreator	9	0x09	R	SERVER_ROLE	0	2005-10-14 01:36:06.923	2005-10-14 01:36:06.923
14	arkar	261	0xF0256F3798393E448D01328C13E1479D	S	SQL_LOGIN	0	2010-10-01 23:17:17.560	2010-10-01 23:17:17.640
15	vishal	262	0x340609DBAAFD3F48BD106F015F9EE4AE	S	SQL_LOGIN	0	2010-10-03 01:05:57.763	2010-10-03 01:05:57.783

Figure 4.28: Acquired evidence of server principals information by using sys.server_principals

By analyzing the result output; the forensic investigator could identify the SQL Server or Windows based logins, when those users were created and given access to the SQL Server.

As for the implied permissions and for the purpose of precisely verification of the access level for every user login within the SQL Server, the acquisition of *SYS.SERVER_ROLE_MEMBERS* was also necessary as it represented the identifiers of logins (*member_principal_id*) for specific roles and groups (*role_principal_id*), according to Fowler (2009). The following

syntax was used to acquire a server role membership list and the result output could be seen in Figure 4.29.

```
:out E:\ServerRoleMembers.txt
Select * from Sys.Server_Role_Members
GO
```

Select * from Sys.Server_Role_Members

role_principal_id	member_principal_id
3	1
3	257
3	258
3	259
3	261

Figure 4.29: Acquired evidence of server role membership information

4.2.3.4.3 Database-Level Authorization Data (DLAD)

Object & Statement Permissions: DLAD could be collected by viewing *SYS.DATABASE_PERMISSIONS* and *SYS.DATABASE_PRINCIPALS* to verify the access level of database user within a given database. By using the following syntaxes, the list of permissions for each user and the list of *database users, roles, or groups* within a given database could be collected (Fowler, 2009).

```
:out E\SystemDatabasePermissions.txt
SELECT * from sys.database_permissions
GO
:out E:\SystemDatabasePrincipals.txt
SELECT * from sys.database_principals
GO
```

After executing the above syntaxes, the acquired results of system database permissions and database principals were saved in the designated evidence collection drive (E:\).

The result of the backend SQL Server system database user permissions (Figure 4.19) was similar to that of the result output in the Figure 4.30. Hence,

the forensic investigator could identify the user who received the permission and the user who granted that permission by examining the *grantee_principal_id* and *grantor_principal_id values*. Moreover, the type of permissions assigned to the users could be identified by checking the *permission_name* column of the result output. Likewise, the type of permissions assigned to the specific object ID (*major_id*) could also be identified from the system database user permission results.

SELECT * FROM sys.database_permissions

class	class_desc	major_id	minor_id	grantee_principal_id	grantor_principal_id	type	permission_name	state	state_desc
0	DATABASE	0	0	1	1	CO	CONNECT	G	GRANT
0	DATABASE	0	0	2	1	CO	CONNECT	G	GRANT
0	DATABASE	0	0	5	1	CO	CONNECT	G	GRANT
0	DATABASE	0	0	5	1	EX	EXECUTE	G	GRANT
0	DATABASE	0	0	6	1	CO	CONNECT	G	GRANT
1	OBJECT_OR_COLUMN	-1073624922	0	0	1	EX	EXECUTE	G	GRANT
1	OBJECT_OR_COLUMN	-1072815163	0	0	1	EX	EXECUTE	G	GRANT
1	OBJECT_OR_COLUMN	-1072372588	0	0	1	SL	SELECT	G	GRANT
1	OBJECT_OR_COLUMN	-1071944761	0	0	1	EX	EXECUTE	G	GRANT
1	OBJECT_OR_COLUMN	-1070913306	0	0	1	EX	EXECUTE	G	GRANT
1	OBJECT_OR_COLUMN	-1070573756	0	0	1	EX	EXECUTE	G	GRANT
1	OBJECT_OR_COLUMN	-1068897509	0	0	1	EX	EXECUTE	G	GRANT

Figure 4.30: Fragment of acquired backend server database user permissions

Similarly, the result of system database principals (Figure 4.31) could give critical information such as the principal_id value of the specific database user, when the user was created, and the like.

SELECT * FROM sys.database_principals

name	principal_id	type	type_desc	default_schema_name	create_date
public	0	R	DATABASE_ROLE	NULL	08/04/2003 09:10:20 a.m.
dbo	1	S	SQL_USER	dbo	08/04/2003 09:10:20 a.m.
guest	2	S	SQL_USER	guest	08/04/2003 09:10:20 a.m.
INFORMATION_SCHEMA	3	S	SQL_USER	NULL	14/10/2005 01:36:07 a.m.
sys	4	S	SQL_USER	NULL	14/10/2005 01:36:07 a.m.
##MS_AgentSigningCertificate##	5	C	CERTIFICATE_MAPPED_USER	NULL	14/10/2005 01:56:19 a.m.
arkar	6	S	SQL_USER	dbo	01/10/2010 11:17:18 p.m.
db_owner	16384	R	DATABASE_ROLE	NULL	08/04/2003 09:10:20 a.m.
db_accessadmin	16385	R	DATABASE_ROLE	NULL	08/04/2003 09:10:20 a.m.
db_securityadmin	16386	R	DATABASE_ROLE	NULL	08/04/2003 09:10:20 a.m.

Figure 4.31: Fragment of the acquired backend server database principals information

Furthermore, the database roles were similar to those of servers and used for managing permissions. In order to acquire a listing of database role membership (for implied permission), the following syntax can be run.

```
:out E:\SystemDatabaseRoleMembers.txt
SELECT * from sys.database_role_members
GO
```

The output result (Figure 4.32), after running the above mentioned syntax, involved two columns. These were *role_principal_id* (identifies roles within current database) and *member_principal_id* (identifies database users who were members of the role) columns.

SELECT * FROM sys.database_role_members

role_principal_id	member_principal_id
16384	1

(1 rows affected)

Figure 4.32: Acquired evidence of server database role membership

4.2.3.4.4 Table Statistics Data

The statistical information about indexes and column data that are stored in the database is collected and updated by the SQL Server (Hanson & Kollar, 2010). The collections of column data are called table statistics and this data is useful for comparing against the present state of data within a table in order to identify the values that have been changed or deleted after the last statistics are updated (Fowler, 2009).

Hence, the last updated time of the table statistics was important during the period of acquisition for the forensic investigator. Firstly, to identify the table statistics generated by the columns within a database table and their last updated time; the following syntax was used (Fowler, 2009, p. 201).

```
:out E:\TableStatistics.txt
SELECT sbj.name AS 'Table', syc.name AS 'Column', STATS_DATE (ssc.object_id,
ssc.stats_id) AS 'Stats Last Updated' FROM sys.sysobjects sbj, sys.stats_columns ssc,
sys.syscolumns syc WHERE ssc.object_id = sbj.id AND syc.colid = ssc.column_ID AND syc.id
= ssc.object_ID ORDER BY [Table], [Column] ASC
GO
```

After execution of the above syntax, the forensic investigator could examine the *Stats Last Updated* (see Figure 4.33) data to determine whether the statistics were updated during the scope of an investigation or the statistics were not concerning with the columns that were relevant in the investigation.

Table	Column	Stats Last Updated
MSreplication_options	optname	14/10/2005 02:00:59 a.m.
queue_messages_1003150619	conversation_group_id	NULL
queue_messages_1003150619	conversation_group_id	NULL
queue_messages_1003150619	conversation_handle	NULL
queue_messages_1003150619	conversation_handle	NULL
queue_messages_1003150619	priority	NULL
queue_messages_1003150619	queuing_order	NULL
queue_messages_1003150619	queuing_order	NULL
queue_messages_1003150619	service_id	NULL
queue_messages_1003150619	status	NULL
queue_messages_1003150619	status	NULL
queue_messages_1035150733	conversation_group_id	NULL
queue_messages_1035150733	conversation_group_id	NULL
queue_messages_1035150733	conversation_handle	NULL
queue_messages_1035150733	conversation_handle	NULL
queue_messages_1035150733	priority	NULL
queue_messages_1035150733	queuing_order	NULL
queue_messages_1035150733	queuing_order	NULL
queue_messages_1035150733	service_id	NULL
queue_messages_1035150733	status	NULL
queue_messages_1035150733	status	NULL
queue_messages_1067150847	conversation_group_id	NULL
queue_messages_1067150847	conversation_group_id	NULL
queue_messages_1067150847	conversation_handle	NULL
queue_messages_1067150847	conversation_handle	NULL
queue_messages_1067150847	priority	NULL
queue_messages_1067150847	queuing_order	NULL
queue_messages_1067150847	queuing_order	NULL
queue_messages_1067150847	service_id	NULL
queue_messages_1067150847	status	NULL
queue_messages_1067150847	status	NULL
spt_values	name	14/10/2005 02:03:37 a.m.
spt_values	name	14/10/2005 02:03:37 a.m.
spt_values	number	14/10/2005 02:03:37 a.m.
spt_values	number	14/10/2005 02:03:37 a.m.
spt_values	type	14/10/2005 02:03:37 a.m.
spt_values	type	14/10/2005 02:03:37 a.m.

Figure 4.33: Fragment of the table statistics data

If the statistics were not relevant in the investigation, the investigator could avoid acquiring the table statistics on the SQL Server. Otherwise, the following command was used to acquire table statistics on the SQL Server 2005.

```
DCC SHOW_STATISTICS ({Table_Name}, {Column}, {Option})

Table_Name: Name of the table owning the column with the statistics
Column: Name of the column the statistics are generated against
Option: Can be use to specify the type of data returned by the command, statistics header,
density, or histogram data. This argument is available on SQL Server 2005 and higher.
:out E:\DBO_RFIDdb_tblStat_03.txt
DBCC SHOW_STATISTICS ('RFID_test..rfid_db', 'Tag or Value or Date')
GO
```

However, according to Fowler (2009, p. 204), the *Histogram* encloses "*a snapshot of actual data values that were taken at the time of the last statistics update*", and it was the significant data to acquire during a forensic investigation. *Histogram* information of the compromised database (RFID_test.mdf) was explicitly retrieved by using the following commands for all the columns and tables of interest.

```
:out E:\ DBO_RFIDdb_tblStat_Tag.txt
DBCC SHOW_STATISTICS ('RFID_test..rfid_db', 'Tag') WITH HISTOGRAM
GO

:out E:\ DBO_RFIDdb_tblStat_Value.txt
DBCC SHOW_STATISTICS ('RFID_test..rfid_db', 'Value') WITH HISTOGRAM
GO

:out E:\ DBO_RFIDdb_tblStat_Date.txt
DBCC SHOW_STATISTICS ('RFID_test..rfid_db', 'Date') WITH HISTOGRAM
GO
```

Once the preceding syntaxes were run, the *Histogram* information results (Figure 4.34, 4.35 and 4.36) against the specific columns of interest were sent to the designated output files of the evidence collection device.

DBCC SHOW_STATISTICS ('RFID_test..rfid_db', 'Tag') WITH HISTOGRAM

RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
E004000074251502	0	24	0	1
E0040000E90A4302	0	4	0	1

(2 rows affected)

Figure 4.34: Acquired statistics histogram information against Tag column of RFID_test.mdf

DBCC SHOW_STATISTICS ('RFID_test..rfid_db', 'Value') WITH HISTOGRAM

RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
10	0	40	0	1

(1 rows affected)

Figure 4.35: Acquired statistics histogram information against Value column of RFID_test.mdf

DBCC SHOW_STATISTICS ('RFID_test..rfid_db', 'Date') WITH HISTOGRAM

RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
01/01/1900 12:00:00 a.m.	0	1	0	1
31/08/2010 10:40:25 p.m.	0	1	0	1
31/08/2010 10:52:08 p.m.	0	1	0	1
31/08/2010 10:54:07 p.m.	0	1	0	1
31/08/2010 10:55:13 p.m.	0	1	0	1
31/08/2010 11:40:49 p.m.	0	1	0	1
31/08/2010 11:43:08 p.m.	0	1	0	1
31/08/2010 11:43:29 p.m.	0	1	0	1
31/08/2010 11:43:39 p.m.	0	1	0	1
31/08/2010 11:44:14 p.m.	1	1	1	1
31/08/2010 11:44:49 p.m.	0	1	0	1
31/08/2010 11:44:53 p.m.	1	1	1	1
31/08/2010 11:48:53 p.m.	0	1	0	1
31/08/2010 11:49:05 p.m.	0	1	0	1
31/08/2010 11:49:08 p.m.	0	1	0	1
31/08/2010 11:49:12 p.m.	0	1	0	1
31/08/2010 11:49:15 p.m.	0	1	0	1
31/08/2010 11:49:26 p.m.	0	1	0	1
31/08/2010 11:49:30 p.m.	0	1	0	1
31/08/2010 11:49:41 p.m.	0	1	0	1
31/08/2010 11:49:44 p.m.	0	1	0	1
31/08/2010 11:49:50 p.m.	0	1	0	1
31/08/2010 11:49:53 p.m.	0	1	0	1
31/08/2010 11:49:54 p.m.	0	1	0	1
31/08/2010 11:49:56 p.m.	0	1	0	1
31/08/2010 11:50:01 p.m.	0	1	0	1
31/08/2010 11:50:04 p.m.	0	1	0	1
01/09/2010 12:10:39 a.m.	0	1	0	1
01/09/2010 12:11:36 a.m.	0	1	0	1
01/09/2010 12:11:48 a.m.	0	1	0	1
01/09/2010 12:12:34 a.m.	1	1	1	1
01/09/2010 12:12:45 a.m.	0	1	0	1
01/09/2010 12:13:28 a.m.	0	1	0	1
01/09/2010 12:13:45 a.m.	0	1	0	1
01/09/2010 12:13:54 a.m.	0	1	0	1
01/09/2010 12:16:06 a.m.	0	1	0	1
01/09/2010 12:21:30 a.m.	0	1	0	1

(37 rows affected)

Figure 4.36: Acquired statistics histogram information against Date column of RFID_test.mdf

However, all the table statistics that were updated before the time of investigation must be acquired if the forensic investigator was not sure about which columns were relevant to the investigation.

```
:out E:\ DBO_RFIDdb_tbl.txt
SELECT * FROM RFID_test..rfid_db
GO
```

After collecting the table statistics, the acquisition of current table data by using the following SELECT command was required in order to identify the values that had been changed or deleted within a table. Thus, the result of the current table data (see Figure 4.37) was saved in the designated file of the evidence collection device.

SELECT * FROM RFID_test..rfid_db

Id	Tag	Value	Date
C5CD48FE-7F1B-4F81-8D32-14D4B0ABB43D	E0040000E90B4322	600	10/12/2010 01:29:52 a.m.
4A9FCEA6-7B0A-4440-8DA6-2470DFF6800D	E0040000E90B4329	600	10/12/2010 01:19:31 a.m.
D8034399-C87A-49A1-B771-399BAA60A24B	E0040000E90B4325	600	10/12/2010 01:59:55 a.m.
7B68131E-B5A0-4697-B582-41CB3501FD59	E004000074251502	600	12/10/2010 03:50:53 p.m.
3A346D5A-C35A-469F-9BE2-424F6A0834A2	E0040000E90B4323	600	10/12/2010 01:39:53 a.m.
EB2DC55F-BF9C-4DA8-8988-571BCF299738	E004000074251502	600	12/10/2010 03:50:47 p.m.
31640470-A357-4BC4-9E6C-585717D0A2E8	E004000074251502	600	12/10/2010 03:50:56 p.m.
ABA821B7-BACD-4F43-9C6F-5F5606E1922D	E004000074251502	600	12/10/2010 03:50:43 p.m.
C305B468-0365-4D66-8147-61A0EC92597A	E0040000E90B4201	600	10/12/2010 01:49:53 a.m.
03450243-4D1C-4CC1-909D-64D3652F41B3	E0040000E90B4327	600	10/12/2010 01:39:57 a.m.
8E48C785-3168-4048-B83F-6D9AD3EE4FBA	E0040000E90B4324	600	10/12/2010 01:49:54 a.m.
0A1AA3F5-EFF8-424C-886B-730B9FF81257	E004000074251502	600	12/10/2010 03:50:50 p.m.
C29F801F-CB87-4D69-A5E3-A02B4D4D4537	E0040000E90B4100	600	10/12/2010 01:39:52 a.m.
3B190E8E-E679-495C-9D6A-A08BD4DD92F6	E0040000E90B4321	600	10/12/2010 01:59:59 a.m.
E0C40FB7-E420-4D1F-8284-B5E957C48B6E	E0040000E90A4302	600	01/01/1900 12:00:00 a.m.
11382165-46DB-4EE7-AAD6-EBA770AEEEFF	E0040000E90B4401	600	10/12/2010 01:39:51 a.m.
06AB7F7E-2AD4-4264-9605-EBA93BFA4B95	E0040000E90B4326	600	10/12/2010 01:09:56 a.m.
7F60CEB5-7FEA-49DB-A69F-EDCBD4E6C02E	E0040000E90B4321	600	10/12/2010 01:19:51 a.m.
3C91B6B7-5546-4828-A870-F0B22A3347A5	E0040000E90B4328	600	10/12/2010 01:29:58 a.m.

(19 rows affected)

Figure 4.37: Acquired current table data of RFID_test.mdf database

Likewise, the acquisitions of the *Histogram* data information against the log file (RFID_test_log.ldf) were performed by using the following syntaxes.

```
:out E:\ DBO_RFIDlog_tblStat_Tag.txt
DBCC SHOW_STATISTICS ('RFID_test..rfid_log', 'Tag') WITH HISTOGRAM
GO

:out E:\ DBO_RFIDlog_tblStat_Value.txt
DBCC SHOW_STATISTICS ('RFID_test..rfid_log', 'Value') WITH HISTOGRAM
GO

:out E:\ DBO_RFIDlog_tblStat_Date.txt
DBCC SHOW_STATISTICS ('RFID_test..rfid_log', 'Date') WITH HISTOGRAM
GO
```

After running the preceding syntaxes, the *Histogram* information results (Figure 4.38, 4.39 and 4.40) against the specific columns of interest (RFID_test_log.ldf) were sent to the designated output files of the evidence collection device.

DBCC SHOW_STATISTICS ('RFID_test..rfid_log', 'Tag') WITH HISTOGRAM

RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
E004000074251502	0	30	0	1
E0040000E90A4302	0	19	0	1

(2 rows affected)

Figure 4.38: Acquired statistics histogram information against Tag column of RFID_test_log.ldf

DBCC SHOW_STATISTICS ('RFID_test..rfid_log', 'Value') WITH HISTOGRAM

Msg 2767, Level 16, State 1, Server TEST-STATION\SQLEXPRESS, Line 1
Could not locate statistics 'Value' in the system catalogs.
DBCC execution completed. If DBCC printed error messages, contact your system administrator.

Figure 4.39: Acquired Statistics Histogram Information against Value Column of RFID_test_log.ldf

Once collecting the table statistics of the RFID_test_log.ldf, the acquisition of current table data by using the following SELECT command was also required in order to identify the values that had been changed or deleted within a table.

```
:out E:\ DBO_RFIDlog_tbl.txt
SELECT * FROM RFID_test..rfid_log
GO
```

Thus, the result of the current table data (see Figure 4.41) was saved in the output file (DBO_RFIDlog_tbl.txt) of the evidence collection device.

RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
31/08/2010 10:52:08 p.m.	0	1	0	1
31/08/2010 10:54:07 p.m.	0	1	0	1
31/08/2010 10:55:13 p.m.	0	1	0	1
31/08/2010 10:56:04 p.m.	0	1	0	1
31/08/2010 10:56:27 p.m.	0	1	0	1
31/08/2010 11:40:49 p.m.	0	1	0	1
31/08/2010 11:41:27 p.m.	1	1	1	1
31/08/2010 11:41:59 p.m.	0	1	0	1
31/08/2010 11:43:08 p.m.	0	1	0	1
31/08/2010 11:43:29 p.m.	0	1	0	1
31/08/2010 11:43:39 p.m.	0	1	0	1
31/08/2010 11:44:14 p.m.	1	1	1	1
31/08/2010 11:44:49 p.m.	0	1	0	1
31/08/2010 11:44:53 p.m.	1	1	1	1
31/08/2010 11:48:53 p.m.	0	1	0	1
31/08/2010 11:49:05 p.m.	0	1	0	1
31/08/2010 11:49:08 p.m.	0	1	0	1
31/08/2010 11:49:12 p.m.	0	1	0	1
31/08/2010 11:49:15 p.m.	0	1	0	1
31/08/2010 11:49:26 p.m.	0	1	0	1
31/08/2010 11:49:30 p.m.	0	1	0	1
31/08/2010 11:49:41 p.m.	0	1	0	1
31/08/2010 11:49:44 p.m.	0	1	0	1
31/08/2010 11:49:50 p.m.	0	1	0	1
31/08/2010 11:49:53 p.m.	0	1	0	1
31/08/2010 11:49:54 p.m.	0	1	0	1
31/08/2010 11:49:56 p.m.	0	1	0	1
31/08/2010 11:50:01 p.m.	0	1	0	1
31/08/2010 11:50:04 p.m.	0	1	0	1
31/08/2010 11:50:12 p.m.	0	1	0	1
01/09/2010 12:10:39 a.m.	0	1	0	1
01/09/2010 12:11:36 a.m.	0	1	0	1
01/09/2010 12:11:48 a.m.	0	1	0	1
01/09/2010 12:12:09 a.m.	0	1	0	1
01/09/2010 12:12:25 a.m.	0	1	0	1
01/09/2010 12:12:45 a.m.	1	1	1	1
01/09/2010 12:13:28 a.m.	0	1	0	1
01/09/2010 12:13:45 a.m.	0	1	0	1
01/09/2010 12:14:05 a.m.	1	1	1	1
01/09/2010 12:14:54 a.m.	0	1	0	1

Figure 4.40: Acquired statistics histogram information against Date column of RFID_test_log.ldf

SELECT * FROM RFID_test..rfid_log

Id	Tag	User_data	Date
B9AD9	E0040000E90B4325	insert into rfid_db Tag Value Date values E0040000E90B4325 1700 01:59:55 12/10/2010	10/12/2010 01:59:55 a.m.
2DEF0(	E0040000E90B4327	insert into rfid_db Tag Value Date values E0040000E90B4327 1100 01:39:57 12/10/2010	10/12/2010 01:39:57 a.m.
3FBB34	E00400007425150	Insert into rfid_db Tag Value Date values E00400007425150 1500 2010-10-12 15:50:43	12/10/2010 03:50:43 p.m.
54FFFC	E0040000E90B4326	insert into rfid_db Tag Value Date values E0040000E90B4326 1600 01:09:56 12/10/2010	10/12/2010 01:09:56 a.m.
C0C8FA	E0040000E90B4401	insert into rfid_db Tag Value Date values E0040000E90B4401 700 01:39:51 12/10/2010	10/12/2010 01:39:51 a.m.
839D22	E0040000E90B4323	insert into rfid_db Tag Value Date values E0040000E90B4323 1200 01:39:53 12/10/2010	10/12/2010 01:39:53 a.m.
AFDFA	E00400007425150	Insert into rfid_db Tag Value Date values E00400007425150 1500 2010-10-12 15:50:56	12/10/2010 03:50:56 p.m.
CDE62(	E00400007425150	Insert into rfid_db Tag Value Date values E00400007425150 1500 2010-10-12 15:50:47	12/10/2010 03:50:47 p.m.
31E72C	E0040000E90B4324	insert into rfid_db Tag Value Date values E0040000E90B4324 1500 01:49:54 12/10/2010	10/12/2010 01:49:54 a.m.
6745F0	E0040000E90B4329	insert into rfid_db Tag Value Date values E0040000E90B4329 600 01:19:31 12/10/2010	10/12/2010 01:19:31 a.m.
6D57A(	E0040000E90B4328	insert into rfid_db Tag Value Date values E0040000E90B4328 700 01:29:58 12/10/2010	10/12/2010 01:29:58 a.m.
5BF84F	E0040000E90B4321	insert into rfid_db Tag Value Date values E0040000E90B4321 1700 01:19:51 12/10/2010	10/12/2010 01:19:51 a.m.
8926DE	E0040000E90A4302	Insert into rfid_db Tag Value Date values E0040000E90A4302 update rfid_db set Value=600 where Tag > E004% -- 2010-10-12 18:39:48	12/10/2010 06:39:48 p.m.
F4F909	E0040000E90B4321	insert into rfid_db Tag Value Date values E0040000E90B4321 1700 01:59:59 12/10/2010	10/12/2010 01:59:59 a.m.
99F0DA	E00400007425150	Insert into rfid_db Tag Value Date values E00400007425150 1500 2010-10-12 15:50:50	12/10/2010 03:50:50 p.m.
A49AE(	E0040000E90B4201	insert into rfid_db Tag Value Date values E0040000E90B4201 1700 01:49:53 12/10/2010	10/12/2010 01:49:53 a.m.
0A329(	E0040000E90B4322	insert into rfid_db Tag Value Date values E0040000E90B4322 1400 01:29:52 12/10/2010	10/12/2010 01:29:52 a.m.
74DA3(	E00400007425150	Insert into rfid_db Tag Value Date values E00400007425150 1500 2010-10-12 15:50:53	12/10/2010 03:50:53 p.m.
E2D7B(	E0040000E90B4100	insert into rfid_db Tag Value Date values E0040000E90B4100 1700 01:39:52 12/10/2010	10/12/2010 01:39:52 a.m.

Figure 4.41: Acquired current table data of RFID_test_log.ldf database

4.2.3.4.5 Collation Settings and Data Types

The way in which data is stored and processed within the SQL Server is determined by the collation settings and data types. Hence, the language and format of the data within the databases of the SQL Server can be identified by gathering *collation settings and data types* in use within the databases and the findings of these *collation settings and data types* will be very useful in analyzing the acquired data such as the transaction log and the like (Fowler, 2009).

Thus, the following syntax, *sys.cloumns* view query run on the victim's backend SQL Server 2005, was used to collect a list of collation and data type used by an individual column within a database.

```
:out E:\CollationAndDataTypes.txt
SELECT * From sys.syscolumns
GO
```

The result of the preceding query (Figure 4.42) was made up of the different fields including *name* (column name), *id* (the object identifier of the table to which the column belongs), *xusertype* (type of the data stored within the column) and *length* (the length of the data within the column).

SELECT * FROM sys.syscolumns

name	id	xtype	typestat	xusertype	length	xprec	xscale	colid	xoffset	bitpos	reserved	colstat
rowsetid	4	127	1	127	8	19	0	1	0	0	0	0
rowsetcolid	4	56	1	56	4	10	0	2	0	0	0	0
hobtcolid	4	56	1	56	4	10	0	3	0	0	0	0
status	4	56	1	56	4	10	0	4	0	0	0	0
rcmodified	4	127	1	127	8	19	0	5	0	0	0	0
maxinrowlen	4	52	1	52	2	5	0	6	0	0	0	0
rowsetid	5	127	1	127	8	19	0	1	0	0	0	0
ownertype	5	48	1	48	1	3	0	2	0	0	0	0
idmajor	5	56	1	56	4	10	0	3	0	0	0	0
idminor	5	56	1	56	4	10	0	4	0	0	0	0
numpart	5	56	1	56	4	10	0	5	0	0	0	0
status	5	56	1	56	4	10	0	6	0	0	0	0
fgidfs	5	52	1	52	2	5	0	7	0	0	0	0
rcrows	5	127	1	127	8	19	0	8	0	0	0	0
auid	7	127	1	127	8	19	0	1	0	0	0	0
type	7	48	1	48	1	3	0	2	0	0	0	0
ownerid	7	127	1	127	8	19	0	3	0	0	0	0
status	7	56	1	56	4	10	0	4	0	0	0	0

Figure 4.42: Fragment of the acquired collation settings and data types of backend SQL Server 2005

4.2.3.4.6 Data Page Allocation Artefact

Every database can consist of several thousands of pages of data. The SQL Server 2005 manages the database by assigning an object identification (ID) to each database object and assigns also page ID for the data belonging to the object (Fowler, 2009).

Thus, the acquisition of *data page allocations* used by the databases within the SQL Server was required as the several artefacts like the *data cache* and the transaction log reference the page identifications (IDs). In order to acquire the *data page allocations*, the following script was run from a trusted SQLCMD session.

```
:out E:\DataPageAllocations.txt
:r E:\Scripts\DpgAlloc.sql
GO
```

The output of the above mention script (see Figure 4.43) returned the database ID, object ID, page IDs and the like.

Database	DatabaseID	Object	ObjectID	PageFID	PagePID
msdb	4	dbo.MSdbms_datatype	1	172	1109578991
msdb	4	dbo.sysmail_servertype	1	415	1134627085
msdb	4	dbo.MSdbms_map	1	178	1173579219
msdb	4	dbo.MSdbms_map	1	183	1173579219
msdb	4	dbo.MSdbms_map	1	186	1173579219
msdb	4	dbo.MSdbms_datatype_mapping	1	180	1381579960
msdb	4	dbo.MSdbms_datatype_mapping	1	185	1381579960
msdb	4	dbo.MSdbms_datatype_mapping	1	187	1381579960
msdb	4	dbo.sysmail_configuration	1	413	1390627997
msdb	4	dbo.backupmediaset	1	545	1858105660
msdb	4	dbo.backupmediafamily	1	549	1890105774
msdb	4	dbo.backupset	1	569	1954106002
msdb	4	dbo.sysdtspackagefolders90	1	522	1979154096
msdb	4	dbo.backupfilegroup	1	573	2002106173
msdb	4	dbo.backupfile	1	575	2050106344
msdb	4	dbo.restorehistory	1	577	2098106515
msdb	4	dbo.restorefile	1	581	2146106686
RFID_test	5	dbo.rfid_db	1	157	165575628
RFID_test	5	dbo.rfid_log	1	158	213575799
(42 rows affected)					

Figure 4.43: Fragment of the acquired data page allocations of backend SQL Server 2005

4.2.3.4.7 Server Hardening Information

The identification of security weakness or hardening of compromised backend server could also give support to forensic investigator for determining the scope and areas of interest in investigation.

According to Fowler (2009, p. 261), "*one of the most well-known SQL Server misconfigurations is the use of blank passwords and logins with blank passwords (example: SA login) are an attractive target for an attacker trying to gain access to a SQL Server instance*".

Thus, all the backend server logins were also collected by using automated WFT from Helix_RFID_IR toolkit. The result of the collected logins could be seen in the following Figure (4.44).

name	createdate	accdate	sysadmin	sid
vishal	03/10/2010 01:05:58 a.m.	03/10/2010 01:05:58 a.m.	0	0x340609DBAAFD3F48BD106F015F9EE4AE
arkar	01/10/2010 11:17:18 p.m.	01/10/2010 11:17:18 p.m.	1	0xF0256F3798393E448D01328C13E1479D
sa	08/04/2003 09:10:35 a.m.	08/04/2003 09:10:35 a.m.	1	0x01
BUILTIN\Users	17/07/2010 12:17:56 a.m.	17/07/2010 12:17:56 a.m.	0	0x0102000000000005200000021020000
BUILTIN\Administrators	17/07/2010 12:17:56 a.m.	17/07/2010 12:17:56 a.m.	1	0x0102000000000005200000020020000

Figure 4.44: Collected SQL Server logins

Furthermore, an investigation on security configurations of the SQL Server was also significant in order to find out whether the malicious hacker or unauthorized user had gained access to the backend database by using the weak point of the security configuration. However, the database attack could be minimized by server administrators by using a SQL Server 2005 Surface Area Configuration (SAC) tool. The Surface Area Configuration (SAC) tool was basically introduced in 2005 and later versions of SQL Servers in order to simply disable the vulnerable features of SQL Servers for hardening the security. By using SAC tool, the administrators could disable unnecessary database features or functionalities such as xp_cmdshell, SQL Server browser, remote connection and the like. Similar to the configuration settings of the SQL Server, SAC could give the forensic examiner the information about which

components of database were enabled or disabled. Hence, the following syntax (Figure 4.45) was used for the acquisition of SAC information, the security hardening status of the SQL Server 2005.

```
:out E\:ScSAC.txt
SELECT * FROM sys.system_components_surface_area_configuration
GO
```

Figure 4.45: Syntax used for the acquisition of SAC information

The output result of acquired data was as shown in the Figure 4.46 below and the result should be checked against the default status of SAC objects documented by Microsoft (http://msdn.microsoft.com/en-us/library/ms183753.aspx) to identify whether the malicious hacker or unauthorized user had used such settings to compromise the SQL Server.

component_name	database_name	schema_name	object_name	state
SQL Mail XPs	mssqlsystemresource	sys	xp_get_mapi_default_profile	0
Database Mail XPs	mssqlsystemresource	sys	xp_sysmail_format_query	0
SMO and DMO XPs	mssqlsystemresource	sys	xp_regread	1
Agent XPs	mssqlsystemresource	sys	xp_regread	0
SMO and DMO XPs	mssqlsystemresource	sys	xp_subdirs	1
Ole Automation Procedures	mssqlsystemresource	sys	sp_OASetProperty	0
SQL Mail XPs	mssqlsystemresource	sys	xp_test_mapi_profile	0
Agent XPs	mssqlsystemresource	sys	xp_sqlagent_param	0
Web Assistant Procedures	mssqlsystemresource	sys	xp_cleanupwebtask	0
Web Assistant Procedures	mssqlsystemresource	sys	xp_dropwebtask	0

Figure 4.46: Surface Area Configuration results by running sys.system_components_surface_area_configuration

4.2.3.5 Record of SQLCMD Disconnection and SQL Server Service Shutdown

Once the acquisition of all volatile and non-volatile data from the running SQL Server, the residual artefacts could be collected while the SQL Server instance was shut down. However, it was significant to note that the all volatile data should be acquired before stopping the SQL Server instance. Similarly, the

client or the owner of the retail shop should be aware of and coordinate the actions taken by the forensic investigator before shutting down the service. Furthermore, it was also important to keep in mind that most volatile data would stay in the memory or local temporary files as long as the SQL Server service was running during the investigation, whereas some volatile data would remain within the memory until the system was powered down (Fowler, 2009).

Mostly importantly, the investigator must record the details information regarding connection session to the backend database SQL Server instance before shutting down the Server instance. In order to record the connection session details, the following syntax was used.

```
:out E:\ConnectionDetails.txt
SELECT 'User: ' + suser_sname()
  + ' | SPID: ' + CAST(@@SPID  AS VARCHAR)
  + ' | Connection time: ' + CAST(sp.login_time  AS VARCHAR)
  + ' | Disconnection time: ' + CAST(GETDATE()  AS VARCHAR(20))
from master..sysprocesses sp where SPID = @@SPID
GO
```

The output after running the above mentioned syntax included user name, service process identification (SPID), connection date and timestamp, and disconnection date and timestamp (see Figure 4.47).

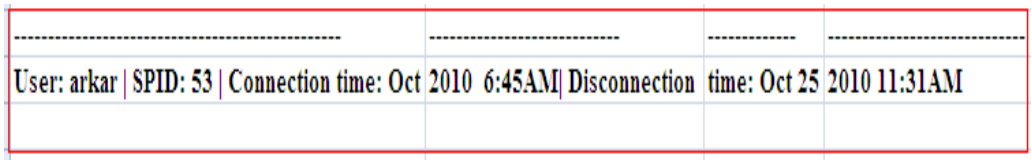


Figure 4.47: SQL Server connection details of the forensic investigator

After recording the connection session details, the SQL Server service was shutdown to collect the residual non-volatile server artefacts.

```
:out E:\ServiceShutDown.txt
SHUTDOWN
GO
```

The record of shutting down the server service was recorded in the output file (E:\ServiceShutDown.txt) within the acquisition device as "*Server shut down by request from login arkar*".

4.2.3.6 Residual Non-Volatile SQL Server Data

The residual non-volatile SQL Server data were also acquired during the investigation. These included the acquisitions of physical data file, the reusable virtual log files (VLFs) command language runtime (CLR) libraries, trace file, the SQL Server error logs and system event logs by using *dcfldd tool* from the forensic USB toolkit which was attached to the compromised system via a hardware write blocker (*Tableau Forensic USB Bridge*).

4.2.3.6.1 Data Files

The deleted data from the data file of the SQL Server could be either partially or completely retrieved as the traces of previously deleted data normally store within multiple areas of a database (Fowler, 2009). To acquire the physical database files, the exact locations of the database files were needed to find out firstly. These database files locations could be found in the result of previously executed SSFA_Databases.sql IR script (Figure 4.48).

database_id	file_id	database	type_desc	file_name	physical_name
1	1	master	ROWS	master	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\master.mdf
1	2	master	LOG	mastlog	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\mastlog.ldf
2	1	tempdb	ROWS	tempdev	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\tempdb.mdf
2	2	tempdb	LOG	templog	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\templog.ldf
3	1	model	ROWS	modeldev	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\model.mdf
3	2	model	LOG	modellog	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\modellog.ldf
4	1	msdb	ROWS	MSDBData	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\MSDBData.mdf
4	2	msdb	LOG	MSDBLog	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\MSDBLog.ldf
5	1	RFID_test	ROWS	RFID_test	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\RFID_test.mdf
5	2	RFID_test	LOG	RFID_test_log	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\RFID_test_log.ldf

Figure 4.48: Physical data file locations from the result of SSFA_Databases.sql script

According to the above mentioned Figure 4.48, the physical locations of FRID_test database and transaction log data files were determined. The

acquisition of these data files were done by using *dcfldd tool*. For acquiring RFID_test.mdf data file, the following syntax was used.

```
dcfldd conv= noerror if="C:\Program Files\Microsoft SQL
Server\MSSQL.1\MSSQL\DATA\RFID_test.mdf" of=E:\RFID_test.mdf hash=md5
hashlog=E:\RFID_test.md5
```

Likewise, the acquisition of RFID_test_log.ldf data file was done by using the following syntax.

```
dcfldd conv= noerror if="C:\Program Files\Microsoft SQL
Server\MSSQL.1\MSSQL\DATA\RFID_test_log.ldf" of=E:\RFID_test_log.ldf hash=md5
hashlog=E:\RFID_test_log.md5
```

The above mentioned scripts sent the image copies of RFID_test.mdf database and RFID_test_log.ldf data files to the evidence collection drive (E:\). The outputs also produced MD5 hashes of the source and image files in order to ensure whether the imaging of data files are done successfully by comparing the MD5 hashes.

4.2.3.6.2 Reusable VLFs Artefact

In the previous Section 4.2.3.3, the active VLF data and the status of VLFs within the transaction log of the database were acquired. Hence, the reusable VLFs was left to acquire for data recovery purpose. According to Fowler (2009), any status code from the output/result of the VLF status rather than the value (2) was referred to as reusable VLF. However, the entire transaction log file was collected (see Section 4.2.3.6.1). In fact, there was no forensic tool available for the collection of data from the reusable VLF areas. The *StartOffset* column values from the result of *DBCC loginfo* (Figure 4.10 in Section 4.2.3.3) represented where the offset of each VLF begins and that offset could be used to perform not only "*log carving on the reusable VLF regions*", but also to "*extract previously inserted or deleted data records*" (Fowler, 2009, p. 216).

4.2.3.6.3 Command Language Runtime (CLR) Libraries

Fowler (2009, p. 217) stated that the CLR libraries supported by the SQL Server 2005 (or SQL Server 2008) were user-created executables that contained

"*application logic and extend the native functionality of the SQL server*". Hence, the acquisition of CLR libraries was needed for reconstruction of activity during the investigation.

However, there were no registered CLR libraries according to the output of the executed SSFA_CLR.sql script in this research experiment (see Figure 4.49). In contrast, the acquisition of the registered CLR libraries within the SQL server instance of interest in investigation could be done by using *dcfldd* utility if any registered CLR library was found in the output of the executed SSFA_CLR.sql script.

database	name	create_date	modify_date	permission_set_desc	file
(0 rows affected)					
database	name	create_date	modify_date	permission_set_desc	file
(0 rows affected)					
database	name	create_date	modify_date	permission_set_desc	file
(0 rows affected)					
database	name	create_date	modify_date	permission_set_desc	file
(0 rows affected)					
database	name	create_date	modify_date	permission_set_desc	file
(0 rows affected)					

Figure 4.49: Registered CLR libraries from the output of SSFA_CLR.sql script

4.2.3.6.4 Trace Files Artefact

In general, a new trace file was generated whenever the SQL Server database engine was started. There are total 5 trace files provided by a SQL Server 2005 (or 2008) by default. These trace files are useful for the reconstruction of event/activity and can be found in the location of C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG directory. In fact, the selected trace files

which contain data within the investigation scope can be acquired by determining their modification dates and times. Initially, the last modification dates of the trace files were determined (as shown in Figure 4.50) by using trusted binaries from HELIX_RFID_IR tool.

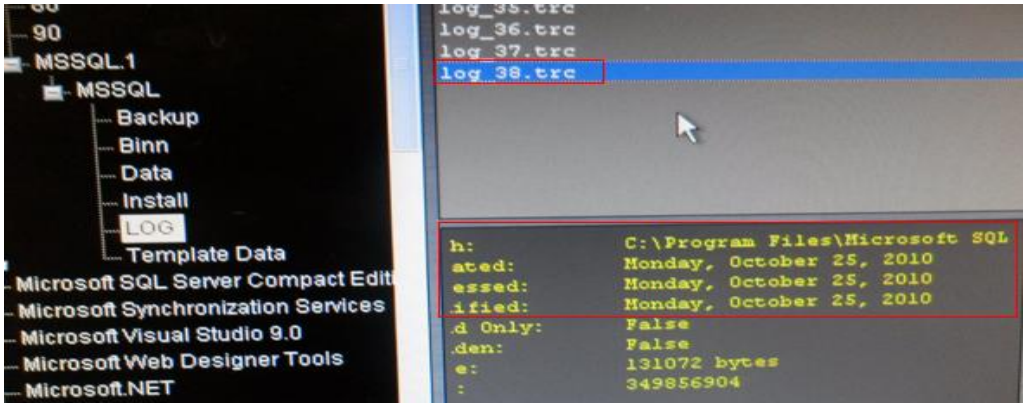


Figure 4.50: Screenshot of the last modification dates of a trace file

Then, the trace files (log_34.trc, log_35.trc, log_36.trc, log_37.trc and log_38.trc) were acquired from the forensic image copy of the victim system in order to prevent the changes in the last accessed times of the trace files (Fowler, 2009). Hence, the acquisitions of all trace files were done by using *dcfldd* utility as shown in the figure below.

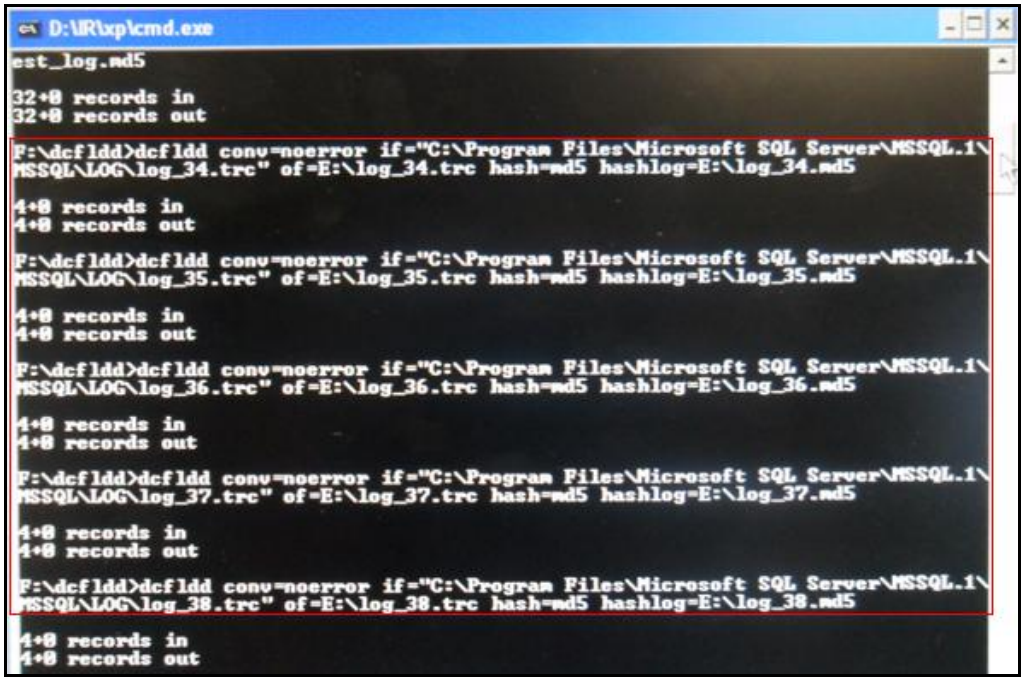


Figure 4.51: Acquisitions of the trace files using *dcfldd tool*

4.2.3.6.5 SQL Server Error Logs

Usually, the SQL Server 2005 preserves seven error logs which are useful for determining the information related to authentication, whether successful or failed login attempts to the SQL Server. By default, the server error logs were located in the same directory as that of trace files (C:\Program Files\Microsoft SQL Server\MSSQL.1\LOG) and the current error log was acquired by using the following syntax:

```
dcfldd conv= noerror if="C:\Program Files\Microsoft SQL
Server\MSSQL.1\MSSQL\LOG\ERRORLOG" of=E:\errorlog hash=md5 hashlog=E:\
errorlog.md5
```

Similarly, the rest of the error logs were also acquired by using *dcfldd tool* for investigation purpose (Figure 4.52)

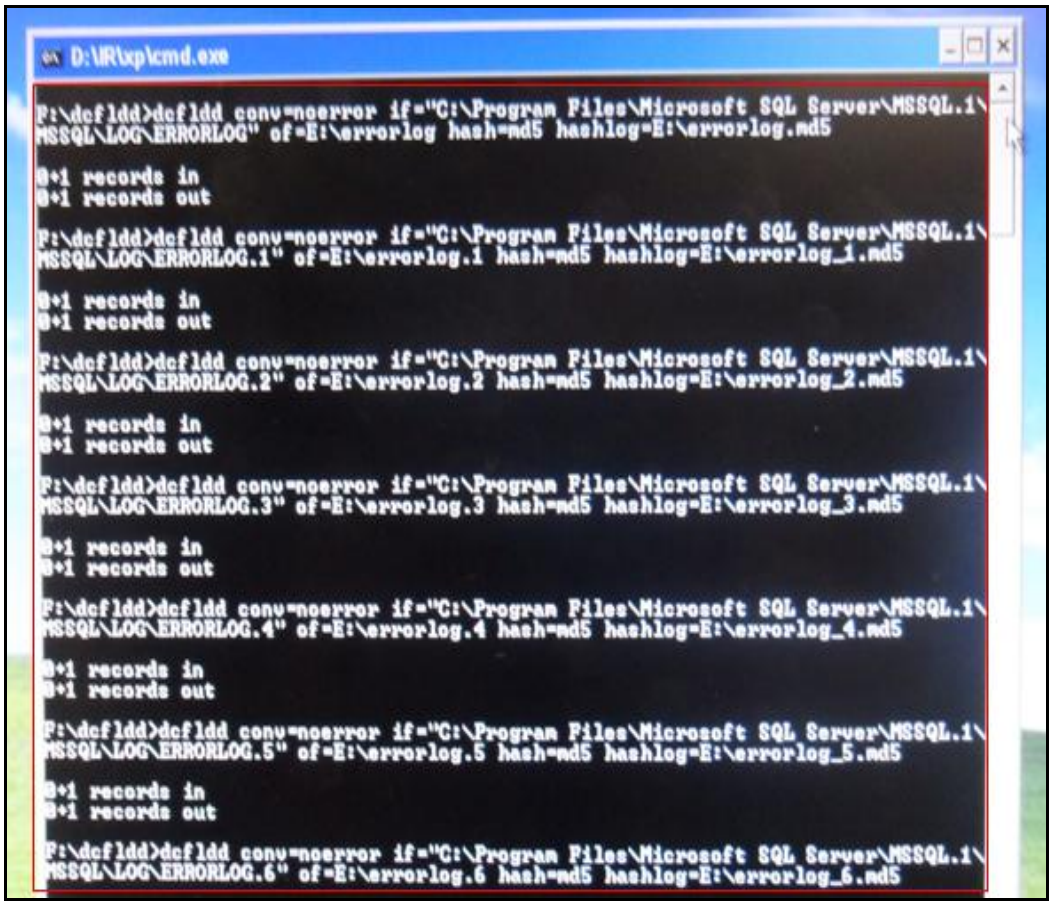


Figure 4.52: Acquisitions of backend SQL Server error logs by using *dcfldd tool*

4.2.3.6.6 System Event Logs

In addition to the SQL Server error and trace file logs, Windows XP (test station) system maintain system, application and security logs that can be related to the investigation. By default, these system event logs were located in *C:\WINDOWS\system32\config* directory in Windows XP and could be acquired by using either *psloglist.exe* utility from customized HELIX_RFID_IR DVD or *dcfldd* to acquire the physical event log files. Hence, the system event logs were extracted by using the following syntax to forward the outputs to the evidence collection drive (E:\).

```
psloglist.exe –s Application >> E:\ApplicationLog.txt
psloglist.exe –s System >> E:\SystemLog.txt
psloglist.exe –s Security >> E:\SecurityLog.txt
```

Hence, the argument –s was used in the syntax for delimiting output, which could help to analyze the logs simply later in the analysis phase of the investigation.

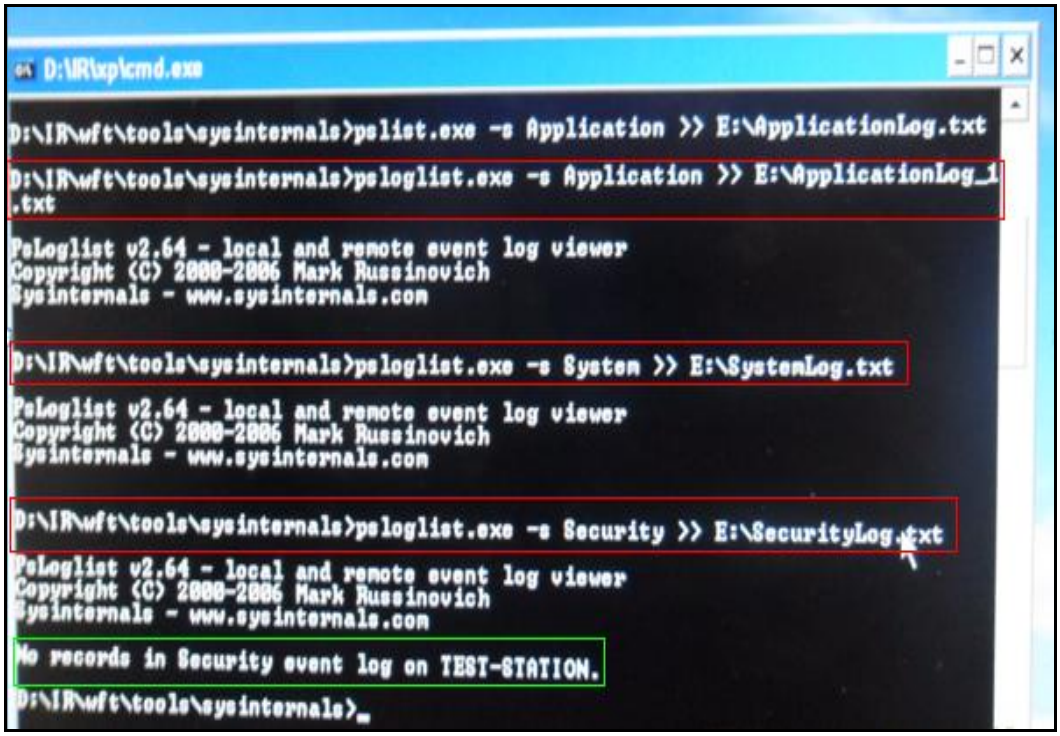


Figure 4.53: Acquisitions of application, system and security error logs by using psloglist.exe

4.2.4 Physical Disk Drive

Finally, the bit-to-bit live acquisition of POS/Server's physical hard drive (see Appendix 17) was performed by using AccessData's Forensic Tool (FTK Imager Lite - Version 2.9.0.1358). The FTK Imager, which was located in the USB forensic flash drive (F:\), was run from the trusted command prompt as shown in the Figure 4.54.

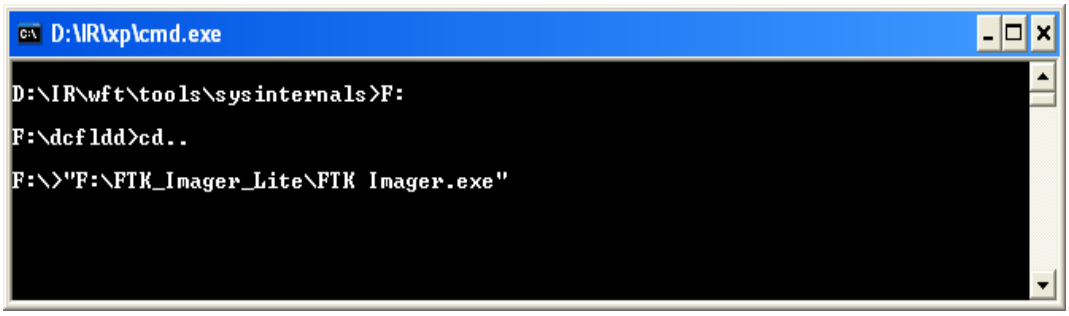


Figure 4.54: FTK Imager was run from USB forensic flash drive during the acquisition

In order to conclude Section 4.2, the acquisitions and preservations of volatile and non-volatile SQL backend server evidence artefacts were done by using the tools (such as *dcfldd*) and customized WFT included in the Helix_RFID_IR toolkit in a forensically sound manner. All SQL Server data collected by WFT could be found in the Appendix 13. Likewise, the acquisition of bit-to-bit data from RFID reader's log (Section 4.2.2) and RAM of POS (Section 4.2.1) were performed by using LET (LogExtraction.exe) and WinEn tools from Helix_RFID_IR toolkit, accordingly. These collected data were preserved and processed as stated in Sections (3.3.4 and 3.3.5) before analyzing. Hence, the analysis of the collected data could be found in the following section.

4.3 ANALYSIS OF THE COLLECTED DATA

As mentioned in the previous chapter (Sections 3.3.5 and 3.3.6), in order to perform analysis of the collected data in a forensically sound manner; there were some procedures needed to follow. Before performing the analysis, the collected data must be firstly imaged and hashed. The image copies of the collected data were only utilized during the analysis phase and the hash values of the collected data were used for the verification of the data integrity. Hence, the original evidence data could be preserved and the forensic investigator can avoid the alteration of any original

evidence data during analysis. Likewise, the use of write blockers during analysis could prevent changes to the original acquired data. Thus, the Tableau forensic hardware write blocker was used to connect to the forensic workstation via USB connection during analysis in order to maintain the preservation of the collected data.

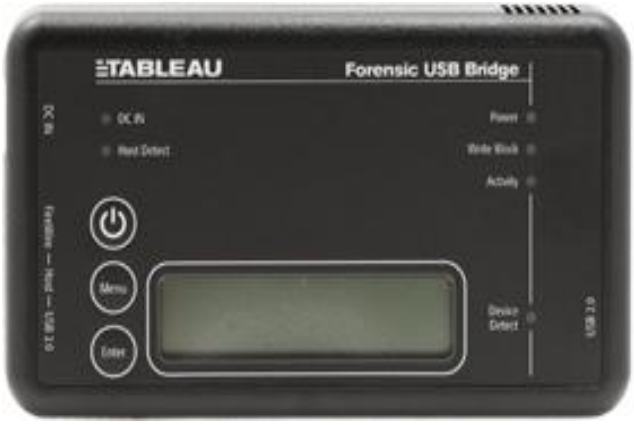


Figure 4.55: Tableau forensic USB bridge
(http://www.tableau.com/images/products/t8_r2_front_hi_res.jpg)

The data analysis was performed by using two different forensic test-stations in the laboratory. One of the forensic test-stations was installed with EnCase for imaging the collected data, and analysis of RFID Reader's memory, POS station's memory, and the physical drive image. The second forensic test-station was installed with a SQL Server 2005 and used to analyze the collected SQL artefacts.

4.3.1 Analysis of RFID Reader's Memory

After acquiring the bit-to-bit image of the logs from RFID reader's memory (Section 4.2.2), its image copy on the forensic test-station was analyzed by using EnCase Forensic Training software. As stated in Section 3.3.2, the fake tag (ID: E0040000E90A4302) was used as a keyword to look for the malicious transaction in the image copy when analyzing. Hence, the notable evidence of the malicious transaction was found along with the tag ID, date and timestamp (as shown in the Figure 4.56).

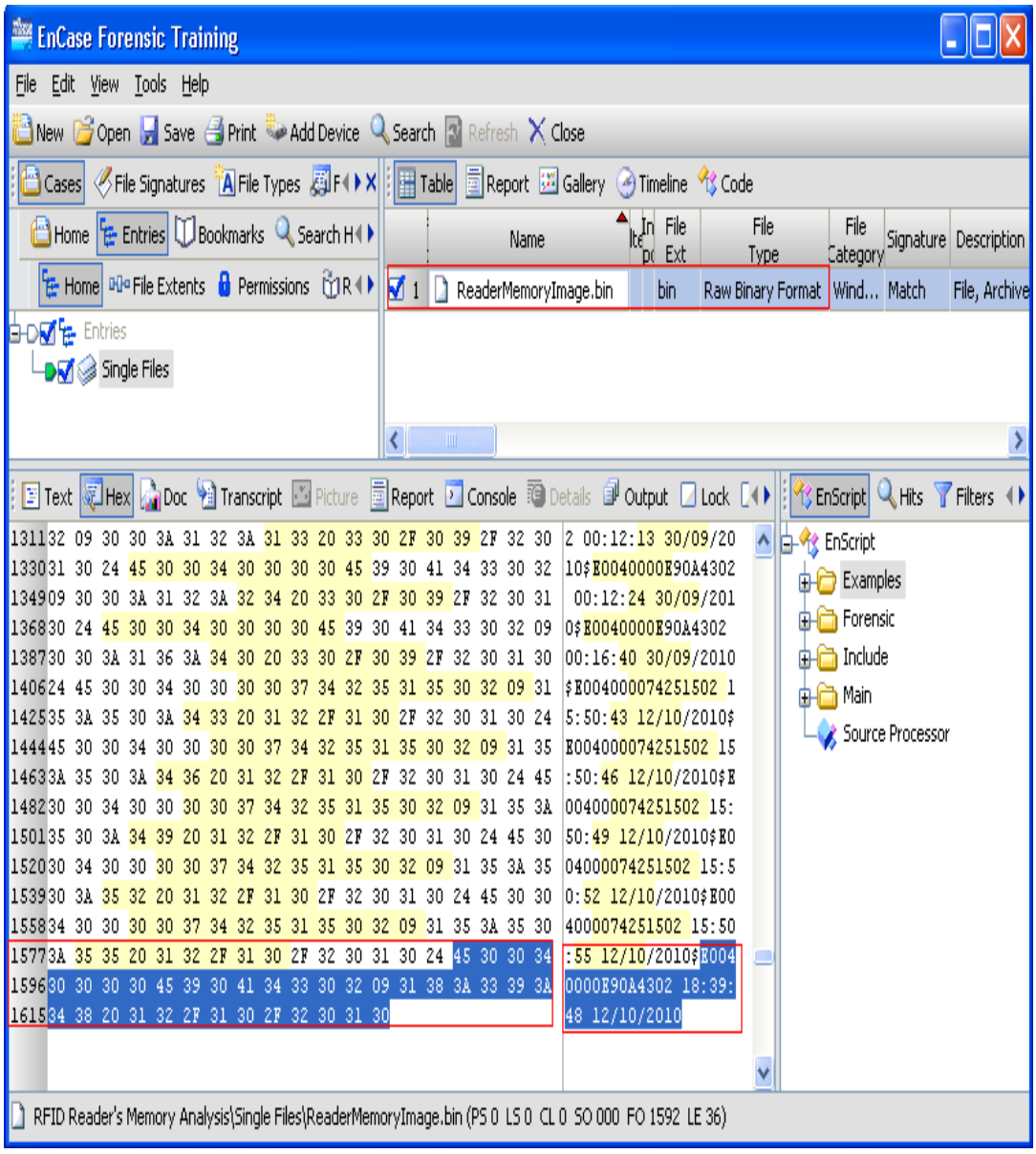


Figure 4.56: Analyzing the image copy of RFID reader's memory with EnCase

4.3.2. Analysis of POS Station's Memory (RAM)

Similar to the analysis method used in Section 4.3.1, the acquired image copy of POS RAM was analyzed by using EnCase. The keyword, fake tag (ID: E0040000E90A4302), was applied during analysis.

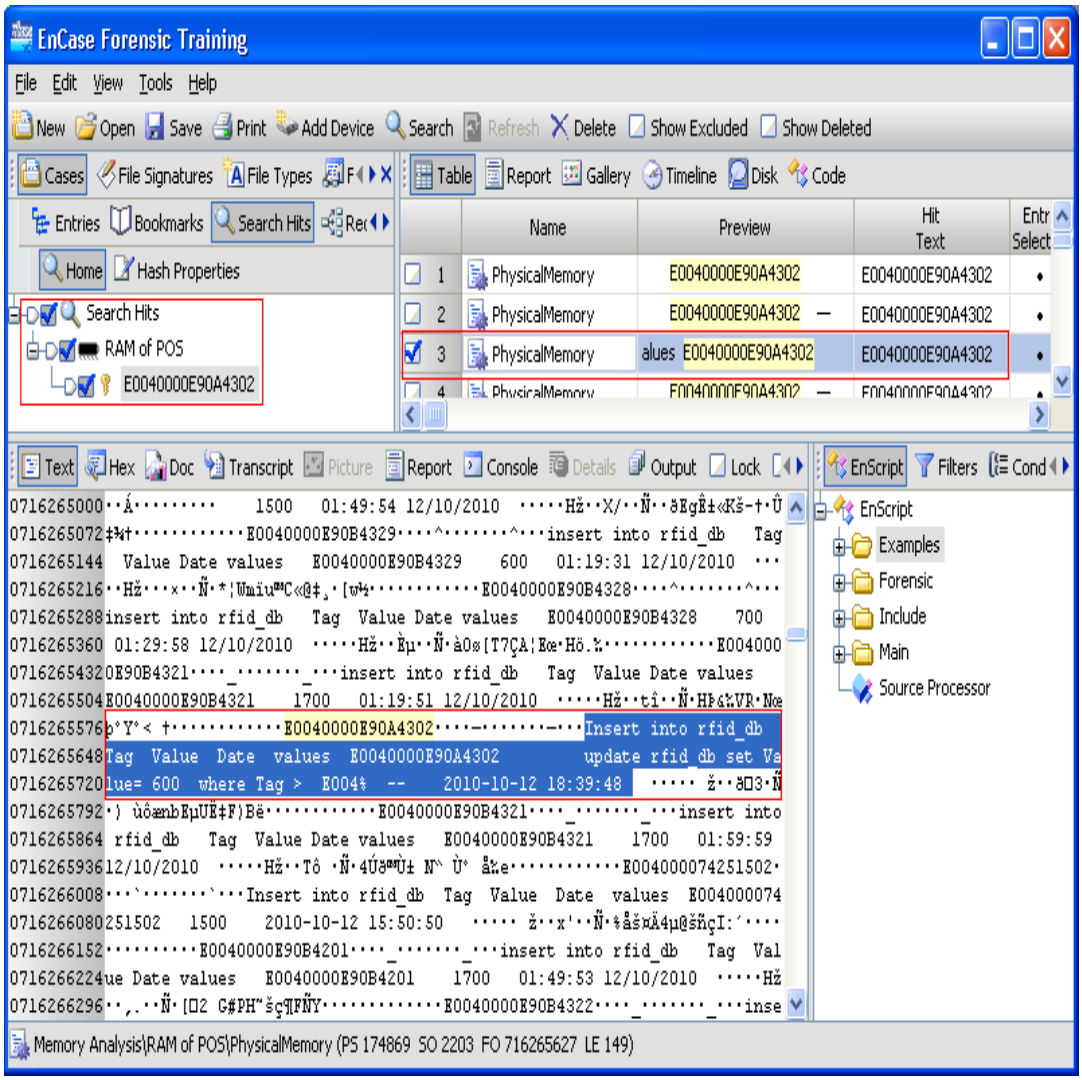


Figure 4.57: Analyzing the image copy of POS RAM with EnCase

The notable evidence of the malicious transaction was also uncovered in the POS RAM image along with the tag ID, date and timestamp (as shown in the Figure 4.56). However, unlike the evidence found in analyzing reader's memory; the analysis result of POS RAM provided the malicious (SQL poisoning) code in addition to fake tag ID, date and timestamp of when the attack was initiated to compromise the backend server of RFID based retail system.

4.3.3 Analysis of SQL Server Artefacts

As previous stated in the report of collected data (Section 4.2.3), some of the collected SQL Server artefacts could be analyzed by using a text viewer, but others

could not. Hence, the artefacts could not be analyzed by using simple text viewers were put into a database of a forensic test-station, on which the SQL Server 2005 was also installed, for analysis. However, the significant findings related to malicious attack after analyzing all acquired SQL artefacts (see Appendix 24) were stated as follow:

Table statistics were useful for comparing against the present state of data within a table to identify the values that have been changed or updated, as stated in Section 4.2.3.4.4. Hence, the last updated time of the table statistics was critical during the forensic data acquisition. However, according to the statistic information findings from the results of acquired Histogram (actual data values that were taken when the last statistics updated) against RFID Tag (Figure 4.34), Value (Figure 4.35) and Date (Figure 4.36) columns of the compromised database (RFID_test.mdf) in Section 4.2.3.4.4; the table statistics were updated before the time of investigation. Hence, the notable result of acquired current table data of the stock management database (Figure 4.37) showed that all the SI values were $600 and confirmed the database was obviously compromised.

Moreover, the acquisition results of current table data of log file - RFID_test_log.ldf (see Figure 4.41) showed the significant evidence of the attack in which the malicious SQL poisoning code was found and the SI (Tag IDs starting with E004) were updated to the values $600 at 06:39:48pm on the 12 October 2010.

4.3.4 Analysis of the Physical Drive Image

The analysis of the physical drive image was not actually required during the investigation. The image of POS/Server TEST-STATION physical drive was acquired (Section 4.2.4) was acquired only for the purpose of preservation.

4.3.5 Analysis of RFID Tags in the Repository

The fictitious tag could be found along with other tags in the repository if the investigation was started before the tags from the sold items in the retail shop were not reused or re-written with the new values within a certain period of time due to the policy of the retail shop. For instance, the tags from the sold items today should not

be re-written for other items within 7 days. Thus, by scanning the tags in the repository; the investigator could find the physical evidence of the fake tag (injected with SQL poisoning code) used by the thief/perpetrator.

4.3.6 Analysis of CCTV Images and Interviews with the Employees

According to the timestamps acquired from analyzing the collected data above, the investigator could further perform the forensic investigation of the incident by analyzing the recorded images of the CCTV (Closed Circuit Television) monitoring system in the retail shop. Likewise, the forensic investigator could interview the employees who were on duty on the day of the incident occurred in the shop to get more information leading to the investigation and reconstruction of the activity in real time.

However, the analysis of CCTV images and interviewing the employees were not in the research scope as previously stated in the limitation section of Chapter 3 (Section 3.4).

4.4 CONCLUSION

This Chapter 4 has reported the findings from the laboratory testing of a business system that was designed to simulate the use of RFID tags in a retail shop. The results show that the extraction of all evidence left after a security breach is complex, time consuming and requires the use of an extended set of Digital Forensic tools.

Nevertheless, the research reports the precise expectations an investigator may have for undertaking such an investigation. The next Chapter (5) will now discuss these findings in relation to the research question, sub questions and hypotheses.

Chapter 5 -

Research Discussion

5.0 INTRODUCTION

In Chapter 4, the findings from the research experiment were reported. The changes made on the proposed data requirements (see Section 3.3), referred to as variations encountered, during the experiment were described in Section 4.1. After performing a number of pilot tests on every phase of the research (for instance; see Appendix 16 for pilot experiment on SQL poisoning attack), the final experiment was conducted in order to answer the selected research questions according to the research methodology developed in Chapter 3.

The significant findings from the research experiment conducted include the malicious SQL poisoning code, a malicious tag's ID, timestamp when the attack occurred, and the values of the current stock items in the backend SQL Server. Moreover, the hash values of all the collected evidence before and after the analysis (see Appendix 22) were exactly the same. Thus the preservation and integrity of the evidence were maintained during the experiment. The purpose of a research methodology and then conducting the forensic investigation was to investigate the theft of stock item (SI) in a RFID based retail shop. The findings after the forensic examination were able to prove the theft of SI in a simulated RFID based stock management system.

Therefore, Chapter 5 will discuss the findings of the research (Section 4.1) in order to evaluate the importance of the results. The finding results evaluation will be discussed in association with the discipline area. In addition, the developed research questions stated in Section 3.2.6 will be answered and discussed in relation to the asserted hypotheses (Section 3.2.7) in Section 5.1. The discussion summaries will be described according to the evidential arguments made, for and against. Subsequently, the discussion of the findings of the research experiment will be presented in Section 5.2 in order to comprehensively evaluate the reported outcomes. Then the recommendations (Section 5.3) will be drawn according to

the knowledge acquired from the research report, followed by the conclusion of the chapter in Section 5.4.

5.1 DISCUSSION OF RESEARCH QUESTION

According to the literature review (Chapter 2) and the review of similar published work in Chapter 3 (Section 3.1), the main research question and secondary research questions (Section 3.2.6) were developed.

In this section, the research questions will be answered by extracting the evidence from the chapter 4 report and using a table format for evaluation. Each table format will include the question asked along with the asserted hypothesis (Section 3.2.7), in which the knowledge acquired from the review of the literature will be briefly explained. Evidence will be tabulated for and against the hypotheses and then a judgement made. The outcome can be accept, reject, or insufficient evidence.

The discussion in the table will be a comparison of the knowledge gained from literature and the findings of the research experiment phases (Chapter 4). The *arguments for* and the *arguments against* the hypotheses made will be presented in order to prove or refute the asserted hypotheses. To validate the report of the arguments presented the references of specific findings will be used. In conclusion of each table discussion, a brief summary of the research question and tested hypothesis will be made based on the research findings achieved in Chapter 4.

5.1.1 Main Research Question and Associated Hypotheses

The main research question was derived in research methodology (Chapter 3) in order to provide a specific objective for phases of research experiment concerning digital forensic investigation (Section 3.2.5). The afore-mentioned main research question was:

> *What should the forensic investigator do in order to perform the complete and accurate forensic examination of a compromised RFID stock management system in the retail sector?*

To answer the main research question proposed in Section 3.2.6, a number of phases of research experiments were proposed and conducted. The stabilized system design testing environment was created after performing pilot tests within

the laboratory to simulate a RFID tag retail stock inventory system (see Section 2.3) with stock item (SI), point-of-sale (POS), and business information system entities (BIS). Hence, the main research question, the associated hypotheses, arguments for and against, and a brief summary of the tested hypotheses will be presented in Tables 5.1 - 5.6.

5.1.2 Secondary Research Question and Associated Hypotheses

As stated in Section 3.2.6, a total of 5 secondary research questions were derived in order to inquire into related areas of concern and hence provide greater depth when answering the main research question.

The following tables (Table 5.1, Table 5.2, Table 5.3, Table 5.4, and Table 5.5) present the discussions of the answers to the sub-question or secondary questions. Table 5.6 has a summary. A statement of accepting, rejecting or considering the hypothesis indeterminate will be expressed for each question based on a summary of evidence and the significant research outcome of each question.

Table 5.1: Secondary Research Question 1 and Tested Hypotheses

Secondary Question 1: *What are the locations where the evidence can be found?*
Hypothesis 1: Backend database server can be compromised by a malicious RFID R/W tag.
Hypothesis 2: The value of a SI tagged with R/W RFID tag can be changed.
Hypothesis 3: There will be transaction traces in the RFID reader's memory after the attack.
Hypothesis 4: There will be transaction traces in the POS/Server's memory after the attack.
Hypothesis 5: There will be transaction traces in the SQL Server transaction logs after the attack.
Hypothesis 6: There will be transaction traces in the SQL Server error logs after the attack.

Hypothesis 7: The malicious tag can be found at the crime scene.	
ARGUMENT FOR: The backend database server was able to be successfully compromised by using a fake RFID R/W tag injected with malicious code (Section 3.3.2). The traces of evidence after the attack were able to be identified, acquired, preserved and analysed according to the digital forensic principles and guidelines for handling the digital evidence. Hence, the evidence of changes to the original values of the stock items tagged with R/W RFID tags was found in the backend database during the investigation (see Figure 4.37 in Section 4.2.3.4.4). Likewise, the evidence of malicious transaction traces including the tag ID, date and timestamp was found in logs of RFID reader's memory (Section 4.3.1) and POS RAM (Section 4.3.2). As stated above, the significant evidence was found in the acquired current table data of the stock management database (Figure 4.37) from the backend SQL Server. It showed that all the SI values were $600	**ARGUMENT AGAINST:** The implemented RFID stock management system was a simple system design as it did not deploy the intrusion detection/prevention system, CCTV system and web server in the research experiment. Furthermore, other modules for Credit Card payment processing, purchasing, receiving, customers, reporting, and the like were not integrated in the system. Hence, additional research are still needed as more evidence traces could be found if some of these modules or software was even integrated in the system design. For instance, the attacker could be identified if he/she paid for the stolen item by using Electronic Funds Transfer at Point of Sale (EFPOTS) or Credit Card instead of cash. Likewise, the evidence such as IP address of the unauthorized user could also be found in the Web Server log if the attack was coming from the Internet (Section 3.1.4). Moreover, the evidence could be found in the records of CCTV if implemented. In the real world, the percentage of prosecuting the perpetrator would be lesser without showing such evidence.

and confirmed the database was obviously compromised. Moreover, the acquisition results of current table data of transaction log file - RFID_test_log.ldf (see Figure 4.41) showed the significant evidence of the attack in which the malicious SQL poisoning code was found and the SI (Tag IDs starting with E004) were updated to the values $600 at 06:39:48pm on the 12 October 2010. A fake tag was found and preserved as a proof of the theft of SI, in the simulation experiment.	Likewise, the findings in SQL Server error logs and system event logs such as security, system and application logs (see Figures 24A.2, 24A.3, 24A.4 in Appendix 24) are not convincing enough to prosecute the malicious attacker. Similarly, the fake tag could be rewritten for another SI if the retailer did not have proper policy of managing and reusing the tag which was just detached from SI sold.
SUMMARY: In the simulation experiment, the RFID stock management system was able to be compromised by using malicious RFID tag injected with SQL poisoning code to steal the SI by paying $600 instead of $1500. The forensic image analysis was performed on each artefact extracted from the three entities of the simulated RFID BS and the sub-systems identified in Table 2.5 (Section 2.3). Even though the significant evidence to prove the theft of SI is found in almost all the areas of the RFID BS, there are some limitations in the finding results. However, the arguments made for and against prove most of the proposed hypotheses (1, 2, 3, 4, and 5) are to be accepted except hypothesis 6. As for hypothesis 7, the arguments made for and against prove to be indeterminate.	

Table 5.2: Secondary Research Question 2 and Tested Hypotheses

Secondary Question 2: *What evidence can be extracted from a compromised RFID based retail system?*
Hypothesis 3: There will be transaction traces in the RFID reader's memory after

<table>
<tr><td colspan="2">the attack.

Hypothesis 4: There will be transaction traces in the POS/Server’s memory after the attack.

Hypothesis 5: There will be transaction traces in the SQL Server transaction logs after the attack.

Hypothesis 6: There will be transaction traces in the SQL Server error logs after the attack.

Hypothesis 7: The malicious tag can be found at the crime scene.</td></tr>
<tr><td>ARGUMENT FOR:
As stated in the “argument for” in Table 5.1 above, the significant evidence was found in each entity of RFID BS. Hence the significant evidence was found in the logs of RFID reader’s memory, POS RAM, and transaction logs of backend SQL Server.

In the simulation experiment, a fake tag was also found and preserved as a proof of the theft of SI.</td><td>ARGUMENT AGAINST:
The significant evidence in RFID reader might not be found if the buffer of the reader’s memory was full when the malicious tag was scanned.

As a result of the firmware used, the RFID reader (Tracient Padl-R UF Reader) deployed in the experiment can currently read not only the upper limit of 15000 tags if the data (TagID) logging is turned off, but also the lower limit of 900 TagIDs and timestamps if the data logging is turned on. However, different tags have different tad ID length and thus there is a small amount of difference. Hence, when the reader’s memory buffer is full due to the maximum tags read; it will continue to read and transmit data from the tag via USB to the backend server. But, no more data will be written to the reader’s memory log.</td></tr>
</table>

SUMMARY:
After analysing all the acquired data (Section 4.3), the significant evidence to prove the theft of SI is found in almost all the areas of the RFID BS, there are a few limitations concerned in the finding results such as the buffer size of the reader's memory and tag re-writing policy of the RFID based retail shop. However, the arguments made for and against prove most of the asserted hypotheses (3, 4, and 5) are to be accepted except hypothesis 6 due to no significant result was not found in the error log. As for hypothesis 7, the arguments made for and against prove to be indeterminate.

Table 5.3: Secondary Research Question 3 and Tested Hypothesis

Secondary Question 3: *What is the best way to preserve the acquired evidence?*	
Hypothesis 9: The MD5 hash values of the collected evidence before and after analysis are the same. Hence the collected evidence is credible, reliable, repeatable, and acceptable to the court of law.	
ARGUMENT FOR: As stated in Section 4.2 and Section 4.3, all the potential evidence in each entity of RFID BS was acquired and analysed. All the collected artefacts were digitally hashed with trusted *Message Digest Algorithm 5* (MD5) by using dcfldd and md5deep tools from RFID_IR tool during the evidential data acquisition. Likewise, the hash values of volatile SQL Server data were automatically generated by Extended Windows Forensic Toolchest (WFT) during the acquisition. Similarly, the physical disk drive image of POS/Server were acquired and hashed with AccessData® FTK®	**ARGUMENT AGAINST:** In the real world, in order to prevent employees of the RFID BS based retail shop using/compromising the potential evidence (for instance: re-writing the malicious tag for another SI or shutting down the POS system); the target system or crime scene should be isolated and secured. Otherwise, the fragile digital and physical evidence states could be compromised during the investigation. Hence, the preservation of the evidence would not be sustained. Chain of custody, security, and transportation of the digital evidence should also be documented and cautiously performed during the digital

Imager (version 2.9.0.1385), and the physical database files of backend SQL Server were also acquired and hashed with dcfldd tool. In addition, RFID reader's memory log image was acquired with customized **ReaderLogExtraction** tool and hashed with md5deep tool during the acquisition phase. In order to preserve the integrity of the collected evidence, all the collected artefacts from each entity of the compromised RFID BS were saved into a forensically sterilized USB destination flash drive (see Appendix 11) as stated in Section 3.3.4 and hashed with EnCase before analysing. Hardware write blocker (*Tableau Forensic USB Bridge*) was deployed in order to preserve the integrity of the collected evidence data during acquisition and analysis phases (Sections 3.3.5 and 3.3.6). Similarly, the acquisition of evidential data was performed from a trusted customized RFID_IR DVD tool kit in order to minimize the changes to the compromised system during the investigation. Analysing the forensic image copy of	forensic investigation.

all the collected digital evidence data was in order to maintain and security of the evidence. Finally, the hash values before and after analysis were compared and found exactly the same (see Appendix 22). Hence, the preservation of the digital evidence data was done in a forensically sound manner during the research experiment as the hash values were not changed.	
SUMMARY: The acquisition and analysis methods of the potential evidence were performed and documented in a forensically sound manner, as stated in Section 4.2 and Section 4.3. MD5 hashing on the acquired data was done by using different forensic tools and techniques. The acquired evidence is credible, reliable and repeatable and acceptable to the court of law as the hash values of collected evidence before and after analysis are precisely the same. Although there are some limitations as stated in the above mentioned *argument against*, the arguments made for and against prove the hypothesis 9 is to be accepted in this research experiment.	

Table 5.4: Secondary Research Question 4 and Tested Hypothesis

Secondary Question 4: *What are the methods to analyze the acquired evidence?*	
Hypothesis 8: The significant evidence can be extracted by analyzing collected data with EnCase forensic software, Windows Forensic Toolchest (WFT), and a hardware write blocker (*Tableau Forensic USB Bridge*).	
ARGUMENT FOR: Forensic copy of the collected data was performed by using a hardware write blocker connected to the USP port of the forensic work-station installed with EnCase forensic software (see Appendix 12). Hence, the integrity of	**ARGUMENT AGAINST:** Physical evidence could also be found at the POS given the Tag repository (assuming the poison Tag had not yet been rewritten). Further investigation of the business

the collected data was maintained by forensically analysing on the forensic image copy of the collected data. Analyses of bit-to-bit image copy of the logs from RFID reader's memory and POS RAM were done by using EnCase. The significant finding results included fake tag ID, timestamp, and malicious code (see Sections 4.3.1 and 4.3.2). The image copy of collected SQL data was analyzed by using text viewer, notepad++ text and source code editor, Microsoft Excel to which the SQL Server data collected by WFT tool was exported, and the like (Section 4.3.3, Appendices 23 and 24). The notable findings included fake tag ID, the current values of stock items, timestamp and malicious SQL poisoning code in order to prove the theft of SI in a RFID based retail shop.	system by interviewing the human participants and reviewing CCTV. In the real world business context, the evidence extracted from the server logs and the POS and scanner/reader logs could be utilised to speed the search of CCTV frames. The dates and time that are located in the logs can be matched against frames of visual surveillance and witness statements. Further research needs to be conducted in a real world business context, as the evidence from CCVT and interviewing human participants can lead to identify and prosecute the malicious attacker or SI thief.
SUMMARY: For further research, the potential evidence such as frames of visual surveillance and human participants could be analysed to prosecute the malicious attacker. However the significant evidence to prove the theft of SI were able to be extracted by analyzing collected data with EnCase forensic software, Windows Forensic Toolchest, a hardware write blocker (*Tableau Forensic USB Bridge*), and the like in this research experiment. The finding results after the analysis proved that the theft of SI was taken place in a RFID BS. Hence, the arguments made for and against prove the hypothesis 8 is to be accepted.	

Table 5.5: Secondary Research Question 5 and Tested Hypotheses

<table>
<tr><td colspan="2">Secondary Question 5: What are the capabilities to determine the attack event?</td></tr>
<tr><td colspan="2">Hypothesis 1: Backend database server can be compromised by a malicious RFID R/W tag.

Hypothesis 2: The value of a SI tagged with R/W RFID tag can be changed.

Hypothesis 3: There will be transaction traces in the RFID reader memory after the attack.

Hypothesis 4: There will be transaction traces in the POS/Server’s memory after the attack.

Hypothesis 5: There will be transaction traces in the SQL Server transaction logs after the attack.

Hypothesis 6: There will be transaction traces in the SQL Server error logs after the attack.

Hypothesis 7: The malicious tag can be found at the crime scene.</td></tr>
<tr><td>ARGUMENT FOR:
The significant evidence was found in each entity of the RFID BS such as the logs of RFID reader’s memory, POS RAM, and transaction logs of backend SQL Server, as described in the “argument for” in Table 5.1 above.

Furthermore, a fake tag was also found and preserved as a proof of the theft of SI in the simulation experiment.</td><td>ARGUMENT AGAINST:
The arguments against are similar to that of Table 5.1 above. Likewise, the evidence from CCVT and interviewing human participants can lead to identify and prosecute the malicious attacker or SI thief.

On the other hand, the evidence from the physical fake tag could not be found if the tag was re-written for another SI on the same day.</td></tr>
<tr><td colspan="2">SUMMARY:
The attack event was able to prove with the finding results after the analysis of collected data including timestamp when the attack was initiated, how the attack</td></tr>
</table>

occurred, what the ID of malicious tag was, what the malicious SQL poisoning code was used to change the values of SI from the backend database, what the current SIs values were in the database of backend RFID stock management system. However, more evidence could be found from CCTV and interviewing human participants if these factors were integrated into the implemented system design and methodology of the research. Nevertheless, the finding results during the investigation of the compromised RFID BS are more than enough to determine and prove the attack event. Thus, the arguments made for and against prove the hypotheses (1, 2, 3, 4, 5, and 6) are to be accepted and hypothesis 7 is to be indeterminate.

Table 5.6: Main Research Question and Tested Hypotheses

Main Question: *What should the forensic investigator do in order to perform the complete and accurate forensic examination of a compromised RFID stock management system in a retail sector?*
Hypothesis 1: Backend database server can be compromised by a malicious RFID R/W tag. **Hypothesis 2:** The value of a SI tagged with R/W RFID tag can be changed. **Hypothesis 3:** There will be transaction traces in the RFID reader memory after the attack. **Hypothesis 4:** There will be transaction traces in the POS/Server's memory after the attack. **Hypothesis 5:** There will be transaction traces in the SQL Server transaction logs after the attack. **Hypothesis 6:** There will be transaction traces in the SQL Server error logs after the attack. **Hypothesis 7:** The malicious tag can be found at the crime scene. **Hypothesis 8:** The significant evidence can be extracted by analyzing collected data with EnCase forensic software, Windows Forensic Toolchest, and a hardware

write blocker (*Tableau Forensic USB Bridge*).

Hypothesis 9: The MD5 hash values of the collected evidence before and after analysis are the same. Hence the collected evidence is credible, reliable, repeatable, and acceptable to the court of law.

ARGUMENT FOR:	**ARGUMENT AGAINST:**
Before performing the forensic examination the tools required the investigation of the compromised RFID stock management system were prepared. As part of the preparation stage, RFID ReaderLogExtraction Tool was developed (see Appendix 3) for the purpose of acquiring bit-to-bit evidence data from RFID reader's memory during the investigation. Similarly, the SQL Server incident response scripts were successfully integrated with the automated WFT (see Appendix 2). Hence, RFID incident response (Helix_RFID_IR) toolkit was created by integrating all the tools required for the investigation such as winen.exe, ReaderLogExtraction tool and the like into the e-fense® Helix3 (version: 2009R1) live forensic toolkit (see Appendix 2). Likewise, the dcfldd incident response USB toolkit (in which FTK imager and dcfldd tool were included) was created (Appendix 11) and the forensic work-station (Appendix 10) was installed in the laboratory.	Even though, the fundamental digital forensic methodologies (Section 3.1.7) were applied during the acquisition and analysis of the collected artefacts; live forensic acquisition methodology (mentioned in Sections 3.1.3 and 3.1.5) was utilized during the investigation of the compromised RFID BS. Therefore, a few areas of the compromised system must have been changed. For instance, attaching the forensic data collection USB device could alter the registry of the compromised system. Thus, the date and the timestamp of a USB device connection should be carefully noted. The information noted could be subtracted from the collected artefacts of the compromised during the analysis before presenting the evidence to the court. However, the forensic analysis of the registry of the compromised RFID BS was not conducted in the research experiment. Such issue will be identified in the future research area in Section 6.2.

The storage drive (USB flash drive) for the digital evidence data was prepared in a forensically sound manner for the purpose of the investigation (see Appendix 11). An authorized administrator account to the compromised was acquired from the system administrator. The potential evidence was identified, collected (according to the volatility) and saved into the trusted USB flash drive by using pre-developed RFID_IR and forensic USB toolkits from the compromised RFID BS (Section 4.2). Hardware write blocker (*Tableau Forensic USB Bridge*) was deployed during the data acquisition, processing and analysis phases (Section 3.3.4, Section 3.3.5, and Section 3.3.6 respectively) in order to maintain the integrity of the evidence. As stated in Table 5.4, the collected data (Section 4.2) were analysed (Section 4.3) and preservation of the digital evidence was accomplished (Table 5.3). Hash values comparison of the	The arguments against are similar to that of Table 5.1 and 5.5 above. The evidence from CCVT and interviewing human participants can lead to identify and prosecute the malicious attacker or SI thief. On the other hand, the malicious tag can be re-written for another SI on the same day if there is no proper policy of re-writing the tag in the retail shop.

collected digital evidence files (before and after analysis) was performed (Appendix 22). The significant findings after analysis of each entity of RFID BS were presented (arguments for in Table 5.1) in order to prove the theft of SI in RFID based stock management system. Hence, the forensic examination of a compromised RFID stock management system in a retail sector (the research experiment in the laboratory) was accomplished.	

SUMMARY:

In order to perform the complete and accurate forensic examination of a compromised RFID stock management system in a retail sector; the forensic investigator had prepared the tools required for acquisition, got an authorized administrator login of the compromised system, secured the crime scene, identified and acquired the potential evidence. Then, the investigator did analyse the acquired evidence by using the forensic analysis tools in a forensically sound manner. After performing the forensic examination, as stated in the summary of Table 5.5; the attack event (theft of SI) was able to prove with the finding results. The limitations such as interviewing human participants, securing and transporting the acquired evidence safely, chain of custody, CCTV analysis were not performed in the research experiment. However, the arguments made for and against prove the hypotheses (1, 2, 3, 4, 5, 6, 8 and 9) are to be accepted and hypothesis 7 is to be indeterminate.

5.2 DISCUSSION OF FINDINGS

The findings of the research were detailed, analysed and reported in the previous chapter (Chapter 4). This section will now discuss the significance of the results related to the theft of SI in a RFID BS. Hence, the discussions will include the phases of research, the evaluation of implemented RFID stock management system design for the research including the attack case scenario presented, hardware and software deployed, forensic investigation processes used for the research experiment.

5.2.1 Discussion of Conducted Research Phases

The research experiment was composed of five different phases (Section 3.2.5) and each phase had its own specific goal. To identify and emphasize the significant findings (Chapter 4), the discussion of research testing phases will involve the stabilized RFID stock management system design setup (Section 3.3.1), attack case scenario for data generation (Section 3.3.2), evidence data acquisition or extraction (Section 3.3.4), data analysis (Section 3.3.6), and data presentation (Section 3.3.7). However, the first phase of the research will not be covered in this section as it was conducting research and analyzing of the previous similar published works (Chapter 2; Sections 3.1, 3.2.1 and 3.2.2 in Chapter 3) in order to derive the methodology for research conducted (Chapter 3).

5.2.2 Discussion of Implemented System Design

The second phase of the research experiment, a stabilized RFID BS was setup (Appendix 6) after conducting a few pilot tests. The problems were encountered during the pilot testing phases of setting up a stabilized system (Section 4.1.1). For instances, the problems such as scanning the SI tag and tuning the Tripwire software during the initial setup in order to create the baseline of the system were encountered. Even though, a stabilized RFID BS was set up to fit a laboratory simulation according to the BS mentioned in Section 2.3; there would be many more tags, scanners, data entries in the log, CCTV cameras and the whole world of commercial human interaction in a commercial business system.

5.2.3 Discussion of Conducted Attack Case Scenario

Likewise, the ways in which the RFID BS was compromised might not be the same in the real world as different types of malicious attacks could come from different areas of RFID enabled BS (Chapter 2; Section 2.4 and Chapter 3; Section 3.2.2). However, the simulated system design (Chapter 3; Section 3.3.1) and the attack case scenario to stress a stabilized system (Section 3.3.2) were sufficient to test the vulnerability of a RFID BS.

5.2.4 Discussions of Identification, Acquisition and Extraction

The identification, acquisition and extraction processes of evidential data, in the fourth phase of the research experiment, were performed by using pre-developed forensic tools (stated in Section 3.3.4; Appendices 2 and 11; Table 5.6). The primary concern is to preserve the evidence. The volatility of the evidence is apparent around the SI (including the scanner) and the POS. The volatility is driven by security policy and the business practices adopted in any particular enterprise. Policies that manage the Tag processes will also determine the availability or otherwise of the poisoned Tag for evidential purposes. In most instances a Tag is removed at the POS and placed in a secure receptacle for future rewriting. Hence, a poisoned Tag has potential for rewriting and the erasure of the poison code (Section 3.2.2.2). Similarly the BIS architecture will determine the storage capabilities of the POS and the scanner. In some instances the scanner has a continuous wireless feed into the BIS and in others it stores evidence. The stored evidence has cycle times where previous entries are erased and the memory reused. The POS is also dependant on BIS architecture. On the other hand, POS stores transactions on a hard drive and in others routs directly to the server/ data warehouse. Each variation requires investigation to determine system and to assure the best preservation rate for evidence.

The identification of evidence in the simulation was simple because the researcher determined the system architecture and chose the hardware capabilities. Also the laboratory context had no business processes or policies to obstruct the access to evidence. As a consequence the poisoned Tag was available after the event, the scanner log was complete, the POS hard drive and RAM chip were available, and the database backend had fewer than 500 entries that could be

accessed. The challenge was to follow correct process in extraction and then to conduct an analysis that would locate the attack event amongst the potential of many transaction occurrences.

In the laboratory simulation the POS had a hard drive and also a RAM chip, and dumped complete records into the SQL Server on demand. Two data bases (RFID_test.mdf & RFID_test_log.ldf) had been created prior to the simulation to hold the authentic SI records and also to record each transaction (Section 3.3.1). The scanner stored 1Mb of log evidence that recorded transactions line by line (ID & Time Stamp). When the buffer was full then the algorithm started again at line one rewriting each record in the Scanner memory.

The evidence acquisitions (Section 3.3.4) were achieved by taking each element of the SI, POS and BIS and then write blocking each extraction (this included using ddcfldd for hashing and acquiring SQL Server data). Helix_RFID_IR tool was taken from the Incident Response Took Kit and customised so that other tools could also be used securely within the extraction framework. This included WinEn for RAM extraction and RFID code extraction. The physical hard drive of the POS was imaged using FTK Imager (Disk Jockey Pro was available as a backup if any problems occurred), and the SQL Server data were acquired by using Windows Forensic Tool Chest (WFT) and ad hoc acquisition methods.

Hence, the laboratory context simplified the extraction of evidence phase. In addition, the further investigation of the business system by interviewing the human participants and reviewing CCTV footage was not relevant (Chapter 3; Section 3.4). In the Business context, the evidence extracted from the server logs and the POS and scanner logs (if available) could be utilised to speed the search of CCTV frames. The dates and time that are located in the logs can be matched against frames of visual surveillance and witness statements.

5.2.5 Discussions of Analysis and Presentation

The analysis (Section 4.3) of forensic image copy focused on each artefact extracted from the three entities of the simulated BIS and the sub-systems identified in Table 2.5 (Chapter 2; Section 2.3). Prior to performing the forensic analysis, the acquired evidence data were forensically copied on the forensic work-station (see Appendix 12). EnCase forensic software installed on the

forensic work-station (Appendix 10) was used as a main forensic analysis tool during the investigation. Initially a visual scan of the Master Server Log was made to quickly identify where SI values had been changed. For instance, it was quick to find out the malicious traces in entries all SIs values of $600 in the acquired current table of RFID_test.mdf database (see Figure 4.37 in Section 4.2.3.4.4). Likewise the fake tag ID, malicious SQL poisoning code and the time stamp were also easily identified (see Figure 4.41 in Section 4.2.3.4.4). Moreover, the acquisition results of current table data of log file - RFID_test_log.ldf (see Figure 4.41) showed the significant evidence of the attack in which the malicious SQL poisoning code was found and the SI (Tag IDs starting with E004) were updated to the values $600 at 06:39:48pm on the 12 October 2010. At this point in the analysis enough evidence had been discovered so that the analysis of the RAM, Tag and Scanner would be for verification and exception evidence.

Then, the fake tag ID was used as a keyword to look for the malicious transaction in the image copy of scanner memory and POS RAM (Section 4.3.1 and Section 4.3.2 respectively) when analysing with EnCase. The significant evidence such as malicious tag ID, time stamp, and SQL poisoning code were also discovered. However, in a real business search; the technique to query the transaction logs in the Master Server Log would have been more complex and time consuming. Such a search can be bench marked from stock controls and the legitimate time stamps of value alterations. The time stamp identified was then used as a keyword search in the other images. In a real search the existence of evidence in these subsystems would be a bonus given the volatile state. Nevertheless, in the simulation experiment; the evidence was present in each of the subsystems and it was searched for time stamp and ID strings. The Tag also contained the malicious code that can be signature matched for identification of potential sources.

As stated in Section 3.3.5, the comparison of the MD5 hash values before and after the analysis (in order to find out whether the evidence has been modified during analysis and acquisition) was performed as part of preservation. The MD5 hash values were the same as before and after analysis. Thus, the evidence files are not tempered during the analysis phase and the integrity is still intact. However, in the real world investigation; the limitations (Section 3.4) such as interviewing human participants, securing and transporting the acquired evidence

safely, chain of custody, CCTV analysis might also be conducted during the forensic examination. Otherwise, the forensic examination of a compromised RFID based stock management system would not be complete and comprehensive. Thus, the perpetrator or theft or malicious attacker would not be able to be prosecuted.

5.3 DISCUSSION OF RECOMMENDATIONS: BEST PRACTICES

The previously discussed sections and the findings of the research experiment (Chapter 4) have guided the forensic investigators in such a way in which the knowledge of the digital forensic practices or procedures in RFID based Business System (RFID BS) need to be broadened. Especially, the digital forensic procedure in the processes of investigation preparedness, artefact acquisition or extraction, analysis and presentation of the theft of SI in a compromised RFID BS was investigated. Hence, the knowledge acquired during the research experiment will now be discussed as the recommendations for forensic investigation related to RFID BS. In addition, the potential evidence can be acquired from RFID BS will be outlined and the recommendations for digital forensic best practices concerned with investigation preparedness, incident response, acquisition, preservation, analysis of a compromised RFID stock management system.

5.3.1 Managing Investigation Output and Investigation Preparedness: Best Practices

Managing the outputs of investigation is significant to any forensic investigator in order to preserve the integrity of collected evidence data, as saving collected evidence on the victim system will compromise the integrity of the evidence. Thus, the collected evidence should be saved in a trusted location. The trusted location could be either on forensic workstation/laptop or on the locally connected external storage media, depending on whether the investigation is conducted interactively on the victim system or via remote connection. However, attaching the storage media for investigation outputs directly to the target victim system will change the system state. Thus, the investigator must document the drive letter representative, connection and drive letter creation time, disconnection time of the output storage media. After documenting the information in detail, the investigator could exclude these changes in the investigation. Similarly, the forensic tools required and

forensic test-station for the investigation must be prepared in advance before initiating the forensic processes such as acquisition, and analysis.

5.3.2 Incident Response: Best Practices

When responding to the incident, a valid SQL Server login with SQL Server system administrator privileges is required in order to access the required databases to perform the investigation. Furthermore, the investigator should gather as much detailed information about the incident from the system administrator or owner of the retail shop. The detailed information such as the timeline and events of the incident, the employees involved during the scope of the incident occurred, the compromised database size and so forth. Likewise, the forensic investigator should initially identify any infrastructure obstacles such as the *intrusion detection or prevention (IDS/IPS), antivirus,* and *Windows firewall* could hinder the investigation (Section 3.1.3). However, as stated in Section 3.1.3, the network-based barriers could be avoided by the investigator performing an interactive investigation on the victim system. But, the same host-based products could hinder the interactive investigation. Thus, the investigator should document the steps taken to disable any host-based product that interfere the investigation.

5.3.3 Precautions for Responding to an SQL Server Incident: Best Practices

In order to minimize or avoid the changes to SQL Server during live investigation; the following precautions should be undertaken when responding to an SQL Server incident (Section 3.13):

- Only the trusted binaries which are located on the RFID IR DVD should be used during live acquisition.

- Data manipulating language (DML) or data definition language (DDL) statements should not be run as these statements could alter the data stored in memory.

- Any actions, such as creating server logins, database user accounts and the like, that could modify SQL Server memory and the data on-disk should be avoided at all cost.

- Limiting the use of memory residents such as temporary tables and variables during investigation would restrict an effect on the SQL Server buffer pool.

- By avoiding the changes in permissions on database objects could help the investigator not to overwrite previous permission assignments on database objects and not to generate various log entries during investigation.

- To avoid the unpleasant affects on the victim system's SQL syntax or logic errors, the executions of required SQL statements for investigation should be initially tested on the test station as a small error in SQL syntax could have catastrophic consequences on the database server.

5.3.4 Identification, Acquisition and Preservation: Best Practices

As previously stated (Section 5.2.4; Sections 3.3.4 and 3.3.5 in Chapter 3), the evidential search of a compromised RFID BS should be preceded by preserving the evidence of the volatile entities and subsystems. Forensic imaging would usually proceed from the most volatile to the least. However, in the real world RFID BS; the key evidence could be located in priority from the Web server, backend database server, the POS and the SI. If the SI evidence had been lost on account of volatility then the Server Image should contain sufficient detail from which to fully investigate the system. Hence the SI and POS could be isolated (define system first) and the Servers imaged as a priority. Human actors or CCTV dimensions must also be considered as the sources of evidence. The evidence from these sources would normally be imaged and documented at the same time.

The acquisition of potential evidence data from a compromised RFID BS should be treated carefully, as the volatile data could be vanished if the backend SQL Server was down. The forensic investigator must deploy a forensically sound methodology of collecting potential evidence data in order to preserve the integrity of the collected data. For instance, saving the collected data into a trusted location, using hardware write blocker during acquisition and analysis, and acquiring and analysing the potential evidence from a compromised system with

the trusted forensic toolkits. In addition, all the forensic tools used and steps performed during the investigation should be documented in order to track the activities performed by the forensic investigator. Moreover, the comparison of hash values (example: Appendix 22) of collected data before and after should also be done in order to verify the integrity of the evidence and preserve the evidence. Securing the crime scene (Section 3.3.4), transporting the collected evidence (Section 3.3.4), chain of custody (Section 3.1.5) and analysing the forensic image copy on a specific forensic workstation are also the most important factors/forensic processes to maintain the integrity of the collected evidence.

5.3.5 Analysis of the Acquired Data: Best Practices

In analysis of the acquired data from a compromised RFID BS, the ID and time stamps from the related sources are significant to speed the extraction of relevant evidence to the theft of SI from the acquired artefacts. Similarly, the analysis phase may also lead to a second round of human interviews. In addition, POS evidence such as CCTV and interview evidence from human participants (Chapter 3; Section 3.4) must also be acquired, analysed and documented in the real world. Moreover, proper forensic tools used (such as EnCase, FTK, Tableau forensic hardware write blocker) and the steps taken during the forensic examination must be documented.

5.4 CONCLUSION

In this chapter, the discussion of findings from the research experiment presented in Chapter 4 was made. The answers to the proposed research questions from the methodology chapter (Section 3.2.6) were discussed in relation to the asserted hypotheses (Section 3.2.7) and a conclusion was reached with regard to the validity of the anticipated hypotheses. Likewise, the findings after the investigation of a compromised RFID system were also discussed and evaluated. Furthermore, the issues related to the investigation were stated.

The main research question was a focal point to prove the theft of SI in a RFID BS and the phases of research model (Section 3.2.5) were established. The tested system design (Section 3.3.1) was also set up based on the main research question. The findings (Chapter 4) after a complete forensic examination were able to prove the theft of SI in a simulated RFID based stock management system.

Moreover, the significant findings were capable of proving the malicious attack, when the attack occurred, a fake tag ID number, the values of SIs changed in the backend server after the attack, and the malicious SQL poisoning code that compromised the backend database.

In the following chapter, the thesis research project will be concluded. A summary of the research conducted and the significant answers to the research questions will be outlined in that conclusion chapter (Chapter 6). Furthermore, the areas for future research will be explained and followed by the conclusion.

Chapter 6 -

Conclusion

6.0 INTRODUCTION

The significant gap in the digital forensics research relating to commercial RFID business system (RFID BS) tools and professional procedures was noted in Chapter 1. The relevant literature was reviewed in Chapter 2, including the three-tier model of a RFID system (Section 2.1, Section 2.2, and Section 2.3), the challenges and RFID security threats (Section 2.4). A summary was made of the relevant issues and problems (Section 2.7), notably that wireless RFID enabled systems are susceptible to malicious attacks. In Chapter 3, a researchable problem and question were identified (Section 3.2.6 and Section 3.2.7) and a plausible research methodology specified (Section 3.3 and Section 3.4).

Then, a prototype of a commercial retail environment using a RFID stock management system was set up in the laboratory and customized RFID middleware (Appendix 4) was developed by using a Software Development Kit (SDK) of a RFID reader manufacturing company to help the requirements of the system design set up. Subsequently, the stabilized RFID BS (Section 3.3.1) was stressed by a SQL poisoning attack through a RFID tag as part of the data generation before performing the digital forensic investigation as stated in Section 3.2.1. Hence, the attack was replicated by reviewing the previous literature (Section 3.1.2). Likewise, the ReaderLogExtractionTool (Appendix 3) was created to acquire bit-to-bit logs from the RFID reader's memory and to develop the customized Helix_RFID_IR toolkit (Appendix 2) before the forensic investigation was initiated. Then, the compromised RFID BS was investigated. The evidential search was performed in each of the entities of the RFID BS (Table 2.5 in Section 2.3) by using a Helix_RFID_IR toolkit. Hence, the findings of the research were presented, analysed and discussed in Chapter 4 and Chapter 5 respectively.

In order to conclude this research project, the following sections are included. Section 6.1 is a summary of findings from the research conducted and

Section 6.2 summarises the answers to the research questions. Then, future research opportunities arising are explained in Section 6.3, followed by the conclusion (Section 6.4).

6.1 SUMMARY OF FINDINGS

During the simulated research experiment, two customized Helix_RFID_IR DVD and dcfldd (USB thumb drive) toolkits were used to collect the potential evidence data from the compromised RFID BS. Tools such as FTK® Imager, *WinMD5.exe* and dcfldd are placed in a sterilized USB (see Appendix 11) to create a dcfldd USB toolkit, whereas Guidance Software® *winen.exe*, developed *ReaderLogExtraction* and extended WFT tools are also integrated into the original Helix3 (Live Forensic Tool) in order to develop a Helix_RFID_IR DVD toolkit. The potential evidence data collected from the compromised system was performed by using live forensic acquisition methods. Then the forensic image copy of the collected data was performed and analysed by using EnCase Forensic Training Software (Version 6.16.1**)** on the pre-installed forensic work-station. A hardware write-blocker (Tableau Forensic USB Bridge) was employed in the investigation for data collection, analysis and preservation of the evidence. Hence, the summary of the specific forensic tools used and purposes of utilizing those tools in the research conducted were presented in Table 6.1 and Appendix 1.

After the analysis of the collected evidence, the significant findings of the research were found in each entity of the RFID BS (Chapter 2; Table 2.5) and can prove that the theft of SI has occurred in the simulated RFID based retail system. Hence, the significant findings are summarized in Table 6.2. The key outputs from the research and the answer to the research question were in the software, the integration of modules, and the learning that can advise professional investigators how to undertake such an investigation and the elements of an effective extraction tool.

Table 6.1: Specific forensic tools used during the experiment and their purposes

Forensic Tools Used	Purpose(s)
Helix_RFID_IR Toolkit	Customized Helix Incident Response DVD toolkit for performing live forensic investigation of RFID BS.
Developed ReaderLogExtraction Tool	For acquisition of the bit-to-bit image of RFID scanner/reader's memory logs.
Extended Windows Forensic Toolchest (WFT) v3.0.05.	For acquisition and analysis of volatile backend SQL Server data.
AccessData® FTK® Imager (version 2.9.0.1385 100406)	For acquisition of the physical disk drive image of POS/Server and MD5 hashing of collected artefacts after analysis for preservation.
dcfldd imaging tool	For acquisitions of the physical database files, trace files and error logs of backend SQL Server.
psloglist.exe tool	For acquisitions of application, system and security logs.
Guidance Software® *winen.exe*	For acquisition of POS memory image.
WinMD5.exe	For MD5 hashing on the Helix3.iso (Helix3 ISO version 2009R1 – Original Helix3 Live Forensic Tool) image in order to find out whether the hash value of the downloaded Helix 3 is the same as that of the value mentioned on the e-fense website. Hence, it is assumed that there is no error during downloading of the original Helix 3 image if the MD5 hash values are the same.
Guidance Software® EnCase Forensic Training (version 6.16.1) installed on the forensic work-station	For creating a forensic image copy of all original collected data, performing MD5 hashing on the collected data before analysis for preservation, forensically sterilizing flash drives for investigations and analysis of POS memory and RFID reader's memory images.
BackTrack 4 Live DVD (which is a Linux-based penetration testing distribution)	For forensically sterilizing a USB flash drive (16GB) for storing the collected data from compromised RFID BS.
Hardware Write-blocker (Tableau Forensic USB Bridge)	For creating a forensic image copy of the original collected data before analysis (attached to the forensic work-station) and the intermediary attachment device between *dcfldd* toolkit and compromised POS/Server of the RFID BS in order to maintain the originality of potential evidence data and minimize the effect of the forensic tool on the compromised system during the data collection.

Table 6.2: Summary of significant findings related to the entities of a RFID BS

Entity	Stock Items (SI)	Point of Sale (POS)	Business Information System (BIS)
Sub-Systems	Tag Scan	Scan TPS	Data Storage IS Management
Services	ID Authentication	De/Attach Authorization	Refresh Audit
Significant Findings	Malicious RFID tag found and preserved as a proof of SI theft in the simulated experiment.	Evidential traces of fake tag ID number, date & timestamp found in the memories of RFID scanner and POS host station/ controller (Chapter 4; Section 4.3.1 & Section 4.3.2 respectively).	Changes to the original values of SIs noticed in the backend database (Chapter 4; Figure 4.37). Evidence of the malicious SQL poisoning code was found in the transaction log of the backend server (all SI values changed to $600 at 06:39:48pm on the 12 October 2010; see Figure 4.41).

6.2 ANSWER TO THE RESEARCH QUESTIONS

The purpose of the main research question was to perform a complete and accurate forensic examination of a compromised RFID stock management system in the retail sector. Thus, identifying and examining the correct procedures for achieving investigation preparedness (Section 5.3.1) and incident response (Section 5.3.2) are significant contributions of the research. In addition, the factors such as cost and the time taken to perform the investigation will increase if the forensic investigator does not prepare for or response well to the incident. Likewise, digital forensic investigation (DFI) procedures such as identification, acquisition, extraction, analysis, preservation, documentation and presentation of

the evidential data are the important phases of this simulated research. Such DFI procedures are utilized to answer the research questions (Chapter 3; Section 3.2.6) in the research conducted. The research sub-questions are derived from the main research question and the answers provided are related to the six categories of the DFI framework (***How***, ***When***, ***Where***, ***What***, ***Why***, and ***Who***) questions as discussed in Section 3.1.7. Hence, the research questions were answered and evaluated against the tested hypotheses by extracting the evidence from the Chapter 4 report and using a table format presented in Tables 5.1 - 5.6 (Chapter 5; Section 5.1). In addition, the summarized significant findings (Table 6.2) after a complete forensic examination, give answers to the DFI framework questions. However, there are limitations in the research conducted (Chapter 3; Section 3.4). For instance, the simulated research was conducted in a closed RFID BS environment and there were limitations to finding potential evidence in POS areas such as CCTV. Furthermore, interview evidence from human participants was not available. Nonetheless, the findings (Chapter 4) after a complete forensic examination were able to prove the theft of SI in a simulated RFID based stock management system.

6.3 CONCLUSION AND FURTHER RESEARCH

The main objective of this research project was achieved. This involved the complete and accurate forensic examination of a compromised RFID stock management in the retail sector. The RFID controlled stock inventories are not trustworthy and even with the best securing, still have a significant residual risk of violation. In the simulated research experiment, the theft of a SI through an orchestrated test scenario of hardware, software and social engineering illustrates the potential vulnerability of the RFID BS. The research conducted confirms that the evidence after such a successful attack can be found principally in the server logs and then around the POS and perhaps (depending on the BS architecture) on the scanner (‘reader’) middleware. Similarly, the evidence can also be found in interviews with human actors and in surveillance devices (an item for further research). To a lesser extent the evidence stored in scanners and tags may be available. Hence, the forensic investigation can proceed using the standard

practice of write blocking, potential evidence identification, collection, forensic imaging, preservation and analysis.

As part of the digital forensic investigation, the evidential search of a compromised RFID business system is conducted on each of the BS entities. Evidence is located in a complex business environment that has elements of technicality, informatics and humanity. These elements keep changing and the specifications change with new innovations (another item for ongoing further research to keep up to date). The forensic imaging usually proceeds from the most volatile to the least, hence in this RFID BS; the key evidence can be located in priority from the POS, the Server BIS, and the SI. The scan ('Reader') logs are complete and show the transaction date and time. In many retail scenarios, this evidence would not be available as most large systems work on a scan-and-dump or on a wireless continuous server contact basis. The main repositories of evidence are found on the backend database of the 'Enterprise System' (Chapter 2; Figure 2.13). In addition, physical evidence would also be found at the POS, given the Tag repository (assuming the poison Tag had not yet been rewritten), CCTV, and interview evidence from the human participants (again items for further research).

Further research can be undertaken into retention and business practices to preserve evidence around the SI and the policy relationship to the Server Image. Policies could assure sufficient detail is available to fully investigate the system. Hence the SI and POS can be isolated (that is, define the system first) and the Server Imaged as a priority. On the other hand, the evidence from these sources would normally be imaged at the same time and in the analysis phase, the ID and time stamps from the related sources used to speed the extraction if the simulation conducted did involve human actors or CCTV dimensions. The analysis phase may also lead to a second round of human interviews. However, these matters are not considered in the simulation. Therefore, only the transaction logs, the master data file, the loading records, and the malicious source code would be available for preservation. Hence further research can be undertaken to assert the optimal configurations and process contingencies.

To conclude, the research conducted demonstrates the enhancement of knowledge in digital forensic investigation (DFI) of RFID based retail systems. Hence, the information contained in this thesis will provide best practices

(Chapter 5; Section 5.3) for both digital forensic researchers and practitioners. Like any other research project, this research also has limitations (Chapter 3; Section 3.4), and opportunities from which future research ideas can be derived. Thus, the following modifications are suggested toward future research for the simulated RFID stock management system design (Section 3.3.1). Firstly, the test-bed system design could be modified by implementing Web Server applications (for example; Microsoft Internet Information Server), dedicated POS and the backend SQL Server, surveillance system (CCTV), intrusion detection systems (IDS), antivirus client (AVC) software, data processing and accounting applications. Then, the modified RFID BS could be connected to the Internet in order to perform diverse malicious attack events coming via the Internet by authorized or unauthorized person. After the attacks, a DFI can be performed on the modified target RFID BS. Furthermore, after performing DFI procedures, the additional findings such as the logs from Web Server, IDS, AVC and CCTV, interviews with human participants, Windows registry, and network devices would give the forensic examiner a complete and holistic picture of the DFI in a RFID based retail system. Therefore, the DFI can lead to consistent reporting that is trustworthy.

References

Abawajy, J. (2009, October). Enhancing RFID tag resistance against cloning attack. Paper presented at the Third International Conference on Network and System Security (NSS 09), 18-23, Gold Coast, Australia. doi:10.1109/NSS.2009.101

AccessData Corp. (2010). *Forensic toolkit® (FTK) imager user's guide*. Lindon, Utah, United States of America: Author.

Albrecht, K. (2006). RFID security: the doomsday scenario. In S. Garfinkel & B. Rosenberg, (Eds.), *RFID Applications, Security, and Privacy* (pp. 259-273). United States of America: Pearson Education, Inc.

Altschaffel, R., Kiltz, S., & Dittmann, J. (2009). From the computer incident taxonomy to a computer forensic examination taxonomy. *Fifth International Conference on IT Security Incident Management and IT Forensics, 2009 (IMF '09)*, 54-68. United States of America: IEEE Computer Society.

Anthony, D., Kotz, D., & Henderson, T. (2007). Privacy in Location-Aware Computing Environments. *Pervasive Computing – IEEE Computer Society (Oct-Dec 2007)*, 64-72. United States of America: IEEE Computer Society.

Banks, J., Hanny, D., Pachano, M. A., & Thompson, L. G. (2007). *RFID applied*. Hoboken, New Jersey: John Wiley & Sons, Inc.

Bhargava, H. (2006). RFID Security: Attacking the backend. *Syngress Force 2006 Emerging Threat Analysis: From mischief of malicious* (pp. 503-513). Rockland, MA: Syngress Publishing, Inc.

Birari, S. M., & Iyer, S. (2005). Mitigating the reader collision problem in RFID networks with mobile readers. *2005 13th IEEE International Conference on Networks, 2005. Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication.*, vol.1, no., pp. 6 pp.-, 16-18 Nov. 2005.

Bolan, C. (2007). A single channel attack on 915MH radio frequency identification systems. In C. Valli & A. Woodward (Ed.), *Proceedings of the 5th Australian Information Security Management Conference* (pp. 8-16). Perth, Western Australia: School of Computer and Information Science.

Bolan, C. (2009). *A spoofing attack against an EPC Class One RFID system*. Paper presented at the 7th Australian Information Security Management Conference Perth, Western Australia.

Borriello, G. (2005). RFID: tagging the world. *The Communications of the ACM, 48*(9), 34-37.

Brainard, J., Juels, A., Rivest, R., Szydlo, M., & Yung, M. (2006). Fourth factor authentication: somebody you know. *In Proceedings of the 13th ACM Conference on Computer and Communications Security*, (168-178). New York, USA: Association for Computing Machinery. doi: 10.1145/1180405.1180427

Britz, M. T. (2009). *Computer forensics and cyber crime*. New Jersey, U.S.A: Prentice Hall.

Carrier, B. (2005). *File system forensic analysis*. Upper Saddle River, New Jersey, United States of America: Pearson Education, Inc.

Cerrudo, C. (2003). *Manipulating Microsoft SQL Server using SQL injection*. Technical report, Application Security Inc., 2003, 1-14. Retrieved from http://www.appsecinc.com/presentations/Manipulating SQL Server Using SQL Injection.pdf

Cerrudo, C. (2009, July). *SQL Server anti-forensics: techniques and countermeasures*. Paper presented at the Black Hat USA 2009 Conference, Las Vegas, United States of America. Retrieved from http://www.blackhat.com/presentations/bh-dc-09/Cerrudo/BlackHat-dc-09-Cerrudo-SQL-Anti-Forensics.pdf

Chalasani, S., & Boppana, R. (2007). Data architectures for RFID transactions. *IEEE Transactions on Industrial Informatics, 3*(3), pp. 246-257.

Chalasani, S., Boppana, R. V., & Sounderpandian, J. (2005, August 11-14). *RFID tag reader designs for retail store applications.* Paper presented at the Proceedings of the Eleventh Americas Conference on Information Systems (AMCIS 2005), Omaha, NE, United States of America.

Chawathe, S. S., Krishnamurthy, V., Ramachandran, S., & Sarma, S. (2004). Managing RFID data. In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, & K. B. Schiefer (Eds.), *In Proceedings of the Thirtieth International Conference on Very Large Data Bases*, *30*(pp. 1189-1195). Toronto, Canada: VLDB Endowment.

Chawla, V., & Ha, D. S. (2007). An overview of passive RFID. *Communications Magazine, IEEE , 45*(9), 11-17, September 2007.

Cohen, F. (2009). Two models of digital forensic examination. *In Proceedings of the Fourth International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering, SADFE'09*, 42-53. Los Alamitos, California, United States of America: IEEE Computer Society. doi:10.1109/SADFE.2009.8

Collins, J., & Hussey, R. (2003). *Business research: a practical guide for undergraduate and postgraduate students* (2nd ed.). United Kingdom: Palgrave Macmillan.

Delaney, K. (2007). *Inside Microsoft SQL Server 2005: the storage engine*. Redmond, Washington: Microsoft Press.

Digital Forensic Research Workshop (DFRWS). (2001, November). *A Road map for digital forensic research*. New York, United States of America: Gary Palmer.

Ding, Z., Li, J., & Bo, F. (2008). A taxonomy model of RFID security threats. *In IEEE International Conference on Communication Technology Proceedings (2008), 765-768.* United States of America: The Institute of Electrical and Electronics Engineers, Inc.

Dolan-Gavitt, B. (2008). Forensic analysis of the Windows registry in memory. *The Journal of Digital Investigation, 5*(2008), S26-S32. doi:10.1016/j.diin.2008.05.003

Dul, J., & Hak, T. (2008). *Case study methodology in business research*. Great Britain: Butterworth-Heinemann.

e-fense, Inc. (2010). *Helix3 Download*. Retrieved from https://www.e-fense.com/store/index.php?_a=viewProd&productId=11

e-fense, Inc. (2009). *Helix3 2009 R2 User Manual - Live Forensics and Incident Response User Manual*. United States of America: Author.

El-Said, M. M., & Woodring, I. (2009). An empirical study for protecting passive RFID systems against cloning. *Sixth International Conference on Information Technology: New Generations, ITNG '09, vol.,* no., pp.558-563, 27-29 April 2009.

Electronic Product Code Global (EPCglobal). (2010). *EPCglobal standards overview*. Retrieved from http://epcglobalinc.org/standards

Finkenzeller, K. (2003). *RFID handbook: fundamentals and applications in contactless smart cards and identification* (2nd ed.). West Sussex, England: John Wiley & Sons, Inc.

Flores, J. L. M., Srikant, S. S., Sareen, B., & Vagga, A. (2005, January). *Performance of RFID tags in near and far field*. Paper presented at the 2005 IEEE International Conference on Personal Wireless Communications, New Delhi, India.

Fowler, K. (2007, July 28-August 2). *A real world scenario of a SQL Server 2005 database forensic investigation*. Paper presented at the Black Hat USA 2007, Las Vegas, United States of America. Retrieved from https://www.blackhat.com/presentations/bh-usa-07/Fowler/Whitepaper/bh-usa-07-fowler-WP.pdf

Fowler, Kevvie. (2009). *SQL Server forensic analysis*. Massachusetts, United States of America: Pearson Education, Inc.

Fujitsu. (2010). *What is FRAM?*. Retrieved from http://www.fujitsu.com/global/services/microelectronics/technical/fram/index.html

Garfinkel, S. L., Juels, A., & Pappu, R. (2005, May-June). RFID privacy: an overview of problems and proposed solutions. *IEEE Security & Privacy Magazine, 3* (3), 34-43.

Gerring, J. (2007). *Case study research: principles and practices*. Cambridge: Cambridge University Press.

Gonzalez, J. J., Sarriegi, J. M, & Gurrutxaga, A. (2006). A framework for conceptualizing social engineering attacks. In J. Lopez (Ed.), *Critical Information Infrastructures Secutiry – Lecture Notes in Computer Science, 4347*(2006), 79-90. doi:10.1007/11962977_7

Grobler, M. M., & Von Solms, S. H. (2009, July 6-8). Best practice approach to live forensic acquisition. In H. Venter, M. Coetzee, & L. Labuschagne (Eds.), *Proceedings of the Information Security for South Africa (ISSA) 2009 Conference*, 3-14. Pretoria, South Africa: Information Security for South Africa (ISSA).

Guidance Software, Inc. (2010). *EnCase® forensic version 6.17 user's guide*. Pasadena, CA, United States of America: Author.

Haines, B. R. (2006a). RFID attacks: tag encoding attacks. *Syngress Force 2006 Emerging Threat Analysis: From mischief of malicious* (pp. 433-445). Rockland, MA: Syngress Publishing, Inc.

Haines, B. R. (2006b). RFID attacks: tag application attacks. *Syngress Force 2006 Emerging Threat Analysis: From mischief of malicious* (pp. 447-460). Rockland, MA: Syngress Publishing, Inc.

Halamka, J., Juels, A., Stubblefield, A., & Westhues, J. (2006). The security implications of VeriChip cloning. *Journal of the American Medical Informatics Association, 13*(5), 601-607.

Harmon, C. K. (2006). The necessity for a uniform organization of user memory in RFID. *International Journal of Radio Frequency Identification Technology and Applications, 1*(1), 41-51.

Harrill, D. C., & Mislan, R. P. (2007). A small scale digital device forensics Ontology. *Journal of Small Scale Digital Forensics, 1*(1), 1-7.

Hendrickson, A. R., Mennecke, B., Scheibe, K., & Townsend, A. M. (2006, September). *Feasibility and proof of concept of utilizing radio frequency identification tagging technology to manage evidentiary materials in forensic laboratories*. United States of America: author.

Heydt-Benjamin, T. S., Bailey, D. V., Fu, K., Juels, A., & O'Hare, T. (2007). Vulnerabilities in first-generation RFID-enabled credit cards. In S. Dietrich & R. Dhamija (Eds.), *Financial Cryptography and Data Security – Lecture Notes in Computer Science, 4886*(2007), 2-14. doi:10.1007/978-3-540-77366-5_2

Hossain, M. M., & Prybutok, V. R. (2008). Consumer acceptance of RFID technology: an exploratory study. *IEEE Transactions on Engineering Management, 55*(2), 316-328. doi:10.1109/TEM.2008.919728

Hunt, V. D., Puglia, A., & Puglia, M. (2007). *RFID - a guide to radio frequency identification*. New Jersey, United States of America: John Wiley & Sons, Inc.

Ieong, R. S. C. (2006). FORZA – Digital forensics investigation framework that incorporate legal issues. *The Journal of Digital Investigation, 3S*(2006), S29-S36. doi:10.1016/j.diin.2006.06.004

Jeng, A. B., Chang, L., & Wei, T. (2009). Survey and remedy of the technologies used for RFID tags against counterfeiting. *2009 International Conference on Machine Learning and Cybernetics, vol.5*, no., pp.2975-2981, 12-15 July 2009.

Jeong, D., Kim, Y., & In, H. (2005). New RFID system architectures supporting situation awareness under ubiquitous environments. *Journal of Computer Science, 1*(2), 114-120.

Jones, A. K., Hoare, R. R., Dontharaju, S. R., Shenchih, T., Sprang, R., Fazekas, J., Cain, J. T., & Mickle, M.H. (2006a). *A field programmable RFID tag and associated design flow. The 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2006. FCCM '06, vol.*, no., pp. 165-174.

Jones, K. J., Bejtlich, R., & Rose, C. W. (2006b). *Real digital forensics: computer security and incident response***.** Upper Saddle River, New Jersey, United States of America: Addison-Wesley.

Jones, R. (2007). *Safer live forensic acquisition*. Retrieved from http://www.cs.kent.ac.uk/pubs/ug/2007/co620-projects/forensic/report.pdf

Juels, A., Rivest, R., & Szydlo, M. (2003). The blocker tag: selective blocking of RFID tags for consumer privacy. In V. Alturi (Ed.), *Proceedings of the 8th ACM conference on computer and communication security, Washington, DC, USA* (pp. 103–111). New York: Association for Computing Machinery (ACM).

Juels, A. (2005). Strengthening EPC tags against cloning. In M. Jakobsson & R. Povendran (Eds.), *Proceedings of ACM workshop on wireless security (WiSec'05)* (pp. 67–76). New York: Association for Computing Machinery (ACM).

Juels, A., & Weis, S. (2005). Authenticating pervasive devices with human protocols. In V. Shoup (Ed.), *Advances in Cryptology (CRYPTO 2005) – Lecture Notes in Computer Science, 3126*(2005), 293-308. doi:10.1007/11535218_18

Juels, A. (2006). RFID security and privacy: a research survey. *IEEE Journal on Selected Areas in Communications, 24*(2), 381-394. doi:10.1109/JSAC.2005.861395

Kamoun, F. (2009). RFID system management: State-of-the art and open research issues. *Journal of IEEE Transactions on Network and Service Management, 6*(3), 190-205. doi: 10.1109/TNSM.2009.03.090305

Karygiannis, A., Phillips, T., & Tsibertzopoulos, A. (2006). RFID security: a taxonomy of risk. *In Proceedings of the First International Conference on Communications and Networking (ChinaCom '06), China* (pp. 1-8). doi:10.1109/CHINACOM.2006.344722

Karygiannis, T., Eydt, B., Barber, G., Bunn, L., & Phillips, T. (2007). *Guidelines for securing radio frequency identification (RFID) systems*, Special Publication 800-98, National Institute of Standards and Technology, April. Retrieved from http://csrc.nist.gov/publications/nistpubs/800-98/SP800-98_RFID-2007.pdf

Khannaa, N., Mikkilinenia, A. K., Martonea, A. F., Alia, G. N., Chiub, G. T., Allebacha, J. P., & Delpa, E. J. (2006). A survey of forensic characterization methods for physical devices. *Digital Investigation, 3S*(2006), S17-S28.

Kim, D. S., Shin, T., & Park, J. S. (2007). A security framework in RFID multi-domain system. *The Second International Conference on Availability, Reliability and Security, ARES 2007, vol.*, no., pp.1227-1234, 10-13 April 2007.

Knospe, H., & Pohl, H. (2004). RFID security. *Information Security Technical Report, 9*(4), 39-50. doi:10.1016/S1363-4127(05)70039-X

Konidala, D. M., Kim, Z., & Kim, K. (2007). A simple and cost-effective RFID tag-reader mutual authentication scheme. *In Proceedings of International Conference on RFID Security 2007 (RFIDSec07), Malaga, Spain (141-152)*.

Kou, D., Zhao, K., Tao, Y., & Kou, W. (2006). RFID technologies and applications. In W. Kou & Y. Yesha (Eds.), *Enabling technologies for wireless e-business* (pp. 89-108). The Netherlands: Springer.

Landt, J. (2005, December 5). The history of RFID. *IEEE Potentials Magazine, 24*(4), 8-11. doi:10.1109/MP.2005.1549751

Laurie, A. (2007). Practical attacks against RFID. *Network Security, 2007*(9), 4–7. doi:10.1016/S1353-4858(07)70080-6

Li, X., Xu, G., & Yu, D. (2008). Security architecture for RFID application in home environment. *The 2nd International Conference on Anti-counterfeiting, Security and Identification, ASID 2008*, 467-470.

Li, H., Yuan, C., Yen, C., Huang, Y., & Lin, W. (2009). Design and implementation of RFID mutual authentication Protocol. *In Proceedings of the Third International Conference on Anti-counterfeiting, Security and Identification in Communication, ASID 2009, Hong Kong* (pp. 229-232). United States of America: IEEE Computer Society. 10.1109/ICASID.2009.5276910

Li, T., & Deng, R. H. (2007). Vulnerability analysis of EMAP-An efficient RFID mutual authentication protocol. *Proceedings of the Second International Conference on Availability, Reliability and Security, ARES 2007, Vienna, Austria* (238-245). doi:ieeecomputersociety.org/10.1109/ARES.2007.159

Martone, A. F., & Delp, E. J. (2007). Characterization of RF devices using two-tone probe signals. *IEEE/SP 14th Workshop on Statistical Signal Processing, SSP '07, vol.*, no., pp.161-165, 26-29.

Martone, A. F., Mikkilineni, A. K., & Delp, E. J. (2006). Forensics of things. *2006 IEEE Southwest Symposium on Image Analysis and Interpretation,* 149-152.

Masters, G., & Turner, P. (2007). Forensic data recovery and examination of magnetic swipe card cloning devices. *Journal of Digital Investigation, 4S*(2007), S16-S22.

Michael, K., & McCathie, L. (2005). The pros and cons of RFID in supply chain management. *In Proceedings of the International Conference on Mobile Business (ICMB'05).* USA: IEEE Computer Society.

Middleton, B. (2004). *Cyber crime investigator's field guide* (2nd ed.). United States of America: Auerbach Publications.

Microsoft. (2007). *SQL Server 2005 Books Online*. Retrieved from http://www.microsoft.com/downloads/en/details.aspx?FamilyId=BE6A2C5D-00DF-4220-B133-29C1E0B6585F&displaylang=en

Mirowski, L., Hartnett, J., & Williams, R. (2009). An RFID attacker behavior taxonomy. *Journal of IEEE Pervasive Computing, 8*(4), 79-84. doi:10.1109/MPRV.2009.68

Mitrokotsa, A., Rieback, M. R., & Tanenbaum, A.S. (2009). Classifying RFID attacks and defenses. *Special Issue on Advances in RFID Technology, Information Systems Frontiers, Springer Science and Business Media, LLC 2009*(July). doi:10.1007/s10796-009-9210-z

Mitrokotsa, A., Beye, M., & Peris-Lopez, P. (2010). In D. C. Ranasinghe, Q. Z. Sheng & S. Zeadally (Eds.), *Threats to networked RFID systems* (pp. 39-63). Heidelberg, Germany: Springer.

Nath, B., Reynolds, F., & Want, R. (2006). RFID technology and applications. *Journal of Pervasive Computing, IEEE Computer Society and Communications Society, 5*(1), 22-24. doi:10.1109/MPRV.2006.13

National Law Enforcement and Corrections Technology Centre - A program of the National Institute of Justice. (2005, Summer). *Technology primer: radio frequency identification*. United States of America: the National Institute of Justice.

Ngai, E. W. T., Moon, K. K. L., Riggins, F. J., & Yi, C. Y. (2008). RFID research: an academic literature review (1995-2005) and future research directions. *In International Journal of Production Economics, 112*(2), 510-520.

Ohkubo, M., Suzuki, K., & Kinoshita, M. (2005). RFID privacy issues and technical challenges. *The Communications of the ACM, 48(9),* 66-71.

Peris-Lopez, P., Hernandez-Castro, J. C., Estevez-Tapiador, J. M., & Ribagorda, A. (2006). EMAP: An efficient mutual authentication protocol for low-cost RFID tags. *In OnTheMove (OTM) Federated Conferences and Workshop: IS Workshop, Montepellier, France.*

Peris-Lopez, P., Hernandez-Castro, J. C., Tapiador, J. M. E., & Ribagorda, A. (2009). Advances in ultra-lightweight cryptography for low-cost RFID tags: Gossamer protocol. In K.-I. Chung, K. Sohn, & M. Yung (Eds.), Workshop on Information Security Application (WISA 2008) – Lecture

Notes in Computer Science 5379(2009), 56-68. doi:10.1007/978-3-642-00306-6

Phillips, A. (1999). *Programmer's file editor home page*. Retrieved from http://www.lancs.ac.uk/staff/steveb/cpaap/pfe/pfefiles.htm

Qin, T., & Burgoon, J. (2007). An investigation of heuristics of human judgment in detecting deception and potential implications in countering social engineering. In G. Muresan, T. Altiok, B. Melamed, & D. Zeng (Eds.), *Proceedings of 2007 IEEE Intelligence and Security Informatics Conference,* 152-159. New Brunswick, New Jersey, United States of America: The Institute of Electrical and Electronics Engineers, Inc. doi:10.1109/ISI.2007.379548

Rao, K. V., Nikitin, P. V., & Lam, S. F. (2005). Antenna design for UHF RFID tags: a review and a practical application. *In IEEE Transactions on Antennas and Propogation, 52*(12), 3870-3876.

Rao, S. (2007). A secure architecture for the use of RFID at home. *In Proceedings of 3rd Annual GRASP Symposium, 2007,* 191-192.

Reith, M., Carr, C., & Gunsch, G. An examination of digital forensic models. *International Journal of Digital Evidence Fall 2002, 1*(3), 1–12.

Rieback, M. R., Simpson, P. N. D., Crispo, B., & Tanenbaum, A. S. (2006a). RFID malware: design principles and examples. *Journal of Pervasive and Mobile Computing, 2*(2006), 405–426.

Rieback, M. R., Crispo, B., & Tanenbaum, A.S. (2006b, March). *Is your cat infected with a computer virus?* Paper presented at the Fourth IEEE International Conference on Pervasive Computing and Communications (PerCom2006), Pisa, Italy.

Roberts, C. M. (2006). Radio frequency identification (RFID). *Journal of Computers & Security (2006),* 18-26.

Rotter, P. (2008). A framework for assessing RFID system security and privacy risks. *Journal of Pervasive Computing, IEEE Computer Society, 7*(2), 70-77. doi:10.1109/MPRV.2008.22

Roussos, G., & Kostakos, V. (2009). RFID in pervasive computing: state-of-the-art and outlook. *Pervasive and Mobile Computing, 5* (1), 110-131. doi:10.1016/j.pmcj.2008.11.004

Samani, R. (2010). Re-defining the human factor. *Journal of Infosecurity, 7*(2), 30-33. doi:10.1016/S1754-4548(10)70039-5

Sanghera, P., Thornton, F., Haines, B., Kleinschmidt, J., Das, A. M., Bhargava, H., & Campbell, A. (2007). *How to cheat at deploying and securing RFID*. Burlington, Massachusetts, United States of America: Syngress Publishing, Inc.

Sangwan, R. S., Qiu, R. G., & Jessen, D. (2005). Using RFID tags for tracking patients, charts and medical equipment within an integrated health delivery network. *In Proceedings of Networking, Sensing and Control, 2005 IEEE*, vol., no., pp. 1070-1074, 19-22 March 2005.

Sheng, Q. Z., Li, X., & Zeadally, S. (2008). Enabling next-generation RFID applications: solutions and challenges. *Journal of Computer, IEEE Computer Society, 41*(9), 21-28. doi:10.1109/MC.2008.386

Shih, D., Lin, C., & Lin, B. (2005). RFID tags: privacy and security aspects. *International Journal of Mobile Communications, 3*(3), 214-230.

Simon, M., & Slay, J. (2009). Enhancement of forensic computing investigations through memory forensic techniques. *In Proceedings of 2009 International Conference on Availability, Reliability and Security, ARES*(2009), 995-1000. Los Alamitos, California, United States of America: IEEE Computer Society. doi:10.1109/ARES.2009.119

Solanas, A., Domingo-Ferrer, J., Martı´nez-Balleste´, A., & Daza, V. (2007). A distributed architecture for scalable private RFID tag identification. *Journal of Computer Networks, 51*(2007), 2268-2279.

Thevissen, P. W., Poelman, G., Cooman, M. D., Puers, R., & Willems, G. (2006). Implantation of an RFID-tag into human molars to reduce hard forensic identification labor. Part I: Working principle. *Journal of Forensic Science International, 159S (2006), 159*(Supplement 1), S33-S39.

Thevissen, P. W., G. Poelman, Cooman, M. D., Puers, R., & Willems, G. (2006). Implantation of an RFID-tag into human molars to reduce hard forensic identification labor. Part 2: Physical properties. *Journal of Forensic Science International, 159*(Supplement 1), S40-S46.

Thornton, F., Haines, B., Das, A. M., Bhargava, H., Campbell, A., & Kleinschmidt, J. (2006). *RFID Security*. Canada: Syngress Publishing, Inc.

Tripwire, Inc. (2010a). *Tripwire Manager & Tripwire for Servers 4.8 reference guide*. Portland, United States of America: Author.

Tripwire, Inc. (2010b). *Tripwire for Servers 4.8 installation guide*. Portland, United States of America: Author.

Tripwire, Inc. (2010c). *Tripwire Manager 4.8 user guide*. Portland, United States of America: Author.

Tu, Y., & Piramuthu, S. (2008). Reducing false reads in RFID-embedded supply chains. *Journal of Theoretical and Applied Electronic Commerce Research, 3*(2), 60-70.

Vacca, J. R. (2005). *Computer forensics: computer crime scene investigation* (2nd ed.). Massachusetts, United States of America: Charles River Media, Inc.

Wang, Z., Sun, X., Zhang, C., & Li, Y. (2007). Issues in integrated circuit design for UHF RFID. *IEEE International Workshop on Radio-Frequency Integration Technology, RFIT 007, vol.,* no., pp.322-328, 9-11 Dec. 2007.

Wang, L., Zhang, R., & Zhang, S. (2009, December 18-20). *A Model of computer live forensics based on physical memory analysis*. Paper presented at the 1st International Conference on Information Science and Engineering (ICISE), Nanjing, China. Retrieved from http://ieeexplore.ieee.org

Weis, S. A. (2003a) *Security and privacy in radio-frequency identification devices*. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Weis, S. A, Sarma, S., Rivest, R., & Engels, D. (2003b). Security and privacy aspects of low-cost radio frequency identification systems. In D. Hutter, G. Müller, W. Stephan, & M. Ullmann (Eds.), *Security in Pervasive computing, Proceedings of the 1st International Conference in Security in pervasive computing, Boppard, Germany, March 12–14, 2003. Lecture notes in computer science* (Vol. 2802, pp. 201–212). Berlin: Springer. doi:10.1007/b95124

Weiser, M. (1991). The computer for the twenty-first century. *Scientific American*. 91-104.

Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the Association for Computing Machinery (ACM), 36(7)*, 75-84.

Workman, M. (2008). Gaining access with social engineering: an empirical study of the threat. *Journal of Information Systems Security, 16*(2008), 315-331.

Xiao, Y., Yu, S., Wu, K., Ni, Q., Janecek, C., & Nordstad, J. (2007). Radio frequency identification: technologies, applications, and research issues. *Journal of Wireless Communications and Mobile Computing 7*(4): 457-472.

Zhang, M., Li, W., Wang, Z., Li, B., & Xia, R. (2007). A RFID-based material tracking information system. Paper presented at the *2007 IEEE International Conference on Automation and Logistics,* vol., no., pp.2922-2926, 18-21 Aug. 2007.

Zhang, Y., & Lin, Y. (2010, April 2-4). *Research on the key technology of secure computer forensics*. Paper presented at The 2010 International Symposium on Intelligent Information Technology and Security Informatics (IITSI), Jinggangshan, China. Retrieved from http://ieeeexplore.ieee.org

Zhou, Z., & Huang, D. (2008). SRK: A distributed RFID data access control mechanism. Paper presented at the *IEEE International Conference on Communications, ICC '08, vol.,* no., pp.2854-2858, 19-23 May 2008.

Digital Forensics in Small Devices: RFID Tag Investigation

Volume 2

AR KAR KYAW
B.Eng.Tech. (Massey University, NEW ZEALAND), B.E. (Mandalay Technological University, BURMA)

A thesis submitted to the Graduate Faculty of Design and Creative Technologies
AUT University
in partial fulfilment of the
requirements for the Degree of
Master of Forensic Information Technology

School of Computing and Mathematical Sciences

Auckland, New Zealand
2010

Table of Contents (Volume 2)

Appendices

Appendix 1: Volatile and Non-volatile Artefacts of backend SQL Server 2005 (Collection Methods & Purposes)

Table A1. 1: Volatile Artefacts and Method of Acquisition (Compiled from Fowler, 2009, p. 192 & p. 93-94)

Volatile SQL Server Artefacts	Automated Artefact Collection (WFT)	Ad Hoc Artefact Collection	Hashing Method Used for Artefact Preservation md5deep (or) dcfldd	Purpose (in order to:)
Data Cache	●		md5deep	Reconstruct Activity
Cache Clock Hands	●		md5deep	Reconstruct Activity
Plan Cache	●		md5deep	Reconstruct Activity
Most Recently Executed (MRE) Statements	●		md5deep	Determine active unauthorized database access
Active Connections	●		md5deep	Determine active unauthorized database access
Active Sessions	●		md5deep	Determine active unauthorized database access
Active Virtual Log Files (VLFs)	▲	●	md5deep/dcfldd	Reconstruct Activity
Ring Buffers		●	dcfldd	Determine login failures and events related to security.

Table A1. 2: Non-volatile Artefacts and Method of Acquisition (Compiled from Fowler, 2009, p. 192 & p. 93-94)

Non-Volatile SQL Server Artefacts	Automated Artefact Collection (WFT)	Ad Hoc Artefact Collection	Hashing Method Used for Artefact Preservation md5deep (or) dcfldd	Purpose (in order to:)
Authentication Settings*		●	dcfldd	Determine authentication and authorization
Authorization Catalogs*		●	dcfldd	Determine authentication and authorization
SQL Server Logins*	●		md5deep	Determine authentication and authorization
Databases*	●		md5deep	Not directly analysed
Database Users*	●		md5deep	Determine authentication and authorization
Database Objects*	●		md5deep	Reconstruct Activity
Triggers*	●		md5deep	Reconstruct Activity
Jobs*	●		md5deep	Reconstruct Activity
Schemas*	●		md5deep	Reconstruct

				Activity
Endpoints*	●		md5deep	Not directly analyzed
Table Statistics*		●	dcfldd	Recover Data
Auto-executing Stored Procedures*	▲	●	md5deep/dcfldd	Reconstruct Activity
Collation Settings and Data Types*		●	dcfldd	Not directly analyzed
Data Page Allocations*		●	dcfldd	Not directly analyzed
Server Versioning*	●		md5deep	Determine configuration and versioning of backend SQL Server
Server Configuration*	●		md5deep	Determine configuration and versioning of backend SQL Server
Server Hardening*		●	dcfldd	Determine configuration and versioning of backend SQL Server
Native Encryption*		●	dcfldd	Determine configuration and versioning of backend SQL Server

Time Configuration*	●		md5deep	Not directly analysed
Data Files	▲	●	md5deep/dcfldd	Recover Data
Reusable VLFs		●	dcfldd	Recover Data
CLR Libraries	▲	●	md5deep/dcfldd	Reconstruct Activity
Trace Files		●	dcfldd	Reconstruct Activity
SQL Server Error Logs		●	dcfldd	Reconstruct Activity
System Event Logs (application, system, and Security logs)		●	dcfldd	Reconstruct Activity
Web Server Logs		●	dcfldd	Reconstruct Activity
External Security Controls		●	dcfldd	Reconstruct Activity

*** Its collection depends on a running MSSQLServer service**

▲ Partial artefact collection ● Full artefact collection

Appendix 2: Steps for Creating Radio Frequency Identification (RFID) Incident Response DVD Toolkit

Nowadays, as stated by Fowler (2009), hundreds of free and commercial forensic tools are available. For instances, *First Responders Evidence Disk (FRED), Computer Online Forensic Evidence (COFEE), Windows Forensic Toolchest (WFT) and Incident Response Collection Report (IRCR)* are popular Windows incident response toolkits available today. However, these available forensic tools are designed to carry out specific purposes and are not precisely suitable for investigation of a compromised RFID Business System (RFID BS). Thus, in this research project, a customized RFID Incident Response (Helix_RFID_IR) DVD toolkit is created as part of the investigation preparedness for forensic investigation of a compromised RFID BS. The idea to create Helix_RFID_IR is based on the creation of a customized *"SQL Server forensic incident response toolkit"* (Fowler, 2009, p.108 -138) and "*Real digital forensics: computer security and incident response"* (Jones et al., 2006b). Hence, the steps for creating RFID IR toolkit are as follow:

1. Download freely available Helix3.iso (version: 2009R1) from e-fense: https://www.e-fense.com/store/index.php?_a=viewProd&productId=11
   MD5 hash value of Helix3.iso, 3ac2ca7d8d1dcc494ef5124c1cf37f7c, was noted.

2. Check the hash value by using WinMD5.exe after downloading the Helix3 in order to find out whether there was an error during the download. Hence, the MD5 hash values were the same.

3. Mount the Helix3 with DAEMON Tools Lite (http://www.disc-tools.com/download/daemon).

4. Create the folder "IR" on the desktop of a computer in the lab.

5. Copy the wft folder from Helix3 to "IR" folder.

6. Copy the trusted Extended WFT Executables.

7. Create SQL folder under Tools and paste those three files (sqlcmd.exe, Batchparser90.dll, sqlcmd.rll)

8. Copy *WFTSQL.bat* and *WFTSQL.cfg* from the companion DVD of *SQL Server Forensic Analysis* (Fowler, 2009)

9. Also, copy all readily available SQL Server Incident Response Scripts from the companion DVD of *SQL Server Forensic Analysis* (Fowler, 2009).

10. Customize the WFTSQL.cfg and WFTSQL.bat files.

11. However, RunSQL.bat was downloaded from the following website operated by Fowler (2009) as that file was missing in the companion DVD:

    http://www.applicationforensics.com/research/microsoft/sql-server/sql-2000-2005-2008/sql-server-forensic-analysis-isbn-0321544366.

12. Then, update the configuration file (WFTSQL.cfg) hashes and replace the existing wftsql.cfg file with the new configuration file that contains the updated hashes in the previous step. But the problem is encountered while updating the configuration file (see Figure A2.1).

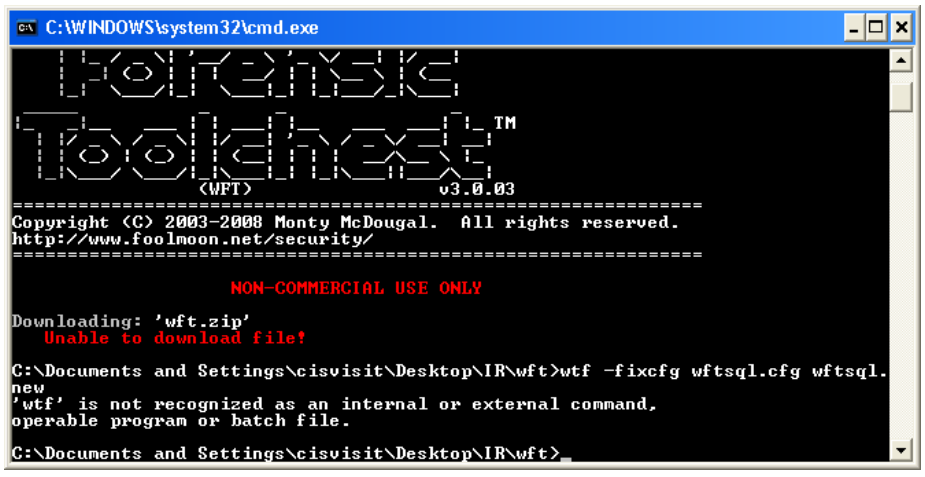


Figure A2.1: Problem encountered when running the command (wft.exe –fixcfg wft.cfg wft.new)

13. Thus, create the WFTSQL folder on the desktop of a computer in the lab for the creation of extended version of WFT tool.

14. Gather the latest versions of WFT executables by using the following command (*WFT's fetchtools arguments*) to download them from http://www.foolmoon.net/security/index.html.

➢ **wft.exe -fetchtools**

15. Verify the downloaded files by using the following command to generate MD5 hashes for all SQL scripts and executables that will be used by WFT framework. Hence, those newly generated hashes are also saved into wft.new configuration file and the execution results can be seen in the figure A2.2.

➢ **wft.exe -fixcfg wtf.cfg wft.new**

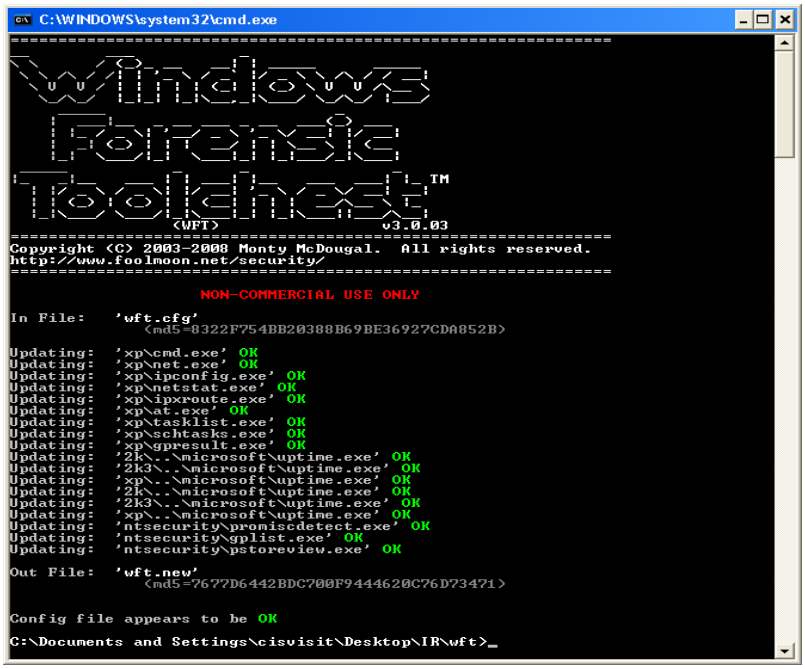


Figure A2.2: Screenshot of the result of wft.exe – fixcfg wft.cfg wft.new

16. Copy SQL incident response files from the companion DVD of SQL Server Forensic Analysis (Fowler, 2009).

17. Verify the copied files by using the following command:

➢ **wft.exe -fixcfg wftsql.cfg wftsql.new**

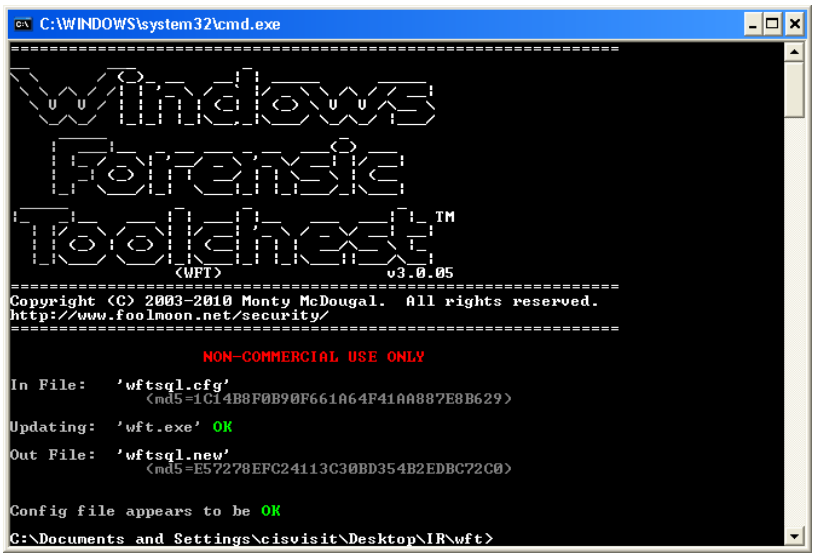


Figure A2.3: Screenshot of the result of wft.exe – fixcfg wftsql.cfg wftsql.new

18. Replace the existing WFT configuration file with the newly created configuration file (wftsql.new, in order to update the MD5 hashes mentioned in above – step 3) by running the following command:

    - **move /y wftsql.new wftsql.cfg**

19. Then, launch the pilot test investigation with the test server by using the following command as the extended version of WFT tool was completed after performing the above mentioned steps. Hence WFT execution by launching WftSQL.bat file *will gather the SQL Server-related inputs and launch the WFT executable in the background* (Fowler, 2009, p.133).

    - wftSQL.bat localhost SA SQLEXPRESS

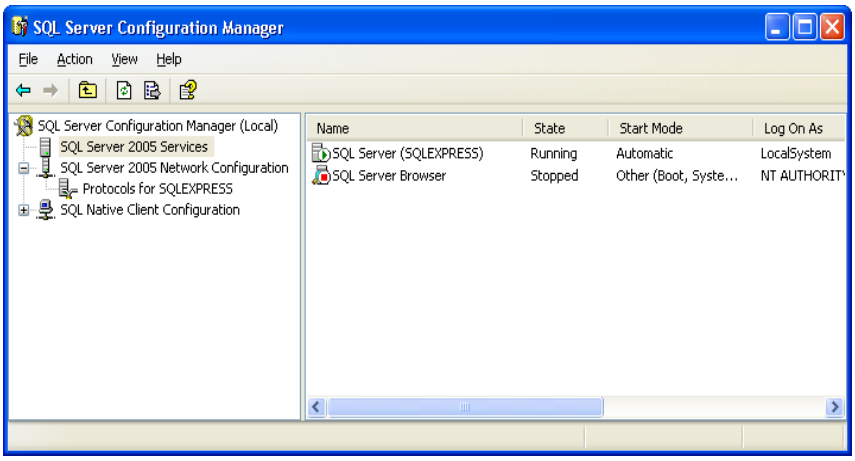


Figure A2.4: Screenshot of the pilot test server information

The following is the detail record of running the wftSQL.bat syntax:

```
> wftSQL.bat localhost SA SQLEXPRESS
        (md5=A2993744A56BA83ADE52948062C968A2)
      'PlanCache.htm'
        (md5=FB59BC80713CA9D39744A02AC548FA1A)
21:59:35: Verifying 'sql\SSFA_TLog.sql' OK
       (md5=F51463E731B82396D80E0D8D21ECD21F)
21:59:35: Verifying 'sql\runsql.bat' OK
       (md5=FEDD1E9DF1B76C877B8027B62E401B50)
21:59:35: Running   'sql\runsql.bat' [#4/23]
        COMPLETE
      'Tlog.txt'
        (md5=A2993744A56BA83ADE52948062C968A2)
      'Tlog.htm'
        (md5=E617F47ABE17AAAD98A08A4C6FAB98E9)
21:59:44: Verifying 'sql\SSFA_RecentStatements.sql' OK
       (md5=D7E3D283EA517BC5B02D6EB71E5F9C2C)
21:59:44: Verifying 'sql\runsql.bat' OK
       (md5=FEDD1E9DF1B76C877B8027B62E401B50)
21:59:44: Running   'sql\runsql.bat' [#5/23]
        COMPLETE
      'RecentStatements.txt'
        (md5=A2993744A56BA83ADE52948062C968A2)
      'RecentStatements.htm'
        (md5=7AF20D605D778409C993EEC453C1EB05)
[Active Connections]
```

21:59:53: Verifying 'sql\SSFA_Connections.sql' OK
(md5=66266B9645B366F4E7167FA909CA30DC)
21:59:53: Verifying 'sql\runsql.bat' OK
(md5=FEDD1E9DF1B76C877B8027B62E401B50)
21:59:53: Running 'sql\runsql.bat' [#6/23]
COMPLETE
'Connections.txt'
(md5=A2993744A56BA83ADE52948062C968A2)
'Connections.htm'
(md5=9829ED5A073E9216A3E29EF62B6DCE6B)
22:00:02: Verifying 'sql\SSFA_Sessions.sql' OK
(md5=67CEE2132CC6059B994D51696BC56C53)
22:00:02: Verifying 'sql\runsql.bat' OK
(md5=FEDD1E9DF1B76C877B8027B62E401B50)
22:00:02: Running 'sql\runsql.bat' [#7/23]
COMPLETE
'Sessions.txt'
(md5=A2993744A56BA83ADE52948062C968A2)
'Sessions.htm'
(md5=FBFC8843EF149D0EC5E314EC58A24FD7)
[DB Objects & Users]
22:00:11: Verifying 'sql\SSFA_Logins.sql' OK
(md5=42DA8E136A8E7B104AFFE7AA678AE141)
22:00:11: Verifying 'sql\runsql.bat' OK
(md5=FEDD1E9DF1B76C877B8027B62E401B50)
22:00:11: Running 'sql\runsql.bat' [#8/23]

COMPLETE

'Logins.txt'

(md5=A2993744A56BA83ADE52948062C968A2)

'Logins.htm'

(md5=FB93421B954A8702A0C7A4CB430956DA)

22:00:20: Verifying 'sql\SSFA_DbUsers.sql' OK

(md5=1FA12BB80DF343E0451D66309334D78D)

22:00:20: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

22:00:20: Running 'sql\runsql.bat' [#9/23]

COMPLETE

'DbUsers.txt'

(md5=A2993744A56BA83ADE52948062C968A2)

'DbUsers.htm'

(md5=58EE24E7F05117741A25A1F34E96A35B)

22:00:29: Verifying 'sql\SSFA_DBObjects.sql' OK

(md5=698ABFF6AACA4C52F6864249BD658F9D)

22:00:29: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

22:00:29: Running 'sql\runsql.bat' [#10/23]

COMPLETE

'DatabaseObjects.txt'

(md5=A2993744A56BA83ADE52948062C968A2)

'DatabaseObjects.htm'

(md5=8D7D14177FEA8B09A090F41CBAE91E85)

22:00:38: Verifying 'sql\SSFA_Triggers.sql' OK

(md5=AB700D66945BBC9FBFA34002A808A208)

22:00:38: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

22:00:38: Running 'sql\runsql.bat' [#11/23]

COMPLETE

'Triggers.txt'

(md5=A2993744A56BA83ADE52948062C968A2)

'Triggers.htm'

(md5=E4C7931A55F840872FEED790B75B55B0)

22:00:47: Verifying 'sql\SSFA_Jobs.sql' OK

(md5=7376F41C4390BEFF4D4CBA4289901885)

22:00:47: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

22:00:47: Running 'sql\runsql.bat' [#12/23]

COMPLETE

'Jobs.txt'

(md5=A2993744A56BA83ADE52948062C968A2)

'Jobs.htm'

(md5=AD5D329C48004A00825C2E2836CBD62C)

22:00:56: Verifying 'sql\SSFA_JobHistory.sql' OK

(md5=A104029D94AA4780E0A4A0A1BC54106A)

22:00:56: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

22:00:56: Running 'sql\runsql.bat' [#13/23]

COMPLETE

'JobHistory.txt'

(md5=A2993744A56BA83ADE52948062C968A2)

'JobHistory.htm'

(md5=33CB1E52133FFC134FFC6E0463B57497)

22:01:05: Verifying 'sql\SSFA_CLR.sql' OK

(md5=B6DAE2E421A639536C01517735B9D6C9)

22:01:05: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

22:01:05: Running 'sql\runsql.bat' [#14/23]

COMPLETE

'CLR.txt'

(md5=A2993744A56BA83ADE52948062C968A2)

'CLR.htm'

(md5=E3231195EBC9B44A3A16D1879BA020E9)

22:01:14: Verifying 'sql\SSFA_Databases.sql' OK

(md5=4DC0305DB714B48B6EC49DE3BA90F471)

22:01:14: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

22:01:14: Running 'sql\runsql.bat' [#15/23]

COMPLETE

'Databases.txt'

(md5=A2993744A56BA83ADE52948062C968A2)

'Databases.htm'

(md5=1C252C6AD64B422CD9230476A9D50EFE)

[DB Configuration]

22:01:23: Verifying 'sql\SSFA_DbSrvInfo.sql' OK

(md5=8EFB9C50FEFCA57E45570D772AABDE22)

22:01:23: Verifying 'sql\runsql.bat' OK
(md5=FEDD1E9DF1B76C877B8027B62E401B50)
22:01:23: Running 'sql\runsql.bat' [#16/23]
COMPLETE
'DBServerInfo.txt'
(md5=A2993744A56BA83ADE52948062C968A2)
'DBServerInfo.htm'
(md5=8CE2E0D419B7B35D8D8A4F2A1266BF86)
22:01:32: Verifying 'sql\SSFA_Configurations.sql' OK
(md5=18D4261C8480B6593861F6704CE1BED6)
22:01:32: Verifying 'sql\runsql.bat' OK
(md5=FEDD1E9DF1B76C877B8027B62E401B50)
22:01:32: Running 'sql\runsql.bat' [#17/23]
COMPLETE
'Configuration.txt'
(md5=A2993744A56BA83ADE52948062C968A2)
'Configuration.htm'
(md5=0A6AE152A5A29D5E60F67B371F406238)
22:01:41: Verifying 'sql\SSFA_Schemas.sql' OK
(md5=026EA797C9FD0681C363B095BC32C049)
22:01:41: Verifying 'sql\runsql.bat' OK
(md5=FEDD1E9DF1B76C877B8027B62E401B50)
22:01:41: Running 'sql\runsql.bat' [#18/23]
COMPLETE
'Schemas.txt'
(md5=A2993744A56BA83ADE52948062C968A2)

'Schemas.htm'

(md5=30BB65A20F1381D77E803A54B2F98BE9)

22:01:50: Verifying 'sql\SSFA_EndPoints.sql' OK

(md5=7BF03126871B6CDDD6ACE5A538EC5E7C)

22:01:50: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

22:01:50: Running 'sql\runsql.bat' [#19/23]

COMPLETE

'Endpoints.txt'

(md5=A2993744A56BA83ADE52948062C968A2)

'Endpoints.htm'

(md5=0622404691D76C6A404205A8AB42E290)

22:01:59: Verifying 'sql\SSFA_AutoEXEC.sql' OK

(md5=9CC9D8FC8FF0C0C10D18DA1AABFF62C8)

22:01:59: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

22:01:59: Running 'sql\runsql.bat' [#20/23]

COMPLETE

'AutoExecProcs.txt'

(md5=A2993744A56BA83ADE52948062C968A2)

'AutoExecProcs.htm'

(md5=32AAD49917356C2C5485022DA852C356)

22:02:08: Verifying 'sql\SSFA_TimeConfig.sql' OK

(md5=EA32425B7410CF9EE4938D2B5F7671BB)

22:02:08: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

22:02:08: Running 'sql\runsql.bat' [#21/23]

COMPLETE

'Time.txt'

(md5=A2993744A56BA83ADE52948062C968A2)

'Time.htm'

(md5=2A49B3D070C66A2B978F76424A9ACD14)

22:02:17: Verifying 'sql\SSFA_ClockHands.sql' OK

(md5=7880CA627188AD8CE9786F7D71F5D392)

22:02:17: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

22:02:17: Running 'sql\runsql.bat' [#22/23]

COMPLETE

'ClockHands.txt'

(md5=A2993744A56BA83ADE52948062C968A2)

'ClockHands.htm'

(md5=AACBC7C4EE1749F1F3263A11D1D07F15)

[DONE]

22:02:26: Verifying '2k\res_kit\now.exe' OK

(md5=1CD2DF306E25FBDDF653A9D9B5DC8A41)

22:02:26: Running '2k\res_kit\now.exe' [#23/23]

COMPLETE

'end.txt'

(md5=8919A639F02F6AE89FC8AF67EC6D1E87)

'end.htm'

(md5=EF41453BE5D9E62AC24C00A057405C66)

[WFT]

22:02:26: Hashing 'wft_cfg.txt'
(md5=E57278EFC24113C30BD354B2EDBC72C0)
'wft_hash.txt'
(md5=82CE9CA84A89381186AF0101041B9D6A)
' C:\Documents' is not recognized as an internal or external command, operable program or batch file.
22:02:26: Reporting 'index.htm'
(md5=75BA672BFB215C7CD6BC1E92DA46988F)
'wft_main.htm'
(md5=0E0F3D9A1B794A9C0E8D4409A8F2676D)
'wft_head.htm'
(md5=672497CDEC922250B766B6588458CE0F)
'wft_menu.htm'
(md5=6295163C389A2AE4457E420B87673E13)
'wft_cfg.htm'
(md5=C2E8C3B0AA1349046B6FE2B3385ED7DC)
'wft_hash.htm'
(md5=7D791197AB3E9C9ACE8E91A16A6719B9)
'wft_help.htm'
(md5=25685F01F71CA6022F9C1B6E7B1D9578)
'wft_tool.htm'
(md5=0F6AFC65C5AD14535C2B4C9017E6F9B2)
'wft_link.htm'
(md5=811A99DD6C6415B7D6D55F84C4ABDDD0)
'wft_splash.htm'
(md5=364C3201D729837D94FA2FB182FAF290)

'wft_splash.js'
(md5=90CC848821AEFB7E11D01731175A22C4)
'fmlogo_sm.png'
(md5=51F885B6AB342A12AA6D050C556C0E3C)
'wft.ico'
(md5=C37E8B76F761570353FAC39EAB67AE7B)
'mail.gif'
(md5=38477E98FB3ED60B82906C7DCE8DA645)
'help.gif'
(md5=DABE723A558A1F324CF367C27EE30258)
'bad.gif'
(md5=DEBF875A10DB19E19A261B3F82ED3DAD)
'missing.gif'
(md5=DD83B55DC42DBFB48EC230CE4D8653EA)
'ok.gif'
(md5=A49085E88FE9D8C1C31928217E15DBC0)
'unknown.gif'
(md5=04108ACB26F82BB7C439D292EE7ABB9E)

==========

22:02:26: [RUN COMPLETE]

==========

Windows Forensic Toolchest(TM) (WFT) v3.0.05

http://www.foolmoon.net/security/

==========

Record any checksum(s) below to later verify log integrity

File: 'wft_log.txt' (md5=3F9F6E6954BFEFD30A7078254FA34CD9)

File: 'wft_rpt.xml' (md5=3C5B2DF48544B69BF1C4214134817FEF)

File: 'wft_log.htm' (md5=ADB50A988BA60FE71C93B60EACC2BDAE)

C:\Documents and Settings\cisvisit\Desktop\IR\wft>

20. Then execute WFT.exe for pilot test run.

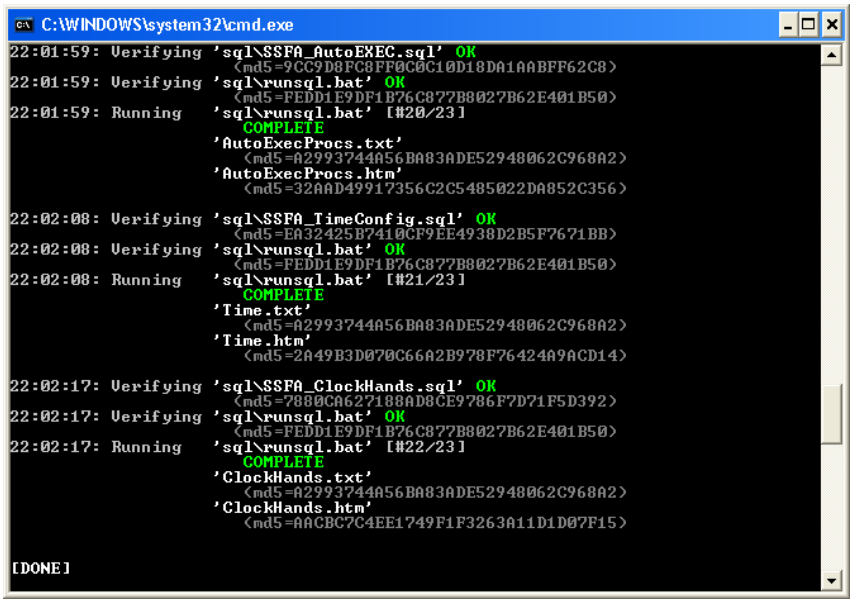


Figure A2.5a: Screenshot of the WFT.exe excution of SQL IR scripts

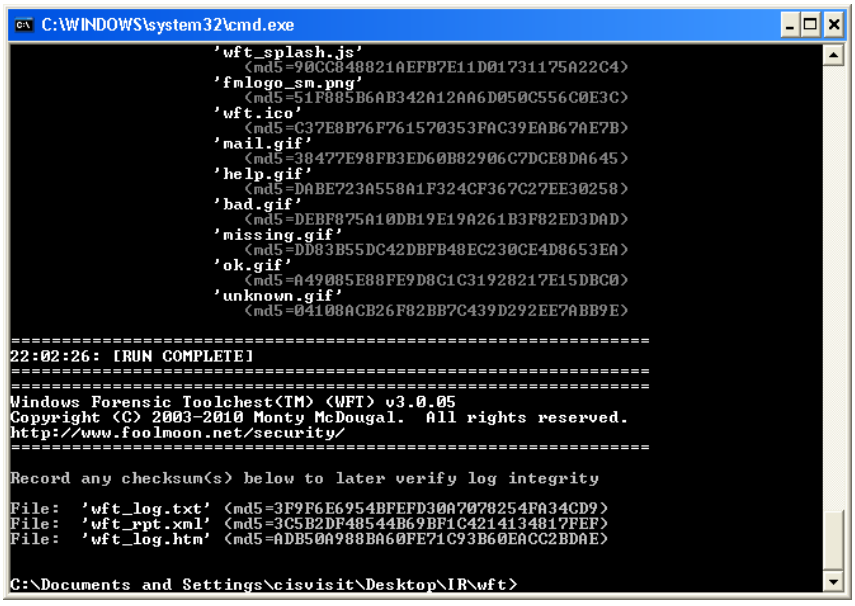


Figure A2.5b: Screenshot of the WFT.exe excution of SQL IR scripts

21. After running the pilot test of the extended WFT, the results could be found in the index.htm (HTML document) files within the configured output directory during the execution (Fowler, 2009), see Figure A2.6. Hence, the Figure A2.7 is the results of the pilot test of extended WFT with integrated SQL Server IR scripts.

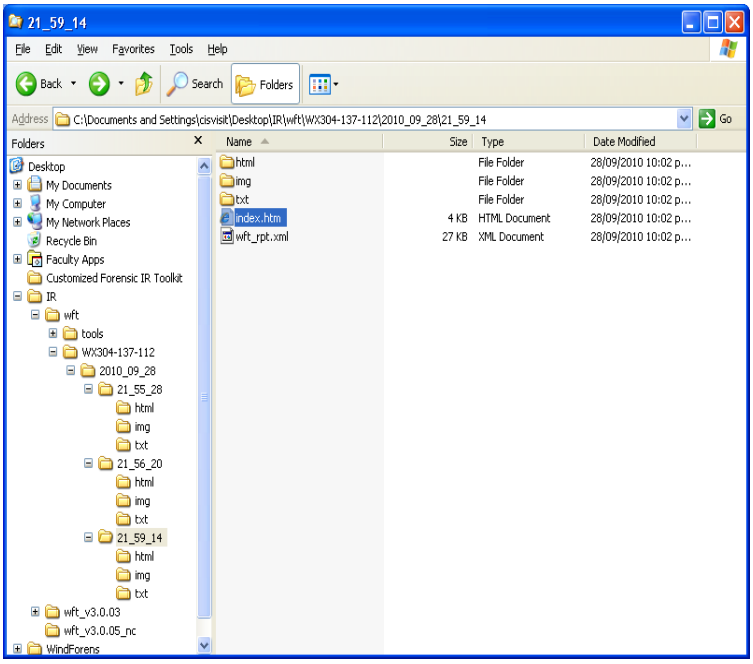


Figure A2.6: Screenshot of the index.htm file during the pilot test of extended WFT tool

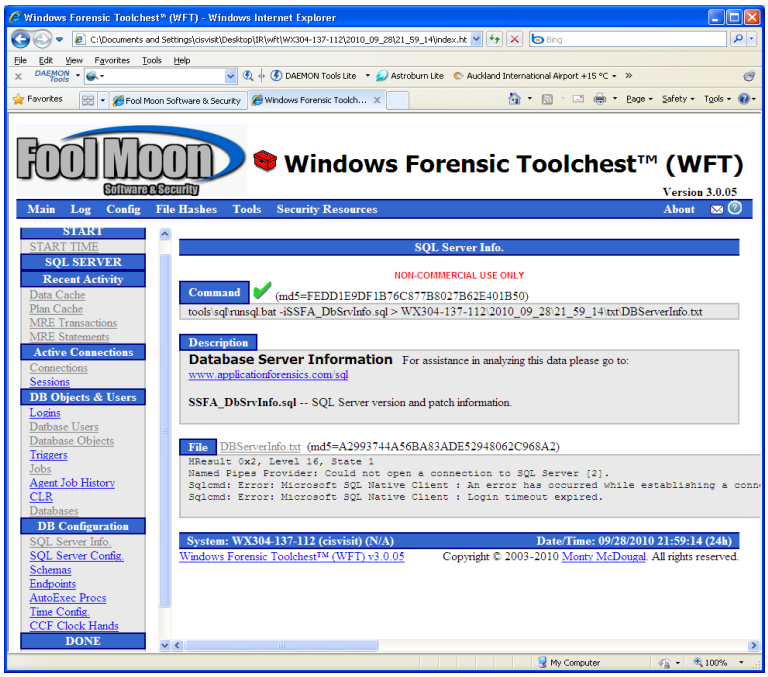


Figure A2.7: Screenshot of the extended WFT reporting interface

22. Although the extended version of SQL Server of WFT is being created successfully at this point, a complete RFID Incident Response DVD toolkit has to be created.

23. Thus, the next step is to copy RAM folder from Helix3 into created "IR" folder (see step 4 above) and the Guidance Software's WinEn is also copied into that RAM folder under IR folder.

24. Likewise, the developed ReaderLogExtraction Tool (see Appendix 3; it is developed for the purpose of acquiring bit-to-bit evidence data from RFID reader's memory during the investigation) folder is placed into the IR folder.

25. At this stage, the IR folder for RFID BS is completely ready and is needed to be replaced with IR folder from the downloaded original Helix3.iso (version: 2009R1).

26. Thus, the existing IR folder from Helix 3 is deleted and replaced with the newly created IR folder from the desktop.

27. Finally, the Helix_RFID_IR DVD Toolkit is successfully created as shown in the following figures (Figures A2.8 and A2.9).

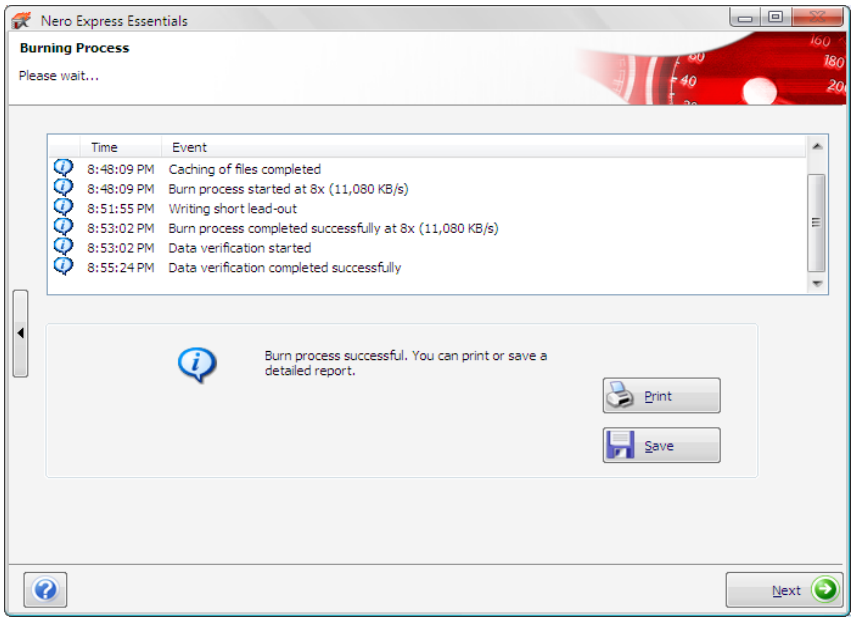


Figure A2.8: Screenshot of the burnning process of Helix_RFID_IR DVD Toolkit

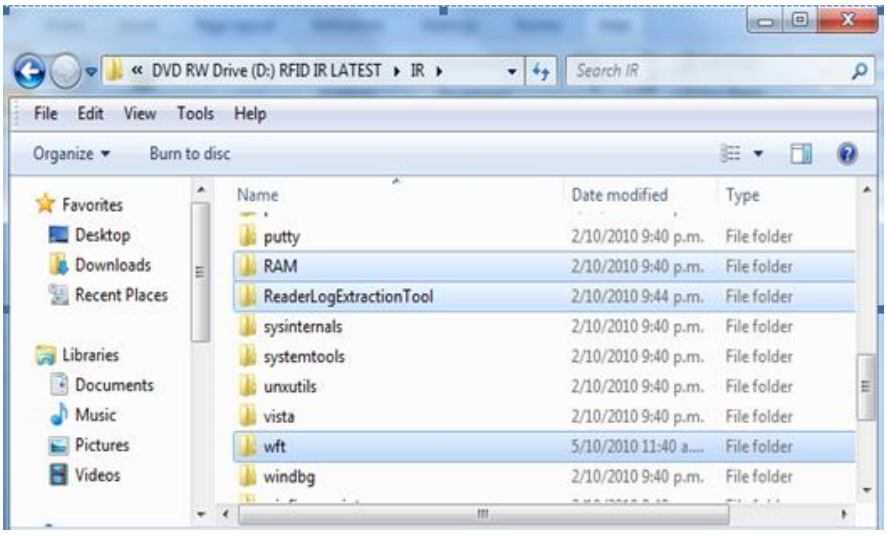


Figure A2.8: Screenshot of the integrated folders in the developed Helix_RFID_IR DVD Toolkit

Appendix 3: Source Code of the Developed RFID ReaderLogExtractionTool

The following is the source code of developed the RFID **ReaderLogExtraction** tool, which is adapted under **NameSpace Ex1:** from the Software Development Kit (SDK) of RFID reader's manufacturer company (Tracient Technologies Ltd., http://www.tracient.com), for the purpose of acquiring bit-to-bit evidence data from RFID reader's memory during the investigation.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;   // for Marshal
using System.Data.SqlClient;
using System.IO;

using Tracient.RFID;

namespace Example1
{
   public partial class Example1 : Form
   {
      RfidUsb usbRFID = new RfidUsb();
      IntPtr hReader;

      public Example1()
```

```
{
    InitializeComponent();

    hReader = IntPtr.Zero;

    uint numDevices = usbRFID.GetNumDevices(false);
    this.RefreshList(numDevices);

    btnDisConnect.Enabled = false;
    btnDownloadLog.Enabled = false;
    btnEraseLog.Enabled = false;

}

private void btnRefresh_Click(object sender, EventArgs e)
{
    this.Cursor = Cursors.WaitCursor;

    uint numDevices = usbRFID.GetNumDevices(true);
    this.RefreshList(numDevices);

    this.Cursor = Cursors.Default;
}

private void btnConnect_Click(object sender, EventArgs e)
{
    if (hReader == IntPtr.Zero)
    {
        int dev = this.cmbReaders.SelectedIndex;
        Object item = this.cmbReaders.SelectedItem;
        if (item.ToString() == "No Readers")
```

```
            {
                MessageBox.Show("No readers available.");
            }
            else
            {
                this.Cursor = Cursors.WaitCursor;

                try
                {   // Open the connection & subscribe to notifications
                    hReader = usbRFID.Open((uint)dev);
                    // Assign the callback delegates we want to use
                    usbRFID.OnConnectionLost += new
RfidUsb.ConnectionLostEventDelegate(OnConnectionLost);
                    usbRFID.OnAsyncRead += new
RfidUsb.AsyncReadEventDelegate(OnAsyncRead);

                    // Read the current Auto-Read setting (fContinuousRead)
                    SDKType.RFID_CONFIG_READER Conf = new
SDKType.RFID_CONFIG_READER();
                    System.UInt32 dwFlags =
(uint)SDKType.ConfigFlag.RFID_CONFIG_GET |
(uint)SDKType.ConfigFlag.RFID_FLG_READER_AUTOREAD;

                    usbRFID.ConfigReader(hReader, ref Conf, ref dwFlags);

                    // Set-up the UI
                    btnConnect.Enabled = false;
                    btnDisConnect.Enabled = true;

                    btnDownloadLog.Enabled = true;
                    btnEraseLog.Enabled = true;
```

```
                tmrHalfSec.Enabled = true;
                lstTagIDs.Items.Clear();
                txtNumDownloaded.Text = 0.ToString();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                this.Cursor = Cursors.Default;
            }
        }
    }
    else
    {
        MessageBox.Show(this, "Reader already connected!", "Example1",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}

private void btnDisConnect_Click(object sender, EventArgs e)
{
    this.Cursor = Cursors.WaitCursor;

    if (hReader != IntPtr.Zero)
    {
        usbRFID.Close(hReader);
        hReader = IntPtr.Zero;
        usbRFID.OnConnectionLost -= OnConnectionLost;
        usbRFID.OnAsyncRead -= OnAsyncRead;
```

```
    }
    btnConnect.Enabled = true;
    btnDisConnect.Enabled = false;

    btnDownloadLog.Enabled = false;
    btnEraseLog.Enabled = false;

    tmrHalfSec.Enabled = false;
    //txtNumEntries.Text = 0.ToString();

    this.Cursor = Cursors.Default;
}

private void Example1_Closing(object sender, FormClosingEventArgs e)
{
    if (hReader != IntPtr.Zero)
    {
        usbRFID.Close(hReader);
    }
}

/// <summary>
/// Update the contents of the list box
/// </summary>
private void RefreshList(uint numDevices)
{
    this.cmbReaders.Items.Clear();

    try
    {
        string desc;
```

```
            this.cmbReaders.BeginUpdate();
            if (numDevices == 0)
            {
                desc = "No Readers";
                this.cmbReaders.Items.Insert(0, desc);
            }
            for (uint dev = 0; dev < numDevices; dev++)
            {
                desc = usbRFID.GetDeviceString(dev);
                this.cmbReaders.Items.Insert((int)dev, desc);
            }
            this.cmbReaders.EndUpdate();
            this.cmbReaders.SelectedIndex = 0;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    // Define the Delegate and Method for detecting unplugged USB
    // from our callback (made in a different thread).
    private delegate void ConnectionLostDelegate(System.IntPtr cyDevice);

    // Our method for updating the Example1 class
    private void ConnectionLostMethod(System.IntPtr cyDevice)
    {
        if (cyDevice == hReader)
        {
            hReader = IntPtr.Zero;
```

```
            MessageBox.Show(this, "Connection to Reader has been lost or Reader has
been turned off!", "Notification", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
            usbRFID.OnConnectionLost -= OnConnectionLost;
            usbRFID.OnAsyncRead -= OnAsyncRead;

            btnConnect.Enabled = true;
            btnDisConnect.Enabled = false;

            btnDownloadLog.Enabled = false;
            btnEraseLog.Enabled = false;
            tmrHalfSec.Enabled = false;
        }
    }

    // Our delegate that gets called by the RfidUsb class
    void OnConnectionLost(System.IntPtr cyDevice)
    {   // Note that the callback is made from the RFID DLLs worker thread,
        // and it invokes this delegate on a Thread from the Thread Pool.
        // So we don't have direct access to the Form here, we have to use
        // another delegate to get to the Form.
        this.Invoke(new ConnectionLostDelegate(ConnectionLostMethod),
                    new Object [] { cyDevice });
    }

    // Define the Delegate and Method for detecting asynchronous reads
    // from our callback (made in a different thread).
    private delegate void AsyncReadDelegate(System.IntPtr cyDevice, string
TagID);

    // Our method for updating the Example1 class
```

```
private void AsyncReadMethod(System.IntPtr cyDevice, string TagID)
{
   if (cyDevice == hReader)
   {
      lstTagIDs.Items.Add(TagID);
   }
}

// Our delegate that gets called by the RfidUsb class
void OnAsyncRead(System.IntPtr cyDevice, StringBuilder strTagID)
{   // Note that the callback is made from the RFID DLLs worker thread,
   // and it invokes this delegate on a Thread from the Thread Pool.
   // So we don't have direct access to the Form here, we have to use
   // another delegate to get to the Form.
   this.Invoke(new AsyncReadDelegate(AsyncReadMethod),
               new Object[] { cyDevice, strTagID.ToString() });
}

private void btnOK_Click(object sender, EventArgs e)
{
   //this.Close();
   save_log_to_bin();
}

private void tmrHalfSec_Tick(object sender, EventArgs e)
{
   if (hReader != IntPtr.Zero)
   {
      try
      {
         uint dwNumEntries;
```

```
                dwNumEntries = usbRFID.GetNumberLogEntries(hReader);
                //this.txtNumEntries.Text = dwNumEntries.ToString();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
                tmrHalfSec.Enabled = false;
            }
        }
        else
        {
            tmrHalfSec.Enabled = false;
        }
    }

    private void btnDownloadLog_Click(object sender, EventArgs e)
    {
        SDKType.RFID_LOG_ITEM LogItem = new
SDKType.RFID_LOG_ITEM();
        System.Runtime.InteropServices.ComTypes.FILETIME ftLocalTime = new
System.Runtime.InteropServices.ComTypes.FILETIME();
        RfidUsb.SYSTEMTIME stTime = new RfidUsb.SYSTEMTIME();

        String strLogEntry;
        uint uiNumDownloaded = 0;

        string LogitemTime;

        bool fMoreLogEntries = true;
        if (hReader != IntPtr.Zero)
```

```
{
    // Temporarily stop the timer while we download
    tmrHalfSec.Enabled = false;

    // Clear the existing display
    lstTagIDs.Items.Clear();
    txtNumDownloaded.Clear();
    btnDownloadLog.Text = "Downloading...";
    this.Refresh();

    Cursor.Current = Cursors.WaitCursor;
    try
    {
        fMoreLogEntries = usbRFID.LogGetFirst(hReader, ref LogItem);
        // First entry got OK, now we loop calling LogGetNext till
        // we get to the end of the log
        while (fMoreLogEntries)
        {
            if ((LogItem.ftCreationTime.dwHighDateTime != 0) &&
                (LogItem.ftCreationTime.dwLowDateTime != 0))
            {
                RfidUsb.NativeTimeMethods.FileTimeToLocalFileTime
                        (ref LogItem.ftCreationTime, out ftLocalTime);
                RfidUsb.NativeTimeMethods.FileTimeToSystemTime
                        (ref ftLocalTime, out stTime);
            }

            // now create the string with tag ID and local time
            strLogEntry = LogItem.szIDBuffer;
            strLogEntry = String.Format("{0}\t{1:00}:{2:00}:{3:00}
{4:00}/{5:00}/{6:00}",
```

```
                                    LogItem.szIDBuffer,
                                    stTime.wHour, stTime.wMinute, stTime.wSecond,
                                    stTime.wDay, stTime.wMonth, stTime.wYear);
                    // add it to the list box
                    lstTagIDs.Items.Add(strLogEntry);
                    LogitemTime = stTime.wYear+"-"+stTime.wMonth+"-
"+stTime.wDay+" "+ stTime.wHour+":"+stTime.wMinute+":"+stTime.wSecond;

                    // increment the downloaded count
                    uiNumDownloaded++;
                    txtNumDownloaded.Text = uiNumDownloaded.ToString();

                    // get the next entry
                    fMoreLogEntries = usbRFID.LogGetNext(hReader, ref LogItem);
                }
                // We've been told there are no more entries...

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {   // tidy everything up
                btnDownloadLog.Text = "Download Log";

                // close the log
                usbRFID.LogClose(hReader);

                // restart the timer
                tmrHalfSec.Enabled = true;
```

```
            // update the cursor
            Cursor.Current = Cursors.Default;
        }

    }
}

private void btnEraseLog_Click(object sender, EventArgs e)
{
    if (hReader != IntPtr.Zero)
    {
        try
        {
            usbRFID.LogEraseFromDevice(hReader);

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}


private void save_log_to_bin()
{
    try
    {
        SaveFileDialog fdialog = new SaveFileDialog();
        fdialog.ShowDialog();
```

```
            if(fdialog.FileName != "")
            {

                FileStream SW = new FileStream(fdialog.FileName,
FileMode.CreateNew, FileAccess.ReadWrite);
                BinaryWriter bwriter = new BinaryWriter(SW);

                foreach (string item in lstTagIDs.Items)
                {
                   bwriter.Write(item);
                }

                bwriter.Close();
                SW.Close();
            }
        }
        catch(FileNotFoundException fileEx)
        {
            MessageBox.Show(fileEx.Message);
            return;
        }

    }
  }
}
```

Appendix 4: Customized RFID Middleware Source Code

The following is the source code of R/W RFID middleware software, which is adapted under **NameSpace Ex1:** from the Software Development Kit (SDK) of RFID reader's manufacturer company (Tracient Technologies Ltd., http://www.tracient.com), according to our experiment requirement. It allows users to configure RFID reader, to collect the data on tag, and write the tag data to the backend database. Basically, it enables reading and writing the tag data from the RFID scanner to the BIS SQL 2005 Server (vice versa).

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;   // for Marshal
using System.Data.SqlClient;

using Tracient.RFID;

namespace Example1
{
   public partial class Example1 : Form
   {
      RfidUsb usbRFID = new RfidUsb();
      IntPtr hReader;
      private SqlConnection Con;
      //private string constr=@"Data
Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\RFID_test.mdf;Inte
grated Security=True;User Instance=True";
```

```
        private string constr = @"Data Source=localhost\SQLEXPRESS;Initial
Catalog=RFID_test;Integrated Security=True";

        public Example1()
        {
            InitializeComponent();

            hReader = IntPtr.Zero;

            uint numDevices = usbRFID.GetNumDevices(false);
            this.RefreshList(numDevices);
            this.cmbMemoryType.SelectedIndex = 0;

            btnDisConnect.Enabled = false;
            btnReadTag.Enabled = false;

        }

        private void btnRefresh_Click(object sender, EventArgs e)
        {
            this.Cursor = Cursors.WaitCursor;

            uint numDevices = usbRFID.GetNumDevices(true);
            this.RefreshList(numDevices);

            this.Cursor = Cursors.Default;
        }

        private void btnConnect_Click(object sender, EventArgs e)
        {
            if (hReader == IntPtr.Zero)
            {
                int dev = this.cmbReaders.SelectedIndex;
                Object item = this.cmbReaders.SelectedItem;
```

```
            if (item.ToString() == "No Readers")
            {
                MessageBox.Show("No readers available.");
            }
            else
            {
                this.Cursor = Cursors.WaitCursor;

                try
                {   // Open the connection & subscribe to notifications
                    hReader = usbRFID.Open((uint)dev);
                    // Assign the callback delegates we want to use
                    usbRFID.OnConnectionLost += new
RfidUsb.ConnectionLostEventDelegate(OnConnectionLost);
                    usbRFID.OnAsyncRead += new
RfidUsb.AsyncReadEventDelegate(OnAsyncRead);

                    // Read the current Auto-Read setting (fContinuousRead)
                    SDKType.RFID_CONFIG_READER Conf = new
SDKType.RFID_CONFIG_READER();
                    System.UInt32 dwFlags =
(uint)SDKType.ConfigFlag.RFID_CONFIG_GET |
(uint)SDKType.ConfigFlag.RFID_FLG_READER_AUTOREAD;

                    usbRFID.ConfigReader(hReader, ref Conf, ref dwFlags);

                    // Set-up the UI
                    btnConnect.Enabled = false;
                    btnDisConnect.Enabled = true;
                    btnReadTag.Enabled = true;
                    tmrHalfSec.Enabled = true;
                    lstTagIDs.Items.Clear();
```

```
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                finally
                {
                    this.Cursor = Cursors.Default;
                }
            }
        }
        else
        {
            MessageBox.Show(this, "Reader already connected!", "Example1",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }

    private void btnDisConnect_Click(object sender, EventArgs e)
    {
        this.Cursor = Cursors.WaitCursor;

        if (hReader != IntPtr.Zero)
        {
            usbRFID.Close(hReader);
            hReader = IntPtr.Zero;
            usbRFID.OnConnectionLost -= OnConnectionLost;
            usbRFID.OnAsyncRead -= OnAsyncRead;
        }
        btnConnect.Enabled = true;
        btnDisConnect.Enabled = false;
        btnReadTag.Enabled = false;

        tmrHalfSec.Enabled = false;
```

```
            //txtNumEntries.Text = 0.ToString();

            this.Cursor = Cursors.Default;
        }

        private void btnReadTag_Click(object sender, EventArgs e)
        {
            string readresult;
            if (hReader != IntPtr.Zero)
            {
                this.Cursor = Cursors.WaitCursor;

                try
                {
                    string TagID;
                    string TagValue;
                    DateTime currentTime = new DateTime();
                    currentTime = DateTime.Now;
                    TagID = usbRFID.Read(hReader);

                    if (TagID.Length > 0)
                    {
                        //startindex ~~~~
                        TagValue = readDataFromTag(TagID);

                        readresult = String.Format("{0}\t{1}\t{2:00}:{3:00}:{4:00}
{5:00}/{6:00}/{7:00}",
                                                   TagID,TagValue,
                                                   currentTime.Hour, currentTime.Minute,
currentTime.Second,
                                                   currentTime.Day, currentTime.Month,
currentTime.Year);

                        lstTagIDs.Items.Add(readresult);
```

```
                insertTagtoDB(TagID, TagValue, currentTime);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        finally
        {
            this.Cursor = Cursors.Default;
        }
    }
}

private string readDataFromTag(string TagID)
{
    string value = null;
    if (hReader != IntPtr.Zero)
    {
        this.Cursor = Cursors.WaitCursor;

        System.IntPtr lpTagData = IntPtr.Zero;

        try
        {
            // Declare our variables
            byte[] abTagDataBytes;
            string strTagData;

            uint uiBytesRead = 0;
            uint uiAddress = 0;
            uint uiBytes = 0;

            // Get the tag ID
```

```
                StringBuilder strTagID = new System.Text.StringBuilder(TagID);

                // Get the memory type they have selected
                Object itemMemType = this.cmbMemoryType.SelectedItem;

                // Get the address and number of bytes to read
                uiAddress = (uint)numStartBlock.Value;
                uiBytes = (uint)numBytesToRead.Value;

                // Sanity checks
                if (uiBytes == 0)
                {
                    MessageBox.Show("Refusing to read 0 bytes.");
                }
                else
                {
                    // Allocate unmanaged memory used to store tag data
                    lpTagData = Marshal.AllocHGlobal((int)uiBytes);

                    // call the correct readtag function
                    switch (itemMemType.ToString())
                    {
                        case ("User"):
                            {
                                uiBytesRead = usbRFID.ReadTag_UserMemory(hReader,
strTagID,
                                                                         lpTagData, uiAddress, uiBytes);
                                break;
                            }

                        case ("OTP"):
                            {
                                uiBytesRead = usbRFID.ReadTag_OTPMemory(hReader,
strTagID,
```

```
                                                    lpTagData, uiAddress, uiBytes);
                    break;
                }

            case ("Protection"):
                {
                    uiBytesRead =
usbRFID.ReadTag_ProtectionMemory(hReader, strTagID,
                                                        lpTagData, uiAddress,
uiBytes);
                    break;
                }

            case ("Read Only"):
                {
                    uiBytesRead =
usbRFID.ReadTag_ProtectionMemory(hReader, strTagID,
                                                        lpTagData, uiAddress,
uiBytes);
                    break;
                }


            case ("Manufacturer"):
                {
                    uiBytesRead =
usbRFID.ReadTag_ManufacturerMemory(hReader, strTagID,
                                                         lpTagData, uiAddress,
uiBytes);
                    break;
                }

            default:
                {
```

```
                    break;
                }
            }

            // Now display the bytes we read as text
            // for this example we are going to assume the data was ascii
            abTagDataBytes = new byte[uiBytesRead];
            Marshal.Copy(lpTagData, abTagDataBytes, 0, (int)uiBytesRead);

            System.Text.ASCIIEncoding asciiEncoder = new
System.Text.ASCIIEncoding();
            strTagData = asciiEncoder.GetString(abTagDataBytes);

            //value = strTagData;
            value = strTagData.Remove(strTagData.IndexOf('\u0000'));
            txtBytesRead.Text = uiBytesRead.ToString();
        }
    }
    catch (Exception ex)
    {
        // Error occurred - clear the data, and set bytes read to zero.
        //txtReadData.Text = "";
        txtBytesRead.Text = 0.ToString();

        MessageBox.Show(ex.Message);
    }
    finally
    {
        // De-allocate unmanaged memory used to store tag data
        if (lpTagData != IntPtr.Zero)
        {
            Marshal.FreeHGlobal(lpTagData);
        }
```

```
            //this.Cursor = Cursors.Default;
        }
    }
    return value;
}


private void btnClearList_Click(object sender, EventArgs e)
{
    lstTagIDs.Items.Clear();
}

private void Example1_Closing(object sender, FormClosingEventArgs e)
{
    if (hReader != IntPtr.Zero)
    {
        usbRFID.Close(hReader);
    }
}

/// <summary>
/// Update the contents of the list box
/// </summary>
private void RefreshList(uint numDevices)
{
    this.cmbReaders.Items.Clear();

    try
    {
        string desc;
        this.cmbReaders.BeginUpdate();
        if (numDevices == 0)
        {
            desc = "No Readers";
```

```
                this.cmbReaders.Items.Insert(0, desc);
            }
            for (uint dev = 0; dev < numDevices; dev++)
            {
                desc = usbRFID.GetDeviceString(dev);
                this.cmbReaders.Items.Insert((int)dev, desc);
            }
            this.cmbReaders.EndUpdate();
            this.cmbReaders.SelectedIndex = 0;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    // Define the Delegate and Method for detecting unplugged USB
    // from our callback (made in a different thread).
    private delegate void ConnectionLostDelegate(System.IntPtr cyDevice);

    // Our method for updating the Example1 class
    private void ConnectionLostMethod(System.IntPtr cyDevice)
    {
        if (cyDevice == hReader)
        {
            hReader = IntPtr.Zero;
            MessageBox.Show(this, "Connection to Reader has been lost or Reader
has been turned off!", "Notification", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
            usbRFID.OnConnectionLost -= OnConnectionLost;
            usbRFID.OnAsyncRead -= OnAsyncRead;

            btnConnect.Enabled = true;
            btnDisConnect.Enabled = false;
```

```
            btnReadTag.Enabled = false;

            tmrHalfSec.Enabled = false;
        }
    }

    // Our delegate that gets called by the RfidUsb class
    void OnConnectionLost(System.IntPtr cyDevice)
    {   // Note that the callback is made from the RFID DLLs worker thread,
        // and it invokes this delegate on a Thread from the Thread Pool.
        // So we don't have direct access to the Form here, we have to use
        // another delegate to get to the Form.
        this.Invoke(new ConnectionLostDelegate(ConnectionLostMethod),
                    new Object [] { cyDevice });
    }

    // Define the Delegate and Method for detecting asynchronous reads
    // from our callback (made in a different thread).
    private delegate void AsyncReadDelegate(System.IntPtr cyDevice, string
TagID);

    // Our method for updating the Example1 class
    private void AsyncReadMethod(System.IntPtr cyDevice, string TagID)
    {
        if (cyDevice == hReader)
        {
            lstTagIDs.Items.Add(TagID);
        }
    }

    // Our delegate that gets called by the RfidUsb class
    void OnAsyncRead(System.IntPtr cyDevice, StringBuilder strTagID)
    {   // Note that the callback is made from the RFID DLLs worker thread,
        // and it invokes this delegate on a Thread from the Thread Pool.
```

```
    // So we don't have direct access to the Form here, we have to use
    // another delegate to get to the Form.
    this.Invoke(new AsyncReadDelegate(AsyncReadMethod),
                new Object[] { cyDevice, strTagID.ToString() });
}

private void btnOK_Click(object sender, EventArgs e)
{
    this.Close();
}

private void tmrHalfSec_Tick(object sender, EventArgs e)
{
    if (hReader != IntPtr.Zero)
    {
        try
        {
            uint dwNumEntries;

            dwNumEntries = usbRFID.GetNumberLogEntries(hReader);
            //this.txtNumEntries.Text = dwNumEntries.ToString();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
            tmrHalfSec.Enabled = false;
        }
    }
    else
    {
        tmrHalfSec.Enabled = false;
    }
}
```

```
        private void insertTagtoDB(string Tag, string TagValue, DateTime
currentTime)
        {
            string ctime;
            ctime = currentTime.ToString("yyyy-MM-dd HH:mm:ss");
            string strsql = "Insert into rfid_db (Tag, Value, Date) values('" + Tag + "','"
+ TagValue + "', '" + ctime + "');";

            //mysql_real_escape_string(strsql);
            string strsql2 = strsql;
            strsql2 = strsql2.Replace("'", " ");
            strsql2 = strsql2.Replace(";", " ");
            strsql2 = strsql2.Replace(",", " ");
            strsql2 = strsql2.Replace("(", " ");
            strsql2 = strsql2.Replace(")", " ");
            //MessageBox.Show(strsql);
            insertLogtoDB(Tag, strsql2, ctime);
            Con = new SqlConnection(constr);
            Con.Open();
            SqlCommand sqlCmd = new SqlCommand(strsql, Con);
            sqlCmd.ExecuteNonQuery();

            Con.Close();
        }

        private void insertLogtoDB(string logItem, string userdata, string logdate)
        {
            Con = new SqlConnection(constr);
            Con.Open();
            string strSql = "Insert into rfid_log (Tag, User_data, Date) values('" +
logItem + "', '" + userdata + "', '" + logdate + "');";
            //MessageBox.Show(strSql);
            SqlCommand sqlCmd = new SqlCommand(strSql, Con);
            sqlCmd.ExecuteNonQuery();
```

```
    Con.Close();
}

private void btnDBload_Click(object sender, EventArgs e)
{
    DataTable dt;
    SqlDataAdapter da;

    Con = new SqlConnection(constr);

    try
    {
        da = new SqlDataAdapter("Select * from rfid_db", Con);
        dt = new DataTable("rfid_db");

        da.Fill(dt);
        dt.DefaultView.Sort = "Date desc";
        dataGridView1.DataSource = dt;
        dataGridView1.Columns["Value"].HeaderText = "Value ($)";

    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}

private void btnLOGload_Click(object sender, EventArgs e)
{
}

private void btnClearGird_Click(object sender, EventArgs e)
{
    dataGridView1.Columns.Clear();
```

```
        }

        private void btnLOGload_Click_1(object sender, EventArgs e)
        {
            DataTable dt;
            SqlDataAdapter da;

            Con = new SqlConnection(constr);

            try
            {
                da = new SqlDataAdapter("Select * from rfid_log", Con);
                dt = new DataTable("rfid_log");

                da.Fill(dt);
                dt.DefaultView.Sort = "Date desc";
                dataGridView1.DataSource = dt;
                //MessageBox.Show("DB loaded");
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

    }
}
```

Appendix 5: Tripwire Setup

After installing all the required software, such as the backend Microsoft SQL Server 2005 and SQL Server Management Studio Express, on a test-bed point-of-sale workstation (POS TEST-STATION), the next step is to install the system integrity checking software (Tripwire for Servers and Tripwire for Manager). Hence the following steps are the installation steps of Tripwire software.

Installation of Tripwire for Server – Version 4.8

Step 1: Create the Tripwire initial policy file.

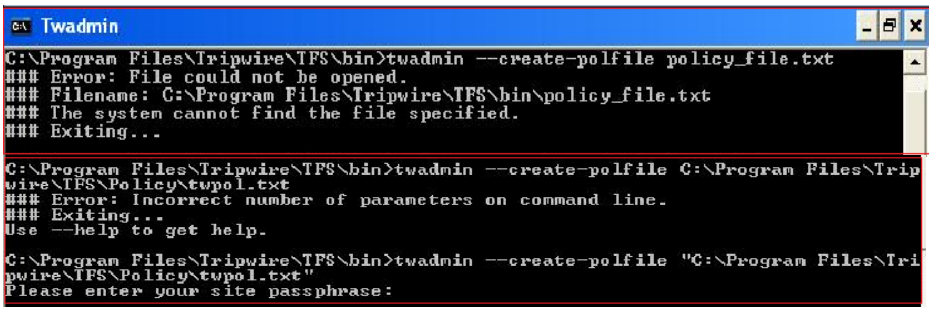


Figure A5. 1: Tripwire initial policy file creation on POS TEST-STATION

Step 2: The policy file is encoded and installed after enter the site passphrase, 'qwe123AUT'.

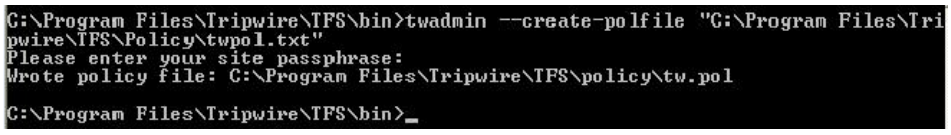
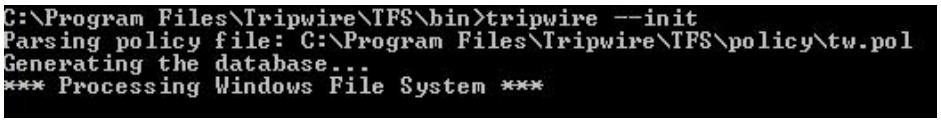


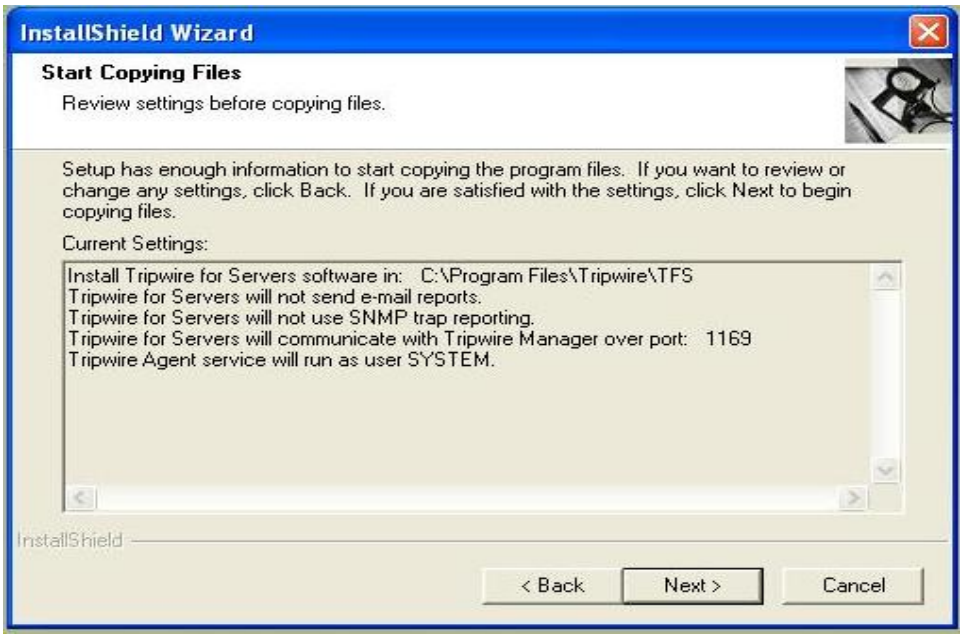
Figure A5. 2: Tripwire initial policy file was encoded and installed on TEST-STATION POS

Step 3: Initialize the Tripwire database file.



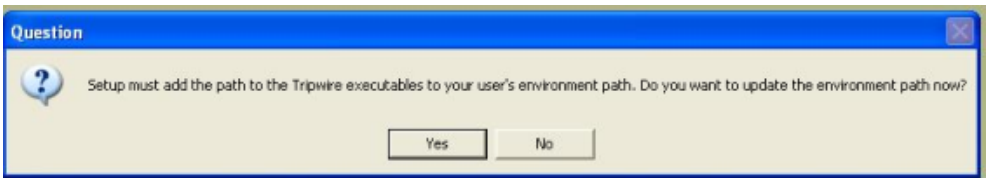
After initializing the database file, the local passphrase must be entered for the Tripwire database security. Hence, the Tripwire database was successfully generated after keying the passphrase (which is also 'qwe123AUT').

Step 4: Then, the Tripwire will perform the integrity checking of the system and review the settings before copying the installation files.



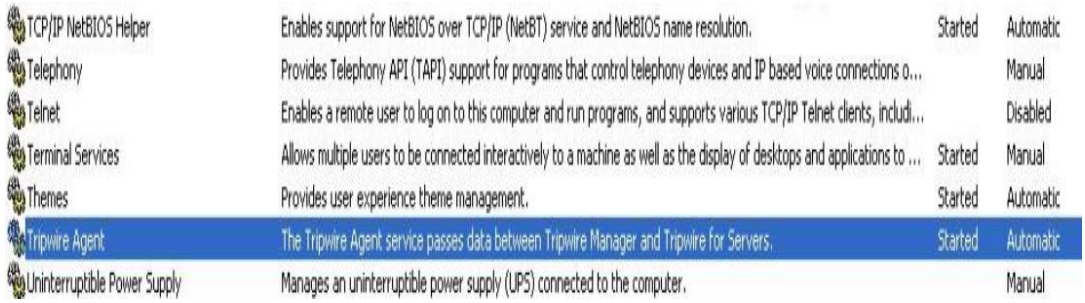
Afterwards, the Tripwire will report the integrity checking.

Step 5: Start Tripwire Agent service when asked and update the environment path to the Tripwire executables.

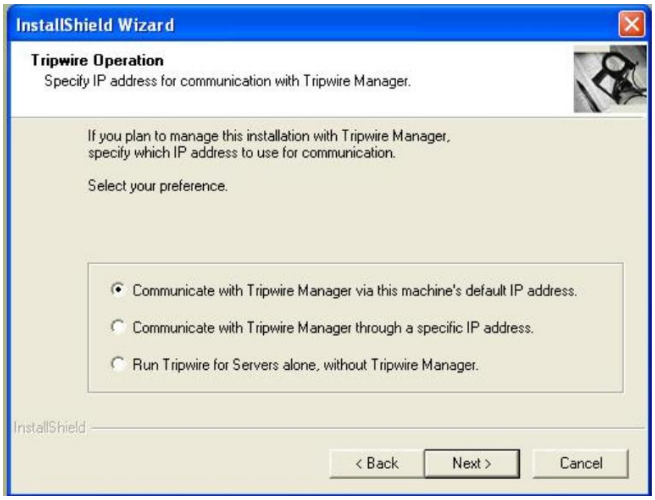


Step 6: Then, the Tripwire for Servers installation is completed.

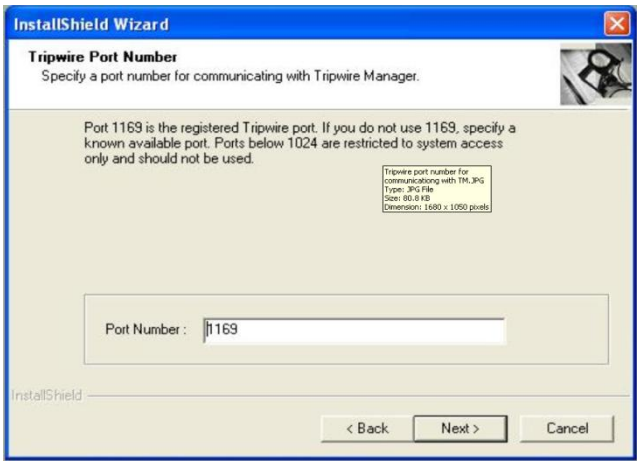
Step 7: Check whether the Tripwire for Servers is started on the POS TEST-STATION. Hence, the Tripwire for Servers has been successfully installed on the POS TEST-STATION.



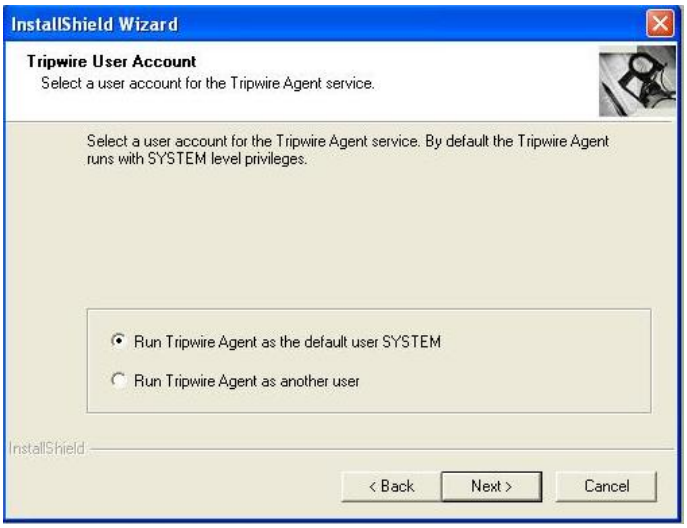
However, the specific assignment of IP address is needed during Tripwire for Servers installation in order to communicate with Tripwire Manager via this machine.



Likewise, the port number should also be assigned accordingly.



Similarly, Tripwire user account should be selected as a default user SYSTEM.



Installation of the Tripwire Manager – Version 4.8

The installation of the Tripwire Manager ™ software is very much straight forward. In fact, the simple instructions are needed to follow in order to install the Tripwire Manager. The following are the screenshots of TM installation.

Figure A5.1: Installation of the Tripwire Manager – Screenshot 1

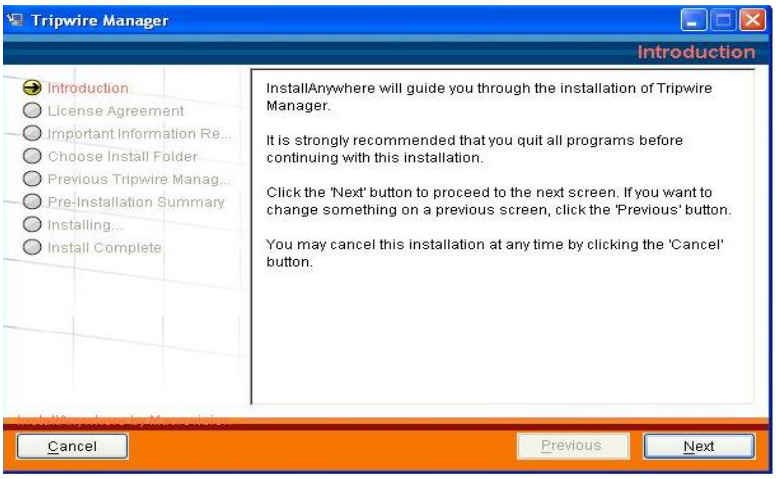


Figure A5.2: Installation of the Tripwire Manager – Screenshot 2

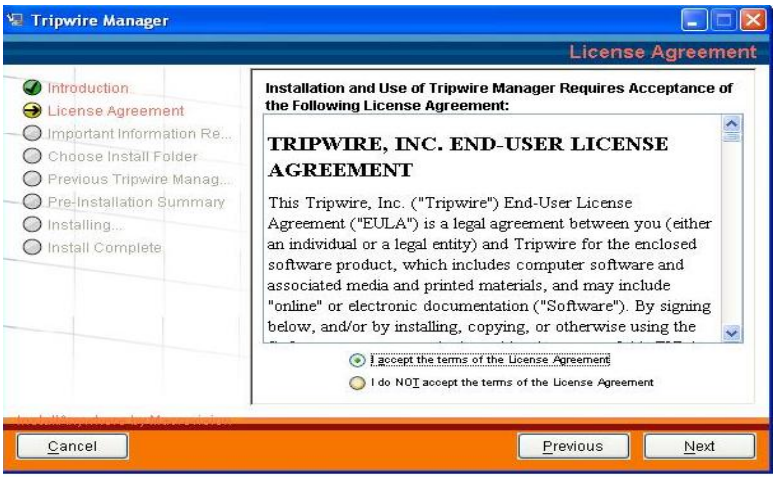


Figure A5.3: Installation of the Tripwire Manager – Screenshot 3

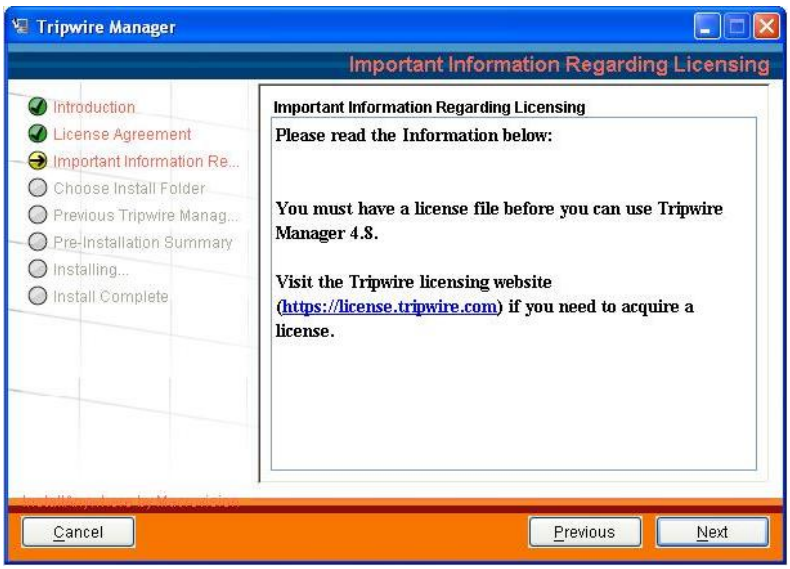


Figure A5.4: Installation of the Tripwire Manager – Screenshot 4

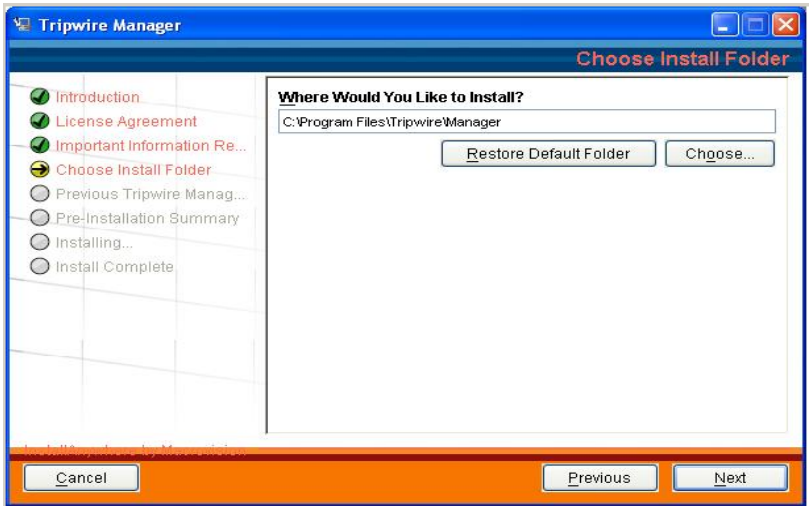


Figure A5.5: Installation of the Tripwire Manager – Screenshot 5

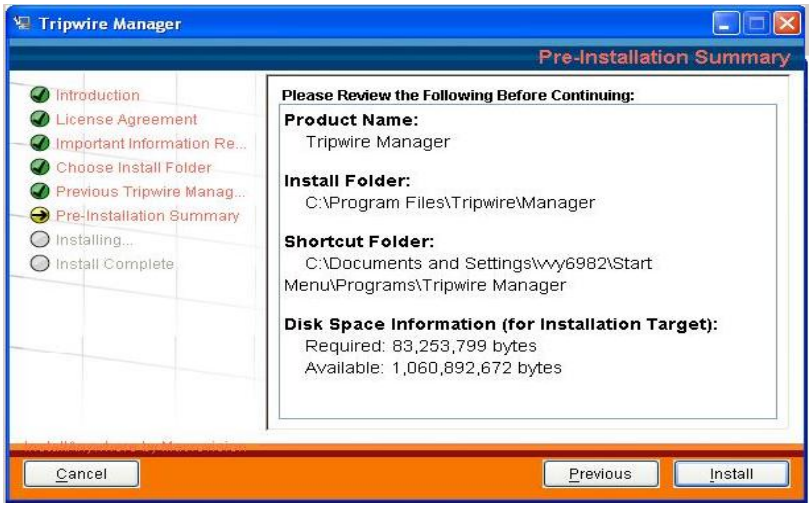


Figure A5.6: Installation of the Tripwire Manager – Screenshot 6

Figure A5.7: Installation of the Tripwire Manager – Screenshot 7

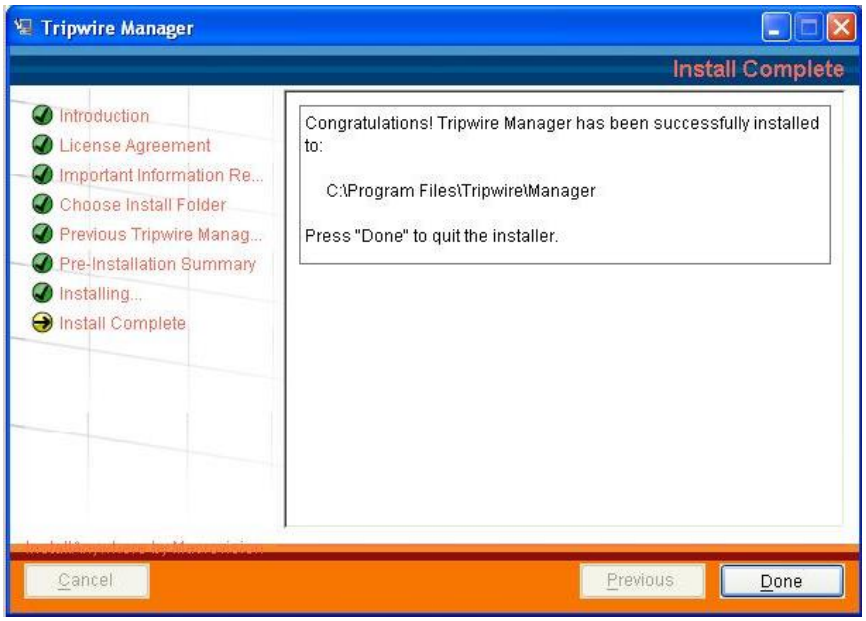


Figure A5.8: Installation of the Tripwire Manager – Screenshot 8

Finally, the Tripwire Manager has been installed successfully. However, the POS machine has to be added to the TM during the set up.

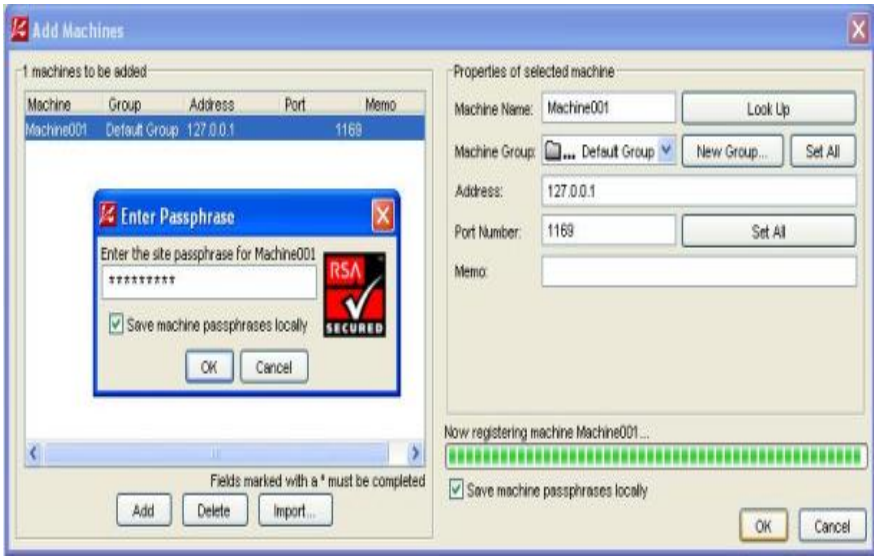


Figure A5.9: Installation of the Tripwire Manager – Screenshot 9

Appendix 6: An Example of the Conducted Pilot Attack on the Simulated Business System (RFID BS) and Findings after the Attack

The focus of conducting an example experiment is to investigate for the evidence remaining after a BS is violated by SQL injection attack.

1. Experiment setting

In the following sections, the complete BS system used in the experiment (as shown in Figure A6.1 below) will be described. The setting has a Tracient's Padl-R UF RFID reader, a tag, one host controller or personal computer (PC) running applications such as Microsoft Windows XP Service Pack 2, the developed middleware, the SQL Server 2005, the SQL Server Management Studio Express, the Tripwire for Servers, the Tripwire Managers and the software development kit (SDK) from the RFID reader's manufacturing company, Tracient Technologies Limited. It will also describe a prototype application for writing logs from RFID reader to the backend server.

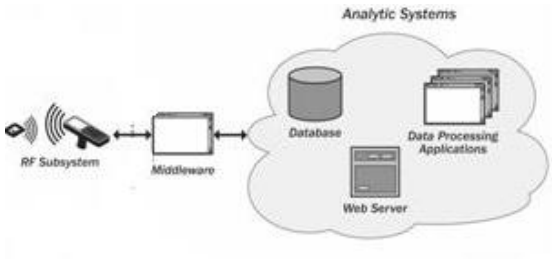


Figure A6.1: Enterprise sub-system in RFID system architecture (Adapted from Karygiannis et al., 2007, pp. 2-15)

RFID reader

The RFID reader used in the experiment is Tracient RFID reader, which is connected through a universal serial bus (USB) to a host PC. The Tracient reader is a Padl-R UF handheld type and further technical details can be found in company website (http://www.tracient.com/nbspnbspnbsp--uhf-readers.html).

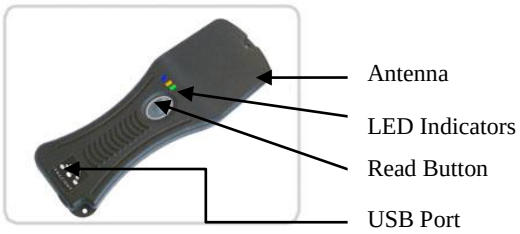


Figure A6.2: Padl-R UF Tracient RFID Reader

RFID Tag

The RFID tag used in the experiment is a Skyetek RFID tag which supports ultra high frequency (UHF) and has user memory to store additional information in addition to RFID tag identification. It is an EPC Gen 2/ 18000-6B standard and operates in the frequency range of 860MHz to 960MHz. The size of the user memory is 1728 bit (216 bytes) according to the information sheet provided by SkyeTek (as shown in the Figure A6.3).

Figure A6.3: Skytek UHF tag

Tracient SDK Software

According to the Tracient Technologies Ltd., the software includes three main applications such as “Control Panel”, “RFID Wedge” and “RFID

Sync" allowing the users to configure RFID reader, to collect the data on tag, and downloading the data to the host PC.

Control Panel

The Control Panel applet is one of the Tracient software applications. The main functions of the Control Panel are RFID test, Datascan and Reader log. RFID test is used for testing an RFID tag to the setting applied. Datascan is used when the user memory is required to be read while reading tag ID. Reader log can only store the identifications of the tags have been read.

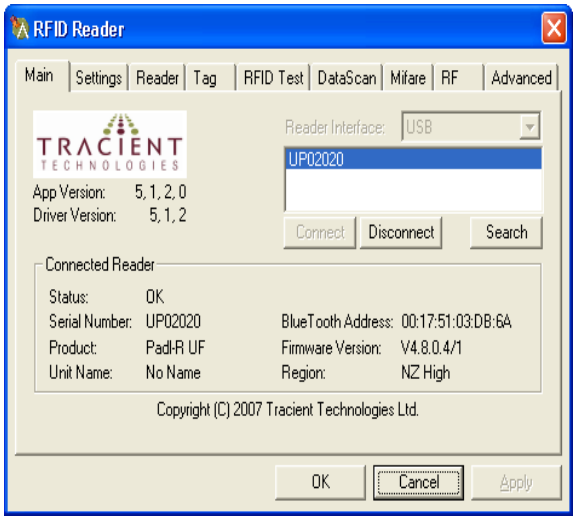


Figure A6.4: RFID Control Panel Applet

The settings used for reader and Skyetek ISO 18000-6B tag in Control Panel can be seen in the Figure A6.5 and Figure A6.6 below.

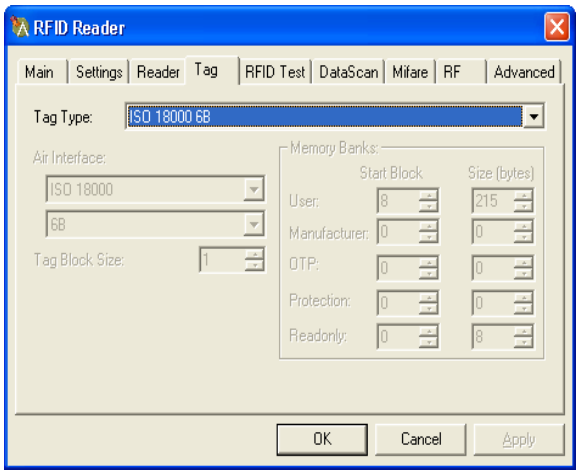


Figure A6.5: Skyetek RFID tag setting in the Control Panel

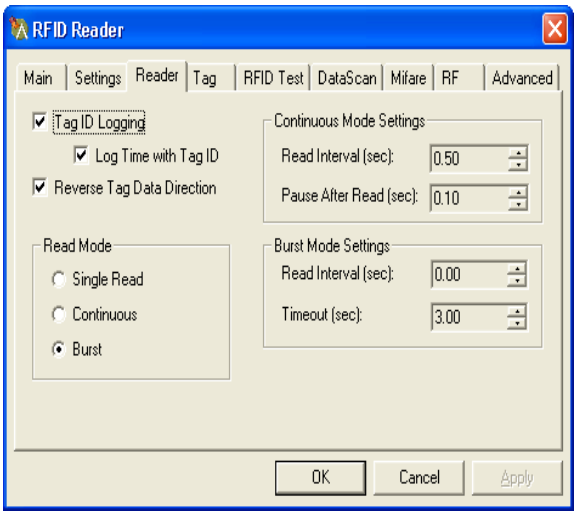


Figure A6.6: Reader setting in the Control Panel

RFID Wedge

The RFID Wedge application supports the transfer of tag data (output) to either selected applications or text file (see Figure A6.7).

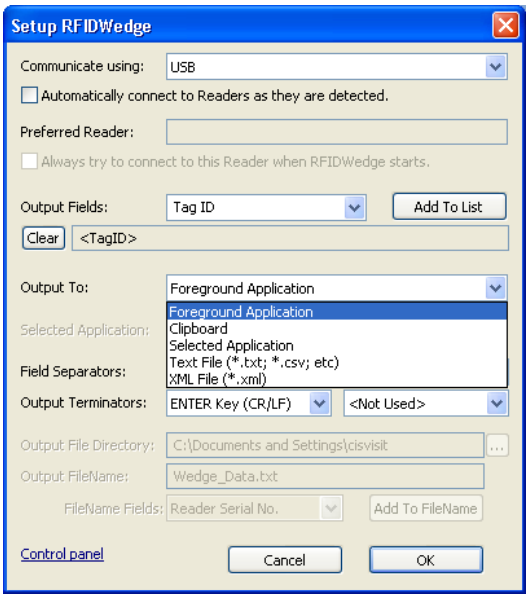


Figure A6.7: RFID Wedge Application Settings

FID Sync

The RFID Sync application allows user to download the collected data from the reader log. The data include tag ID, user memory, reader ID, reader name and timestamps.

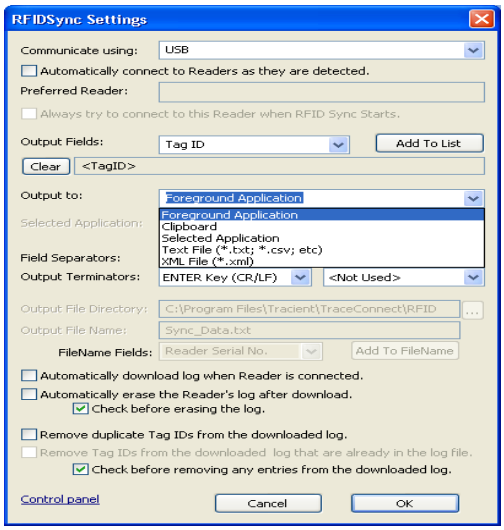


Figure A6.8: RFID Sync Application Settings

Similar to RFID Wedge application, Sync application output can be either text or XML flie.

2. RFID Middleware Creation with Software Development Kit (SDK)

According to the experiment requirement, the middleware software is created in Visual C Sharp (C#) programming language by using software development kit (SDK). The basic function of RFID middleware can be found in the Figure A6.9. The main purpose of creating customized RFID middleware is to read and write the tag data and writing the logs from the RFID reader memory to the backend database.

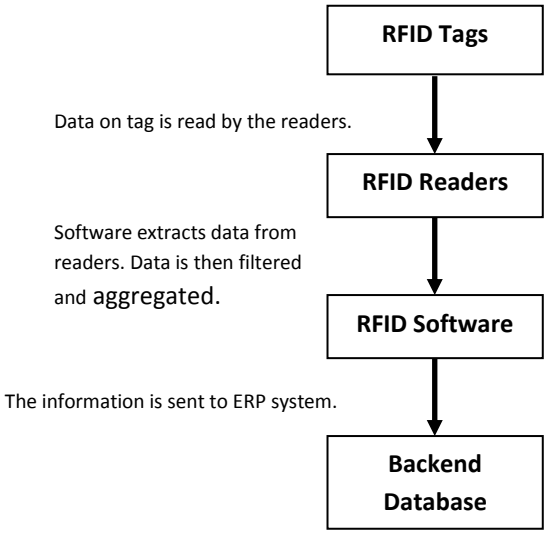


Figure A6.9: Basic Functions of RFID Middleware Software

3. Creating the Databases in the Backend Server

Before the attack simulation, the two databases were created in the backend Microsoft SQL 2005 server. The primary data file named RFID_test.mdf was used for storing product values according to the RFID tags. Likewise, the transaction log file named RFID_test_log.ldf was used for storing the reader logs to the backend server. Then, all the fictitious product values and tag IDs were pre-keyed into the database files. Examples of inserting the values (1000) into the primary data and log files are as follows:

```
insert into rfid_db (Tag, Value, Date) VALUES ('E0040000E90A4301', '1000','17:19:51 02/07/2010');

insert into rfid_log (Tag, Date) VALUES ('E0040000E90A4301', '17:19:51 02/07/2010');
```

4. Injecting the Malicious SQL Code into the Fake RFID Tag

As for the experiment, another SkyeTek RFID tag (ID: E0040000E90A4302) is used. This tag is also an EPC Gen 2/ 18000-6B standard operating in the frequency range of 860MHz to 960MHz. The malicious code is written into the user memory of the fake RFID tag (as shown in the Figure A6.10) in order to change the value from 1000 to 10. The inserted malicious code is only 55 bytes and the command is:

```
');update rfid_db set Value='10' where Tag > 'E004%' --
```

The injected command will change the values relating to the tag IDs starting with E004 in the database table.

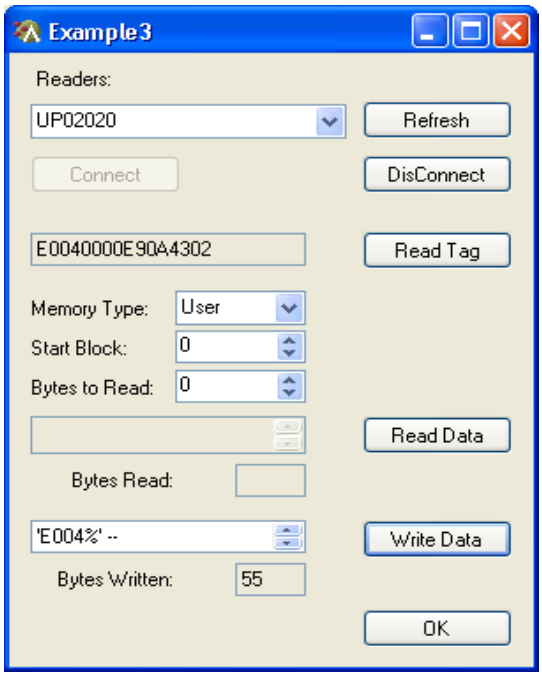


Figure A6.10: Screenshot of writing the SQL injection code to the fake tag

5. An Attack in Action

The testing BIS is a simple system as it has a reader connected to a POS terminal and the backend database is running on the host computer. When the customer is checking out the item he/she bought, the tag attachment is detached from the product and scanned at the POS terminal.

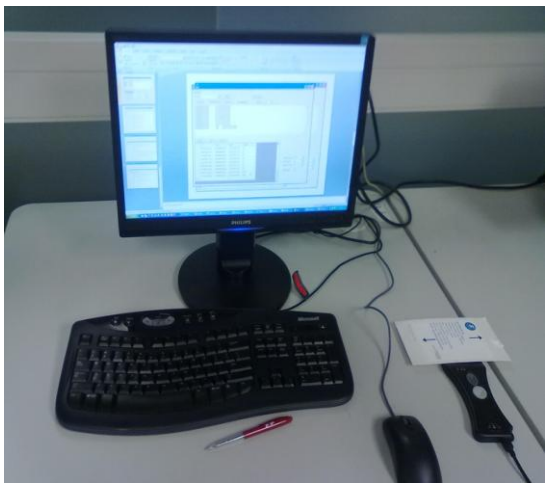


Figure A6.11: Genuine tag is read at the POS terminal

However, the fake tag is replaced by the attacker with the help of a malicious employee in store and scanned at the POS terminal (as shown in Figure A6.12); the value attached to the item will be changed in the backend database as the

malicious SQL injection code is executed at the backend database (Rieback et al., 2006).

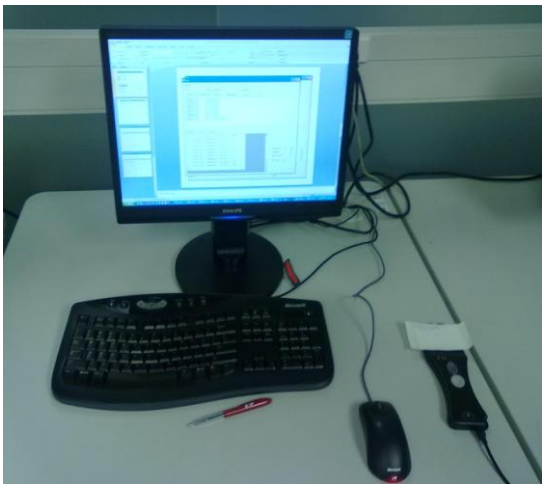
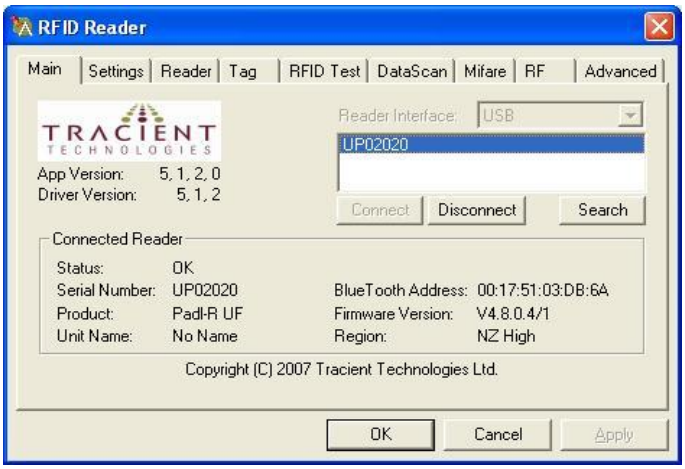


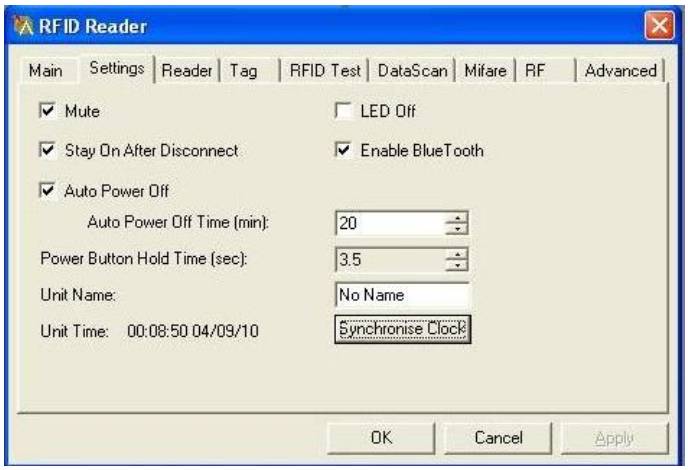
Figure A6.12: Fake tag is read by the RFID reader

6. Detail Procedures of a Replicated Attack Example and Findings after the Attack

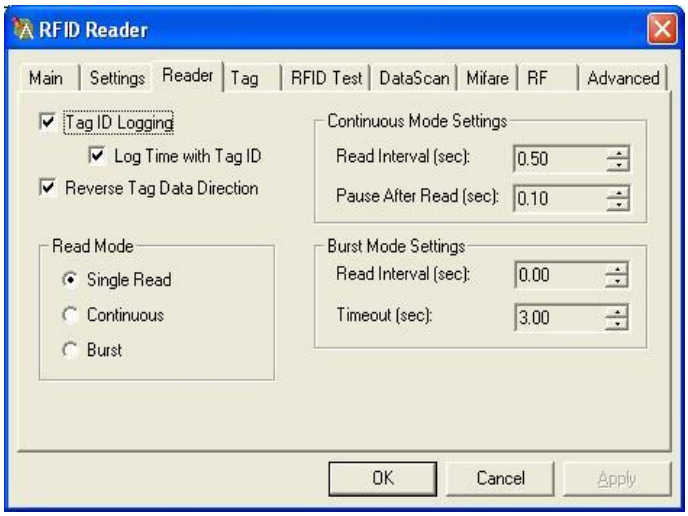
Step 1: Configure the RFID reader.



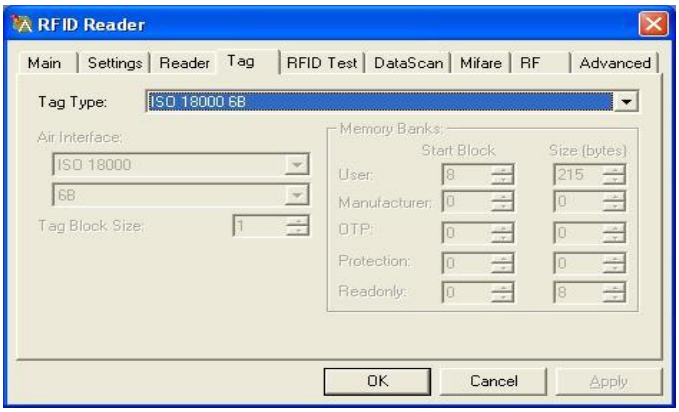
Step 2: Synchronize reader's clock with the TEST-STATION.



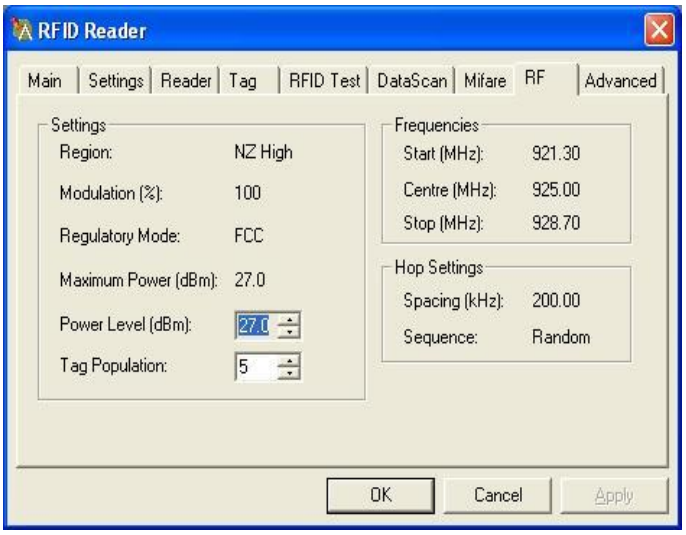
Step 3: Configure the reader's *Tag Logging.*



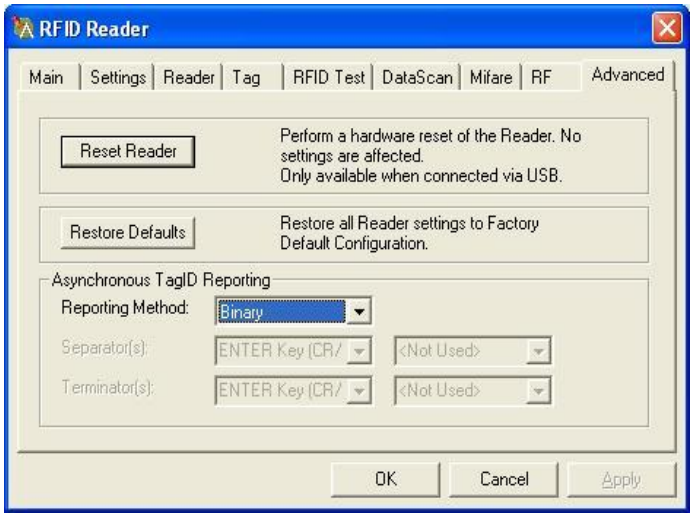
Step 4: Configure the type of RFID tag.



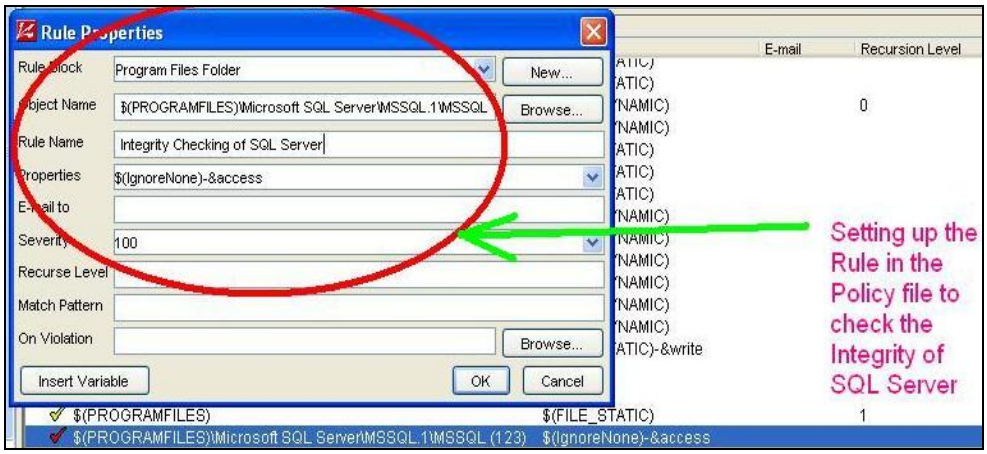
Step 5: Check the frequency ranges in operation.



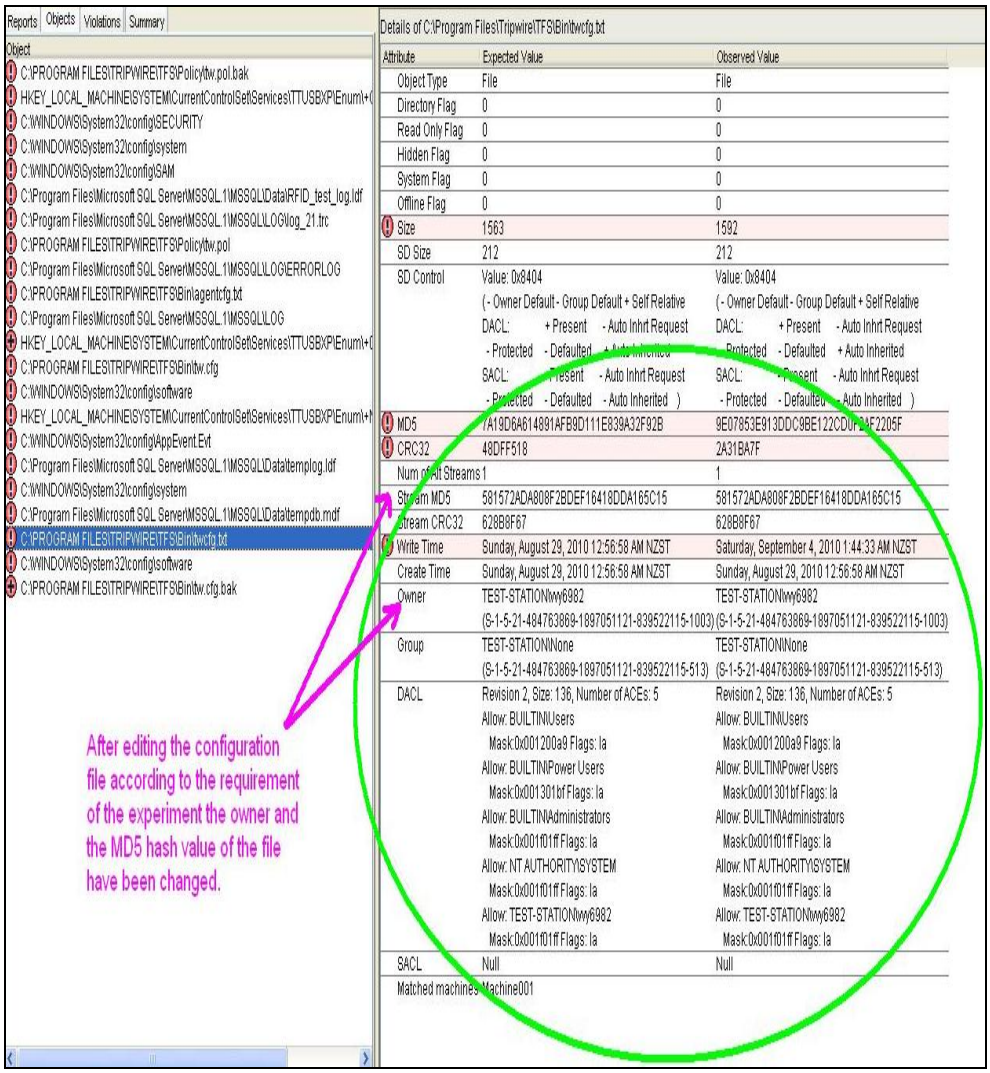
Step 6: Binary tag ID reporting method was chosen in the reader configuration.



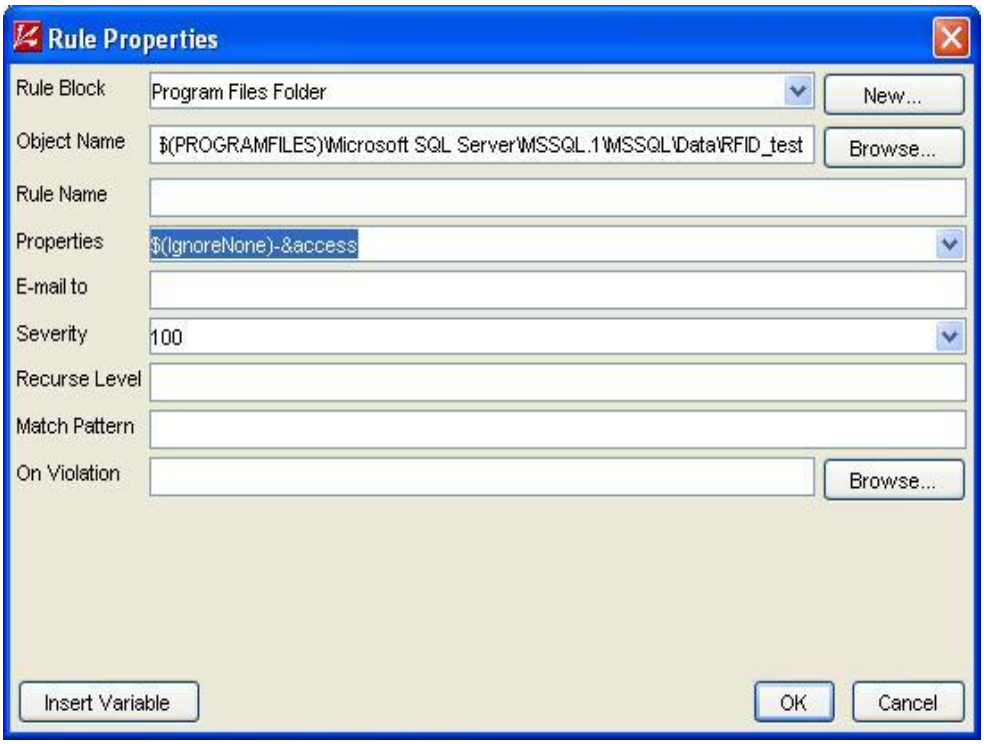
Step 7: Set up the rule in Policy File for integrity checking of the SQL Server by using Tripwire Manager (Version 4.8).



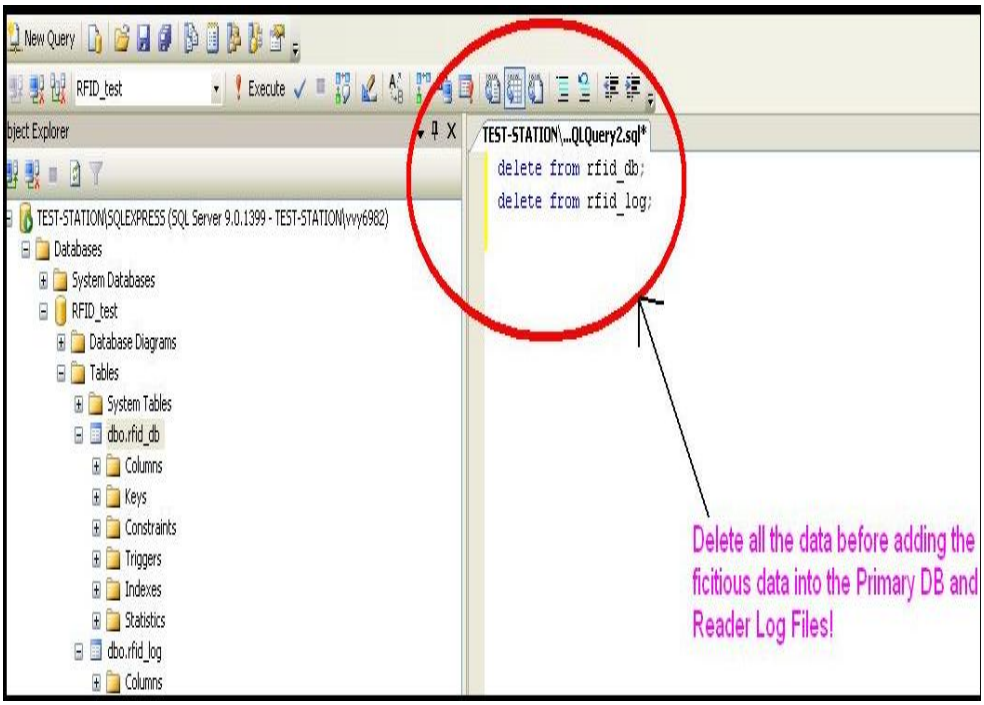
Step 8: Edit the Tripwire configuration file according to the requirement of the experiment conducted.



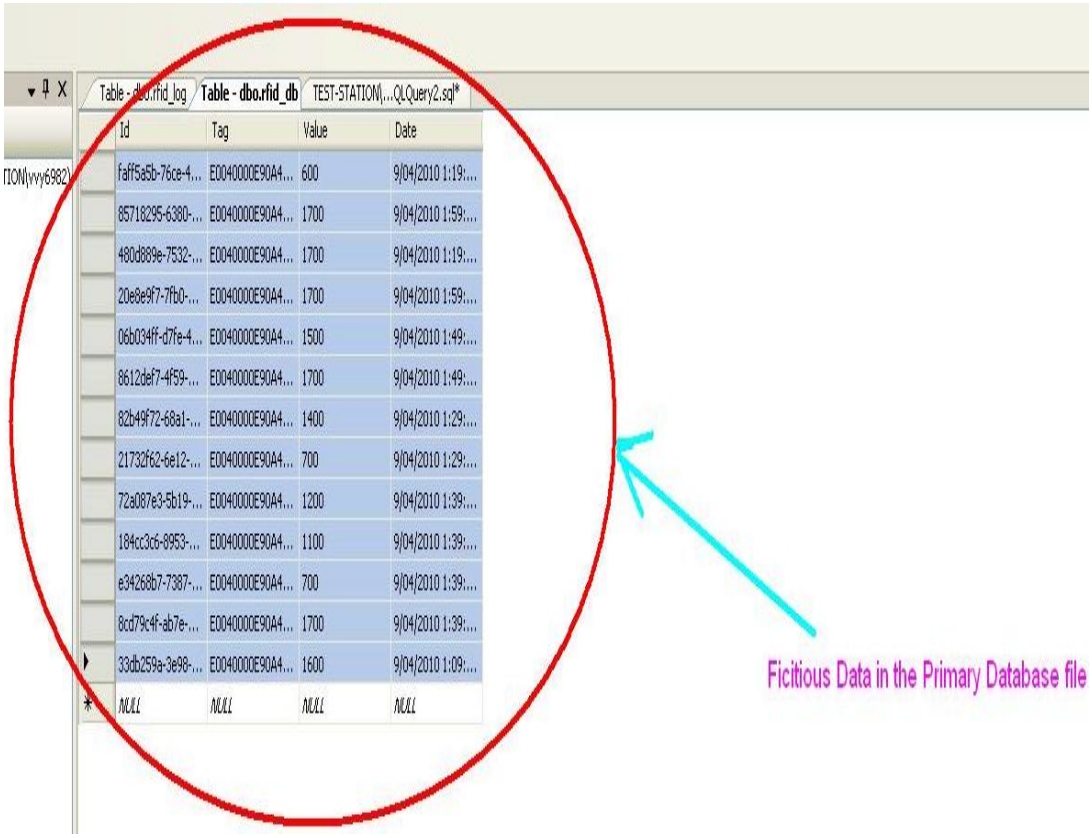
Step 9: Set up the rule in Policy File for integrity checking of the RFID_test primary database file by using Tripwire Manager.



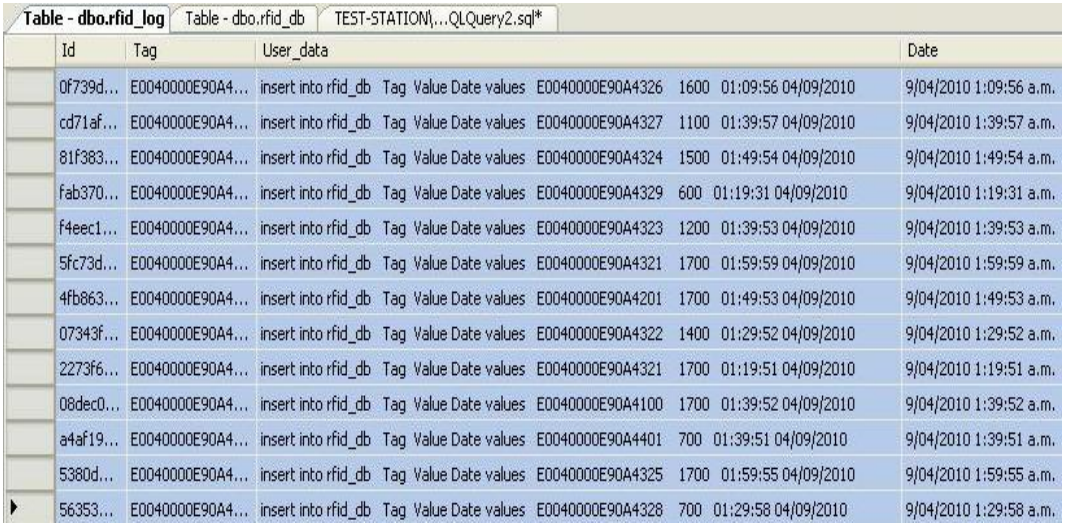
Step 10: Clear all the data from the backend databases.



Step 11: Key the fictitious data in the primary database file.

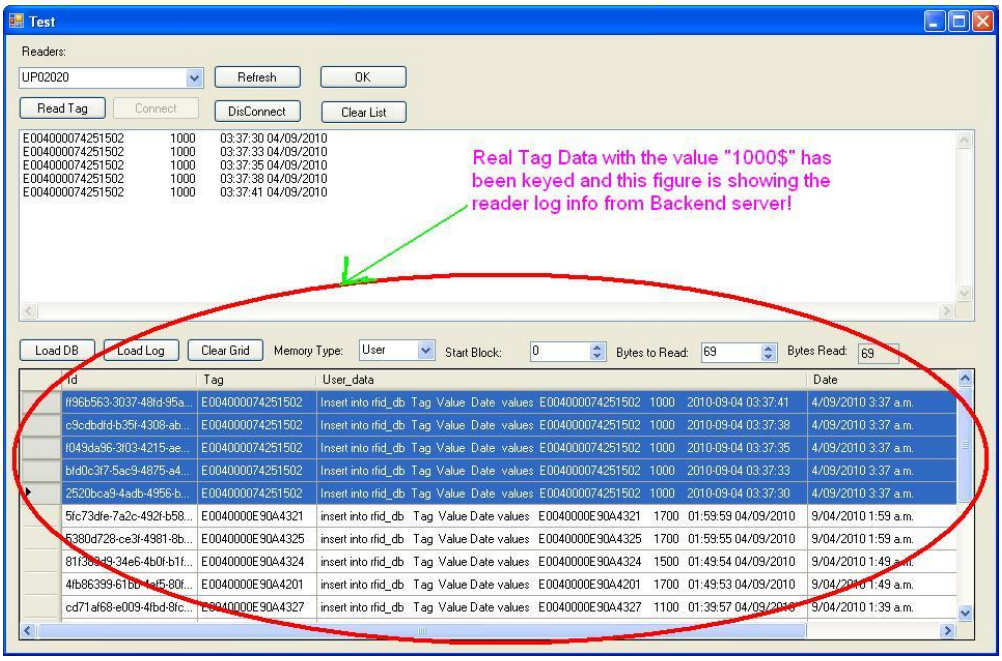


Step 12: Record of the fictitious logs before the tag data from a genuine tag has been read by the reader.

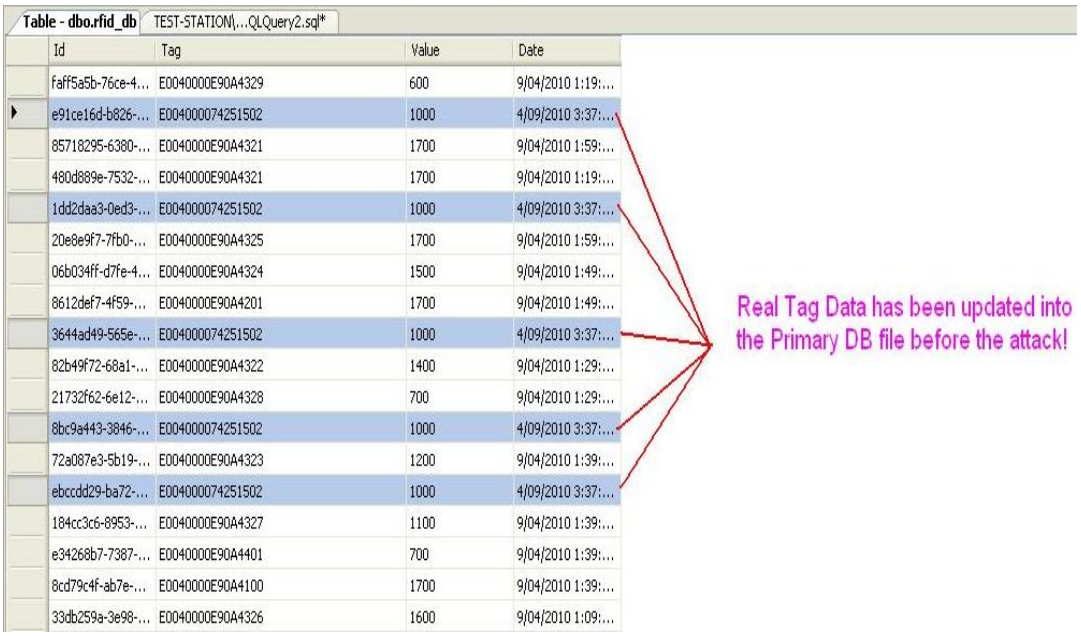


Step 13: Then, read the genuine tag for five times to have the transaction logs in the reader memory as keying the fictitious data into the backend database will not be logged into the reader's memory.

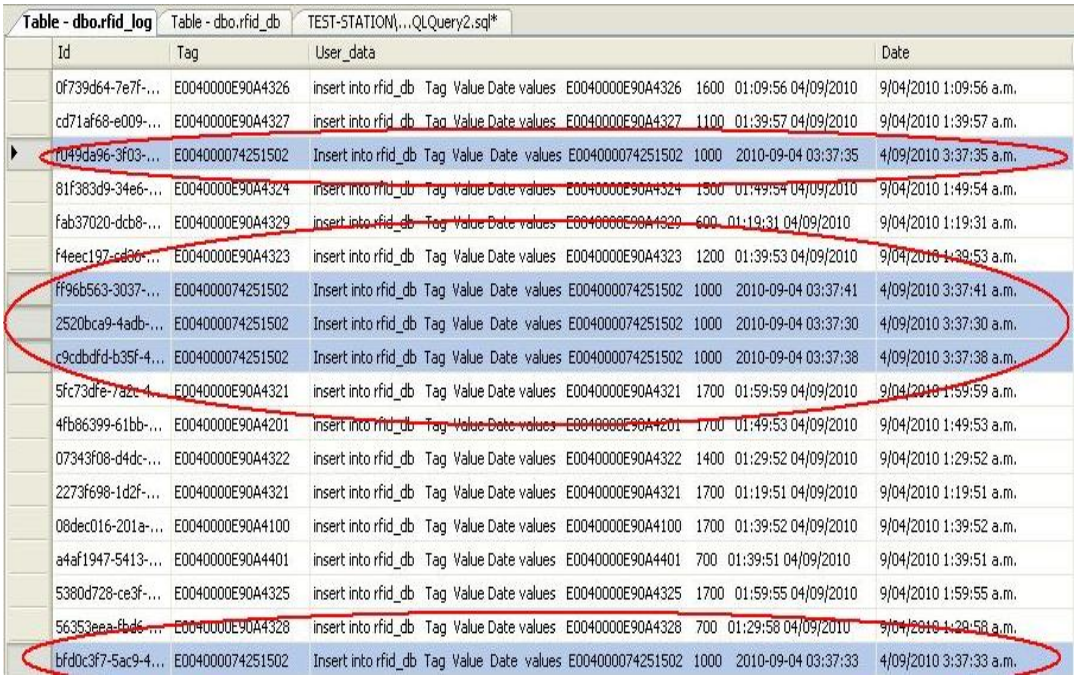
Step 14: Load the logs from the reader before the attack.



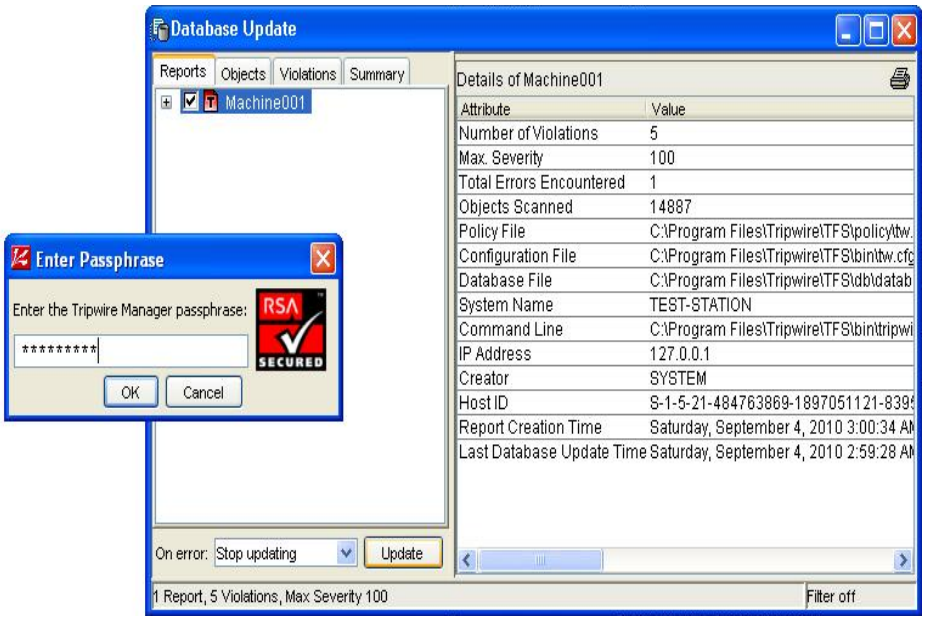
Step 15: Check whether the genuine or real tag data has been updated into the primary database file before the attack occurs. Hence, the real tag data has been updated already in the backend database.



Step 16: Similarly, check whether the real tag data has also been logged into the transaction logs from reader's memory in the log of the backend SQL Server.

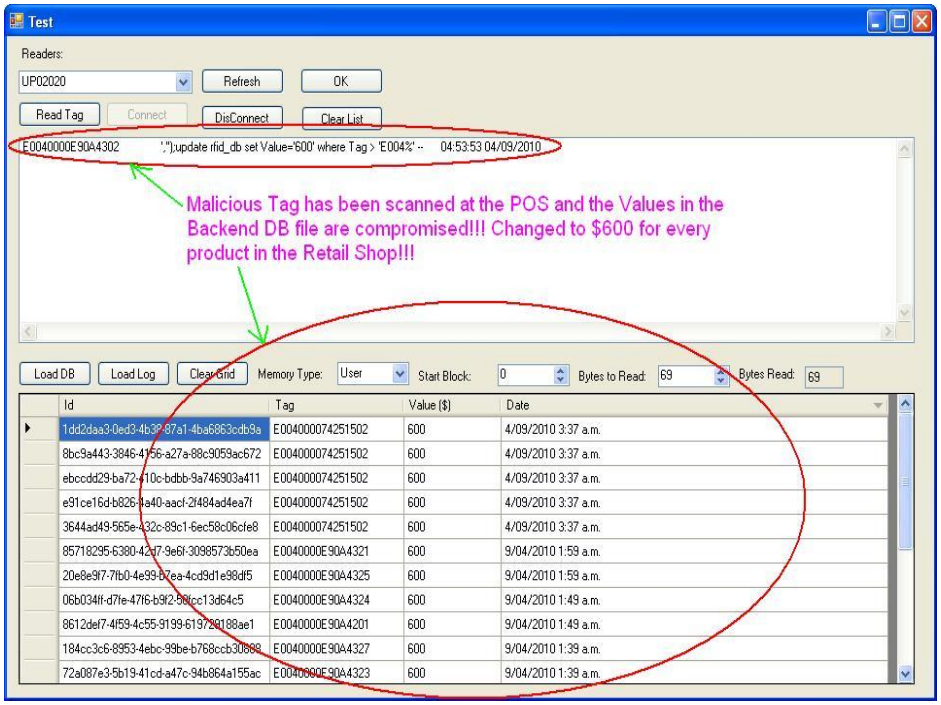


Step 17: Then, scan the fake tag injected with a malicious SQL poisoning code and update the database file of the Tripwire.

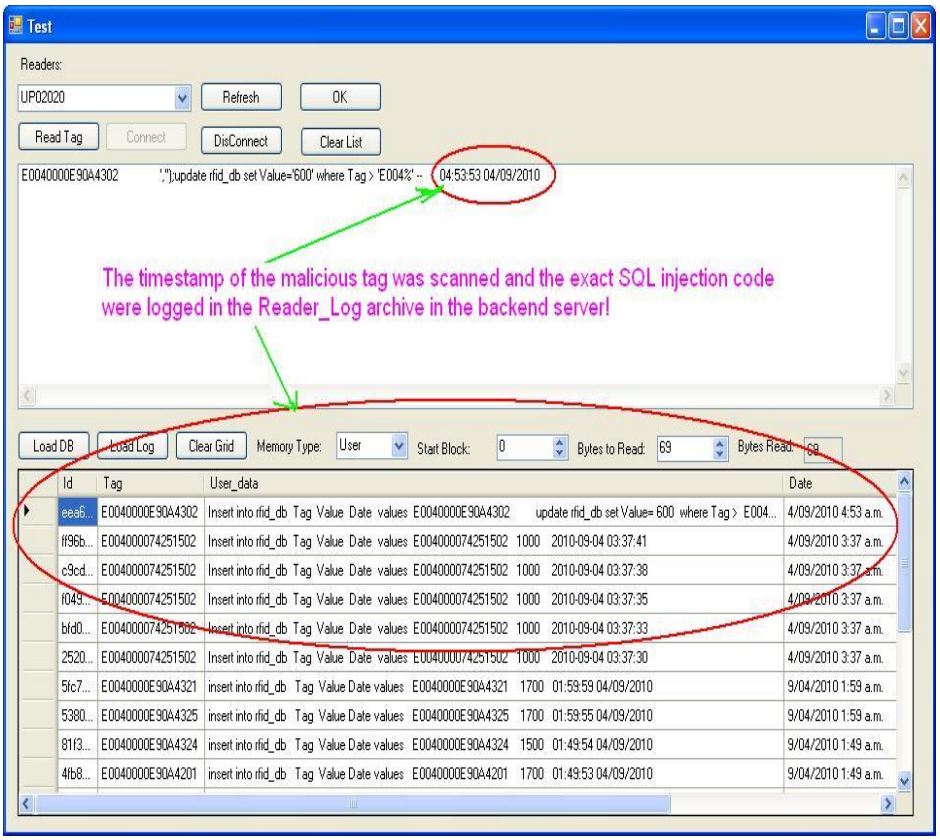


Hence, after performing the SQL Server's integrity check with Tripwire, the MD5 hashes (before and after the malicious attack) of the database under Tripwire's supervision were not the same according to the report.

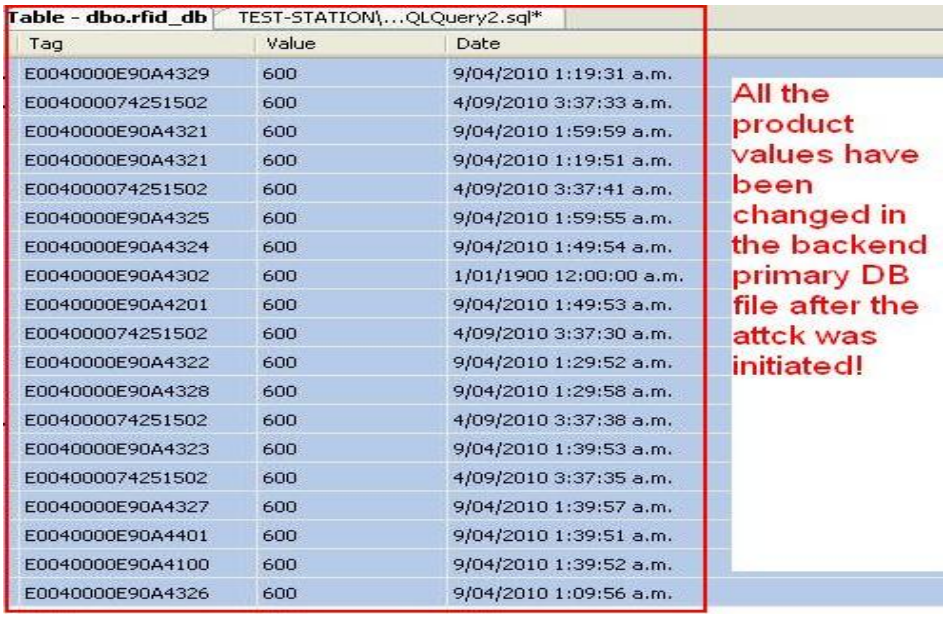
Step 18: Load the database after the malicious attack. The finding shows that all the stock item values have been changed to $600.



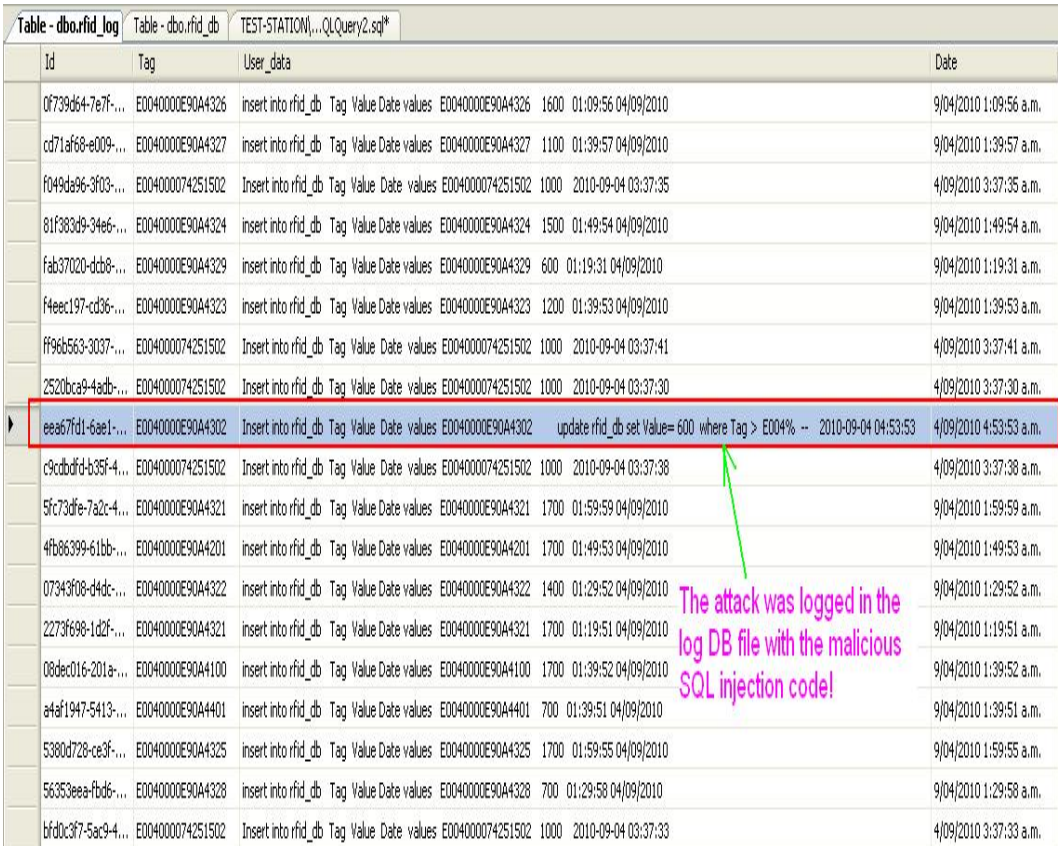
Step 19: Load the reader logs from the backend server after the malicious attack.



Step 20: Check the primary database file on the backend SQL Server. Hence, all the product values have been changed to $600 after the attack



Step 21: Then, check the transaction log file of the backend server. It is found that the attack has been recorded in the transaction logs of the backend server.



Appendix 7: Steps taken before the attack was initiated

Step 1: Fake data was keyed into the rfid_db and rfid_log databases.

```
insert into rfid_db ( Tag, Value,Date) values ('E0040000E90B4321',
'1700','01:19:51 12/10/2010');

insert into rfid_db ( Tag, Value,Date) values ('E0040000E90B4322',
'1400','01:29:52 12/10/2010');

insert into rfid_db ( Tag, Value,Date) values ('E0040000E90B4323',
'1200','01:39:53 12/10/2010');

insert into rfid_db ( Tag, Value,Date) values ('E0040000E90B4324',
'1500','01:49:54 12/10/2010');

insert into rfid_db ( Tag, Value,Date) values ('E0040000E90B4325',
'1700','01:59:55 12/10/2010');

insert into rfid_db ( Tag, Value,Date) values ('E0040000E90B4326',
'1600','01:09:56 12/10/2010');

insert into rfid_db ( Tag, Value,Date) values ('E0040000E90B4327',
'1100','01:39:57 12/10/2010');

insert into rfid_db ( Tag, Value,Date) values ('E0040000E90B4328',
'700','01:29:58 12/10/2010');

insert into rfid_db ( Tag, Value,Date) values ('E0040000E90B4329',
'600','01:19:31 12/10/2010');

insert into rfid_db ( Tag, Value,Date) values ('E0040000E90B4100',
'1700','01:39:52 12/10/2010');

insert into rfid_db ( Tag, Value,Date) values ('E0040000E90B4201',
'1700','01:49:53 12/10/2010');

insert into rfid_db ( Tag, Value,Date) values ('E0040000E90B4321',
'1700','01:59:59 12/10/2010');
```

```
insert into rfid_db ( Tag, Value,Date) values ('E0040000E90B4401',
'700','01:39:51 12/10/2010');

insert into rfid_log ( Tag, User_data, Date) values
('E0040000E90B4321', 'insert into rfid_db Tag Value Date values
E0040000E90B4321 1700 01:19:51 12/10/2010 ', '01:19:51
12/10/2010');

insert into rfid_log ( Tag, User_data, Date) values
('E0040000E90B4322', 'insert into rfid_db Tag Value Date values
E0040000E90B4322 1400 01:29:52 12/10/2010 ', '01:29:52
12/10/2010');

insert into rfid_log ( Tag, User_data, Date) values
('E0040000E90B4323', 'insert into rfid_db Tag Value Date values
E0040000E90B4323 1200 01:39:53 12/10/2010 ', '01:39:53
12/10/2010');

insert into rfid_log ( Tag, User_data, Date) values
('E0040000E90B4324', 'insert into rfid_db Tag Value Date values
E0040000E90B4324 1500 01:49:54 12/10/2010 ', '01:49:54
12/10/2010');

insert into rfid_log ( Tag, User_data, Date) values
('E0040000E90B4325', 'insert into rfid_db Tag Value Date values
E0040000E90B4325 1700 01:59:55 12/10/2010 ', '01:59:55
12/10/2010');

insert into rfid_log ( Tag, User_data, Date) values
('E0040000E90B4326', 'insert into rfid_db Tag Value Date values
E0040000E90B4326 1600 01:09:56 12/10/2010 ', '01:09:56
12/10/2010');

insert into rfid_log ( Tag, User_data, Date) values
('E0040000E90B4327', 'insert into rfid_db Tag Value Date values
E0040000E90B4327 1100 01:39:57 12/10/2010 ', '01:39:57
12/10/2010');
```

```
insert into rfid_log ( Tag, User_data, Date) values
('E0040000E90B4328', 'insert into rfid_db Tag Value Date values
E0040000E90B4328 700 01:29:58 12/10/2010 ', '01:29:58
12/10/2010');

insert into rfid_log ( Tag, User_data, Date) values
('E0040000E90B4329', 'insert into rfid_db Tag Value Date values
E0040000E90B4329 600 01:19:31 12/10/2010 ', '01:19:31
12/10/2010');

insert into rfid_log ( Tag, User_data, Date) values
('E0040000E90B4100', 'insert into rfid_db Tag Value Date values
E0040000E90B4100 1700 01:39:52 12/10/2010 ', '01:39:52
12/10/2010');

insert into rfid_log ( Tag, User_data, Date) values
('E0040000E90B4201', 'insert into rfid_db Tag Value Date values
E0040000E90B4201 1700 01:49:53 12/10/2010 ', '01:49:53
12/10/2010');

insert into rfid_log ( Tag, User_data, Date) values
('E0040000E90B4321', 'insert into rfid_db Tag Value Date values
E0040000E90B4321 1700 01:59:59 12/10/2010 ', '01:59:59
12/10/2010');

insert into rfid_log ( Tag, User_data, Date) values
('E0040000E90B4401', 'insert into rfid_db Tag Value Date values
E0040000E90B4401 700 01:39:51 12/10/2010 ', '01:39:51
12/10/2010');
```

Step 2: Checking the fictitious data keyed into the rfid_db and rfid_log databases on the backend server.

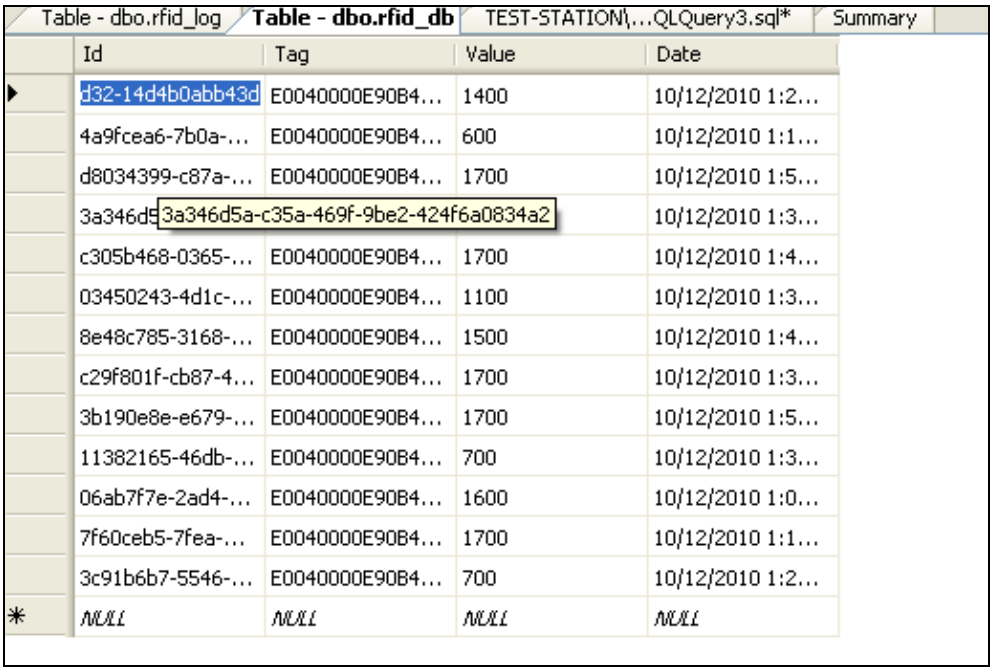


Figure A7.1: Pre-keyed data in the rfid_db database

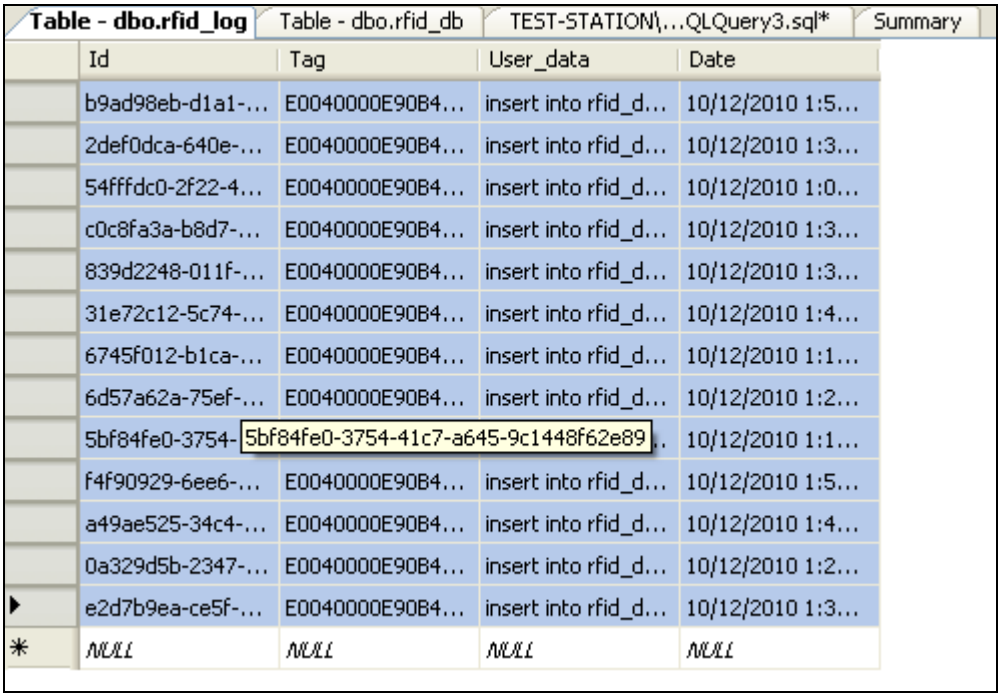


Figure A7.2: Pre-keyed data in the rfid_log database

Step 3: Checking the fictitious data keyed into the rfid_db by using the R/W RFID middleware API.

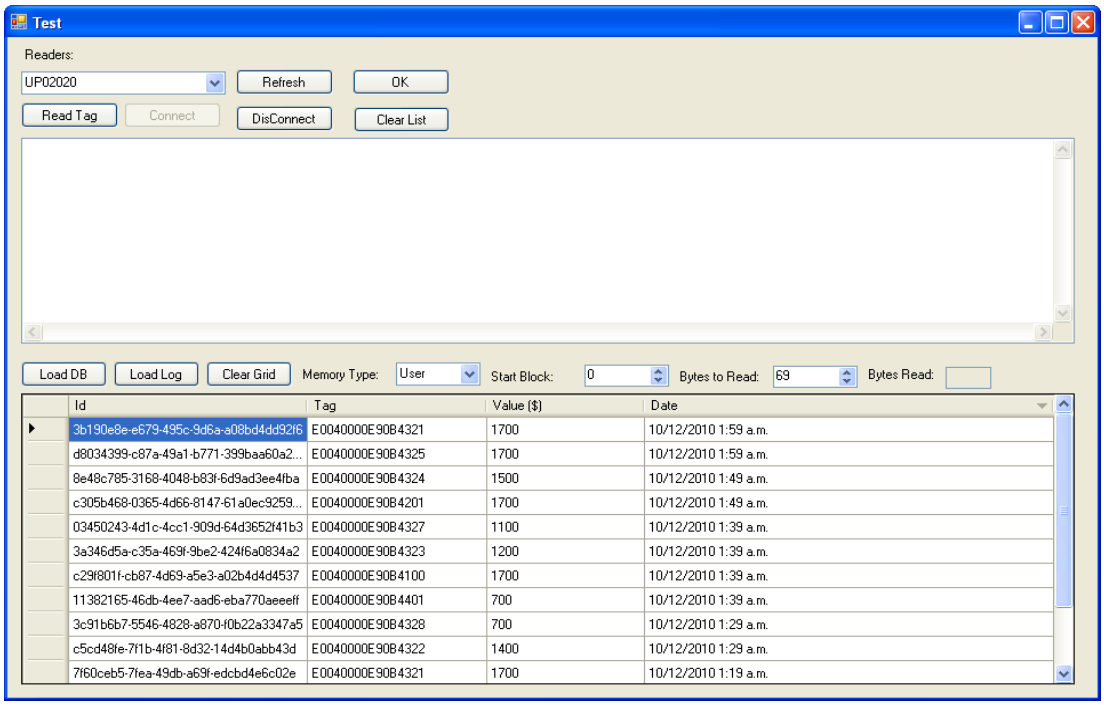


Figure A7.3: Pre-keyed data in the rfid_db database is loaded by the R/W RFID middleware API

Step 4: Inserted the value of $1500, which was written on the genuine SI, to the backend databases for 5 times (the data was read by RFID reader and sent to the backend server by the customized middleware software). Then the databases were loaded to verify (see the figures A7.4 and A7.5).

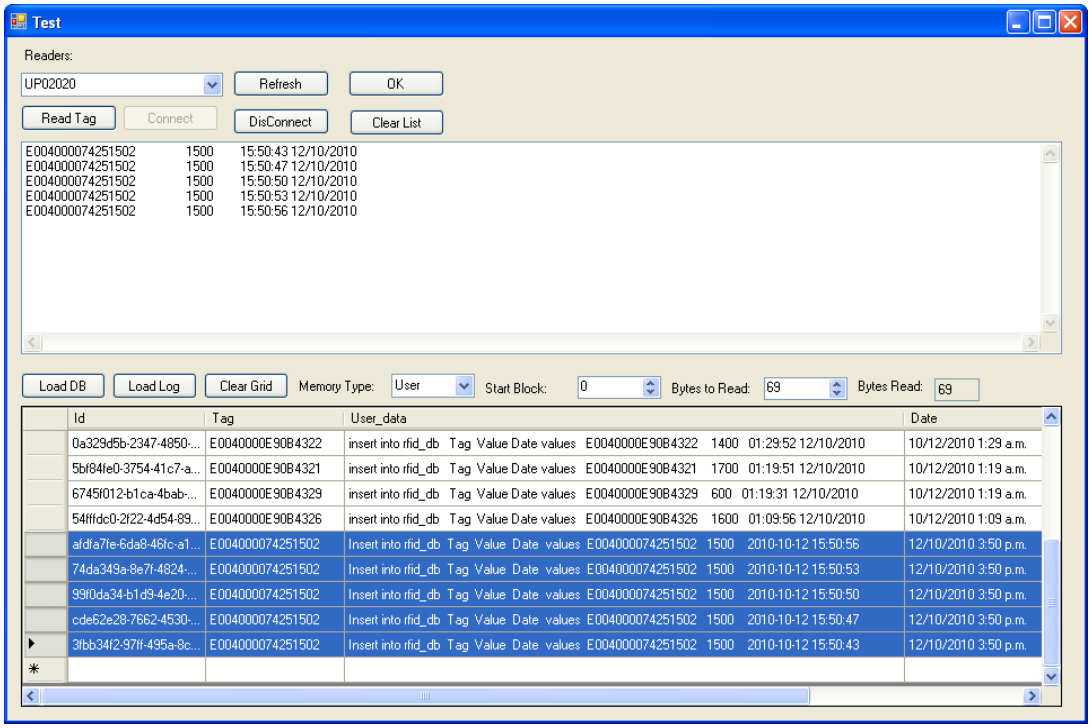


Figure A7.4: RFID log database from the backend server is loaded by the R/W RFID middleware API

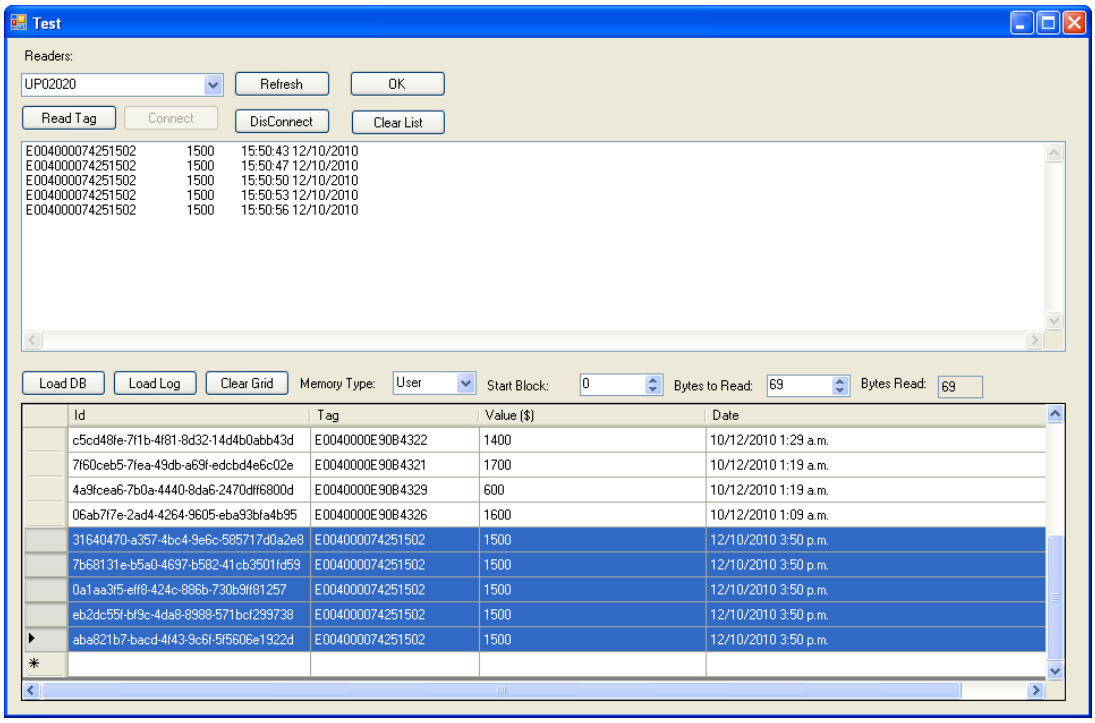


Figure A7.5: RFID database from the backend server is loaded by the R/W RFID middleware API

Step 5: Checking the pre-keyed data from the databases on the backend server.

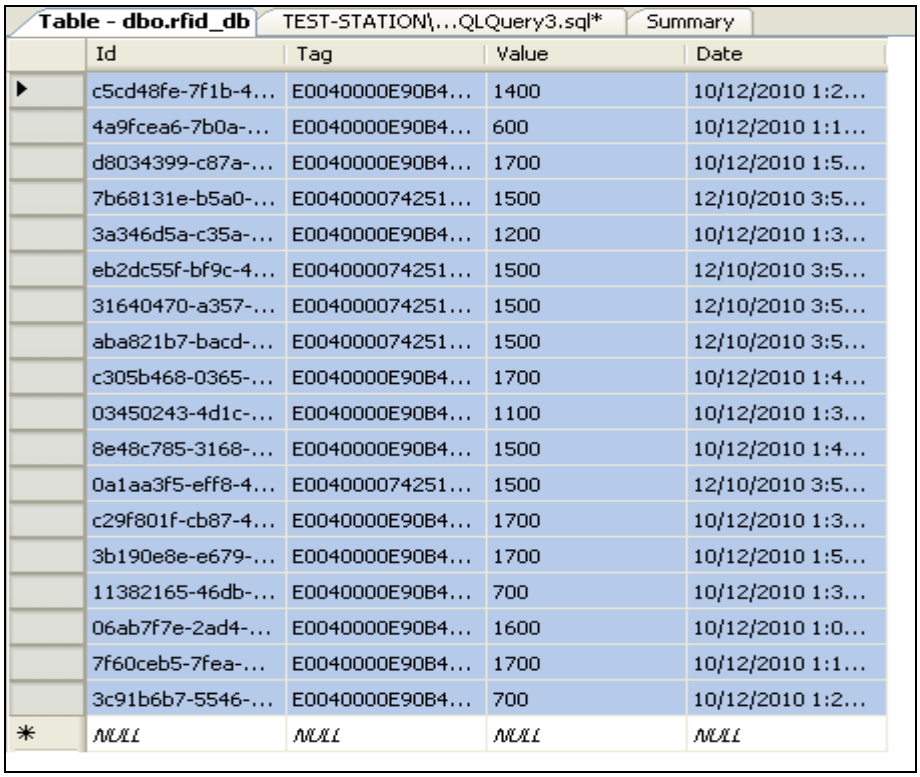


Figure A7.6: Pre-keyed data in the rfid_db database after keying five times of the genuine SI (with tag ID: E004000074251502) with a value of $1500

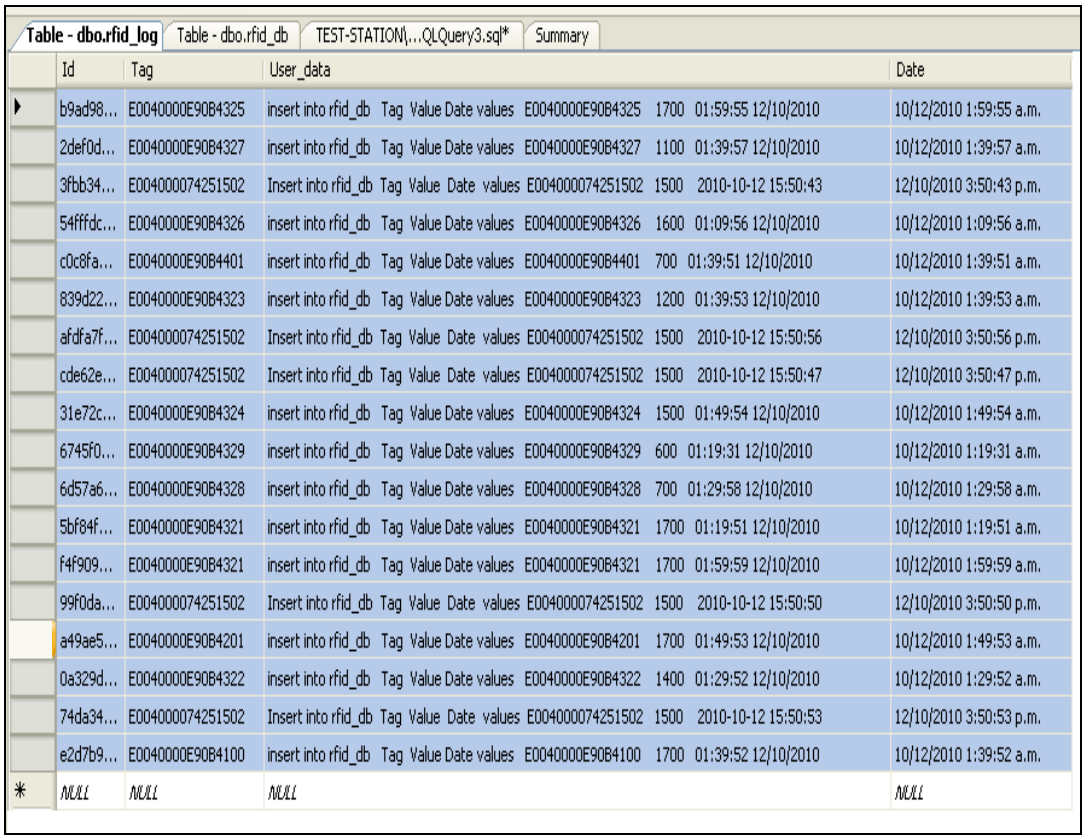


Table - dbo.rfid_log | Table - dbo.rfid_db | TEST-STATION\...QLQuery3.sql* | Summary

Id	Tag	User_data	Date
b9ad98...	E0040000E90B4325	insert into rfid_db Tag Value Date values E0040000E90B4325 1700 01:59:55 12/10/2010	10/12/2010 1:59:55 a.m.
2def0d...	E0040000E90B4327	insert into rfid_db Tag Value Date values E0040000E90B4327 1100 01:39:57 12/10/2010	10/12/2010 1:39:57 a.m.
3fbb34...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:43	12/10/2010 3:50:43 p.m.
54fffdc...	E0040000E90B4326	insert into rfid_db Tag Value Date values E0040000E90B4326 1600 01:09:56 12/10/2010	10/12/2010 1:09:56 a.m.
c0c8fa...	E0040000E90B4401	insert into rfid_db Tag Value Date values E0040000E90B4401 700 01:39:51 12/10/2010	10/12/2010 1:39:51 a.m.
839d22...	E0040000E90B4323	insert into rfid_db Tag Value Date values E0040000E90B4323 1200 01:39:53 12/10/2010	10/12/2010 1:39:53 a.m.
afdfa7f...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:56	12/10/2010 3:50:56 p.m.
cde62e...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:47	12/10/2010 3:50:47 p.m.
31e72c...	E0040000E90B4324	insert into rfid_db Tag Value Date values E0040000E90B4324 1500 01:49:54 12/10/2010	10/12/2010 1:49:54 a.m.
6745f0...	E0040000E90B4329	insert into rfid_db Tag Value Date values E0040000E90B4329 600 01:19:31 12/10/2010	10/12/2010 1:19:31 a.m.
6d57a6...	E0040000E90B4328	insert into rfid_db Tag Value Date values E0040000E90B4328 700 01:29:58 12/10/2010	10/12/2010 1:29:58 a.m.
5bf84f...	E0040000E90B4321	insert into rfid_db Tag Value Date values E0040000E90B4321 1700 01:19:51 12/10/2010	10/12/2010 1:19:51 a.m.
f4f909...	E0040000E90B4321	insert into rfid_db Tag Value Date values E0040000E90B4321 1700 01:59:59 12/10/2010	10/12/2010 1:59:59 a.m.
99f0da...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:50	12/10/2010 3:50:50 p.m.
a49ae5...	E0040000E90B4201	insert into rfid_db Tag Value Date values E0040000E90B4201 1700 01:49:53 12/10/2010	10/12/2010 1:49:53 a.m.
0a329d...	E0040000E90B4322	insert into rfid_db Tag Value Date values E0040000E90B4322 1400 01:29:52 12/10/2010	10/12/2010 1:29:52 a.m.
74da34...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:53	12/10/2010 3:50:53 p.m.
e2d7b9...	E0040000E90B4100	insert into rfid_db Tag Value Date values E0040000E90B4100 1700 01:39:52 12/10/2010	10/12/2010 1:39:52 a.m.
NULL	*NULL*	*NULL*	*NULL*

Figure A7.7: Pre-keyed data in the rfid_log database after keying five times of the genuine SI (with tag ID: E004000074251502) with a value of $1500

Step 6: Checking the integrity of the databases by using Tripwire Manager (TM Version 4.8.0.246).

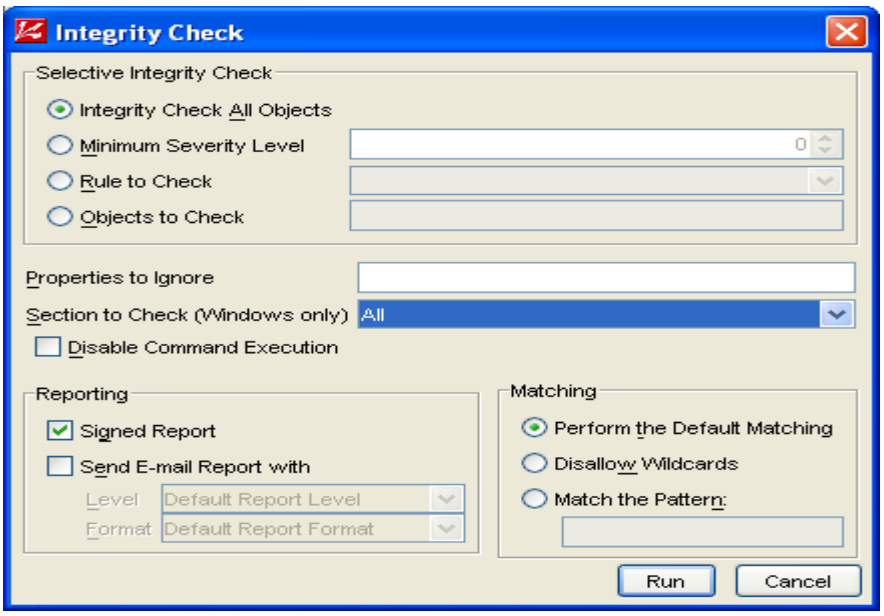


Figure A7.8: Integrity Check on SQL Server is initiated with TM

Step 7: However, TM passphrase had to be entered in order to perform integrity

checking by using, as for security reason.

Figure A7.9: Entering the passphrase to perform Integrity Check on SQL Server is initiated with TM

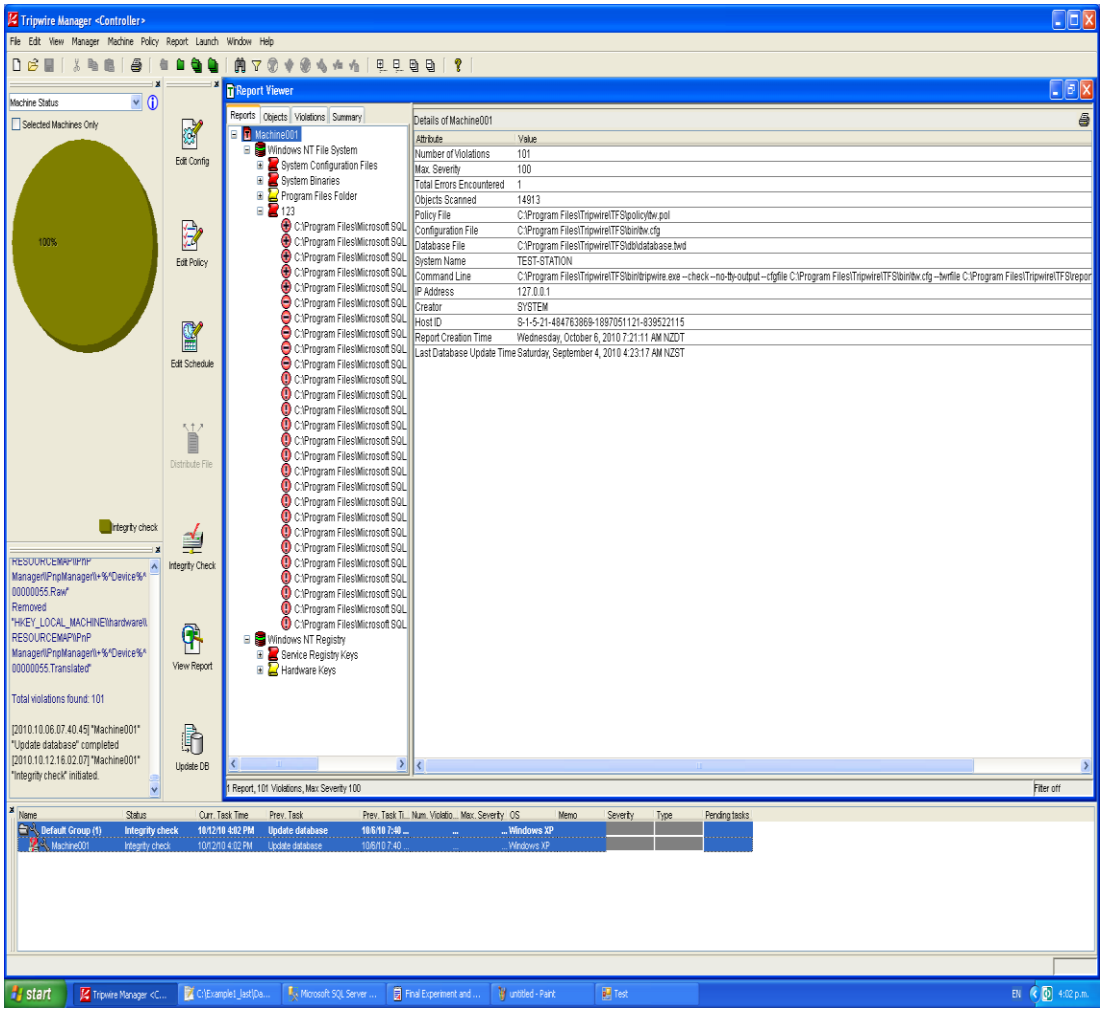


Figure A7.10: Screenshot during Integrity Check on SQL Server

Step 8: Then, the Tripwire report was viewed to check the hash value before the attack occurred (see Figure A7.12).

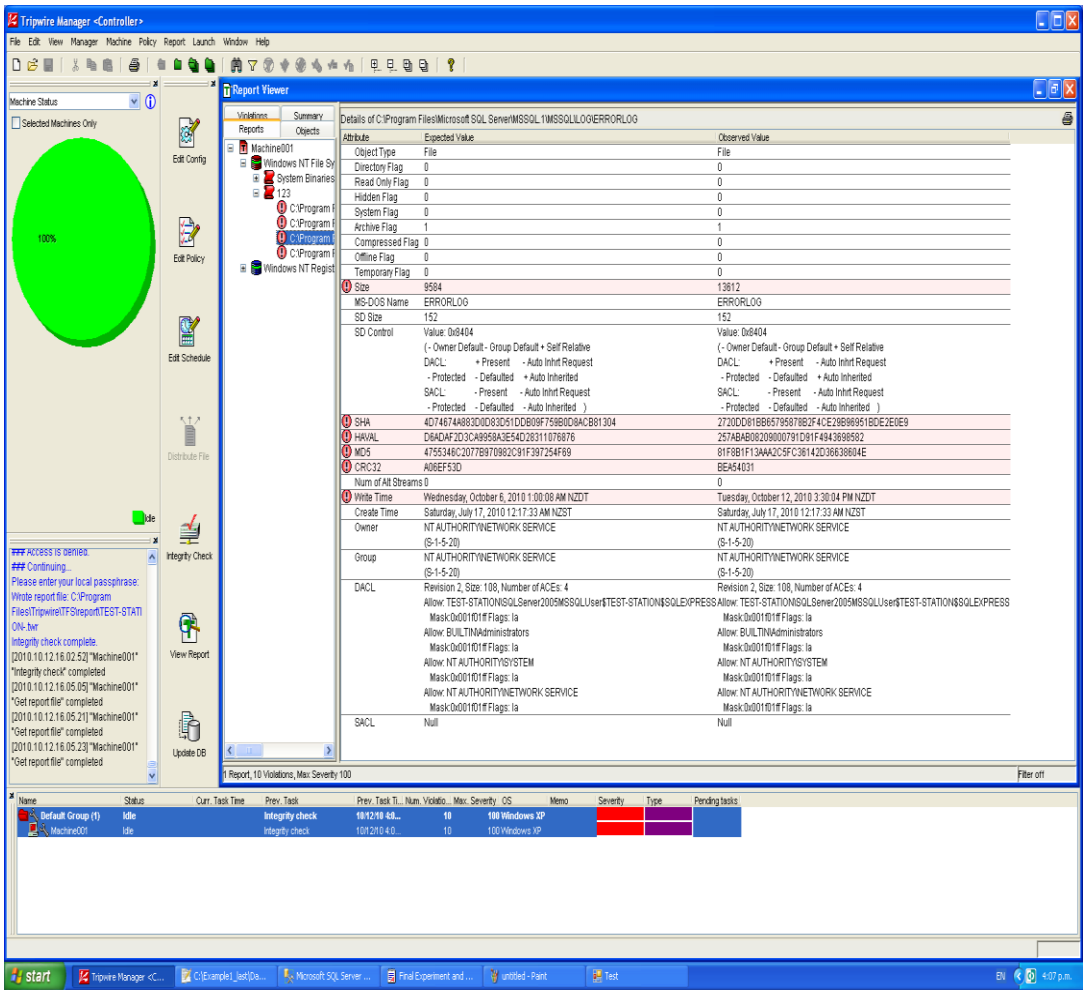


Figure A7.11: Screenshot of Tripwire report before the attack

Details of C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG

Attribute	Expected Value	Observed Value
Object Type	File	File
Directory Flag	0	0
Read Only Flag	0	0
Hidden Flag	0	0
System Flag	0	0
Archive Flag	1	1
Compressed Flag	0	0
Offline Flag	0	0
Temporary Flag	0	0
Size	9584	13612
MS-DOS Name	ERRORLOG	ERRORLOG
SD Size	152	152
SHA	4D74674A883D0D83D51DDB09F759B0D8ACB81304	2720DD81BB65795878B2F4CE29B96951BDE2E0E9
HAVAL	D6ADAF2D3CA9958A3E54D28311076876	257ABAB08209000791D91F4943698582
MD5	4755346C2077B970982C91F397254F69	81F8B1F13AAA2C5FC36142D36638604E
CRC32	A06EF53D	BEA54031
Num of Alt Streams	0	0
Write Time	Wednesday, October 6, 2010 1:00:08 AM NZDT	Tuesday, October 12, 2010 3:30:04 PM NZDT
Create Time	Saturday, July 17, 2010 12:17:33 AM NZST	Saturday, July 17, 2010 12:17:33 AM NZST

Figure A7.12: The extracted information of server error log from Tripwire report before the attack

Step 9: Tripwire database was updated, but the passphrase (for security purpose) was needed to perform the update on Tripwire database. Thus, only the authorized person like system administrator of the RFID based retail system could perform the database update.

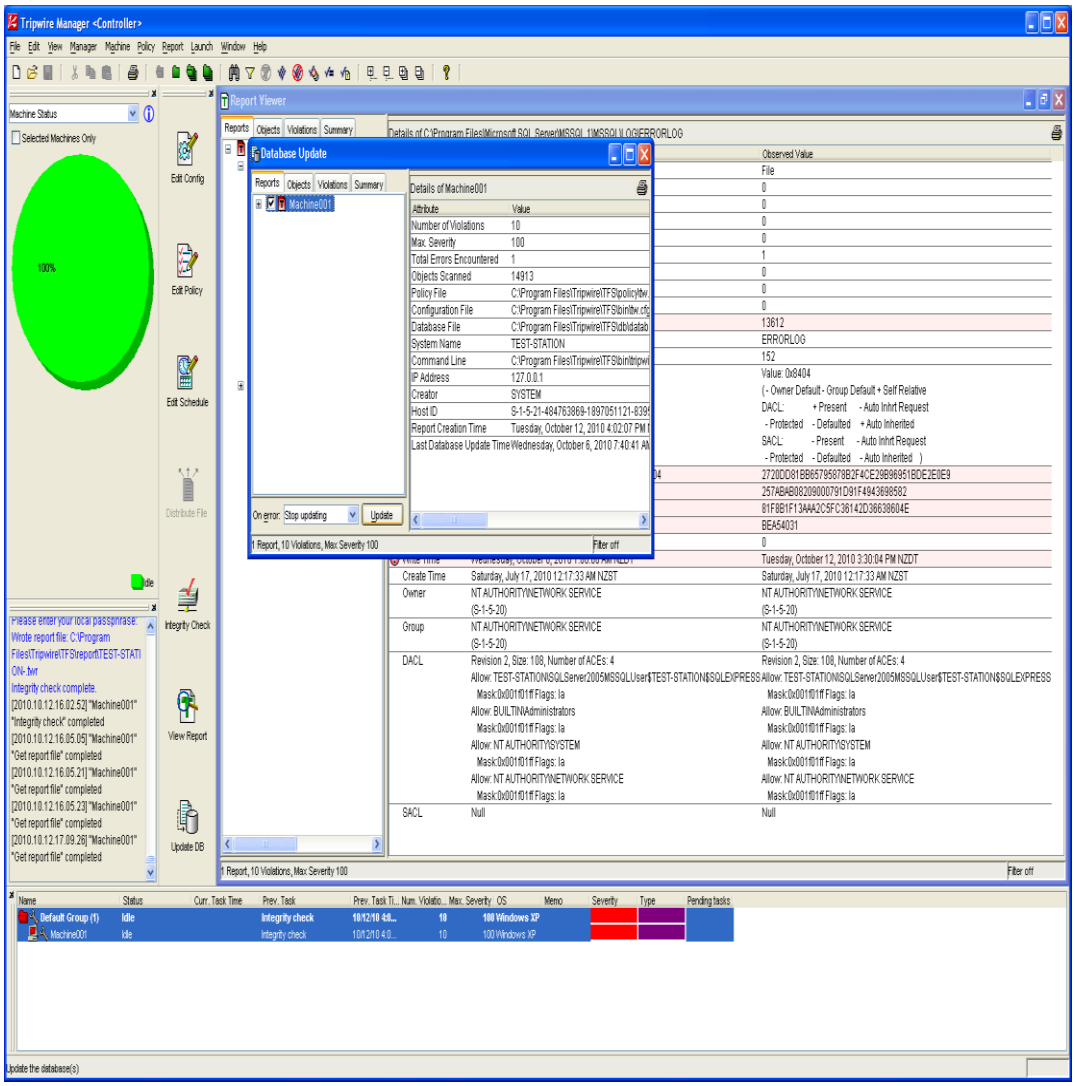


Figure A7.13: Screenshot of the Tripwire database update

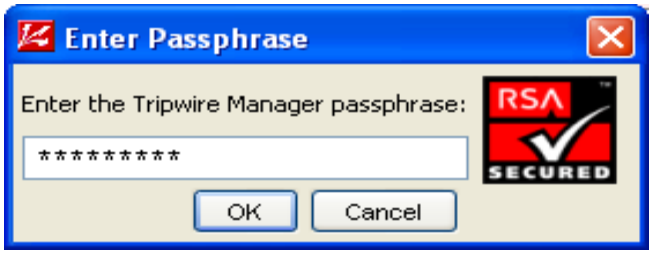


Figure A7.14: Entering the passphrase to perform Tripwire database update

Step 10: Once Tripwire database was updated, the report was reviewed again.

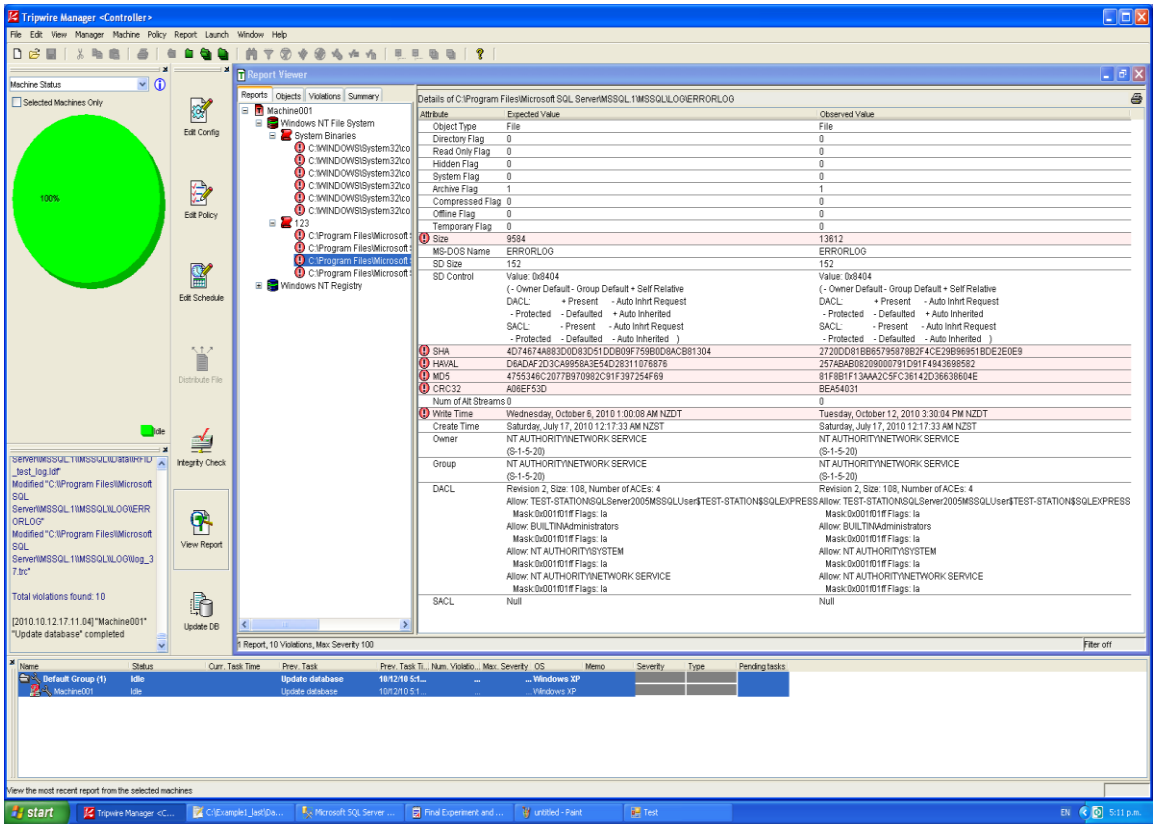


Figure A7.15: Screenshot of Tripwire report before the attack

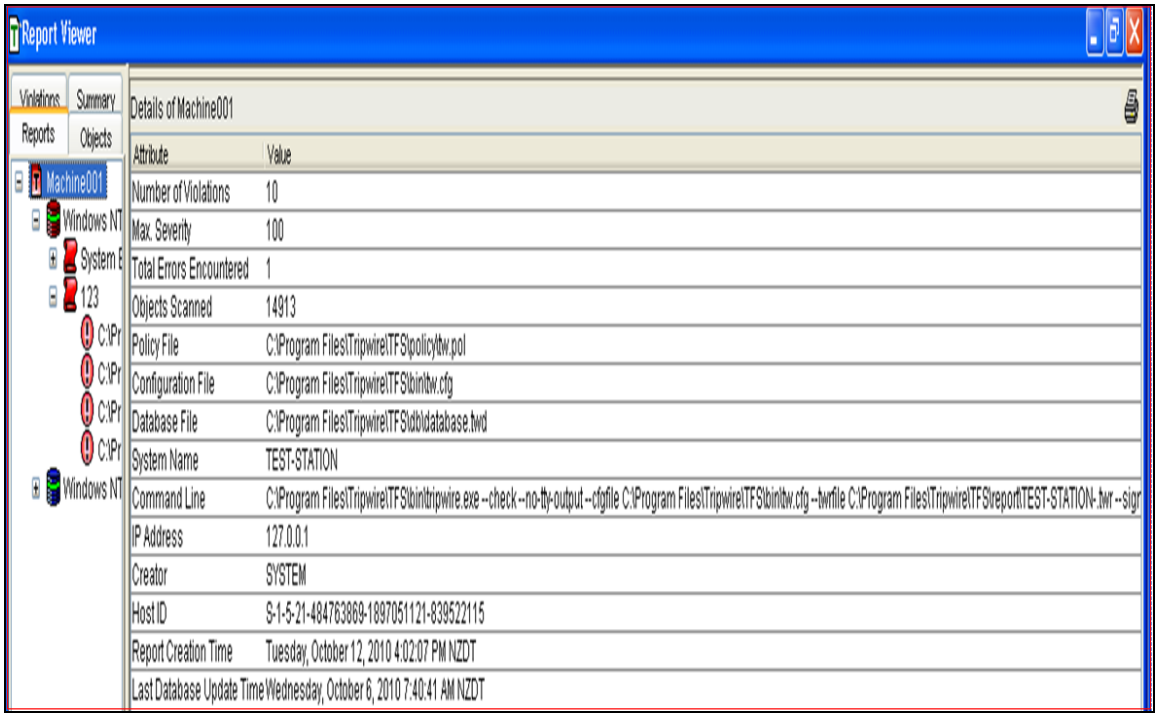
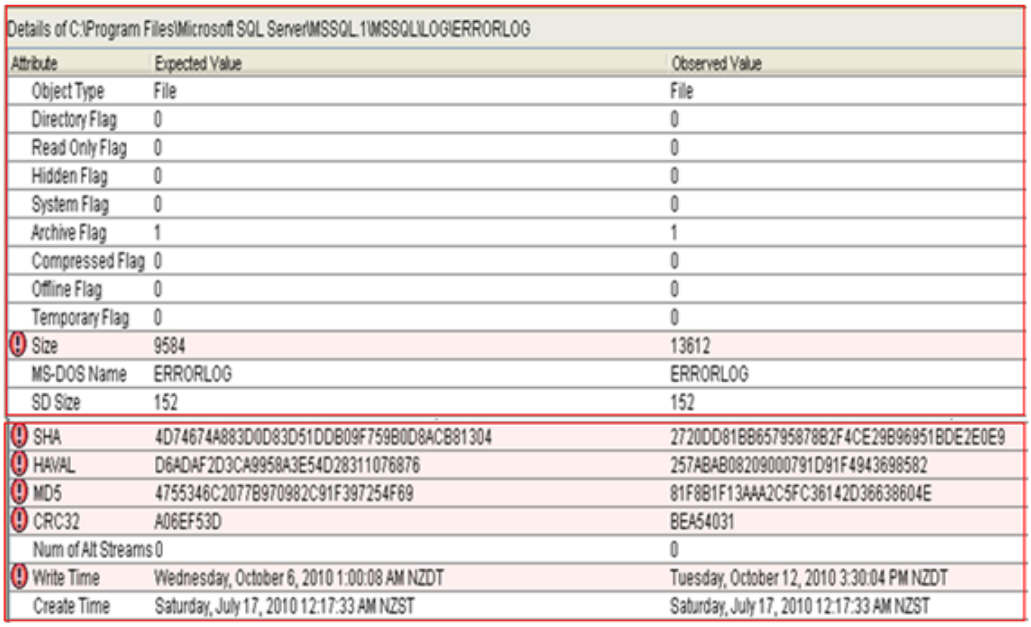


Figure A7.16: Extracted information about Tripwire report

Step 11: The system was stabilised and the hash values of SQL Server error log

monitored by Tripwire for Servers extracted from the Tripwire Manager Report was as shown in the figure below.



Details of C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG

Attribute	Expected Value	Observed Value
Object Type	File	File
Directory Flag	0	0
Read Only Flag	0	0
Hidden Flag	0	0
System Flag	0	0
Archive Flag	1	1
Compressed Flag	0	0
Offline Flag	0	0
Temporary Flag	0	0
Size	9584	13612
MS-DOS Name	ERRORLOG	ERRORLOG
SD Size	152	152
SHA	4D74674A883D0D83D51DDB09F759B0D8ACB81304	2720DD81BB65795878B2F4CE29B96951BDE2E0E9
HAVAL	D6ADAF2D3CA9958A3E54D28311076876	257ABAB08209000791D91F4943698582
MD5	4755346C2077B970982C91F397254F69	81F8B1F13AAA2C5FC36142D36638604E
CRC32	A06EF53D	BEA54031
Num of Alt Streams	0	0
Write Time	Wednesday, October 6, 2010 1:00:08 AM NZDT	Tuesday, October 12, 2010 3:30:04 PM NZDT
Create Time	Saturday, July 17, 2010 12:17:33 AM NZST	Saturday, July 17, 2010 12:17:33 AM NZST

Figure A7.17: The extracted information of server error log from Tripwire report before the attack

The MD5 hash value of the monitoring file was **81F8B1F13AAA2C5FC36142D36638604E** as of Tuesday, at 3:30:04pm on the 12th of October, 2010 before the attack was initiated.

Appendix 8: Steps taken after the SQL poisoning attack was initiated

Step 1: Assuming that the fake tag was not re-written after the attack. The screenshot of the SI attached with a fake RFID tag was read by the RFID scanner. It revealed the fake tag (ID: Fake tag ID: E0040000E90A4302) was read by the POS at 6:39pm on 12th October 2010. Hence the injected malicious code was also found as shown in the figure below. Thus, the evidence acquired from the physical tag (fake one) could be mapped with the data acquired from other areas of RFID BS.

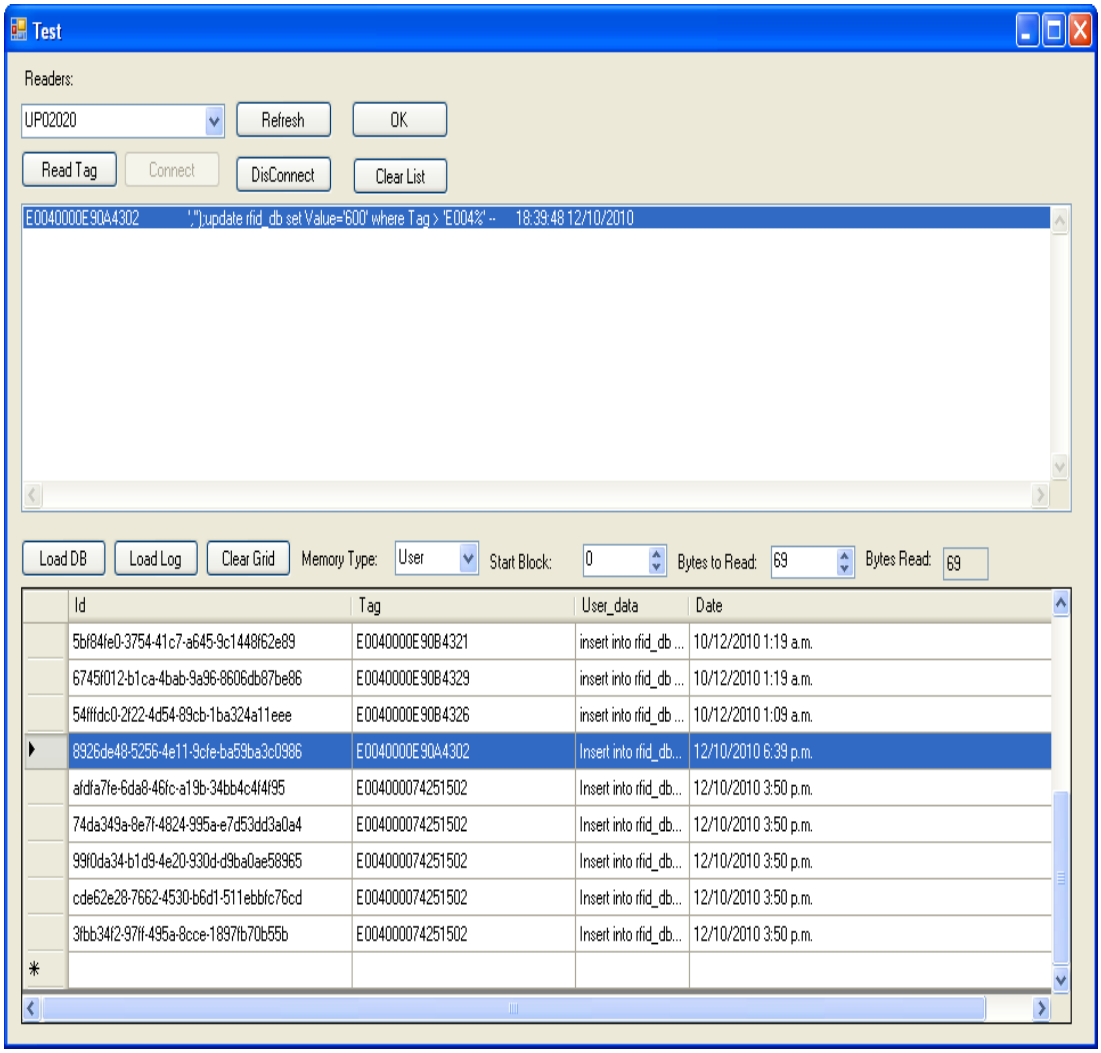


Figure A8.1: Fake RFID tag was read by the scanner

Step 2: The backend database files was loaded to check the effect of the malicious.

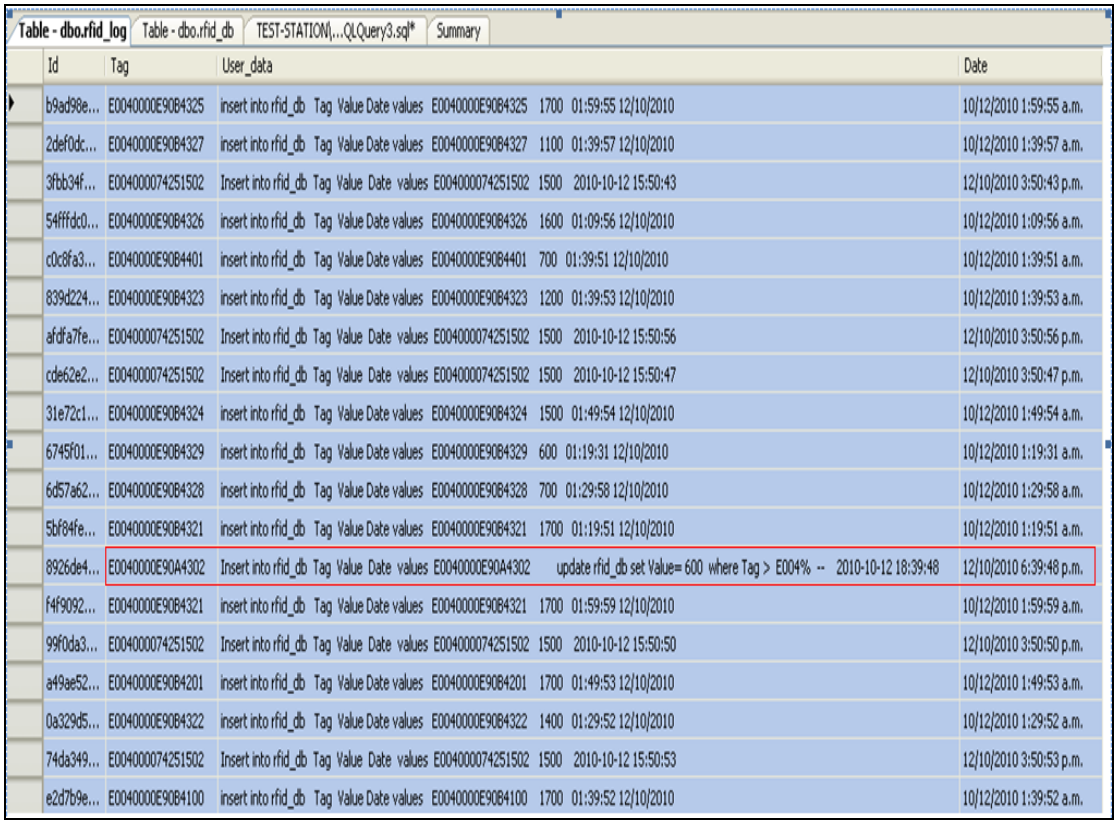


Table - dbo.rfid_log | Table - dbo.rfid_db | TEST-STATION\...QLQuery3.sql* | Summary

Id	Tag	User_data	Date
b9ad98e...	E0040000E90B4325	insert into rfid_db Tag Value Date values E0040000E90B4325 1700 01:59:55 12/10/2010	10/12/2010 1:59:55 a.m.
2def0dc...	E0040000E90B4327	insert into rfid_db Tag Value Date values E0040000E90B4327 1100 01:39:57 12/10/2010	10/12/2010 1:39:57 a.m.
3fbb34f...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:43	12/10/2010 3:50:43 p.m.
54fffdc0...	E0040000E90B4326	insert into rfid_db Tag Value Date values E0040000E90B4326 1600 01:09:56 12/10/2010	10/12/2010 1:09:56 a.m.
c0c8fa3...	E0040000E90B4401	insert into rfid_db Tag Value Date values E0040000E90B4401 700 01:39:51 12/10/2010	10/12/2010 1:39:51 a.m.
839d224...	E0040000E90B4323	insert into rfid_db Tag Value Date values E0040000E90B4323 1200 01:39:53 12/10/2010	10/12/2010 1:39:53 a.m.
afdfa7fe...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:56	12/10/2010 3:50:56 p.m.
cde62e2...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:47	12/10/2010 3:50:47 p.m.
31e72c1...	E0040000E90B4324	insert into rfid_db Tag Value Date values E0040000E90B4324 1500 01:49:54 12/10/2010	10/12/2010 1:49:54 a.m.
6745f01...	E0040000E90B4329	insert into rfid_db Tag Value Date values E0040000E90B4329 600 01:19:31 12/10/2010	10/12/2010 1:19:31 a.m.
6d57a62...	E0040000E90B4328	insert into rfid_db Tag Value Date values E0040000E90B4328 700 01:29:58 12/10/2010	10/12/2010 1:29:58 a.m.
5bf84fe...	E0040000E90B4321	insert into rfid_db Tag Value Date values E0040000E90B4321 1700 01:19:51 12/10/2010	10/12/2010 1:19:51 a.m.
8926de4...	E0040000E90A4302	Insert into rfid_db Tag Value Date values E0040000E90A4302 update rfid_db set Value= 600 where Tag > E004% -- 2010-10-12 18:39:48	12/10/2010 6:39:48 p.m.
f4f9092...	E0040000E90B4321	insert into rfid_db Tag Value Date values E0040000E90B4321 1700 01:59:59 12/10/2010	10/12/2010 1:59:59 a.m.
99f0da3...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:50	12/10/2010 3:50:50 p.m.
a49ae52...	E0040000E90B4201	insert into rfid_db Tag Value Date values E0040000E90B4201 1700 01:49:53 12/10/2010	10/12/2010 1:49:53 a.m.
0a329d5...	E0040000E90B4322	insert into rfid_db Tag Value Date values E0040000E90B4322 1400 01:29:52 12/10/2010	10/12/2010 1:29:52 a.m.
74da349...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:53	12/10/2010 3:50:53 p.m.
e2d7b9e...	E0040000E90B4100	insert into rfid_db Tag Value Date values E0040000E90B4100 1700 01:39:52 12/10/2010	10/12/2010 1:39:52 a.m.

Figure A8.2: Screenshot of the backend SQL Server log file

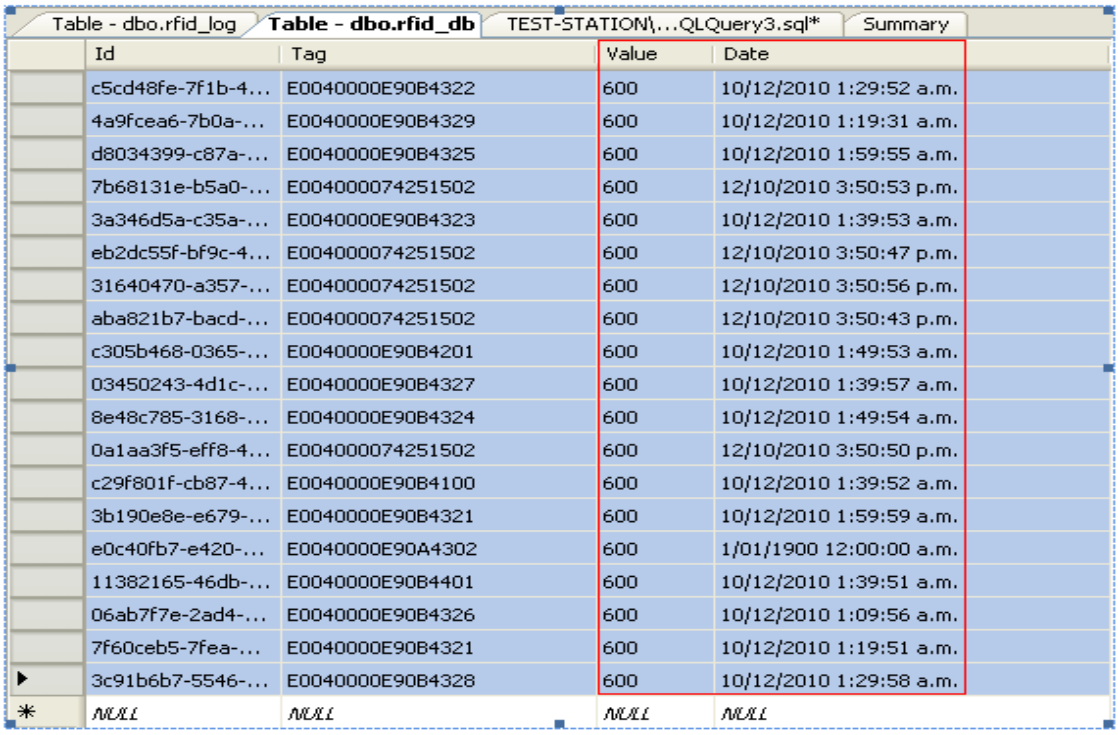


Table - dbo.rfid_log | **Table - dbo.rfid_db** | TEST-STATION\...QLQuery3.sql* | Summary

Id	Tag	Value	Date
c5cd48fe-7f1b-4...	E0040000E90B4322	600	10/12/2010 1:29:52 a.m.
4a9fcea6-7b0a-...	E0040000E90B4329	600	10/12/2010 1:19:31 a.m.
d8034399-c87a-...	E0040000E90B4325	600	10/12/2010 1:59:55 a.m.
7b68131e-b5a0-...	E004000074251502	600	12/10/2010 3:50:53 p.m.
3a346d5a-c35a-...	E0040000E90B4323	600	10/12/2010 1:39:53 a.m.
eb2dc55f-bf9c-4...	E004000074251502	600	12/10/2010 3:50:47 p.m.
31640470-a357-...	E004000074251502	600	12/10/2010 3:50:56 p.m.
aba821b7-bacd-...	E004000074251502	600	12/10/2010 3:50:43 p.m.
c305b468-0365-...	E0040000E90B4201	600	10/12/2010 1:49:53 a.m.
03450243-4d1c-...	E0040000E90B4327	600	10/12/2010 1:39:57 a.m.
8e48c785-3168-...	E0040000E90B4324	600	10/12/2010 1:49:54 a.m.
0a1aa3f5-eff8-4...	E004000074251502	600	12/10/2010 3:50:50 p.m.
c29f801f-cb87-4...	E0040000E90B4100	600	10/12/2010 1:39:52 a.m.
3b190e8e-e679-...	E0040000E90B4321	600	10/12/2010 1:59:59 a.m.
e0c40fb7-e420-...	E0040000E90A4302	600	1/01/1900 12:00:00 a.m.
11382165-46db-...	E0040000E90B4401	600	10/12/2010 1:39:51 a.m.
06ab7f7e-2ad4-...	E0040000E90B4326	600	10/12/2010 1:09:56 a.m.
7f60ceb5-7fea-...	E0040000E90B4321	600	10/12/2010 1:19:51 a.m.
3c91b6b7-5546-...	E0040000E90B4328	600	10/12/2010 1:29:58 a.m.
NULL	NULL	NULL	NULL

Figure A8.3: Screenshot of the backend database file

As shown in Figure A8.3, all of the values of SI were changed to the value $600 after the attack.

Step 3: The integrity check was initiated by using Tripwire Manager (Figure A8.4). Hence, the passphrase was needed to enter (Figure A8.5).

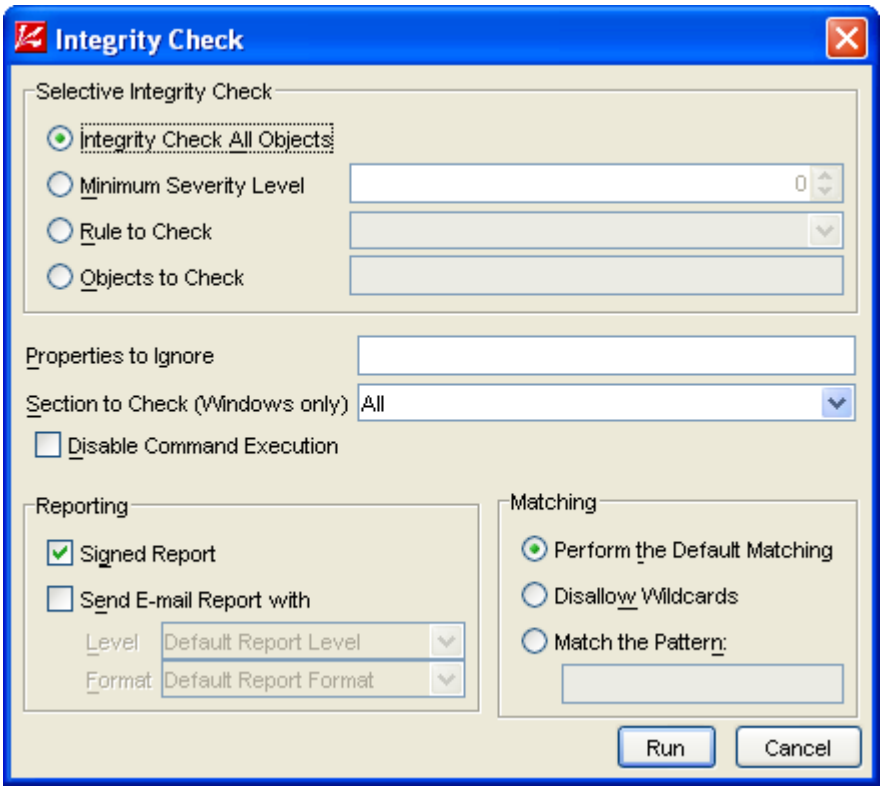


Figure A8.4: Integrity check was initiated after the attack

Figure A8.5: Keying the passphrase (qwe123AUT) to logon Tripwire Manager

Step 4: A few screenshots were taken during the integrity check by using Tripwire Manager (for example; Figure A8.6).

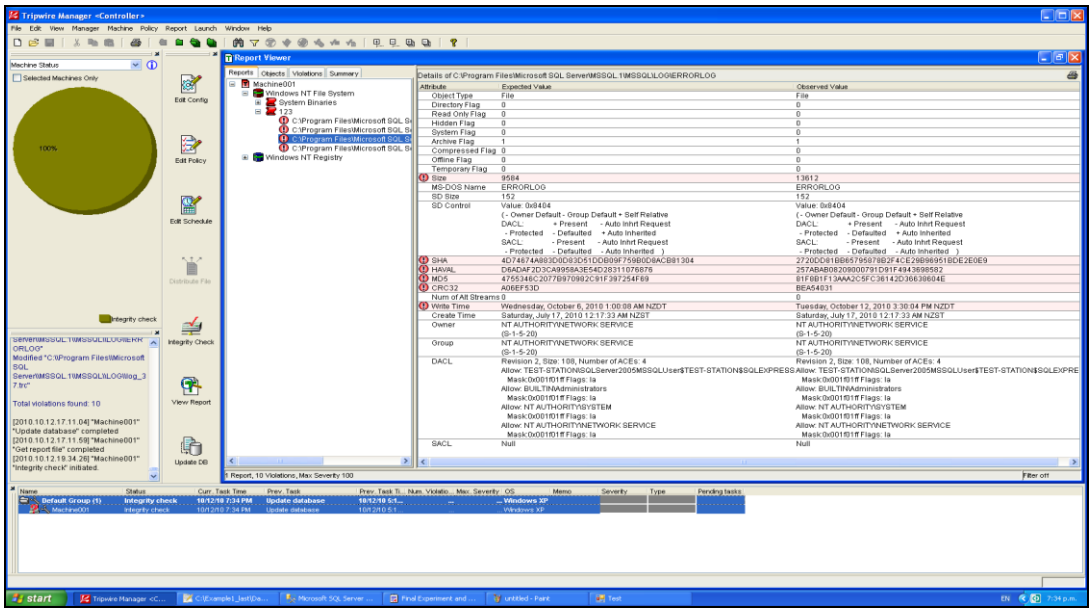


Figure A8.6: Screenshot during the integrity check

Step 5: The integrity check was completed. The following information was a snippet from the output of Tripwire Manager (as the output file has more than 100 pages).

```
[2010.10.12.17.11.04] "Machine001" "Update database" completed
[2010.10.12.17.11.59] "Machine001" "Get report file" completed
[2010.10.12.19.34.26] "Machine001" "Integrity check" initiated.
Parsing policy file: C:\Program Files\Tripwire\TFS\policy\tw.pol
*** Gathering Event Data. ***
*** Processing Windows File System ***
Performing integrity check...
*** Processing Windows Registry ***
Performing integrity check...
### ….
### ….
### ….
### ….
### ….
### Continuing...
Please enter your local passphrase:
Wrote report file: C:\Program Files\Tripwire\TFS\report\TEST-STATION-.twr
Integrity check complete.
[2010.10.12.19.35.09] "Machine001" "Integrity check" completed
[2010.10.12.19.35.21] "Machine001" "Get report file" completed
```

Figure A8.7: Snippet of Tripwire Manger Output File

Step 6: The report was viewed by using Tripwire Manager after the attack (see Figures A8.8 and A8.9).

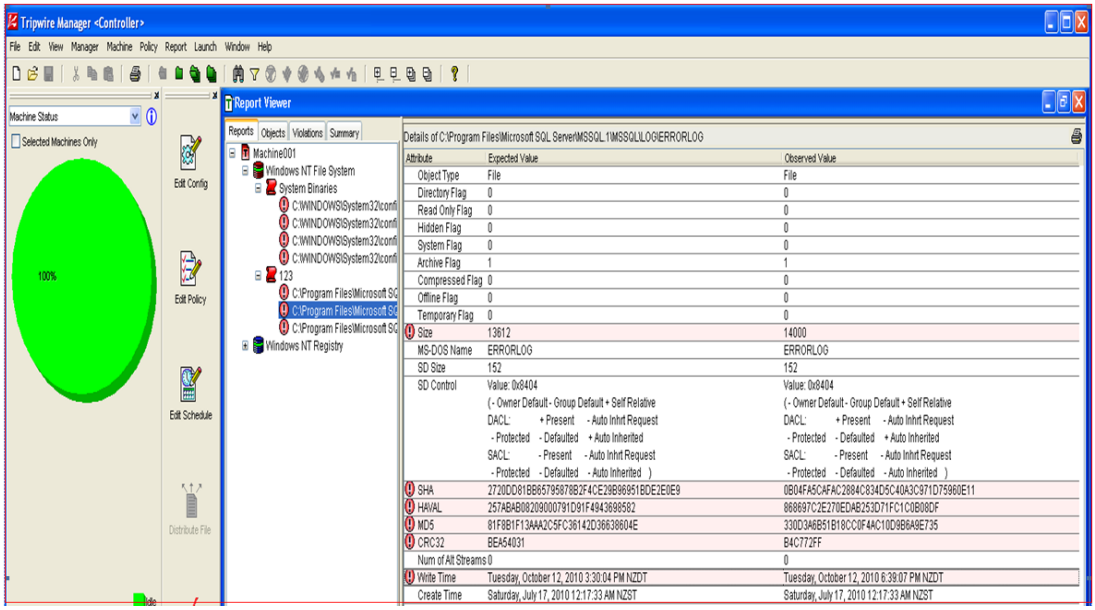


Figure A8.8: Screenshot of the Tripwire report on SQL Server error log

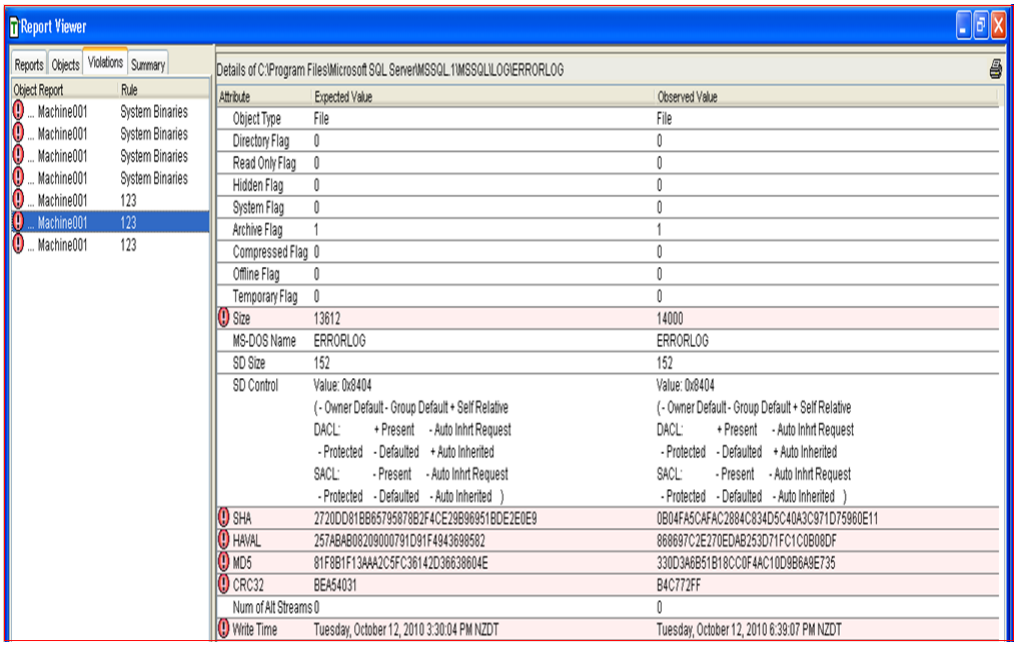
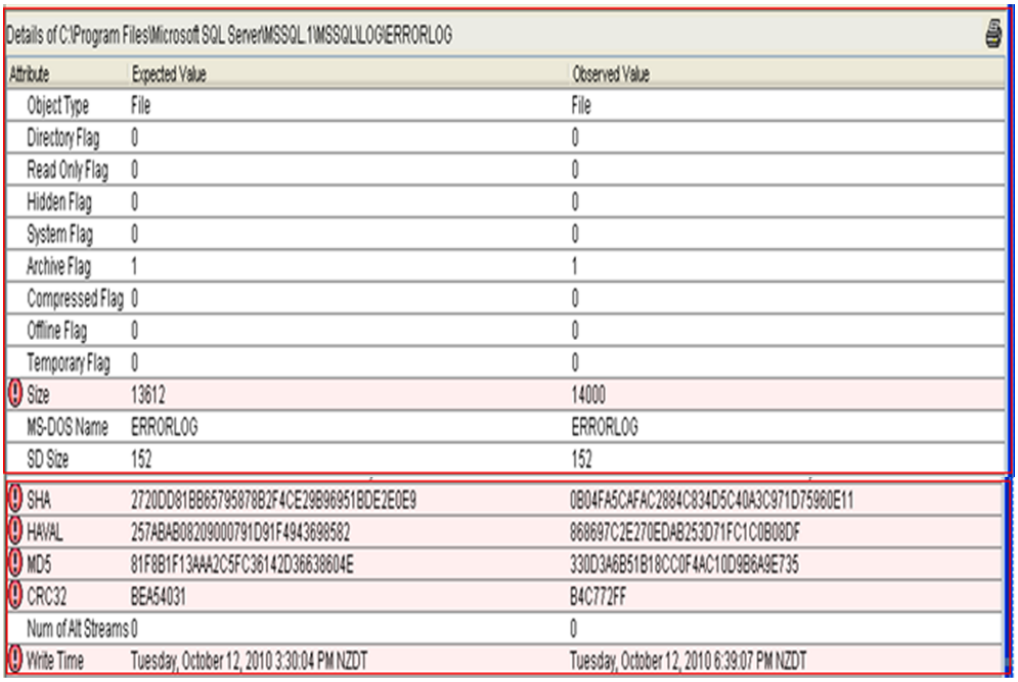


Figure A8.9: Information of the Tripwire report on SQL Server error log

Step 7: The report information was extracted to check the hash values of the SQL Server error log before and after the attack (see Figures A8.10).



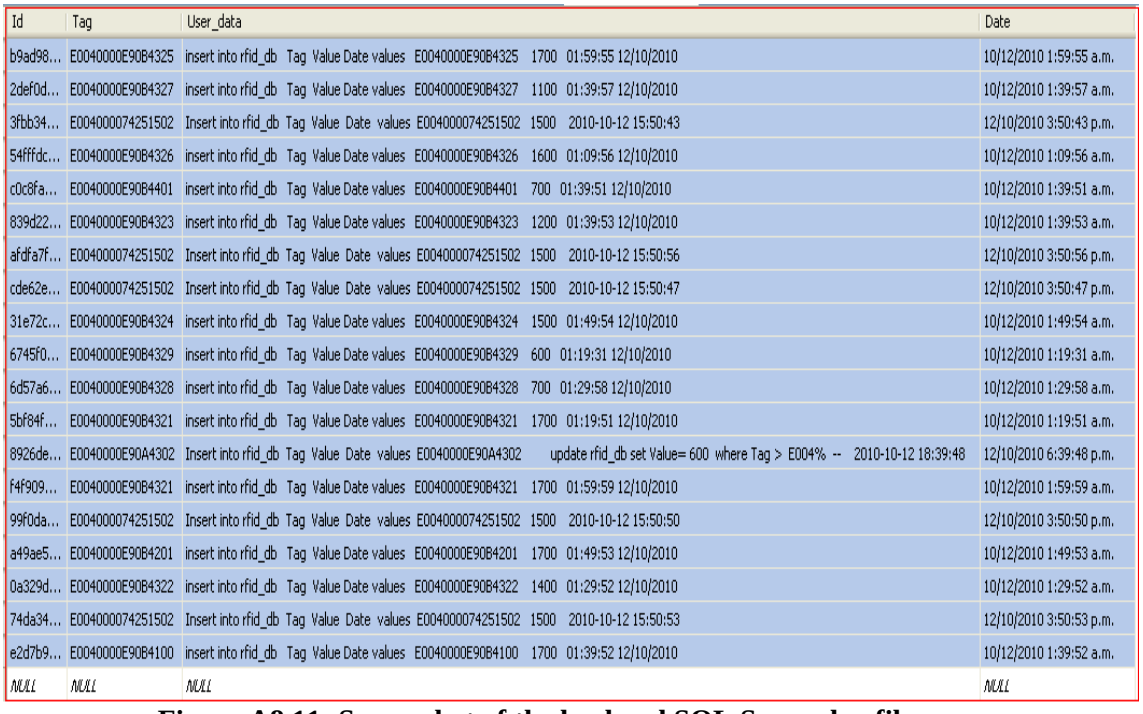
Details of C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG

Attribute	Expected Value	Observed Value
Object Type	File	File
Directory Flag	0	0
Read Only Flag	0	0
Hidden Flag	0	0
System Flag	0	0
Archive Flag	1	1
Compressed Flag	0	0
Offline Flag	0	0
Temporary Flag	0	0
Size	13612	14000
MS-DOS Name	ERRORLOG	ERRORLOG
SD Size	152	152
SHA	2720DD81BB65795878B2F4CE29B969518DE2E0E9	0B04FA5CAFAC2884C834D5C40A3C971D75960E11
HAVAL	2574BAB08209000791D91F4943698582	868697C2E270EDAB253D71FC1C0B08DF
MD5	81F8B1F13AAA2C5FC36142D36638604E	330D3A6B51B18CC0F4AC10D9B6A9E735
CRC32	BEA54031	B4C772FF
Num of Alt Streams	0	0
Write Time	Tuesday, October 12, 2010 3:30:04 PM NZDT	Tuesday, October 12, 2010 6:39:07 PM NZDT

Figure A8.10: Extracted hash values of SQL Server error log from Tripwire Manager Report before and after the attack

It was acknowledged that the MD5 hash values before and after the attack were not the same according the Tripwire report.

Step 8: The database files from the backend SQL 2005 Server were also viewed in order to check whether the attack was successful or not (see Figures A8.11 and A8.12).



Id	Tag	User_data	Date
b9ad98...	E0040000E90B4325	insert into rfid_db Tag Value Date values E0040000E90B4325 1700 01:59:55 12/10/2010	10/12/2010 1:59:55 a.m.
2def0d...	E0040000E90B4327	insert into rfid_db Tag Value Date values E0040000E90B4327 1100 01:39:57 12/10/2010	10/12/2010 1:39:57 a.m.
3fbb34...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:43	12/10/2010 3:50:43 p.m.
54fffdc...	E0040000E90B4326	insert into rfid_db Tag Value Date values E0040000E90B4326 1600 01:09:56 12/10/2010	10/12/2010 1:09:56 a.m.
c0c8fa...	E0040000E90B4401	insert into rfid_db Tag Value Date values E0040000E90B4401 700 01:39:51 12/10/2010	10/12/2010 1:39:51 a.m.
839d22...	E0040000E90B4323	insert into rfid_db Tag Value Date values E0040000E90B4323 1200 01:39:53 12/10/2010	10/12/2010 1:39:53 a.m.
afdfa7f...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:56	12/10/2010 3:50:56 p.m.
cde62e...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:47	12/10/2010 3:50:47 p.m.
31e72c...	E0040000E90B4324	insert into rfid_db Tag Value Date values E0040000E90B4324 1500 01:49:54 12/10/2010	10/12/2010 1:49:54 a.m.
6745f0...	E0040000E90B4329	insert into rfid_db Tag Value Date values E0040000E90B4329 600 01:19:31 12/10/2010	10/12/2010 1:19:31 a.m.
6d57a6...	E0040000E90B4328	insert into rfid_db Tag Value Date values E0040000E90B4328 700 01:29:58 12/10/2010	10/12/2010 1:29:58 a.m.
5bf84f...	E0040000E90B4321	insert into rfid_db Tag Value Date values E0040000E90B4321 1700 01:19:51 12/10/2010	10/12/2010 1:19:51 a.m.
8926de...	E0040000E90A4302	Insert into rfid_db Tag Value Date values E0040000E90A4302 update rfid_db set Value= 600 where Tag > E004% -- 2010-10-12 18:39:48	12/10/2010 6:39:48 p.m.
f4f909...	E0040000E90B4321	insert into rfid_db Tag Value Date values E0040000E90B4321 1700 01:59:59 12/10/2010	10/12/2010 1:59:59 a.m.
99f0da...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:50	12/10/2010 3:50:50 p.m.
a49ae5...	E0040000E90B4201	insert into rfid_db Tag Value Date values E0040000E90B4201 1700 01:49:53 12/10/2010	10/12/2010 1:49:53 a.m.
0a329d...	E0040000E90B4322	insert into rfid_db Tag Value Date values E0040000E90B4322 1400 01:29:52 12/10/2010	10/12/2010 1:29:52 a.m.
74da34...	E004000074251502	Insert into rfid_db Tag Value Date values E004000074251502 1500 2010-10-12 15:50:53	12/10/2010 3:50:53 p.m.
e2d7b9...	E0040000E90B4100	insert into rfid_db Tag Value Date values E0040000E90B4100 1700 01:39:52 12/10/2010	10/12/2010 1:39:52 a.m.
NULL	NULL	NULL	NULL

Figure A8.11: Screenshot of the backend SQL Server log file

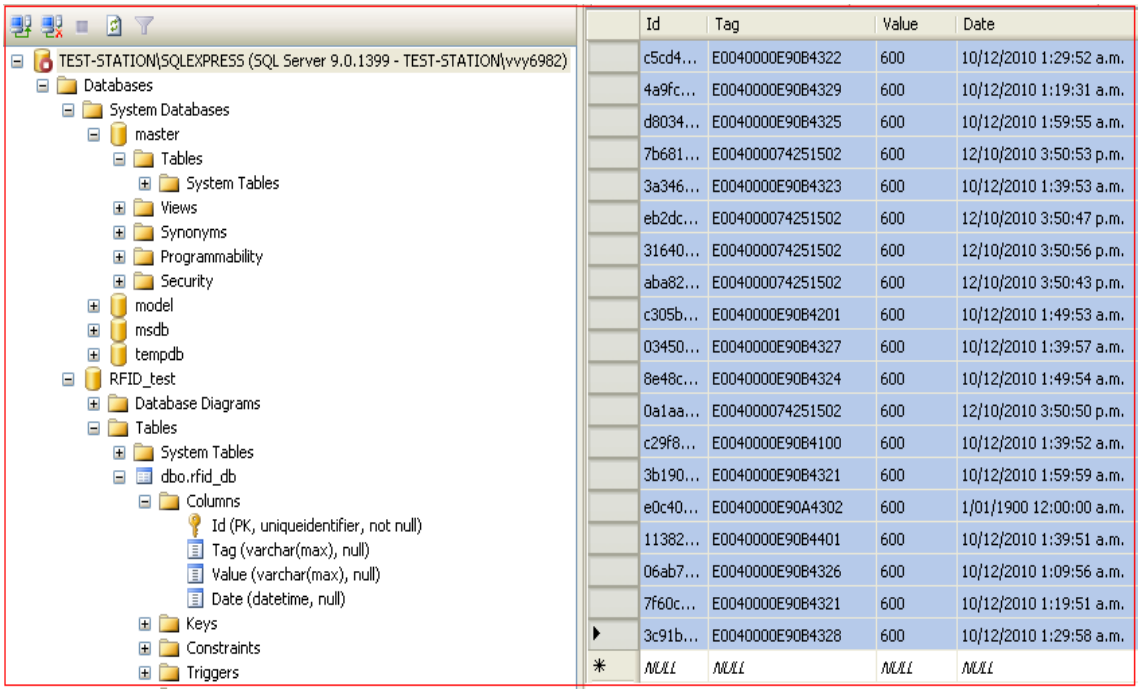


Figure A8.12: Screenshot of the backend database file

As shown in Figures (A8.11 and A8.12) above, all of the values of SI were changed to the value $600 on the backend SQL Server after the attack. Hence, according to the different hash values of the monitoring file from Tripwire report and further verification on the backend SQL Server; the administrator of the RFID based retail shop confirmed that the changes made in the databases were unauthorized and escalated the problem to the incident respond team or forensic investigator via the owner of the retail shop. Then the forensic investigation was initiated.

Appendix 9: Screenshots of Compromised TEST-STATION's System and Installed Software Information

As a result of the hardware and software compatibility issues, the operating system of the point of sale (POS) **TEST-STATION** was installed with Microsoft Windows XP Professional Service Pack 2. The detail system and installed software information of the POS **TEST-STATION** could be seen in the following figures.

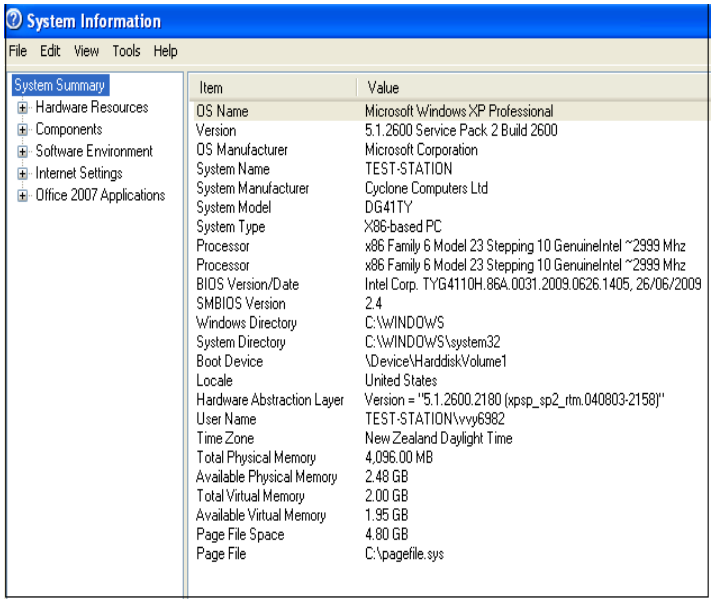


Figure A9.1: TEST-STATION System Information Summary

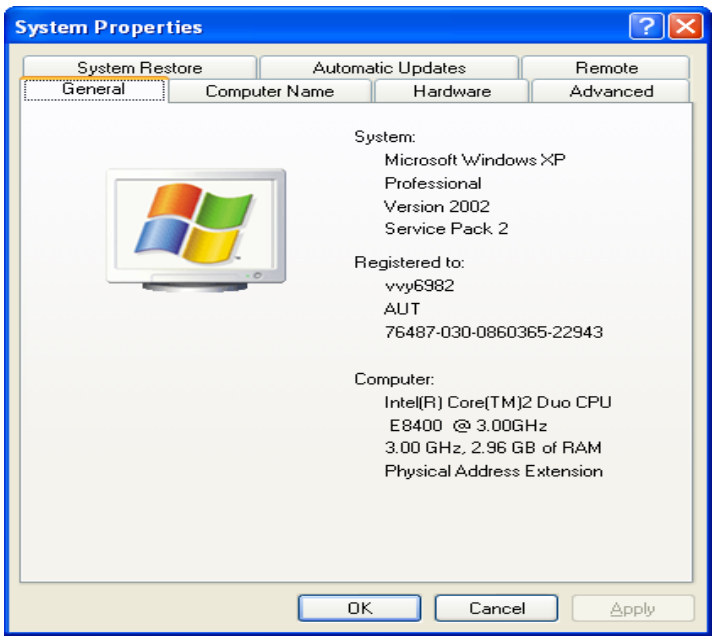


Figure A9.2: TEST-STATION System Properties Information

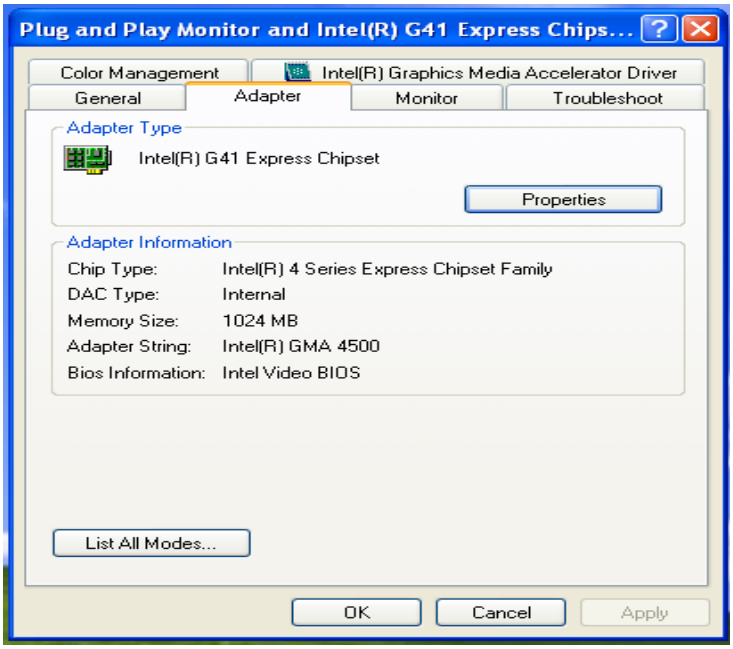


Figure A9.3: Network Interface Card Information installed on the TEST-STATION

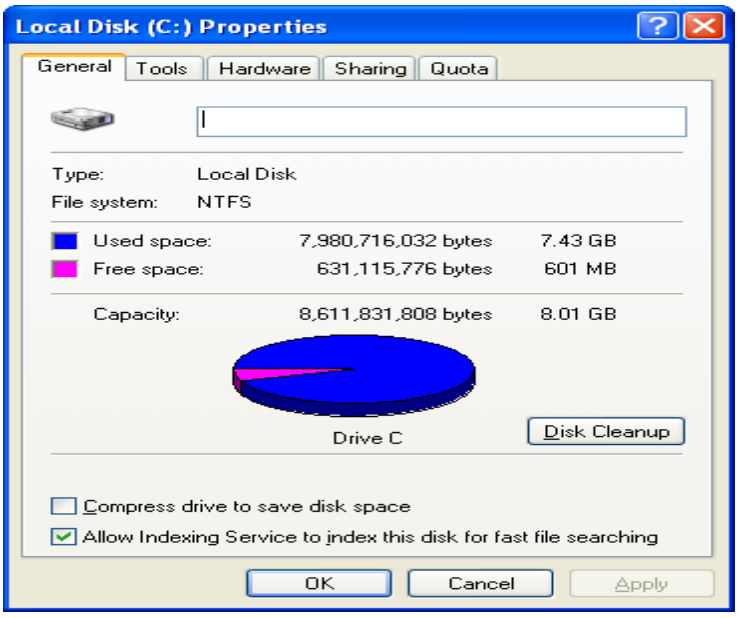


Figure A9.4: Local Disk (C:\) Properties Information of the TEST-STATION

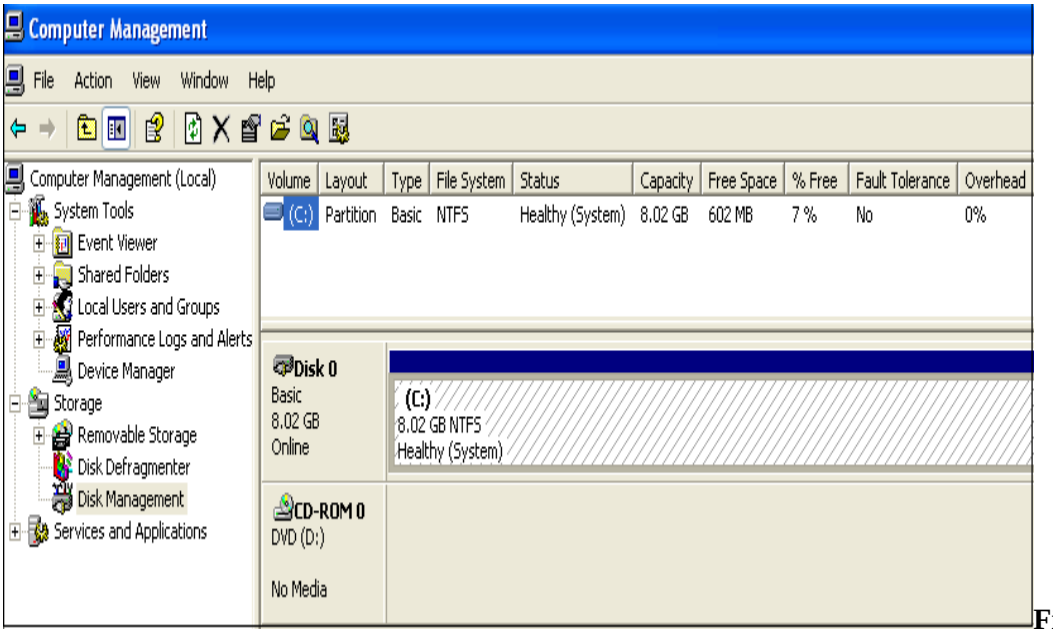


Figure A9.5: Only one NTFS partition on the physical drive of the TEST-STATION

Figure A9.6: SQL Server Management Studio Express is installed on the TEST-STATION

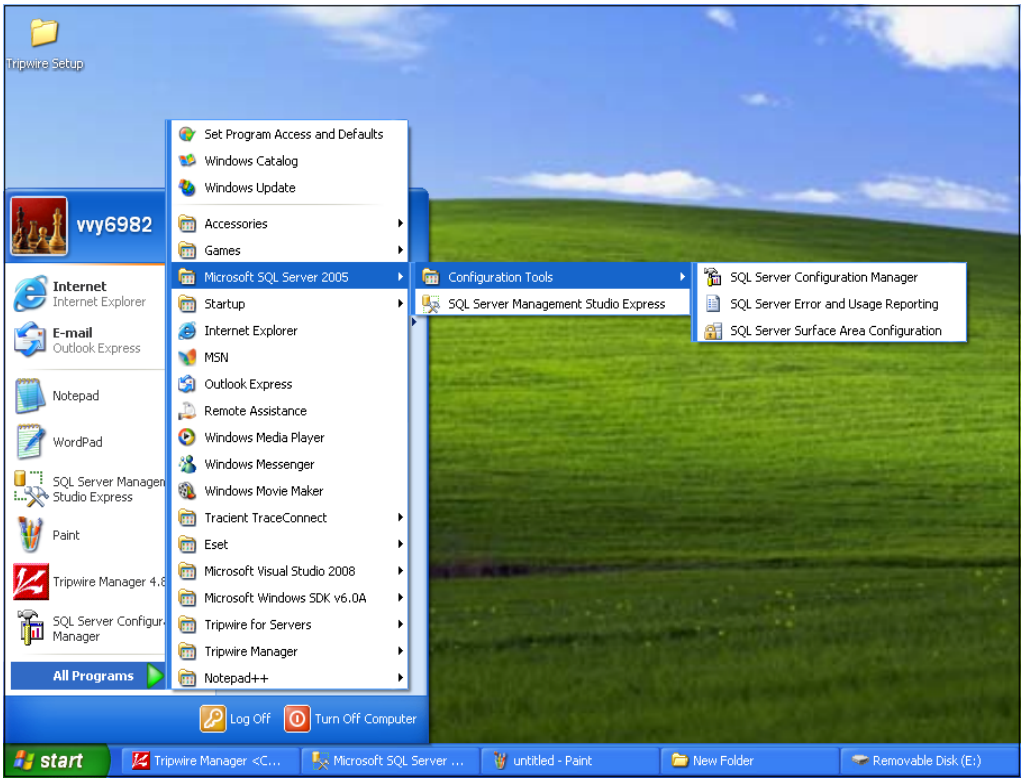


Figure A9.7: Enterprise Version of SQL Server 2005 is installed on the TEST-STATION

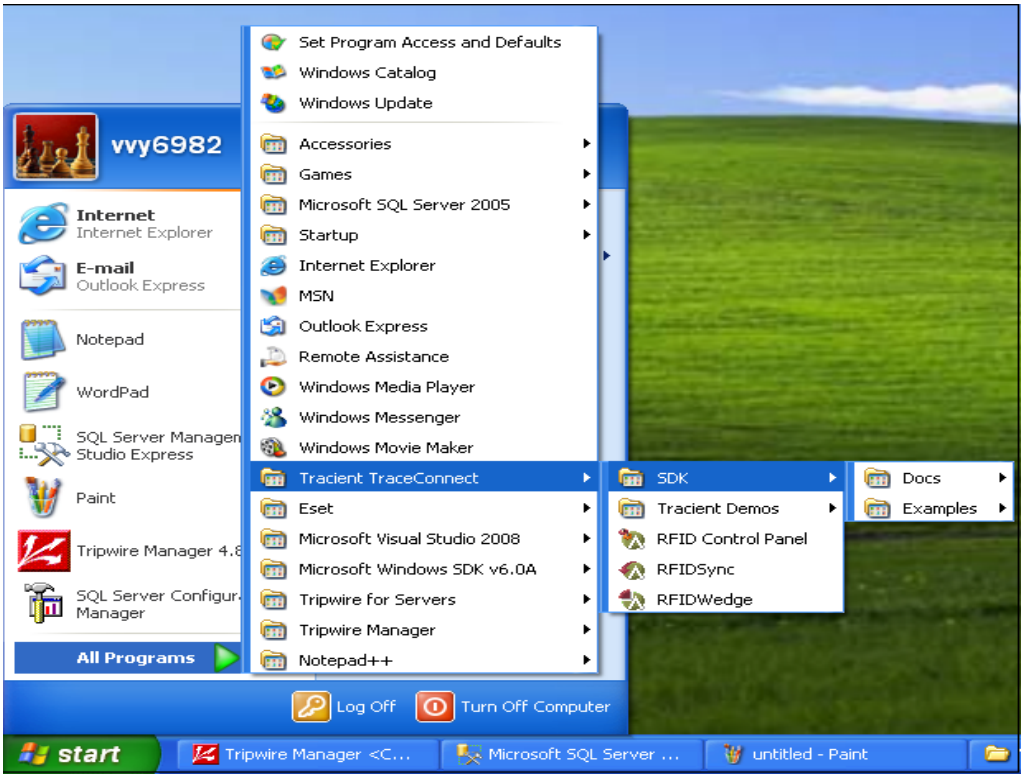


Figure A9.8: Tracient TraceConnect Software (SDK) is installed on the TEST-STATION

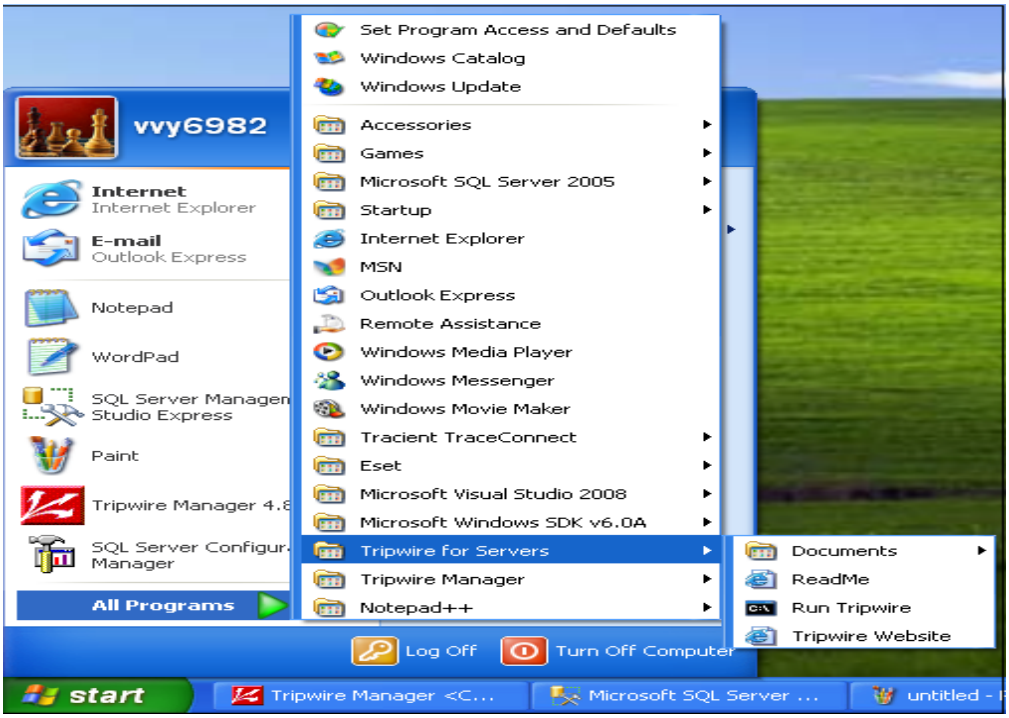


Figure A9.9: Tripwire for Servers (Version 4.8) is installed on the TEST-STATION

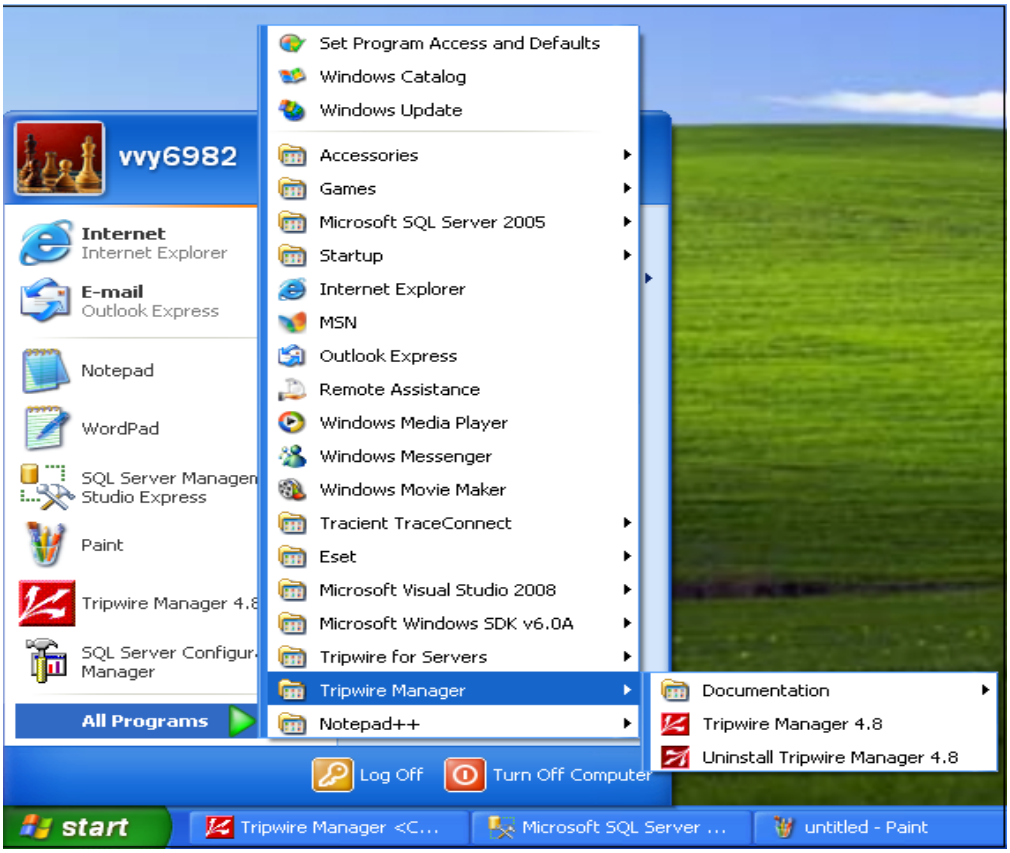


Figure A9.10: Tripwire for Managers (Version 4.8) is installed on the TEST-STATION

Appendix 10: Screenshots of Forensic Station - System Information and Installed Software Information

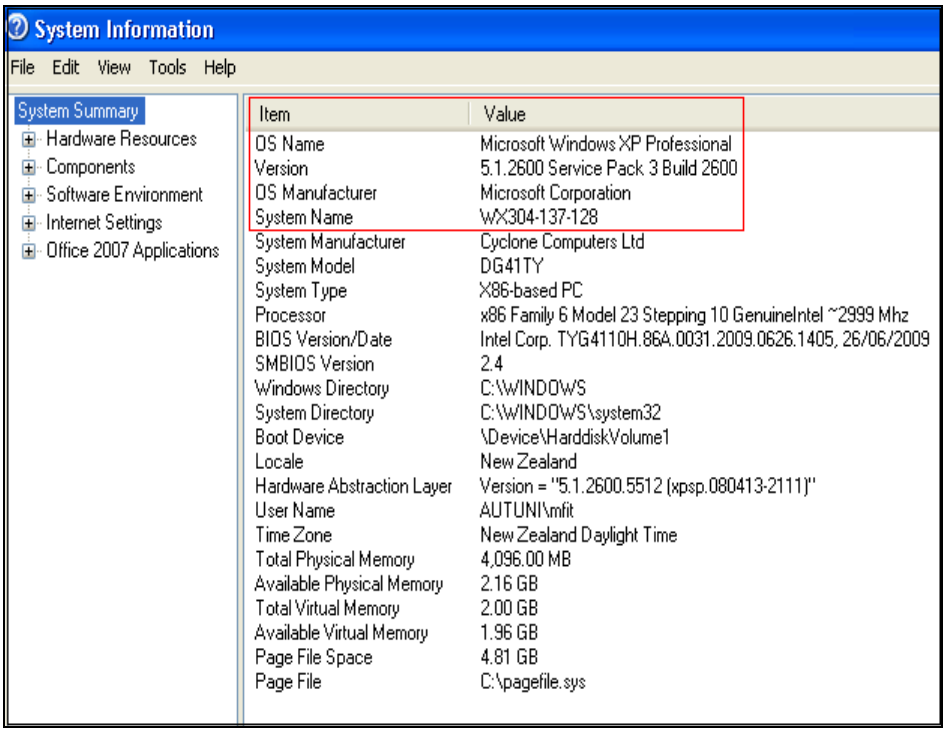


Figure A10. 1: Forensic Workstation System Information Summary

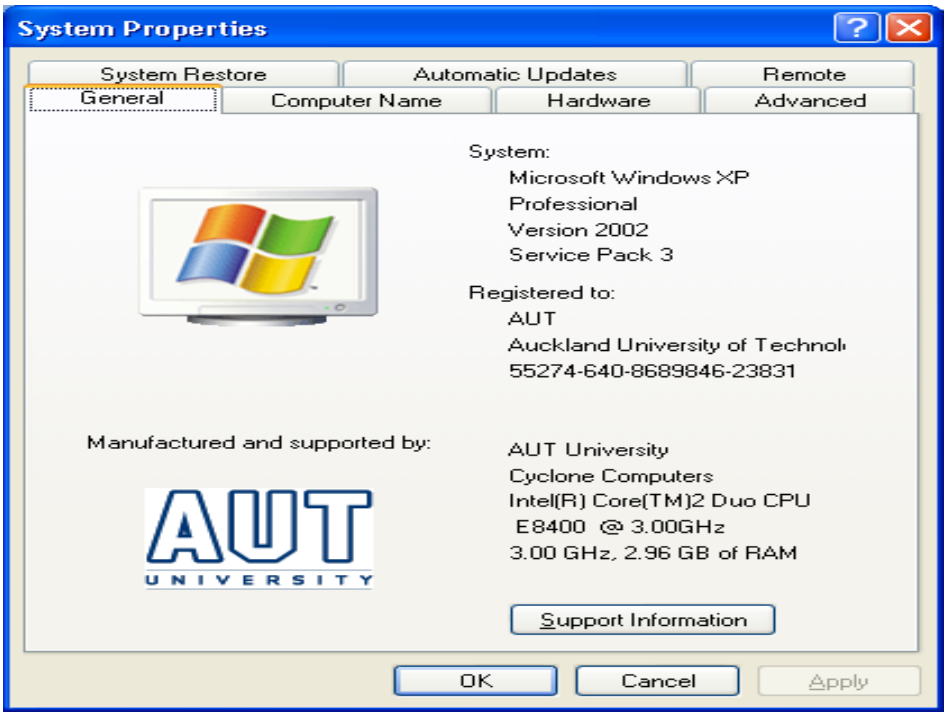


Figure A10. 2: Forensic Workstation System Properties Information

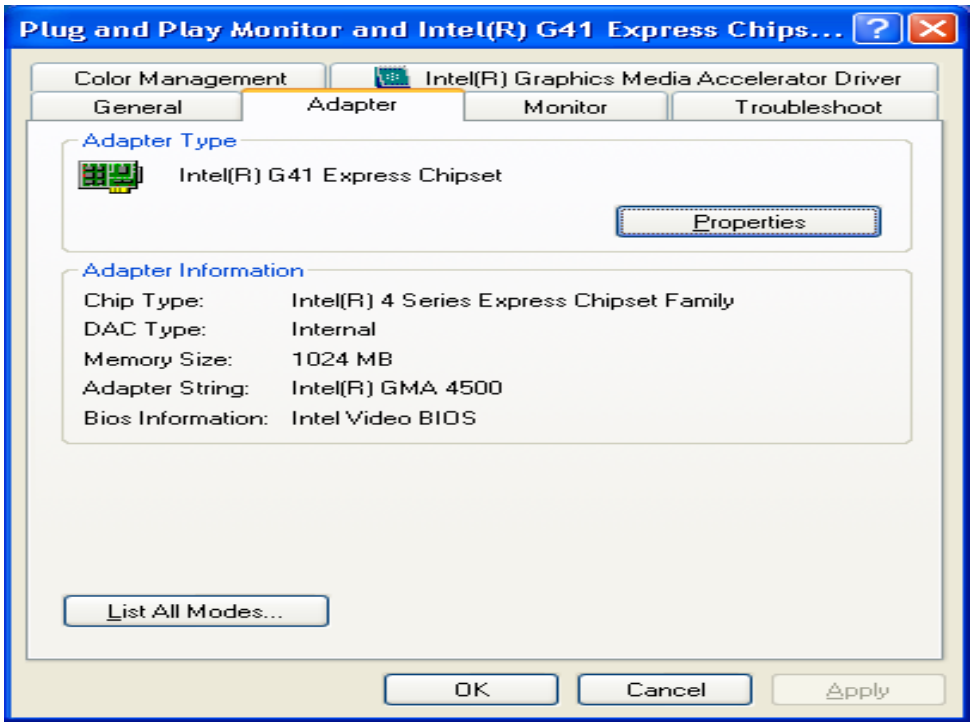


Figure A10. 3: Network Interface Card Information installed on the TEST-STATION

Figure A10. 4: EnCase Forensic Training Software (Version 6.16.1) installed on the Forensic Workstation

Appendix 11: Forensically Sterilizing Flash Drives for Digital Investigation

For the purpose of research experiment, two flash drives are forensically wiped or sterilized (one for storage of collected data during acquisition process and one for dcfldd USB toolkit).

For collected data storage: The following figure is a screenshot of forensically wiping a flash drive (capacity 16GB) by using BackTrack 4 Live DVD (which is a Linux-based penetration testing distribution; source: http://www.backtrack-linux.org/) that will be used as a storage destination for all collected data during the investigation.

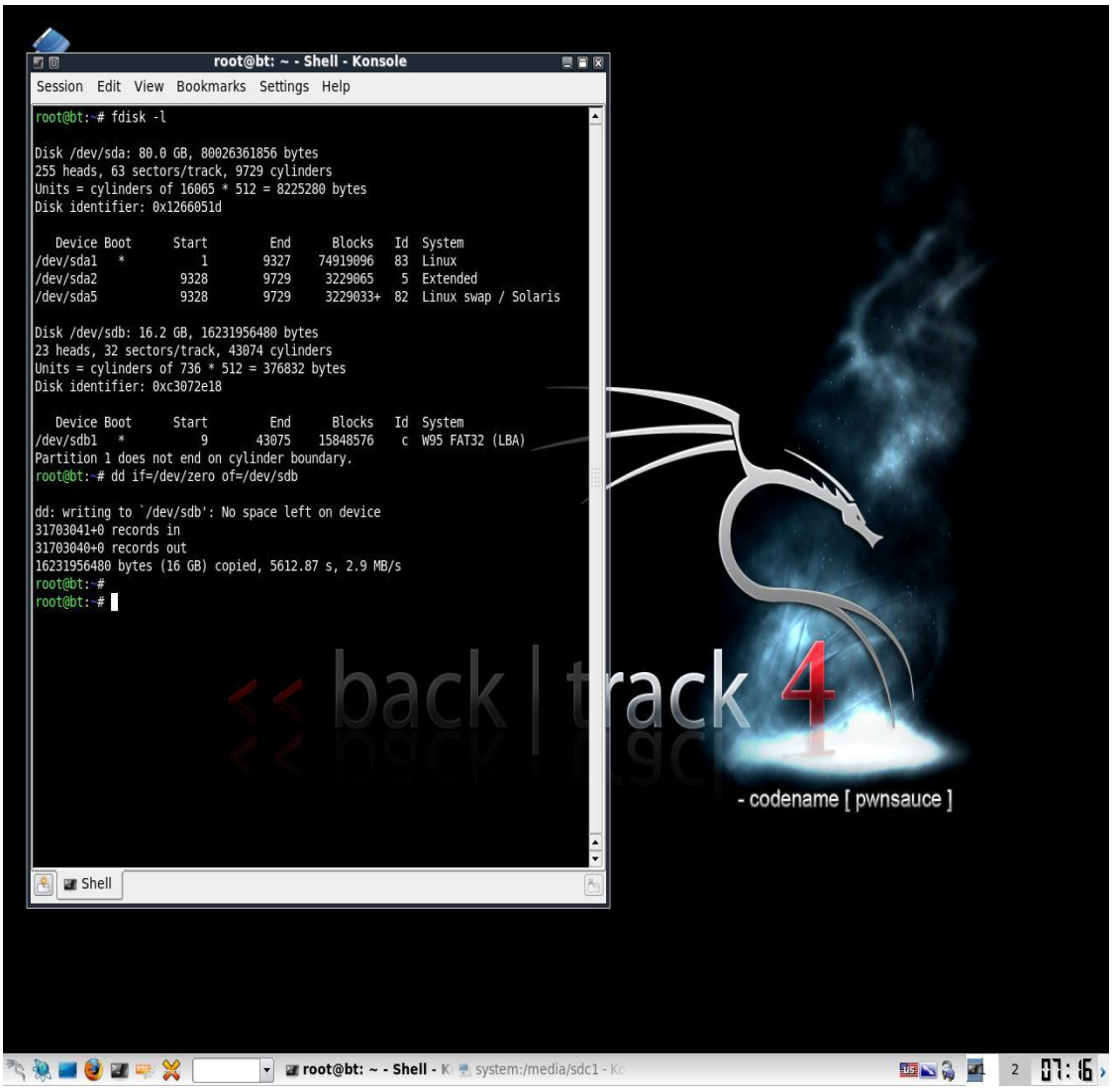


Figure A11. 1: Screenshot of successfully wiped a flash drive

For dcfldd tool: The following steps/figures are the screenshots of forensically wiping a flash drive by using EnCase Forensics Training Software (**Version 6.16.1)** that is used as a USB thumb drive dcfldd acquisition tool for all collected data during the investigation.

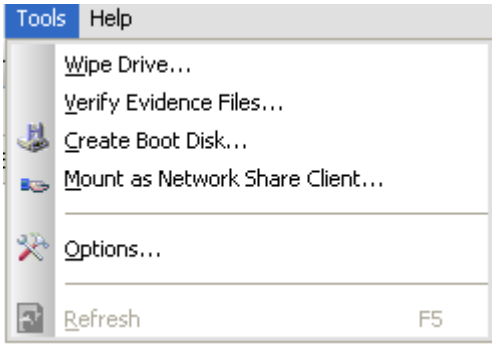


Figure A11. 2: Opening the EnCase on the forensic workstation and click on Tools menu (Step 1)

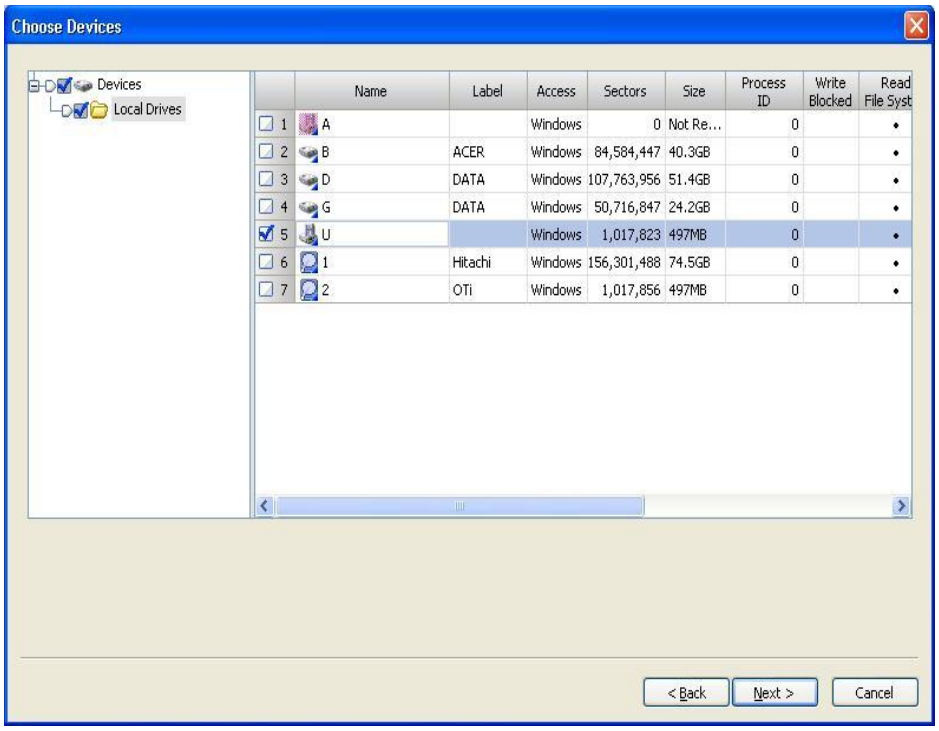


Figure A11. 3: Choose the USB thumb drive to wipe from Local drive (Step 2)

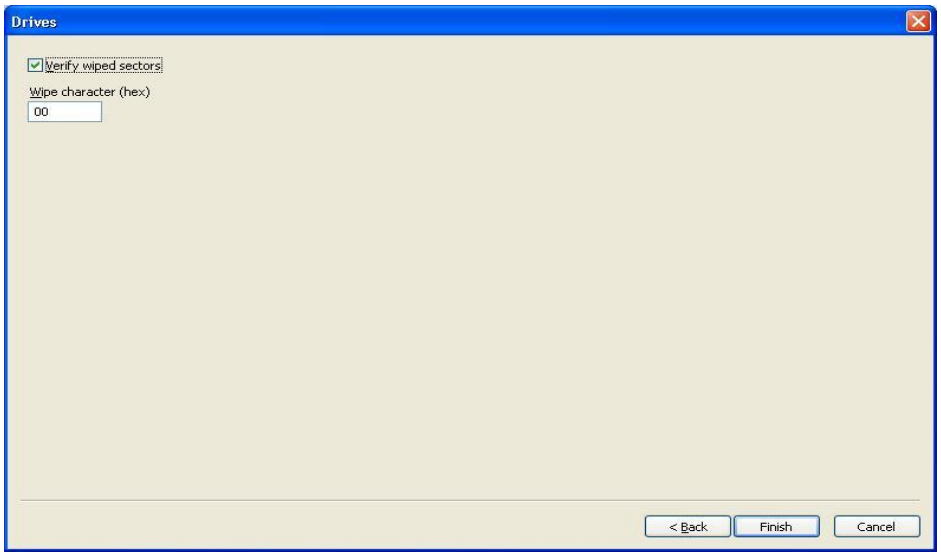


Figure A11. 4: Wiping all the sectors of USB drive with hexadecimal values "00" (Step 3)

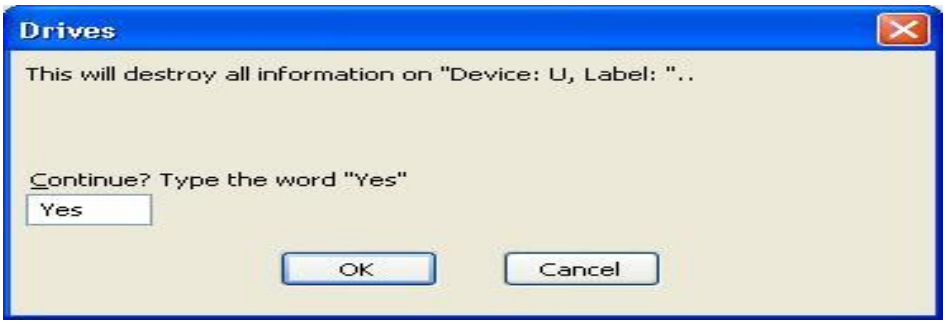


Figure A11. 5: Confirming to wipe the USB drive (Step 4)

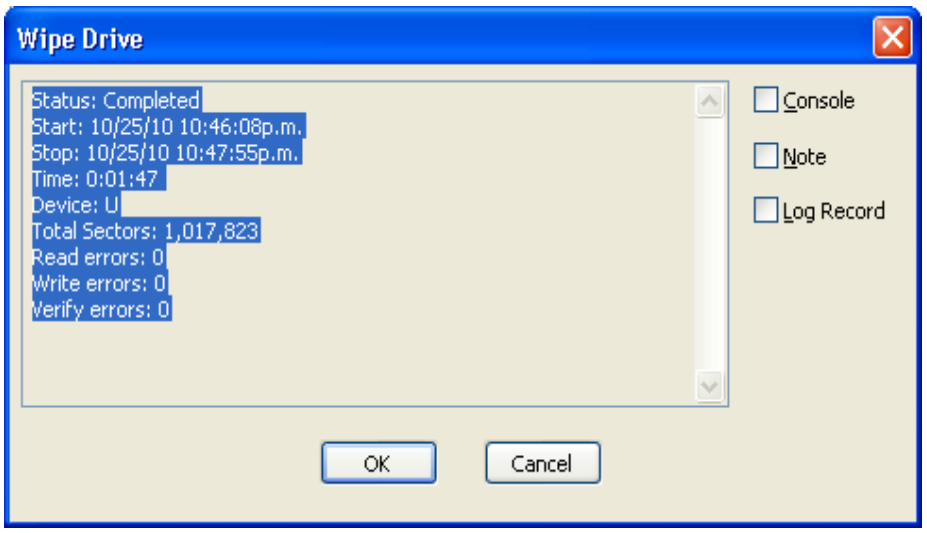


Figure A11. 6: Screenshot of the wipe process completion (Step 5)

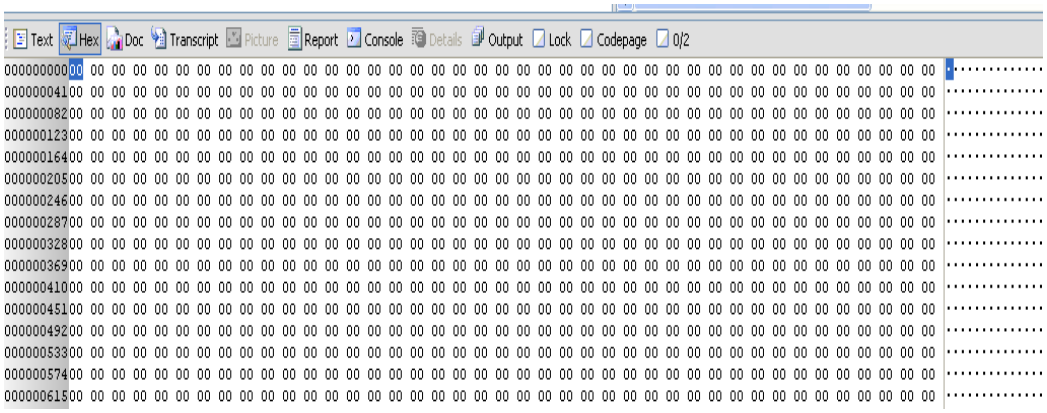


Figure A11. 7: Previewing the sterilized USB drive after wiping (Step 6)

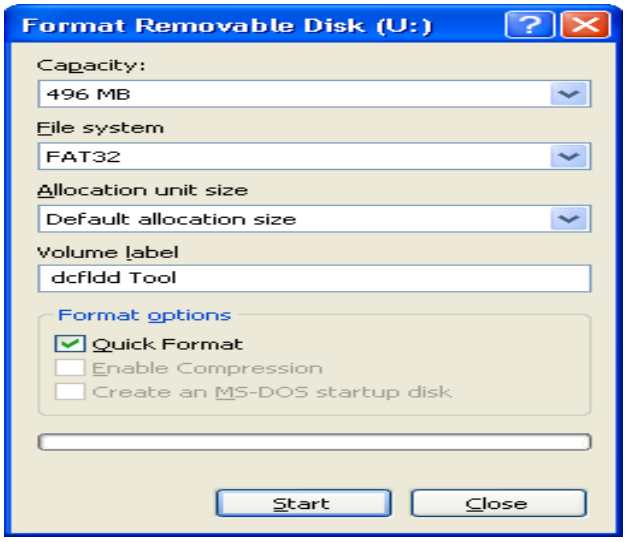


Figure A11. 8: Formatting the sterilized USB drive with FAT 32 after wiping (Step 7)

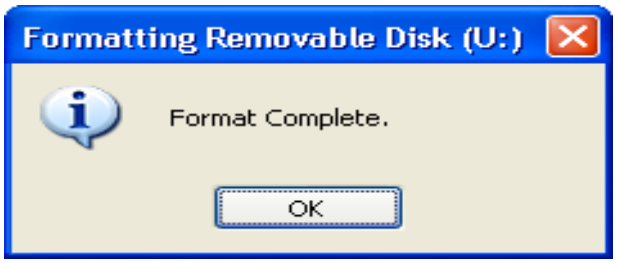


Figure A11. 9: Formatting completed screenshot of the sterilized USB drive (Step 8)

Finally, all the USB drives are forensically sterilized and ready to use for the specific purposes mentioned above.

Appendix 12: Steps for Copying and Hashing the Original Evidence Images from Flash (Evidence) Drive

1. Tableau Forensic USB Bridge Write Blocker (WB) was connected to the forensic work-station (FW).
2. The WB was tested without attaching the evidence drive before copying the original evidence images from the flash drive (which was ARKAR_USB).
3. After confirming the WB device was in working condition, the flash drive was attached to the WB via USB port. As seen in the Figure A12.1, the drive is detected as B: on the FW.

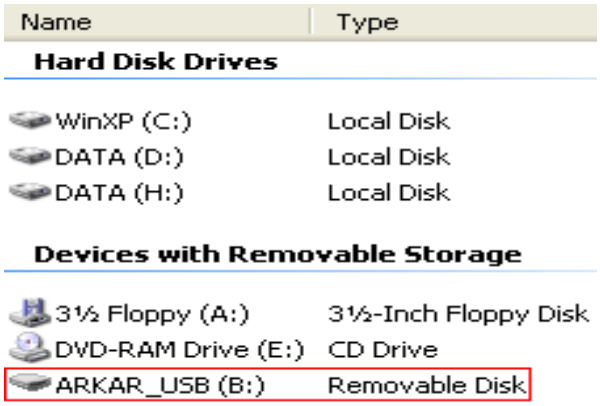


Figure A12.1: The evidence drive is detected on the forensic workstation

4. Opened the EnCase software (v. 6.16.1.4) running on the FW and created a new case as shown in the figure below.

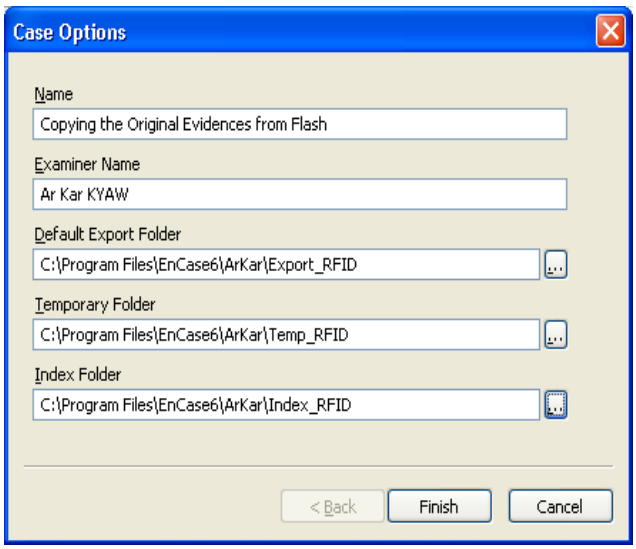


Figure A12.2: Creating a new case

5. Added the original evidence drive to the case by clicking “Add Device” icon and chose the “Local Devices (see Figure A12.3)” before clicking on “Next” button.

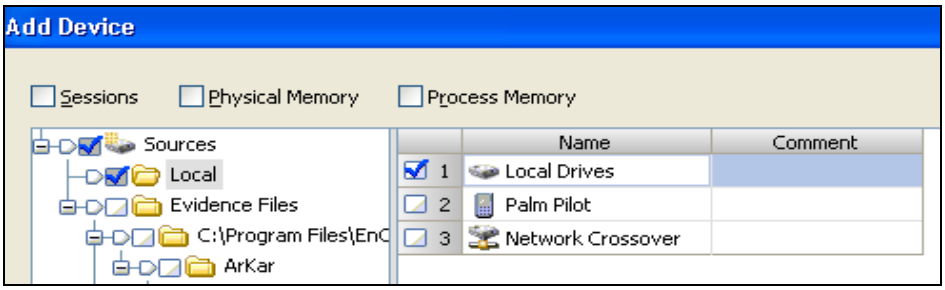


Figure A12.3: Adding the device on EnCase

6. According to the Figure A12.4, the original evidence drive B (ARKAR_USB) was detected as a write blocked drive and click the “Next” button.

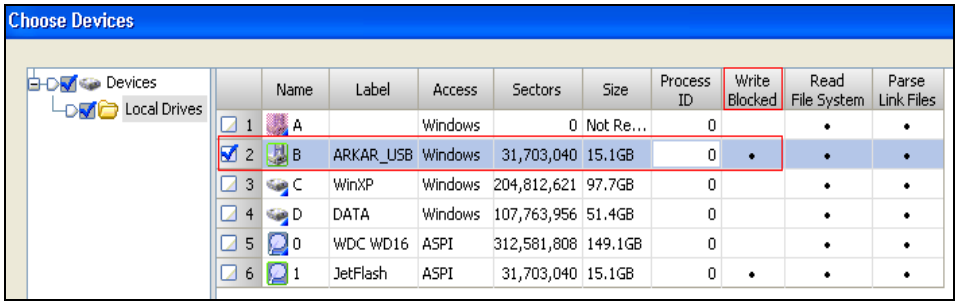


Figure A12.4: Screenshot of the chosen original evidence flash drive B

7. Then, EnCase added the device to the case as a preview after clicking “Finish” button.

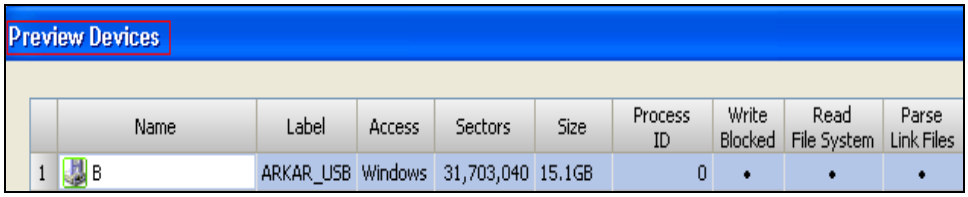


Figure A12.5: Adding the device to the case created in step 4

8. After previewing the original evidence drive, the acquisition of all the original evidence files from flash drive was performed in the next step by highlighting, right-clicking on the previewed drive B and clicking on “**Acquire**” button (as shown in the Figure A12.6).

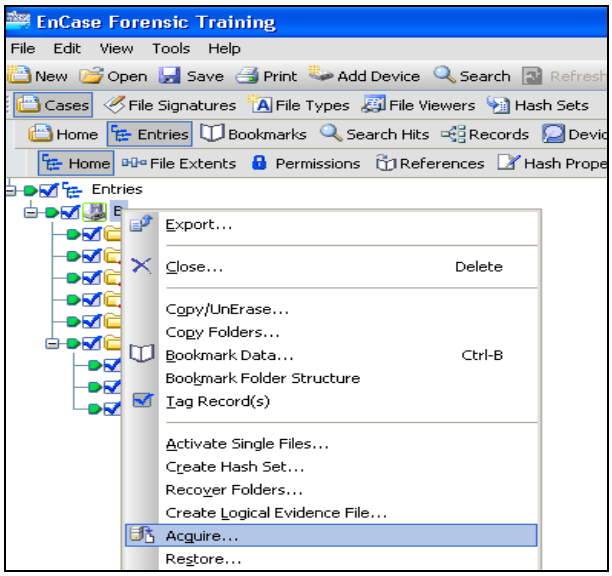


Figure A12.6: Acquiring the forensic image from preview

9. As soon as "**Acquire**" was selected, the options such as "**Do not add**", "**Add to Case**", "**Replace source drive**" and the like were shown in the "After Acquisition" for new image file. Chose the default setting as shown in Figure A12.7.

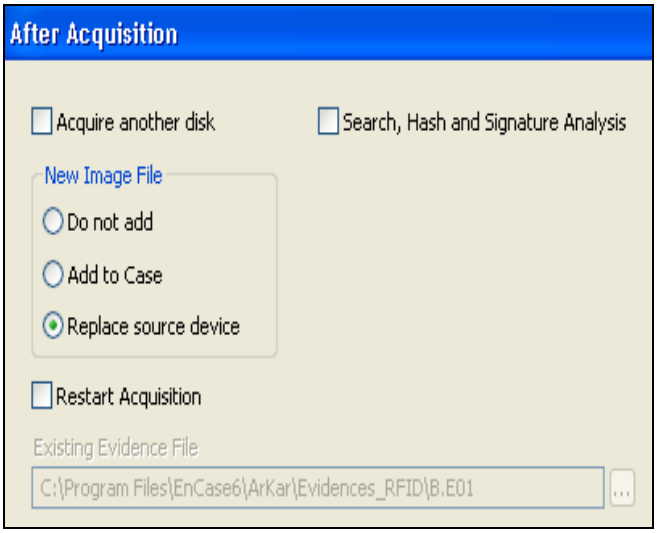


Figure A12.7: Acquiring the forensic image from preview

10. Then, click "**Next**" button and the following options for acquisition were chosen (see Figure A12.8).

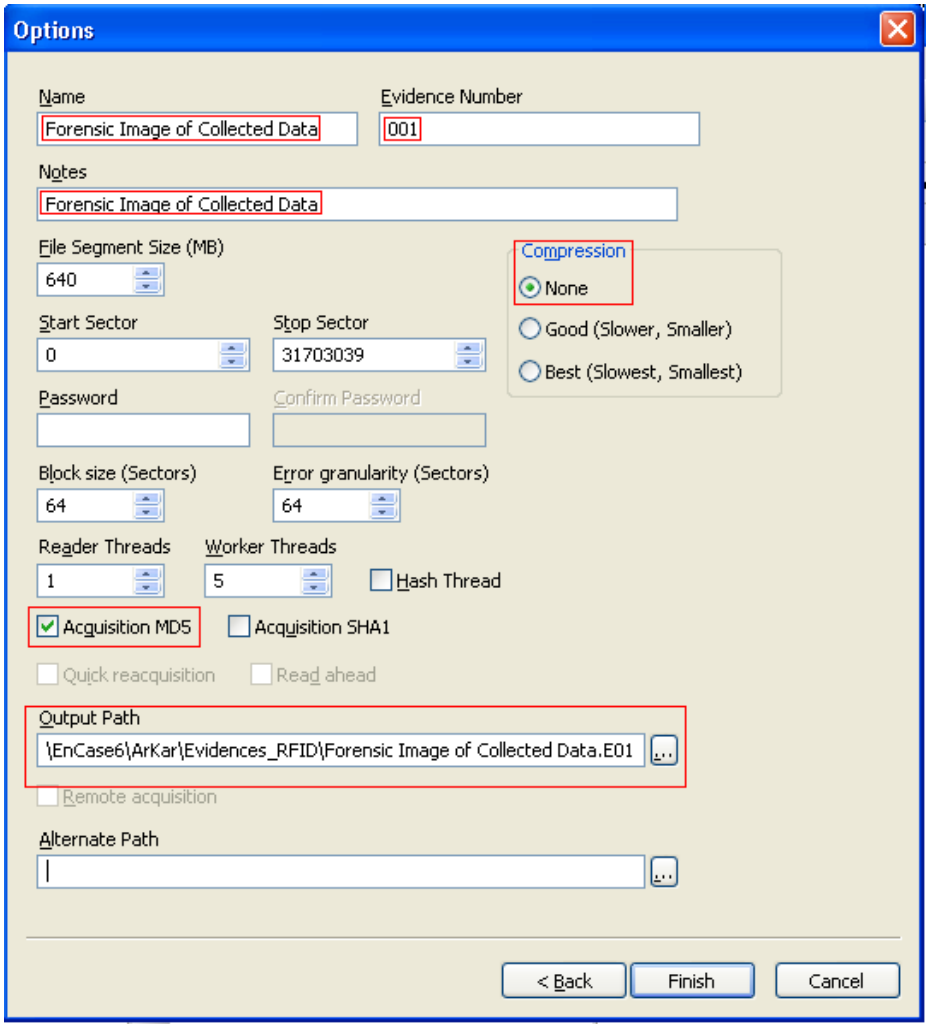


Figure A12.8: Acquisition setting

11. Once "**Finish**" button was clicked, the forensic software EnCase started acquiring the original collected data from drive B to the output folder (C:\Program Files\EnCase6\ArKar\Evidences_RFID).

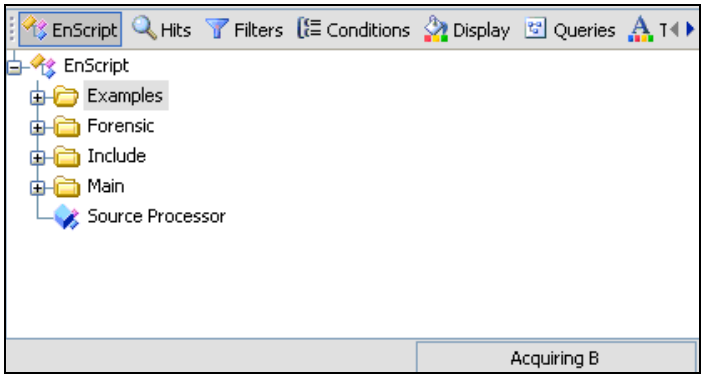


Figure A12.9: Screenshot of the acquisition process by EnCase

12. However, the acquisition was cancelled due to the fact that the investigator thought the acquisition by using the default compression method could save the time to acquire the evidence from the flash drive B. But, there was not affect on the collected evidences even though the data compression method was used.

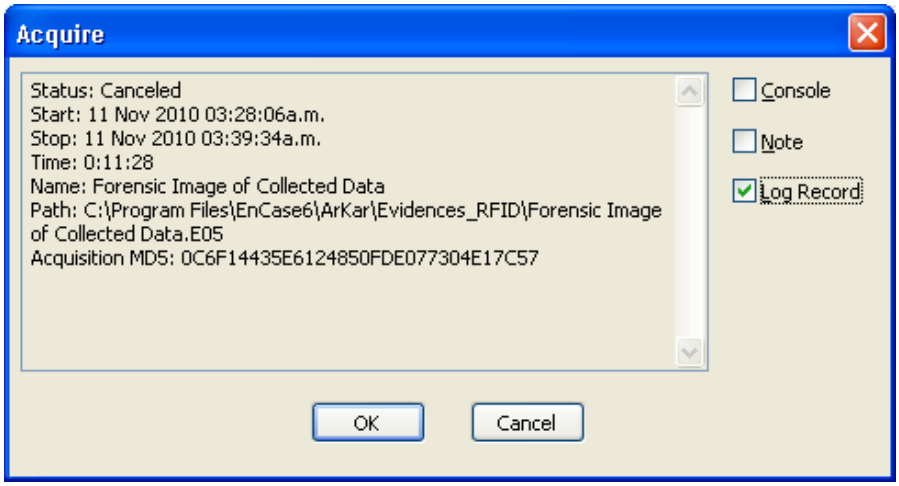


Figure A12.10: Screenshot of the cancellation of acquisition process

13. Created a new case and performed the same steps (1 to 9) mentioned above and chose the options (see the Figure A12.11) for acquisition.

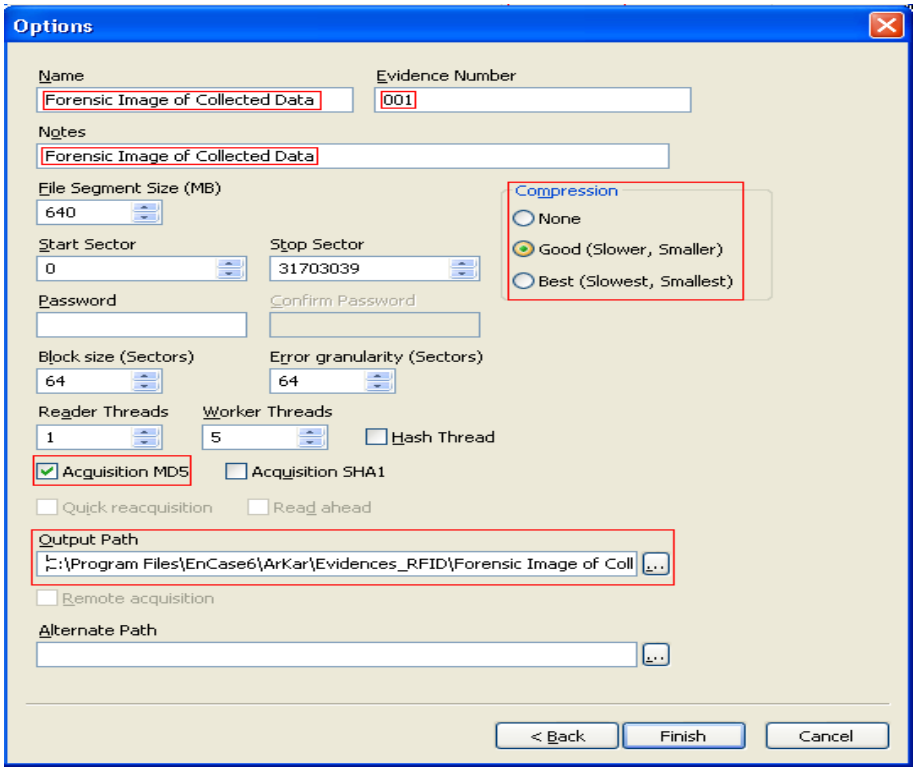


Figure A12.11: Screenshot of the new acquisition process using data compression level

14. Similar to the step (11), once "**Finish**" button was clicked; the forensic software EnCase started acquiring the original collected data from the flash drive B to the output folder (C:\Program Files\EnCase6\ArKar\Evidences_RFID).

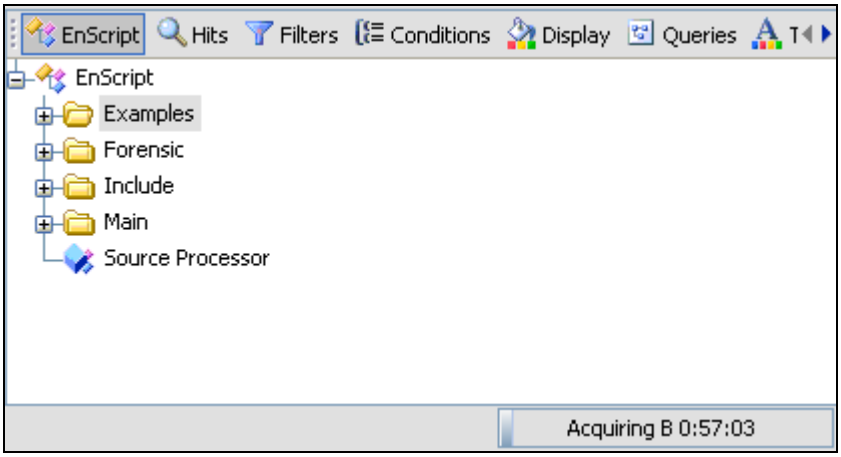


Figure 12.12: Screenshot of the acquisition process by EnCase

Astonishingly, the time taken to acquire the collected evidences from the flash drive by using the data compression method ("**Good**") was almost the same as that of the method used without compression.

The forensic image acquisition was completed. The MD5 hash value of the acquired data, the time taken to acquire the original evidence from the flash drive, and the location of the image copy can be found in the completion result (Figure A12.13).

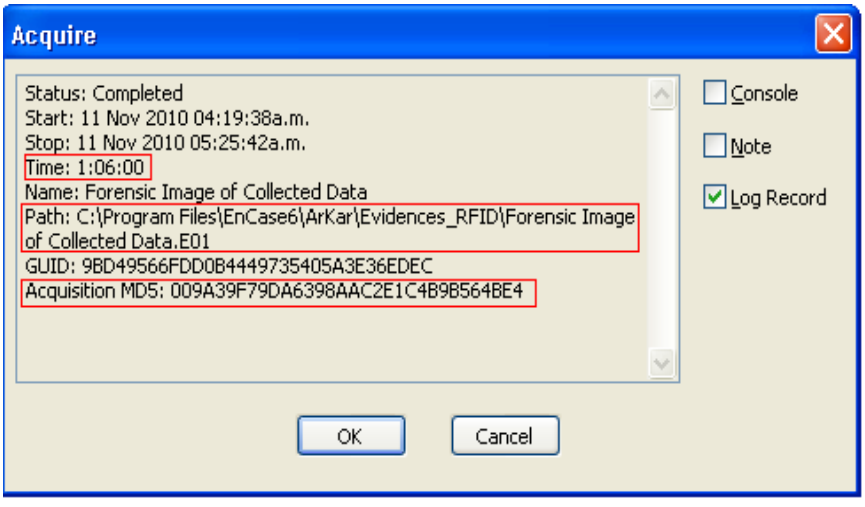


Figure A12.13: Screenshot of the Completed Acquisition

15. Then, the original evidence drive (B:\) was removed from the forensic workstation for preservation and the forensic analysis would be done by using the image copy.
16. The very first step performed during analysis of image copy was verifying the file signatures and hashing all the files within the case by using "**Search**" function on the tool bar of the EnCase (see Figure A12.14).

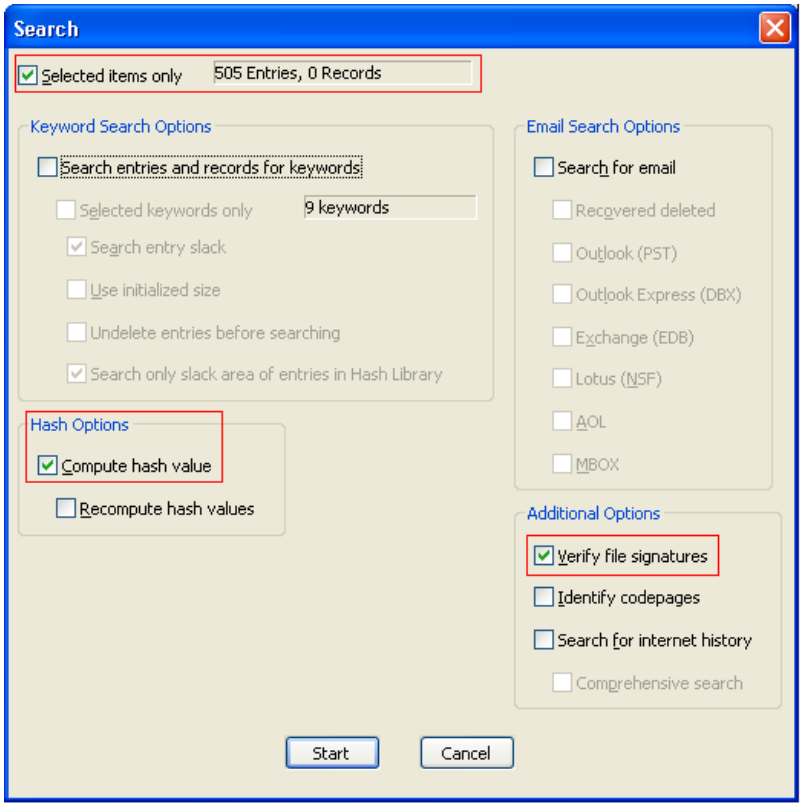


Figure A12.14: Screenshot of the "Search" options

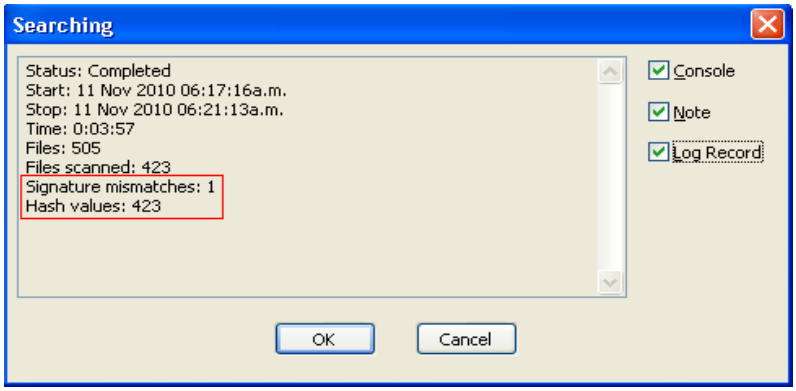


Figure A12.15: Screenshot of the Searching Completion

17. Then, the investigator performed copying the entire files that could be seen in the following figures.

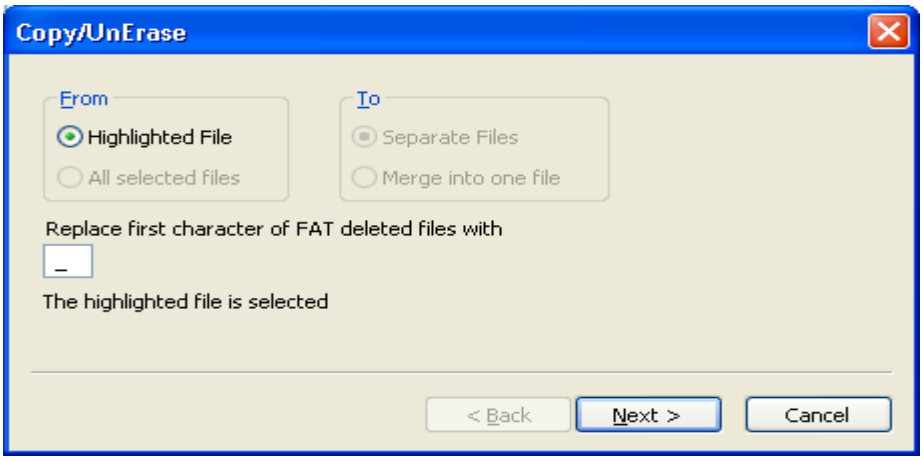


Figure A12.16: Step 1_ Entire File Copying

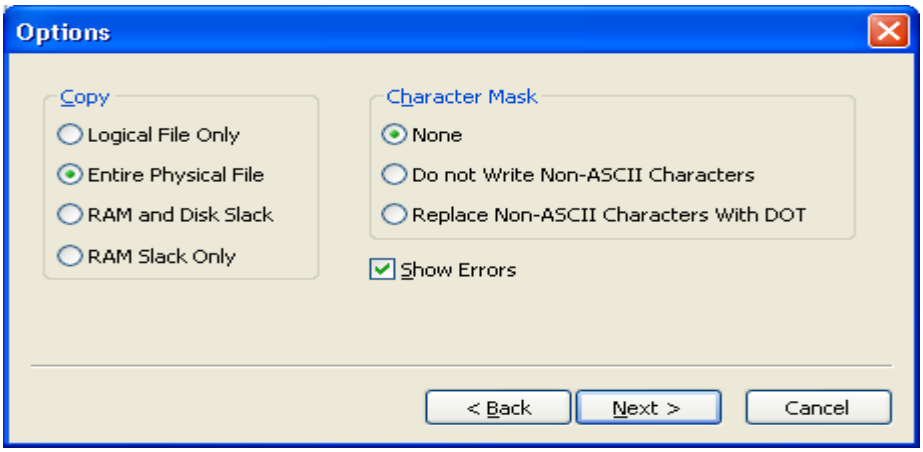


Figure A12.17: Step 2_ Entire File Copying

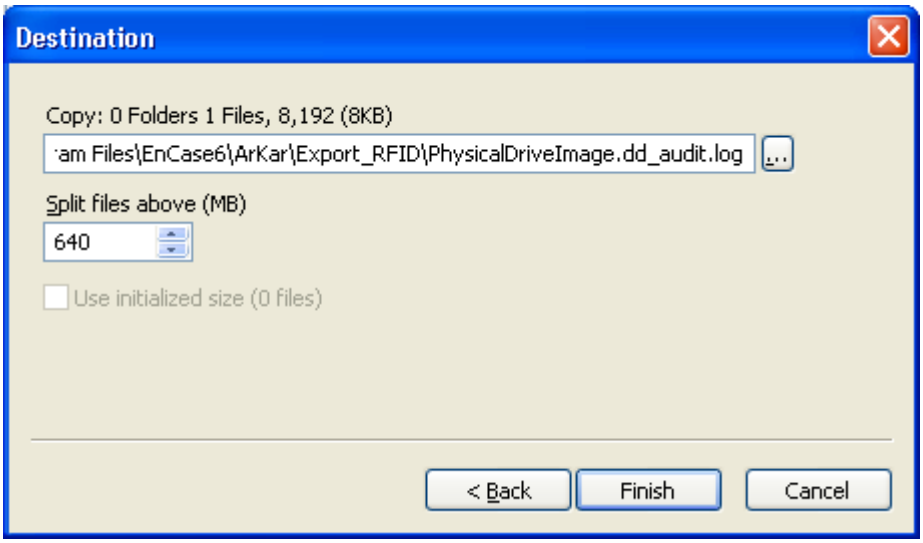


Figure A12.18: Step 3_ Entire File Copying

18. Then, a bad signature file was copies for further analysis if needed.

19. The hash set creation was performed on all the evidence files (see the figures below).

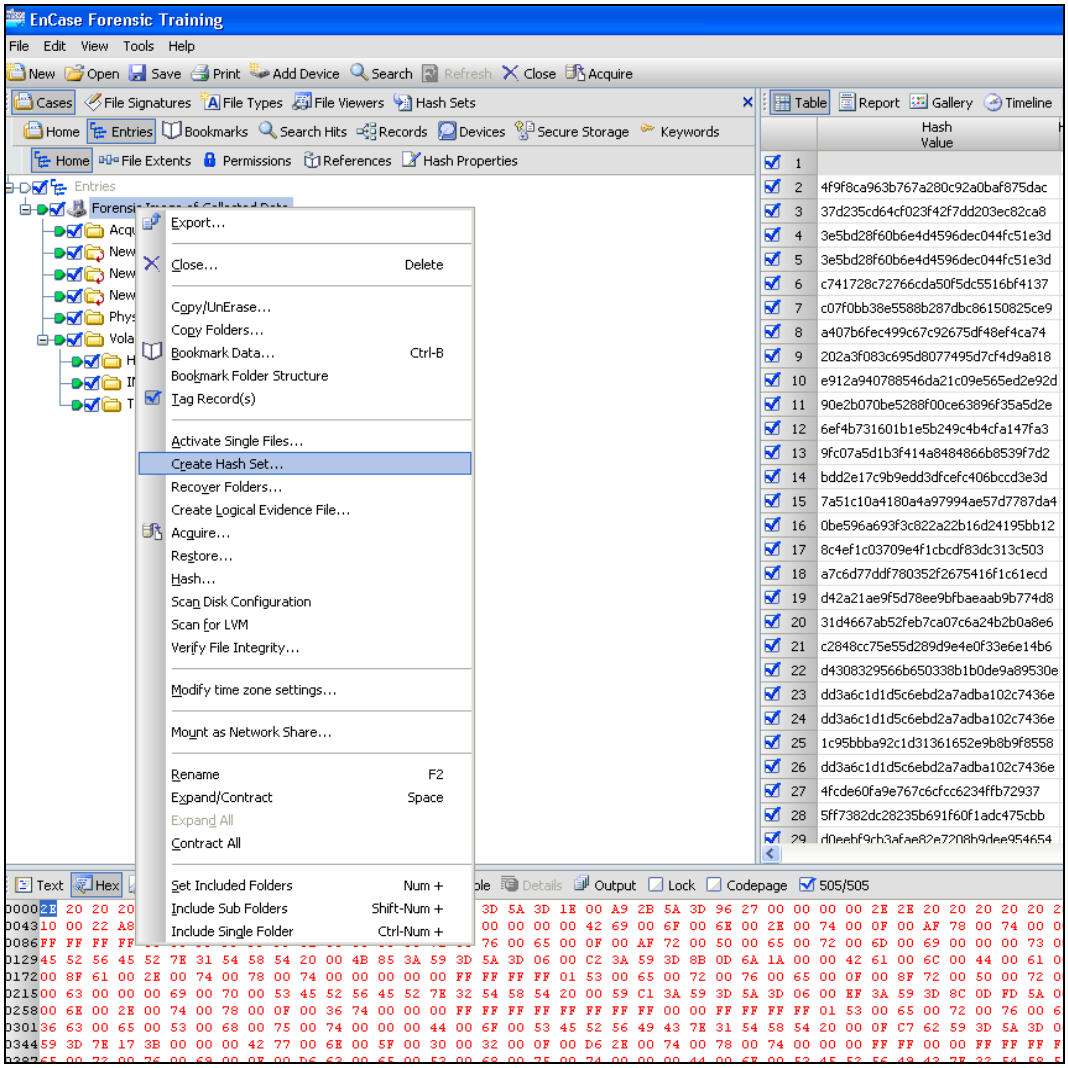


Figure A12.19: Hash Set Creation on all the acquired files

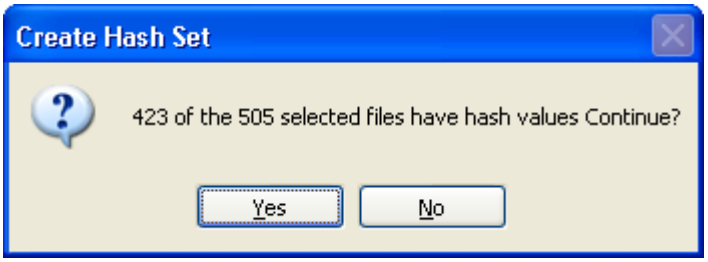


Figure A12.20: Hash Set Creation was completed

20. All the hash values of evidence files were exported as a file (HashValues of the Acquired Evidences.csv) on the forensic workstation before analysis for the purposes of the preservation and integrity checking needed to perform later (see the figures below).

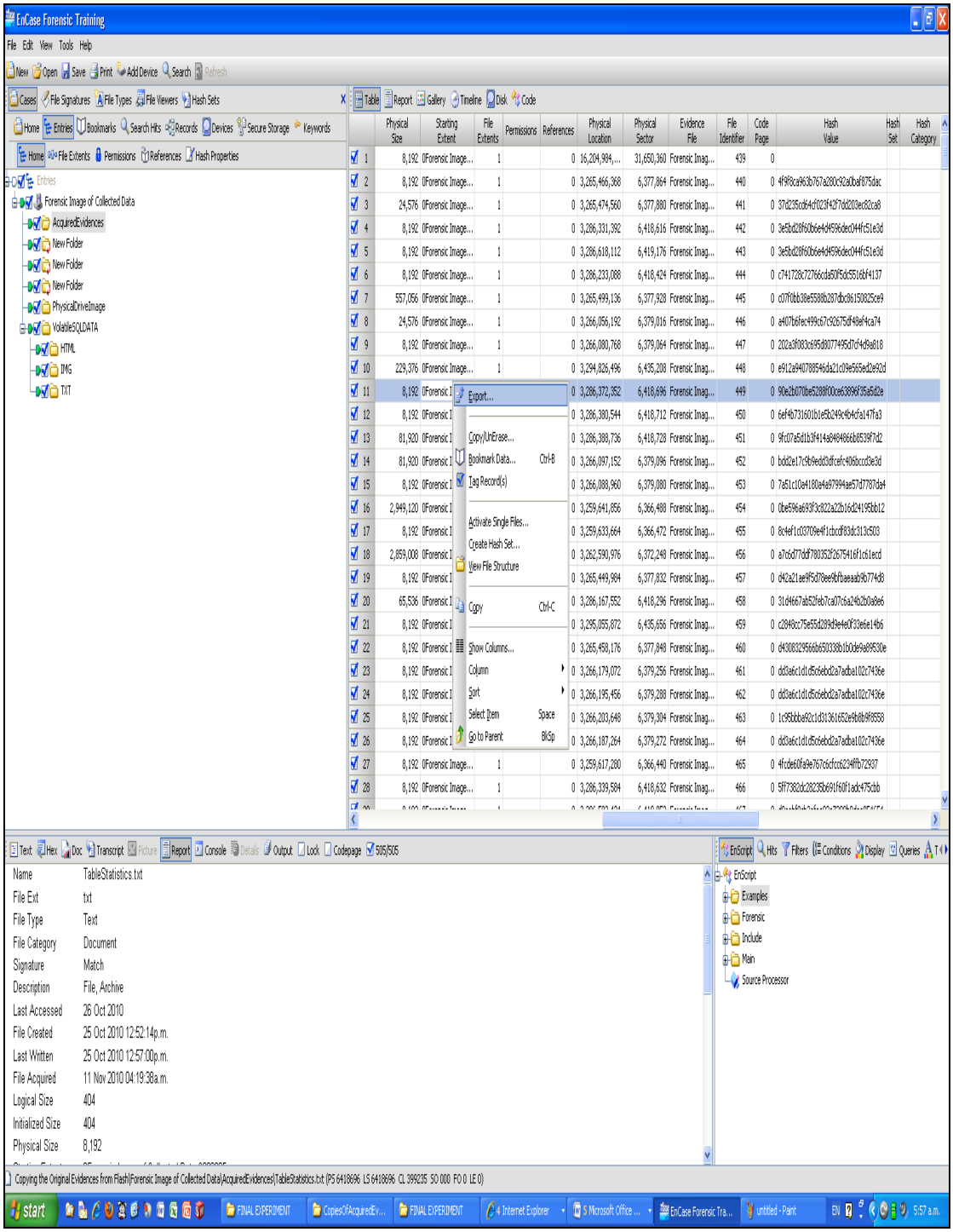


Figure A12.21: Export all the hashes for preservation

Appendix 13: Artefacts Collected by Extended WFT from HELIX_RFID_IR Tool

name	object_id	schema_id	create_date	modify_date	is_ms_shipped	is_auto_executed
(0 rows affected)						
(0 rows affected)						
(0 rows affected)						
(0 rows affected)						
(0 rows affected)						
User: arkar	Script: SSF	SPID: 53	Closed on Oct 25 2010 6:32AM			
(0 rows affected)						

Figure A13.1: Information of stored procedures set to auto-execute on the backend SQL Server

cache_address	name	type	clock_hand	clock_status	rounds_count	moved_all_rounds_co	ated_last_round_co	moved_last_round_cou	last_tick_time	round_start_time	last_round_start_time
0x038E4380	EventNotificationCache	CACHESTORE_EVENTS	HAND_EXTERNAL	SUSPENDED	6904	0	0	0	819299554938	819299554938	180288
0x038E4380	EventNotificationCache	CACHESTORE_EVENTS	HAND_INTERNAL	SUSPENDED	0	0	0	0	0	0	0
0x03909738	Object Plans	CACHESTORE_OBJCP	HAND_EXTERNAL	SUSPENDED	6904	0	0	0	819299555586	819299555586	180288
0x03909738	Object Plans	CACHESTORE_OBJCP	HAND_INTERNAL	SUSPENDED	0	0	0	0	0	0	0
0x03909AE8	SQL Plans	CACHESTORE_SQLCP	HAND_EXTERNAL	SUSPENDED	6904	1	0	0	819299556261	819299556261	180288
0x03909AE8	SQL Plans	CACHESTORE_SQLCP	HAND_INTERNAL	SUSPENDED	0	0	0	0	0	0	0
0x03932048	Bound Trees	CACHESTORE_PHDR	HAND_EXTERNAL	SUSPENDED	6904	0	0	0	819299556936	819299556936	180306
0x03932048	Bound Trees	CACHESTORE_PHDR	HAND_INTERNAL	SUSPENDED	0	0	0	0	0	0	0
0x039323F8	Extended Stored Proced	CACHESTORE_XPROC	HAND_EXTERNAL	SUSPENDED	6904	0	0	0	819299557620	819299557620	180333
0x039323F8	Extended Stored Proced	CACHESTORE_XPROC	HAND_INTERNAL	SUSPENDED	0	0	0	0	0	0	0
0x039331A0	Temporary Tables & Tabl	CACHESTORE_TEMPTABLES	HAND_EXTERNAL	SUSPENDED	6904	0	0	0	819299558295	819299558295	180378
0x039331A0	Temporary Tables & Tabl	CACHESTORE_TEMPTABLES	HAND_INTERNAL	SUSPENDED	0	0	0	0	0	0	0
0x03BC57A8	msdb	USERSTORE_DBMETADATA	HAND_EXTERNAL	SUSPENDED	6904	14	0	0	819299512782	819299512782	180792
0x03BC57A8	msdb	USERSTORE_DBMETADATA	HAND_INTERNAL	SUSPENDED	0	0	0	0	0	0	0
0x03BC5AE0	ObjPerm - msdb	USERSTORE_OBJPERM	HAND_EXTERNAL	SUSPENDED	6904	0	0	0	819299515860	819299515860	180693
0x03BC5AE0	ObjPerm - msdb	USERSTORE_OBJPERM	HAND_INTERNAL	SUSPENDED	0	0	0	0	0	0	0
0x03B897A8	tempdb	USERSTORE_DBMETADATA	HAND_EXTERNAL	SUSPENDED	6904	14	0	0	819299517174	819299517174	180756
0x03B897A8	tempdb	USERSTORE_DBMETADATA	HAND_INTERNAL	SUSPENDED	0	0	0	0	0	0	0
0x03B89AE0	ObjPerm - tempdb	USERSTORE_OBJPERM	HAND_EXTERNAL	SUSPENDED	6904	0	0	0	819299518083	819299518083	180693
0x03B89AE0	ObjPerm - tempdb	USERSTORE_OBJPERM	HAND_INTERNAL	SUSPENDED	0	0	0	0	0	0	0
0x03D337A8	RFID_test	USERSTORE_DBMETADATA	HAND_EXTERNAL	SUSPENDED	0	0	0	0	0	0	0
0x03D337A8	RFID_test	USERSTORE_DBMETADATA	HAND_INTERNAL	SUSPENDED	0	0	0	0	0	0	0
0x03D33AE0	ObjPerm - RFID_test	USERSTORE_OBJPERM	HAND_EXTERNAL	SUSPENDED	0	0	0	0	0	0	0
0x03D33AE0	ObjPerm - RFID_test	USERSTORE_OBJPERM	HAND_INTERNAL	SUSPENDED	0	0	0	0	0	0	0
(90 rows affected)											
User: arkar	Script: SSFA_ClockHands	SPID: 53	Closed on Oct 25 2010 6:32AM								
(90 rows affected)											

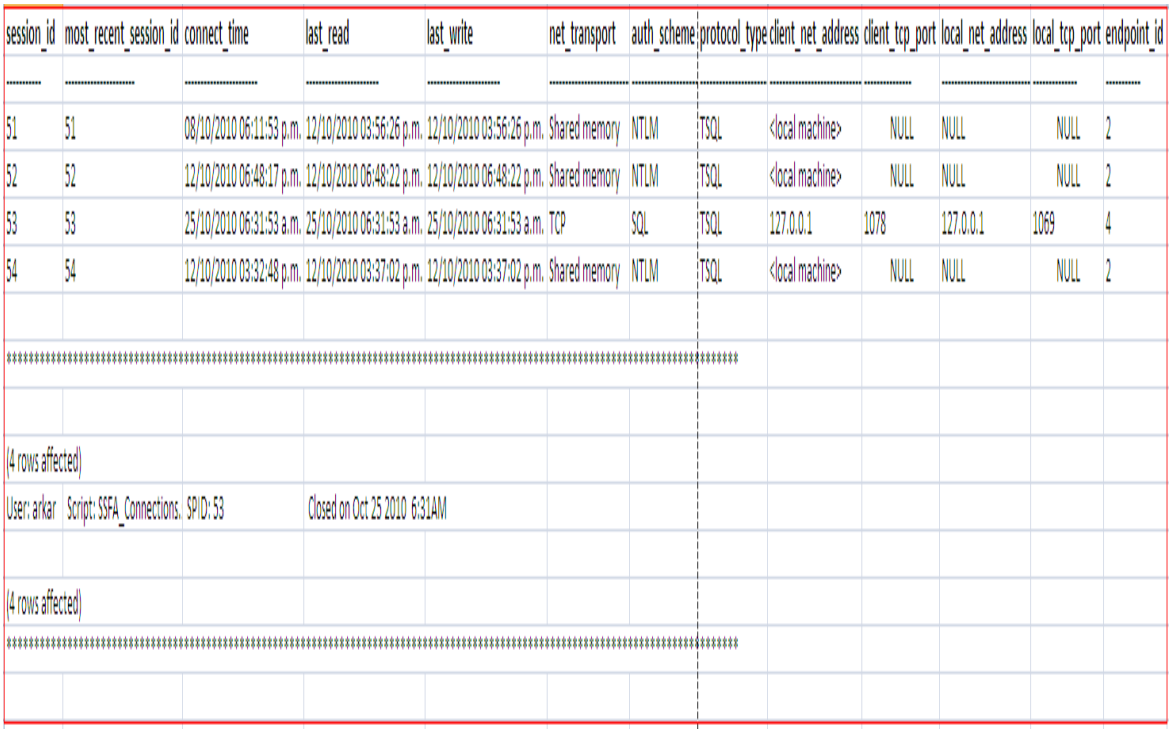
Figure A13.2: Fragment of Acquired Clock Hands information

database	name	create_date	modify_date	permission_set_desc	file	content
--------	------------------------------	----------------	----------------------------	------------------------------	----------	--------------
(0 rows affected)						
database	name	create_date	modify_date	permission_set_desc	file	content
--------	------------------------------	----------------	----------------------------	------------------------------	----------	--------------
(0 rows affected)						
database	name	create_date	modify_date	permission_set_desc	file	content
--------	------------------------------	----------------	----------------------------	------------------------------	----------	--------------
(0 rows affected)						
database	name	create_date	modify_date	permission_set_desc	file	content
--------	------------------------------	----------------	----------------------------	------------------------------	----------	--------------
(0 rows affected)						
database	name	create_date	modify_date	permission_set_desc	file	content
---------	------------------------------	----------------	----------------------------	------------------------------	----------	--------------
(0 rows affected)						
==========						
(0 rows affected)						
User: arkar	Script: SSFA_CLR.sql	SPID: 53	Closed on Oct 25 2010 6:32AM			
(0 rows affected)						
==========						

Figure A13.3: Acquired CLR information

configuration_id	name	value	minimum	maximum	value_in_use	description	is_dynamic	is_advanced
----------------	------------------------------	----------	----------	----------	----------	--	-------	----------
101	recovery interval (min)	0	0	32767	0	Maximum recovery interval in minutes	1	1
102	allow updates	0	0	1	0	Allow updates to system tables	1	0
103	user connections	0	0	32767	0	Number of user connections allowed	0	1
106	locks	0	5000	2147483647	0	Number of locks for all users	0	1
107	open objects	0	0	2147483647	0	Number of open database objects	0	1
109	fill factor (%)	0	0	100	0	Default fill factor percentage	0	1
114	disallow results from triggers	0	0	1	0	Disallow returning results from triggers	1	1
115	nested triggers	1	0	1	1	Allow triggers to be invoked within triggers	1	0
116	server trigger recursion	1	0	1	1	Allow recursion for server level triggers	1	0
117	remote access	1	0	1	1	Allow remote access	0	0
124	default language	0	0	9999	0	default language	1	0
1505	index create memory (KB)	0	704	2147483647	0	Memory for index create sorts (kBytes)	1	1
1519	remote login timeout (s)	20	0	2147483647	20	remote login timeout	1	0
1520	remote query timeout (s)	600	0	2147483647	600	remote query timeout	1	0
1543	min server memory (MB)	0	0	2147483647	8	Minimum size of server memory (MB)	1	1
1544	max server memory (MB)	2147483647	16	2147483647	2147483647	Maximum size of server memory (MB)	1	1
1545	query governor cost limit	0	0	2147483647	0	Maximum estimated cost allowed by query governor	1	1
1547	scan for startup procs	0	0	1	0	scan for startup stored procedures	0	1
1568	default trace enabled	1	0	1	1	Enable or disable the default trace	1	1
1570	in-doubt xact resolution	0	0	2	0	Recovery policy for DTC transactions with unknown outcome	1	1
1575	user instances enabled	0	0	1	0	Enable or disable creation of user instances	1	0
1576	remote admin connections	0	0	1	0	Dedicated Admin Connections are allowed from remote clients	1	0
16384	Agent XPs	0	0	1	0	Enable or disable Agent XPs	1	1
16385	SQL Mail XPs	0	0	1	0	Enable or disable SQL Mail XPs	1	1
16386	Database Mail XPs	0	0	1	0	Enable or disable Database Mail XPs	1	1
16387	SMO and DMO XPs	1	0	1	1	Enable or disable SMO and DMO XPs	1	1
16388	Ole Automation Procedures	0	0	1	0	Enable or disable Ole Automation Procedures	1	1
16389	Web Assistant Procedures	0	0	1	0	Enable or disable Web Assistant Procedures	1	1
16390	xp_cmdshell	0	0	1	0	Enable or disable command shell	1	1
16391	Ad Hoc Distributed Queries	0	0	1	0	Enable or disable Ad Hoc Distributed Queries	1	1
16392	Replication XPs	0	0	1	0	Enable or disable Replication XPs	1	1
==========								
(64 rows affected)								
User: arkar	Script: SSFA_Configurations.sql	SPID: 53	Closed on Oct 25 2010 6:32AM					

Figure A13.4: Fragment of Acquired backend SQL Server configuration information



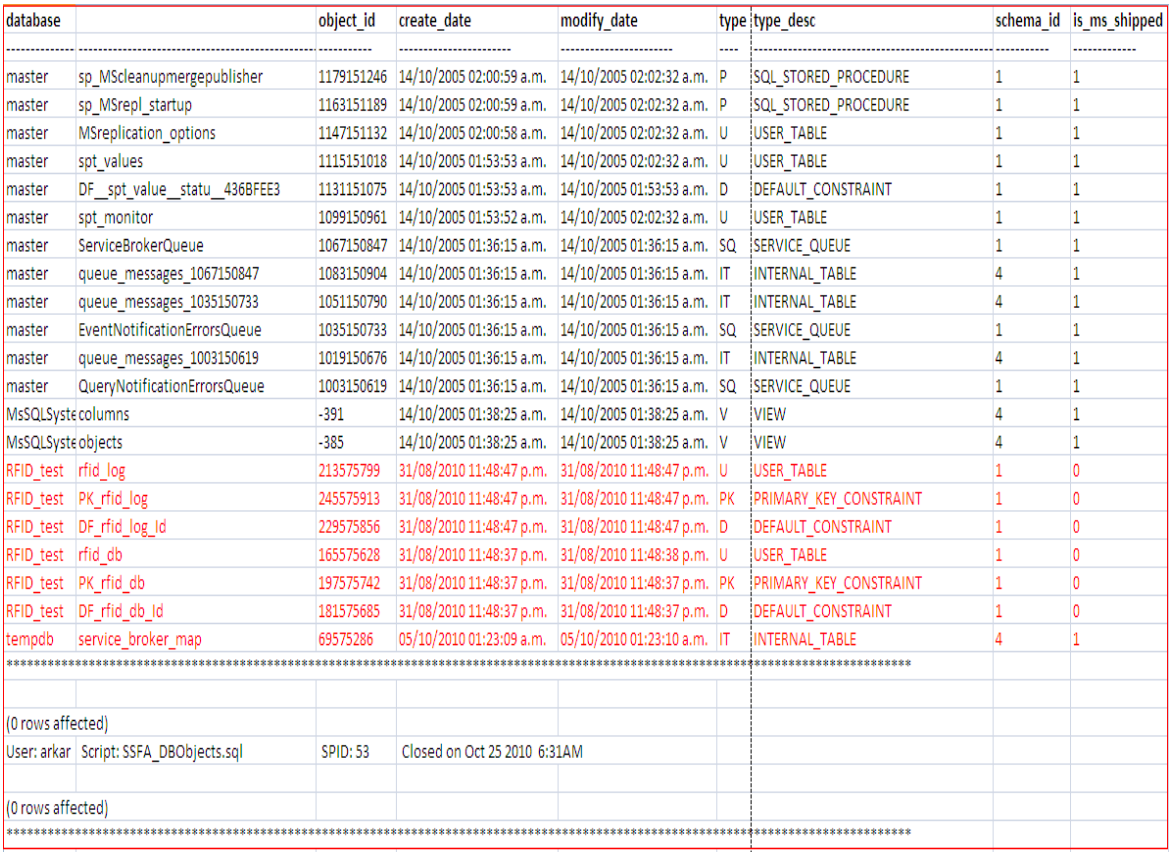
session_id	most_recent_session_id	connect_time	last_read	last_write	net_transport	auth_scheme	protocol_type	client_net_address	client_tcp_port	local_net_address	local_tcp_port	endpoint_id
51	51	08/10/2010 06:11:53 p.m.	12/10/2010 03:56:26 p.m.	12/10/2010 03:56:26 p.m.	Shared memory	NTLM	TSQL	<local machine>	NULL	NULL	NULL	2
52	52	12/10/2010 06:48:17 p.m.	12/10/2010 06:48:22 p.m.	12/10/2010 06:48:22 p.m.	Shared memory	NTLM	TSQL	<local machine>	NULL	NULL	NULL	2
53	53	25/10/2010 06:31:53 a.m.	25/10/2010 06:31:53 a.m.	25/10/2010 06:31:53 a.m.	TCP	SQL	TSQL	127.0.0.1	1078	127.0.0.1	1069	4
54	54	12/10/2010 03:32:48 p.m.	12/10/2010 03:37:02 p.m.	12/10/2010 03:37:02 p.m.	Shared memory	NTLM	TSQL	<local machine>	NULL	NULL	NULL	2

(4 rows affected)

User: arkar Script: SSFA_Connections. SPID: 53 Closed on Oct 25 2010 6:31AM

(4 rows affected)

Figure A13.5: Acquired backend SQL Server connection information



database		object_id	create_date	modify_date	type	type_desc	schema_id	is_ms_shipped
master	sp_MScleanupmergepublisher	1179151246	14/10/2005 02:00:59 a.m.	14/10/2005 02:02:32 a.m.	P	SQL_STORED_PROCEDURE	1	1
master	sp_MSrepl_startup	1163151189	14/10/2005 02:00:59 a.m.	14/10/2005 02:02:32 a.m.	P	SQL_STORED_PROCEDURE	1	1
master	MSreplication_options	1147151132	14/10/2005 02:00:58 a.m.	14/10/2005 02:02:32 a.m.	U	USER_TABLE	1	1
master	spt_values	1115151018	14/10/2005 01:53:53 a.m.	14/10/2005 02:02:32 a.m.	U	USER_TABLE	1	1
master	DF__spt_value__statu__436BFEE3	1131151075	14/10/2005 01:53:53 a.m.	14/10/2005 01:53:53 a.m.	D	DEFAULT_CONSTRAINT	1	1
master	spt_monitor	1099150961	14/10/2005 01:53:52 a.m.	14/10/2005 02:02:32 a.m.	U	USER_TABLE	1	1
master	ServiceBrokerQueue	1067150847	14/10/2005 01:36:15 a.m.	14/10/2005 01:36:15 a.m.	SQ	SERVICE_QUEUE	1	1
master	queue_messages_1067150847	1083150904	14/10/2005 01:36:15 a.m.	14/10/2005 01:36:15 a.m.	IT	INTERNAL_TABLE	4	1
master	queue_messages_1035150733	1051150790	14/10/2005 01:36:15 a.m.	14/10/2005 01:36:15 a.m.	IT	INTERNAL_TABLE	4	1
master	EventNotificationErrorsQueue	1035150733	14/10/2005 01:36:15 a.m.	14/10/2005 01:36:15 a.m.	SQ	SERVICE_QUEUE	1	1
master	queue_messages_1003150619	1019150676	14/10/2005 01:36:15 a.m.	14/10/2005 01:36:15 a.m.	IT	INTERNAL_TABLE	4	1
master	QueryNotificationErrorsQueue	1003150619	14/10/2005 01:36:15 a.m.	14/10/2005 01:36:15 a.m.	SQ	SERVICE_QUEUE	1	1
MsSQLSyste	columns	-391	14/10/2005 01:38:25 a.m.	14/10/2005 01:38:25 a.m.	V	VIEW	4	1
MsSQLSyste	objects	-385	14/10/2005 01:38:25 a.m.	14/10/2005 01:38:25 a.m.	V	VIEW	4	1
RFID_test	rfid_log	213575799	31/08/2010 11:48:47 p.m.	31/08/2010 11:48:47 p.m.	U	USER_TABLE	1	0
RFID_test	PK_rfid_log	245575913	31/08/2010 11:48:47 p.m.	31/08/2010 11:48:47 p.m.	PK	PRIMARY_KEY_CONSTRAINT	1	0
RFID_test	DF_rfid_log_Id	229575856	31/08/2010 11:48:47 p.m.	31/08/2010 11:48:47 p.m.	D	DEFAULT_CONSTRAINT	1	0
RFID_test	rfid_db	165575628	31/08/2010 11:48:37 p.m.	31/08/2010 11:48:38 p.m.	U	USER_TABLE	1	0
RFID_test	PK_rfid_db	197575742	31/08/2010 11:48:37 p.m.	31/08/2010 11:48:37 p.m.	PK	PRIMARY_KEY_CONSTRAINT	1	0
RFID_test	DF_rfid_db_Id	181575685	31/08/2010 11:48:37 p.m.	31/08/2010 11:48:37 p.m.	D	DEFAULT_CONSTRAINT	1	0
tempdb	service_broker_map	69575286	05/10/2010 01:23:09 a.m.	05/10/2010 01:23:10 a.m.	IT	INTERNAL_TABLE	4	1

(0 rows affected)

User: arkar Script: SSFA_DBObjects.sql SPID: 53 Closed on Oct 25 2010 6:31AM

(0 rows affected)

Figure A13.6: Fragment of Acquired backend SQL Server Database Object information

database_id	file_id	database	type_desc	file_name	physical_name
1	1	master	ROWS	master	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\master.mdf
1	2	master	LOG	mastlog	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\mastlog.ldf
2	1	tempdb	ROWS	tempdev	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\tempdb.mdf
2	2	tempdb	LOG	templog	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\templog.ldf
3	1	model	ROWS	modeldev	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\model.mdf
3	2	model	LOG	modellog	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\modellog.ldf
4	1	msdb	ROWS	MSDBData	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\MSDBData.mdf
4	2	msdb	LOG	MSDBLog	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\MSDBLog.ldf
5	1	RFID_test	ROWS	RFID_test	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\RFID_test.mdf
5	2	RFID_test	LOG	RFID_test_log	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\RFID_test_log.ldf

(10 rows affected)

User: arkar Script: SSFA_Dblisting.sql SPID: 53 Executed on Oct 25 2010 6:32AM

(10 rows affected)

Figure A13.7: Fragment of Acquired backend SQL Server Database files information

database_id	file_id	page_id	page_level	page_type	row_count	is_modified
1	1	66	0	DATA_PAGE	46	0
1	1	9	0	BOOT_PAGE	1	0
1	1	68	0	DATA_PAGE	103	0
2	1	70	1	INDEX_PAGE	5	0
2	1	66	1	INDEX_PAGE	5	0
2	1	16	0	DATA_PAGE	152	1
2	1	17	0	DATA_PAGE	80	1
2	1	19	0	DATA_PAGE	80	1
2	1	20	0	DATA_PAGE	91	1
3	1	144	0	DATA_PAGE	5	0
3	1	147	0	DATA_PAGE	15	0
3	1	23	0	DATA_PAGE	2	0
4	1	226	0	DATA_PAGE	2	0
4	1	214	0	DATA_PAGE	2	0
5	1	105	0	INDEX_PAGE	102	0
5	1	157	0	DATA_PAGE	19	1
5	1	161	0	IAM_PAGE	2	0
5	1	142	0	DATA_PAGE	8	0
5	1	75	0	DATA_PAGE	27	0
5	1	147	0	DATA_PAGE	14	0
5	1	47	0	TEXT_MIX_PAGE	1	0

8 rows affected)

8 rows affected)

User: arkar Script: SSFA_DataCache.sql SPID: 53 on Oct 25 2010 6:31AM

Figure A13.8: Fragment of Acquired backend SQL Server Data Cache Entries information

SQL SERVER - DATABASE SERVER INFORMATION

********** **********

Instance Name: TEST-STATION\SQLEXPRESS

Edition: Express Edition

Version: 9.00.1399.06

Service Pack: RTM

Process ID: 1508

Integrated Security Only: 0

Collation: SQL_Latin1_General_CP1_CI_AS

Windows Locale: 1033

Clusterd: 0

FullText Enabled: 0

Character Set: iso_1

Sort Order: nocase_iso

Resource DB Last Updated: Oct 14 2005 1:56AM

Resource DB Version: 9.00.1399

CLR Version: v2.0.50727

Figure A13.9: Acquired backend SQL Server information

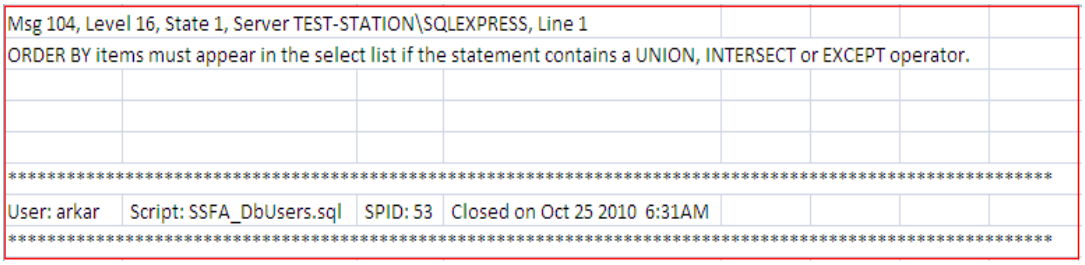


Figure A13.10: Acquired backend SQL Server Database Users information

Mon Oct 25 06:34:27 2010 -- WFT Run Complete

Figure A13.11: Completion of the Acquisition of backend SQL Server Artefacts by using WFT

protocol_desc	type_desc	state_desc	is_admin_endpoint	endpoint_id	principal_id
TCP	TSQL	STARTED	1	1	1
SHARED_MEMORY	TSQL	STARTED	0	2	1
NAMED_PIPES	TSQL	STARTED	0	3	1
TCP	TSQL	STARTED	0	4	1
VIA	TSQL	STARTED	0	5	1

(5 rows affected)

User: arkar Script: SSFA_EndPoints.sql SPID: 53 Closed on Oct 25 2010 6:32AM

(5 rows affected)

Figure A13.12: Acquired backend SQL Server Database Endpoints information

job_id	step_id	step_name	command	sql_message_id	message	run_status	run_date	run_time	run_duration

(0 rows affected)

(0 rows affected)

(0 rows affected)

(0 rows affected)

(0 rows affected)

User: arkar Script: SSFA_JobHistory.sql SPID: 53 Closed on Oct 25 2010 6:32AM

(0 rows affected)

Figure A13.13: Acquired backend SQL Server Job History information

job_name	job_id	step_id	step_name	command	enabled	description	start_step_id	owner_sid	date_created	date_modified

(0 rows affected)

(0 rows affected)

(0 rows affected)

(0 rows affected)

(0 rows affected)

User: arkar Script: SSFA_Jobs.sql SPID: 53 Closed on Oct 25 2010 6:32AM

(0 rows affected)

Figure A13.14: Acquired backend SQL Server Agent Jobs information

name	createdate	updatedate	accdate	status	sysadmin
vishal	03/10/2010 01:05:58 a.m.	03/10/2010 01:05:58 a.m.	03/10/2010 01:05:58 a.m.	1	0
arkar	01/10/2010 11:17:18 p.m.	01/10/2010 11:17:18 p.m.	01/10/2010 11:17:18 p.m.	1	1
sa	08/04/2003 09:10:35 a.m.	01/10/2010 10:37:40 p.m.	08/04/2003 09:10:35 a.m.	1	1
TEST-STATION\SQLServer2005MSSQLUser$TEST-STATION$	17/07/2010 12:17:56 a.m.	17/07/2010 12:17:56 a.m.	17/07/2010 12:17:56 a.m.	1	1
BUILTIN\Users	17/07/2010 12:17:56 a.m.	17/07/2010 12:17:56 a.m.	17/07/2010 12:17:56 a.m.	1	0
NT AUTHORITY\SYSTEM	17/07/2010 12:17:56 a.m.	17/07/2010 12:17:56 a.m.	17/07/2010 12:17:56 a.m.	1	1
BUILTIN\Administrators	17/07/2010 12:17:56 a.m.	17/07/2010 12:17:56 a.m.	17/07/2010 12:17:56 a.m.	1	1
##MS_AgentSigningCertificate##	14/10/2005 01:56:19 a.m.	14/10/2005 01:56:19 a.m.	14/10/2005 01:56:19 a.m.	1	0
##MS_SQLReplicationSigningCertificate##	14/10/2005 01:52:54 a.m.	14/10/2005 01:52:54 a.m.	14/10/2005 01:52:54 a.m.	0	0
##MS_SQLAuthenticatorCertificate##	14/10/2005 01:52:54 a.m.	14/10/2005 01:52:54 a.m.	14/10/2005 01:52:54 a.m.	0	0
##MS_SQLResourceSigningCertificate##	14/10/2005 01:52:54 a.m.	14/10/2005 01:52:54 a.m.	14/10/2005 01:52:54 a.m.	0	0

(11 rows affected)

User: arkar Script: SSFA_Logins.sql SPID: 53 Closed on Oct 25 2010 6:31AM

(11 rows affected)

Figure A13.15: Fragment of the acquired backend SQL Server Logins information

(0 rows affected)

(0 rows affected)

(0 rows affected)

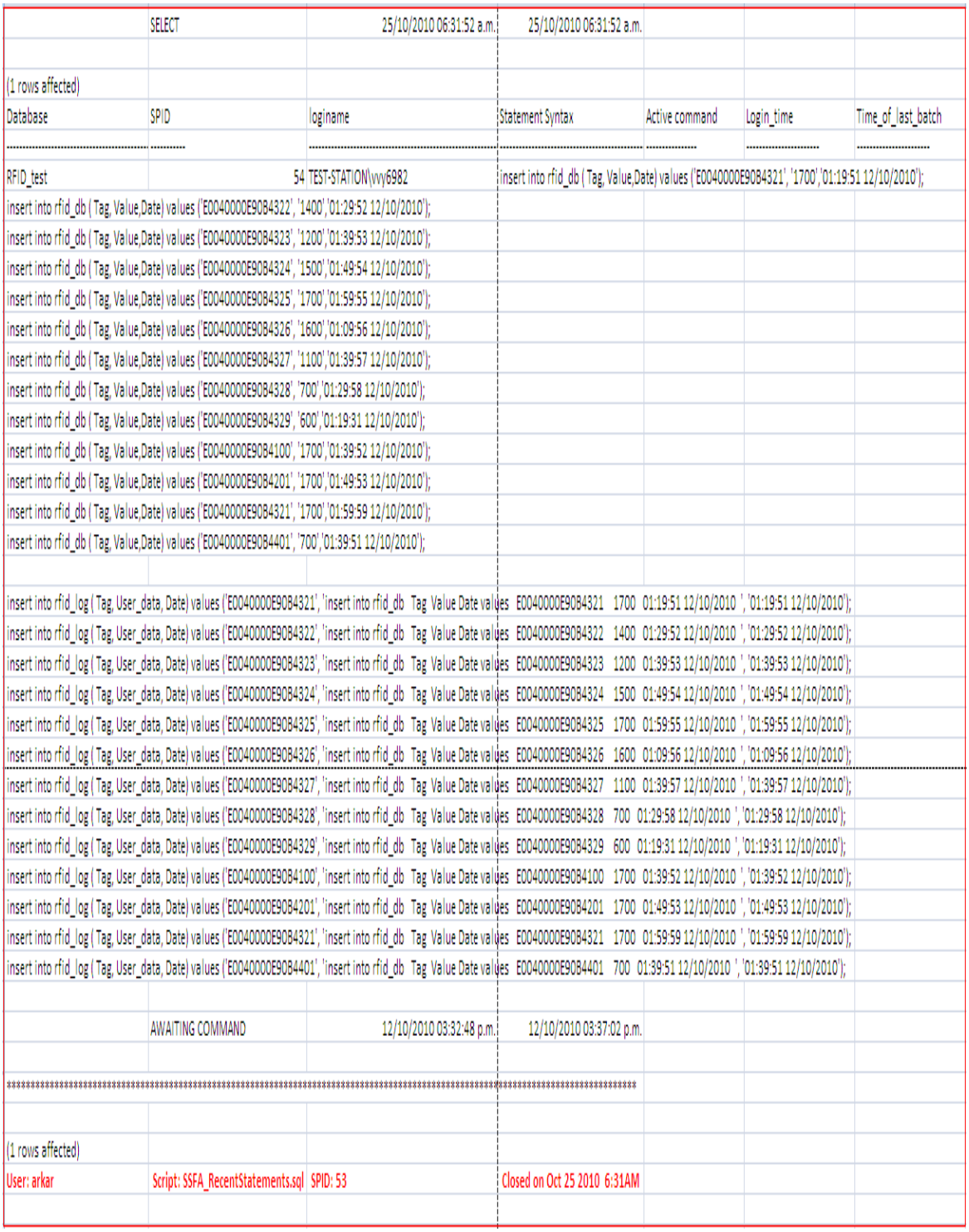
(0 rows affected)

(0 rows affected)

User: arkar Script: SSFA_PlanCache.sql SPID: 53 Closed on Oct 25 2010 6:31AM

(0 rows affected)

Figure A13.16: Result of the acquired backend SQL Server Plan Cache Entries information



	SELECT	25/10/2010 06:31:52 a.m.	25/10/2010 06:31:52 a.m.			
(1 rows affected)						
Database	SPID	loginame	Statement Syntax	Active command	Login_time	Time_of_last_batch
RFID_test	54	TEST-STATION\vvy6982	insert into rfid_db (Tag, Value,Date) values ('E0040000E90B4321', '1700','01:19:51 12/10/2010');			
insert into rfid_db (Tag, Value,Date) values ('E0040000E90B4322', '1400','01:29:52 12/10/2010');						
insert into rfid_db (Tag, Value,Date) values ('E0040000E90B4323', '1200','01:39:53 12/10/2010');						
insert into rfid_db (Tag, Value,Date) values ('E0040000E90B4324', '1500','01:49:54 12/10/2010');						
insert into rfid_db (Tag, Value,Date) values ('E0040000E90B4325', '1700','01:59:55 12/10/2010');						
insert into rfid_db (Tag, Value,Date) values ('E0040000E90B4326', '1600','01:09:56 12/10/2010');						
insert into rfid_db (Tag, Value,Date) values ('E0040000E90B4327', '1100','01:39:57 12/10/2010');						
insert into rfid_db (Tag, Value,Date) values ('E0040000E90B4328', '700','01:29:58 12/10/2010');						
insert into rfid_db (Tag, Value,Date) values ('E0040000E90B4329', '600','01:19:31 12/10/2010');						
insert into rfid_db (Tag, Value,Date) values ('E0040000E90B4100', '1700','01:39:52 12/10/2010');						
insert into rfid_db (Tag, Value,Date) values ('E0040000E90B4201', '1700','01:49:53 12/10/2010');						
insert into rfid_db (Tag, Value,Date) values ('E0040000E90B4321', '1700','01:59:59 12/10/2010');						
insert into rfid_db (Tag, Value,Date) values ('E0040000E90B4401', '700','01:39:51 12/10/2010');						
insert into rfid_log (Tag, User_data, Date) values ('E0040000E90B4321', 'insert into rfid_db Tag Value Date values E0040000E90B4321 1700 01:19:51 12/10/2010 ', '01:19:51 12/10/2010');						
insert into rfid_log (Tag, User_data, Date) values ('E0040000E90B4322', 'insert into rfid_db Tag Value Date values E0040000E90B4322 1400 01:29:52 12/10/2010 ', '01:29:52 12/10/2010');						
insert into rfid_log (Tag, User_data, Date) values ('E0040000E90B4323', 'insert into rfid_db Tag Value Date values E0040000E90B4323 1200 01:39:53 12/10/2010 ', '01:39:53 12/10/2010');						
insert into rfid_log (Tag, User_data, Date) values ('E0040000E90B4324', 'insert into rfid_db Tag Value Date values E0040000E90B4324 1500 01:49:54 12/10/2010 ', '01:49:54 12/10/2010');						
insert into rfid_log (Tag, User_data, Date) values ('E0040000E90B4325', 'insert into rfid_db Tag Value Date values E0040000E90B4325 1700 01:59:55 12/10/2010 ', '01:59:55 12/10/2010');						
insert into rfid_log (Tag, User_data, Date) values ('E0040000E90B4326', 'insert into rfid_db Tag Value Date values E0040000E90B4326 1600 01:09:56 12/10/2010 ', '01:09:56 12/10/2010');						
insert into rfid_log (Tag, User_data, Date) values ('E0040000E90B4327', 'insert into rfid_db Tag Value Date values E0040000E90B4327 1100 01:39:57 12/10/2010 ', '01:39:57 12/10/2010');						
insert into rfid_log (Tag, User_data, Date) values ('E0040000E90B4328', 'insert into rfid_db Tag Value Date values E0040000E90B4328 700 01:29:58 12/10/2010 ', '01:29:58 12/10/2010');						
insert into rfid_log (Tag, User_data, Date) values ('E0040000E90B4329', 'insert into rfid_db Tag Value Date values E0040000E90B4329 600 01:19:31 12/10/2010 ', '01:19:31 12/10/2010');						
insert into rfid_log (Tag, User_data, Date) values ('E0040000E90B4100', 'insert into rfid_db Tag Value Date values E0040000E90B4100 1700 01:39:52 12/10/2010 ', '01:39:52 12/10/2010');						
insert into rfid_log (Tag, User_data, Date) values ('E0040000E90B4201', 'insert into rfid_db Tag Value Date values E0040000E90B4201 1700 01:49:53 12/10/2010 ', '01:49:53 12/10/2010');						
insert into rfid_log (Tag, User_data, Date) values ('E0040000E90B4321', 'insert into rfid_db Tag Value Date values E0040000E90B4321 1700 01:59:59 12/10/2010 ', '01:59:59 12/10/2010');						
insert into rfid_log (Tag, User_data, Date) values ('E0040000E90B4401', 'insert into rfid_db Tag Value Date values E0040000E90B4401 700 01:39:51 12/10/2010 ', '01:39:51 12/10/2010');						
	AWAITING COMMAND	12/10/2010 03:32:48 p.m.	12/10/2010 03:37:02 p.m.			
**********						
(1 rows affected)						
User: arkar	Script: SSFA_RecentStatements.sql	SPID: 53	Closed on Oct 25 2010 6:31AM			

Figure A13.17: Result of the recently executed SQL statements information from victim's SQL Server

database	name	schema_id	principal_id		
---	---	---	---		
RFID_test	dbo	1	1		
RFID_test	guest	2	2		
RFID_test	INFORMATION_SCHEMA	3	3		
RFID_test	sys	4	4		
RFID_test	db_owner	16384	16384		
RFID_test	db_accessadmin	16385	16385		
RFID_test	db_securityadmin	16386	16386		
RFID_test	db_ddladmin	16387	16387		
RFID_test	db_backupoperator	16389	16389		
RFID_test	db_datareader	16390	16390		
RFID_test	db_datawriter	16391	16391		
RFID_test	db_denydatareader	16392	16392		
RFID_test	db_denydatawriter	16393	16393		
**********	**********	**********	**********	**********	**********
(0 rows affected)					
User: arkar	Script: SSFA_Schemas.sql	SPID: 53	Closed on Oct 25 2010 6:32AM		
(0 rows affected)					
**********	**********	**********	**********	**********	**********

Figure A13.18: Result of the Acquired Victim's SQL Server Schema Information

session_id	login_time	host_name	program_name	login_name	nt_domain	nt_user_na
1	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
2	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
3	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
4	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
5	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
6	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
7	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
8	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
9	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
10	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
11	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
12	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
13	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
14	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
15	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
16	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
17	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
18	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
19	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
20	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
21	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
22	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
23	05/10/2010 01:23:09 a.m.	NULL	NULL	sa	NULL	NULL
51	08/10/2010 06:11:53 p.m.	TEST-STATION	Microsoft SQL Server Management Studio Express	TEST-STATION\vvy6982	TEST-STATION	vvy6982
52	12/10/2010 06:48:17 p.m.	TEST-STATION	Microsoft SQL Server Management Studio Express	TEST-STATION\vvy6982	TEST-STATION	vvy6982
53	25/10/2010 06:31:54 a.m.	TEST-STATION	SQLCMD	arkar		
54	12/10/2010 03:32:48 p.m.	TEST-STATION	Microsoft SQL Server Management Studio Express - Query	TEST-STATION\vvy6982	TEST-STATION	vvy6982
**********	**********	**********	**********	**********	**********	**********
(27 rows affected)						
User: arkar	Script: SSFA_Sessions.sql	SPID: 53	Closed on Oct 25 2010 6:31AM			

Figure A13.19: Fragment of the Acquired Victim's SQL Server Schema Information

Mon Oct 25 06:31:36 2010 -- Starting WFT Run...

Figure A13.20: Timestamp of the beginning of evidence acquisition by using WFT

cpu_ticks	ms_ticks	date_time				
5250503355222950	1750216093	25/10/2010 06:32:18 a.m.				
(1 rows affected)						
(1 rows affected)						
(1 rows affected)						
(1 rows affected)						
(1 rows affected)						
User: arkar	Script: SSFA_TimeConfig.sql	SPID: 53	Closed on Oct 25 2010 6:32AM			
(1 rows affected)						

Figure A13.21: Result of the Acquired Victim's SQL Server Schema Information

RFID_test	00000014:00000176:000d	LOP_MODIFY_COLUMN	0000:00000339	dbo.rfid_db.PK_rfid_db	0001:0000009d	12	NULL	NULL	NULL	NULL	NULL	NULL	NULL	
RFID_test	00000014:00000176:000e	LOP_MODIFY_COLUMN	0000:00000339	dbo.rfid_db.PK_rfid_db	0001:0000009d	13	NULL	NULL	NULL	NULL	NULL	NULL	NULL	
RFID_test	00000014:00000176:000f	LOP_MODIFY_COLUMN	0000:00000339	dbo.rfid_db.PK_rfid_db	0001:0000009d	14	NULL	NULL	NULL	NULL	NULL	NULL	NULL	
RFID_test	00000014:00000176:0010	LOP_MODIFY_ROW	0000:00000339	dbo.rfid_db.PK_rfid_db	0001:0000009d	15	53	NULL	NULL	NULL	NULL	NULL	NULL	
RFID_test	00000014:00000176:0011	LOP_MODIFY_COLUMN	0000:00000339	dbo.rfid_db.PK_rfid_db	0001:0000009d	16	NULL	NULL	NULL	NULL	NULL	NULL	NULL	
RFID_test	00000014:00000176:0012	LOP_MODIFY_COLUMN	0000:00000339	dbo.rfid_db.PK_rfid_db	0001:0000009d	17	NULL	NULL	NULL	NULL	NULL	NULL	NULL	
RFID_test	00000014:00000176:0013	LOP_MODIFY_ROW	0000:00000339	dbo.rfid_db.PK_rfid_db	0001:0000009d	18	53	NULL	NULL	NULL	NULL	NULL	NULL	
RFID_test	00000014:00000176:0014	LOP_COMMIT_XACT	0000:00000339	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	2010/10/12 18:39:50:030	
(0 rows affected)														
(0 rows affected)														
(0 rows affected)														
(0 rows affected)														
(0 rows affected)														
User: arkar	Script: SSFA_Tlog.sql	SPID: 53	Closed on Oct 25 2010 6:31AM											
(0 rows affected)														

Figure A13.22: Result of the Acquired SQL Server Transaction Log Entries Information

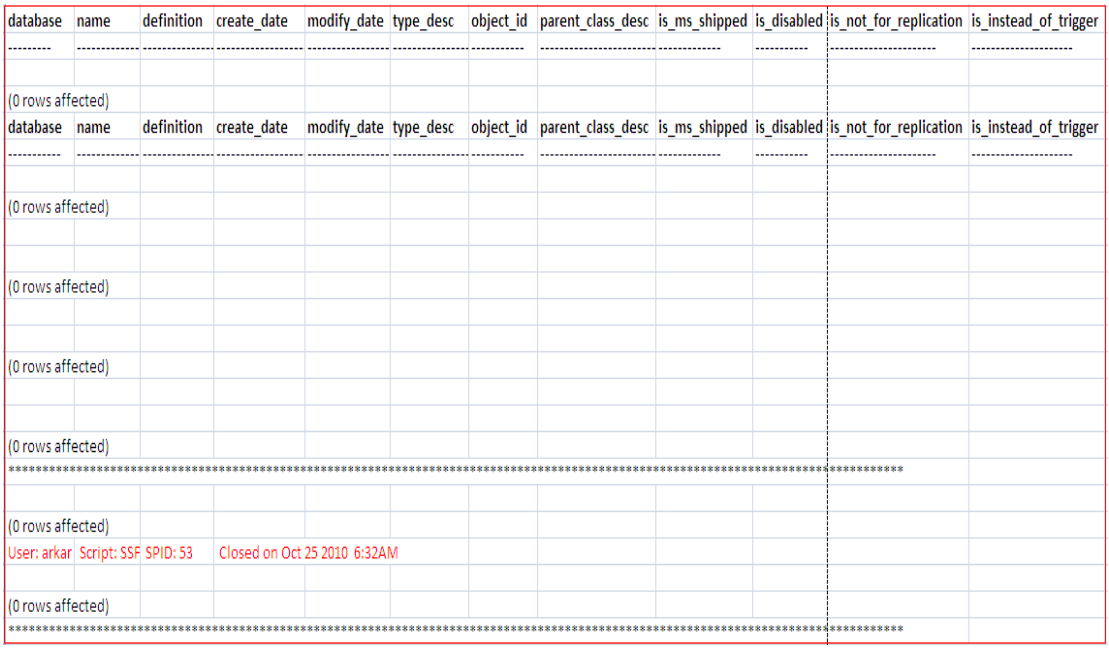


Figure A13.23: Result of the Acquired SQL Server Trigger Information

Appendix 14: Hash Values of the Acquired Evidences by using Extended WFT

```
============================================================
Windows Forensic Toolchest(TM) (WFT) v3.0.05
Copyright (C) 2003-2010 Monty McDougal.  All rights reserved.
http://www.foolmoon.net/security/
============================================================

BC93A20C1717175A5202555F516ED202          start.txt
AB95BD58A9CD21187A4DF339E8040A6E          DataCache.txt
AF3692B98C409A707DA4818ED8B70A2F          PlanCache.txt
4F3F10EBEE09A5E8DD7EE65D4BFE3DE8          Tlog.txt
194CAFE3E27FAF49F92C9CD23B5F7DD3          RecentStatements.txt
A7A43C52521F85BDD6683D9E0E4A00BD          Connections.txt
37F47A54ED052372F512CDA52AB33335          Sessions.txt
B1A6082CB887D2DE069E59F56B63A309          Logins.txt
99027343E1CA315527BEAAB193A19460          DbUsers.txt
D20DEC1F09FE06E10E1305AD89649E3C          DatabaseObjects.txt
C4BC1039910D88DDDC28F115C79B7936          Triggers.txt
5991A01962426D20E4415920A6FB4C31          Jobs.txt
4A2C5101BD3258D6C6ECCD06A9D286A9          JobHistory.txt
AA14DC79708388E4E1797AF2EDA8D4D0          CLR.txt
EE8A1A7D7E32043FDD6468DFC85F24BF          Databases.txt
33D139D0230377D50B1D51F56EA8D5B9 DB       ServerInfo.txt
88F65AA67A74CB9218BECAD1B9153CED          Configuration.txt
ECF3CB5D8F259AD36814696C680BBB81          Schemas.txt
9C6239D845BCDC15C9504FF7D2B303A9          Endpoints.txt
```

72CBADAE0A1F9FF34D4A35A9AB26BDB8 AutoExecProcs.txt

4E7DAAC41474DCC469FAF57620EE1A8E Time.txt

D28A565258A6C5F5863014FBA380D1E9 ClockHands.txt

037130BF88E524D8F61AA95EB8AF3C3B pclip.txt

A7537E6630A8D7DC8F1A8455A8FDC68E MEM_P.txt

C9DD6E1FFA4782613E65C87312F0E4BB MEM_D.txt

61C47A3D9F316C4DABBAD16AF15EE48A hostname.txt

E80251575BCD586A4C633D296BBB6F79 uname.txt

BD434867C60C1D657D9F6CAF64D80FF9 ver.txt

229AF64FD3E6A9F8A4C4CE3AC16359C3 environm.txt

654FF4D9BA4050B6495A62C636F45BD6 whoami.txt

B82FE5D8926AF2831BADDE7174DDFEEC netdom.txt

655988D2A3E42136AB57887597F99545 netuser.txt

C5FC9C8CE7E90103BBB77435370BF16C netgroup.txt

E6D74D46AECD969282A7362B8998A746 netlgrp.txt

83A23E9992174906473E0A739AA4DD6A netacct.txt

641E4B46F39DC9F57FE69FB0884016D7 netacctdom.txt

963D3EF25E8F338E59B9A30C3A1B1187 auditpol.txt

EF6E3F3F2E976685811C3A89F8CD0EB6 ps.txt

496E9B6CB213A8E50D587AC081B1DCD7 pstat.txt

CDF2D49E302522E370A4918E70E81D88 tlist_v.txt

9B5DD3F38833885843BCC2206DE0E50D tlist_s.txt

470422FD9F503565B1BA186311C94EAF tlist_c.txt

B44C85C704337F007FB47EDDA814865D cmdline.txt

5105E5160BC35EB5D56A4E2829795E7D procinterrogate.txt

1590BFF76FD1C79E28A6862B93E49EEB sc_query_ex.txt

1B3142D14ABE08FB5956CF273E926E48 netstart.txt

8E74213360EA0EAF0342B055528A7574	srvc.txt
96B45C89C8F164B9E9AB255A0D00E3CE	TaskList_v.txt
5B2E0D4C2359C98FA6F76D61A6C7E59D	TaskList_svc.txt
728F0B3A7C6DE4B7B9D838F9C0AFAB44	drivers.txt
9256B6C363C4296005BEAD1298BD643A	ipconfig.txt
FC52F40362493703F44B895FD7FF9149	iplist.txt
DFB5DDDFC15A7925629205E6EDF8562D	arp.txt
30FBE777154B4875B64912C34DE745AC	rtable.txt
620AD12BD3DFB5B47BFC128B13DDEFCD	netstat.txt
52EE5EEACDFD2BA082F52D9CFEA10708	netstatn.txt
9FC5FD6A033D6B3617840ECF63DED76D	fport_p.txt
9FC5FD6A033D6B3617840ECF63DED76D	fport_a.txt
4D06B65D83F276728E817C02FD95C40C	openports.txt
3DC9C248379C160466F583FC26752962	nbtstatn.txt
3DC9C248379C160466F583FC26752962	nbtstatc.txt
647578D95B7F3B11B6CCF833E296701E	nbtstats.txt
99553CCD8E14CAE6DB789BCE98CFCB73	hunt.txt
32F60215D6F7CE348C66FE6D6AFB7081	netshare.txt
9D8780AF2919A4F1B1532C208720ED14	netuse.txt
768165E0ABF16BF3056836D5431A7296	netview.txt
768165E0ABF16BF3056836D5431A7296	netsessi.txt
6D4F36367CB6120BA92ADAA77DA42BEA	ndis.txt
E9B3135E5CD46F2E434E67A0DE631F95	promiscdetect.txt
AB20A9482744E11727C791BC855717B8	netusers_local.txt
C4DA0F6D85F641B3E5A0F065AEDB2701	netusers_local_history.txt
2428169B31C96FE4A767215000AA0008	success.txt
2428169B31C96FE4A767215000AA0008	failed.txt

2428169B31C96FE4A767215000AA0008	interact.txt
2428169B31C96FE4A767215000AA0008	remote.txt
7583EE9E2E28231E98A616966696F439	syslog.txt
4A25F00467D15FE9FAE2777769518518	applog.txt
D41D8CD98F00B204E9800998ECF8427E	seclog.txt
768165E0ABF16BF3056836D5431A7296	netfile.txt
877A6DF5F576B096D28D049AE2D1645C	recent.txt
8609E8E78D5E28A8DB50BA4457383F6D	C_recycle.txt
11B255D75EE6393D6CD71BF1879C3488	prefetch.txt
620AD01A32B3E6F4BC54365E576E9122	C_freesp.txt
D41D8CD98F00B204E9800998ECF8427E	autoexec.txt
8C1DFF16CD151B49143C48796BB750FD	win_ini.txt
B143A6852C9EF93E0BDECB02F524F9F2	sys_ini.txt
DF5DC1ABC0D52F3C9E931E26A7C0065C	winstart.txt
DF5DC1ABC0D52F3C9E931E26A7C0065C	init_ini.txt
A220C7BD4ACE2153DE5E30EDA85C945B	startup.txt
C5B62B38F85A47009AC1362C007FF7A0	user_startup.txt
597F96E64A39D161F9C79BEE54ACBB42	tasks.txt
90E2BE5857E337CBDBFFDE58B2E91C81	at.txt
FA00DDD2F8FE5C3A55E96BCCB6C87FFF	schtasks.txt
688BC52D6AA90F2204DC16E71B81C24E	hklm_r.txt
992371965FAE1F03A17F5AAF5E6598BC	hklm_ro.txt
2222CCDED19CD28CF6B9C0432C445664	hklm_rox.txt
629680BFDCB03EB78749D094FEBDBCB9	hklm_rs.txt
629680BFDCB03EB78749D094FEBDBCB9	hklm_rso.txt
629680BFDCB03EB78749D094FEBDBCB9	hklm_scripts.txt
629680BFDCB03EB78749D094FEBDBCB9	hklm_expl_run.txt

9E7B99199B2931E811AB482B833C2006	hkcu_r.txt
992371965FAE1F03A17F5AAF5E6598BC	hkcu_ro.txt
629680BFDCB03EB78749D094FEBDBCB9	hkcu_rox.txt
629680BFDCB03EB78749D094FEBDBCB9	hkcu_rs.txt
629680BFDCB03EB78749D094FEBDBCB9	hkcu_rso.txt
629680BFDCB03EB78749D094FEBDBCB9	hkcu_scripts.txt
629680BFDCB03EB78749D094FEBDBCB9	hkcu_expl_run.txt
E9B3135E5CD46F2E434E67A0DE631F95	gplist.txt
96584E6AD16BE6C847166D809F70033A	gpuser.txt
C944FA3824417A3AE083C45473281BD7	gpsys.txt
D702985D915284E12B59EACB37A6F7A1	run_hist.txt
D51794C380B3AAA1435D0892A0F8C5CA	rcnt_doc.txt
82B262A274831C8AEEFF58322720484F	lastsave.txt
49BCFC17BA427FA4B3EE188147DA0BEE	installh.txt
2B0BED64D2E3DAC4053F40CF2982B75C	hklm_safemin.txt
DD074F35C25931D4C193287358E5F594	hklm_safenet.txt
062E3074795A47CF35DEB4E8197C3EE9	iehv.txt
60623F0A8E30BE61CDE155C22626877B	type_url.txt
7FE3BD462D6F904DDB033D2B1A93BB04	search_h.txt
E9B3135E5CD46F2E434E67A0DE631F95	pstoreview.txt
D41D8CD98F00B204E9800998ECF8427E	doskey.txt
70EE549E2B10AF81209C3B57B8A6A517	mdm.txt
A5F255DDBD261CFED9DCB16263317FF5	end.txt

Appendix 15: WFT Tool Log during the Acquisition of SQL Server Artefacts from the compromised RFID Business System

Investigator Name: Ar Kar KYAW

Case: SQL Server Incident

OS: xp (auto detected)

Shell: 'tools\xp\cmd.exe'

(md5=6D778E0F95447E6546553EEEA709D03C)

Config File: 'wftsql.cfg'

(md5=1D378B80F8E0E7EA4F35EC8B92989CD1)

Tool Path: 'tools\'

Destination: 'E:\VolatileSQLDATA\'

Hash: md5

Drive(s): 'C'

Run Slow Apps: FALSE

Run Disk Modifying Apps: FALSE

Run Reports: TRUE

Run With Prunetools: TRUE

Run With Color: TRUE

Run Interactive: FALSE

Run With Prompting: FALSE

Run Web Browser: FALSE

Report Date/Time: 10/25/2010 06:18:44 (24h)

==========

06:31:34: [RUN STARTING]

==========

06:31:34: Verifying 'wft.exe' OK

(md5=99528174FD235E51F67C127766E7FD4A)

[START]

06:31:35: Verifying '2k\res_kit\now.exe' OK

(md5=1CD2DF306E25FBDDF653A9D9B5DC8A41)

06:31:35: Running '2k\res_kit\now.exe' [#1/169]

COMPLETE

'start.txt'

(md5=BC93A20C1717175A5202555F516ED202)

'start.htm'

(md5=4D3EF39474C2436B270A26700E510D95)

06:31:36: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:31:36: Verifying 'sql\sqlcmd.exe' OK

(md5=28731C04B854CC1570DBDACC89A6C3F2)

[SQL SERVER]

[Recent Activity]

06:31:36: Verifying 'sql\SSFA_DataCache.sql' OK

(md5=B812FF207D53B8DDA0F5C3A56CFDDB0E)

06:31:36: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:31:36: Running 'sql\runsql.bat' [#2/169]

COMPLETE

'DataCache.txt'

(md5=AB95BD58A9CD21187A4DF339E8040A6E)

'DataCache.htm'

(md5=F53C4254FD4D19E858C5F921AE220373)

06:31:41: Verifying 'sql\SSFA_PlanCache.sql' OK

(md5=9A216F5FAC3478EC8B488B0C8F5406F5)

06:31:41: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:31:41: Running 'sql\runsql.bat' [#3/169]

COMPLETE

'PlanCache.txt'

(md5=AF3692B98C409A707DA4818ED8B70A2F)

'PlanCache.htm'

(md5=911BBF966F83A04687BA8833A34A5120)

06:31:48: Verifying 'sql\SSFA_TLog.sql' OK

(md5=F51463E731B82396D80E0D8D21ECD21F)

06:31:48: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:31:48: Running 'sql\runsql.bat' [#4/169]

COMPLETE

'Tlog.txt'

(md5=4F3F10EBEE09A5E8DD7EE65D4BFE3DE8)

'Tlog.htm'

(md5=41C77BAB1DB11C0610CFDB1EBE6728AE)

06:31:51: Verifying 'sql\SSFA_RecentStatements.sql' OK

(md5=D7E3D283EA517BC5B02D6EB71E5F9C2C)

06:31:52: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:31:52: Running 'sql\runsql.bat' [#5/169]

COMPLETE

'RecentStatements.txt'

(md5=194CAFE3E27FAF49F92C9CD23B5F7DD3)

'RecentStatements.htm'

(md5=C719DD7E6DDEBBC5A3D7302E11185A76)

[Active Connections]

06:31:53: Verifying 'sql\SSFA_Connections.sql' OK

(md5=66266B9645B366F4E7167FA909CA30DC)

06:31:53: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:31:53: Running 'sql\runsql.bat' [#6/169]

COMPLETE

'Connections.txt'

(md5=A7A43C52521F85BDD6683D9E0E4A00BD)

'Connections.htm'

(md5=2CEA5C9923F35D0465906A22AA67BBFD)

06:31:53: Verifying 'sql\SSFA_Sessions.sql' OK

(md5=67CEE2132CC6059B994D51696BC56C53)

06:31:53: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:31:53: Running 'sql\runsql.bat' [#7/169]

COMPLETE

'Sessions.txt'

(md5=37F47A54ED052372F512CDA52AB33335)

'Sessions.htm'

(md5=D2F90BF630FC5A49E55F0DE80C08E4B0)

[DB Objects & Users]

06:31:54: Verifying 'sql\SSFA_Logins.sql' OK

(md5=42DA8E136A8E7B104AFFE7AA678AE141)

06:31:54: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:31:54: Running 'sql\runsql.bat' [#8/169]

COMPLETE

'Logins.txt'

(md5=B1A6082CB887D2DE069E59F56B63A309)

'Logins.htm'

(md5=FE0B6F041AB85C34D586DC002CD06AD8)

06:31:57: Verifying 'sql\SSFA_DbUsers.sql' OK

(md5=1FA12BB80DF343E0451D66309334D78D)

06:31:57: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:31:57: Running 'sql\runsql.bat' [#9/169]

COMPLETE

'DbUsers.txt'

(md5=99027343E1CA315527BEAAB193A19460)

'DbUsers.htm'

(md5=1CDDFCA585210CB3A355D049399378D4)

06:31:58: Verifying 'sql\SSFA_DBObjects.sql' OK

(md5=698ABFF6AACA4C52F6864249BD658F9D)

06:31:58: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:31:58: Running 'sql\runsql.bat' [#10/169]

COMPLETE

'DatabaseObjects.txt'

(md5=D20DEC1F09FE06E10E1305AD89649E3C)

'DatabaseObjects.htm'

(md5=BD347B43DA5EE38E15ECD482F473E197)

06:31:59: Verifying 'sql\SSFA_Triggers.sql' OK

(md5=AB700D66945BBC9FBFA34002A808A208)

06:31:59: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:31:59: Running 'sql\runsql.bat' [#11/169]

COMPLETE

'Triggers.txt'

(md5=C4BC1039910D88DDDC28F115C79B7936)

'Triggers.htm'

(md5=DEE8D4F690AC5196253059CAE0F1BAB1)

06:32:00: Verifying 'sql\SSFA_Jobs.sql' OK

(md5=7376F41C4390BEFF4D4CBA4289901885)

06:32:00: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:32:00: Running 'sql\runsql.bat' [#12/169]

COMPLETE

'Jobs.txt'

(md5=5991A01962426D20E4415920A6FB4C31)

'Jobs.htm'

(md5=9219AFE63FFD1E4E308B6216083F387A)

06:32:01: Verifying 'sql\SSFA_JobHistory.sql' OK

(md5=A104029D94AA4780E0A4A0A1BC54106A)

06:32:02: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:32:02: Running 'sql\runsql.bat' [#13/169]

COMPLETE

'JobHistory.txt'

(md5=4A2C5101BD3258D6C6ECCD06A9D286A9)

'JobHistory.htm'

(md5=EA63C5853BB71AC2FCDD36CC132B7F31)

06:32:03: Verifying 'sql\SSFA_CLR.sql' OK

(md5=B6DAE2E421A639536C01517735B9D6C9)

06:32:03: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:32:03: Running 'sql\runsql.bat' [#14/169]

COMPLETE

'CLR.txt'

(md5=AA14DC79708388E4E1797AF2EDA8D4D0)

'CLR.htm'

(md5=A670969DC7976DEFC19B976CD850B1ED)

06:32:03: Verifying 'sql\SSFA_Databases.sql' OK

(md5=4DC0305DB714B48B6EC49DE3BA90F471)

06:32:03: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:32:03: Running 'sql\runsql.bat' [#15/169]

COMPLETE

'Databases.txt'

(md5=EE8A1A7D7E32043FDD6468DFC85F24BF)

'Databases.htm'

(md5=F416BF7233E960130FDDD1340E5DBC5B)

[DB Configuration]

06:32:04: Verifying 'sql\SSFA_DbSrvInfo.sql' OK
(md5=8EFB9C50FEFCA57E45570D772AABDE22)
06:32:04: Verifying 'sql\runsql.bat' OK
(md5=FEDD1E9DF1B76C877B8027B62E401B50)
06:32:04: Running 'sql\runsql.bat' [#16/169]
COMPLETE
'DBServerInfo.txt'
(md5=33D139D0230377D50B1D51F56EA8D5B9)
'DBServerInfo.htm'
(md5=C7618040A9D3BA9242842A2F7D4E300F)

06:32:08: Verifying 'sql\SSFA_Configurations.sql' OK
(md5=18D4261C8480B6593861F6704CE1BED6)
06:32:08: Verifying 'sql\runsql.bat' OK
(md5=FEDD1E9DF1B76C877B8027B62E401B50)
06:32:08: Running 'sql\runsql.bat' [#17/169]
COMPLETE
'Configuration.txt'
(md5=88F65AA67A74CB9218BECAD1B9153CED)
'Configuration.htm'
(md5=554C0C23FFB63ECCD6A3A3E18D084CA4)

06:32:15: Verifying 'sql\SSFA_Schemas.sql' OK

(md5=026EA797C9FD0681C363B095BC32C049)

06:32:15: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:32:15: Running 'sql\runsql.bat' [#18/169]

COMPLETE

'Schemas.txt'

(md5=ECF3CB5D8F259AD36814696C680BBB81)

'Schemas.htm'

(md5=9B01A502967CBD60726ACE5CD227601E)

06:32:16: Verifying 'sql\SSFA_EndPoints.sql' OK

(md5=7BF03126871B6CDDD6ACE5A538EC5E7C)

06:32:16: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:32:16: Running 'sql\runsql.bat' [#19/169]

COMPLETE

'Endpoints.txt'

(md5=9C6239D845BCDC15C9504FF7D2B303A9)

'Endpoints.htm'

(md5=D23EEF6404CA44B2ABF23CD51025522E)

06:32:16: Verifying 'sql\SSFA_AutoEXEC.sql' OK

(md5=9CC9D8FC8FF0C0C10D18DA1AABFF62C8)

06:32:16: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:32:16: Running 'sql\runsql.bat' [#20/169]

COMPLETE

'AutoExecProcs.txt'

(md5=72CBADAE0A1F9FF34D4A35A9AB26BDB8)

'AutoExecProcs.htm'

(md5=FE962FCFBD57BC62420E26EBE5869D4C)

06:32:17: Verifying 'sql\SSFA_TimeConfig.sql' OK

(md5=EA32425B7410CF9EE4938D2B5F7671BB)

06:32:17: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:32:17: Running 'sql\runsql.bat' [#21/169]

COMPLETE

'Time.txt'

(md5=4E7DAAC41474DCC469FAF57620EE1A8E)

'Time.htm'

(md5=B12DD416C44CAE9F863EAED2D7443410)

06:32:18: Verifying 'sql\SSFA_ClockHands.sql' OK

(md5=7880CA627188AD8CE9786F7D71F5D392)

06:32:18: Verifying 'sql\runsql.bat' OK

(md5=FEDD1E9DF1B76C877B8027B62E401B50)

06:32:18: Running 'sql\runsql.bat' [#22/169]

COMPLETE

'ClockHands.txt'

(md5=D28A565258A6C5F5863014FBA380D1E9)

'ClockHands.htm'

(md5=3D1C61D98B842866905A2BA74178C11F)

[MEMORY]

06:32:19: Verifying 'unxutils\pclip.exe' OK

(md5=1C35D256AC672A8738D5A172C06CC125)

06:32:19: Running 'unxutils\pclip.exe' [#23/169]

NOTE: pclip will return an error if there is no text on clipboard

COMPLETE

'pclip.txt'

(md5=037130BF88E524D8F61AA95EB8AF3C3B)

'pclip.htm'

(md5=95B886FC2579C025397977B1688F7173)

06:32:20: Verifying 'xp\mem.exe' OK

(md5=390762963E6B4C861E5E0CA5A3E56E40)

06:32:20: Running 'xp\mem.exe' [#24/169]

COMPLETE

'MEM_P.txt'

(md5=A7537E6630A8D7DC8F1A8455A8FDC68E)

'MEM_P.htm'

(md5=8F98ECB2391B2C1064DCED44B523E326)

06:32:21: Verifying 'xp\mem.exe' OK

(md5=390762963E6B4C861E5E0CA5A3E56E40)

06:32:21: Running 'xp\mem.exe' [#25/169]

COMPLETE

'MEM_D.txt'

(md5=C9DD6E1FFA4782613E65C87312F0E4BB)

'MEM_D.htm'

(md5=DE7632796CC0475B4154B139D3D4574E)

[MAC TIME]

06:32:21: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:32:21: Running 'tools\xp\cmd.exe' [#26/169]

SKIPPED (via '-noslow' parameter)

'C_atime.txt'

(md5=5F45B5367C7ED885270EA969851CB439)

'C_atime.htm'

(md5=63B9EC1C0055B67FCF64D8BE9AE7CD43)

06:32:21: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:32:21: Running 'tools\xp\cmd.exe' [#27/169]

SKIPPED (via '-noslow' parameter)

'C_ctime.txt'

(md5=5F45B5367C7ED885270EA969851CB439)

'C_ctime.htm'

(md5=94E433599BC06482D5DCDEAA405F94F7)

06:32:21: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:32:21: Running 'tools\xp\cmd.exe' [#28/169]

SKIPPED (via '-noslow' parameter)

'C_mtime.txt'

(md5=5F45B5367C7ED885270EA969851CB439)

'C_mtime.htm'

(md5=4FF13F8F3B55CB67AF42448531F6478C)

06:32:22: Verifying 'perl\p2x561.dll' OK

(md5=AB773B261948B8D1DED5454DB66CBB41)

06:32:22: Verifying 'perl\mac.exe' OK

(md5=388631FC7DD59959A26F246FC37034FA)

06:32:22: Running 'perl\mac.exe' [#29/169]

SKIPPED (via '-nowrite' parameter)

'C_mac.txt'

(md5=76DCF11D5C62E135333988A177EB49BA)

'C_mac.htm'

(md5=B4D8B1AF2818E68F7C6A666FD55D5AD3)

[SYSTEM INFO]

06:32:22: Verifying 'sysinternals\psinfo.exe' OK

(md5=797D6659261D6E6D31AA6086A90B971A)

06:32:22: Running 'sysinternals\psinfo.exe' [#30/169]

SKIPPED (via '-nowrite' parameter)

'psinfo.txt'

(md5=1709BDB4E50D6884682A5077CAB5507B)

'psinfo.htm'

(md5=5F9DBF2F0C7E01CD972B300BAE0F8748)

06:32:22: Verifying 'xp\hostname.exe' OK

(md5=13253731D13168EF06DCA97F70AD57CC)

06:32:22: Running 'xp\hostname.exe' [#31/169]

COMPLETE

'hostname.txt'

(md5=61C47A3D9F316C4DABBAD16AF15EE48A)

'hostname.htm'

(md5=884632E5F6DBC750B13E6B028EF0C02D)

06:32:22: Verifying 'unxutils\uname.exe' OK

(md5=463CFAC34C9BD65C77BD98C529DF845A)

06:32:23: Running 'unxutils\uname.exe' [#32/169]

COMPLETE

'uname.txt'

(md5=E80251575BCD586A4C633D296BBB6F79)

'uname.htm'

(md5=F78E203E3479254600B8BE2E0CBED42E)

06:32:23: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:32:23: Running 'tools\xp\cmd.exe' [#33/169]

COMPLETE

'ver.txt'

(md5=BD434867C60C1D657D9F6CAF64D80FF9)

'ver.htm'

(md5=5E0A1DD0A0BB0375FD4B229223E84CDA)

06:32:23: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:32:23: Running 'tools\xp\cmd.exe' [#34/169]

COMPLETE

'environm.txt'

(md5=229AF64FD3E6A9F8A4C4CE3AC16359C3)

'environm.htm'

(md5=AB1CAD60791F2DFAB972A674465D963D)

06:32:23: Verifying 'xp\..\microsoft\uptime.exe' OK

(md5=D41D8CD98F00B204E9800998ECF8427E)

06:32:23: Running 'xp\..\microsoft\uptime.exe' [#37/169]

SKIPPED (via '-noslow' parameter)

'uptime.txt'

(md5=3D823D7CC9EA0ED4D6E4A20D88395CC3)

'uptime.htm'

(md5=B7456D103CBF5E49F1574A845EF79218)

06:32:23: Verifying 'xp\..\microsoft\uptime.exe' OK

(md5=D41D8CD98F00B204E9800998ECF8427E)

06:32:23: Running 'xp\..\microsoft\uptime.exe' [#40/169]

SKIPPED (via '-noslow' parameter)

'uptime_h.txt'

(md5=3D823D7CC9EA0ED4D6E4A20D88395CC3)

'uptime_h.htm'

(md5=4975585259A0481F4323AC25E6ACDCF9)

06:32:23: Verifying 'unxutils\whoami.exe' OK

(md5=D166374D267A2B4CF8F5E00ABE8BEDF1)

06:32:23: Running 'unxutils\whoami.exe' [#41/169]

COMPLETE

'whoami.txt'

(md5=654FF4D9BA4050B6495A62C636F45BD6)

'whoami.htm'

(md5=65DEF1693A2DDBE20D365F8B29399857)

06:32:24: Verifying 'xp\net.exe' OK

(md5=FD3DA8425624B98903407DF608CF2C11)

06:32:24: Running 'xp\net.exe' [#42/169]

COMPLETE

'netdom.txt'

(md5=B82FE5D8926AF2831BADDE7174DDFEEC)

'netdom.htm'

(md5=924195E7FF4DE13F4F07BFC1599929BE)

06:32:24: Verifying 'xp\net.exe' OK

(md5=FD3DA8425624B98903407DF608CF2C11)

06:32:24: Running 'xp\net.exe' [#43/169]

COMPLETE

'netuser.txt'

(md5=655988D2A3E42136AB57887597F99545)

'netuser.htm'

(md5=4F0337A047D690382370317266069C59)

06:32:24: Verifying 'xp\net.exe' OK

(md5=FD3DA8425624B98903407DF608CF2C11)

06:32:24: Running 'xp\net.exe' [#44/169]

COMPLETE

'netgroup.txt'

(md5=C5FC9C8CE7E90103BBB77435370BF16C)

'netgroup.htm'

(md5=4AF52DBB00F3B4F9DE9B2BEA14B244BC)

06:32:25: Verifying 'xp\net.exe' OK

(md5=FD3DA8425624B98903407DF608CF2C11)

06:32:25: Running 'xp\net.exe' [#45/169]

COMPLETE

'netlgrp.txt'

(md5=E6D74D46AECD969282A7362B8998A746)

'netlgrp.htm'

(md5=8147E2BEA1E81DA7733766C35269C93A)

06:32:25: Verifying 'xp\net.exe' OK

(md5=FD3DA8425624B98903407DF608CF2C11)

06:32:25: Running 'xp\net.exe' [#46/169]

COMPLETE

'netacct.txt'

(md5=83A23E9992174906473E0A739AA4DD6A)

'netacct.htm'

(md5=AD1ED006744E205AD27C196A10DB8CA4)

06:32:25: Verifying 'xp\net.exe' OK

(md5=FD3DA8425624B98903407DF608CF2C11)

06:32:25: Running 'xp\net.exe' [#47/169]

COMPLETE

'netacctdom.txt'

(md5=641E4B46F39DC9F57FE69FB0884016D7)

'netacctdom.htm'

(md5=376B0014907EB503118C604B967FEEB0)

06:32:25: Verifying '2k\res_kit\auditpol.exe' OK

(md5=2F0050F870B2D49E0880334E4938D528)

06:32:25: Running '2k\res_kit\auditpol.exe' [#48/169]

COMPLETE

'auditpol.txt'

(md5=963D3EF25E8F338E59B9A30C3A1B1187)

'auditpol.htm'

(md5=86DF1FFA937DA57655981A8CC17A340D)

[PROCESSES]

06:32:26: Verifying 'sysinternals\pslist.exe' OK

(md5=61FD7759F215F9F88AE88525FD30AF21)

06:32:26: Running 'sysinternals\pslist.exe' [#49/169]

SKIPPED (via '-nowrite' parameter)

'pslist.txt'

(md5=5F2A642D67EB471CD74D5529C519EF31)

'pslist.htm'

(md5=79DBEA76B9829726A0D7EF704B12774B)

06:32:26: Verifying 'sysinternals\listdlls.exe' OK

(md5=6DB9565378D0268DCD88288C5E961611)

06:32:26: Running 'sysinternals\listdlls.exe' [#52/169]

SKIPPED (via '-nowrite' parameter)

'listdlls.txt'

(md5=7E2C1A0055336DA76A6CA8D03FA4B015)

'listdlls.htm'

(md5=ACDDF8B75BE89C52BC8B987C80858416)

06:32:26: Verifying 'cygwin\cygwin1.dll' OK

(md5=E8CD5A2BA5D93ACCE6C28C26BF5717FB)

06:32:26: Verifying 'cygwin\ps.exe' OK

(md5=52C2ABBC6ACB9C8A48BF200617214A9D)

06:32:26: Running 'cygwin\ps.exe' [#53/169]

COMPLETE

'ps.txt'

(md5=EF6E3F3F2E976685811C3A89F8CD0EB6)

'ps.htm'

(md5=91F4CE8AC2039D5E6F7716DC1E0130AB)

06:32:28: Verifying '2k\res_kit\pstat.exe' OK

(md5=83C409F2459F565EC259E021BD23B0F2)

06:32:28: Running '2k\res_kit\pstat.exe' [#54/169]

COMPLETE

'pstat.txt'

(md5=496E9B6CB213A8E50D587AC081B1DCD7)

'pstat.htm'

(md5=A9853633741F6A687E0FF281F8E8B018)

06:32:29: Verifying 'microsoft\tlist.exe' OK

(md5=3A1A3E2BE29C0558BE4958394B7114BB)

06:32:29: Running 'microsoft\tlist.exe' [#55/169]

COMPLETE

'tlist_v.txt'

(md5=CDF2D49E302522E370A4918E70E81D88)

'tlist_v.htm'

(md5=C82F579A95F60EF09CFDE1315AF9EB30)

06:32:30: Verifying 'microsoft\tlist.exe' OK

(md5=3A1A3E2BE29C0558BE4958394B7114BB)

06:32:30: Running 'microsoft\tlist.exe' [#56/169]

COMPLETE

'tlist_s.txt'

(md5=9B5DD3F38833885843BCC2206DE0E50D)

'tlist_s.htm'

(md5=DA213C53C804EFBFDB3D904C8BEA7652)

06:32:30: Verifying 'microsoft\tlist.exe' OK

(md5=3A1A3E2BE29C0558BE4958394B7114BB)

06:32:30: Running 'microsoft\tlist.exe' [#57/169]

COMPLETE

'tlist_c.txt'

(md5=470422FD9F503565B1BA186311C94EAF)

'tlist_c.htm'

(md5=614831108114806947BA7FC95817CDC0)

06:32:30: Verifying 'diamondcs\cmdline.exe' OK

(md5=2D75635F4FAB479E3385DC0A1EE51F36)

06:32:30: Running 'diamondcs\cmdline.exe' [#58/169]

COMPLETE

'cmdline.txt'

(md5=B44C85C704337F007FB47EDDA814865D)

'cmdline.htm'

(md5=E973CF07176D49FBA0D156AC1661B4F6)

06:32:30: Verifying 'sysinternals\handle.exe' OK

(md5=8D4150A8B6DD5B25577D4E39EE0DD6D8)

06:32:31: Running 'sysinternals\handle.exe' [#59/169]

SKIPPED (via '-nowrite' parameter)

'handle.txt'

(md5=9B80DD8A3300571406BE1429F663778F)

'handle.htm'

(md5=A8FD25F379CDA2A73D9E01EC1A5F4C4D)

06:32:31: Verifying 'winfingerprint\procinterrogate.exe' OK

(md5=59F4952531F1E90F566E0ACE739D8E8A)

06:32:31: Running 'winfingerprint\procinterrogate.exe' [#60/169]

COMPLETE

'procinterrogate.txt'

(md5=5105E5160BC35EB5D56A4E2829795E7D)

'procinterrogate.htm'

(md5=86589FE6B57CB113C3779BDC117C56F1)

[SERVICES]

06:32:32: Verifying 'sysinternals\psservice.exe' OK

(md5=5115187A2CC3AD6F983BF8DD6EFA3969)

06:32:32: Running 'sysinternals\psservice.exe' [#61/169]

SKIPPED (via '-nowrite' parameter)

'psservice.txt'

(md5=6EE3210D36A0988409608D8823A72F9A)

'psservice.htm'

(md5=BC1976560904B5269F3866184C666F15)

06:32:32: Verifying '2k\res_kit\sc.exe' OK

(md5=0C264A329931469DD50EDB6AE446C45A)

06:32:32: Running '2k\res_kit\sc.exe' [#62/169]

COMPLETE

'sc_query_ex.txt'

(md5=1590BFF76FD1C79E28A6862B93E49EEB)

'sc_query_ex.htm'

(md5=32C72D6BCA4EC46B5DCC4A5D72D517A3)

06:32:32: Verifying 'xp\net.exe' OK

(md5=FD3DA8425624B98903407DF608CF2C11)

06:32:32: Running 'xp\net.exe' [#63/169]

COMPLETE

'netstart.txt'

(md5=1B3142D14ABE08FB5956CF273E926E48)

'netstart.htm'

(md5=3D782C8667F2F5403117BBAE63A468F8)

06:32:32: Verifying 'netlatency\servicelist.exe' OK

(md5=EF97AA16ADE0A9F531F0EA8AA88F001D)

06:32:32: Running 'netlatency\servicelist.exe' [#64/169]

COMPLETE

'srvc.txt'

(md5=8E74213360EA0EAF0342B055528A7574)

'srvc.htm'

(md5=D55407E53427AF5B58E37BE7EE2B7AB6)

06:32:33: Verifying 'xp\tasklist.exe' OK

(md5=E8B108654C5789AD3F75E08B0A89C609)

06:32:33: Running 'xp\tasklist.exe' [#67/169]

COMPLETE

'TaskList_v.txt'

(md5=96B45C89C8F164B9E9AB255A0D00E3CE)

'TaskList_v.htm'

(md5=B70A085C1191ACA409F0DCEA2D8A260C)

06:32:33: Verifying 'xp\tasklist.exe' OK

(md5=E8B108654C5789AD3F75E08B0A89C609)

06:32:33: Running 'xp\tasklist.exe' [#68/169]

COMPLETE

'TaskList_svc.txt'

(md5=5B2E0D4C2359C98FA6F76D61A6C7E59D)

'TaskList_svc.htm'

(md5=53EB68F191F91A5F85AF6C74EBC86AA8)

[DRIVERS]

06:32:34: Verifying '2k\res_kit\drivers.exe' OK
(md5=43E2B940767D4F671A7D27A7B29D8A22)
06:32:34: Running '2k\res_kit\drivers.exe' [#71/169]
COMPLETE
'drivers.txt'
(md5=728F0B3A7C6DE4B7B9D838F9C0AFAB44)
'drivers.htm'
(md5=B87D89ADE181019C3CE4CA1F7ED13F4B)

[NETWORK INFO]

06:32:34: Verifying 'xp\ipconfig.exe' OK
(md5=34781A7E9683F42C4B2FE6F09456568C)
06:32:34: Running 'xp\ipconfig.exe' [#72/169]
COMPLETE
'ipconfig.txt'
(md5=9256B6C363C4296005BEAD1298BD643A)
'ipconfig.htm'
(md5=60264E9D8DA4EA0F3223ACEC4FC6EE34)

06:32:35: Verifying 'diamondcs\iplist.exe' OK

(md5=501008D70AEF2B7E4E010DB29E561598)

06:32:35: Running 'diamondcs\iplist.exe' [#73/169]

COMPLETE

'iplist.txt'

(md5=FC52F40362493703F44B895FD7FF9149)

'iplist.htm'

(md5=076EE1EAAB219DF451B23C6DBCF57F67)

06:32:36: Verifying 'xp\arp.exe' OK

(md5=33F9B0E02D9D93F920605D02FB53F3FD)

06:32:36: Running 'xp\arp.exe' [#74/169]

COMPLETE

'arp.txt'

(md5=DFB5DDDFC15A7925629205E6EDF8562D)

'arp.htm'

(md5=37397999D7D058E14FEADAB3AC2FF6B0)

06:32:36: Verifying 'xp\route.exe' OK

(md5=67D442F0DBEE60CFB43F821B554F44F6)

06:32:36: Running 'xp\route.exe' [#75/169]

COMPLETE

'rtable.txt'

(md5=30FBE777154B4875B64912C34DE745AC)

'rtable.htm'

(md5=30615C2E84547DB91A77BD11C7F0EE76)

06:32:37: Verifying 'xp\netstat.exe' OK

(md5=8A7EE413726790398D6B315B7CFB5B0A)

06:32:37: Running 'xp\netstat.exe' [#76/169]

COMPLETE

'netstat.txt'

(md5=620AD12BD3DFB5B47BFC128B13DDEFCD)

'netstat.htm'

(md5=DBEAC920688CECC9BF775B9C74F0C1CA)

06:32:37: Verifying 'xp\netstat.exe' OK

(md5=8A7EE413726790398D6B315B7CFB5B0A)

06:32:37: Running 'xp\netstat.exe' [#77/169]

COMPLETE

'netstatn.txt'

(md5=52EE5EEACDFD2BA082F52D9CFEA10708)

'netstatn.htm'

(md5=9329D66D41228831C1408A1F255DB6A4)

06:32:38: Verifying 'foundstone\fport.exe' OK

(md5=DBB75488AA2FA22BA6950AEAD1EF30D5)

06:32:38: Running 'foundstone\fport.exe' [#78/169]

COMPLETE

'fport_p.txt'

(md5=9FC5FD6A033D6B3617840ECF63DED76D)

'fport_p.htm'

(md5=BB5127D25439D1B50A305B1CF865FE7A)

06:32:38: Verifying 'foundstone\fport.exe' OK

(md5=DBB75488AA2FA22BA6950AEAD1EF30D5)

06:32:38: Running 'foundstone\fport.exe' [#79/169]

COMPLETE

'fport_a.txt'

(md5=9FC5FD6A033D6B3617840ECF63DED76D)

'fport_a.htm'

(md5=4753417C7C8129D12755395560F24D19)

06:32:39: Verifying 'xp\..\diamondcs\openports.exe' OK

(md5=A9373778D62495E1C537AD12AB2EFA3D)

06:32:39: Running 'xp\..\diamondcs\openports.exe' [#82/169]

COMPLETE

'openports.txt'

(md5=4D06B65D83F276728E817C02FD95C40C)

'openports.htm'

(md5=1658C56BA5517E2BCB730FC20093A888)

06:32:39: Verifying 'xp\ipxroute.exe' OK

(md5=5A3EB34CC341E09A55CEDB01F0D901B2)

06:32:39: Running 'xp\ipxroute.exe' [#85/169]

SKIPPED (via '-noslow' parameter)

'ipxroute.txt'

(md5=581C52AB1684E14B67BDB45C4528F992)

'ipxroute.htm'

(md5=97D4E1DD9621B68E55C1E7B292462634)

06:32:39: Verifying 'xp\nbtstat.exe' OK

(md5=4827686E0DAE2C302B02578DF7941B15)

06:32:39: Running 'xp\nbtstat.exe' [#86/169]

COMPLETE

'nbtstatn.txt'

(md5=3DC9C248379C160466F583FC26752962)

'nbtstatn.htm'

(md5=452DFAC476BD1E13A9F03A64C2DD6025)

06:32:39: Verifying 'xp\nbtstat.exe' OK

(md5=4827686E0DAE2C302B02578DF7941B15)

06:32:39: Running 'xp\nbtstat.exe' [#87/169]

COMPLETE

'nbtstatc.txt'

(md5=3DC9C248379C160466F583FC26752962)

'nbtstatc.htm'

(md5=7A39093E22B2C228AB8D555D9B4C753F)

06:32:39: Verifying 'xp\nbtstat.exe' OK

(md5=4827686E0DAE2C302B02578DF7941B15)

06:32:39: Running 'xp\nbtstat.exe' [#88/169]

COMPLETE

'nbtstats.txt'

(md5=647578D95B7F3B11B6CCF833E296701E)

'nbtstats.htm'

(md5=AC4DEF77CA515CEAB894EE19D3E8B863)

06:32:40: Verifying 'foundstone\hunt.exe' OK

(md5=356C4848D606D55BBAA7C7075F5A6E1C)

06:32:40: Running 'foundstone\hunt.exe' [#89/169]

NOTE: Hunt sometimes fails given an IP address

COMPLETE

'hunt.txt'

(md5=99553CCD8E14CAE6DB789BCE98CFCB73)

'hunt.htm'

(md5=D59D46C7DA7BED9E79C68A4C5F7BDFE8)

06:32:40: Verifying 'xp\net.exe' OK

(md5=FD3DA8425624B98903407DF608CF2C11)

06:32:40: Running 'xp\net.exe' [#90/169]

COMPLETE

'netshare.txt'

(md5=32F60215D6F7CE348C66FE6D6AFB7081)

'netshare.htm'

(md5=52AAE9B29695344DA8DDC0F0DCE05DC4)

06:32:40: Verifying 'xp\net.exe' OK

(md5=FD3DA8425624B98903407DF608CF2C11)

06:32:40: Running 'xp\net.exe' [#91/169]

COMPLETE

'netuse.txt'

(md5=9D8780AF2919A4F1B1532C208720ED14)

'netuse.htm'

(md5=3F296F71EDA5A090A372B8C6FDF3E6E8)

06:32:40: Verifying 'xp\net.exe' OK

(md5=FD3DA8425624B98903407DF608CF2C11)

06:32:40: Running 'xp\net.exe' [#92/169]

COMPLETE

'netview.txt'

(md5=768165E0ABF16BF3056836D5431A7296)

'netview.htm'

(md5=CFCE3727BED1EC8C7294D014BD400AE9)

06:32:41: Verifying 'xp\net.exe' OK

(md5=FD3DA8425624B98903407DF608CF2C11)

06:32:41: Running 'xp\net.exe' [#93/169]

COMPLETE

'netsessi.txt'

(md5=768165E0ABF16BF3056836D5431A7296)

'netsessi.htm'

(md5=A4922FA39F9FC802E373CC969E6D7635)

06:32:41: Verifying 'perl\p2x588.dll' OK

(md5=A236999C6D5CF814CD562CFAB4BAB8FA)

06:32:41: Verifying 'perl\re.dll' OK

(md5=6239608ECF09DE1BA88D21635C9C1C6A)

06:32:41: Verifying 'perl\util.dll' OK

(md5=6DBEA3D86C51BF65ECBDAF0A91EE5FEA)

06:32:41: Verifying 'perl\ole.dll' OK

(md5=08FF61A0FC0509D11D3DD4DDE750311A)

06:32:41: Verifying 'perl\cwd.dll' OK

(md5=4051A8FF7E38F23C037BCE5378B714AD)

06:32:41: Verifying 'perl\ndis.exe' OK

(md5=BD49690B71C750EA8D9B3B7E784FD897)

06:32:41: Running 'perl\ndis.exe' [#94/169]

COMPLETE

'ndis.txt'

(md5=6D4F36367CB6120BA92ADAA77DA42BEA)

'ndis.htm'

(md5=3DF33FCD8526FE114BBC1FE420CDBD83)

06:32:42: Verifying 'ntsecurity\promiscdetect.exe' OK

(md5=D41D8CD98F00B204E9800998ECF8427E)

06:32:42: Running 'ntsecurity\promiscdetect.exe' [#95/169]

COMPLETE

'promiscdetect.txt'

(md5=E9B3135E5CD46F2E434E67A0DE631F95)

'promiscdetect.htm'

(md5=221BC9B76C231FB1E947E779F863A02B)

[LOGINS]

06:33:09: Verifying 'sysinternals\psloggedon.exe' OK
(md5=6500C15F856BBFD0B28BD4EBF6E1662A)
06:33:12: Running 'sysinternals\psloggedon.exe' [#96/169]
SKIPPED (via '-nowrite' parameter)
'psloggedon.txt'
(md5=B2E118703DF65508E00E4C17C7651AB9)
'psloggedon.htm'
(md5=AE76742479D2F0406C008C2D58A464B6)

06:33:12: Verifying 'systemtools\netusers.exe' OK
(md5=B01A4D9FBB85B387DD890FCA73C212D3)
06:33:12: Running 'systemtools\netusers.exe' [#97/169]
COMPLETE
'netusers_local.txt'
(md5=AB20A9482744E11727C791BC855717B8)
'netusers_local.htm'
(md5=6B21AF94AB439F4B9853A44CC3F95AAF)

06:33:13: Verifying 'systemtools\netusers.exe' OK
(md5=B01A4D9FBB85B387DD890FCA73C212D3)
06:33:13: Running 'systemtools\netusers.exe' [#98/169]
COMPLETE
'netusers_local_history.txt'

(md5=C4DA0F6D85F641B3E5A0F065AEDB2701)

'netusers_local_history.htm'

(md5=00D8BDF28C0673483F39E2BF4B010F99)

06:33:13: Verifying 'foundstone\ntlast.exe' OK

(md5=1128A558328023F6006327570C4D201F)

06:33:13: Running 'foundstone\ntlast.exe' [#99/169]

COMPLETE

'success.txt'

(md5=2428169B31C96FE4A767215000AA0008)

'success.htm'

(md5=5DA3161760F7B453690885D31F203B82)

06:33:13: Verifying 'foundstone\ntlast.exe' OK

(md5=1128A558328023F6006327570C4D201F)

06:33:13: Running 'foundstone\ntlast.exe' [#100/169]

COMPLETE

'failed.txt'

(md5=2428169B31C96FE4A767215000AA0008)

'failed.htm'

(md5=3EB913F1254157D414366B076FD49A23)

06:33:14: Verifying 'foundstone\ntlast.exe' OK

(md5=1128A558328023F6006327570C4D201F)

06:33:14: Running 'foundstone\ntlast.exe' [#101/169]

COMPLETE

'interact.txt'

(md5=2428169B31C96FE4A767215000AA0008)

'interact.htm'

(md5=F2FA3FB08A91DDFD6F00B5343B4F4483)

06:33:14: Verifying 'foundstone\ntlast.exe' OK

(md5=1128A558328023F6006327570C4D201F)

06:33:14: Running 'foundstone\ntlast.exe' [#102/169]

COMPLETE

'remote.txt'

(md5=2428169B31C96FE4A767215000AA0008)

'remote.htm'

(md5=0938AACD71A620B109690296D51348BA)

[EVENT LOGS]

06:33:14: Verifying '2k\res_kit\dumpel.exe' OK

(md5=3C2447E278318C4E6B50D5095E7028E5)

06:33:14: Running '2k\res_kit\dumpel.exe' [#103/169]

COMPLETE

'syslog.txt'

(md5=7583EE9E2E28231E98A616966696F439)

'syslog.htm'

(md5=B3E81A2D1F78BB58278986BD03187DF4)

06:33:15: Verifying '2k\res_kit\dumpel.exe' OK
(md5=3C2447E278318C4E6B50D5095E7028E5)
06:33:15: Running '2k\res_kit\dumpel.exe' [#104/169]
COMPLETE
'applog.txt'
(md5=4A25F00467D15FE9FAE2777769518518)
'applog.htm'
(md5=96898F0D4EB344CD6F15C91EAA929FA7)

06:33:18: Verifying '2k\res_kit\dumpel.exe' OK
(md5=3C2447E278318C4E6B50D5095E7028E5)
06:33:18: Running '2k\res_kit\dumpel.exe' [#105/169]
COMPLETE
'seclog.txt'
(md5=D41D8CD98F00B204E9800998ECF8427E)
'seclog.htm'
(md5=276339791A0E9DBDE562D7B474DDDB8E)

06:33:18: Verifying 'sysinternals\psloglist.exe' OK
(md5=BC4F65879BE48DE9A7C2AA075248D846)
06:33:18: Running 'sysinternals\psloglist.exe' [#106/169]
SKIPPED (via '-nowrite' parameter)
'evtlog.txt'
(md5=D7F5E4736E3C7E2BC580E63814B73A2C)
'evtlog.htm'
(md5=C3AA752B275FE1720B351E0F7D74F351)

06:33:18: Verifying 'sysinternals\psloglist.exe' OK
(md5=BC4F65879BE48DE9A7C2AA075248D846)
06:33:18: Running 'sysinternals\psloglist.exe' [#107/169]
SKIPPED (via '-nowrite' parameter)
'log_sys.txt'
(md5=D7F5E4736E3C7E2BC580E63814B73A2C)
'log_sys.htm'
(md5=2B7A2354330161E84E102E2A3A91FFBF)

06:33:18: Verifying 'sysinternals\psloglist.exe' OK
(md5=BC4F65879BE48DE9A7C2AA075248D846)
06:33:18: Running 'sysinternals\psloglist.exe' [#108/169]
SKIPPED (via '-nowrite' parameter)
'log_app.txt'
(md5=D7F5E4736E3C7E2BC580E63814B73A2C)
'log_app.htm'
(md5=A166009966815B486C1660D6924BA01F)

06:33:18: Verifying 'sysinternals\psloglist.exe' OK
(md5=BC4F65879BE48DE9A7C2AA075248D846)
06:33:18: Running 'sysinternals\psloglist.exe' [#109/169]
SKIPPED (via '-nowrite' parameter)
'log_Sec.txt'
(md5=D7F5E4736E3C7E2BC580E63814B73A2C)
'log_Sec.htm'

(md5=E9D558E387FC220866B3DA9826ACB474)

[FILE SYSTEM]

06:33:18: Verifying 'sysinternals\ntfsinfo.exe' OK
(md5=C8F3A485E66CBCD3731B22991C85002E)
06:33:18: Running 'sysinternals\ntfsinfo.exe' [#110/169]
SKIPPED (via '-nowrite' parameter)
'C_ntfsinfo.txt'
(md5=E9F22F0DA88CAC4BC36566D480906FD4)
'C_ntfsinfo.htm'
(md5=CC43BF083BBAC6E06DC7642C228CA559)

06:33:18: Verifying 'sysinternals\psfile.exe' OK
(md5=CB623488009F084EC53CB62E45CBCF72)
06:33:18: Running 'sysinternals\psfile.exe' [#111/169]
SKIPPED (via '-nowrite' parameter)
'psfile.txt'
(md5=799CA951BC9BDDD731DA227C9D2A0902)
'psfile.htm'
(md5=D6D58AD5914EF24B8E71AB3AA036CD3E)

06:33:18: Verifying 'xp\net.exe' OK
(md5=FD3DA8425624B98903407DF608CF2C11)
06:33:18: Running 'xp\net.exe' [#112/169]

COMPLETE

'netfile.txt'

(md5=768165E0ABF16BF3056836D5431A7296)

'netfile.htm'

(md5=76B383A9ED49CBB0A30AF9C4C7E19642)

06:33:19: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:33:19: Running 'tools\xp\cmd.exe' [#113/169]

SKIPPED (via '-noslow' parameter)

'C_filestg.txt'

(md5=5F45B5367C7ED885270EA969851CB439)

'C_filestg.htm'

(md5=BBB3AFAA88F73986921AB8AF42FA163E)

06:33:19: Verifying 'foundstone\hfind.exe' OK

(md5=E490103DC92D1F0F840848691A99AD96)

06:33:19: Running 'foundstone\hfind.exe' [#114/169]

SKIPPED (via '-noslow' parameter)

'C_hfind.txt'

(md5=413B8EAC24A155C702221CF89581E55C)

'C_hfind.htm'

(md5=6F2CA064A84A605E699BD2BED21689F3)

06:33:19: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:33:19: Running 'tools\xp\cmd.exe' [#115/169]

SKIPPED (via '-noslow' parameter)

'C_hidden.txt'

(md5=5F45B5367C7ED885270EA969851CB439)

'C_hidden.htm'

(md5=7AF8D6E75FFD1C6E6ED7B6D5B350B7D1)

06:33:19: Verifying 'sysinternals\streams.exe' OK

(md5=9C2318174AFD1535B1F33AFE3D34A255)

06:33:19: Running 'sysinternals\streams.exe' [#116/169]

SKIPPED (via '-nowrite' parameter)

'C_streams.txt'

(md5=0BDBFA670C99DEEF38FB86A21B39A9E6)

'C_streams.htm'

(md5=DDAE288F7AD3FA459B77D1580A453782)

06:33:19: Verifying 'foundstone\sfind.exe' OK

(md5=C16EA9CADC953125B86FFF45505DBAA6)

06:33:19: Running 'foundstone\sfind.exe' [#117/169]

SKIPPED (via '-noslow' parameter)

'C_sfind.txt'

(md5=B5BA059D7BFD0F4E4D005CD2E9E497B3)

'C_sfind.htm'

(md5=752BCD075879AB05E582C2BAFBD1019A)

06:33:19: Verifying '2k\res_kit\efsinfo.exe' OK

(md5=238EB692A776AE1DC038BC6A0CC2E87E)

06:33:19: Running '2k\res_kit\efsinfo.exe' [#118/169]

SKIPPED (via '-noslow' parameter)

'C_efsinfo.txt'

(md5=C218C5009550EFFCC9C8332EF65B6302)

'C_efsinfo.htm'

(md5=8F8A235B3C090CD0F72C2B2EB21BE226)

06:33:19: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:33:19: Running 'tools\xp\cmd.exe' [#119/169]

COMPLETE

'recent.txt'

(md5=877A6DF5F576B096D28D049AE2D1645C)

'recent.htm'

(md5=C27313FE31892EEE745EE95A6078A79D)

06:33:20: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:33:20: Running 'tools\xp\cmd.exe' [#120/169]

COMPLETE

'C_recycle.txt'

(md5=8609E8E78D5E28A8DB50BA4457383F6D)

'C_recycle.htm'

(md5=F10EFEF39D9E768E3B3BA80DCC4F792B)

06:33:20: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:33:20: Running 'tools\xp\cmd.exe' [#121/169]

COMPLETE

'prefetch.txt'

(md5=11B255D75EE6393D6CD71BF1879C3488)

'prefetch.htm'

(md5=C4B22C721EC1AA70314C108BB0924B18)

06:33:20: Verifying 'netlatency\freespace.exe' OK

(md5=136BE49B6BD168E07E8078ED1A0F2611)

06:33:20: Running 'netlatency\freespace.exe' [#122/169]

COMPLETE

'C_freesp.txt'

(md5=620AD01A32B3E6F4BC54365E576E9122)

'C_freesp.htm'

(md5=651BCCC73A3EC0C715EE2B915F68EE9D)

[AUTO START]

06:33:20: Verifying 'sysinternals\autorunsc.exe' OK

(md5=F0AAB3D6FCA0C5883BD198224CF6D646)

06:33:20: Running 'sysinternals\autorunsc.exe' [#123/169]

SKIPPED (via '-nowrite' parameter)

'autoruns.txt'

(md5=88996E636C7AFDF72301FBBC98382608)

'autoruns.htm'

(md5=F5B2C94DF50B21F376EE5A23A05FCCA5)

06:33:20: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:33:20: Running 'tools\xp\cmd.exe' [#124/169]

COMPLETE

'autoexec.txt'

(md5=D41D8CD98F00B204E9800998ECF8427E)

'autoexec.htm'

(md5=59206AFEB091ABBEC7BD1C8F54E575A7)

06:33:21: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:33:21: Running 'tools\xp\cmd.exe' [#125/169]

COMPLETE

'win_ini.txt'

(md5=8C1DFF16CD151B49143C48796BB750FD)

'win_ini.htm'

(md5=5AB1A502B1FC79402851CCBBB7B4ED4A)

06:33:21: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:33:21: Running 'tools\xp\cmd.exe' [#126/169]

COMPLETE

'sys_ini.txt'

(md5=B143A6852C9EF93E0BDECB02F524F9F2)

'sys_ini.htm'

(md5=B0404DB2B8AB003C5FE334D5573B5705)

06:33:21: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:33:21: Running 'tools\xp\cmd.exe' [#127/169]

COMPLETE

'winstart.txt'

(md5=DF5DC1ABC0D52F3C9E931E26A7C0065C)

'winstart.htm'

(md5=8A785CA570F0AED312216E01CE112B77)

06:33:21: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:33:21: Running 'tools\xp\cmd.exe' [#128/169]

COMPLETE

'init_ini.txt'

(md5=DF5DC1ABC0D52F3C9E931E26A7C0065C)

'init_ini.htm'

(md5=2FA3B3B46F122B085EE5115829D0239E)

06:33:21: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:33:21: Running 'tools\xp\cmd.exe' [#129/169]

COMPLETE

'startup.txt'

(md5=A220C7BD4ACE2153DE5E30EDA85C945B)

'startup.htm'

(md5=A907D696B5F75CA6D7B15AF5B854716C)

06:33:22: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:33:22: Running 'tools\xp\cmd.exe' [#130/169]

COMPLETE

'user_startup.txt'

(md5=C5B62B38F85A47009AC1362C007FF7A0)

'user_startup.htm'

(md5=C485E8C09304F8D231D1B11DD6C788A8)

06:33:22: Verifying 'tools\xp\cmd.exe' OK

(md5=6D778E0F95447E6546553EEEA709D03C)

06:33:22: Running 'tools\xp\cmd.exe' [#131/169]

COMPLETE

'tasks.txt'

(md5=597F96E64A39D161F9C79BEE54ACBB42)

'tasks.htm'

(md5=6E2D8754E8439627D257CC6A69CB5C45)

06:33:22: Verifying 'xp\at.exe' OK

(md5=FF90EEC0FAB78482BE97C5637FEF4090)

06:33:22: Running 'xp\at.exe' [#132/169]

COMPLETE

'at.txt'

(md5=90E2BE5857E337CBDBFFDE58B2E91C81)

'at.htm'

(md5=ABDAEA6C66C986D70C5540C686AB9AD8)

06:33:22: Verifying 'xp\schtasks.exe' OK

(md5=CF106F24787BA435A6BCA51A54C44193)

06:33:22: Running 'xp\schtasks.exe' [#134/169]

COMPLETE

'schtasks.txt'

(md5=FA00DDD2F8FE5C3A55E96BCCB6C87FFF)

'schtasks.htm'

(md5=899EEA950EDB07903D43DF49A3CA0496)

06:33:23: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:23: Running 'microsoft\reg.exe' [#136/169]

COMPLETE

'hklm_r.txt'

(md5=688BC52D6AA90F2204DC16E71B81C24E)

'hklm_r.htm'

(md5=BEBABAA2A860214D17D8304B65FF3590)

06:33:23: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:23: Running 'microsoft\reg.exe' [#137/169]

COMPLETE

'hklm_ro.txt'

(md5=992371965FAE1F03A17F5AAF5E6598BC)

'hklm_ro.htm'

(md5=A3F1D4F77F7AC928F288A0E50EAA6DFA)

06:33:23: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:23: Running 'microsoft\reg.exe' [#138/169]

COMPLETE

'hklm_rox.txt'

(md5=2222CCDED19CD28CF6B9C0432C445664)

'hklm_rox.htm'

(md5=261B895631D0DB0B501CF5B54884AE0C)

06:33:24: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:24: Running 'microsoft\reg.exe' [#139/169]

COMPLETE

'hklm_rs.txt'

(md5=629680BFDCB03EB78749D094FEBDBCB9)

'hklm_rs.htm'

(md5=F756A379DF8579C22CECAAC3FC75BBDD)

06:33:24: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:24: Running 'microsoft\reg.exe' [#140/169]

COMPLETE

'hklm_rso.txt'

(md5=629680BFDCB03EB78749D094FEBDBCB9)

'hklm_rso.htm'

(md5=0DE98750F47FAD1105788B69881792AD)

06:33:24: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:24: Running 'microsoft\reg.exe' [#141/169]

COMPLETE

'hklm_scripts.txt'

(md5=629680BFDCB03EB78749D094FEBDBCB9)

'hklm_scripts.htm'

(md5=C86779DBDF7E7A8B0C8362CAE86A05A1)

06:33:24: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:24: Running 'microsoft\reg.exe' [#142/169]

COMPLETE

'hklm_expl_run.txt'

(md5=629680BFDCB03EB78749D094FEBDBCB9)

'hklm_expl_run.htm'

(md5=3A3E1639160EB64837572D7ADADB329C)

06:33:24: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:24: Running 'microsoft\reg.exe' [#143/169]

COMPLETE

'hkcu_r.txt'

(md5=9E7B99199B2931E811AB482B833C2006)

'hkcu_r.htm'

(md5=89F4DE3FC5B222F63AA86624C6C87DA7)

06:33:24: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:24: Running 'microsoft\reg.exe' [#144/169]

COMPLETE

'hkcu_ro.txt'

(md5=992371965FAE1F03A17F5AAF5E6598BC)

'hkcu_ro.htm'

(md5=B096B4DB3A315E0870FD154CCC5E8553)

06:33:25: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:25: Running 'microsoft\reg.exe' [#145/169]

COMPLETE

'hkcu_rox.txt'

(md5=629680BFDCB03EB78749D094FEBDBCB9)

'hkcu_rox.htm'

(md5=4380D8F2014A5C35BF59DF30D079F0BB)

06:33:25: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:25: Running 'microsoft\reg.exe' [#146/169]

COMPLETE

'hkcu_rs.txt'

(md5=629680BFDCB03EB78749D094FEBDBCB9)

'hkcu_rs.htm'

(md5=77E38E4C9531A15D6EC4BA85A9A8380A)

06:33:25: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:25: Running 'microsoft\reg.exe' [#147/169]

COMPLETE

'hkcu_rso.txt'

(md5=629680BFDCB03EB78749D094FEBDBCB9)

'hkcu_rso.htm'

(md5=4B12E8F88BAC5D29408D8D527E7D6EB4)

06:33:25: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:25: Running 'microsoft\reg.exe' [#148/169]

COMPLETE

'hkcu_scripts.txt'

(md5=629680BFDCB03EB78749D094FEBDBCB9)

'hkcu_scripts.htm'

(md5=43ACF347D1BD6A412ED67A2D233161B6)

06:33:25: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:33:25: Running 'microsoft\reg.exe' [#149/169]

COMPLETE

'hkcu_expl_run.txt'

(md5=629680BFDCB03EB78749D094FEBDBCB9)

'hkcu_expl_run.htm'

(md5=B1F50E3968B3DD72474899A53A4EA544)

[REGISTRY]

06:33:26: Verifying 'ntsecurity\gplist.exe' OK

(md5=D41D8CD98F00B204E9800998ECF8427E)

06:33:26: Running 'ntsecurity\gplist.exe' [#150/169]

COMPLETE

'gplist.txt'

(md5=E9B3135E5CD46F2E434E67A0DE631F95)

'gplist.htm'

(md5=9C8299EF8518081FBE0FEF1C1BEF44CD)

06:34:09: Verifying 'xp\gpresult.exe' OK

(md5=1832FE014FF6B1FBB3A5E62E39218B82)

06:34:09: Running 'xp\gpresult.exe' [#151/169]

COMPLETE

'gpuser.txt'

(md5=96584E6AD16BE6C847166D809F70033A)

'gpuser.htm'

(md5=714BDB6449A71E9376F335785F42AE14)

06:34:14: Verifying 'xp\gpresult.exe' OK

(md5=1832FE014FF6B1FBB3A5E62E39218B82)

06:34:14: Running 'xp\gpresult.exe' [#152/169]

COMPLETE

'gpsys.txt'

(md5=C944FA3824417A3AE083C45473281BD7)

'gpsys.htm'

(md5=F3C86BECC5BD97408C6C98F105A9033D)

06:34:14: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:34:14: Running 'microsoft\reg.exe' [#153/169]

COMPLETE

'run_hist.txt'

(md5=D702985D915284E12B59EACB37A6F7A1)

'run_hist.htm'

(md5=203F83A77141742FA6577B28BE686953)

06:34:14: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:34:14: Running 'microsoft\reg.exe' [#154/169]

COMPLETE

'rcnt_doc.txt'

(md5=D51794C380B3AAA1435D0892A0F8C5CA)

'rcnt_doc.htm'

(md5=7A4E4FAC9EB93BDD1D20C17E032D8D75)

06:34:14: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:34:14: Running 'microsoft\reg.exe' [#155/169]

COMPLETE

'lastsave.txt'

(md5=82B262A274831C8AEEFF58322720484F)

'lastsave.htm'

(md5=C6271E4393D3A09D3AD0CD688358D660)

06:34:15: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:34:15: Running 'microsoft\reg.exe' [#156/169]

COMPLETE

'installh.txt'

(md5=49BCFC17BA427FA4B3EE188147DA0BEE)

'installh.htm'

(md5=DE48E6481CCEE6DABB54A59CBA945F3A)

06:34:15: Verifying 'microsoft\regdmp.exe' OK

(md5=BBC3AFA4512E90491787B3DA837C057F)

06:34:15: Running 'microsoft\regdmp.exe' [#157/169]

SKIPPED (via '-noslow' parameter)

'regdmp.txt'

(md5=44D3C46C824986693D8ACC7EAD3075C1)

'regdmp.htm'

(md5=51E651A9495AD2A0183D382068A3EDFB)

06:34:15: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:34:15: Running 'microsoft\reg.exe' [#158/169]

COMPLETE

'hklm_safemin.txt'

(md5=2B0BED64D2E3DAC4053F40CF2982B75C)

'hklm_safemin.htm'

(md5=21C76F57DC5D19A7EB353B6D563031CD)

06:34:15: Verifying 'microsoft\reg.exe' OK

(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:34:15: Running 'microsoft\reg.exe' [#159/169]

COMPLETE

'hklm_safenet.txt'

(md5=DD074F35C25931D4C193287358E5F594)

'hklm_safenet.htm'

(md5=5727676A6EA203A955B4FC1A4D14048F)

[IE ACTIVITY]

06:34:15: Verifying 'nirsoft\iehv.chm' OK
(md5=11437E75A4763768DFDFCF1819BA8B9B)
06:34:15: Verifying 'nirsoft\iehv.exe' OK
(md5=B2D5574738CB4E772A1B849695C19A2A)
06:34:15: Running 'nirsoft\iehv.exe' [#160/169]
COMPLETE
'iehv.txt'
(md5=062E3074795A47CF35DEB4E8197C3EE9)
'iehv.htm'
(md5=8B4E210BFA88F8426DBB9CB5EF424702)

06:34:16: Verifying 'microsoft\reg.exe' OK
(md5=C193E1D68A7921A6AEC298BA50A1AABA)
06:34:16: Running 'microsoft\reg.exe' [#161/169]
COMPLETE
'type_url.txt'
(md5=60623F0A8E30BE61CDE155C22626877B)
'type_url.htm'
(md5=76BE91706908E6636A4E91D099510087)

06:34:16: Verifying 'microsoft\reg.exe' OK
(md5=C193E1D68A7921A6AEC298BA50A1AABA)

06:34:16: Running 'microsoft\reg.exe' [#162/169]

COMPLETE

'search_h.txt'

(md5=7FE3BD462D6F904DDB033D2B1A93BB04)

'search_h.htm'

(md5=A618290A3DACF05978AD39EC6F2D742F)

06:34:16: Verifying 'ntsecurity\pstoreview.exe' OK

(md5=D41D8CD98F00B204E9800998ECF8427E)

06:34:16: Running 'ntsecurity\pstoreview.exe' [#163/169]

COMPLETE

'pstoreview.txt'

(md5=E9B3135E5CD46F2E434E67A0DE631F95)

'pstoreview.htm'

(md5=88D7BF3078BE7AF65B27EC3814B812AA)

[MISC]

06:34:25: Verifying 'xp\doskey.exe' OK

(md5=D7325A9F12013345434E1CC58B58D98C)

06:34:25: Running 'xp\doskey.exe' [#164/169]

COMPLETE

'doskey.txt'

(md5=D41D8CD98F00B204E9800998ECF8427E)

'doskey.htm'

(md5=1458731AC76BF6587B9ABF60B5E6F730)

06:34:26: Verifying 'perl\p2x588.dll' OK

(md5=A236999C6D5CF814CD562CFAB4BAB8FA)

06:34:26: Verifying 'perl\re.dll' OK

(md5=6239608ECF09DE1BA88D21635C9C1C6A)

06:34:26: Verifying 'perl\util.dll' OK

(md5=6DBEA3D86C51BF65ECBDAF0A91EE5FEA)

06:34:26: Verifying 'perl\ole.dll' OK

(md5=08FF61A0FC0509D11D3DD4DDE750311A)

06:34:26: Verifying 'perl\cwd.dll' OK

(md5=4051A8FF7E38F23C037BCE5378B714AD)

06:34:26: Verifying 'perl\mdm.exe' OK

(md5=E4B41C5B54F11DE96AE509894861E5E6)

06:34:26: Running 'perl\mdm.exe' [#165/169]

COMPLETE

'mdm.txt'

(md5=70EE549E2B10AF81209C3B57B8A6A517)

'mdm.htm'

(md5=A5E04CE095305295AE2D230D8A624A63)

06:34:26: Verifying 'xp\..\sysinternals\rootkitrevealer.chm' OK

(md5=F0F3E20B031C0C87586B8DA9020195E8)

06:34:27: Verifying 'xp\..\sysinternals\rootkitrevealer.exe' OK

(md5=EE738FE9BCDD605821002CEC8C7206DB)

06:34:27: Running 'xp\..\sysinternals\rootkitrevealer.exe' [#168/169]

SKIPPED (via '-nowrite' parameter)

'rootkit.txt'

(md5=BB67B3F5126A55A3DA7742791F825DF6)

'rootkit.htm'

(md5=DB2003EF0E28276F6A60FF1A8E7BAA8C)

[DONE]

06:34:27: Verifying '2k\res_kit\now.exe' OK

(md5=1CD2DF306E25FBDDF653A9D9B5DC8A41)

06:34:27: Running '2k\res_kit\now.exe' [#169/169]

COMPLETE

'end.txt'

(md5=A5F255DDBD261CFED9DCB16263317FF5)

'end.htm'

(md5=7C7181F3C4EAE5EAE77EA0BDF548A73A)

[WFT]

06:34:28: Hashing 'wft_cfg.txt'

(md5=1D378B80F8E0E7EA4F35EC8B92989CD1)

'wft_hash.txt'

(md5=C5E1F3D439DE301813665F1590D540E4)

06:37:03: Reporting 'index.htm'
(md5=46A56B4D790D32BD26DB6F43A813CE2B)
'wft_main.htm'
(md5=06024A3756B6A1C2594CA5FF643E3108)
'wft_head.htm'
(md5=D3FF0A02281099CE200B8CB3C44B0FCF)
'wft_menu.htm'
(md5=6BF1F7C4269C42DFC2CDC943084F7B4B)
'wft_cfg.htm'
(md5=DF567D7E0808AB01C83D7962AE056FB5)
'wft_hash.htm'
(md5=F1A040556A759801DEB5EB19D8E6DBB9)
'wft_help.htm'
(md5=E37C3B9E98C1B86646E56C7602539800)
'wft_tool.htm'
(md5=9DF253F6612F7BFE34C8830041155B4C)
'wft_link.htm'
(md5=8B51153EB5ED9AAD154BD8B1873873B1)
'wft_splash.htm'
(md5=75689FFB8E54FA562C2CB2F44051A1E2)
'wft_splash.js'
(md5=90CC848821AEFB7E11D01731175A22C4)
'fmlogo_sm.png'
(md5=51F885B6AB342A12AA6D050C556C0E3C)
'wft.ico'
(md5=C37E8B76F761570353FAC39EAB67AE7B)

'mail.gif'

(md5=38477E98FB3ED60B82906C7DCE8DA645)

'help.gif'

(md5=DABE723A558A1F324CF367C27EE30258)

'bad.gif'

(md5=DEBF875A10DB19E19A261B3F82ED3DAD)

'missing.gif'

(md5=DD83B55DC42DBFB48EC230CE4D8653EA)

'ok.gif'

(md5=A49085E88FE9D8C1C31928217E15DBC0)

'unknown.gif'

(md5=04108ACB26F82BB7C439D292EE7ABB9E)

==========

06:37:04: [RUN COMPLETE]

==========

Appendix 16: Documentation of the SQL Poisoning Attack

(An Example of a Pilot Experiment)

Step 1: After stabilizing the RFID Business System (as stated in Appendix 7), the fake RFID tag was injected with malicious code.

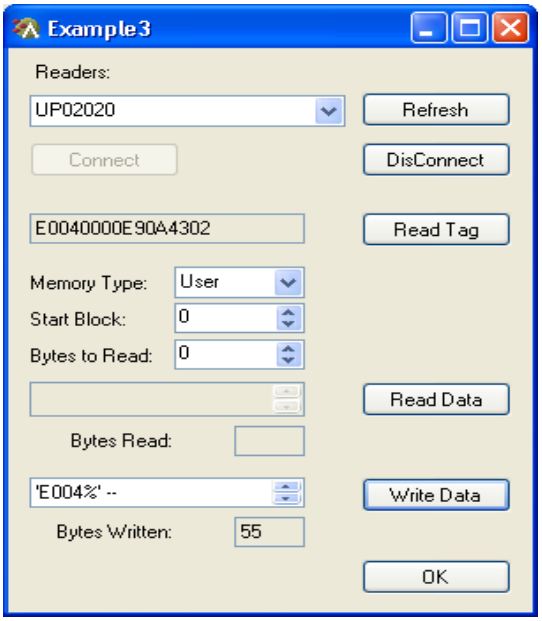


Figure A16.1: Writing the malicious code into the fake tag, successful!

Step 2: The genuine tag data was scanned a few times before the attack was initiated. Hence the values of all the products were $1000, as shown in the figure below.

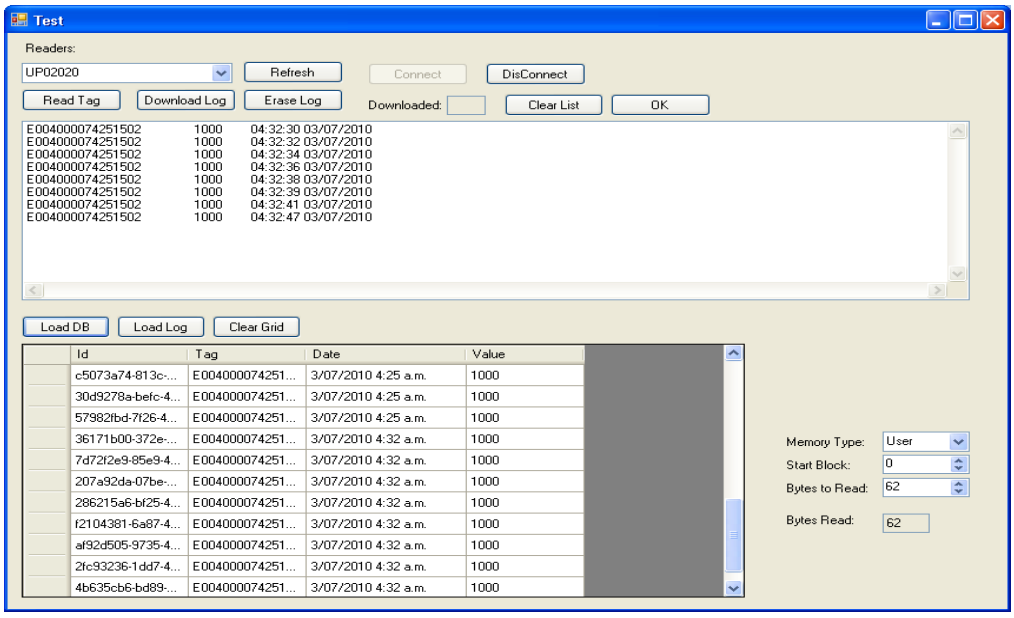


Figure A16.2: Scanning the genuine RFID tag

Step 3: The SQL poisoning attack was initiated by scanning a fake tag injected with RFID reader.

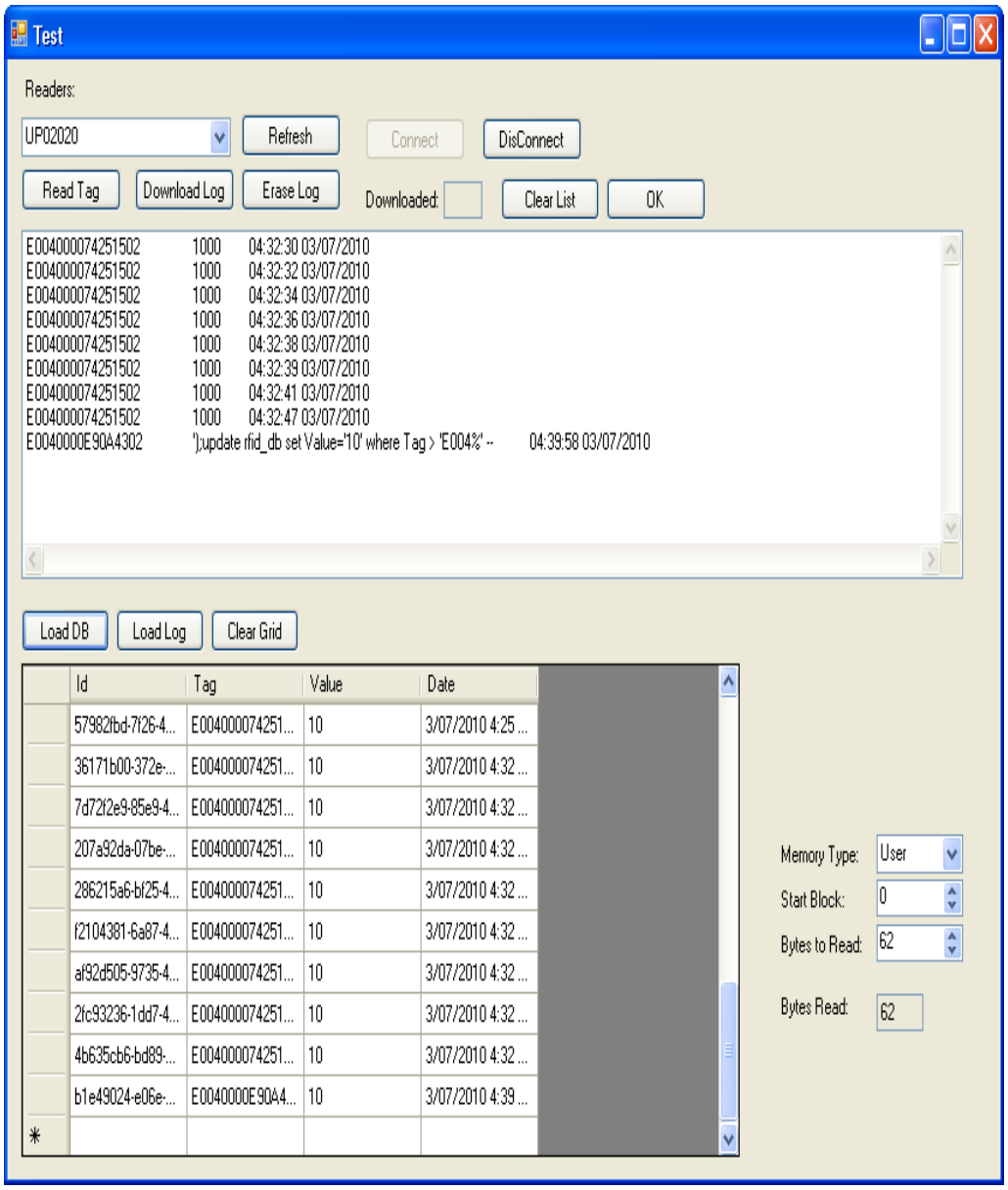


Figure A16.3: Scanning the fake RFID tag (an attack initiation)

Step 4: After scanning a fake tag injected with malicious code, the values were changed from 1000 to 10 dollars. Hence, the attack was successful and the injected malicious code was: ');update rfid_db set Value='10' where Tag > 'E004%' --

Step 5: Then, the genuine RFID tag was read a couple of times again (Figure A16.4) and the pilot testing of an attack to the backend server was completed.

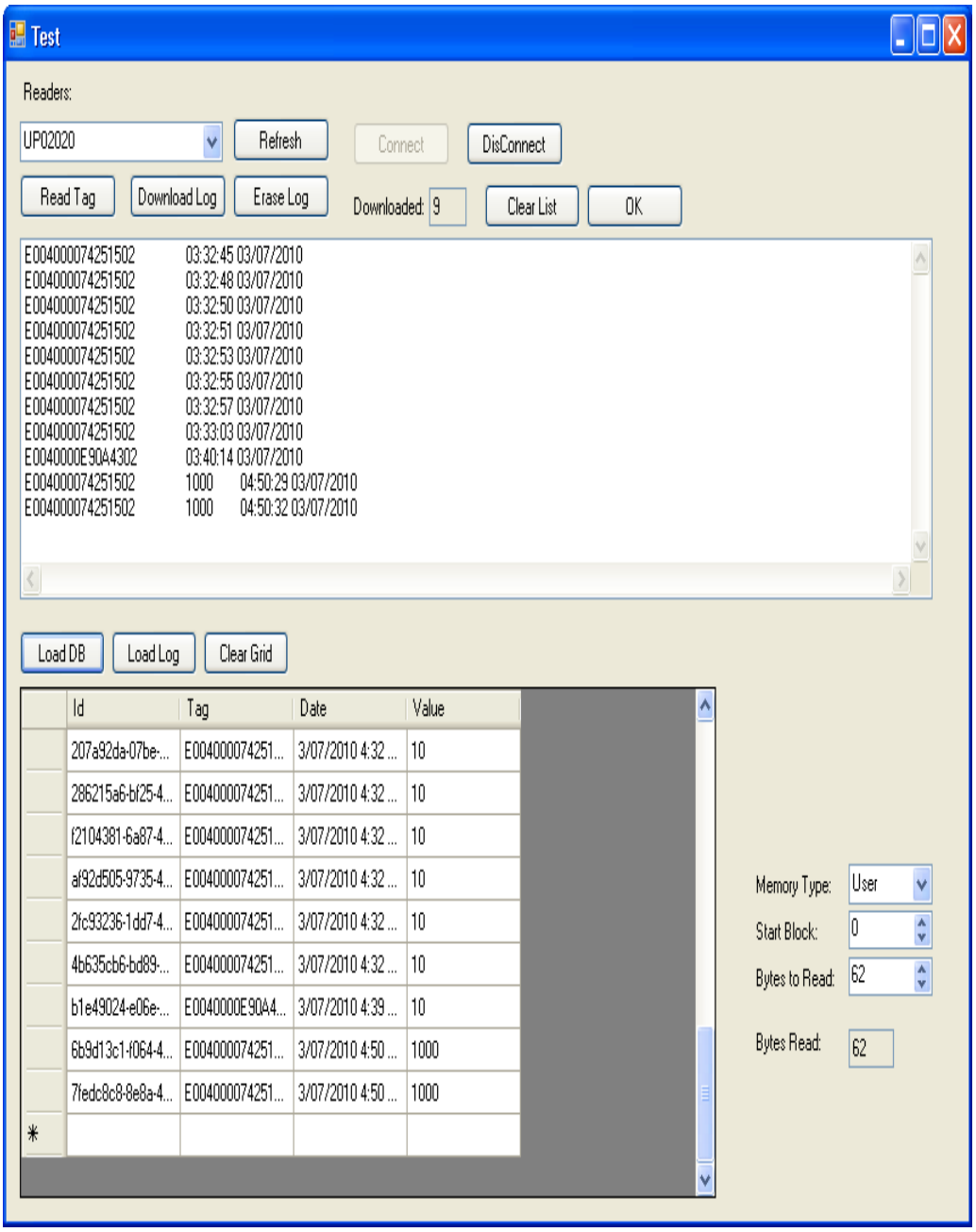


Figure A16.4: Writing original data back in the backend by scanning a genuine tag

Appendix 17: Live Acquisition of Physical Hard Drive

Step 1: The live acquisition was performed by using dd tool from customized Helix_RFID_IR tool (see Figure A17.1), after acquiring all the volatile artefacts of the RFID BS such as Reader's memory, POS memory, and SQL Server artefacts.

Figure A17.1: Physical drive acquisition by using dd tool

As seen the figure above, the options used in dd command included the conversion (conv=noerror), block size (default) and checksum calculation (MD5) and splitting the image (none).

Step 2: The physical drive image output (PhysicalDriveImage.dd), hash value calculation in MD5, and verification of the image were confirmed to save into the evidence acquisition drive (E:\PhysicalDriveImage\).

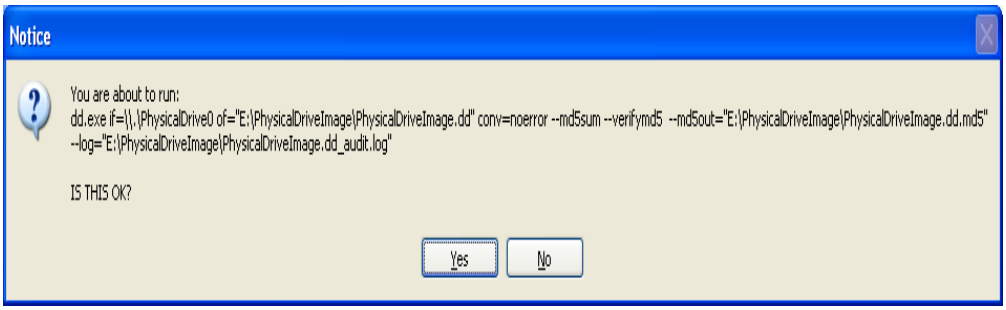


Figure A17.2: Notice of confirmation before running dd tool

Step 3: However, the problem was encountered during the acquisition process when the imaging got of the physical drive got to the processing status of about 50%.

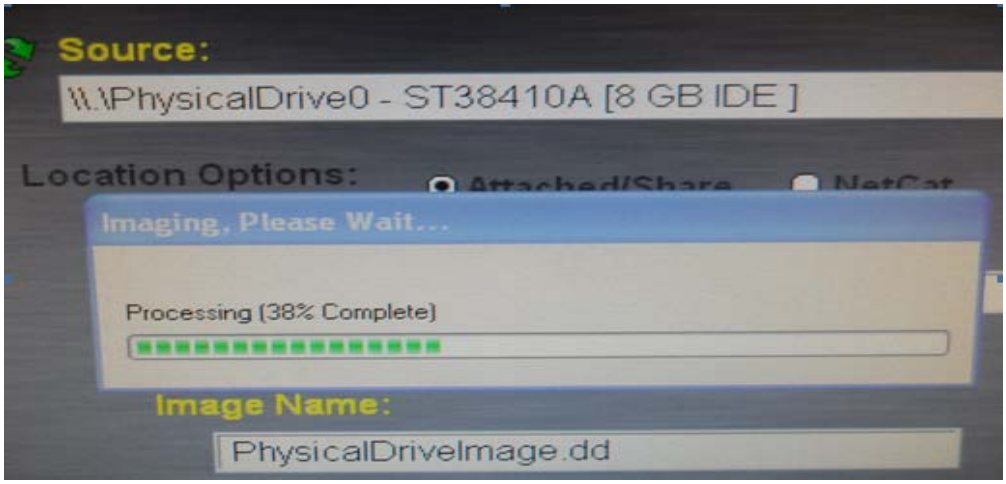


Figure A17.3: Acquisition in action by using dd tool

Step 4: Hence, the process of acquisition by using dd tool was terminated (see Figure A17.4).

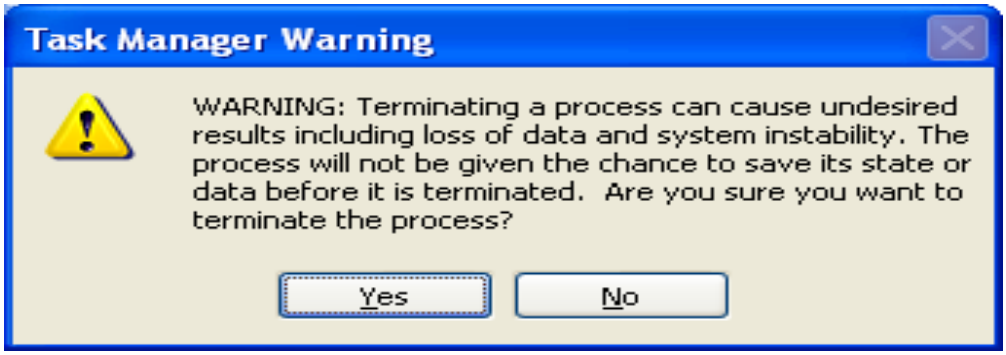


Figure A17.4: Terminating the acquisition process by dd tool

The termination was performed due to the fact that the destination drive (forensic acquisition drive – E:\) was FAT32 file system and could not save a single maximum file size of more than approximately 4GB while the capacity of acquired physical drive was 8 GB (detected by dd tool). The following was fragment information confirming an error of dd acquisition process.

```
Copying \\.\PhysicalDrive0 to E:\PhysicalDriveImage\PhysicalDriveImage.dd...
D:\IR\FAU\dd.exe:
        E:\PhysicalDriveImage\PhysicalDriveImage.dd: No space left on device
Output E:\PhysicalDriveImage\PhysicalDriveImage.dd (4294963200 bytes)
1048576+0 records in
1048575+0 records out
```

The problem issue was clearly stated that the evidence image should be split when acquiring the physical evidence of the victim's system (victim's hard drive) by using dd tool and the destination drive with FAT32 file system.

Step 4: Hence, the process of acquisition was resumed with AccessData's Forensic Tool Kit (FTK Imager Lite - version 2.9.0.1358). As the FTK Imager was not integrated in the RFID_IR DVD Toolkit, another customized forensic toolkit (USB Flash Drive F:\, that was loaded with FTK Imager, dcfldd, WinMD5 tools) was deployed from the trusted command prompt as shown in the following figures (Figure A17.5).

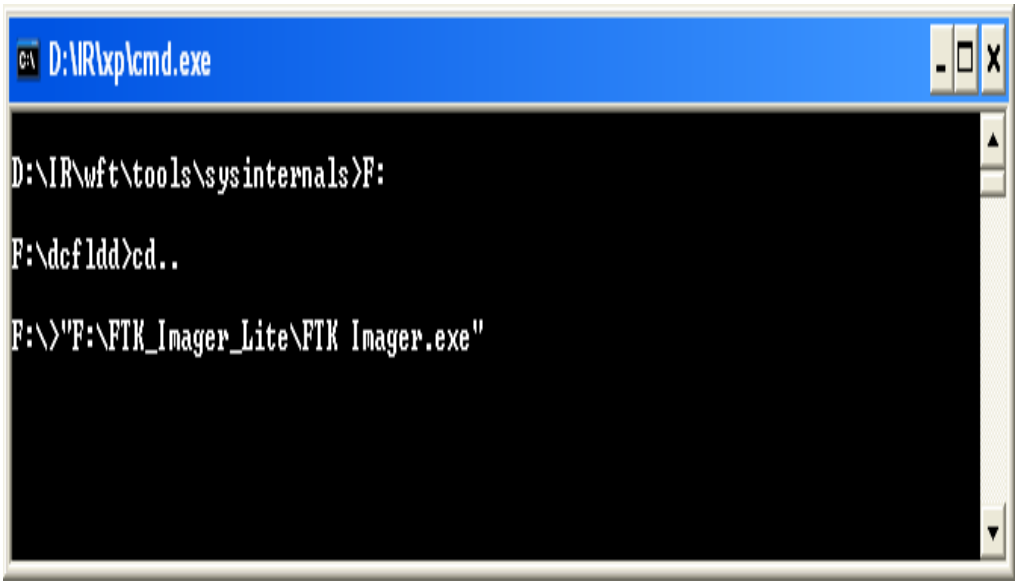


Figure A17.5: FTK Imager in action for the acquisition of physical drive

Step 5: Then, the source evidence type was chosen as physical drive after running FTK Imager (Figures A17.6 and A17.7).

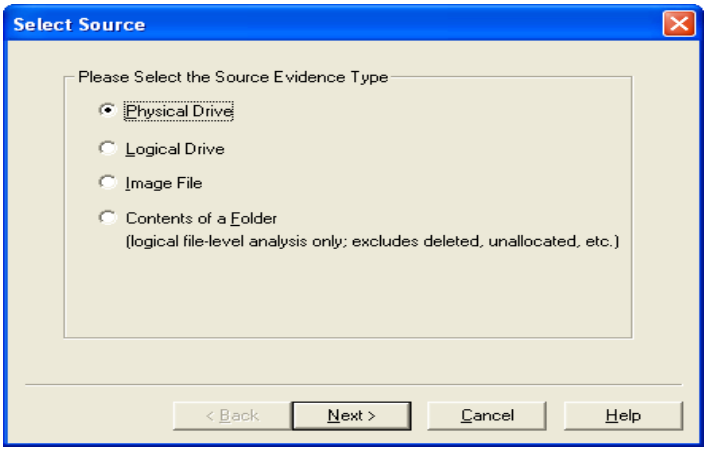


Figure A17.6: Selection of the source drive to acquire (part A)

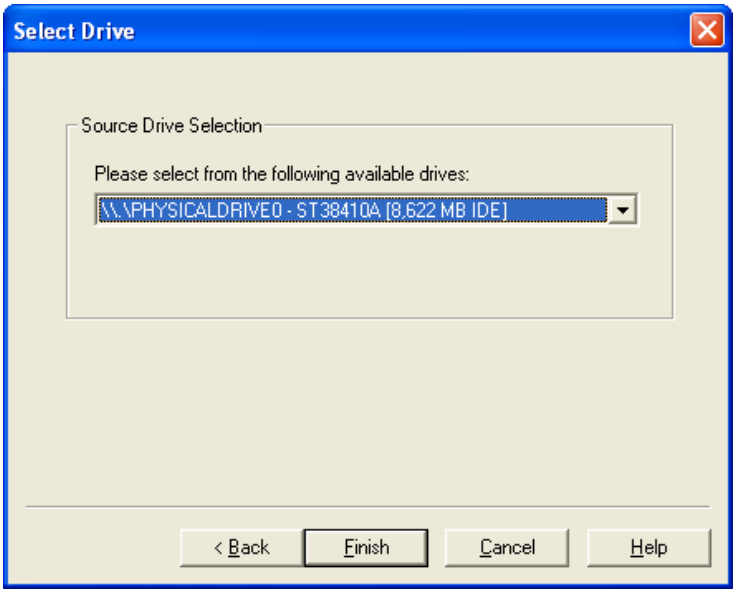


Figure A17.7: Selection of the source drive to acquire (part B)

Step 6: In order to analyse the forensic image with the Guidance Software's forensic tool (EnCase – Version 6.16.1.4), the E01 image type was chosen for the destination image type (Figure A17.8).

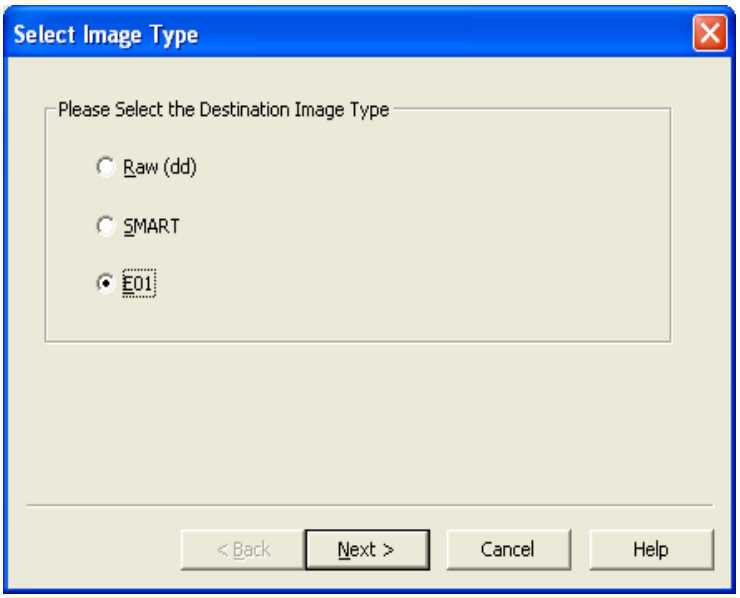


Figure A17.8: Selection of the destination image type

Step 7: The details of evidence item information such as the name of examiner, case and evidence numbers were recorded as shown in the figure below.

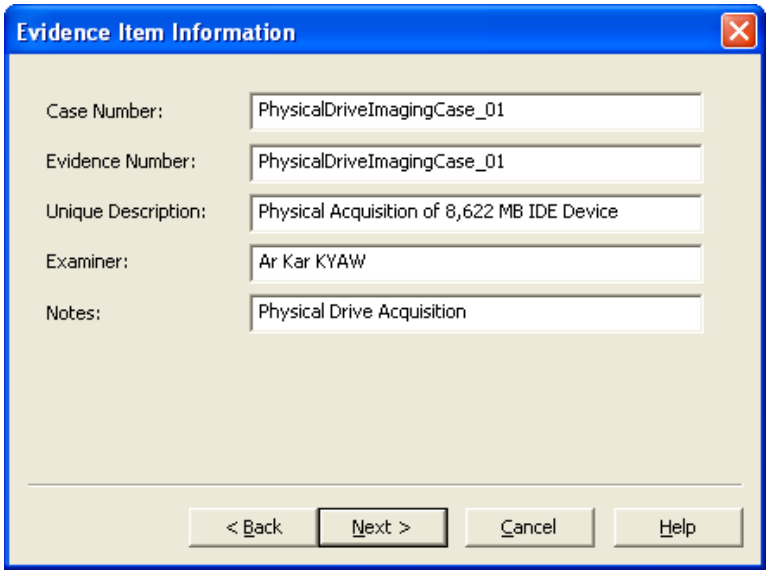


Figure A17.9: Evidence item information

Step 8: After clicking next button on the evidence item information dialog box, the details of the evidence image destination folder (E:\PhysicalDrive\Image), the evidence image file name, and the like were entered in the "Select Image Destination" box (see Figure A17.10).

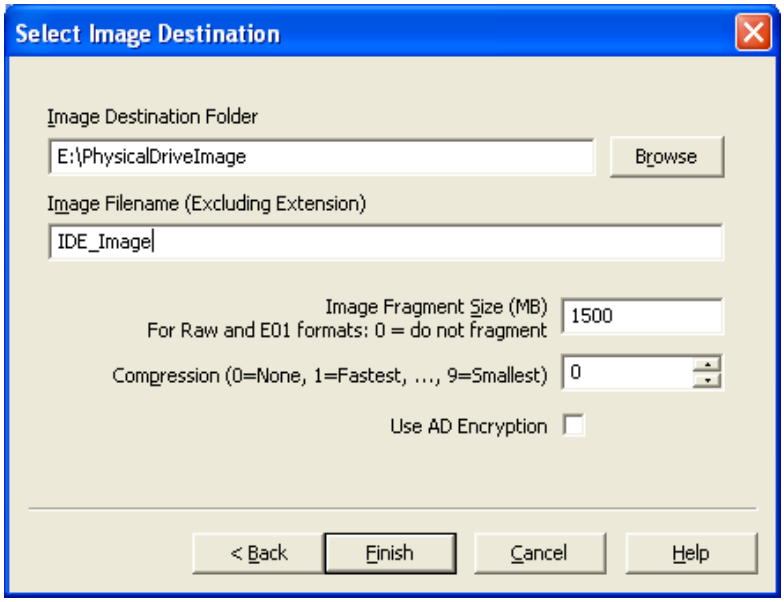


Figure A17.10: Selecting Image Destination

Step 9: Then, the final stage of configuration set up included choosing the options such as the verification of the evidence images after the creation, confirmation of the source and destination for the evidence images, and pre-calculation on the progress statistics (see Figure A17.11).

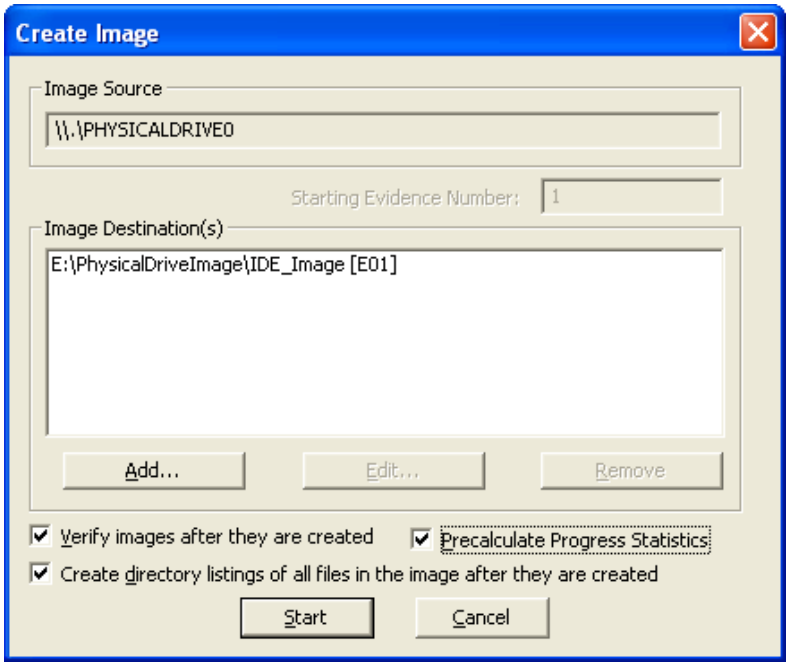


Figure A17.11: "Create Image" dialog box settings

Step 10: Then, the FTK Imager started creating directory listings of all files in the image after clicking "**Start**" button on the "Create Image" dialog box.

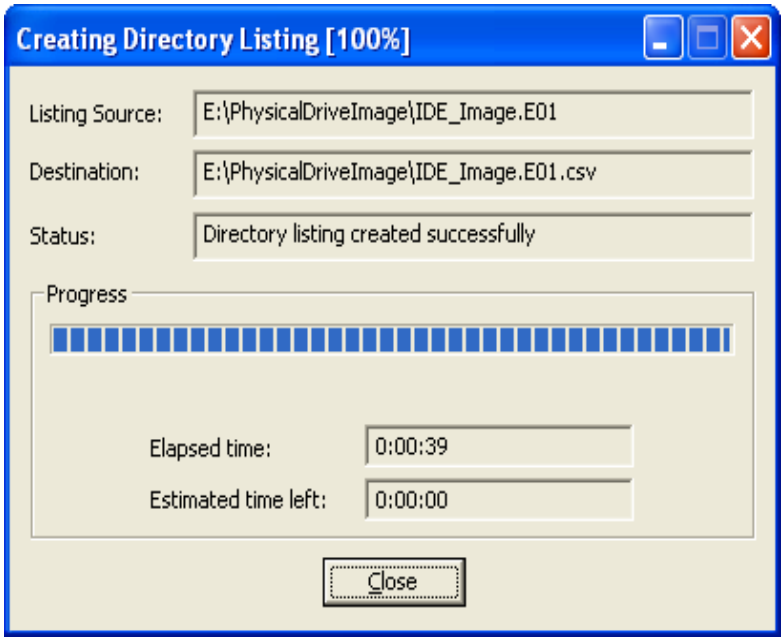


Figure A17.12: Creating directory listings of all files in the image

Step 11: The physical drive image was created in 22 minutes by using FTK imager (see Figure A17.13).

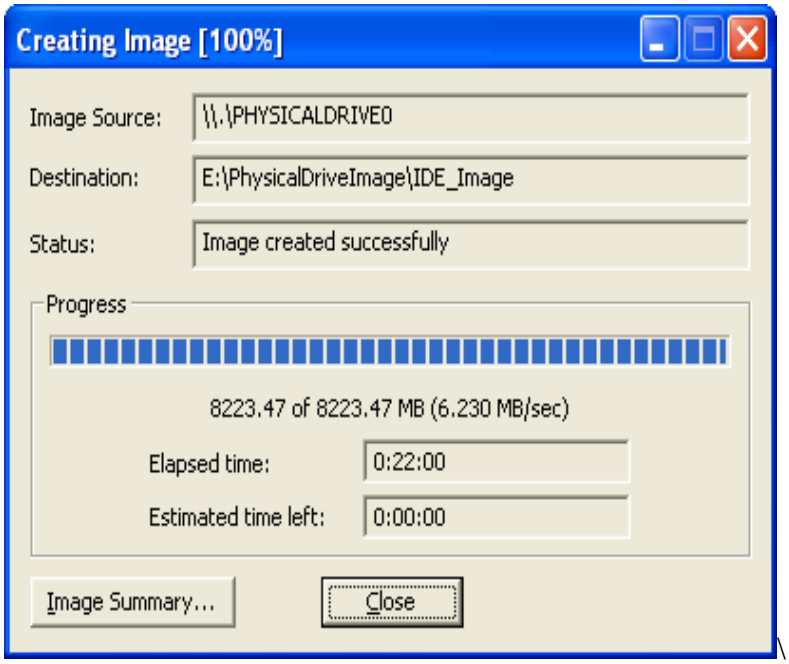


Figure A17.13: Successful creation of physical image

The image was segmented into six image segments and was successfully acquired into the forensic acquisition drive. The following is the information about physical image acquisition result, that is extracted from the final report of physical acquisition (IDE_Image.E01.txt) produced by FTK.

Image Information:
Acquisition started: Tue Oct 26 04:04:58 2010
Acquisition finished: Tue Oct 26 04:26:58 2010
Segment list:
E:\PhysicalDriveImage\IDE_Image.E01
E:\PhysicalDriveImage\IDE_Image.E02
E:\PhysicalDriveImage\IDE_Image.E03
E:\PhysicalDriveImage\IDE_Image.E04
E:\PhysicalDriveImage\IDE_Image.E05
E:\PhysicalDriveImage\IDE_Image.E06

Figure A17.14: Snippet of successful creation of physical image report

Step 12: Subsequently, the verification process in cryptographic hash functions such as MD5 Algorithm (128 bits) and Secure Hash Algorithm 1 (SHA1 - 160 bits) were done by comparing the stored verification hash and computed hash values during acquisition.

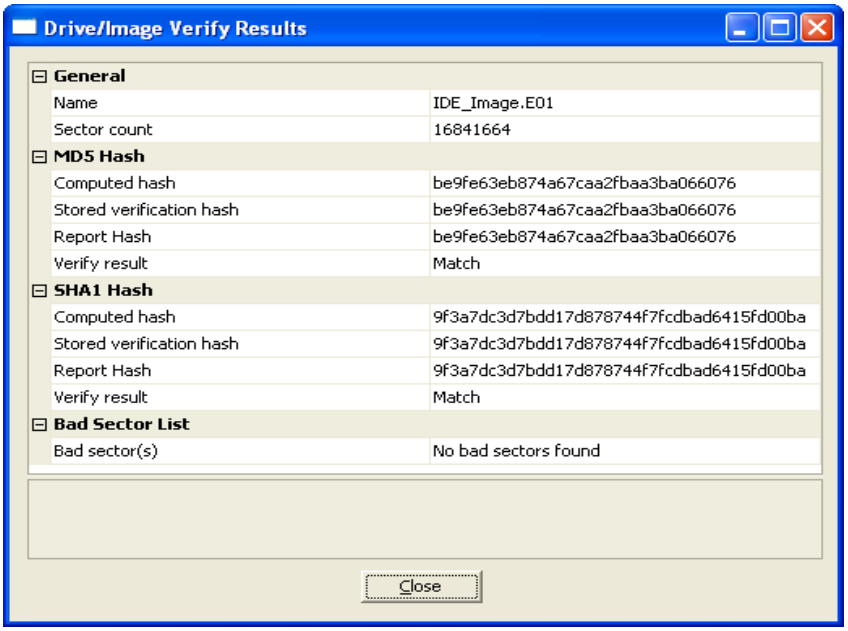


Figure A17.15: Drive or image verification results

The verification results showed no bad sectors and the hash values were the same (see Figure A17.15). Hence, the acquisition of the physical drive was done successfully without any error. The final report of physical acquisition (IDE_Image.E01.txt) produced by FTK could be seen in the following.

Physical Drive Acquisition Report by FTK Imager

Created By AccessData® FTK® Imager 2.9.0.1385 100406

Case Information:
Case Number: PhysicalDriveImagingCase_01
Evidence Number: PhysicalDriveImagingCase_01
Unique description: Physical Acquisition of 8,622 MB IDE Device
Examiner: Ar Kar KYAW
Notes: Physical Drive Acquisition

Information for E:\PhysicalDriveImage\IDE_Image:

Physical Evidentiary Item (Source) Information:
[Drive Geometry]
Cylinders: 1,048
Tracks per Cylinder: 255
Sectors per Track: 63
Bytes per Sector: 512
Sector Count: 16,841,664
[Physical Drive Information]
Drive Model: ST38410A
Drive Serial Number: 5CS078C3
Drive Interface Type: IDE
Source data size: 8223 MB
Sector count: 16841664
[Computed Hashes]
MD5 checksum: be9fe63eb874a67caa2fbaa3ba066076
SHA1 checksum: 9f3a7dc3d7bdd17d878744f7fcdbad6415fd00ba

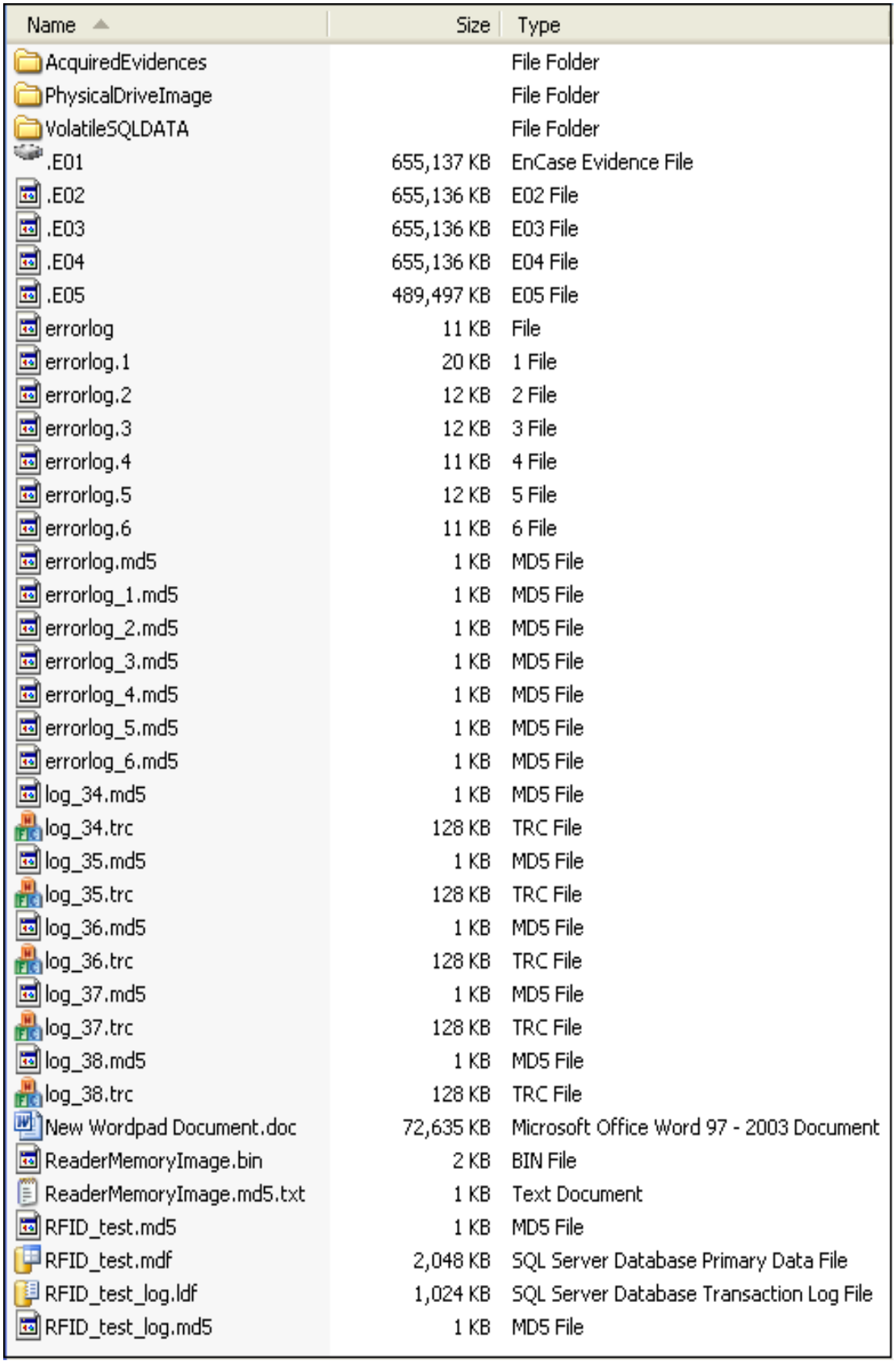
Image Information:
Acquisition started: Tue Oct 26 04:04:58 2010
Acquisition finished: Tue Oct 26 04:26:58 2010
Segment list:
E:\PhysicalDriveImage\IDE_Image.E01
E:\PhysicalDriveImage\IDE_Image.E02
E:\PhysicalDriveImage\IDE_Image.E03
E:\PhysicalDriveImage\IDE_Image.E04
E:\PhysicalDriveImage\IDE_Image.E05
E:\PhysicalDriveImage\IDE_Image.E06

Image Verification Results:
Verification started: Tue Oct 26 04:26:58 2010
Verification finished: Tue Oct 26 04:39:23 2010
MD5 checksum: be9fe63eb874a67caa2fbaa3ba066076 : verified
SHA1 checksum: 9f3a7dc3d7bdd17d878744f7fcdbad6415fd00ba : verified

Appendix 18: Screenshots of Image Copies of the Acquired Evidence Files from Forensic Workstation

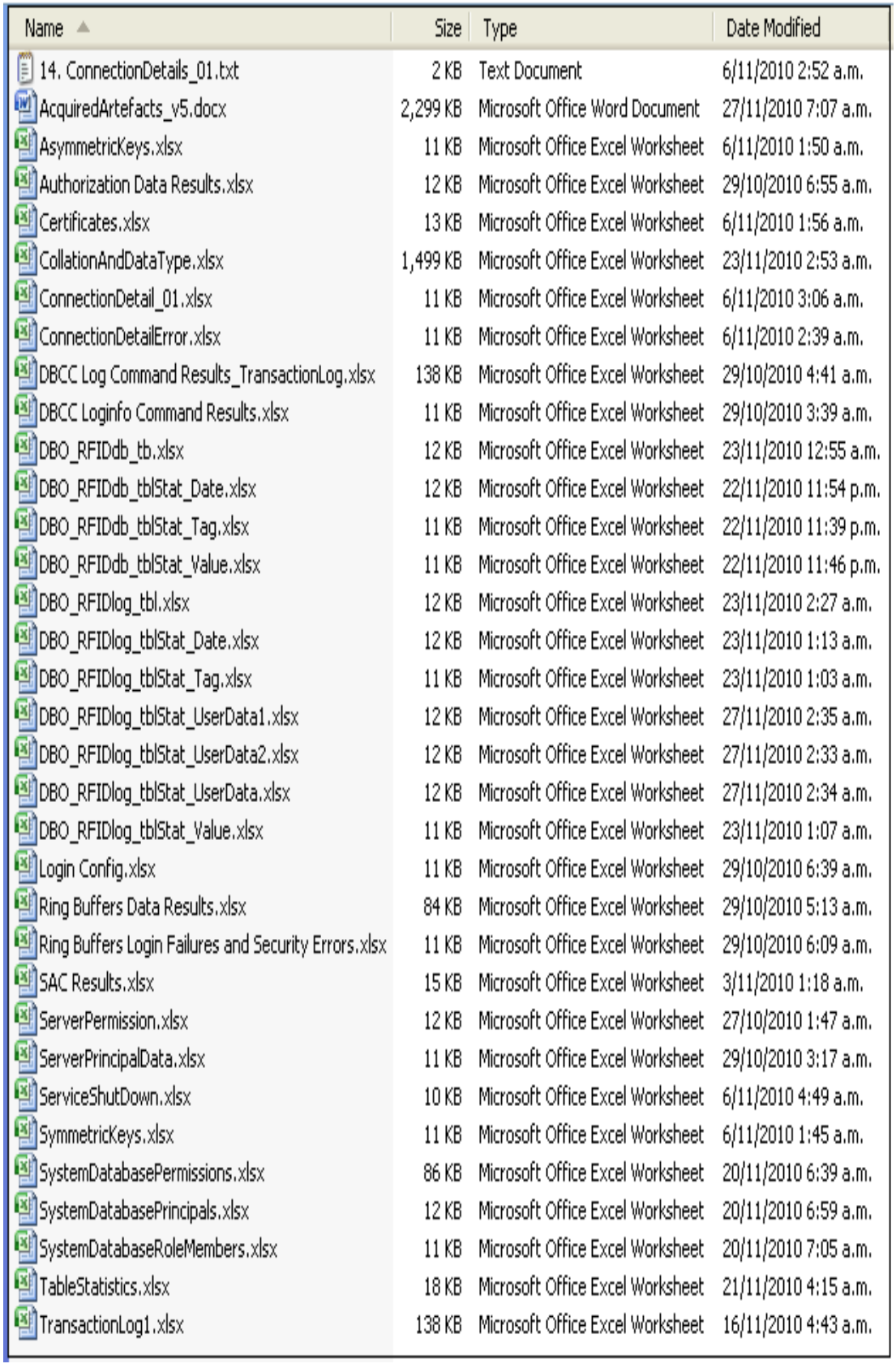
```
2010-10-25 12:37:04.13 Server      (c) 2005 Microsoft Corporation.
2010-10-25 12:37:04.13 Server      All rights reserved.
2010-10-25 12:37:04.13 Server      Server process ID is 1296.
2010-10-25 12:37:04.13 Server      Logging SQL Server messages in file 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG'.
2010-10-25 12:37:04.13 Server      This instance of SQL Server last reported using a process ID of 1508 at 10/25/2010 12:22:56 PM (local) 10/24/2010 11:22:56 PM (UTC).
2010-10-25 12:37:04.13 Server      Registry startup parameters:
2010-10-25 12:37:04.13 Server        -d C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\master.mdf
2010-10-25 12:37:04.13 Server        -e C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG
2010-10-25 12:37:04.13 Server        -l C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\mastlog.ldf
2010-10-25 12:37:04.15 Server      SQL Server is starting at normal priority base (=7). This is an informational message only. No user action is required.
2010-10-25 12:37:04.15 Server      Detected 2 CPUs. This is an informational message; no user action is required.
2010-10-25 12:37:04.51 Server      Using dynamic lock allocation.  Initial allocation of 2500 Lock blocks and 5000 Lock Owner blocks per node.  This is an informational
2010-10-25 12:37:04.51 Server      Database Mirroring Transport is disabled in the endpoint configuration.
2010-10-25 12:37:04.51 spid5s      Starting up database 'master'.
2010-10-25 12:37:04.67 spid5s      Recovery is writing a checkpoint in database 'master' (1). This is an informational message only. No user action is required.
2010-10-25 12:37:04.84 spid5s      SQL Trace ID 1 was started by login "sa".
2010-10-25 12:37:04.90 spid5s      Starting up database 'mssqlsystemresource'.
2010-10-25 12:37:05.35 spid8s      Starting up database 'model'.
2010-10-25 12:37:05.35 spid5s      Server name is 'TEST-STATION\SQLEXPRESS'. This is an informational message only. No user action is required.
2010-10-25 12:37:05.37 spid5s      Starting up database 'msdb'.
2010-10-25 12:37:05.56 Server      A self-generated certificate was successfully loaded for encryption.
2010-10-25 12:37:05.56 Server      Server is listening on [ 'any' <ipv4> 1069].
2010-10-25 12:37:05.56 Server      Server local connection provider is ready to accept connection on [ \\.\pipe\SQLLocal\SQLEXPRESS ].
2010-10-25 12:37:05.56 Server      Server local connection provider is ready to accept connection on [ \\.\pipe\MSSQL$SQLEXPRESS\sql\query ].
2010-10-25 12:37:05.56 Server      Dedicated administrator connection support was not started because it is not available on this edition of SQL Server. This is an info
2010-10-25 12:37:05.57 Server      The SQL Network Interface library could not register the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Failu
2010-10-25 12:37:05.57 Server      SQL Server is now ready for client connections. This is an informational message; no user action is required.
2010-10-25 12:37:06.23 spid8s      Clearing tempdb database.
2010-10-25 12:37:07.42 spid8s      Starting up database 'tempdb'.
2010-10-25 12:37:07.74 spid5s      Recovery is complete. This is an informational message only. No user action is required.
2010-10-25 12:37:07.74 spid11s     The Service Broker protocol transport is disabled or not configured.
2010-10-25 12:37:07.74 spid11s     The Database Mirroring protocol transport is disabled or not configured.
2010-10-25 12:37:07.92 spid11s     Service Broker manager has started.
2010-10-25 12:38:07.20 spid51      DBCC TRACEON 3604, server process ID (SPID) 51. This is an informational message only; no user action is required.
2010-10-25 12:39:47.26 spid51      DBCC TRACEON 3604, server process ID (SPID) 51. This is an informational message only; no user action is required.
2010-10-25 12:45:37.29 spid52      Starting up database 'RFID_test'.
2010-10-25 12:45:37.45 spid52      Recovery is writing a checkpoint in database 'RFID_test' (5). This is an informational message only. No user action is required.
2010-10-25 13:53:22.35 spid11s     Service Broker manager has shut down.
2010-10-25 13:53:22.81 spid51      Server shut down by request from login arkar.
2010-10-25 13:53:22.81 spid51      SQL Trace was stopped due to server shutdown. Trace ID = '1'. This is an informational message only; no user action is required.
```

Figure A18.1: The screenshot of the error log (Not in the scope of the investigation)



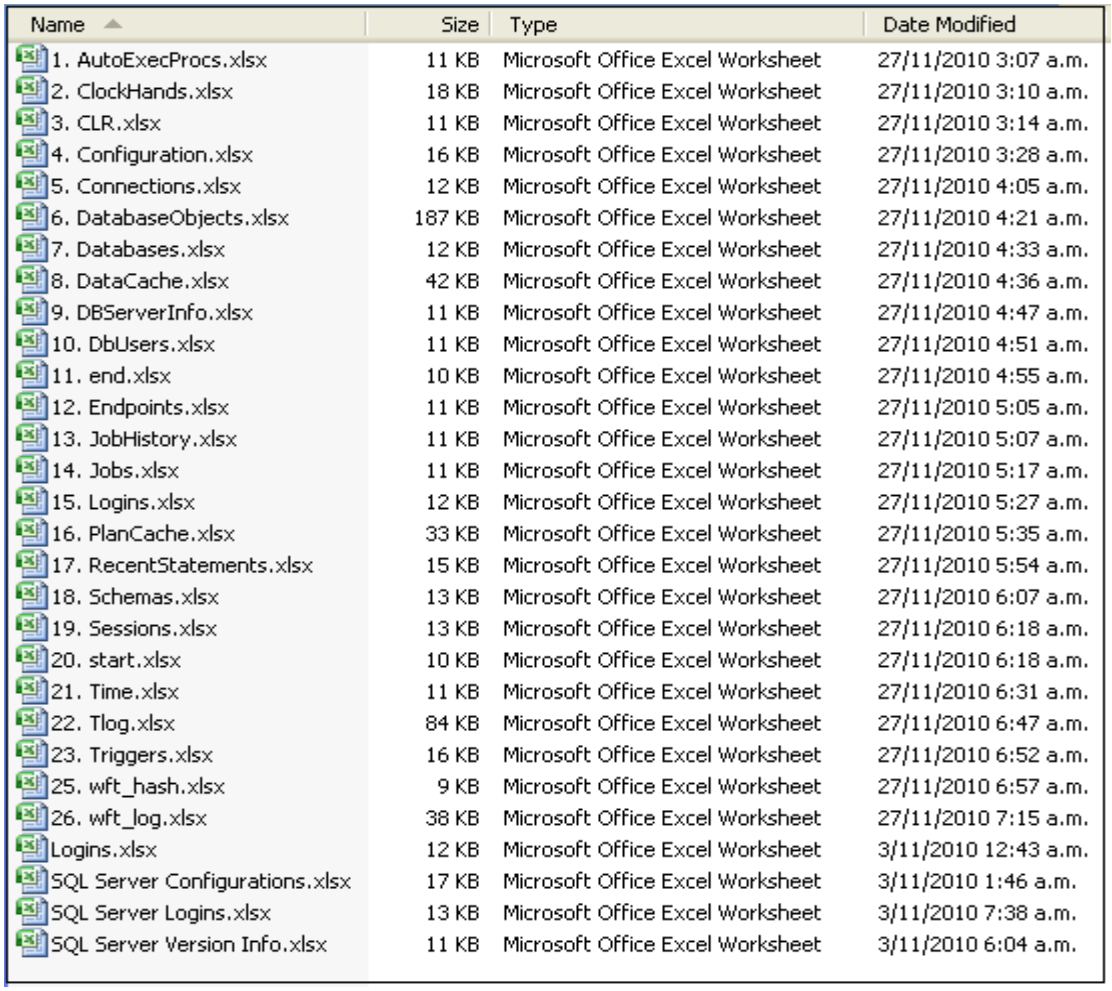
Name	Size	Type
AcquiredEvidences		File Folder
PhysicalDriveImage		File Folder
VolatileSQLDATA		File Folder
.E01	655,137 KB	EnCase Evidence File
.E02	655,136 KB	E02 File
.E03	655,136 KB	E03 File
.E04	655,136 KB	E04 File
.E05	489,497 KB	E05 File
errorlog	11 KB	File
errorlog.1	20 KB	1 File
errorlog.2	12 KB	2 File
errorlog.3	12 KB	3 File
errorlog.4	11 KB	4 File
errorlog.5	12 KB	5 File
errorlog.6	11 KB	6 File
errorlog.md5	1 KB	MD5 File
errorlog_1.md5	1 KB	MD5 File
errorlog_2.md5	1 KB	MD5 File
errorlog_3.md5	1 KB	MD5 File
errorlog_4.md5	1 KB	MD5 File
errorlog_5.md5	1 KB	MD5 File
errorlog_6.md5	1 KB	MD5 File
log_34.md5	1 KB	MD5 File
log_34.trc	128 KB	TRC File
log_35.md5	1 KB	MD5 File
log_35.trc	128 KB	TRC File
log_36.md5	1 KB	MD5 File
log_36.trc	128 KB	TRC File
log_37.md5	1 KB	MD5 File
log_37.trc	128 KB	TRC File
log_38.md5	1 KB	MD5 File
log_38.trc	128 KB	TRC File
New Wordpad Document.doc	72,635 KB	Microsoft Office Word 97 - 2003 Document
ReaderMemoryImage.bin	2 KB	BIN File
ReaderMemoryImage.md5.txt	1 KB	Text Document
RFID_test.md5	1 KB	MD5 File
RFID_test.mdf	2,048 KB	SQL Server Database Primary Data File
RFID_test_log.ldf	1,024 KB	SQL Server Database Transaction Log File
RFID_test_log.md5	1 KB	MD5 File

Figure A18.2: Screenshot of the Acquired Image Copies of Evidence Files from the Forensic Workstation



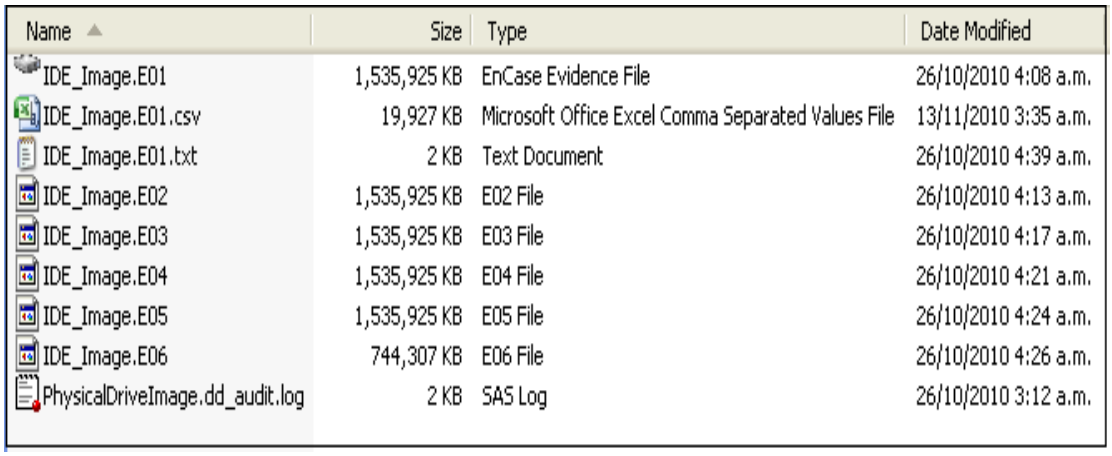
Name	Size	Type	Date Modified
14. ConnectionDetails_01.txt	2 KB	Text Document	6/11/2010 2:52 a.m.
AcquiredArtefacts_v5.docx	2,299 KB	Microsoft Office Word Document	27/11/2010 7:07 a.m.
AsymmetricKeys.xlsx	11 KB	Microsoft Office Excel Worksheet	6/11/2010 1:50 a.m.
Authorization Data Results.xlsx	12 KB	Microsoft Office Excel Worksheet	29/10/2010 6:55 a.m.
Certificates.xlsx	13 KB	Microsoft Office Excel Worksheet	6/11/2010 1:56 a.m.
CollationAndDataType.xlsx	1,499 KB	Microsoft Office Excel Worksheet	23/11/2010 2:53 a.m.
ConnectionDetail_01.xlsx	11 KB	Microsoft Office Excel Worksheet	6/11/2010 3:06 a.m.
ConnectionDetailError.xlsx	11 KB	Microsoft Office Excel Worksheet	6/11/2010 2:39 a.m.
DBCC Log Command Results_TransactionLog.xlsx	138 KB	Microsoft Office Excel Worksheet	29/10/2010 4:41 a.m.
DBCC Loginfo Command Results.xlsx	11 KB	Microsoft Office Excel Worksheet	29/10/2010 3:39 a.m.
DBO_RFIDdb_tb.xlsx	12 KB	Microsoft Office Excel Worksheet	23/11/2010 12:55 a.m.
DBO_RFIDdb_tblStat_Date.xlsx	12 KB	Microsoft Office Excel Worksheet	22/11/2010 11:54 p.m.
DBO_RFIDdb_tblStat_Tag.xlsx	11 KB	Microsoft Office Excel Worksheet	22/11/2010 11:39 p.m.
DBO_RFIDdb_tblStat_Value.xlsx	11 KB	Microsoft Office Excel Worksheet	22/11/2010 11:46 p.m.
DBO_RFIDlog_tbl.xlsx	12 KB	Microsoft Office Excel Worksheet	23/11/2010 2:27 a.m.
DBO_RFIDlog_tblStat_Date.xlsx	12 KB	Microsoft Office Excel Worksheet	23/11/2010 1:13 a.m.
DBO_RFIDlog_tblStat_Tag.xlsx	11 KB	Microsoft Office Excel Worksheet	23/11/2010 1:03 a.m.
DBO_RFIDlog_tblStat_UserData1.xlsx	12 KB	Microsoft Office Excel Worksheet	27/11/2010 2:35 a.m.
DBO_RFIDlog_tblStat_UserData2.xlsx	12 KB	Microsoft Office Excel Worksheet	27/11/2010 2:33 a.m.
DBO_RFIDlog_tblStat_UserData.xlsx	12 KB	Microsoft Office Excel Worksheet	27/11/2010 2:34 a.m.
DBO_RFIDlog_tblStat_Value.xlsx	11 KB	Microsoft Office Excel Worksheet	23/11/2010 1:07 a.m.
Login Config.xlsx	11 KB	Microsoft Office Excel Worksheet	29/10/2010 6:39 a.m.
Ring Buffers Data Results.xlsx	84 KB	Microsoft Office Excel Worksheet	29/10/2010 5:13 a.m.
Ring Buffers Login Failures and Security Errors.xlsx	11 KB	Microsoft Office Excel Worksheet	29/10/2010 6:09 a.m.
SAC Results.xlsx	15 KB	Microsoft Office Excel Worksheet	3/11/2010 1:18 a.m.
ServerPermission.xlsx	12 KB	Microsoft Office Excel Worksheet	27/10/2010 1:47 a.m.
ServerPrincipalData.xlsx	11 KB	Microsoft Office Excel Worksheet	29/10/2010 3:17 a.m.
ServiceShutDown.xlsx	10 KB	Microsoft Office Excel Worksheet	6/11/2010 4:49 a.m.
SymmetricKeys.xlsx	11 KB	Microsoft Office Excel Worksheet	6/11/2010 1:45 a.m.
SystemDatabasePermissions.xlsx	86 KB	Microsoft Office Excel Worksheet	20/11/2010 6:39 a.m.
SystemDatabasePrincipals.xlsx	12 KB	Microsoft Office Excel Worksheet	20/11/2010 6:59 a.m.
SystemDatabaseRoleMembers.xlsx	11 KB	Microsoft Office Excel Worksheet	20/11/2010 7:05 a.m.
TableStatistics.xlsx	18 KB	Microsoft Office Excel Worksheet	21/11/2010 4:15 a.m.
TransactionLog1.xlsx	138 KB	Microsoft Office Excel Worksheet	16/11/2010 4:43 a.m.

Figure A18.3: Screenshots of the Acquired SQL Artefacts by WFT (already exported to Excel formats)



Name	Size	Type	Date Modified
1. AutoExecProcs.xlsx	11 KB	Microsoft Office Excel Worksheet	27/11/2010 3:07 a.m.
2. ClockHands.xlsx	18 KB	Microsoft Office Excel Worksheet	27/11/2010 3:10 a.m.
3. CLR.xlsx	11 KB	Microsoft Office Excel Worksheet	27/11/2010 3:14 a.m.
4. Configuration.xlsx	16 KB	Microsoft Office Excel Worksheet	27/11/2010 3:28 a.m.
5. Connections.xlsx	12 KB	Microsoft Office Excel Worksheet	27/11/2010 4:05 a.m.
6. DatabaseObjects.xlsx	187 KB	Microsoft Office Excel Worksheet	27/11/2010 4:21 a.m.
7. Databases.xlsx	12 KB	Microsoft Office Excel Worksheet	27/11/2010 4:33 a.m.
8. DataCache.xlsx	42 KB	Microsoft Office Excel Worksheet	27/11/2010 4:36 a.m.
9. DBServerInfo.xlsx	11 KB	Microsoft Office Excel Worksheet	27/11/2010 4:47 a.m.
10. DbUsers.xlsx	11 KB	Microsoft Office Excel Worksheet	27/11/2010 4:51 a.m.
11. end.xlsx	10 KB	Microsoft Office Excel Worksheet	27/11/2010 4:55 a.m.
12. Endpoints.xlsx	11 KB	Microsoft Office Excel Worksheet	27/11/2010 5:05 a.m.
13. JobHistory.xlsx	11 KB	Microsoft Office Excel Worksheet	27/11/2010 5:07 a.m.
14. Jobs.xlsx	11 KB	Microsoft Office Excel Worksheet	27/11/2010 5:17 a.m.
15. Logins.xlsx	12 KB	Microsoft Office Excel Worksheet	27/11/2010 5:27 a.m.
16. PlanCache.xlsx	33 KB	Microsoft Office Excel Worksheet	27/11/2010 5:35 a.m.
17. RecentStatements.xlsx	15 KB	Microsoft Office Excel Worksheet	27/11/2010 5:54 a.m.
18. Schemas.xlsx	13 KB	Microsoft Office Excel Worksheet	27/11/2010 6:07 a.m.
19. Sessions.xlsx	13 KB	Microsoft Office Excel Worksheet	27/11/2010 6:18 a.m.
20. start.xlsx	10 KB	Microsoft Office Excel Worksheet	27/11/2010 6:18 a.m.
21. Time.xlsx	11 KB	Microsoft Office Excel Worksheet	27/11/2010 6:31 a.m.
22. Tlog.xlsx	84 KB	Microsoft Office Excel Worksheet	27/11/2010 6:47 a.m.
23. Triggers.xlsx	16 KB	Microsoft Office Excel Worksheet	27/11/2010 6:52 a.m.
25. wft_hash.xlsx	9 KB	Microsoft Office Excel Worksheet	27/11/2010 6:57 a.m.
26. wft_log.xlsx	38 KB	Microsoft Office Excel Worksheet	27/11/2010 7:15 a.m.
Logins.xlsx	12 KB	Microsoft Office Excel Worksheet	3/11/2010 12:43 a.m.
SQL Server Configurations.xlsx	17 KB	Microsoft Office Excel Worksheet	3/11/2010 1:46 a.m.
SQL Server Logins.xlsx	13 KB	Microsoft Office Excel Worksheet	3/11/2010 7:38 a.m.
SQL Server Version Info.xlsx	11 KB	Microsoft Office Excel Worksheet	3/11/2010 6:04 a.m.

Figure A18.4: Screenshots of the Acquired Volatile SQL Artefacts by WFT (already exported to Excel formats)



Name	Size	Type	Date Modified
IDE_Image.E01	1,535,925 KB	EnCase Evidence File	26/10/2010 4:08 a.m.
IDE_Image.E01.csv	19,927 KB	Microsoft Office Excel Comma Separated Values File	13/11/2010 3:35 a.m.
IDE_Image.E01.txt	2 KB	Text Document	26/10/2010 4:39 a.m.
IDE_Image.E02	1,535,925 KB	E02 File	26/10/2010 4:13 a.m.
IDE_Image.E03	1,535,925 KB	E03 File	26/10/2010 4:17 a.m.
IDE_Image.E04	1,535,925 KB	E04 File	26/10/2010 4:21 a.m.
IDE_Image.E05	1,535,925 KB	E05 File	26/10/2010 4:24 a.m.
IDE_Image.E06	744,307 KB	E06 File	26/10/2010 4:26 a.m.
PhysicalDriveImage.dd_audit.log	2 KB	SAS Log	26/10/2010 3:12 a.m.

Figure A18.5: Screenshot of the Acquired Evidence Copy of Physical Drive Images

Appendix 19: An Example of a Successful Pilot Test in Data Collection Process by Using Extended WFT from Customized Helix_RFID_IR Toolkit

Step 1: After placing the customized Incident Response (Helix_RIFD_IR) toolkit into the DVD drive of the compromised system, the investigator firstly entered the investigative notes (Figure A19.1) and recorded the target system information (Figure A19.2a and Figure A19.2b).

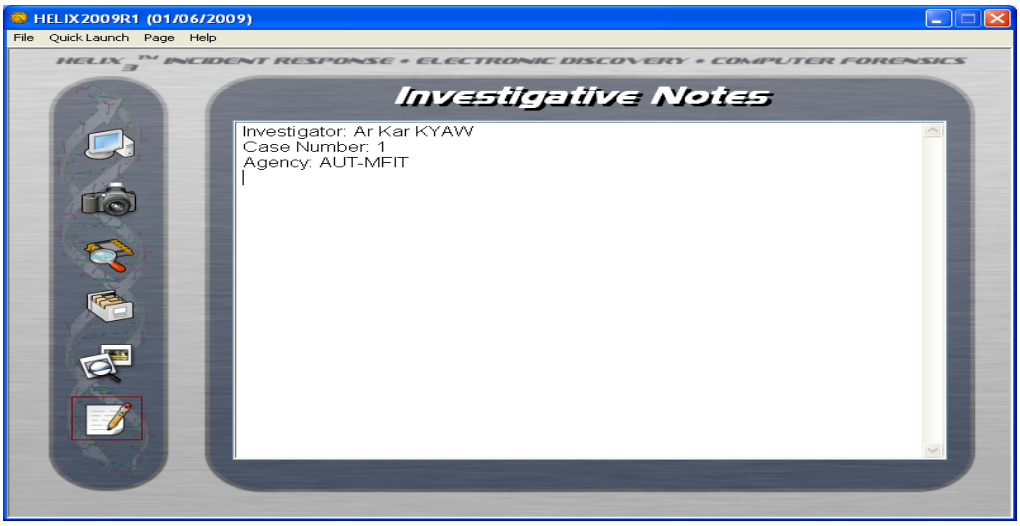


Figure A19.1: Screenshot of the Investigative Notes

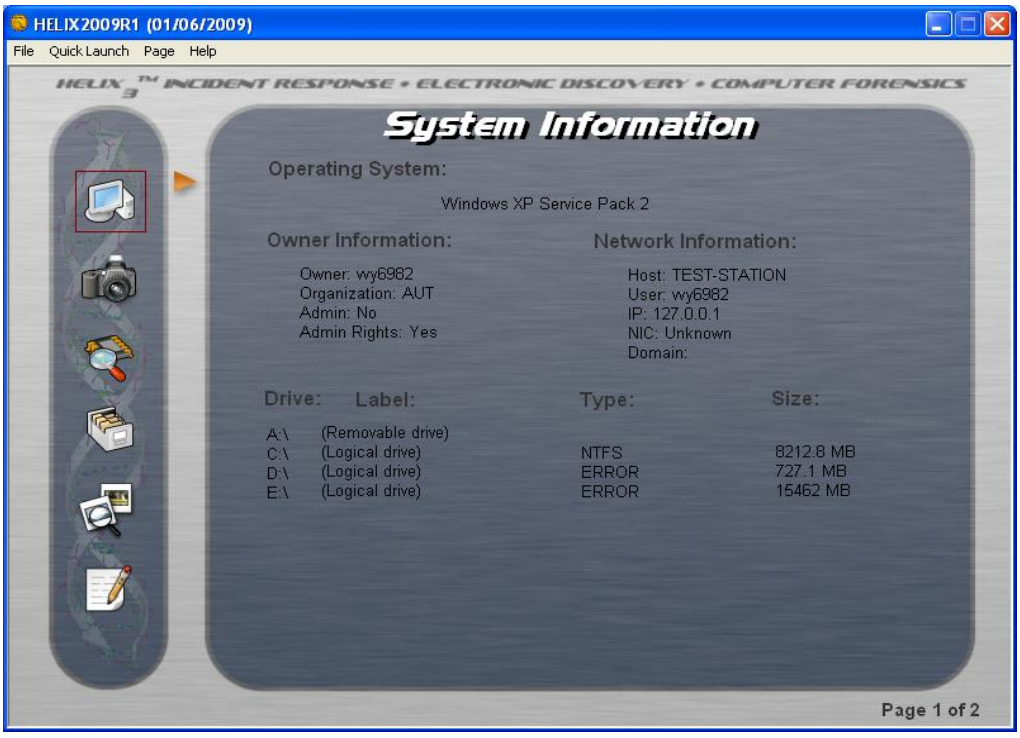


Figure A19.2a: Screenshot of the Investigative Notes

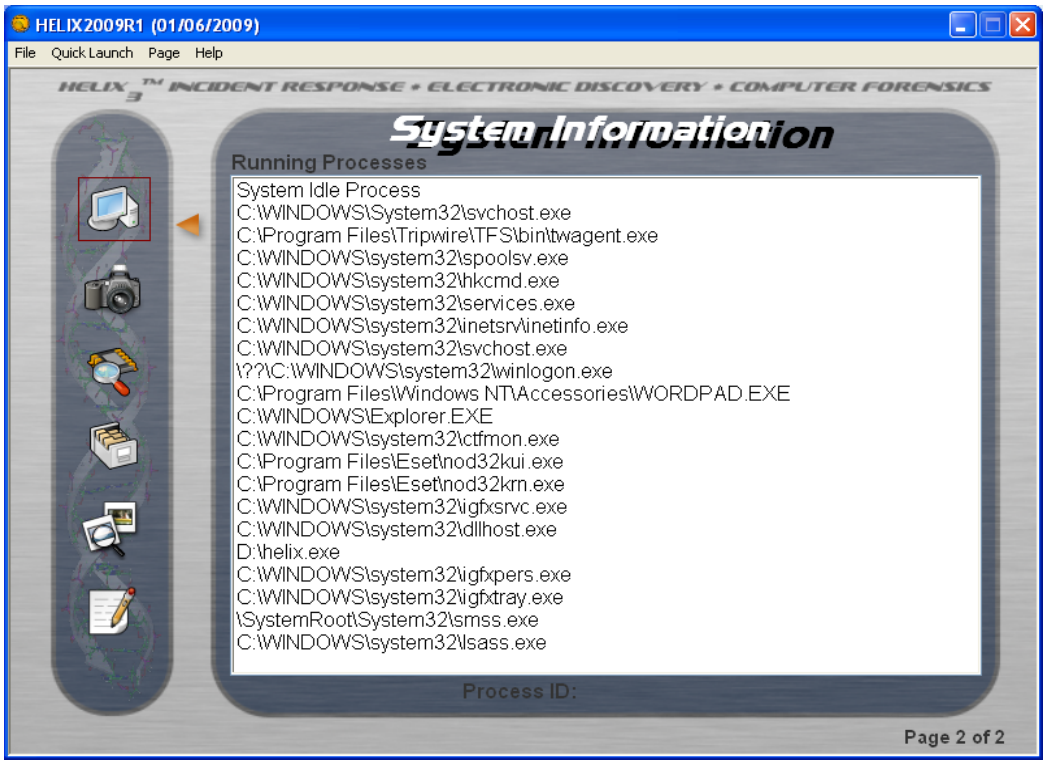


Figure A19.2b: Screenshot of the Investigative Notes

Step 2: The next step was to use the trusted command prompt from the Helix_RFID_IR toolkit DVD (D:\IR\wft\tools\xp\cmd.exe) in order to execute the extended *Windows Forensic Toolchest* (WFT) for the purpose of the SQL Server data acquisition.

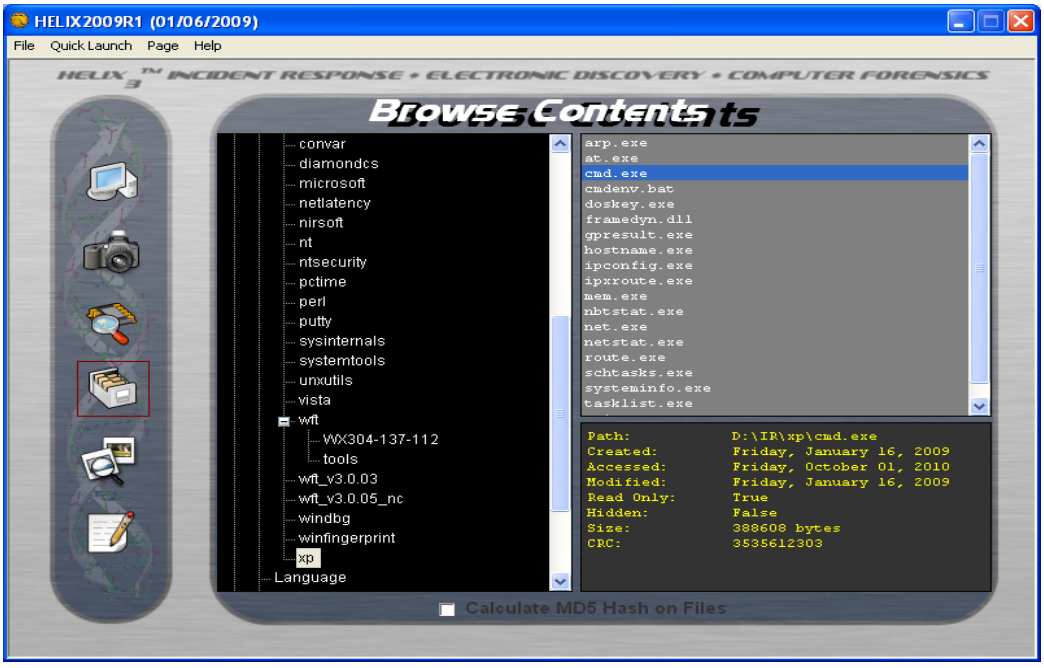


Figure A19.3: Screenshot of the location of the trusted command prompt

Step 3: To perform the acquisition of the SQL Server data with WFT, the batch file (wftSQL.bat) was run by using the following syntax.

```
D:\IR\wft\wftSQL.bat TEST-STATION\SQLEXPRESS arkar qwe123AUT
```

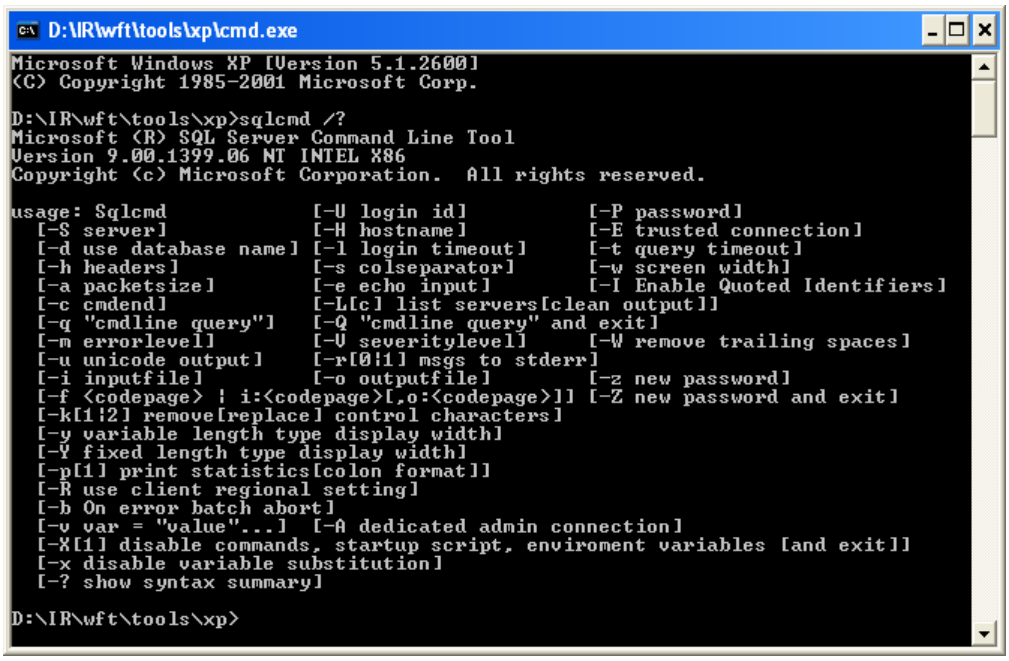


Figure A19.4: Screenshot of the trusted command prompt running on the compromised system

Step 4: Then, the investigator pressed 'ENTER' to start the acquisition.

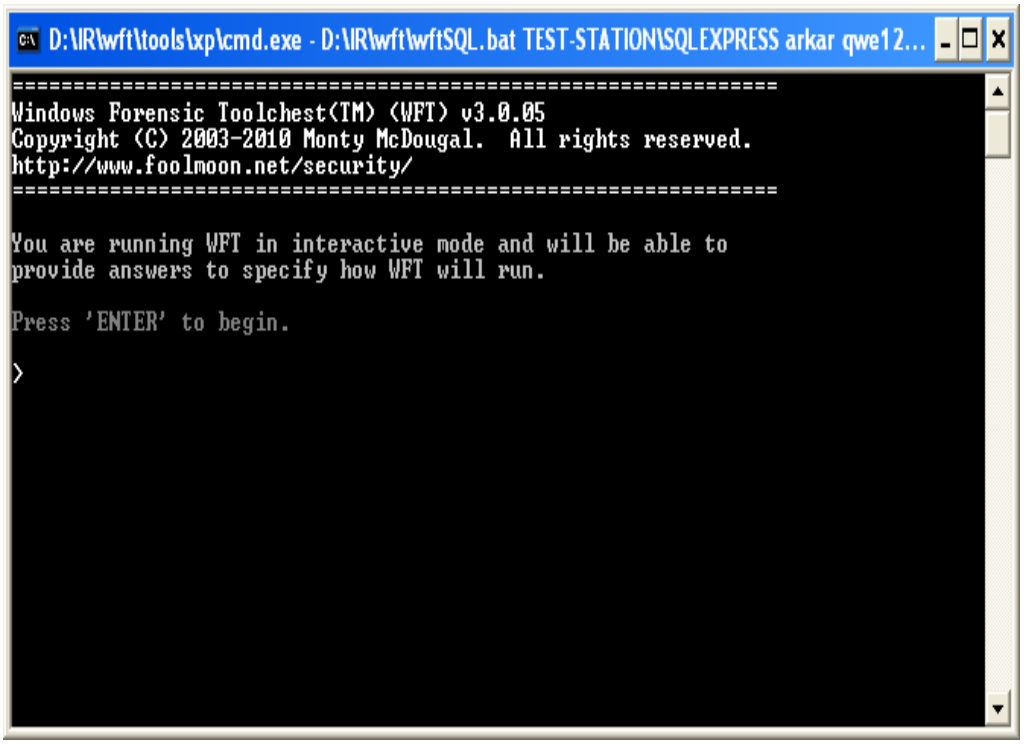


Figure A19.5: Screenshot of the launching WFT tool

Step 5: Then, the investigator chose the specific WFT configuration file.

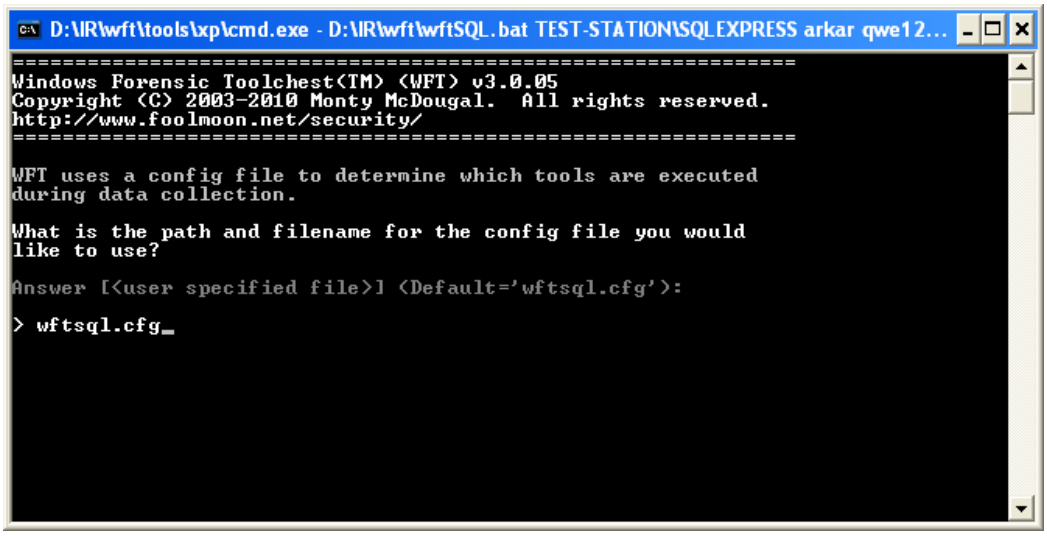


Figure A19.6: Screenshot of choosing the specific WFT configuration file

Step 6: Then, the investigator chose the specific tools.

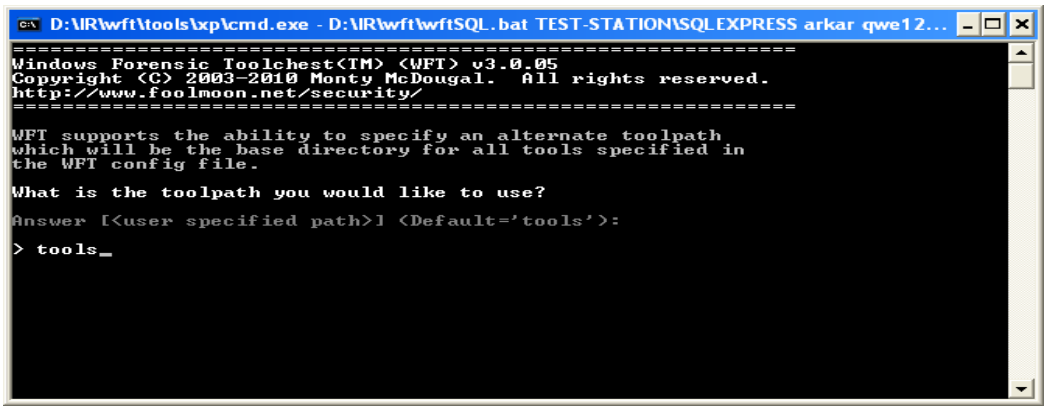


Figure A19.7: Choosing the specific WFT configuration file

Step 7: The default Operating System (OS) path, '*auto'*, was chosen to run the OS specific commands by utilizing the trusted binaries for tool execution.

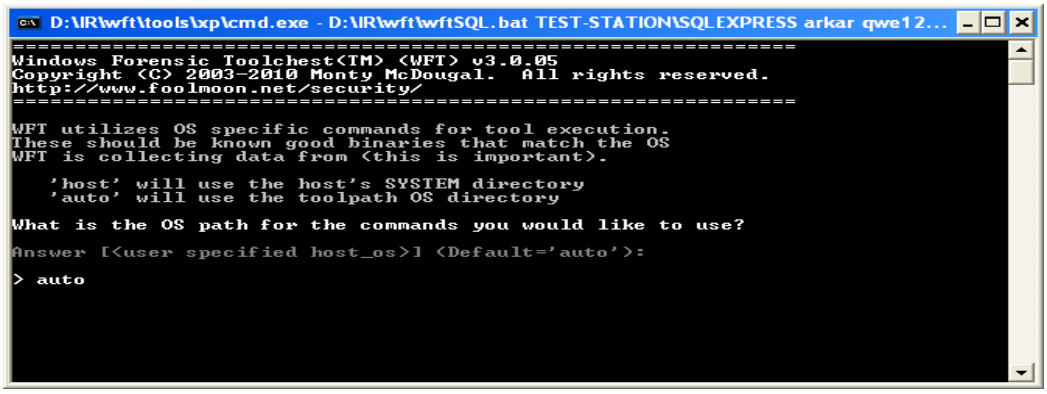


Figure A19.8: Choosing the OS path

Step 8: Likewise, the trusted command shell was chosen (Figure A19.9) and the trusted destination path for storing the potential evidence was keyed as shown in the Figure A19.10.

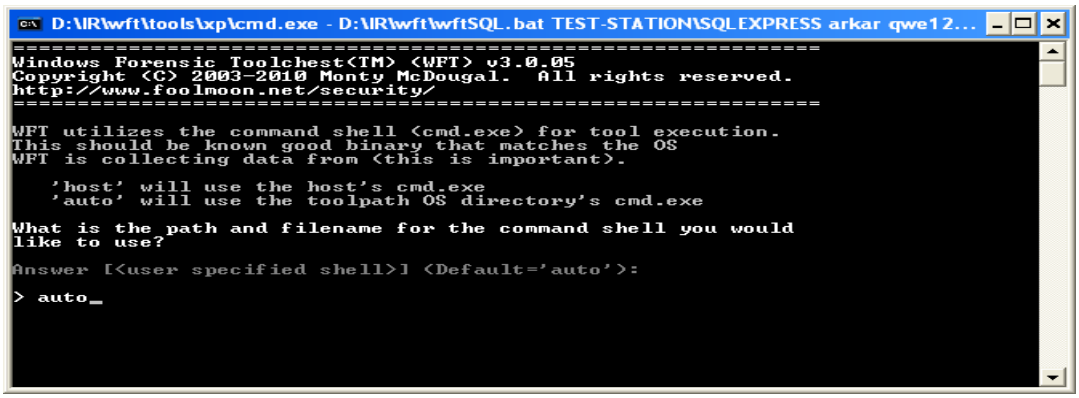


Figure A19.9: Choosing the OS path

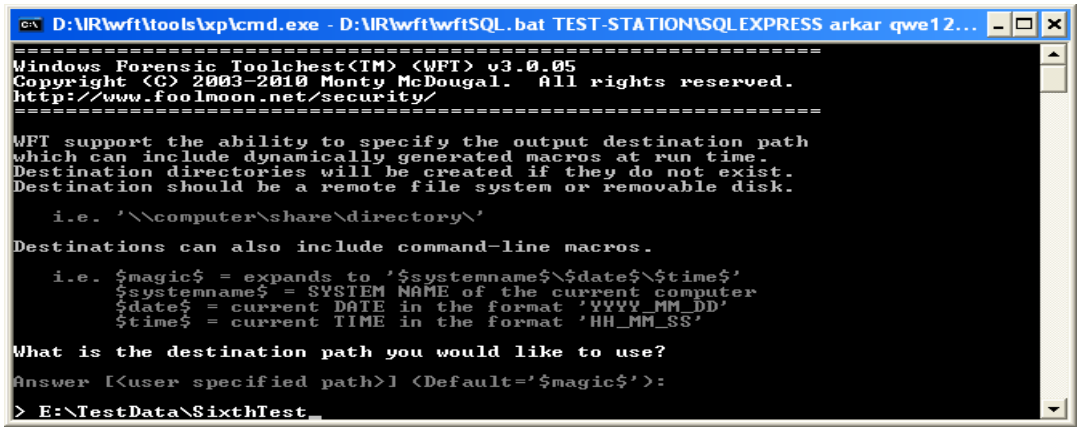


Figure A19.10: Choosing the OS path

Step 9: Then, the default 'auto' was chosen to use the drive during the acquisition (Figure A19.11), the name of the investigator was recorded (see Figure A19.12) and the name of the case was entered by the forensic examiner (see Figure A19.13).

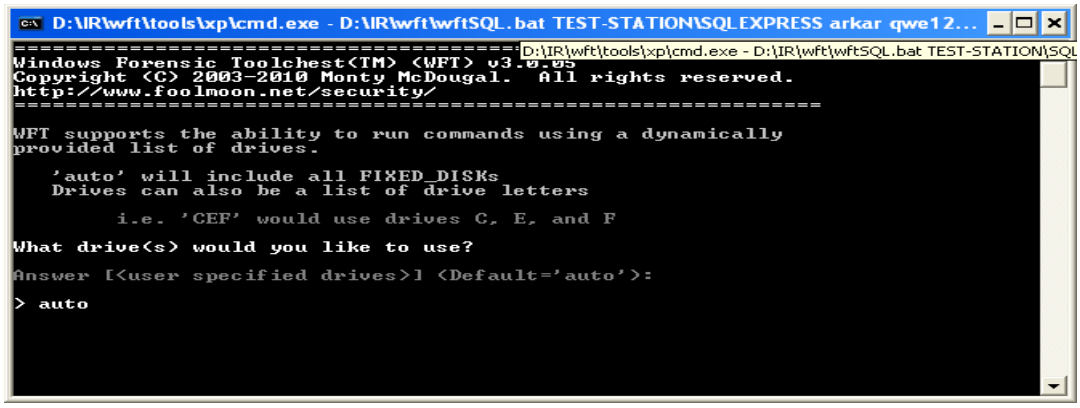


Figure A19.11: Choosing the auto for 'default' drive

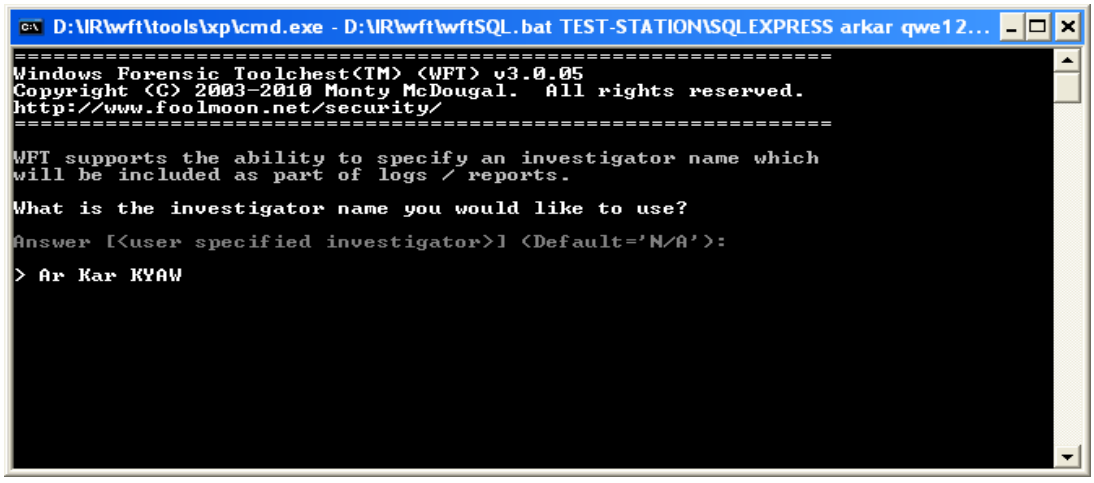


Figure A19.12: Entering the name of the forensic examiner

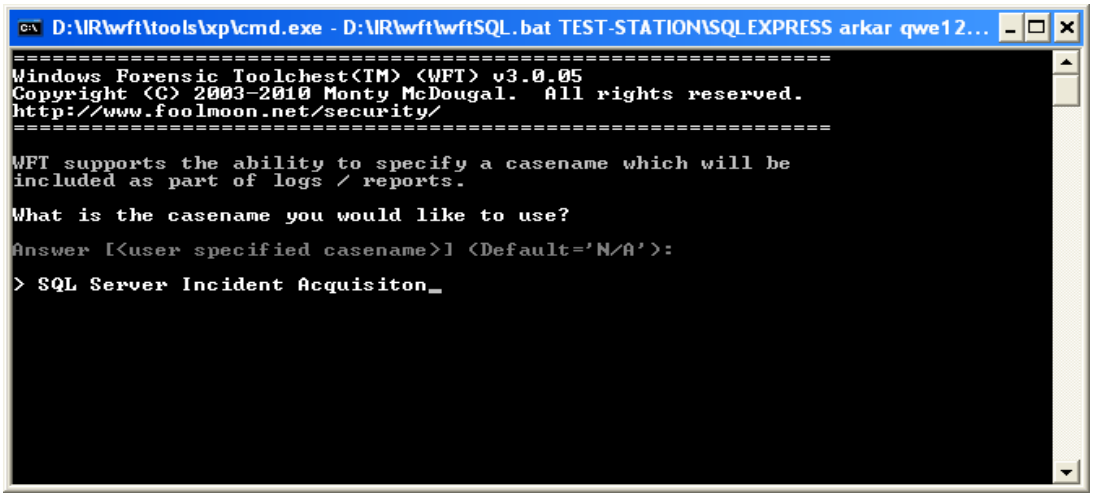


Figure A19.13: Giving the name of the incident case

Step 10: Afterwards, the digital hashing method (MD5) was chosen.

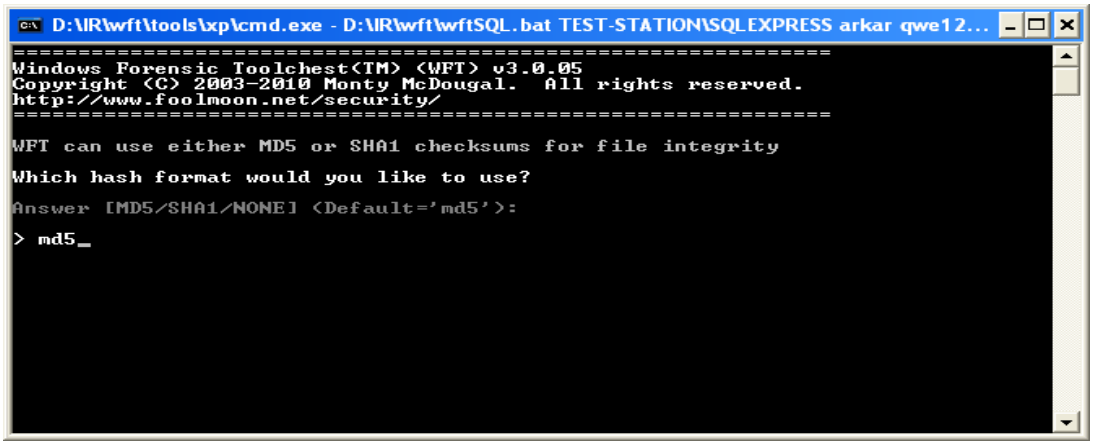


Figure A19.14: Chooing the digital hashing method

Step 11: Similarly, running the *slow* tools (Figure A19.15) and running the tools that could write to the compromised system (Figure A19.16) were avoided during the potential evidence data acquisition.

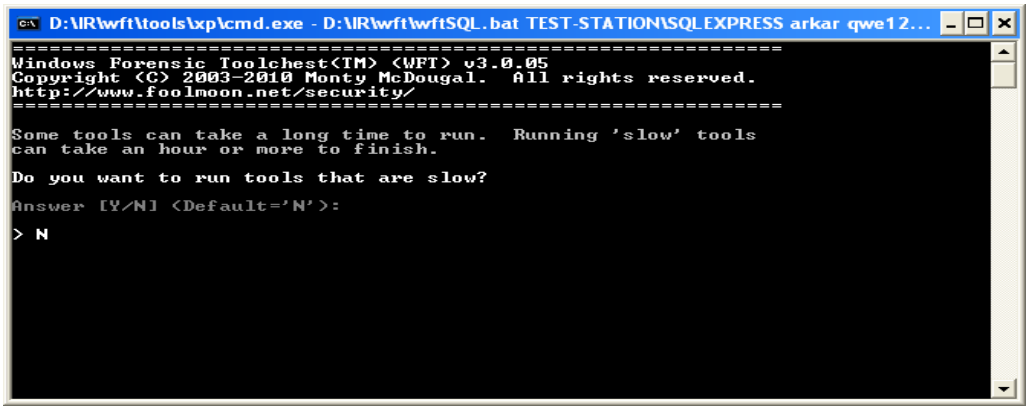


Figure A19.15: Avoiding the *slow* tools running on the compromised system

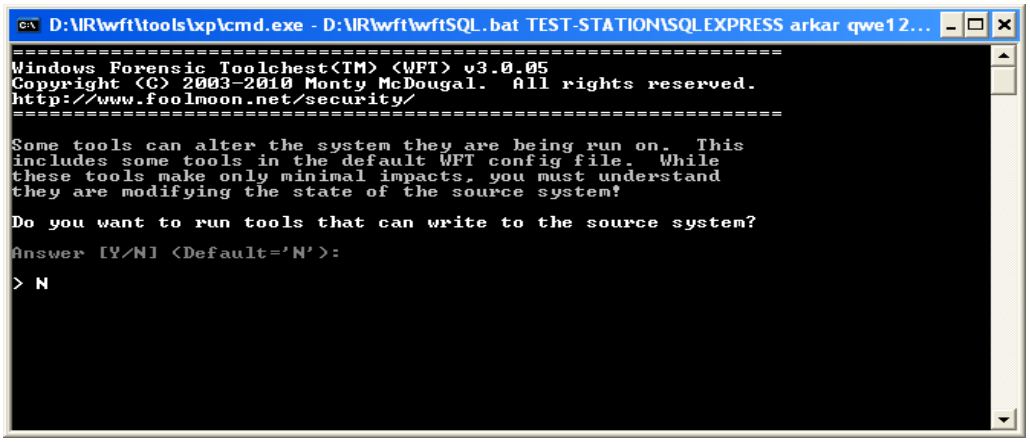


Figure A19.16: Avoiding the tools that could write to the source of the compromised system

Step 12: Then, HTML report format was also selected for the output of the collected SQL Server data, in addition to the text reports.

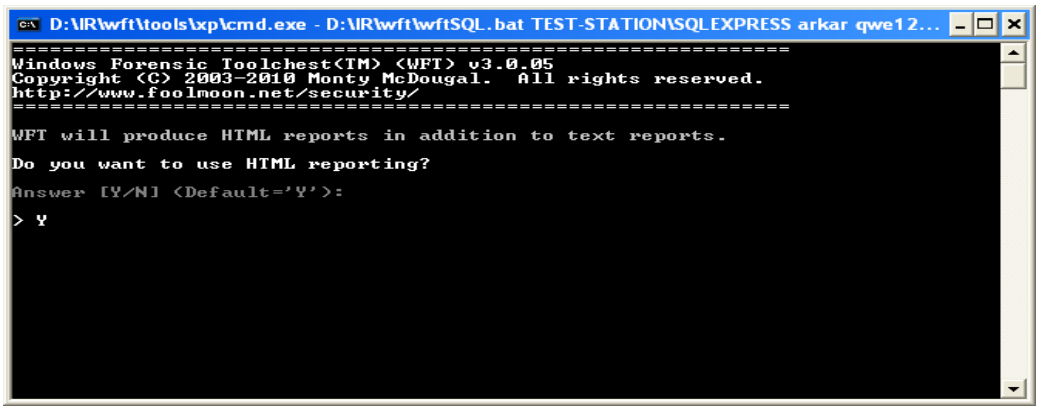


Figure A19.17: Selecting HTML report format for the collected data

Step 13: Likewise, the '*enable prompting*' by WFT tool was prevented during the acquisition of SQL Server data.

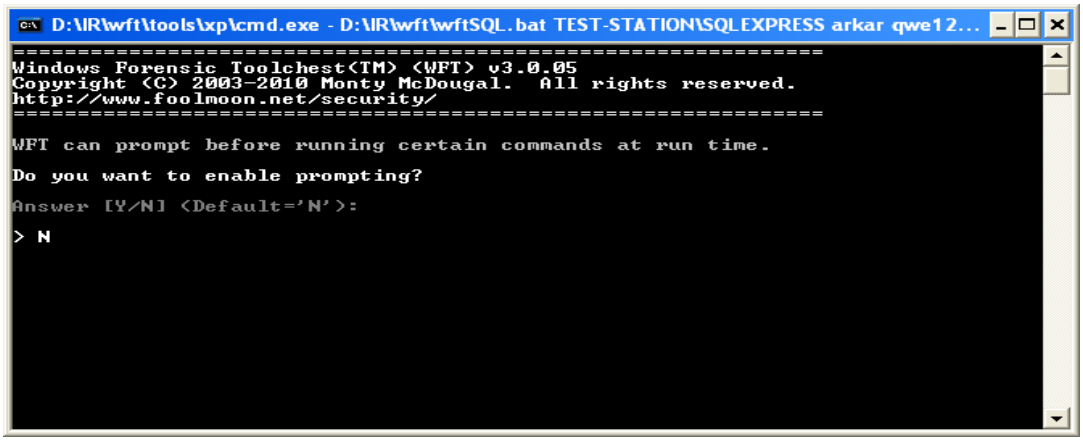


Figure A19.18: '*Enable prompting*' by WFT tool was avoided

Step 14: Automatic opening of the output HTML report in an external browser

after the completion of the SQL Server data acquisition was not selected during the investigation.

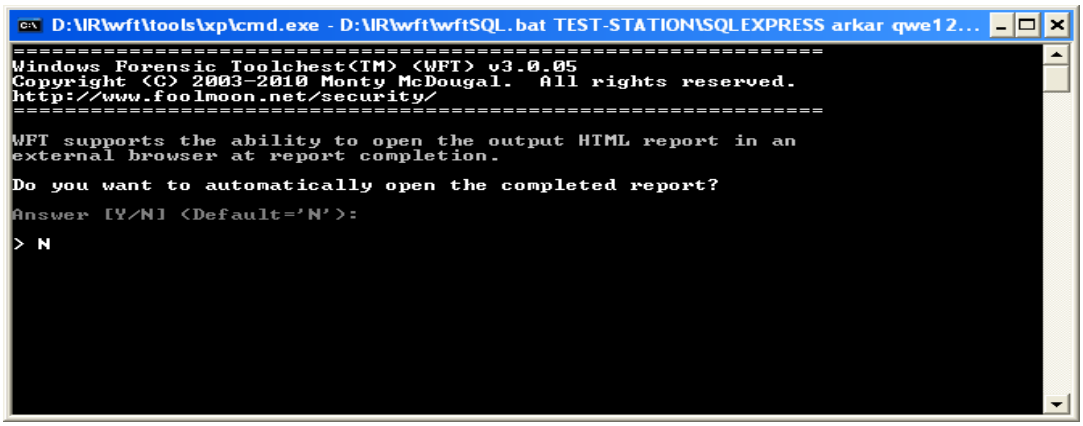


Figure A19.19: Avoiding the output HTML files to open automatically

Step 15: Finally, the investigator confirmed to start the acquisition of the SQL Server data by selecting 'Y' (Figure A19.20).

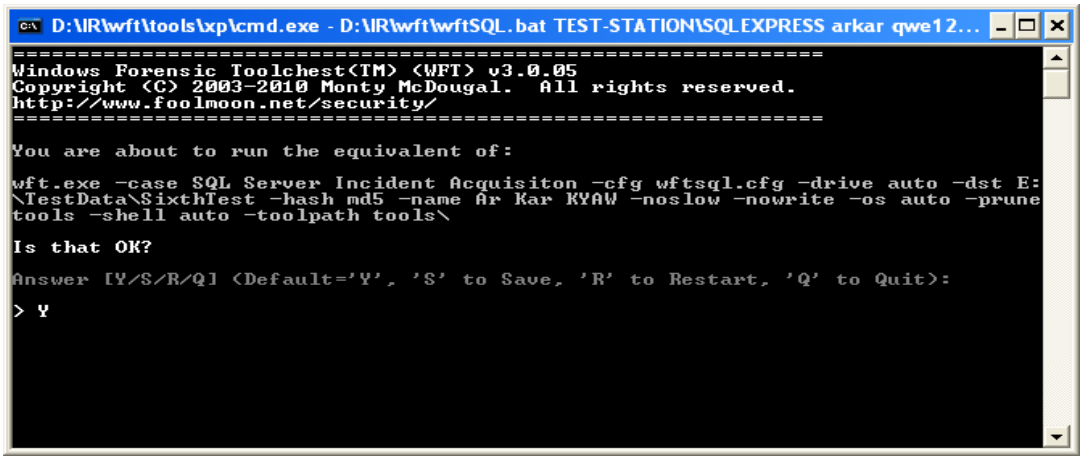


Figure A19.20: Avoiding the output HTML files to open automatically

Step 16: Some of the screenshots during the SQL Server data acquisition by using the WFT tool on a compromised system were captured as follow.

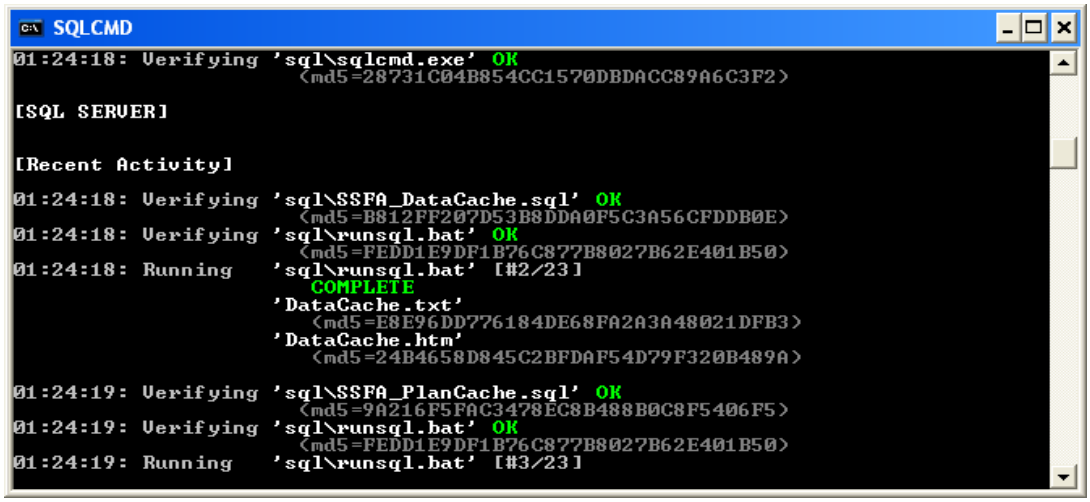


Figure A19.21a: WFT tool in action during acquisition

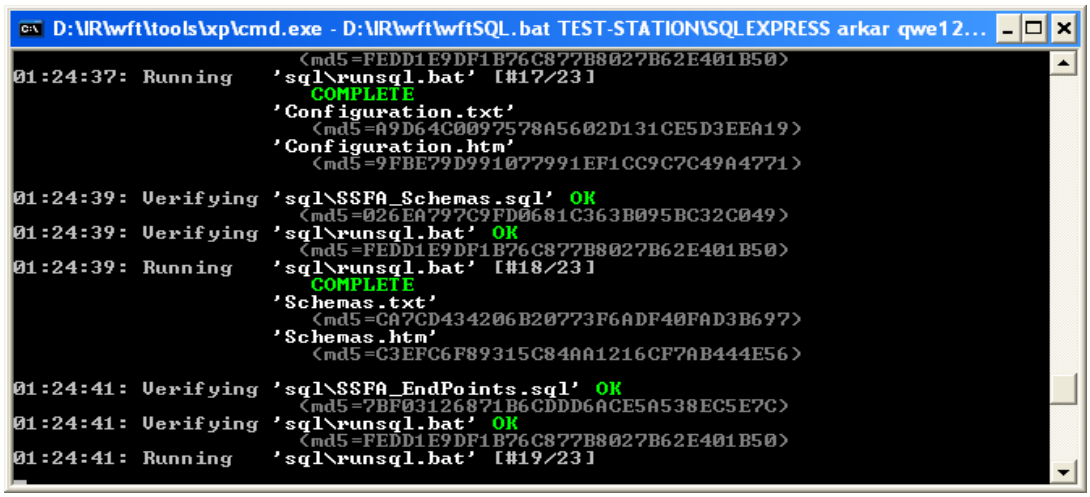


Figure A19.21b: WFT tool in action during acquisition

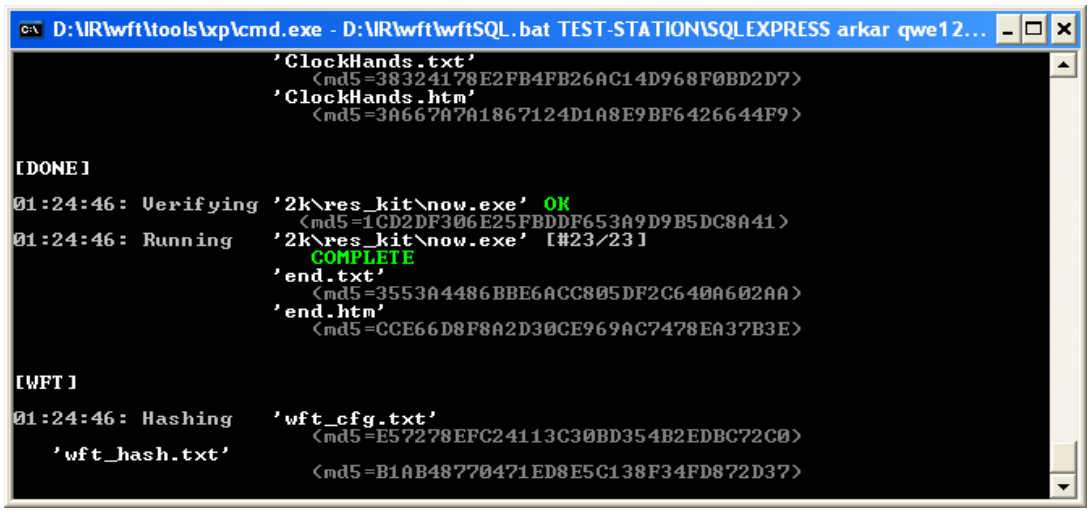


Figure A19.21c: WFT tool in action during acquisition

Step 17: The SQL Server data acquisition was successfully completed by using

the extended WFT tool from the customized RFID IR toolkit (see Figure A19.22a and Figure A19.22b).

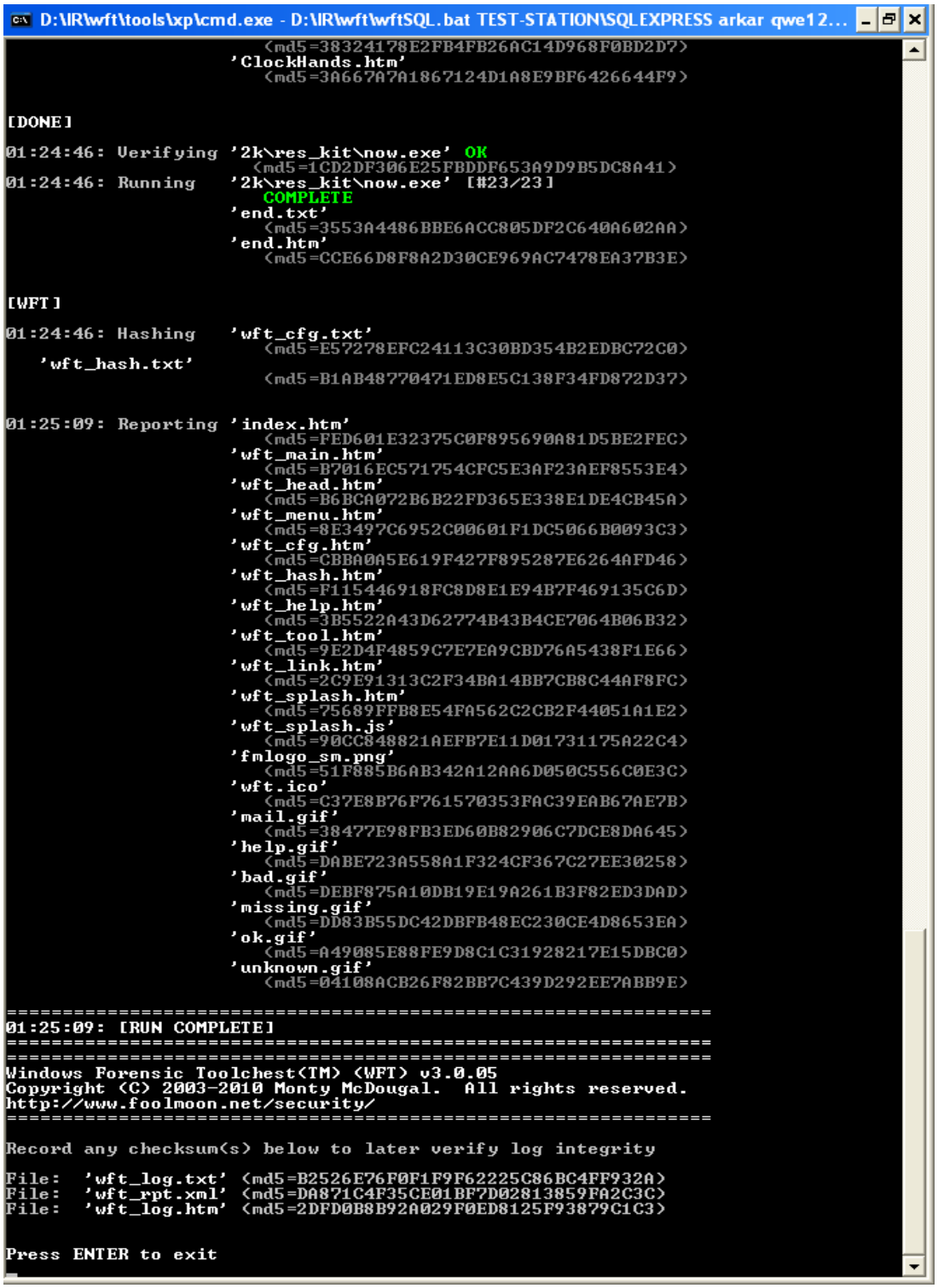


Figure A19.22a: The acquisition of the SQL Server data by using WFT tool was completed

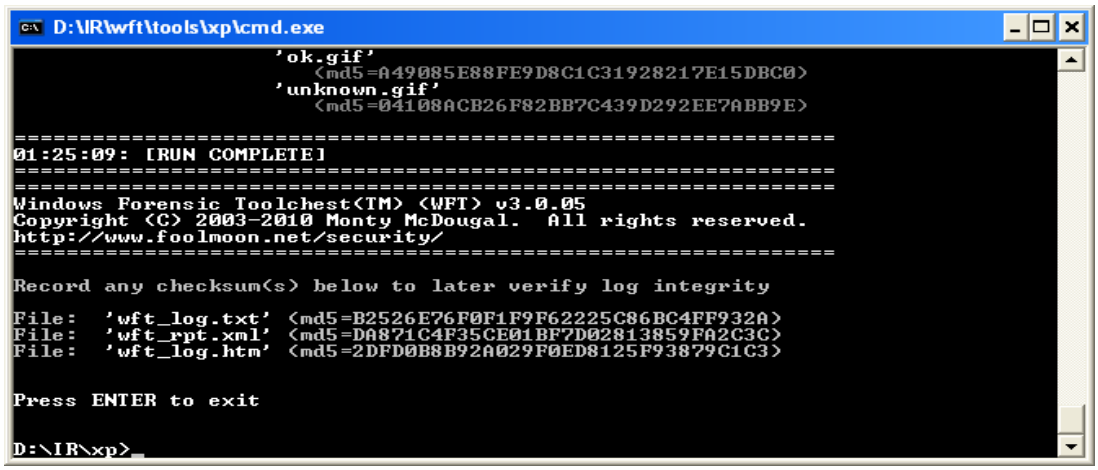


Figure A19.22b: The acquisition of the SQL Server data by using WFT tool was completed

Step 17: After the acquisition, the investigator could review the reports of the collected SQL Server data in HTML format.

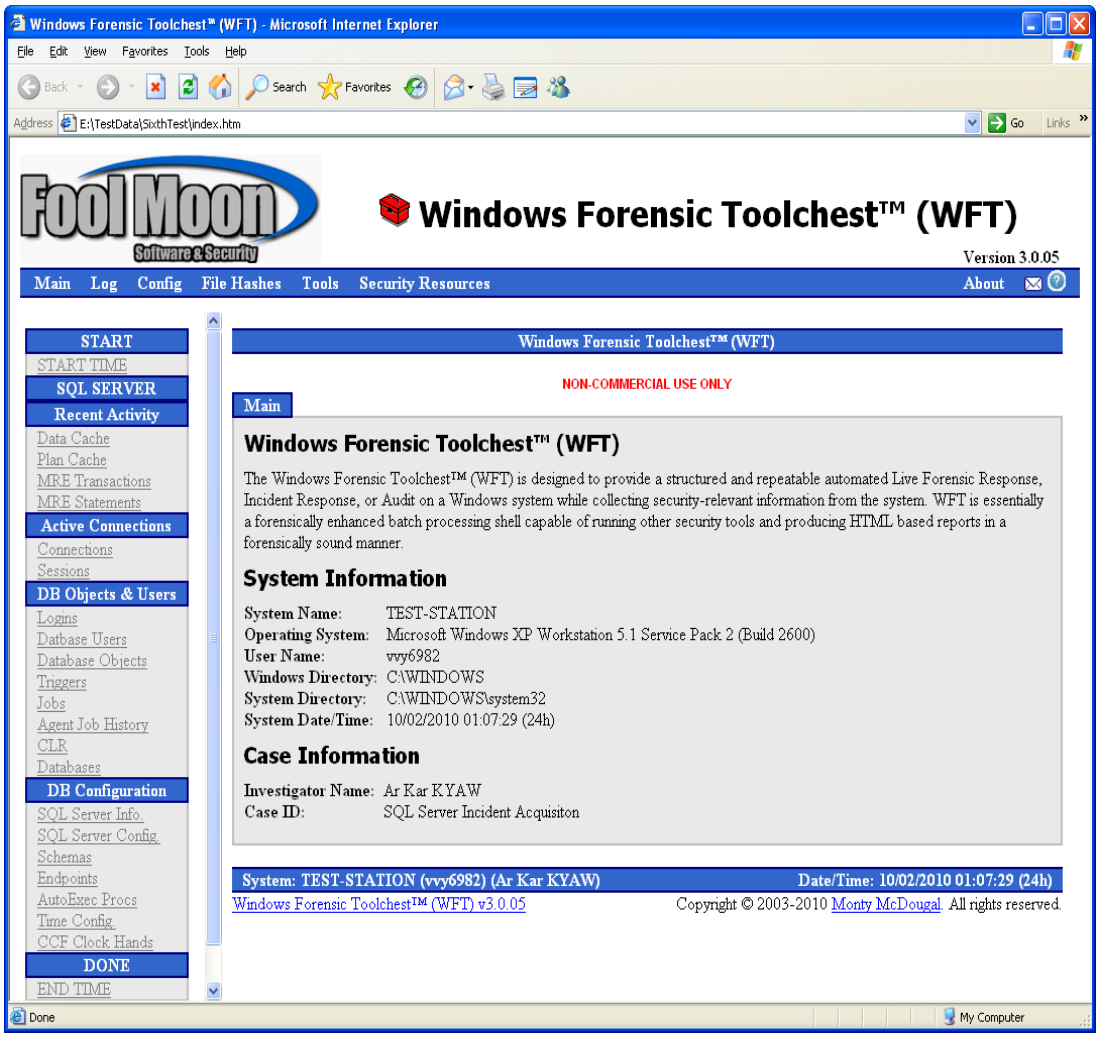


Figure A19.23: Observing the collected SQL Server data in a web browser

Step 18: Then, the investigator chose to save all the actions performed in a PDF log file before exiting from the application of RFID IR toolkit (see Figure A19.24 and Figure A19.25).

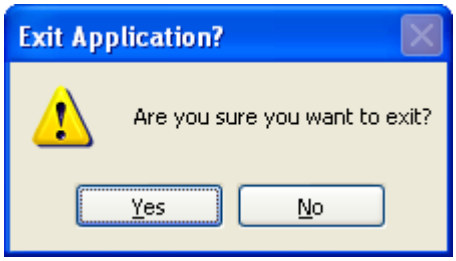


Figure A19.23: Exiting from the application of RFID IR toolkit

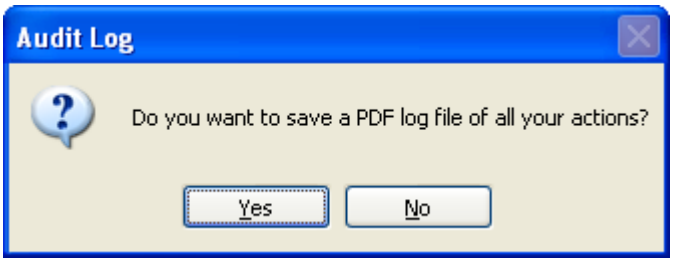


Figure A19.23: Saving all the actions performed in a PDF log file

Appendix 20: Extended wftSQL Batchfile of Windows Forensic Toolchest from Helix_RFID_IR toolkit

```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: wftSQL.bat v1.0.03
:: Copyright (C) 2007 Monty McDougal
::
:: This program is free software: you can redistribute it and/or modify
:: it under the terms of the GNU General Public License as published by
:: the Free Software Foundation, either version 3 of the License, or
:: (at your option) any later version.
::
:: This program is distributed in the hope that it will be useful,
:: MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
:: GNU General Public License for more details.
::
:: You should have received a copy of the GNU General Public License
:: along with this program.  If not, see <http://www.gnu.org/licenses/>.
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

@ECHO OFF

PUSHD %~dp0

echo
============================================================
==
echo wftSQL.bat v1.0.03
echo Copyright (C) 2007 Monty McDougal
echo Released under the terms of the GNU General Public License
echo
```

```
============================================================
echo.
echo Usage: wftSQL.bat [SQLserver]
echo Usage: wftSQL.bat [SQLserver] [SQLuser] [SQLpass]
echo.

IF EXIST wft.exe GOTO CFGCHECK
GOTO WFTERROR

:CFGCHECK

IF EXIST wftsql.cfg GOTO START
GOTO CFGERROR

:START

IF NOT "%1"=="" GOTO GOTSQLSERVER
:GETSQLSERVER
SET /P SQLSERVER=Enter target SQL Server Instance:
IF "%SQLSERVER%"=="" GOTO GETSQLSERVER
GOTO DONESQLSERVER
:GOTSQLSERVER
SET SQLSERVER=%1
:DONESQLSERVER

IF "%1"=="" GOTO DEFAULTUSERPROMPT
IF "%2"=="" GOTO DEFAULTUSER
IF "%3"=="" GOTO DEFAULTUSER
GOTO NOTDEFAULTUSER

:DEFAULTUSERPROMPT
```

```
SET /P USERQUESTION=Use currently logged in user credentials (y/n)?
IF "%USERQUESTION%"=="y" GOTO DEFAULTUSER
IF "%USERQUESTION%"=="Y" GOTO DEFAULTUSER
IF "%USERQUESTION%"=="n" GOTO NOTDEFAULTUSER
IF "%USERQUESTION%"=="N" GOTO NOTDEFAULTUSER
GOTO DEFAULTUSERPROMPT

:NOTDEFAULTUSER

IF NOT "%2"=="" GOTO GOTSQLUSER
:GETSQLUSER
SET /P SQLUSER=Enter user name:
IF "%SQLUSER%"=="" GOTO GETSQLUSER
GOTO DONESQLUSER
:GOTSQLUSER
SET SQLUSER=%2
:DONESQLUSER

IF NOT "%3"=="" GOTO GOTSQLPASS
:GETSQLPASS
SET /P SQLPASS=Enter password:
IF "%SQLPASS%"=="" GOTO GETSQLPASS
GOTO DONESQLPASS
:GOTSQLPASS
SET SQLPASS=%3
:DONESQLPASS

:DEFAULTUSER

wft.exe -cfg wftsql.cfg -nowrite -noslow
```

```
SET SQLSERVER=
SET SQLUSER=
SET SQLPASS=

GOTO END

:WFTERROR
echo Error:  'wft.exe' does not exist in '%~dp0'
pause
GOTO END

:CFGERROR
echo Error:  'wftSQL.cfg' does not exist in '%~dp0'
pause
GOTO END

:END

POPD
```

Appendix 21_Screenshots of Hashing Collected Artefacts after Analysis

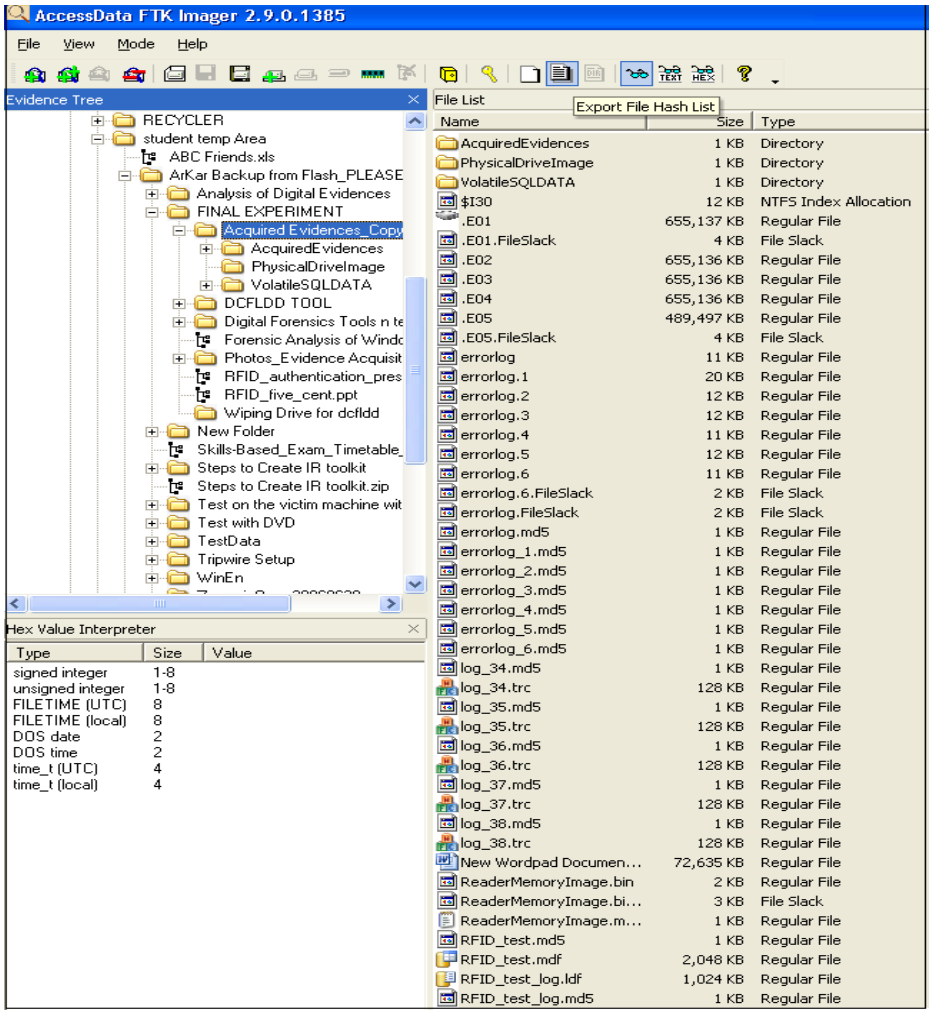


Figure A21.1: Hashing all the acquired artefacts of RFID BS after analysis

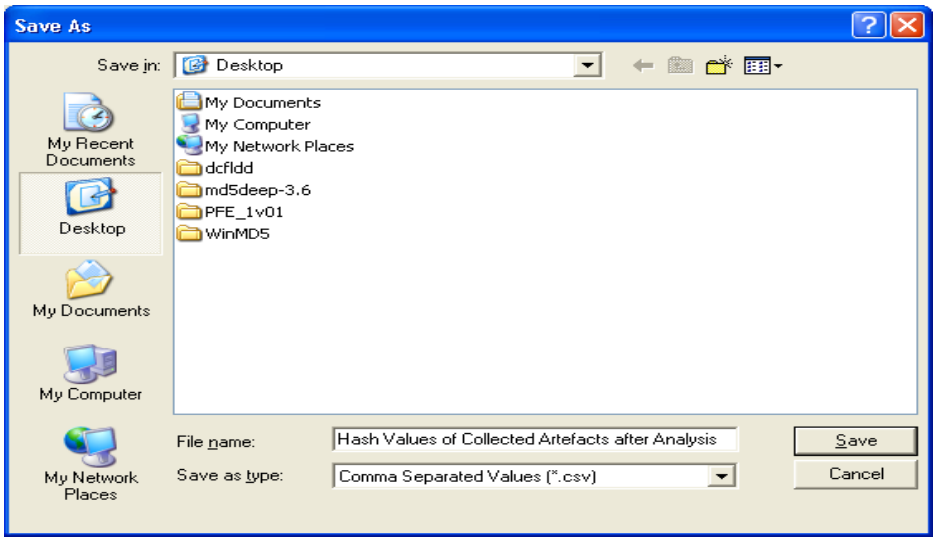


Figure A21.2: Hash values will be saved as a Comma Separated Values (CSV) file format on the desktop of a forensic workstation

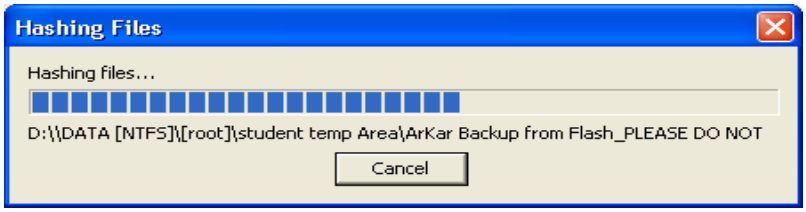


Figure A21.3: Screenshot of the progress of hashing collected artefacts by FTK

Figure A21.4: Hash values were saved in a designated file on the desktop

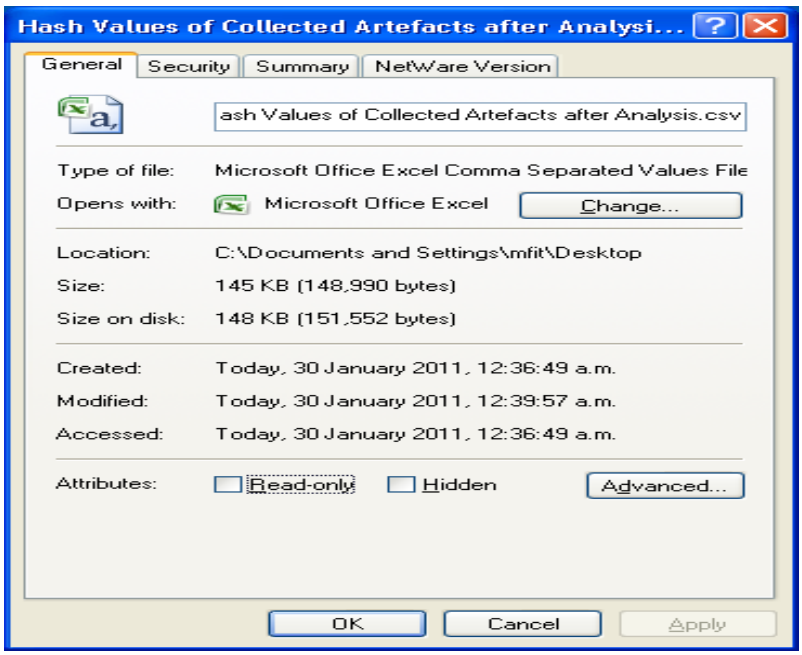


Figure A21.5: Hash values after analysis file was created

Appendix 22: Hash Values Comparison of All Acquired Files

Acquired Evidences	File Type	MD5 Hash Values (Before Analysis)	Hash Values (After Analysis)	Integrity Maintained (Yes/No)
ServerPermission.txt	Text	4f9f8ca963b767a280c92a0baf875dac	4f9f8ca963b767a280c92a0baf875dac	**Yes**
ServerPrincipalData.txt	Text	37d235cd64cf023f42f7dd203ec82ca8	37d235cd64cf023f42f7dd203ec82ca8	**Yes**
ServiceShutDown.txt	Text	3e5bd28f60b6e4d4596dec044fc51e3d	3e5bd28f60b6e4d4596dec044fc51e3d	**Yes**
SymmetricKeys.txt	Text	c741728c72766cda50f5dc5516bf4137	c741728c72766cda50f5dc5516bf4137	**Yes**
SystemDatabasePermissions.txt	Text	c07f0bb38e5588b287dbc86150825ce9	c07f0bb38e5588b287dbc86150825ce9	**Yes**
SystemDatabasePrincipals.txt	Text	a407b6fec499c67c92675df48ef4ca74	a407b6fec499c67c92675df48ef4ca74	**Yes**
SystemDatabaseRoleMembers.txt	Text	202a3f083c695d8077495d7cf4d9a818	202a3f083c695d8077495d7cf4d9a818	**Yes**
SystemLog.txt	Text	e912a940788546da21c09e565ed2e92d	e912a940788546da21c09e565ed2e92d	**Yes**

TableStatstics1.txt	Text	bdd2e17c9b9edd3dfcefc406bccd3e3d	bdd2e17c9b9edd3dfcefc406bccd3e3d	**Yes**
TransactionLog1.txt	Text	0be596a693f3c822a22b16d24195bb12	0be596a693f3c822a22b16d24195bb12	**Yes**
RingBuffer.txt	Text	a7c6d77ddf780352f2675416f1c61ecd	a7c6d77ddf780352f2675416f1c61ecd	**Yes**
RingBuffers.txt	Text	d42a21ae9f5d78ee9bfbaeaab9b774d8	d42a21ae9f5d78ee9bfbaeaab9b774d8	**Yes**
ScSAC.txt	Text	31d4667ab52feb7ca07c6a24b2b0a8e6	31d4667ab52feb7ca07c6a24b2b0a8e6	**Yes**
SecurityLog.txt	Text	c2848cc75e55d289d9e4e0f33e6e14b6	c2848cc75e55d289d9e4e0f33e6e14b6	**Yes**
LoginConfig.txt	Text	d4308329566b650338b1b0de9a89530e	d4308329566b650338b1b0de9a89530e	**Yes**
InitialConnection_01.txt	Text	5ff7382dc28235b691f60f1adc475cbb	5ff7382dc28235b691f60f1adc475cbb	**Yes**
DBO_RFIDdb_tblStat_Date.txt	Text	d0eebf9cb3afae82e7208b9dee954654	d0eebf9cb3afae82e7208b9dee954654	**Yes**
DBO_RFIDdb_tblStat_Tag.txt	Text	d09bf1cc3b410aff8c0be7c2d8fe5653	d09bf1cc3b410aff8c0be7c2d8fe5653	**Yes**
DBO_RFIDdb_tblStat_Value.txt	Text	8d5562f81f430924a50c542f4bb19f08	8d5562f81f430924a50c542f4bb19f08	**Yes**
DBO_RFIDlog_tbl.txt	Text	f00f78af761da4a7b0542539f82c71c2	f00f78af761da4a7b0542539f82c71c2	**Yes**

DBO_RFIDlog_tblStat_Date.txt	Text	4691ddfb0f023e968149af81f1305be5	4691ddfb0f023e968149af81f1305be5	**Yes**
DBO_RFIDlog_tblStat_Tag.txt	Text	705eb130b180a184ead7bf88c4b0b2d4	705eb130b180a184ead7bf88c4b0b2d4	**Yes**
DBO_RFIDlog_tblStat_UserData2.txt	Text	6cf7dcdbcfdd404ac3d12f0e211c1a16	6cf7dcdbcfdd404ac3d12f0e211c1a16	**Yes**
DBO_RFIDlog_tblStat_Value.txt	Text	3aaecfd5ee1048ec439645e8ff1a415e	3aaecfd5ee1048ec439645e8ff1a415e	**Yes**
DBSE_LGNF.txt	Text	26be8852d4f78adc049cd58b38dceb60	26be8852d4f78adc049cd58b38dceb60	**Yes**
ApplicationLog_1.txt	Text	ab1c6e64a8a76dcf52e6bd9fff432fe7	ab1c6e64a8a76dcf52e6bd9fff432fe7	**Yes**
AsymKeys.txt	Text	a41c25701ee8d2fa3e24073810de0d13	a41c25701ee8d2fa3e24073810de0d13	**Yes**
Certificates.txt	Text	e1786da5347bafb6d39a8e2e2d97de52	e1786da5347bafb6d39a8e2e2d97de52	**Yes**
CollationAndDataType.txt	Text	37a07ffe3ab6f936fd77c7642e020551	37a07ffe3ab6f936fd77c7642e020551	**Yes**
ConnectionDetails.txt*	Text	f4785e7b6af42435b5a7a07dabcaf6de	f4785e7b6af42435b5a7a07dabcaf6de	**Yes**
ConnectionDetails_01.txt	Text	8e35ad90088f631f493a8b0f668116e4	8e35ad90088f631f493a8b0f668116e4	**Yes**
DataPgAlloc.txt	Text	d41d8cd98f00b204e9800998ecf8427e	d41d8cd98f00b204e9800998ecf8427e	**Yes**

DBO_RFIDdb_tbl.txt	Text	e5a5c6f5fb2f12d39ef145ae3c649134	e5a5c6f5fb2f12d39ef145ae3c649134	**Yes**
DBO_RFIDdb_tblStat_03.txt	Text	d09bf1cc3b410aff8c0be7c2d8fe5653	d09bf1cc3b410aff8c0be7c2d8fe5653	**Yes**
PhysicalDriveImage	**Folder Name**			
PhysicalDriveImage.dd_audit.log*	Log	05f92e6ba0af514098e6d1b1ad290990	05f92e6ba0af514098e6d1b1ad290990	**Yes**
IDE_Image.E01	EnCase Image	dd1f2ab771001cc793caebd1381355d8	dd1f2ab771001cc793caebd1381355d8	**Yes**
IDE_Image.E02	EnCase Image	0241af9a089ab9745fd88f9820e6df35	0241af9a089ab9745fd88f9820e6df35	**Yes**
IDE_Image.E03	EnCase Image	54ceeef91976a05febc9bd14abf0ec44	54ceeef91976a05febc9bd14abf0ec44	**Yes**
IDE_Image.E04	EnCase Image	0f397e77b25e0cc06c7f947fce75ea40	0f397e77b25e0cc06c7f947fce75ea40	**Yes**
IDE_Image.E05	EnCase Image	40a69a04a918a7262ad22dcd67502898	40a69a04a918a7262ad22dcd67502898	**Yes**

IDE_Image.E06	EnCase Image	eaa9522adb28ee98dac1df906aaa26e9	eaa9522adb28ee98dac1df906aaa26e9	**Yes**
IDE_Image.E01.txt	Text	9b58029460a1e8d19085620409f215d5	9b58029460a1e8d19085620409f215d5	**Yes**
PhysicalDriveImagingCase_01	FTK Log Information	be9fe63eb874a67caa2fbaa3ba066076	be9fe63eb874a67caa2fbaa3ba066076	**Yes**
VolatileSQLDATA	**Folder Name**			
HTML	**Folder Name**			
WFT_MENU.HTM	Web Page	6bf1f7c4269c42dfc2cdc943084f7b4b	6bf1f7c4269c42dfc2cdc943084f7b4b	**Yes**
START.HTM	Web Page	4d3ef39474c2436b270a26700e510d95	4d3ef39474c2436b270a26700e510d95	**Yes**
DataCache.htm	Web Page	f53c4254fd4d19e858c5f921ae220373	f53c4254fd4d19e858c5f921ae220373	**Yes**
PlanCache.htm	Web Page	911bbf966f83a04687ba8833a34a5120	911bbf966f83a04687ba8833a34a5120	**Yes**
Tlog.htm	Web Page	41c77bab1db11c0610cfdb1ebe6728ae	41c77bab1db11c0610cfdb1ebe6728ae	**Yes**
RecentStatements.htm	Web Page	c719dd7e6ddebbc5a3d7302e11185a76	c719dd7e6ddebbc5a3d7302e11185a76	**Yes**

Connections.htm	Web Page	2cea5c9923f35d0465906a22aa67bbfd	2cea5c9923f35d0465906a22aa67bbfd	**Yes**
Sessions.htm	Web Page	d2f90bf630fc5a49e55f0de80c08e4b0	d2f90bf630fc5a49e55f0de80c08e4b0	**Yes**
Logins.htm	Web Page	fe0b6f041ab85c34d586dc002cd06ad8	fe0b6f041ab85c34d586dc002cd06ad8	**Yes**
DbUsers.htm	Web Page	1cddfca585210cb3a355d049399378d4	1cddfca585210cb3a355d049399378d4	**Yes**
DatabaseObjects.htm	Web Page	bd347b43da5ee38e15ecd482f473e197	bd347b43da5ee38e15ecd482f473e197	**Yes**
Triggers.htm	Web Page	dee8d4f690ac5196253059cae0f1bab1	dee8d4f690ac5196253059cae0f1bab1	**Yes**
Jobs.htm	Web Page	9219afe63ffd1e4e308b6216083f387a	9219afe63ffd1e4e308b6216083f387a	**Yes**
JobHistory.htm	Web Page	ea63c5853bb71ac2fcdd36cc132b7f31	ea63c5853bb71ac2fcdd36cc132b7f31	**Yes**
CLR.HTM	Web Page	a670969dc7976defc19b976cd850b1ed	a670969dc7976defc19b976cd850b1ed	**Yes**
Databases.htm	Web Page	f416bf7233e960130fddd1340e5dbc5b	f416bf7233e960130fddd1340e5dbc5b	**Yes**
DBServerInfo.htm	Web Page	c7618040a9d3ba9242842a2f7d4e300f	c7618040a9d3ba9242842a2f7d4e300f	**Yes**
Configuration.htm	Web Page	554c0c23ffb63eccd6a3a3e18d084ca4	554c0c23ffb63eccd6a3a3e18d084ca4	**Yes**
Schemas.htm	Web Page	9b01a502967cbd60726ace5cd227601e	9b01a502967cbd60726ace5cd227601e	**Yes**

Endpoints.htm	Web Page	d23eef6404ca44b2abf23cd51025522e	d23eef6404ca44b2abf23cd51025522e	**Yes**
AutoExecProcs.htm	Web Page	fe962fcfbd57bc62420e26ebe5869d4c	fe962fcfbd57bc62420e26ebe5869d4c	**Yes**
Time.htm	Web Page	b12dd416c44cae9f863eaed2d7443410	b12dd416c44cae9f863eaed2d7443410	**Yes**
ClockHands.htm	Web Page	3d1c61d98b842866905a2ba74178c11f	3d1c61d98b842866905a2ba74178c11f	**Yes**
PCLIP.HTM	Web Page	95b886fc2579c025397977b1688f7173	95b886fc2579c025397977b1688f7173	**Yes**
MEM_P.HTM	Web Page	8f98ecb2391b2c1064dced44b523e326	8f98ecb2391b2c1064dced44b523e326	**Yes**
MEM_D.HTM	Web Page	de7632796cc0475b4154b139d3d4574e	de7632796cc0475b4154b139d3d4574e	**Yes**
C_atime.htm	Web Page	63b9ec1c0055b67fcf64d8be9ae7cd43	63b9ec1c0055b67fcf64d8be9ae7cd43	**Yes**
C_ctime.htm	Web Page	94e433599bc06482d5dcdeaa405f94f7	94e433599bc06482d5dcdeaa405f94f7	**Yes**
C_mtime.htm	Web Page	4ff13f8f3b55cb67af42448531f6478c	4ff13f8f3b55cb67af42448531f6478c	**Yes**
C_mac.htm	Web Page	b4d8b1af2818e68f7c6a666fd55d5ad3	b4d8b1af2818e68f7c6a666fd55d5ad3	**Yes**
PSINFO.HTM	Web Page	5f9dbf2f0c7e01cd972b300bae0f8748	5f9dbf2f0c7e01cd972b300bae0f8748	**Yes**
HOSTNAME.HTM	Web Page	884632e5f6dbc750b13e6b028ef0c02d	884632e5f6dbc750b13e6b028ef0c02d	**Yes**

UNAME.HTM	Web Page	f78e203e3479254600b8be2e0cbed42e	f78e203e3479254600b8be2e0cbed42e	**Yes**
VER.HTM	Web Page	5e0a1dd0a0bb0375fd4b229223e84cda	5e0a1dd0a0bb0375fd4b229223e84cda	**Yes**
ENVIRONM.HTM	Web Page	ab1cad60791f2dfab972a674465d963d	ab1cad60791f2dfab972a674465d963d	**Yes**
UPTIME.HTM	Web Page	b7456d103cbf5e49f1574a845ef79218	b7456d103cbf5e49f1574a845ef79218	**Yes**
UPTIME_H.HTM	Web Page	4975585259a0481f4323ac25e6acdcf9	4975585259a0481f4323ac25e6acdcf9	**Yes**
WHOAMI.HTM	Web Page	65def1693a2ddbe20d365f8b29399857	65def1693a2ddbe20d365f8b29399857	**Yes**
NETDOM.HTM	Web Page	924195e7ff4de13f4f07bfc1599929be	924195e7ff4de13f4f07bfc1599929be	**Yes**
NETUSER.HTM	Web Page	4f0337a047d690382370317266069c59	4f0337a047d690382370317266069c59	**Yes**
NETGROUP.HTM	Web Page	4af52dbb00f3b4f9de9b2bea14b244bc	4af52dbb00f3b4f9de9b2bea14b244bc	**Yes**
NETLGRP.HTM	Web Page	8147e2bea1e81da7733766c35269c93a	8147e2bea1e81da7733766c35269c93a	**Yes**
NETACCT.HTM	Web Page	ad1ed006744e205ad27c196a10db8ca4	ad1ed006744e205ad27c196a10db8ca4	**Yes**
netacctdom.htm	Web Page	376b0014907eb503118c604b967feeb0	376b0014907eb503118c604b967feeb0	**Yes**
AUDITPOL.HTM	Web Page	86df1ffa937da57655981a8cc17a340d	86df1ffa937da57655981a8cc17a340d	**Yes**

PSLIST.HTM	Web Page	79dbea76b9829726a0d7ef704b12774b	79dbea76b9829726a0d7ef704b12774b	**Yes**
LISTDLLS.HTM	Web Page	acddf8b75be89c52bc8b987c80858416	acddf8b75be89c52bc8b987c80858416	**Yes**
PS.HTM	Web Page	91f4ce8ac2039d5e6f7716dc1e0130ab	91f4ce8ac2039d5e6f7716dc1e0130ab	**Yes**
PSTAT.HTM	Web Page	a9853633741f6a687e0ff281f8e8b018	a9853633741f6a687e0ff281f8e8b018	**Yes**
TLIST_V.HTM	Web Page	c82f579a95f60ef09cfde1315af9eb30	c82f579a95f60ef09cfde1315af9eb30	**Yes**
TLIST_S.HTM	Web Page	da213c53c804efbfdb3d904c8bea7652	da213c53c804efbfdb3d904c8bea7652	**Yes**
TLIST_C.HTM	Web Page	614831108114806947ba7fc95817cdc0	614831108114806947ba7fc95817cdc0	**Yes**
CMDLINE.HTM	Web Page	e973cf07176d49fba0d156ac1661b4f6	e973cf07176d49fba0d156ac1661b4f6	**Yes**
HANDLE.HTM	Web Page	a8fd25f379cda2a73d9e01ec1a5f4c4d	a8fd25f379cda2a73d9e01ec1a5f4c4d	**Yes**
procinterrogate.htm	Web Page	86589fe6b57cb113c3779bdc117c56f1	86589fe6b57cb113c3779bdc117c56f1	**Yes**
psservice.htm	Web Page	bc1976560904b5269f3866184c666f15	bc1976560904b5269f3866184c666f15	**Yes**
sc_query_ex.htm	Web Page	32c72d6bca4ec46b5dcc4a5d72d517a3	32c72d6bca4ec46b5dcc4a5d72d517a3	**Yes**
NETSTART.HTM	Web Page	3d782c8667f2f5403117bbae63a468f8	3d782c8667f2f5403117bbae63a468f8	**Yes**

SRVC.HTM	Web Page	d55407e53427af5b58e37be7ee2b7ab6	d55407e53427af5b58e37be7ee2b7ab6	**Yes**
TaskList_v.htm	Web Page	b70a085c1191aca409f0dcea2d8a260c	b70a085c1191aca409f0dcea2d8a260c	**Yes**
TaskList_svc.htm	Web Page	53eb68f191f91a5f85af6c74ebc86aa8	53eb68f191f91a5f85af6c74ebc86aa8	**Yes**
DRIVERS.HTM	Web Page	b87d89ade181019c3ce4ca1f7ed13f4b	b87d89ade181019c3ce4ca1f7ed13f4b	**Yes**
IPCONFIG.HTM	Web Page	60264e9d8da4ea0f3223acec4fc6ee34	60264e9d8da4ea0f3223acec4fc6ee34	**Yes**
IPLIST.HTM	Web Page	076ee1eaab219df451b23c6dbcf57f67	076ee1eaab219df451b23c6dbcf57f67	**Yes**
ARP.HTM	Web Page	37397999d7d058e14feadab3ac2ff6b0	37397999d7d058e14feadab3ac2ff6b0	**Yes**
RTABLE.HTM	Web Page	30615c2e84547db91a77bd11c7f0ee76	30615c2e84547db91a77bd11c7f0ee76	**Yes**
NETSTAT.HTM	Web Page	dbeac920688cecc9bf775b9c74f0c1ca	dbeac920688cecc9bf775b9c74f0c1ca	**Yes**
NETSTATN.HTM	Web Page	9329d66d41228831c1408a1f255db6a4	9329d66d41228831c1408a1f255db6a4	**Yes**
FPORT_P.HTM	Web Page	bb5127d25439d1b50a305b1cf865fe7a	bb5127d25439d1b50a305b1cf865fe7a	**Yes**
FPORT_A.HTM	Web Page	4753417c7c8129d12755395560f24d19	4753417c7c8129d12755395560f24d19	**Yes**
openports.htm	Web Page	1658c56ba5517e2bcb730fc20093a888	1658c56ba5517e2bcb730fc20093a888	**Yes**

IPXROUTE.HTM	Web Page	97d4e1dd9621b68e55c1e7b292462634	97d4e1dd9621b68e55c1e7b292462634	**Yes**
NBTSTATN.HTM	Web Page	452dfac476bd1e13a9f03a64c2dd6025	452dfac476bd1e13a9f03a64c2dd6025	**Yes**
NBTSTATC.HTM	Web Page	7a39093e22b2c228ab8d555d9b4c753f	7a39093e22b2c228ab8d555d9b4c753f	**Yes**
NBTSTATS.HTM	Web Page	ac4def77ca515ceab894ee19d3e8b863	ac4def77ca515ceab894ee19d3e8b863	**Yes**
HUNT.HTM	Web Page	d59d46c7da7bed9e79c68a4c5f7bdfe8	d59d46c7da7bed9e79c68a4c5f7bdfe8	**Yes**
NETSHARE.HTM	Web Page	52aae9b29695344da8ddc0f0dce05dc4	52aae9b29695344da8ddc0f0dce05dc4	**Yes**
NETUSE.HTM	Web Page	3f296f71eda5a090a372b8c6fdf3e6e8	3f296f71eda5a090a372b8c6fdf3e6e8	**Yes**
NETVIEW.HTM	Web Page	cfce3727bed1ec8c7294d014bd400ae9	cfce3727bed1ec8c7294d014bd400ae9	**Yes**
NETSESSI.HTM	Web Page	a4922fa39f9fc802e373cc969e6d7635	a4922fa39f9fc802e373cc969e6d7635	**Yes**
NDIS.HTM	Web Page	3df33fcd8526fe114bbc1fe420cdbd83	3df33fcd8526fe114bbc1fe420cdbd83	**Yes**
promiscdetect.htm	Web Page	221bc9b76c231fb1e947e779f863a02b	221bc9b76c231fb1e947e779f863a02b	**Yes**
psloggedon.htm	Web Page	ae76742479d2f0406c008c2d58a464b6	ae76742479d2f0406c008c2d58a464b6	**Yes**
netusers_local.htm	Web Page	6b21af94ab439f4b9853a44cc3f95aaf	6b21af94ab439f4b9853a44cc3f95aaf	**Yes**

netusers_local_history.htm	Web Page	00d8bdf28c0673483f39e2bf4b010f99	00d8bdf28c0673483f39e2bf4b010f99	**Yes**
SUCCESS.HTM	Web Page	5da3161760f7b453690885d31f203b82	5da3161760f7b453690885d31f203b82	**Yes**
FAILED.HTM	Web Page	3eb913f1254157d414366b076fd49a23	3eb913f1254157d414366b076fd49a23	**Yes**
INTERACT.HTM	Web Page	f2fa3fb08a91ddfd6f00b5343b4f4483	f2fa3fb08a91ddfd6f00b5343b4f4483	**Yes**
REMOTE.HTM	Web Page	0938aacd71a620b109690296d51348ba	0938aacd71a620b109690296d51348ba	**Yes**
SYSLOG.HTM	Web Page	b3e81a2d1f78bb58278986bd03187df4	b3e81a2d1f78bb58278986bd03187df4	**Yes**
APPLOG.HTM	Web Page	96898f0d4eb344cd6f15c91eaa929fa7	96898f0d4eb344cd6f15c91eaa929fa7	**Yes**
SECLOG.HTM	Web Page	276339791a0e9dbde562d7b474dddb8e	276339791a0e9dbde562d7b474dddb8e	**Yes**
EVTLOG.HTM	Web Page	c3aa752b275fe1720b351e0f7d74f351	c3aa752b275fe1720b351e0f7d74f351	**Yes**
LOG_SYS.HTM	Web Page	2b7a2354330161e84e102e2a3a91ffbf	2b7a2354330161e84e102e2a3a91ffbf	**Yes**
LOG_APP.HTM	Web Page	a166009966815b486c1660d6924ba01f	a166009966815b486c1660d6924ba01f	**Yes**
log_Sec.htm	Web Page	e9d558e387fc220866b3da9826acb474	e9d558e387fc220866b3da9826acb474	**Yes**
C_ntfsinfo.htm	Web Page	cc43bf083bbac6e06dc7642c228ca559	cc43bf083bbac6e06dc7642c228ca559	**Yes**

PSFILE.HTM	Web Page	d6d58ad5914ef24b8e71ab3aa036cd3e	d6d58ad5914ef24b8e71ab3aa036cd3e	**Yes**
NETFILE.HTM	Web Page	76b383a9ed49cbb0a30af9c4c7e19642	76b383a9ed49cbb0a30af9c4c7e19642	**Yes**
C_filestg.htm	Web Page	bbb3afaa88f73986921ab8af42fa163e	bbb3afaa88f73986921ab8af42fa163e	**Yes**
C_hfind.htm	Web Page	6f2ca064a84a605e699bd2bed21689f3	6f2ca064a84a605e699bd2bed21689f3	**Yes**
C_hidden.htm	Web Page	7af8d6e75ffd1c6e6ed7b6d5b350b7d1	7af8d6e75ffd1c6e6ed7b6d5b350b7d1	**Yes**
C_streams.htm	Web Page	ddae288f7ad3fa459b77d1580a453782	ddae288f7ad3fa459b77d1580a453782	**Yes**
C_sfind.htm	Web Page	752bcd075879ab05e582c2bafbd1019a	752bcd075879ab05e582c2bafbd1019a	**Yes**
C_efsinfo.htm	Web Page	8f8a235b3c090cd0f72c2b2eb21be226	8f8a235b3c090cd0f72c2b2eb21be226	**Yes**
RECENT.HTM	Web Page	c27313fe31892eee745ee95a6078a79d	c27313fe31892eee745ee95a6078a79d	**Yes**
C_recycle.htm	Web Page	f10efef39d9e768e3b3ba80dcc4f792b	f10efef39d9e768e3b3ba80dcc4f792b	**Yes**
PREFETCH.HTM	Web Page	c4b22c721ec1aa70314c108bb0924b18	c4b22c721ec1aa70314c108bb0924b18	**Yes**
C_freesp.htm	Web Page	651bccc73a3ec0c715ee2b915f68ee9d	651bccc73a3ec0c715ee2b915f68ee9d	**Yes**
AUTORUNS.HTM	Web Page	f5b2c94df50b21f376ee5a23a05fcca5	f5b2c94df50b21f376ee5a23a05fcca5	**Yes**

AUTOEXEC.HTM	Web Page	59206afeb091abbec7bd1c8f54e575a7	59206afeb091abbec7bd1c8f54e575a7	**Yes**
WIN_INI.HTM	Web Page	5ab1a502b1fc79402851ccbbb7b4ed4a	5ab1a502b1fc79402851ccbbb7b4ed4a	**Yes**
SYS_INI.HTM	Web Page	b0404db2b8ab003c5fe334d5573b5705	b0404db2b8ab003c5fe334d5573b5705	**Yes**
WINSTART.HTM	Web Page	8a785ca570f0aed312216e01ce112b77	8a785ca570f0aed312216e01ce112b77	**Yes**
INIT_INI.HTM	Web Page	2fa3b3b46f122b085ee5115829d0239e	2fa3b3b46f122b085ee5115829d0239e	**Yes**
STARTUP.HTM	Web Page	a907d696b5f75ca6d7b15af5b854716c	a907d696b5f75ca6d7b15af5b854716c	**Yes**
user_startup.htm	Web Page	c485e8c09304f8d231d1b11dd6c788a8	c485e8c09304f8d231d1b11dd6c788a8	**Yes**
TASKS.HTM	Web Page	6e2d8754e8439627d257cc6a69cb5c45	6e2d8754e8439627d257cc6a69cb5c45	**Yes**
AT.HTM	Web Page	abdaea6c66c986d70c5540c686ab9ad8	abdaea6c66c986d70c5540c686ab9ad8	**Yes**
SCHTASKS.HTM	Web Page	899eea950edb07903d43df49a3ca0496	899eea950edb07903d43df49a3ca0496	**Yes**
HKLM_R.HTM	Web Page	bebabaa2a860214d17d8304b65ff3590	bebabaa2a860214d17d8304b65ff3590	**Yes**
HKLM_RO.HTM	Web Page	a3f1d4f77f7ac928f288a0e50eaa6dfa	a3f1d4f77f7ac928f288a0e50eaa6dfa	**Yes**
HKLM_ROX.HTM	Web Page	261b895631d0db0b501cf5b54884ae0c	261b895631d0db0b501cf5b54884ae0c	**Yes**

HKLM_RS.HTM	Web Page	f756a379df8579c22cecaac3fc75bbdd	f756a379df8579c22cecaac3fc75bbdd	**Yes**
HKLM_RSO.HTM	Web Page	0de98750f47fad1105788b69881792ad	0de98750f47fad1105788b69881792ad	**Yes**
hklm_scripts.htm	Web Page	c86779dbdf7e7a8b0c8362cae86a05a1	c86779dbdf7e7a8b0c8362cae86a05a1	**Yes**
hklm_expl_run.htm	Web Page	3a3e1639160eb64837572d7adadb329c	3a3e1639160eb64837572d7adadb329c	**Yes**
HKCU_R.HTM	Web Page	89f4de3fc5b222f63aa86624c6c87da7	89f4de3fc5b222f63aa86624c6c87da7	**Yes**
HKCU_RO.HTM	Web Page	b096b4db3a315e0870fd154ccc5e8553	b096b4db3a315e0870fd154ccc5e8553	**Yes**
HKCU_ROX.HTM	Web Page	4380d8f2014a5c35bf59df30d079f0bb	4380d8f2014a5c35bf59df30d079f0bb	**Yes**
HKCU_RS.HTM	Web Page	77e38e4c9531a15d6ec4ba85a9a8380a	77e38e4c9531a15d6ec4ba85a9a8380a	**Yes**
HKCU_RSO.HTM	Web Page	4b12e8f88bac5d29408d8d527e7d6eb4	4b12e8f88bac5d29408d8d527e7d6eb4	**Yes**
hkcu_scripts.htm	Web Page	43acf347d1bd6a412ed67a2d233161b6	43acf347d1bd6a412ed67a2d233161b6	**Yes**
hkcu_expl_run.htm	Web Page	b1f50e3968b3dd72474899a53a4ea544	b1f50e3968b3dd72474899a53a4ea544	**Yes**
GPLIST.HTM	Web Page	9c8299ef8518081fbe0fef1c1bef44cd	9c8299ef8518081fbe0fef1c1bef44cd	**Yes**
GPUSER.HTM	Web Page	714bdb6449a71e9376f335785f42ae14	714bdb6449a71e9376f335785f42ae14	**Yes**

GPSYS.HTM	Web Page	f3c86becc5bd97408c6c98f105a9033d	f3c86becc5bd97408c6c98f105a9033d	**Yes**
RUN_HIST.HTM	Web Page	203f83a77141742fa6577b28be686953	203f83a77141742fa6577b28be686953	**Yes**
RCNT_DOC.HTM	Web Page	7a4e4fac9eb93bdd1d20c17e032d8d75	7a4e4fac9eb93bdd1d20c17e032d8d75	**Yes**
LASTSAVE.HTM	Web Page	c6271e4393d3a09d3ad0cd688358d660	c6271e4393d3a09d3ad0cd688358d660	**Yes**
INSTALLH.HTM	Web Page	de48e6481ccee6dabb54a59cba945f3a	de48e6481ccee6dabb54a59cba945f3a	**Yes**
REGDMP.HTM	Web Page	51e651a9495ad2a0183d382068a3edfb	51e651a9495ad2a0183d382068a3edfb	**Yes**
hklm_safemin.htm	Web Page	21c76f57dc5d19a7eb353b6d563031cd	21c76f57dc5d19a7eb353b6d563031cd	**Yes**
hklm_safenet.htm	Web Page	5727676a6ea203a955b4fc1a4d14048f	5727676a6ea203a955b4fc1a4d14048f	**Yes**
IEHV.HTM	Web Page	8b4e210bfa88f8426dbb9cb5ef424702	8b4e210bfa88f8426dbb9cb5ef424702	**Yes**
TYPE_URL.HTM	Web Page	76be91706908e6636a4e91d099510087	76be91706908e6636a4e91d099510087	**Yes**
SEARCH_H.HTM	Web Page	a618290a3dacf05978ad39ec6f2d742f	a618290a3dacf05978ad39ec6f2d742f	**Yes**
pstoreview.htm	Web Page	88d7bf3078be7af65b27ec3814b812aa	88d7bf3078be7af65b27ec3814b812aa	**Yes**
DOSKEY.HTM	Web Page	1458731ac76bf6587b9abf60b5e6f730	1458731ac76bf6587b9abf60b5e6f730	**Yes**

MDM.HTM	Web Page	a5e04ce095305295ae2d230d8a624a63	a5e04ce095305295ae2d230d8a624a63	**Yes**
ROOTKIT.HTM	Web Page	db2003ef0e28276f6a60ff1a8e7baa8c	db2003ef0e28276f6a60ff1a8e7baa8c	**Yes**
END.HTM	Web Page	7c7181f3c4eae5eae77ea0bdf548a73a	7c7181f3c4eae5eae77ea0bdf548a73a	**Yes**
WFT_HEAD.HTM	Web Page	d3ff0a02281099ce200b8cb3c44b0fcf	d3ff0a02281099ce200b8cb3c44b0fcf	**Yes**
wft_splash.js	Java Script	90cc848821aefb7e11d01731175a22c4	90cc848821aefb7e11d01731175a22c4	**Yes**
wft_splash.htm	Web Page	75689ffb8e54fa562c2cb2f44051a1e2	75689ffb8e54fa562c2cb2f44051a1e2	**Yes**
WFT_LINK.HTM	Web Page	8b51153eb5ed9aad154bd8b1873873b1	8b51153eb5ed9aad154bd8b1873873b1	**Yes**
WFT_HELP.HTM	Web Page	e37c3b9e98c1b86646e56c7602539800	e37c3b9e98c1b86646e56c7602539800	**Yes**
WFT_TOOL.HTM	Web Page	9df253f6612f7bfe34c8830041155b4c	9df253f6612f7bfe34c8830041155b4c	**Yes**
WFT_MAIN.HTM	Web Page	06024a3756b6a1c2594ca5ff643e3108	06024a3756b6a1c2594ca5ff643e3108	**Yes**
WFT_CFG.HTM	Web Page	df567d7e0808ab01c83d7962ae056fb5	df567d7e0808ab01c83d7962ae056fb5	**Yes**
WFT_HASH.HTM	Web Page	f1a040556a759801deb5eb19d8e6dbb9	f1a040556a759801deb5eb19d8e6dbb9	**Yes**
WFT_LOG.HTM	Web Page	a403e6d35cec0fbac30519c7abf1820f	a403e6d35cec0fbac30519c7abf1820f	**Yes**

IMG	**Folder Name**			
fmlogo_sm.png	Portable Networks Graphic	51f885b6ab342a12aa6d050c556c0e3c	51f885b6ab342a12aa6d050c556c0e3c	**Yes**
WFT.ICO	Windows Icon	c37e8b76f761570353fac39eab67ae7b	c37e8b76f761570353fac39eab67ae7b	**Yes**
MAIL.GIF	GIF	38477e98fb3ed60b82906c7dce8da645	38477e98fb3ed60b82906c7dce8da645	**Yes**
HELP.GIF	GIF	dabe723a558a1f324cf367c27ee30258	dabe723a558a1f324cf367c27ee30258	**Yes**
BAD.GIF	GIF	debf875a10db19e19a261b3f82ed3dad	debf875a10db19e19a261b3f82ed3dad	**Yes**
MISSING.GIF	GIF	dd83b55dc42dbfb48ec230ce4d8653ea	dd83b55dc42dbfb48ec230ce4d8653ea	**Yes**
OK.GIF	GIF	a49085e88fe9d8c1c31928217e15dbc0	a49085e88fe9d8c1c31928217e15dbc0	**Yes**
UNKNOWN.GIF	GIF	04108acb26f82bb7c439d292ee7abb9e	04108acb26f82bb7c439d292ee7abb9e	**Yes**
TXT	**Folder Name**			

WFT_CFG.TXT	Text	1d378b80f8e0e7ea4f35ec8b92989cd1	1d378b80f8e0e7ea4f35ec8b92989cd1	**Yes**
WFT_LOG.TXT	Text	1a72e21e9657e9458731e70c60736513	1a72e21e9657e9458731e70c60736513	**Yes**
WFT_HASH.TXT	Text	c5e1f3d439de301813665f1590d540e4	c5e1f3d439de301813665f1590d540e4	**Yes**
START.TXT	Text	bc93a20c1717175a5202555f516ed202	bc93a20c1717175a5202555f516ed202	**Yes**
DataCache.txt	Text	ab95bd58a9cd21187a4df339e8040a6e	ab95bd58a9cd21187a4df339e8040a6e	**Yes**
PlanCache.txt	Text	af3692b98c409a707da4818ed8b70a2f	af3692b98c409a707da4818ed8b70a2f	**Yes**
Tlog.txt	Text	4f3f10ebee09a5e8dd7ee65d4bfe3de8	4f3f10ebee09a5e8dd7ee65d4bfe3de8	**Yes**
RecentStatements.txt	Text	194cafe3e27faf49f92c9cd23b5f7dd3	194cafe3e27faf49f92c9cd23b5f7dd3	**Yes**
Connections.txt	Text	a7a43c52521f85bdd6683d9e0e4a00bd	a7a43c52521f85bdd6683d9e0e4a00bd	**Yes**
Sessions.txt	Text	37f47a54ed052372f512cda52ab33335	37f47a54ed052372f512cda52ab33335	**Yes**
Logins.txt	Text	b1a6082cb887d2de069e59f56b63a309	b1a6082cb887d2de069e59f56b63a309	**Yes**
DbUsers.txt	Text	99027343e1ca315527beaab193a19460	99027343e1ca315527beaab193a19460	**Yes**
DatabaseObjects.txt	Text	d20dec1f09fe06e10e1305ad89649e3c	d20dec1f09fe06e10e1305ad89649e3c	**Yes**

Triggers.txt	Text	c4bc1039910d88dddc28f115c79b7936	c4bc1039910d88dddc28f115c79b7936	**Yes**
Jobs.txt	Text	5991a01962426d20e4415920a6fb4c31	5991a01962426d20e4415920a6fb4c31	**Yes**
JobHistory.txt	Text	4a2c5101bd3258d6c6eccd06a9d286a9	4a2c5101bd3258d6c6eccd06a9d286a9	**Yes**
CLR.TXT	Text	aa14dc79708388e4e1797af2eda8d4d0	aa14dc79708388e4e1797af2eda8d4d0	**Yes**
Databases.txt	Text	ee8a1a7d7e32043fdd6468dfc85f24bf	ee8a1a7d7e32043fdd6468dfc85f24bf	**Yes**
DBServerInfo.txt	Text	33d139d0230377d50b1d51f56ea8d5b9	33d139d0230377d50b1d51f56ea8d5b9	**Yes**
Configuration.txt	Text	88f65aa67a74cb9218becad1b9153ced	88f65aa67a74cb9218becad1b9153ced	**Yes**
Schemas.txt	Text	ecf3cb5d8f259ad36814696c680bbb81	ecf3cb5d8f259ad36814696c680bbb81	**Yes**
Endpoints.txt	Text	9c6239d845bcdc15c9504ff7d2b303a9	9c6239d845bcdc15c9504ff7d2b303a9	**Yes**
AutoExecProcs.txt	Text	72cbadae0a1f9ff34d4a35a9ab26bdb8	72cbadae0a1f9ff34d4a35a9ab26bdb8	**Yes**
Time.txt	Text	4e7daac41474dcc469faf57620ee1a8e	4e7daac41474dcc469faf57620ee1a8e	**Yes**
ClockHands.txt	Text	d28a565258a6c5f5863014fba380d1e9	d28a565258a6c5f5863014fba380d1e9	**Yes**
END.TXT	Text	a5f255ddbd261cfed9dcb16263317ff5	a5f255ddbd261cfed9dcb16263317ff5	**Yes**

PCLIP.TXT	Text	037130bf88e524d8f61aa95eb8af3c3b	037130bf88e524d8f61aa95eb8af3c3b	**Yes**
MEM_P.TXT	Text	a7537e6630a8d7dc8f1a8455a8fdc68e	a7537e6630a8d7dc8f1a8455a8fdc68e	**Yes**
MEM_D.TXT	Text	c9dd6e1ffa4782613e65c87312f0e4bb	c9dd6e1ffa4782613e65c87312f0e4bb	**Yes**
C_atime.txt	Text	5f45b5367c7ed885270ea969851cb439	5f45b5367c7ed885270ea969851cb439	**Yes**
C_ctime.txt	Text	5f45b5367c7ed885270ea969851cb439	5f45b5367c7ed885270ea969851cb439	**Yes**
C_mtime.txt	Text	5f45b5367c7ed885270ea969851cb439	5f45b5367c7ed885270ea969851cb439	**Yes**
C_mac.txt	Text	76dcf11d5c62e135333988a177eb49ba	76dcf11d5c62e135333988a177eb49ba	**Yes**
PSINFO.TXT	Text	1709bdb4e50d6884682a5077cab5507b	1709bdb4e50d6884682a5077cab5507b	**Yes**
HOSTNAME.TXT	Text	61c47a3d9f316c4dabbad16af15ee48a	61c47a3d9f316c4dabbad16af15ee48a	**Yes**
UNAME.TXT	Text	e80251575bcd586a4c633d296bbb6f79	e80251575bcd586a4c633d296bbb6f79	**Yes**
VER.TXT	Text	bd434867c60c1d657d9f6caf64d80ff9	bd434867c60c1d657d9f6caf64d80ff9	**Yes**
ENVIRONM.TXT	Text	229af64fd3e6a9f8a4c4ce3ac16359c3	229af64fd3e6a9f8a4c4ce3ac16359c3	**Yes**
UPTIME.TXT	Text	3d823d7cc9ea0ed4d6e4a20d88395cc3	3d823d7cc9ea0ed4d6e4a20d88395cc3	**Yes**

UPTIME_H.TXT	Text	3d823d7cc9ea0ed4d6e4a20d88395cc3	3d823d7cc9ea0ed4d6e4a20d88395cc3	**Yes**
WHOAMI.TXT	Text	654ff4d9ba4050b6495a62c636f45bd6	654ff4d9ba4050b6495a62c636f45bd6	**Yes**
NETDOM.TXT	Text	b82fe5d8926af2831badde7174ddfeec	b82fe5d8926af2831badde7174ddfeec	**Yes**
NETUSER.TXT	Text	655988d2a3e42136ab57887597f99545	655988d2a3e42136ab57887597f99545	**Yes**
NETGROUP.TXT	Text	c5fc9c8ce7e90103bbb77435370bf16c	c5fc9c8ce7e90103bbb77435370bf16c	**Yes**
NETLGRP.TXT	Text	e6d74d46aecd969282a7362b8998a746	e6d74d46aecd969282a7362b8998a746	**Yes**
NETACCT.TXT	Text	83a23e9992174906473e0a739aa4dd6a	83a23e9992174906473e0a739aa4dd6a	**Yes**
netacctdom.txt	Text	641e4b46f39dc9f57fe69fb0884016d7	641e4b46f39dc9f57fe69fb0884016d7	**Yes**
AUDITPOL.TXT	Text	963d3ef25e8f338e59b9a30c3a1b1187	963d3ef25e8f338e59b9a30c3a1b1187	**Yes**
PSLIST.TXT	Text	5f2a642d67eb471cd74d5529c519ef31	5f2a642d67eb471cd74d5529c519ef31	**Yes**
LISTDLLS.TXT	Text	7e2c1a0055336da76a6ca8d03fa4b015	7e2c1a0055336da76a6ca8d03fa4b015	**Yes**
PS.TXT	Text	ef6e3f3f2e976685811c3a89f8cd0eb6	ef6e3f3f2e976685811c3a89f8cd0eb6	**Yes**
PSTAT.TXT	Text	496e9b6cb213a8e50d587ac081b1dcd7	496e9b6cb213a8e50d587ac081b1dcd7	**Yes**

TLIST_V.TXT	Text	cdf2d49e302522e370a4918e70e81d88	cdf2d49e302522e370a4918e70e81d88	**Yes**
TLIST_S.TXT	Text	9b5dd3f38833885843bcc2206de0e50d	9b5dd3f38833885843bcc2206de0e50d	**Yes**
TLIST_C.TXT	Text	470422fd9f503565b1ba186311c94eaf	470422fd9f503565b1ba186311c94eaf	**Yes**
CMDLINE.TXT	Text	b44c85c704337f007fb47edda814865d	b44c85c704337f007fb47edda814865d	**Yes**
HANDLE.TXT	Text	9b80dd8a3300571406be1429f663778f	9b80dd8a3300571406be1429f663778f	**Yes**
procinterrogate.txt	Text	5105e5160bc35eb5d56a4e2829795e7d	5105e5160bc35eb5d56a4e2829795e7d	**Yes**
psservice.txt	Text	6ee3210d36a0988409608d8823a72f9a	6ee3210d36a0988409608d8823a72f9a	**Yes**
sc_query_ex.txt	Text	1590bff76fd1c79e28a6862b93e49eeb	1590bff76fd1c79e28a6862b93e49eeb	**Yes**
NETSTART.TXT	Text	1b3142d14abe08fb5956cf273e926e48	1b3142d14abe08fb5956cf273e926e48	**Yes**
SRVC.TXT	Text	8e74213360ea0eaf0342b055528a7574	8e74213360ea0eaf0342b055528a7574	**Yes**
TaskList_v.txt	Text	96b45c89c8f164b9e9ab255a0d00e3ce	96b45c89c8f164b9e9ab255a0d00e3ce	**Yes**
TaskList_svc.txt	Text	5b2e0d4c2359c98fa6f76d61a6c7e59d	5b2e0d4c2359c98fa6f76d61a6c7e59d	**Yes**
DRIVERS.TXT	Text	728f0b3a7c6de4b7b9d838f9c0afab44	728f0b3a7c6de4b7b9d838f9c0afab44	**Yes**

IPCONFIG.TXT	Text	9256b6c363c4296005bead1298bd643a	9256b6c363c4296005bead1298bd643a	**Yes**
IPLIST.TXT	Text	fc52f40362493703f44b895fd7ff9149	fc52f40362493703f44b895fd7ff9149	**Yes**
ARP.TXT	Text	dfb5dddfc15a7925629205e6edf8562d	dfb5dddfc15a7925629205e6edf8562d	**Yes**
RTABLE.TXT	Text	30fbe777154b4875b64912c34de745ac	30fbe777154b4875b64912c34de745ac	**Yes**
NETSTAT.TXT	Text	620ad12bd3dfb5b47bfc128b13ddefcd	620ad12bd3dfb5b47bfc128b13ddefcd	**Yes**
NETSTATN.TXT	Text	52ee5eeacdfd2ba082f52d9cfea10708	52ee5eeacdfd2ba082f52d9cfea10708	**Yes**
FPORT_P.TXT	Text	9fc5fd6a033d6b3617840ecf63ded76d	9fc5fd6a033d6b3617840ecf63ded76d	**Yes**
FPORT_A.TXT	Text	9fc5fd6a033d6b3617840ecf63ded76d	9fc5fd6a033d6b3617840ecf63ded76d	**Yes**
openports.txt	Text	4d06b65d83f276728e817c02fd95c40c	4d06b65d83f276728e817c02fd95c40c	**Yes**
IPXROUTE.TXT	Text	581c52ab1684e14b67bdb45c4528f992	581c52ab1684e14b67bdb45c4528f992	**Yes**
NBTSTATN.TXT	Text	3dc9c248379c160466f583fc26752962	3dc9c248379c160466f583fc26752962	**Yes**
NBTSTATC.TXT	Text	3dc9c248379c160466f583fc26752962	3dc9c248379c160466f583fc26752962	**Yes**
NBTSTATS.TXT	Text	647578d95b7f3b11b6ccf833e296701e	647578d95b7f3b11b6ccf833e296701e	**Yes**

HUNT.TXT	Text	99553ccd8e14cae6db789bce98cfcb73	99553ccd8e14cae6db789bce98cfcb73	**Yes**
NETSHARE.TXT	Text	32f60215d6f7ce348c66fe6d6afb7081	32f60215d6f7ce348c66fe6d6afb7081	**Yes**
NETUSE.TXT	Text	9d8780af2919a4f1b1532c208720ed14	9d8780af2919a4f1b1532c208720ed14	**Yes**
NETVIEW.TXT	Text	768165e0abf16bf3056836d5431a7296	768165e0abf16bf3056836d5431a7296	**Yes**
NETSESSI.TXT	Text	768165e0abf16bf3056836d5431a7296	768165e0abf16bf3056836d5431a7296	**Yes**
NDIS.TXT	Text	6d4f36367cb6120ba92adaa77da42bea	6d4f36367cb6120ba92adaa77da42bea	**Yes**
promiscdetect.txt	Text	e9b3135e5cd46f2e434e67a0de631f95	e9b3135e5cd46f2e434e67a0de631f95	**Yes**
psloggedon.txt	Text	b2e118703df65508e00e4c17c7651ab9	b2e118703df65508e00e4c17c7651ab9	**Yes**
netusers_local.txt	Text	ab20a9482744e11727c791bc855717b8	ab20a9482744e11727c791bc855717b8	**Yes**
netusers_local_history.txt	Text	c4da0f6d85f641b3e5a0f065aedb2701	c4da0f6d85f641b3e5a0f065aedb2701	**Yes**
SUCCESS.TXT	Text	2428169b31c96fe4a767215000aa0008	2428169b31c96fe4a767215000aa0008	**Yes**
FAILED.TXT	Text	2428169b31c96fe4a767215000aa0008	2428169b31c96fe4a767215000aa0008	**Yes**
INTERACT.TXT	Text	2428169b31c96fe4a767215000aa0008	2428169b31c96fe4a767215000aa0008	**Yes**

REMOTE.TXT	Text	2428169b31c96fe4a767215000aa0008	2428169b31c96fe4a767215000aa0008	**Yes**
SYSLOG.TXT	Text	7583ee9e2e28231e98a616966696f439	7583ee9e2e28231e98a616966696f439	**Yes**
APPLOG.TXT	Text	4a25f00467d15fe9fae2777769518518	4a25f00467d15fe9fae2777769518518	**Yes**
SECLOG.TXT	Text	d41d8cd98f00b204e9800998ecf8427e	d41d8cd98f00b204e9800998ecf8427e	**Yes**
EVTLOG.TXT	Text	d7f5e4736e3c7e2bc580e63814b73a2c	d7f5e4736e3c7e2bc580e63814b73a2c	**Yes**
LOG_SYS.TXT	Text	d7f5e4736e3c7e2bc580e63814b73a2c	d7f5e4736e3c7e2bc580e63814b73a2c	**Yes**
LOG_APP.TXT	Text	d7f5e4736e3c7e2bc580e63814b73a2c	d7f5e4736e3c7e2bc580e63814b73a2c	**Yes**
log_Sec.txt	Text	d7f5e4736e3c7e2bc580e63814b73a2c	d7f5e4736e3c7e2bc580e63814b73a2c	**Yes**
C_ntfsinfo.txt	Text	e9f22f0da88cac4bc36566d480906fd4	e9f22f0da88cac4bc36566d480906fd4	**Yes**
PSFILE.TXT	Text	799ca951bc9bddd731da227c9d2a0902	799ca951bc9bddd731da227c9d2a0902	**Yes**
NETFILE.TXT	Text	768165e0abf16bf3056836d5431a7296	768165e0abf16bf3056836d5431a7296	**Yes**
C_filestg.txt	Text	5f45b5367c7ed885270ea969851cb439	5f45b5367c7ed885270ea969851cb439	**Yes**
C_hfind.txt	Text	413b8eac24a155c702221cf89581e55c	413b8eac24a155c702221cf89581e55c	**Yes**

C_hidden.txt	Text	5f45b5367c7ed885270ea969851cb439	5f45b5367c7ed885270ea969851cb439	**Yes**
C_streams.txt	Text	0bdbfa670c99deef38fb86a21b39a9e6	0bdbfa670c99deef38fb86a21b39a9e6	**Yes**
C_sfind.txt	Text	b5ba059d7bfd0f4e4d005cd2e9e497b3	b5ba059d7bfd0f4e4d005cd2e9e497b3	**Yes**
C_efsinfo.txt	Text	c218c5009550effcc9c8332ef65b6302	c218c5009550effcc9c8332ef65b6302	**Yes**
RECENT.TXT	Text	877a6df5f576b096d28d049ae2d1645c	877a6df5f576b096d28d049ae2d1645c	**Yes**
C_recycle.txt	Text	8609e8e78d5e28a8db50ba4457383f6d	8609e8e78d5e28a8db50ba4457383f6d	**Yes**
PREFETCH.TXT	Text	11b255d75ee6393d6cd71bf1879c3488	11b255d75ee6393d6cd71bf1879c3488	**Yes**
C_freesp.txt	Text	620ad01a32b3e6f4bc54365e576e9122	620ad01a32b3e6f4bc54365e576e9122	**Yes**
AUTORUNS.TXT	Text	88996e636c7afdf72301fbbc98382608	88996e636c7afdf72301fbbc98382608	**Yes**
AUTOEXEC.TXT	Text	d41d8cd98f00b204e9800998ecf8427e	d41d8cd98f00b204e9800998ecf8427e	**Yes**
WIN_INI.TXT	Text	8c1dff16cd151b49143c48796bb750fd	8c1dff16cd151b49143c48796bb750fd	**Yes**
SYS_INI.TXT	Text	b143a6852c9ef93e0bdecb02f524f9f2	b143a6852c9ef93e0bdecb02f524f9f2	**Yes**
WINSTART.TXT	Text	df5dc1abc0d52f3c9e931e26a7c0065c	df5dc1abc0d52f3c9e931e26a7c0065c	**Yes**

INIT_INI.TXT	Text	df5dc1abc0d52f3c9e931e26a7c0065c	df5dc1abc0d52f3c9e931e26a7c0065c	**Yes**
STARTUP.TXT	Text	a220c7bd4ace2153de5e30eda85c945b	a220c7bd4ace2153de5e30eda85c945b	**Yes**
user_startup.txt	Text	c5b62b38f85a47009ac1362c007ff7a0	c5b62b38f85a47009ac1362c007ff7a0	**Yes**
TASKS.TXT	Text	597f96e64a39d161f9c79bee54acbb42	597f96e64a39d161f9c79bee54acbb42	**Yes**
AT.TXT	Text	90e2be5857e337cbdbffde58b2e91c81	90e2be5857e337cbdbffde58b2e91c81	**Yes**
SCHTASKS.TXT	Text	fa00ddd2f8fe5c3a55e96bccb6c87fff	fa00ddd2f8fe5c3a55e96bccb6c87fff	**Yes**
HKLM_R.TXT	Text	688bc52d6aa90f2204dc16e71b81c24e	688bc52d6aa90f2204dc16e71b81c24e	**Yes**
HKLM_RO.TXT	Text	992371965fae1f03a17f5aaf5e6598bc	992371965fae1f03a17f5aaf5e6598bc	**Yes**
HKLM_ROX.TXT	Text	2222ccded19cd28cf6b9c0432c445664	2222ccded19cd28cf6b9c0432c445664	**Yes**
HKLM_RS.TXT	Text	629680bfdcb03eb78749d094febdbcb9	629680bfdcb03eb78749d094febdbcb9	**Yes**
HKLM_RSO.TXT	Text	629680bfdcb03eb78749d094febdbcb9	629680bfdcb03eb78749d094febdbcb9	**Yes**
hklm_scripts.txt	Text	629680bfdcb03eb78749d094febdbcb9	629680bfdcb03eb78749d094febdbcb9	**Yes**
hklm_expl_run.txt	Text	629680bfdcb03eb78749d094febdbcb9	629680bfdcb03eb78749d094febdbcb9	**Yes**

HKCU_R.TXT	Text	9e7b99199b2931e811ab482b833c2006	9e7b99199b2931e811ab482b833c2006	**Yes**
HKCU_RO.TXT	Text	992371965fae1f03a17f5aaf5e6598bc	992371965fae1f03a17f5aaf5e6598bc	**Yes**
HKCU_ROX.TXT	Text	629680bfdcb03eb78749d094febdbcb9	629680bfdcb03eb78749d094febdbcb9	**Yes**
HKCU_RS.TXT	Text	629680bfdcb03eb78749d094febdbcb9	629680bfdcb03eb78749d094febdbcb9	**Yes**
HKCU_RSO.TXT	Text	629680bfdcb03eb78749d094febdbcb9	629680bfdcb03eb78749d094febdbcb9	**Yes**
hkcu_scripts.txt	Text	629680bfdcb03eb78749d094febdbcb9	629680bfdcb03eb78749d094febdbcb9	**Yes**
hkcu_expl_run.txt	Text	629680bfdcb03eb78749d094febdbcb9	629680bfdcb03eb78749d094febdbcb9	**Yes**
GPLIST.TXT	Text	e9b3135e5cd46f2e434e67a0de631f95	e9b3135e5cd46f2e434e67a0de631f95	**Yes**
GPUSER.TXT	Text	96584e6ad16be6c847166d809f70033a	96584e6ad16be6c847166d809f70033a	**Yes**
GPSYS.TXT	Text	c944fa3824417a3ae083c45473281bd7	c944fa3824417a3ae083c45473281bd7	**Yes**
RUN_HIST.TXT	Text	d702985d915284e12b59eacb37a6f7a1	d702985d915284e12b59eacb37a6f7a1	**Yes**
RCNT_DOC.TXT	Text	d51794c380b3aaa1435d0892a0f8c5ca	d51794c380b3aaa1435d0892a0f8c5ca	**Yes**
LASTSAVE.TXT	Text	82b262a274831c8aeeff58322720484f	82b262a274831c8aeeff58322720484f	**Yes**

INSTALLH.TXT	Text	49bcfc17ba427fa4b3ee188147da0bee	49bcfc17ba427fa4b3ee188147da0bee	**Yes**
REGDMP.TXT	Text	44d3c46c824986693d8acc7ead3075c1	44d3c46c824986693d8acc7ead3075c1	**Yes**
hklm_safemin.txt	Text	2b0bed64d2e3dac4053f40cf2982b75c	2b0bed64d2e3dac4053f40cf2982b75c	**Yes**
hklm_safenet.txt	Text	dd074f35c25931d4c193287358e5f594	dd074f35c25931d4c193287358e5f594	**Yes**
IEHV.TXT	Text	062e3074795a47cf35deb4e8197c3ee9	062e3074795a47cf35deb4e8197c3ee9	**Yes**
TYPE_URL.TXT	Text	60623f0a8e30be61cde155c22626877b	60623f0a8e30be61cde155c22626877b	**Yes**
SEARCH_H.TXT	Text	7fe3bd462d6f904ddb033d2b1a93bb04	7fe3bd462d6f904ddb033d2b1a93bb04	**Yes**
pstoreview.txt	Text	e9b3135e5cd46f2e434e67a0de631f95	e9b3135e5cd46f2e434e67a0de631f95	**Yes**
DOSKEY.TXT	Text	d41d8cd98f00b204e9800998ecf8427e	d41d8cd98f00b204e9800998ecf8427e	**Yes**
MDM.TXT	Text	70ee549e2b10af81209c3b57b8a6a517	70ee549e2b10af81209c3b57b8a6a517	**Yes**
ROOTKIT.TXT	Text	bb67b3f5126a55a3da7742791f825df6	bb67b3f5126a55a3da7742791f825df6	**Yes**
INDEX.HTM	Web Page	46a56b4d790d32bd26db6f43a813ce2b	46a56b4d790d32bd26db6f43a813ce2b	**Yes**
WFT_RPT.XML	XML Document	f9f003994001cd366d89e647b577f6c0	f9f003994001cd366d89e647b577f6c0	**Yes**

ARKAR_US.B	BASIC Source Code			
.E01		ed68a4e4a9a187abe303cc7617ce49ec	ed68a4e4a9a187abe303cc7617ce49ec	**Yes**
.E02		3ea1d1c9c52f095340255153735c4e30	3ea1d1c9c52f095340255153735c4e30	**Yes**
.E03		c2e11183bb3007c399d47141c8264600	c2e11183bb3007c399d47141c8264600	**Yes**
.E04		bbb263f8c90b848962cf99510969b125	bbb263f8c90b848962cf99510969b125	**Yes**
.E05		23b898043a68cdc9412917b7a2be0eb7	23b898043a68cdc9412917b7a2be0eb7	**Yes**
_EADER~1.BIN	**Raw Binary Format**			
ReaderMemoryImage.bin	Raw Binary Format	5fa92fb2cf633d014cfc103dc2fedfde	5fa92fb2cf633d014cfc103dc2fedfde	**Yes**

ReaderMemoryImage.md5.txt	Text	8c4d840bce0ea858a6a7fb2dff514dac	8c4d840bce0ea858a6a7fb2dff514dac	**Yes**
InitialConnection.txt	Text	4fcde60fa9e767c6cfcc6234ffb72937	4fcde60fa9e767c6cfcc6234ffb72937	**Yes**
TransactionLog.txt	Text	8c4ef1c03709e4f1cbcdf83dc313c503	8c4ef1c03709e4f1cbcdf83dc313c503	**Yes**
RingBuffer.txt	Text	a7c6d77ddf780352f2675416f1c61ecd	a7c6d77ddf780352f2675416f1c61ecd	**Yes**
RingBuffers.txt	Text	d42a21ae9f5d78ee9bfbaeaab9b774d8	d42a21ae9f5d78ee9bfbaeaab9b774d8	**Yes**
LoginConfig.txt	Text	d4308329566b650338b1b0de9a89530e	d4308329566b650338b1b0de9a89530e	**Yes**
ServerPermission.txt	Text	4f9f8ca963b767a280c92a0baf875dac	4f9f8ca963b767a280c92a0baf875dac	**Yes**
ServerPrincipalData.txt	Text	37d235cd64cf023f42f7dd203ec82ca8	37d235cd64cf023f42f7dd203ec82ca8	**Yes**
SystemDatabasePermissions.txt	Text	c07f0bb38e5588b287dbc86150825ce9	c07f0bb38e5588b287dbc86150825ce9	**Yes**
SystemDatabasePrincipals.txt	Text	a407b6fec499c67c92675df48ef4ca74	a407b6fec499c67c92675df48ef4ca74	**Yes**
SystemDatabaseRoleMembers.txt	Text	202a3f083c695d8077495d7cf4d9a818	202a3f083c695d8077495d7cf4d9a818	**Yes**
TableStatstics.txt	Text	90e2b070be5288f00ce63896f35a5d2e	90e2b070be5288f00ce63896f35a5d2e	**Yes**
TableStatstics1.txt	Text	6ef4b731601b1e5b249c4b4cfa147fa3	6ef4b731601b1e5b249c4b4cfa147fa3	**Yes**
RFID_db.txt	Text	dd3a6c1d1d5c6ebd2a7adba102c7436e	dd3a6c1d1d5c6ebd2a7adba102c7436e	**Yes**
RFID_db_TableStats2.txt	Text	1c95bbba92c1d31361652e9b8b9f8558	1c95bbba92c1d31361652e9b8b9f8558	**Yes**

CollationAndDataType.txt	Text	37a07ffe3ab6f936fd77c7642e020551	37a07ffe3ab6f936fd77c7642e020551	**Yes**
DBO_RFIDdb_tblSta.txt	Text	609da9494a99f62d36ba15a375a23a3c	609da9494a99f62d36ba15a375a23a3c	**Yes**
DBO_RFIDdb_tblSta1.txt	Text	80dd813b03059d565c1e3dcf6823c597	80dd813b03059d565c1e3dcf6823c597	**Yes**
ServiceShutDown.txt	Text	3e5bd28f60b6e4d4596dec044fc51e3d	3e5bd28f60b6e4d4596dec044fc51e3d	**Yes**
InitialConnection_01.txt	Text	5ff7382dc28235b691f60f1adc475cbb	5ff7382dc28235b691f60f1adc475cbb	**Yes**
DBO_RFIDdb_tblStats.txt	Text	609da9494a99f62d36ba15a375a23a3c	609da9494a99f62d36ba15a375a23a3c	**Yes**
DBO_RFIDdb_tblStats1.txt	Text	31ae206c9c280faa143462b1af7952e5	31ae206c9c280faa143462b1af7952e5	**Yes**
DBO_RFIDdb_tblStats_01.txt	Text	e7acc61413b1615e7c9bb85b799dba96	e7acc61413b1615e7c9bb85b799dba96	**Yes**
TableStatistics.txt	Text	90e2b070be5288f00ce63896f35a5d2e	90e2b070be5288f00ce63896f35a5d2e	**Yes**
TableStatistics_01.txt	Text	6ef4b731601b1e5b249c4b4cfa147fa3	6ef4b731601b1e5b249c4b4cfa147fa3	**Yes**
TableStatistics_02.txt	Text	9fc07a5d1b3f414a8484866b8539f7d2	9fc07a5d1b3f414a8484866b8539f7d2	**Yes**
DBO_RFIDdb_tblStat.txt	Text	e7acc61413b1615e7c9bb85b799dba96	e7acc61413b1615e7c9bb85b799dba96	**Yes**
DBO_RFIDdb_tblStat_Tag.txt	Text	d09bf1cc3b410aff8c0be7c2d8fe5653	d09bf1cc3b410aff8c0be7c2d8fe5653	**Yes**
DBO_RFIDdb_tblStat_Value.txt	Text	8d5562f81f430924a50c542f4bb19f08	8d5562f81f430924a50c542f4bb19f08	**Yes**

DBO_RFIDdb_tblStat_Date.txt	Text	d0eebf9cb3afae82e7208b9dee954654	d0eebf9cb3afae82e7208b9dee954654	**Yes**
DBO_RFIDdb_tbl.txt	Text	e5a5c6f5fb2f12d39ef145ae3c649134	e5a5c6f5fb2f12d39ef145ae3c649134	**Yes**
DBO_RFIDlog_tbl.txt	Text	f00f78af761da4a7b0542539f82c71c2	f00f78af761da4a7b0542539f82c71c2	**Yes**
DBO_RFIDlog_tblStat_Tag.txt	Text	705eb130b180a184ead7bf88c4b0b2d4	705eb130b180a184ead7bf88c4b0b2d4	**Yes**
DBO_RFIDlog_tblStat_Value.txt	Text	3aaecfd5ee1048ec439645e8ff1a415e	3aaecfd5ee1048ec439645e8ff1a415e	**Yes**
DBO_RFIDlog_tblStat_Date.txt	Text	4691ddfb0f023e968149af81f1305be5	4691ddfb0f023e968149af81f1305be5	**Yes**
~~DBO_RFIDlog_tblStat_UserData.txt~~	~~Text~~	~~c377e8ceb3ca70160648ea0747c93a0c~~	~~c377e8ceb3ca70160648ea0747c93a0c~~	~~**Yes**~~
DBO_RFIDlog_tblStat_UserData1.txt	Text	c377e8ceb3ca70160648ea0747c93a0c	c377e8ceb3ca70160648ea0747c93a0c	**Yes**
DBO_RFIDlog_tblStat_UserData2.txt	Text	6cf7dcdbcfdd404ac3d12f0e211c1a16	6cf7dcdbcfdd404ac3d12f0e211c1a16	**Yes**
ConnectionDetails_02.txt	Text	d1a7d24f1b58f3448b09f480bceca232	d1a7d24f1b58f3448b09f480bceca232	**Yes**
ServiceShutDown_02.txt	Text	3e5bd28f60b6e4d4596dec044fc51e3d	3e5bd28f60b6e4d4596dec044fc51e3d	**Yes**
RFID_test.mdf		1b5a64213fef691fbac302a9deb444f0	1b5a64213fef691fbac302a9deb444f0	**Yes**

RFID_test.md5		45c0868ea943c2171ff3554354fd0256	45c0868ea943c2171ff3554354fd0256	**Yes**
RFID_test_log.ldf		6e492f050440d6fce0cdf755ce4f4a85	6e492f050440d6fce0cdf755ce4f4a85	**Yes**
RFID_test_log.md5		88df3fe969f18dd0cf7cb55e9ed3d8a2	88df3fe969f18dd0cf7cb55e9ed3d8a2	**Yes**
LOG_34.TRC		bcd5e8dcf2347d1374680bdf1a383c02	bcd5e8dcf2347d1374680bdf1a383c02	**Yes**
LOG_34.MD5		cab781bdbc167d2a3f2ca7507fc38188	cab781bdbc167d2a3f2ca7507fc38188	**Yes**
LOG_35.TRC		2c2c486e2c34dbe67958ae614f8184a7	2c2c486e2c34dbe67958ae614f8184a7	**Yes**
LOG_35.MD5		7d9601ded7348e0eddfe10f33802baa0	7d9601ded7348e0eddfe10f33802baa0	**Yes**
LOG_36.TRC		6386e16ea0b9f680665a26f4fbc93620	6386e16ea0b9f680665a26f4fbc93620	**Yes**
LOG_36.MD5		6955f61691621056a831f52fdba4f3d3	6955f61691621056a831f52fdba4f3d3	**Yes**
LOG_37.TRC		518b9e0fc2cef59c4cae7e8fc1333644	518b9e0fc2cef59c4cae7e8fc1333644	**Yes**
LOG_37.MD5		aa4b556156b808400e862dc3b4346906	aa4b556156b808400e862dc3b4346906	**Yes**

LOG_38.TRC		bc70e43b797bd20d587af7a2d9911326	bc70e43b797bd20d587af7a2d9911326	**Yes**
LOG_38.MD5		b4f8face164d7f814a20325012025fee	b4f8face164d7f814a20325012025fee	**Yes**
ERRORLOG		653a99af416636ab023441a8b154d380	653a99af416636ab023441a8b154d380	**Yes**
ERRORLOG.MD5		2e4e49ddd785e1e6e5001954894daa16	2e4e49ddd785e1e6e5001954894daa16	**Yes**
ERRORLOG.1		5104f97dd59a41008fe4ac8001bd5e8a	5104f97dd59a41008fe4ac8001bd5e8a	**Yes**
errorlog_1.md5		944a55e69038cca4d0d7c1879902c496	944a55e69038cca4d0d7c1879902c496	**Yes**
ERRORLOG.2		131dfc823e7aea83db63fae3cdda131b	131dfc823e7aea83db63fae3cdda131b	**Yes**
errorlog_2.md5		2d0441be1fe4ca208c9109468f1a6a03	2d0441be1fe4ca208c9109468f1a6a03	**Yes**
ERRORLOG.3		d6e116cdb3124cd8b75fd240b1b2b872	d6e116cdb3124cd8b75fd240b1b2b872	**Yes**
errorlog_3.md5		393443b2b8d0c58dc75c26732a017a5f	393443b2b8d0c58dc75c26732a017a5f	**Yes**
ERRORLOG.4		fb8195b9c82bbc3e07b3d20015b9fe74	fb8195b9c82bbc3e07b3d20015b9fe74	**Yes**

errorlog_4.md5		3f7f86d69760b535bd829ed2a0666dee	3f7f86d69760b535bd829ed2a0666dee	**Yes**
ERRORLOG.5		9d6e83db6063a8e14e51c65eb481bff4	9d6e83db6063a8e14e51c65eb481bff4	**Yes**
errorlog_5.md5		ecb8833e2bfa166de2ab2bff7e1e8f46	ecb8833e2bfa166de2ab2bff7e1e8f46	**Yes**
ERRORLOG.6		ca21b0eda2cefe9416ea62e5acc23fe8	ca21b0eda2cefe9416ea62e5acc23fe8	**Yes**
errorlog_6.md5		2e5610d78728a8342349e466c6943602	2e5610d78728a8342349e466c6943602	**Yes**
ApplicationLog.txt*	Text	fbfba18f6775ee9d4d2cde6577ae747d	fbfba18f6775ee9d4d2cde6577ae747d	**Yes**
ApplicationLog_1.txt	Text	ab1c6e64a8a76dcf52e6bd9fff432fe7	ab1c6e64a8a76dcf52e6bd9fff432fe7	**Yes**
SystemLog.txt	Text	e912a940788546da21c09e565ed2e92d	e912a940788546da21c09e565ed2e92d	**Yes**
SecurityLog.txt	Text	c2848cc75e55d289d9e4e0f33e6e14b6	c2848cc75e55d289d9e4e0f33e6e14b6	**Yes**

Where:

Analysed Volatile SQL Data (in txt format) Acquired by Extended Windows Forensic Toolchest (WFT) v3.0.05.

RFID Reader's Memory Image Acquired by Developed ReaderLogExtractionTool.

Acquired Physical Disk Drive Image of POS/Server Acquired by AccessData® FTK® Imager 2.9.0.1385 100406.

All SQL Data Acquired (in html format) by Extended Windows Forensic Toolchest (WFT) v3.0.05.

Physical Database Files of Backend SQL Server Acquired by dcfldd tool.

Trace Files and Error Logs Acquired by dcfldd tool.

Application, System, Security Logs Acquired by *psloglist.exe* utility from customized HELIX_RFID_IR Toolkit.

POS Memory Image Acquired by Guidance Software® *winen.exe* was located in the customized Helix_RFID_IR Toolkit.

Appendix 23: Analysis of Acquired SQL Server Error Log Files

The following acquired seven SQL Server error logs are analyzed by using notepad++ version 5.0.3, which is an open source text and source code editor for Windows system (see Figure A24.1 below).

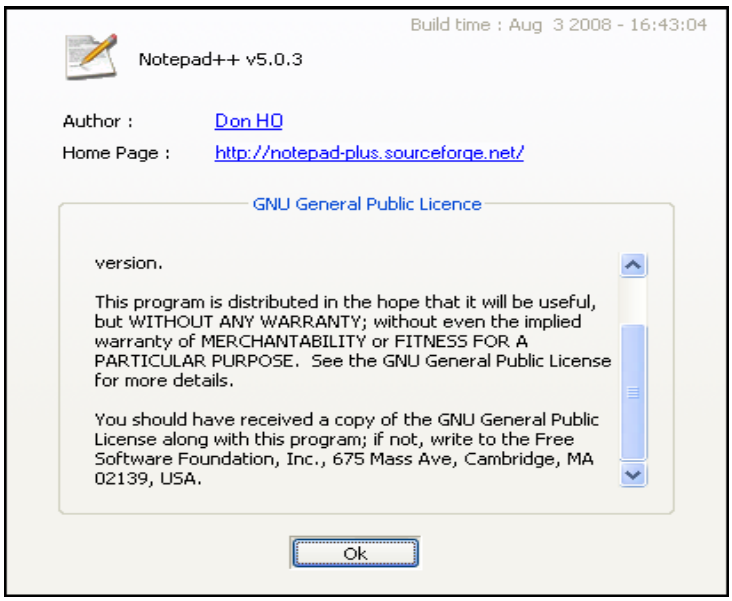


Figure A23.1: Screenshot of text and source code editor information

1. File Name: errorlog (Current Error Log)

2010-10-25 12:37:04.13 Server Microsoft SQL Server 2005 - 9.00.1399.06 (Intel X86)

Oct 14 2005 00:33:37

Copyright (c) 1988-2005 Microsoft Corporation

Express Edition on Windows NT 5.1 (Build 2600: Service Pack 2)

2010-10-25 12:37:04.13 Server (c) 2005 Microsoft Corporation.

2010-10-25 12:37:04.13 Server All rights reserved.

2010-10-25 12:37:04.13 Server Server process ID is 1296.

2010-10-25 12:37:04.13 Server Logging SQL Server messages in file 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG'.

2010-10-25 12:37:04.13 Server This instance of SQL Server last reported using a process ID of 1508 at 10/25/2010 12:22:56 PM (local) 10/24/2010

11:22:56 PM (UTC). This is an informational message only; no user action is required.

2010-10-25 12:37:04.13 Server Registry startup parameters:

2010-10-25 12:37:04.13 Server -d C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\master.mdf

2010-10-25 12:37:04.13 Server -e C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG

2010-10-25 12:37:04.13 Server -l C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\mastlog.ldf

2010-10-25 12:37:04.15 Server SQL Server is starting at normal priority base (=7). This is an informational message only. No user action is required.

2010-10-25 12:37:04.15 Server Detected 2 CPUs. This is an informational message; no user action is required.

2010-10-25 12:37:04.51 Server Using dynamic lock allocation. Initial allocation of 2500 Lock blocks and 5000 Lock Owner blocks per node. This is an informational message only. No user action is required.

2010-10-25 12:37:04.51 Server Database Mirroring Transport is disabled in the endpoint configuration.

2010-10-25 12:37:04.51 spid5s Starting up database 'master'.

2010-10-25 12:37:04.67 spid5s Recovery is writing a checkpoint in database 'master' (1). This is an informational message only. No user action is required.

2010-10-25 12:37:04.84 spid5s SQL Trace ID 1 was started by login "sa".

2010-10-25 12:37:04.90 spid5s Starting up database 'mssqlsystemresource'.

2010-10-25 12:37:05.35 spid8s Starting up database 'model'.

2010-10-25 12:37:05.35 spid5s Server name is 'TEST-STATION\SQLEXPRESS'. This is an informational message only. No user action is required.

2010-10-25 12:37:05.37 spid5s Starting up database 'msdb'.

2010-10-25 12:37:05.56 Server A self-generated certificate was successfully loaded for encryption.

2010-10-25 12:37:05.56 Server Server is listening on ['any' <ipv4> 1069].

2010-10-25 12:37:05.56 Server Server local connection provider is ready to accept connection on [\\.\pipe\SQLLocal\SQLEXPRESS].

2010-10-25 12:37:05.56 Server Server local connection provider is ready to accept connection on [\\.\pipe\MSSQL$SQLEXPRESS\sql\query].

2010-10-25 12:37:05.56 Server Dedicated administrator connection support was not started because it is not available on this edition of SQL Server. This is an informational message only. No user action is required.

2010-10-25 12:37:05.57 Server The SQL Network Interface library could not register the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Failure to register an SPN may cause integrated authentication to fall back to NTLM instead of Kerberos. This is an informational message. Further action is only required if Kerberos authentication is required by authentication policies.

2010-10-25 12:37:05.57 Server SQL Server is now ready for client connections. This is an informational message; no user action is required.

2010-10-25 12:37:06.23 spid8s Clearing tempdb database.

2010-10-25 12:37:07.42 spid8s Starting up database 'tempdb'.

2010-10-25 12:37:07.74 spid5s Recovery is complete. This is an informational message only. No user action is required.

2010-10-25 12:37:07.74 spid11s The Service Broker protocol transport is disabled or not configured.

2010-10-25 12:37:07.74 spid11s The Database Mirroring protocol transport is disabled or not configured.

2010-10-25 12:37:07.92 spid11s Service Broker manager has started.

2010-10-25 12:38:07.20 spid51 DBCC TRACEON 3604, server process ID (SPID) 51. This is an informational message only; no user action is required.

2010-10-25 12:39:47.26 spid51 DBCC TRACEON 3604, server process ID (SPID) 51. This is an informational message only; no user action is required.

2010-10-25 12:45:37.29 spid52 Starting up database 'RFID_test'.

2010-10-25 12:45:37.45 spid52 Recovery is writing a checkpoint in database 'RFID_test' (5). This is an informational message only. No user action is required.

2010-10-25 13:53:22.35 spid11s Service Broker manager has shut down.

2010-10-25 13:53:22.81 spid51 Server shut down by request from login arkar.

2010-10-25 13:53:22.81 spid51 SQL Trace was stopped due to server shutdown. Trace ID = '1'. This is an informational message only; no user action is required.

2010-10-25 13:53:22.95 Server The SQL Network Interface library could not deregister the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Administrator should deregister this SPN manually to avoid client authentication errors.

2. File Name: errorlog.1

2010-10-05 01:22:55.48 Server Microsoft SQL Server 2005 - 9.00.1399.06 (Intel X86)

Oct 14 2005 00:33:37

Copyright (c) 1988-2005 Microsoft Corporation

Express Edition on Windows NT 5.1 (Build 2600: Service Pack 2)

2010-10-05 01:22:55.48 Server (c) 2005 Microsoft Corporation.

2010-10-05 01:22:55.48 Server All rights reserved.

2010-10-05 01:22:55.48 Server Server process ID is 1508.

2010-10-05 01:22:55.48 Server Logging SQL Server messages in file 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG'.

2010-10-05 01:22:55.48 Server This instance of SQL Server last reported using a process ID of 1516 at 10/5/2010 12:33:19 AM (local) 10/4/2010 11:33:19 AM (UTC). This is an informational message only; no user action is required.

2010-10-05 01:22:55.48 Server Registry startup parameters:

2010-10-05 01:22:55.50 Server -d C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\master.mdf

2010-10-05 01:22:55.50 Server -e C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG

2010-10-05 01:22:55.50 Server -l C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\mastlog.ldf

2010-10-05 01:22:55.53 Server SQL Server is starting at normal priority base (=7). This is an informational message only. No user action is required.

2010-10-05 01:22:55.53 Server Detected 2 CPUs. This is an informational message; no user action is required.

2010-10-05 01:22:56.56 Server Using dynamic lock allocation. Initial allocation of 2500 Lock blocks and 5000 Lock Owner blocks per node. This is an informational message only. No user action is required.

2010-10-05 01:22:57.68 Server Database Mirroring Transport is disabled in the endpoint configuration.

2010-10-05 01:22:57.90 spid5s Starting up database 'master'.

2010-10-05 01:22:58.31 spid5s Recovery is writing a checkpoint in database 'master' (1). This is an informational message only. No user action is required.

2010-10-05 01:22:58.73 spid5s SQL Trace ID 1 was started by login "sa".

2010-10-05 01:22:58.79 spid5s Starting up database 'mssqlsystemresource'.

2010-10-05 01:22:59.37 spid8s Starting up database 'model'.

2010-10-05 01:22:59.42 spid5s Server name is 'TEST-STATION\SQLEXPRESS'. This is an informational message only. No user action is required.

2010-10-05 01:22:59.42 spid5s Starting up database 'msdb'.

2010-10-05 01:22:59.51 Server A self-generated certificate was successfully loaded for encryption.

2010-10-05 01:22:59.51 Server Server is listening on ['any' <ipv4> 1069].

2010-10-05 01:22:59.51 Server Server local connection provider is ready to accept connection on [\\.\pipe\SQLLocal\SQLEXPRESS].

2010-10-05 01:22:59.56 Server Server local connection provider is ready to accept connection on [\\.\pipe\MSSQL$SQLEXPRESS\sql\query].

2010-10-05 01:22:59.57 Server Dedicated administrator connection support was not started because it is not available on this edition of SQL Server. This is an informational message only. No user action is required.

2010-10-05 01:22:59.59 Server The SQL Network Interface library could not register the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Failure to register an SPN may cause integrated authentication to fall back to NTLM instead of Kerberos. This is an informational message. Further action is only required if Kerberos authentication is required by authentication policies.

2010-10-05 01:22:59.59 Server SQL Server is now ready for client connections. This is an informational message; no user action is required.

2010-10-05 01:23:00.63 spid8s Clearing tempdb database.

2010-10-05 01:23:06.92 spid8s Starting up database 'tempdb'.

2010-10-05 01:23:08.79 spid5s Recovery is complete. This is an informational message only. No user action is required.

2010-10-05 01:23:08.87 spid11s The Service Broker protocol transport is disabled or not configured.

2010-10-05 01:23:08.87 spid11s The Database Mirroring protocol transport is disabled or not configured.

2010-10-05 01:23:10.06 spid11s Service Broker manager has started.

2010-10-05 01:26:49.76 Server Server resumed execution after being idle 181 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-05 01:27:14.85 spid51 Using 'xpstar90.dll' version '2005.90.1399' to execute extended stored procedure 'xp_enumerrorlogs'. This is an informational message only; no user action is required.

2010-10-05 01:39:58.62 spid51 Starting up database 'RFID_test'.

2010-10-06 01:00:08.01 spid1s Server resumed execution after being idle 81945 seconds. Reason: timer event.

2010-10-07 01:00:13.01 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-07 13:57:33.64 Server Server resumed execution after being idle 45727 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-07 13:57:33.67 Logon Error: 18456, Severity: 14, State: 8.

2010-10-07 13:57:33.67 Logon Login failed for user 'arkar'. [CLIENT: <local machine>]

2010-10-07 14:16:23.56 Server Server resumed execution after being idle 69 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-07 15:20:54.21 Server Server resumed execution after being idle 2946 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-08 01:00:18.00 spid1s Server resumed execution after being idle 33829 seconds. Reason: timer event.

2010-10-08 18:08:55.25 Server Server resumed execution after being idle 60803 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-08 18:48:23.31 Server Server resumed execution after being idle 10 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-09 01:00:22.98 spid1s Server resumed execution after being idle 21319 seconds. Reason: timer event.

2010-10-10 01:00:27.98 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-11 01:00:32.98 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-12 01:00:37.98 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-12 15:30:04.32 Server Server resumed execution after being idle 51252 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-12 18:39:06.70 Server Server resumed execution after being idle 8313 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-13 01:00:42.96 spid1s Server resumed execution after being idle 20876 seconds. Reason: timer event.

2010-10-14 01:00:47.96 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-15 01:00:52.96 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-16 01:00:57.96 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-17 01:00:02.96 spid1s Server resumed execution after being idle 85431 seconds. Reason: timer event.

2010-10-18 01:00:07.96 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-19 01:00:12.96 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-20 01:00:17.96 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-21 01:00:22.96 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-22 01:00:27.96 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-23 01:00:32.96 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-24 01:00:37.96 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-25 01:00:42.96 spid1s Server resumed execution after being idle 85491 seconds. Reason: timer event.

2010-10-25 06:31:38.40 Server Server resumed execution after being idle 18941 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-25 07:18:50.25 spid53 Using 'xplog70.dll' version '2005.90.1399' to execute extended stored procedure 'xp_loginconfig'. This is an informational message only; no user action is required.

2010-10-25 08:14:36.96 Server Server resumed execution after being idle 247 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-25 11:18:52.50 Server Server resumed execution after being idle 8263 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-25 11:57:08.93 Server Server resumed execution after being idle 498 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-25 12:17:39.26 Server Server resumed execution after being idle 83 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-25 12:22:55.26 spid11s Service Broker manager has shut down.

2010-10-25 12:22:55.95 spid53 Server shut down by request from login arkar.

2010-10-25 12:22:56.03 spid53 SQL Trace was stopped due to server shutdown. Trace ID = '1'. This is an informational message only; no user action is required.

2010-10-25 12:22:56.31 Server The SQL Network Interface library could not deregister the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Administrator should deregister this SPN manually to avoid client authentication errors.

3. File Name: errorlog.2

2010-10-03 00:59:51.40 Server Microsoft SQL Server 2005 - 9.00.1399.06 (Intel X86)

Oct 14 2005 00:33:37

Copyright (c) 1988-2005 Microsoft Corporation

Express Edition on Windows NT 5.1 (Build 2600: Service Pack 2)

2010-10-03 00:59:51.40 Server (c) 2005 Microsoft Corporation.

2010-10-03 00:59:51.40 Server All rights reserved.

2010-10-03 00:59:51.40 Server Server process ID is 1516.

2010-10-03 00:59:51.40 Server Logging SQL Server messages in file 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG'.

2010-10-03 00:59:51.40 Server This instance of SQL Server last reported using a process ID of 3492 at 10/2/2010 2:40:05 AM (local) 10/1/2010 2:40:05 PM (UTC). This is an informational message only; no user action is required.

2010-10-03 00:59:51.40 Server Registry startup parameters:

2010-10-03 00:59:51.42 Server -d C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\master.mdf

2010-10-03 00:59:51.42 Server -e C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG

2010-10-03 00:59:51.42 Server -l C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\mastlog.ldf

2010-10-03 00:59:51.65 Server SQL Server is starting at normal priority base (=7). This is an informational message only. No user action is required.

2010-10-03 00:59:51.65 Server Detected 2 CPUs. This is an informational message; no user action is required.

2010-10-03 00:59:52.90 Server Using dynamic lock allocation. Initial allocation of 2500 Lock blocks and 5000 Lock Owner blocks per node. This is an informational message only. No user action is required.

2010-10-03 00:59:53.95 Server Database Mirroring Transport is disabled in the endpoint configuration.

2010-10-03 00:59:54.17 spid5s Starting up database 'master'.

2010-10-03 00:59:54.62 spid5s Recovery is writing a checkpoint in database 'master' (1). This is an informational message only. No user action is required.

2010-10-03 00:59:55.12 spid5s SQL Trace ID 1 was started by login "sa".

2010-10-03 00:59:55.18 spid5s Starting up database 'mssqlsystemresource'.

2010-10-03 00:59:55.78 spid8s Starting up database 'model'.

2010-10-03 00:59:55.81 spid5s Server name is 'TEST-STATION\SQLEXPRESS'. This is an informational message only. No user action is required.

2010-10-03 00:59:55.81 spid5s Starting up database 'msdb'.

2010-10-03 00:59:55.92 Server A self-generated certificate was successfully loaded for encryption.

2010-10-03 00:59:55.93 Server Server is listening on ['any' <ipv4> 1069].

2010-10-03 00:59:55.93 Server Server local connection provider is ready to accept connection on [\\.\pipe\SQLLocal\SQLEXPRESS].

2010-10-03 00:59:55.96 Server Server local connection provider is ready to accept connection on [\\.\pipe\MSSQL$SQLEXPRESS\sql\query].

2010-10-03 00:59:55.96 Server Dedicated administrator connection support was not started because it is not available on this edition of SQL Server. This is an informational message only. No user action is required.

2010-10-03 00:59:55.98 Server The SQL Network Interface library could not register the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Failure to register an SPN may cause integrated authentication to fall back to NTLM instead of Kerberos. This is an informational message. Further action is only required if Kerberos authentication is required by authentication policies.

2010-10-03 00:59:55.98 Server SQL Server is now ready for client connections. This is an informational message; no user action is required.

2010-10-03 00:59:56.84 spid8s Clearing tempdb database.

2010-10-03 00:59:58.23 spid8s Starting up database 'tempdb'.

2010-10-03 00:59:58.56 spid5s Recovery is complete. This is an informational message only. No user action is required.

2010-10-03 00:59:58.62 spid11s The Service Broker protocol transport is disabled or not configured.

2010-10-03 00:59:58.62 spid11s The Database Mirroring protocol transport is disabled or not configured.

2010-10-03 00:59:59.48 spid11s Service Broker manager has started.

2010-10-03 01:00:58.57 spid1s Server resumed execution after being idle 24 seconds. Reason: timer event.

2010-10-03 01:04:43.90 spid52 Using 'xpstar90.dll' version '2005.90.1399' to execute extended stored procedure 'xp_instance_regread'. This is an informational message only; no user action is required.

2010-10-03 01:05:57.42 spid52 Starting up database 'RFID_test'.

2010-10-03 01:52:43.70 Server Server resumed execution after being idle 1759 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-03 04:36:28.17 Server Server resumed execution after being idle 3407 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-04 02:00:03.56 spid1s Server resumed execution after being idle 76100 seconds. Reason: timer event.

2010-10-05 00:33:19.59 Server SQL Server is terminating because of a system shutdown. This is an informational message only. No user action is required.

2010-10-05 00:33:21.93 spid11s Service Broker manager has shut down.

2010-10-05 00:33:21.93 spid11s Error: 17054, Severity: 16, State: 1.

2010-10-05 00:33:21.93 spid11s The current event was not reported to the Windows Events log. Operating system error = 31(A device attached to the system is not functioning.). You may need to clear the Windows Events log if it is full.

2010-10-05 00:33:23.76 spid5s SQL Trace was stopped due to server shutdown. Trace ID = '1'. This is an informational message only; no user action is required.

2010-10-05 00:33:23.90 Server The SQL Network Interface library could not deregister the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Administrator should deregister this SPN manually to avoid client authentication errors.

4. File Name: errorlog.3

2010-10-02 00:11:57.85 Server Microsoft SQL Server 2005 - 9.00.1399.06 (Intel X86)

Oct 14 2005 00:33:37

Copyright (c) 1988-2005 Microsoft Corporation

Express Edition on Windows NT 5.1 (Build 2600: Service Pack 2)

2010-10-02 00:11:57.85 Server (c) 2005 Microsoft Corporation.

2010-10-02 00:11:57.85 Server All rights reserved.

2010-10-02 00:11:57.85 Server Server process ID is 3492.

2010-10-02 00:11:57.85 Server Logging SQL Server messages in file 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG'.

2010-10-02 00:11:57.85 Server This instance of SQL Server last reported using a process ID of 3716 at 10/2/2010 12:10:54 AM (local) 10/1/2010 12:10:54 PM (UTC). This is an informational message only; no user action is required.

2010-10-02 00:11:57.85 Server Registry startup parameters:

2010-10-02 00:11:57.85 Server -d C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\master.mdf

2010-10-02 00:11:57.85 Server -e C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG

2010-10-02 00:11:57.85 Server -l C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\mastlog.ldf

2010-10-02 00:11:57.90 Server SQL Server is starting at normal priority base (=7). This is an informational message only. No user action is required.

2010-10-02 00:11:57.90 Server Detected 2 CPUs. This is an informational message; no user action is required.

2010-10-02 00:11:58.26 Server Using dynamic lock allocation. Initial allocation of 2500 Lock blocks and 5000 Lock Owner blocks per node. This is an informational message only. No user action is required.

2010-10-02 00:11:58.26 Server Database Mirroring Transport is disabled in the endpoint configuration.

2010-10-02 00:11:58.26 spid5s Starting up database 'master'.

2010-10-02 00:11:58.48 spid5s Recovery is writing a checkpoint in database 'master' (1). This is an informational message only. No user action is required.

2010-10-02 00:11:58.64 spid5s SQL Trace ID 1 was started by login "sa".

2010-10-02 00:11:58.71 spid5s Starting up database 'mssqlsystemresource'.

2010-10-02 00:11:59.10 spid8s Starting up database 'model'.

2010-10-02 00:11:59.12 spid5s Server name is 'TEST-STATION\SQLEXPRESS'. This is an informational message only. No user action is required.

2010-10-02 00:11:59.12 spid5s Starting up database 'msdb'.

2010-10-02 00:11:59.28 Server A self-generated certificate was successfully loaded for encryption.

2010-10-02 00:11:59.28 Server Server is listening on ['any' <ipv4> 1069].

2010-10-02 00:11:59.28 Server Server local connection provider is ready to accept connection on [\\.\pipe\SQLLocal\SQLEXPRESS].

2010-10-02 00:11:59.28 Server Server local connection provider is ready to accept connection on [\\.\pipe\MSSQL$SQLEXPRESS\sql\query].

2010-10-02 00:11:59.28 Server Dedicated administrator connection support was not started because it is not available on this edition of SQL Server. This is an informational message only. No user action is required.

2010-10-02 00:11:59.28 Server The SQL Network Interface library could not register the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Failure to register an SPN may cause integrated authentication to fall back to NTLM instead of Kerberos. This is an informational message. Further action is only required if Kerberos authentication is required by authentication policies.

2010-10-02 00:11:59.28 Server SQL Server is now ready for client connections. This is an informational message; no user action is required.

2010-10-02 00:11:59.60 spid8s Clearing tempdb database.

2010-10-02 00:12:00.79 spid8s Starting up database 'tempdb'.

2010-10-02 00:12:01.09 spid5s Recovery is complete. This is an informational message only. No user action is required.

2010-10-02 00:12:01.09 spid11s The Service Broker protocol transport is disabled or not configured.

2010-10-02 00:12:01.09 spid11s The Database Mirroring protocol transport is disabled or not configured.

2010-10-02 00:12:01.20 spid11s Service Broker manager has started.

2010-10-02 00:14:53.89 Server Server resumed execution after being idle 140 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-02 00:14:59.40 spid51 Using 'xpstar90.dll' version '2005.90.1399' to execute extended stored procedure 'xp_enum_oledb_providers'. This is an informational message only; no user action is required.

2010-10-02 00:18:45.10 spid52 Starting up database 'RFID_test'.

2010-10-02 01:00:06.07 spid1s Server resumed execution after being idle 1540 seconds. Reason: timer event.

2010-10-02 01:24:18.51 Server Server resumed execution after being idle 538 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-02 01:47:05.82 Server Server resumed execution after being idle 427 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-02 02:40:05.35 Server SQL Server is terminating because of a system shutdown. This is an informational message only. No user action is required.

2010-10-02 02:40:06.93 spid11s Service Broker manager has shut down.

2010-10-02 02:40:06.98 spid11s Error: 17054, Severity: 16, State: 1.

2010-10-02 02:40:06.98 spid11s The current event was not reported to the Windows Events log. Operating system error = 1717(The interface is unknown.). You may need to clear the Windows Events log if it is full.

2010-10-02 02:40:08.60 spid5s SQL Trace was stopped due to server shutdown. Trace ID = '1'. This is an informational message only; no user action is required.

2010-10-02 02:40:08.78 Server The SQL Network Interface library could not deregister the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Administrator should deregister this SPN manually to avoid client authentication errors.

5. File Name: errorlog.4

2010-10-01 23:20:14.01 Server Microsoft SQL Server 2005 - 9.00.1399.06 (Intel X86)

Oct 14 2005 00:33:37

Copyright (c) 1988-2005 Microsoft Corporation

Express Edition on Windows NT 5.1 (Build 2600: Service Pack 2)

2010-10-01 23:20:14.03 Server (c) 2005 Microsoft Corporation.

2010-10-01 23:20:14.03 Server All rights reserved.

2010-10-01 23:20:14.03 Server Server process ID is 3716.

2010-10-01 23:20:14.03 Server Logging SQL Server messages in file 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG'.

2010-10-01 23:20:14.03 Server This instance of SQL Server last reported using a process ID of 2864 at 10/1/2010 11:20:12 PM (local) 10/1/2010 11:20:12 AM (UTC). This is an informational message only; no user action is required.

2010-10-01 23:20:14.03 Server Registry startup parameters:

2010-10-01 23:20:14.03 Server -d C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\master.mdf

2010-10-01 23:20:14.03 Server -e C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG

2010-10-01 23:20:14.03 Server -l C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\mastlog.ldf

2010-10-01 23:20:14.03 Server SQL Server is starting at normal priority base (=7). This is an informational message only. No user action is required.

2010-10-01 23:20:14.03 Server Detected 2 CPUs. This is an informational message; no user action is required.

2010-10-01 23:20:14.40 Server Using dynamic lock allocation. Initial allocation of 2500 Lock blocks and 5000 Lock Owner blocks per node. This is an informational message only. No user action is required.

2010-10-01 23:20:14.40 Server Database Mirroring Transport is disabled in the endpoint configuration.

2010-10-01 23:20:14.40 spid4s Starting up database 'master'.

2010-10-01 23:20:14.56 spid4s Recovery is writing a checkpoint in database 'master' (1). This is an informational message only. No user action is required.

2010-10-01 23:20:14.70 spid4s SQL Trace ID 1 was started by login "sa".

2010-10-01 23:20:14.73 spid4s Starting up database 'mssqlsystemresource'.

2010-10-01 23:20:15.15 spid8s Starting up database 'model'.

2010-10-01 23:20:15.17 spid4s Server name is 'TEST-STATION\SQLEXPRESS'. This is an informational message only. No user action is required.

2010-10-01 23:20:15.17 spid4s Starting up database 'msdb'.

2010-10-01 23:20:15.28 Server A self-generated certificate was successfully loaded for encryption.

2010-10-01 23:20:15.28 Server Server is listening on ['any' <ipv4> 1069].

2010-10-01 23:20:15.28 Server Server local connection provider is ready to accept connection on [\\.\pipe\SQLLocal\SQLEXPRESS].

2010-10-01 23:20:15.28 Server Server local connection provider is ready to accept connection on [\\.\pipe\MSSQL$SQLEXPRESS\sql\query].

2010-10-01 23:20:15.28 Server Dedicated administrator connection support was not started because it is not available on this edition of SQL Server. This is an informational message only. No user action is required.

2010-10-01 23:20:15.28 Server The SQL Network Interface library could not register the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Failure to register an SPN may cause integrated authentication to fall back to NTLM instead of Kerberos. This is an informational message. Further action is only required if Kerberos authentication is required by authentication policies.

2010-10-01 23:20:15.28 Server SQL Server is now ready for client connections. This is an informational message; no user action is required.

2010-10-01 23:20:15.68 spid8s Clearing tempdb database.

2010-10-01 23:20:17.03 spid8s Starting up database 'tempdb'.

2010-10-01 23:20:17.34 spid4s Recovery is complete. This is an informational message only. No user action is required.

2010-10-01 23:20:17.34 spid11s The Service Broker protocol transport is disabled or not configured.

2010-10-01 23:20:17.34 spid11s The Database Mirroring protocol transport is disabled or not configured.

2010-10-01 23:20:17.46 spid11s Service Broker manager has started.

2010-10-01 23:22:36.84 Server Server resumed execution after being idle 107 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-01 23:44:42.84 spid52 Using 'xpstar90.dll' version '2005.90.1399' to execute extended stored procedure 'xp_instance_regread'. This is an informational message only; no user action is required.

2010-10-01 23:44:51.79 spid52 Starting up database 'RFID_test'.

2010-10-02 00:10:53.76 spid11s Service Broker manager has shut down.

2010-10-02 00:10:54.42 spid4s SQL Server is terminating in response to a 'stop' request from Service Control Manager. This is an informational message only. No user action is required.

2010-10-02 00:10:54.42 spid4s SQL Trace was stopped due to server shutdown. Trace ID = '1'. This is an informational message only; no user action is required.

2010-10-02 00:10:54.59 Server The SQL Network Interface library could not deregister the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Administrator should deregister this SPN manually to avoid client authentication errors.

6. File Name: errorlog.5

2010-10-01 23:14:19.10 Server Microsoft SQL Server 2005 - 9.00.1399.06 (Intel X86)

Oct 14 2005 00:33:37

Copyright (c) 1988-2005 Microsoft Corporation

Express Edition on Windows NT 5.1 (Build 2600: Service Pack 2)

2010-10-01 23:14:19.10 Server (c) 2005 Microsoft Corporation.

2010-10-01 23:14:19.10 Server All rights reserved.

2010-10-01 23:14:19.10 Server Server process ID is 2864.

2010-10-01 23:14:19.10 Server Logging SQL Server messages in file 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG'.

2010-10-01 23:14:19.10 Server This instance of SQL Server last reported using a process ID of 2412 at 10/1/2010 11:14:13 PM (local) 10/1/2010 11:14:13 AM (UTC). This is an informational message only; no user action is required.

2010-10-01 23:14:19.10 Server Registry startup parameters:

2010-10-01 23:14:19.10 Server -d C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\master.mdf

2010-10-01 23:14:19.10 Server -e C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG

2010-10-01 23:14:19.10 Server -l C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\mastlog.ldf

2010-10-01 23:14:19.12 Server SQL Server is starting at normal priority base (=7). This is an informational message only. No user action is required.

2010-10-01 23:14:19.12 Server Detected 2 CPUs. This is an informational message; no user action is required.

2010-10-01 23:14:19.50 Server Using dynamic lock allocation. Initial allocation of 2500 Lock blocks and 5000 Lock Owner blocks per node. This is an informational message only. No user action is required.

2010-10-01 23:14:19.50 Server Database Mirroring Transport is disabled in the endpoint configuration.

2010-10-01 23:14:19.50 spid5s Starting up database 'master'.

2010-10-01 23:14:19.71 spid5s Recovery is writing a checkpoint in database 'master' (1). This is an informational message only. No user action is required.

2010-10-01 23:14:19.87 spid5s SQL Trace ID 1 was started by login "sa".

2010-10-01 23:14:19.93 spid5s Starting up database 'mssqlsystemresource'.

2010-10-01 23:14:20.32 spid8s Starting up database 'model'.

2010-10-01 23:14:20.32 spid5s Server name is 'TEST-STATION\SQLEXPRESS'. This is an informational message only. No user action is required.

2010-10-01 23:14:20.32 spid5s Starting up database 'msdb'.

2010-10-01 23:14:20.42 Server A self-generated certificate was successfully loaded for encryption.

2010-10-01 23:14:20.43 Server Server is listening on ['any' <ipv4> 1069].

2010-10-01 23:14:20.43 Server Server local connection provider is ready to accept connection on [\\.\pipe\SQLLocal\SQLEXPRESS].

2010-10-01 23:14:20.43 Server Server local connection provider is ready to accept connection on [\\.\pipe\MSSQL$SQLEXPRESS\sql\query].

2010-10-01 23:14:20.43 Server Dedicated administrator connection support was not started because it is not available on this edition of SQL Server. This is an informational message only. No user action is required.

2010-10-01 23:14:20.43 Server The SQL Network Interface library could not register the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Failure to register an SPN may cause integrated authentication to fall back to NTLM instead of Kerberos. This is an informational message. Further action is only required if Kerberos authentication is required by authentication policies.

2010-10-01 23:14:20.43 Server SQL Server is now ready for client connections. This is an informational message; no user action is required.

2010-10-01 23:14:20.89 spid8s Clearing tempdb database.

2010-10-01 23:14:22.03 spid8s Starting up database 'tempdb'.

2010-10-01 23:14:22.35 spid5s Recovery is complete. This is an informational message only. No user action is required.

2010-10-01 23:14:22.35 spid11s The Service Broker protocol transport is disabled or not configured.

2010-10-01 23:14:22.35 spid11s The Database Mirroring protocol transport is disabled or not configured.

2010-10-01 23:14:22.51 spid11s Service Broker manager has started.

2010-10-01 23:14:39.12 Logon Error: 18452, Severity: 14, State: 1.

2010-10-01 23:14:39.12 Logon Login failed for user 'sa'. The user is not associated with a trusted SQL Server connection. [CLIENT: <local machine>]

2010-10-01 23:15:13.70 Server Server resumed execution after being idle 18 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-01 23:15:20.25 spid52 Using 'xpstar90.dll' version '2005.90.1399' to execute extended stored procedure 'xp_instance_regread'. This is an informational message only; no user action is required.

2010-10-01 23:15:52.40 spid52 Starting up database 'RFID_test'.

2010-10-01 23:17:30.71 Logon Error: 18452, Severity: 14, State: 1.

2010-10-01 23:17:30.71 Logon Login failed for user 'arkar'. The user is not associated with a trusted SQL Server connection. [CLIENT: <local machine>]

2010-10-01 23:19:45.56 spid52 Using 'xplog70.dll' version '2005.90.1399' to execute extended stored procedure 'xp_msver'. This is an informational message only; no user action is required.

2010-10-01 23:20:11.53 spid11s Service Broker manager has shut down.

2010-10-01 23:20:12.21 spid5s SQL Server is terminating in response to a 'stop' request from Service Control Manager. This is an informational message only. No user action is required.

2010-10-01 23:20:12.21 spid5s SQL Trace was stopped due to server shutdown. Trace ID = '1'. This is an informational message only; no user action is required.

2010-10-01 23:20:12.40 Server The SQL Network Interface library could not deregister the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Administrator should deregister this SPN manually to avoid client authentication errors.

7. File Name: errorlog.6

2010-10-01 23:12:00.40 Server Microsoft SQL Server 2005 - 9.00.1399.06 (Intel X86)

Oct 14 2005 00:33:37

Copyright (c) 1988-2005 Microsoft Corporation

Express Edition on Windows NT 5.1 (Build 2600: Service Pack 2)

2010-10-01 23:12:00.40 Server (c) 2005 Microsoft Corporation.

2010-10-01 23:12:00.40 Server All rights reserved.

2010-10-01 23:12:00.40 Server Server process ID is 2412.

2010-10-01 23:12:00.40 Server Logging SQL Server messages in file 'C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG'.

2010-10-01 23:12:00.40 Server This instance of SQL Server last reported using a process ID of 2404 at 10/1/2010 11:11:54 PM (local) 10/1/2010 11:11:54 AM (UTC). This is an informational message only; no user action is required.

2010-10-01 23:12:00.40 Server Registry startup parameters:

2010-10-01 23:12:00.40 Server -d C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\master.mdf

2010-10-01 23:12:00.40 Server -e C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\LOG\ERRORLOG

2010-10-01 23:12:00.40 Server -l C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\mastlog.ldf

2010-10-01 23:12:00.45 Server SQL Server is starting at normal priority base (=7). This is an informational message only. No user action is required.

2010-10-01 23:12:00.45 Server Detected 2 CPUs. This is an informational message; no user action is required.

2010-10-01 23:12:00.81 Server Using dynamic lock allocation. Initial allocation of 2500 Lock blocks and 5000 Lock Owner blocks per node. This is an informational message only. No user action is required.

2010-10-01 23:12:00.81 Server Database Mirroring Transport is disabled in the endpoint configuration.

2010-10-01 23:12:00.81 spid4s Starting up database 'master'.

2010-10-01 23:12:01.01 spid4s Recovery is writing a checkpoint in database 'master' (1). This is an informational message only. No user action is required.

2010-10-01 23:12:01.17 spid4s SQL Trace ID 1 was started by login "sa".

2010-10-01 23:12:01.23 spid4s Starting up database 'mssqlsystemresource'.

2010-10-01 23:12:01.62 spid8s Starting up database 'model'.

2010-10-01 23:12:01.62 spid4s Server name is 'TEST-STATION\SQLEXPRESS'. This is an informational message only. No user action is required.

2010-10-01 23:12:01.62 spid4s Starting up database 'msdb'.

2010-10-01 23:12:01.71 Server A self-generated certificate was successfully loaded for encryption.

2010-10-01 23:12:01.71 Server Server is listening on ['any' <ipv4> 1069].

2010-10-01 23:12:01.71 Server Server local connection provider is ready to accept connection on [\\.\pipe\SQLLocal\SQLEXPRESS].

2010-10-01 23:12:01.71 Server Server local connection provider is ready to accept connection on [\\.\pipe\MSSQL$SQLEXPRESS\sql\query].

2010-10-01 23:12:01.71 Server Dedicated administrator connection support was not started because it is not available on this edition of SQL Server. This is an informational message only. No user action is required.

2010-10-01 23:12:01.71 Server The SQL Network Interface library could not register the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Failure to register an SPN may cause integrated authentication to fall back to NTLM instead of Kerberos. This is an informational message. Further action is only required if Kerberos authentication is required by authentication policies.

2010-10-01 23:12:01.71 Server SQL Server is now ready for client connections. This is an informational message; no user action is required.

2010-10-01 23:12:02.21 spid8s Clearing tempdb database.

2010-10-01 23:12:03.35 spid8s Starting up database 'tempdb'.

2010-10-01 23:12:03.65 spid4s Recovery is complete. This is an informational message only. No user action is required.

2010-10-01 23:12:03.65 spid11s The Service Broker protocol transport is disabled or not configured.

2010-10-01 23:12:03.65 spid11s The Database Mirroring protocol transport is disabled or not configured.

2010-10-01 23:12:03.87 spid11s Service Broker manager has started.

2010-10-01 23:12:10.65 Logon Error: 18452, Severity: 14, State: 1.

2010-10-01 23:12:10.65 Logon Login failed for user 'sa'. The user is not associated with a trusted SQL Server connection. [CLIENT: <local machine>]

2010-10-01 23:12:36.67 Server Server resumed execution after being idle 0 seconds: user activity awakened the server. This is an informational message only. No user action is required.

2010-10-01 23:12:36.70 Logon Error: 18452, Severity: 14, State: 1.

2010-10-01 23:12:36.70 Logon Login failed for user 'sa'. The user is not associated with a trusted SQL Server connection. [CLIENT: <local machine>]

2010-10-01 23:14:12.67 spid11s Service Broker manager has shut down.

2010-10-01 23:14:13.07 spid4s SQL Server is terminating in response to a 'stop' request from Service Control Manager. This is an informational message only. No user action is required.

2010-10-01 23:14:13.07 spid4s SQL Trace was stopped due to server shutdown. Trace ID = '1'. This is an informational message only; no user action is required.

2010-10-01 23:14:13.23 Server The SQL Network Interface library could not deregister the Service Principal Name (SPN) for the SQL Server service. Error: 0x54b. Administrator should deregister this SPN manually to avoid client authentication errors.

Appendix 24: Analysis of Acquired SQL Server Artefacts

1. Analysis of Volatile and Non-Volatile SQL Server

According to the acquired evidence of the compromised server configuration, the attacker or unauthorized user could not perform the system table updates during the attack as the value of allowing system table updates was still set to "0" (Figure 4.19).

The notable results from acquired *ring buffer* gave the timestamp of each buffer entry and the record of SQL Server events (Figure 4.23). Likewise, the acquired results in *ring buffer security error* could lead the security related errors such as login failures and events related to the security. But, the attacker did not perform the theft of SI by logging on the system directly as there were no errors in *ring buffer security error* (Figure 4.24).

As stated in Section 4.2.3.4.4, table statistics are useful for comparing against the present state of data within a table to identify the values that have been changed or updated. Hence, the last updated time of the table statistics was critical during the forensic data acquisition. However, according to the statistic information findings from the results of acquired Histogram (actual data values that were taken when the last statistics updated) against RFID Tag (Figure 4.34), Value (Figure 4.35) and Date (Figure 4.36) columns of the compromised database (RFID_test.mdf) in Section 4.2.3.4.4; the table statistics were updated before the time of investigation. Hence, the notable result of acquired current table data of the stock management database (Figure 4.37) showed that all the SI values were $600 and confirmed the database was obviously compromised.

Moreover, the acquisition results of current table data of log file - RFID_test_log.ldf (see Figure 4.41) showed the significant evidence of the attack in which the malicious SQL poisoning code was found and the SI (Tag IDs starting with E004) were updated to the values $600 at 06:39:48pm on the 12 October 2010.

The finding information related to the collation setting and data types of SQL Server (Figure 4.42) could lead to determination of how data was stored and

processed. Likewise, the acquired result of the compromised server's data page allocation (Figure 4.43) was also useful for forensic investigation. Hence, these finding results could be very valuable in analysing the transaction log and so forth.

According to the finding results of SAC of the backend server (Figure 4.46 in Section 4.2.3.4.7), it was found that the attacker or unauthorized user had not gained access to SI management database by using the weak point of server security configuration, as only the extended stored procedures such as *Server Management Objects and Distributed Management Objects (SMO and DMO XPs)* were set the value to "1 (the default value)".

2. Analysis of Residual Non-Volatile SQL Server Data

The physical database files (RFID_test.mdf and RFID_test_log.ldf) were identified (Figure 4.48 in Section 4.2.3.6.1) and collected in Section 4.2.3.6.2 as part of preservation of evidences during the investigation. Likewise the bit-to-bit POS/Server's physical hard drive was acquired for the purpose of the preservation of evidence (Section 4.2.4).

Additionally, the results of volatile active VLF data from transaction log (Figure 4.22 in Section 4.2.3.3.1), ring buffer data (Figure 4.23 and Figure 4.24 in Section 4.2.3.3.2), and reusable VLFs data (Section 4.2.3.6.2) were useful for recovering the original data. However, the data recovery was not in the scope of this research project.

Even though the results of CLR libraries were important for reconstruction of activity, there were no registered CLR libraries according to the findings in Section 4.2.3.6.3 (Figure 4.49).

As stated in Section 4.2.3.6.4, the trace files artefacts which contained data within the investigation scope were acquired and analysed from the forensic copy of the victim system in order to prevent the changes in the last accessed times of the trace file. However, the acquired trace files artefacts could not be analysed as the SQL Server Express instance was used in the experiment system. Furthermore, these SQL Server default trace files could only be opened and

analysed with SQL Server Profiler. Hence, the analyses of the default trace files are out of the scope of this research.

The acquired seven SQL Server error logs (Section 4.2.3.6.5) are analysed by using notepad++ version 5.0.3, which is an open source text and source code editor for Windows system (see Appendix 23). Even though these server error logs are useful for determining the information related to authentication, notable results concerned with the malicious attack cannot find in all the error logs within the scope of the attack except the following finding in the error log file (errorlog.1).

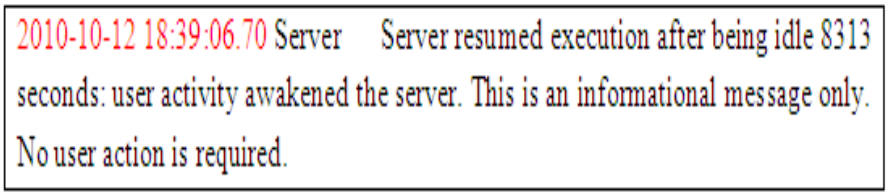


Figure 24A.1: Suspicious trace in errorlog.1

However, the malicious attack cannot be judged with the only suspicious trace found in the SQL Server error log file.

In addition to trace file logs and the error logs of the SQL Server, system event logs such as application, system, and security logs could give the potential evidence related to the investigation.

After analysing the acquired system event logs by exporting them into Microsoft Excel spreadsheets, it is noted that the findings (see Figures 24.2, 24.3, and 24.4) are not much convincible traces to prove the theft of SI in the compromised RFID stock management system.

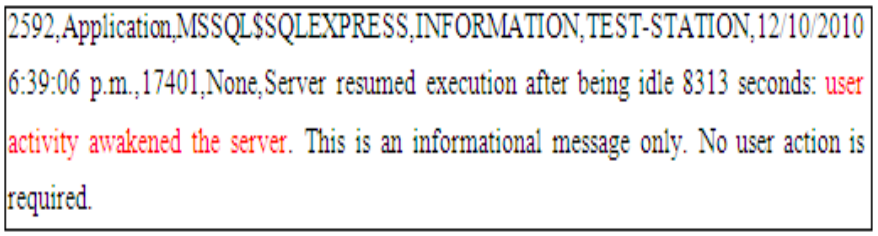


Figure 24A.2: A snippet of a suspicious trace in acquired Application Log

```
Security log on \\TEST-STATION:

D:\IR\wft\tools\sysinternals>psloglist.exe -s Security >>
E:\SecurityLog.txt

PsLoglist v2.64 - local and remote event log viewer
Copyright (C) 2000-2006 Mark Russinovich
Sysinternals - www.sysinternals.com

No records in Security event log on TEST-STATION.

D:\IR\wft\tools\sysinternals>
```

Figure 24A.3: There is no record in acquired Security Event Log

1226,System,Service Control Manager,INFORMATION,TEST-STATION,25/10/2010 5:00:37 a.m.,7036,None,The IMAPI CD-Burning COM Service service entered the stopped state.
1225,System,Service Control Manager,INFORMATION,TEST-STATION,25/10/2010 5:00:26 a.m.,7036,None,The IMAPI CD-Burning COM Service service entered the running state.
1224,System,Service Control Manager,INFORMATION,TEST-STATION,25/10/2010 5:00:26 a.m.,7035,SYSTEM\NT AUTHORITY,The IMAPI CD-Burning COM Service service was successfully sent a start control.
1223,System,Service Control Manager,INFORMATION,TEST-STATION,7/10/2010 2:20:14 p.m.,7036,None,The Windows Image Acquisition (WIA) service entered the running state.
1222,System,Service Control Manager,INFORMATION,TEST-STATION,7/10/2010 2:20:14 p.m.,7035,SYSTEM\NT AUTHORITY,The Windows Image Acquisition (WIA) service was successfully sent a start control.

Figure 24A.4: There is no significant record in acquired System Log within the timeframe of the malicious attack