Investigating Evidence

Produced by Online Session Spoofing:

The Xbox 360

Nicholas Robinson, BCIS

A thesis submitted to the faculty of design and creative technologies
Auckland University of Technology
in partial fulfillment of the
requirements for the degree of
Master of Forensic Information Technology

School of Computer and Mathematical Sciences

Auckland, New Zealand
2014

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.

.....
Nicholas Robinson

Abstract

Video games and the consoles upon which they are played have become progressively popular in recent times, however the body of work in regards to how they should be handled forensically remains surprisingly lacking. Furthermore, these consoles rely at least in part on a content distribution method known as digital distribution, wherein they offer digital goods and services to the consumer via various forms of digital storefront, as is also becoming quite common amongst the video game playing users of the PC. The protections surrounding these storefronts, beyond the encryptions used for communications and transactions, are typically account-based; this however leaves both the user and the owner of the distribution services open to a type of attack known as session spoofing, wherein an attacker masks their identity using a pre-existing and active 'session' of communication between a given service and a victim user.

As such, the aims of this thesis are twofold; the first is to explore the vulnerabilities that such a service has in the domain of video game consoles, and the second is in some way to improve the existing body of knowledge in regards to video game consoles, session spoofing attacks, and forensic techniques that can be applied to both consoles and session spoofing attacks. Towards these goals, this thesis first begins with a literature review of current publications in the domains of video game forensics, session spoofing, online marketplaces, and ultimately focuses on the Xbox 360, due to its design familiarities with the modern PC. Of prime concern within this review are the unique challenges that come from attempting forensic methods upon video game consoles, mainly due to their unique design and the heavy anti-piracy and encryption methods that are typically deployed with them.

For testing all tests were carried out in an isolated network that was compliant with university policies and the law so that the only session identification items collected were ones sent to the test computer as the result of the Xbox 360 performing its usual and expected actions for a game player. An initial test of attempting to short-circuit the login process of an Xbox 360 into the Xbox Live service, causing one user to appear as another while allowing the Xbox 360 to take care of all other communications once the spoof is successful, was attempted. This, unfortunately, proved to be untenable due to the heavy

encryption used by the Xbox 360 for its communications with the Xbox Live service. As such, a revised test is devised that instead focused on how the Xbox Live service may, through the course of normal use by a user, allow a session spoofing attack to be performed; towards this end, the traffic from a Xbox 360 is analyzed, with attention paid to the Xbox Live service, the applications that it provides, and ultimately the Internet Explorer application that is provided by the service. This allows for the observation that the Internet Explorer application (app) is a simple, albeit unpatched, version of the actual Internet Explorer program for PCs, and as such is vulnerable to session spoofing attacks using session identifications sent via IE. Unfortunately, in the process of this adjusted test, the plans for a forensic investigation of the Xbox 360 were forced to be dropped and are not part of this thesis.

From these findings, areas of weakness are identified in the Xbox Live service, and the attack itself is compared with the findings of the papers reviewed in the literature. In addition to these findings, more work still needs to be done in this area, notably in the forensic field; further exploratory research should also be made on the Xbox Live service and its kin on other consoles and the PC itself. Such work would better prepare investigators for console investigations and alert console developers to weaknesses that can be patched.

Table of Contents

Table of Figures

Chapter 1

Introduction

1.0 Background

Ever since the invention of coded messages, be they ciphers, puzzles, or nonsense given meaning only to those who sent or received such a thing, there has been an equal interest in the interception and reading of such messages. Typically, this is by parties who aren't meant to be receiving them. As society has progressed, so too have our means of communicating in such coded forms; from humble smoke signal, floriography, to the classic ciphers, we have since progressed technologically to almost all communications being in some way encrypted. Nearly any device with an internet connection of some sort will encode the information it is transmitting in some way, in order to both protect against prying eyes watching upon a network as it goes about its business, and to prevent that information from being altered mid-transit. But to begin with, there needs to be some way for the devices transmitting across a network to identify each other; and it is this process that a session hijack, or session spoof, attacks (Jess, 2006; Mocanita, 2008).

With the rise of online business has come a similar reliance on such methods of encryption to safeguard important financial or personal details. Beyond the obvious applications in the world of business, stocks, and financial institutes, one of the biggest modern industries to make use of the internet to provide their products is the video game industry. While the majority of sales are still done via physical transactions (Willingham, 2012), the video game industry is seeing a steady rise in the number of sales done digitally, through the myriad number of distribution platforms that are know available; simply upon the PC, a user can buy their preferred digital interactive entertainment via the Steam, Origin, and Good Old Games platforms. But beyond the realm of the PC are that of the dedicated machines for playing video games, that of video game consoles. Each member of the at the time of writing current generation of video game consoles, namely Microsoft's Xbox 360, Nintendo's Wii, and Sony's Playstation 3, offer some form of account-based digital distribution method, offering either unique digital-only releases or even entire games that would normally be obtained through a physical store.

While this is excellent for both the game producers and their users, it does mean that all these video games end up tied to a single account, which raises the potential subject of subverting how these systems communicate with the servers that conduct their business; namely, if one is able to obtain how a system identifies itself, it becomes possible to simply repeat the metaphorical smoke signals and convince said servers that an entity is someone other than who you they are. In other words, a session spoof attack (Jess, 2006; Reed, Taylor, Mackay, 2008).

Of course, with any such attack, there needs to be some method of defending against it; and in order to defend against something, one must learn about it and be able to identify it. This, thus, identifies three areas of potential research; how to actually commit a session spoof against digital distribution services, how to defend against such attacks, and how to identify when an attack has happened or has past one's defenses. As such, the following research question was formulated, with focus being given to the Xbox 360 due to its similarities in design, both in terms of hardware and software, to the modern PC:

> *Does the Xbox 360 retain forensic evidence in the event of a session spoofing attack?*

1.1 Motivation and Method

In Chapter 1.0, the background to this proposed project was given, as inputting the subjects of digital distribution and session spoofing. In this portion of the chapter, brief explanations will be given for the motivation of the writer in investigating these subjects of research, in order to better illuminate the reasoning behind this research. Finally, a proposed method for obtaining relevant data for these areas will be offered.

The primary purpose for this research is to add to knowledge of session spoofing vulnerabilities of the Xbox 360 console and to promote a greater understanding of the problem in general for all consoles. In the preparation for this thesis, general areas of interest to the researcher were investigated, specifically those relating to video games and the work that was currently being done with them in a forensic environment, and it was noted that there was a dearth of materials relating to games and consoles; the majority of research that could be found tended to be of a psychological slant (Bolt, 2011).

As such, the motivation for this research is to simply expand the currently available knowledge base in regards to forensics and their applications towards the world of video games and their players. Any such research to be done would thus need to provide a clear answer to the three areas of research suggested in Chapter 1.0; how an attack is done, how it can be defended against, and how it can be detected, both during the attack and afterward, the latter coming in the form of forensic evidence (Vaughan, 2004; Conrad, Dom, Craiger, 2009; Xynos, Harries, Sutherland, Davies, Blyth, 2010).

These three areas in turn offer a broad plan of attack when it comes to gathering relevant data that would be useful to the forensic community. To start with, commit a successful session spoofing attack using one of the aforementioned consoles, in this case using the Xbox 360. During the attack, note any sign that an attack is actually taking place, which may in this particular case take the form of irregularities or issues with gameplay occurring on the console. Once that is done, forensically investigate any systems used to discern any remaining evidence of the attack that has been done; in the process, also identifying any weakness in the console's security. The scope of the research will be modified according to the findings and the difficulty/complexity found in each test so that the time budget for the thesis will be complied.

1.2 Structure of the Thesis

Beyond the introduction in Chapter 1 that is currently providing a simple overview of the work to be done and the reasoning behind it, and Chapter 6, which will involve a summary of findings and a final conclusion to the thesis, there will be four main sections of work involved in this research.

Starting with Chapter 2, a literature review is presented on the existing body of work in the previously identified areas of interest. These reviews are vital in helping the researcher understand what other people working within these areas have done already; as such, it also helps prevent needlessly repeating work that has been completed. To begin with, the chapter provides a look into the current state of game console security and forensics. This primarily focuses on the subjects of video game piracy, due to the concern in the industry of the subject. As part of this, a discussion on how a video game console prevents its software and hardware from being modified will be done, with attention paid to the current modern generation of consoles as well as notable examples

from the recent history of video games. Finally for this section, a review of the current state of video game forensics will be done, in light of the previous eccentricities of video game console hardware and software. The next section of Chapter 2 will concern itself with providing an explanation and brief history of session spoofing, namely how it is done and what is required of an attacker to attempt such an attack. This will in turn be followed by the existing methods of preventing a session spoof attack, and then the methods of detecting whether a spoof is currently ongoing or has recently occurred. The rest of the section will involve giving background on online marketplaces and their gradual but growing popularity, as well as information on the inner workings of the Xbox 360, to educate the reader and explain in more detail why this variety of research is necessary.

Chapter 3 begins with an in depth review of the three existing published papers in the areas of interest to this research. Particular attention is paid to the methodologies employed by the scholars in gathering their research data, as well as the models they then employ in order to catalogue and present their findings. Finally, the research question is expanded upon, and the data requirements for this research are explained, alongside an explanation of the testing that will be required to answer the research questions.

Chapter 4 reports the findings that result from the aforementioned testing, beginning with any variations to the suggested testing, acknowledging and explaining them as they were required in order to advance the testing alongside any changes to the data collection methods that were similarly required. Once this is complete, the results from the testing will be discussed at length, explaining how the data was discovered, areas of interest within the data, and any meaningful findings that result from it.

Chapter 5 begins with a answers to the research questions. It provides answers to them in light of the processed data that was collected, as well as how the hypothesis developed conformed with the realities of the testing done. What follows the continues taking this information and refines it further from Chapter 4, taking what was learned and comparing and contrasting it with the literature review done in Chapter 2, as well as the more in-depth look into the selected papers of Chapter 3. Once this is done, recommendations for both the Xbox 360's capabilities in regards to session spoofing are offered, as well as recommendations for improving this testing.

1.3 Conclusion

This chapter seeks to present a basic background and brief justification for the work to be done, giving an outline on which the rest of the thesis may in turn be expand upon. A look into the motivations and intended methods of the researcher are then offered, as well as an overview of the contents of the chapters still to come.

The primary purpose of this thesis is to fill in an unfortunate gap in the existing work surrounding video game consoles and the application of forensic techniques to them. As such, what will be presented in Chapter 2 will be some of the existing work in these areas in the form of a literature review.

Chapter 2

Literature Review

2.0 Introduction

The world of video games is a recent phenomenon where people play intense and stimulating games online with millions of other people. The Xbox console family and Playstation console family are extensions of this world where people play singularly or in teams at rich and captivating games. The interest of this thesis is security vulnerabilities that may be exploited for ill-gotten gain by players or criminals. However the literature available on Video game forensics is extremely limited. Most of the available literature is not found in academic Journals yet, but rather in white papers, news releases, and web wiki sites. Consequently the literature reviewed in this chapter is a mixture of industry sources such as the "Digital Investigation" Journal, Microsoft white papers, and online technical details. It also tends to be descriptive and talk about possibilities rather than things in the past. Section 2.1 provides an introduction to the topic and the structure of this chapter.

2.1 Digital Change

Digital distribution is the result of the evolution of a new content delivery medium - the internet - crossing with humanity's desire for entertainment. Books have begun the transition to paperless eReader files, movies are capable of being streamed directly across the internet without ever having to permanently touch an end-user's hard drive, and even entire complex physical mechanisms can be designed, turned into schema, transmitted across the world, and built with the assistance of a three-dimensional (3D) printer. But it seems that above all, the video game market is taking the most advantage from digital distribution. Steam, the digital distribution platform created by Valve Software, has managed to surpass forty million users (Levitan, 2012). Home videogame consoles have also gotten into the act, with all three major consoles - the Wii, Playstation 3, and Xbox 360 - all offering digital distribution variations. The most popular of these consoles at the time of writing is the Xbox 360 and its distribution platform Xbox Live, which as of

October 2012 has enjoyed twenty-one months at being number one on the sales charts (Meisner, 2012).

But as purchases shift from being permanent, physical objects to being easily replicated chunks of data, the methods of identifying the purchaser become more vital than ever to prevent unauthorised copies of a game from being distributed. The abuse of these methods of identification is known as session spoofing, or session hijacking. With the popularity of the Xbox 360 being kept in mind, along with the myriad ways with which its Live service intersects with other media; including amongst them various social services such as Facebook and Twitter. There are also movies and music via Youtube, Netflix, and the Zune marketplace, amongst others, and even sports casting from ESPN and the UFC (Microsoft Corp., 2011). The risk if such heavily subscribed media is compromised is the disclosure of private information that may be used inappropriately. A spoofed Live account could potentially lead to personal information leaks and financial loss. Such disclosure can potentially give the spoofer access to all the material linked to an account and accesses to not only disrupt services but also to create fake identities and gain access to online valuables. Being able to detect such a leak on the user's end, or be able to derive evidence of account spoofing on the attacker's end, would thus be invaluable to ensuring the safety of the users and provide evidence in the event of prosecution.

This chapter contains a literature review of digital distribution platforms and their attractions to the current market (section 2.4), focusing primarily on the widespread nature of digital distribution platforms and their popularity, as well as some of the security issues that have resulted from their use. Section 2.2 will discuss existing work done in regards to game console forensics and their vulnerabilities, notably the methods used and the general design put into console hardware and software. Session spoofing/session hijacking will be covered in section 2.3, noting the method that goes into a spoof, and the techniques that have been developed to counteract attempted spoofing - meanwhile, Section 2.5 will give an overview of the Xbox 360 and Xbox Live, specifically the system architecture, hardware, and methods that are used to link bought/obtained content to the user's account, as well as the original machine it is downloaded for. Finally, section 2.6 will provide a summary of the issues thus presented

by spoofing a Xbox Live connection, and the potential forensic difficulties, while section 2.6 will provide a conclusion to the literature review process.

2.1 - Game Console Security & Forensics

A prime focus for game console design and development is preventing piracy, or making it as difficult as possible to deter would-be pirates. This is somewhat understandable, and appears to be working, when comparing the discrepancies between the console and personal computer markets: a prime example is the approximate 4.1 million illegal downloads of 2009's *Call of Duty: Modern Warfare* for the Personal Computer (PC) against its 300,000 legal sales for the months of November and December, 2009, compare this to the six million legal purchases for game consoles against 970,000 pirated within the same time span (Oxford, 2010). These numbers seem to support the widespread view amongst game developers that game console copy protection is more difficult to circumvent, versus the more widely known and understood PC architecture, as demonstrated by Evolution Studio's Matt Southern:

> *"Piracy is a serious issue in the console market. It always has been," said Southern. "I don't mean to insult hardcore gamers, I'm one myself, but I think there's probably a very strong correlation between things like early adopters of technology, high-end PC users, and at the very least, people with an understanding of the mechanisms behind games. The PC is a very open platform; consoles are more closed, which makes it a smaller minority of people, but it's still too many. It's really just fundamentally unfair." (Langshaw, 2011, para. 10)*

However, these very protective measures which make game consoles attractive to developers lead to unique challenges for forensic experts called in to analyse them - after all, the very intent behind the security systems is to prevent and obstruct analysis as much as possible. As such, this section will first examine some of the systems implemented in game consoles, and then discuss the forensic challenges that they present.

2.1.1 - Game Console Security

At their core, all game consoles are essentially pre-packaged PC systems, comprising at the very least of a motherboard, at least one Central Processing Unit (CPU) - responsible

for general logic and command processing - and Graphics Processing Unit (GPU) - designed specifically to process graphical data - various types of volatile memory, and some form of ROM to store the console's operating system. The most direct example of this would be the original Xbox, from the sixth generation of video game consoles - a term used in the video game community to refer to a particular series of console releases, consisting of the Sega Dreamcast, Sony Playstation 2, Nintendo Gamecube, and the Microsoft Xbox (Sunhede, 2008).

The Xbox, Microsoft's first console entry into the video game market and the only console of that generation to feature a standard, built-in hard drive, utilized a stripped down version of Windows 2000 to form its kernel, running Microsoft signed and approved software in the 2048 bit RSA encrypted .xbe format, and relying on a 32 bit AT Attachment (ATA) password to prevent unapproved access to its hard drive (Vaughan, 2004). As discussed by Vaughan, enterprising hackers found a method around this via the use of a integer underflow exploit, allowing them to replace the .xbe dashboard - or 'home' of the system - with their choice of alternative software, from there allowing them to load more alternate software.

With the modern generation of consoles - also known as the seventh generation, consisting of the Nintendo Wii, Sony Playstation 3, and the Microsoft Xbox 360 - the Xbox 360 (Berardini, 2005) and Playstation 3 (Altizer, n.d.) both originally came standard with hard drives, with the Xbox 360's being easily removable and replaceable due to its exterior location, while the Nintendo Wii relies upon 512MB of internal flash memory (Nintendo, 2012). Note that, other than the Wii, both the Xbox 360 and Playstation 3 have featured hardware redesign and improvements, resulting in the Xbox 360 S and Playstation 3 Slim. As a final note, all three of these systems provide methods of expanding or moving the memory from console to console, with the Xbox 360 having a removable hard drive, the Nintendo Wii having a built-in Secure Digital (SD) - a form of flash memory - card slot, and all three being able to be connected to USB memory sticks to facilitate data transfer.

The fact that these systems are removable, or that media located upon them can be changed, brings up the problem of copyright - as many games now feature downloadable content or can be entirely downloaded via the various marketplaces available, all of the

current generation use some form of unique identifier to prevent copying of copyrighted code, and to prevent unauthorised programs from being run upon them. The exact mechanism varies depending on the system in question:

The Nintendo Wii utilizes a One Time Programmable memory chip, consisting of thirty-two four-byte words (Wiibrew, 2009). As the name suggests, the chip is programmed once and is designed to never be changed again - this is important, as the chip contains many values that are vital to the rest of the system's operation. Amongst these are the unique system identifier (ID), the boot hash which is used when booting the system, and a random number generator seed - as the system also uses a system where each machine has one and only one online account associated with it, this ensures that downloads can be tracked, verified, and locked to that singular account and the machine.

The Xbox 360 uses a method of signing all attached memory media via the generation of Console Security Certificates. When media is attached and formatted to work with the Xbox 360 in question, a hidden folder (in the case of Universal Serial Bus, a cross-platform compatible data connection, also known as USB) or a portion of the memory is reserved (in hard drives), which record details in relation to which machine formatted that particular piece of memory (Free60, 2012). The resulting information is used to encode the RSA signatures - a method of public-key encryption defined by and named after Rivest, Shamir, and Adleman (1978) - that make up the bulk of the 360's software-related security. The source of this information is located on a 'secret ROM' (Fowler, 2011), having been determined to be NAND format memory (Free60, 2012), and is further responsible for allowing access to the hard drive itself - unless a Xbox 360 hard drive is unlocked via the boot process and the keys located on this secret ROM, it will not allow any form of access. As a final note, the Xbox 360 uses specially prepared hard drives, which contain a special, pre-formatted 'Security Sector' - this sector is presumably used by Xbox 360 consoles to identify the hardware as being compatible with the machine (Free60, 2012).

Finally, the PS3 system uses a series of keys that are generated based upon a single key, located within the 'bootldr' portion of the Cell architecture, known within the system as per_console_root_key_0. This key is responsible for generation of sequentially numbered keys that are used at differing levels of the hardware and software - it's direct

descendant, key_1, being used for all subsequent key generations, while also being responsible for decrypting the unique id, or EID, of the system (ps3devwiki, 2012). Again, these keys are responsible for two things - one is the prevention of unauthorised code being run on the system, and to obfuscate and stamp any digital downloads that are obtained for use on the system. It should be noted that this particular system seems to be much less researched than the other two within the hacker communities - apart from the leak of the 'master key' upon which all the other keys were based (BBC News, 2012), the security of the PS3 has proven extremely difficult to work against, not least due to the ability for Sony to quickly release software updates to block existing hacks.

2.1.2 - Game Console Forensics

As discussed above, a big issue of modern console forensics is simply getting around the safeguards that are baked into the system at a hardware level. Because of this, and due to the copyright-heavy nature of the architecture behind the systems making it unlikely or severely difficult for traditional forensic software to be run on their systems, forensic investigators need to rely more heavily on the communities that form around the public's desire to use game consoles in methods not approved by the console makers - that is, hackers.

As an example, Conrad, Rodriguez, Marberry and Craiger (2009) investigated the Sony PSP, a handheld gaming device. They discuss how, via an exploit in how one particular videogame (Wipeout Pure) handles its digital content downloading, hackers were able to discover the directory layout and gain access to these files. From there, they were able to start modifying and learning about the code base - this in turn led to the rewriting of the software responsible for the system's boot process, EBOOT.BIN. From there, software could be run despite it lacking Sony's authorization - the PSP, unlike modern consoles, lacks a method of digitally signing its downloaded software, allowing it to be shared on these unlocked handheld devices.

These hacks, however, also allowed the investigators access to the normally inaccessible files that hold the user's personal information - notably, their browsing history using the PSP's built-in web browser, their RSS histories, and internet search queries. This, in turn, enabled them to study the way the PSP handles file deletion, finding it to be surprisingly easy to recover such files - and even recover overwritten files in the process.

Their research, it should be noted, was made considerably easier by the PSP's use of the FAT16 file format. This enabled the easy use of forensic software that is normally used to analyse such files, notably Encase and FTK. Modern consoles, however, do not have this particular saving grace - modern consoles are increasingly using proprietary or altered formats.

The Xbox 360 is a prime example in this case - it makes use of no less than five identified proprietary file systems:

- File Allocation Table for Xbox (FATX), used for the storage of data on memory units. It is based on the MS-DOS file system, and is also known as XTAF due to its little endian design.
- Game Disc Format for Xbox (GDFX), which is the format used for the game discs.
- Secure Transacted File System (STFS) for the downloaded content and game saves, as well as any 'transaction' based data file that requires safe passage of separate packets - this format makes use of SHA1 and RSA signing due to its packeted nature.
- NAND File System, used specifically for the NAND 'secret ROM'. This is used for the sensitive console-specific information, such as the system's unique ID, its bootloaders, and keyvaults. This specific file format is interesting, as while the majority of it has been cracked and understood, the code responsible for detecting errors has not (Free60, 2012).

Due to this unique collection of file systems, and as discussed by Fowler (2012), this makes the Xbox 360's memory - beyond the earlier discussed need for the system to first 'unlock' a hard drive before it can be accessed - incompatible with the currently available forensic software. As the Xbox 360's architecture has become better studied and understood by hackers and users who want full access to the machine, a series of tools have been developed by these communities that can be used in a forensic context.

Two of these programs can be related directly to use by forensic investigators - the first being Xplorer 360 (oz_paulb, pedrospad, Andrew, and huceke, 2006) and the Xbox Dumper (Roofus and AngerWound, 2009). Both allow for the extraction of an image of the original Xbox's hard drive, as well as the Xbox 360 - the Xbox Dumper,

however, allows for a greater degree of control over what is and isn't dumped, along with viewing of images, and full access to all the partitions generated by the Xbox 360 when formatting one of its hard drives. It can, furthermore, dump from an attached Xbox 360 memory unit. The latter, unfortunately, requires some major physical changes to the memory unit, as detailed by maximilian0017 (2006).

The Xbox 360, of course, isn't the only console of interest to forensic investigators - heavy work has been made, both in regards to research and by hackers, into accessing the Playstation 3. Conrad, Dorn, and Craiger (2009) discussed some of the issues with investigating a PS3 console, running into particular difficulties in regard to the console's use of a proprietary operating system, file system, and the PS3's encryption of its hard drive rendered most of their tests impossible - in their suggestion for forensically analysing a PS3, they end up having to suggest doing a bit-level copy of the hard drive, followed by using the PS3 itself to analyse its contents.

There have been two major changes since the release of these suggestions - the first is, with the aforementioned release of the 'master' key for the PS3, a lot more work has been done in the area of gaining access to what Conrad, Dorn, and Craiger refer to as the 'Game OS' - this has yet to produce any tools, but means that forensic investigators can rely on more traditional methods of obtaining a image of the hard drive in question. The second major change is from Sony themselves - as of the middle of 2010, Sony released a firmware update that removed the option to install a secondary OS onto the system (Kotaku, 2010), as well as removing the option from all consoles produced since that firmware update, including the recently released PS3 Slim. This latter point is important for a few reasons - if a PS3 does have a 'Other OS' installed, it indicates that the system must have been avoiding firmware updates for a few years, and may indicate a hacked or altered system. As for data integrity, this means that the investigator must be careful not to trigger the PS3's automated updating system, or else it may mean the encryption of the 'Other OS' partition and/or a loss of data in the process.

2.2 - Session Spoofing

Session spoofing, also known as session hijacking, is given a simple definition in 'Session Hijacking in Windows Networks' (Jess, 2006, pp. 13) as follows:

> *Session hijack attacks are defined as taking over an active TCP/IP communication session without their permission or knowledge. When implemented successfully, attackers assume the identity of the compromised user, enjoying the same access to resources as the compromised user.*

Jess goes on to define the three types of session hijacks: active, passive, and hybrid. In an active attack, the attacker basically replaces the victim in the line of communication between the victim and the target server/service. This typically involves preventing attempts at communication from leaving the victim, often via the use of distributed denial of service attacks. Passive is where the attacker allows communication between the victim and the service is allowed to continue, albeit monitored - this allows the attacker to gather information as it flows, essentially letting the victim do the work. Hybrid is where the attacker shifts from passive to active midway through the process, and possibly back.

In this section, we'll be discussing the process of a session spoof, methods of prevention that currently exist, and the existing methods of detection.

2.2.1 - Spoofing a Session

Jess goes on to briefly discuss the five steps that make up a session hijack attack, as taken from Eric Cole's "Hackers Beware: The Ultimate Guide to Network Security" (2002): locating a target, finding an active session, performing sequence number prediction, taking one of the parties offline, and then taking over the session fully and maintaining the connection.

In regards to locating a target, and in turn an active session, Jess makes mention of using packet sniffers - utilities or programs designed to analyse the flow of traffic through a network connection, be it wired or wireless. This is mainly done to find a target of interest, one with high traffic to exploit and hide the additional packets going in and out. Jess makes no mention of connection types, while others such as Dave Dittrich (1999) focus exclusively on there being a direct connection between the attacker and either the target or the victim, by being on the same network. This may be due to the rise of the wireless network as being a common means of connection to the internet being relatively recent - these days, just as a wireless connection allows a roaming user to drop

in and out, it also allows for a spoofer to sidle in and monitor the incoming/outgoing traffic for anything interesting. It is at this point that a passive hijacker would stop, only observing - active attackers, or after a time hybrid attackers, would move on to attempt a complete hijack.

Sequence number prediction is a tricky yet necessary step to being able to hijack a session. When packets are transferred across networks, they are given an identifying number that help the receiver determine what order to arrange everything in, and what commands should be run in which sequence. However, as explained by Himanshu Arora (2012), this is only part of the process - in order to make use of the sequence number, the attacker needs to silence their victim first, usually through a simple denial of service attack, in order to prevent them from 'catching up' to the predicted sequence number and leaving it viable. This makes up the fourth step of the process - the fifth is comprised of sending off a faked packet, complete with the victim's Internet Protocol (IP) address and the sequence number, along with whatever data/commands are expected. This could involve, as demonstrated by Dittrich, abusing the security of the target to create a backdoor, allowing the hijacker to return the session to the victim and simply regain access via the backdoor.

2.2.2 - Preventing Spoofing

The art of preventing spoofing involves attacking the limitations of the spoof itself. Key amongst these is the act of guessing the sequence numbers, as discussed above - unfortunately, the sequence numbers are generated via very simple algorithms, making guessing usually a simple process of monitoring and timing. This appears to be due to the lack of cryptographic requirements for TCP/IP connections - that is, connections based upon the Internet Protocol Suite standard, which is usually referred to as TCP/IP due to the first protocols to be included within the standard, the Transmission Control Protocol and Internet Protocol - as discussed in the RFC 1948 informational paper (Bellovin, 1996), part of a series of Requests For Comments memorandums published by the Internet Engineering Task Force, which provide various suggestions, standard updates, and various errata in relation to the internet or services based on the internet. However, a recent standard suggestion provides both an updated discussion on this very topic, as well as a solution, in RFC 6528 (Gont, Bellovin, 2012), which improves upon

the former algorithms via the use of a MD5 hash. However, this in turn suggests that sequence numbers be avoided entirely, via the implementation of either IPsec as described under RFC 4301 (Ken, Seo, 2005) or TCP-AO as per RFC 5925 (Touch, Mankin, Bonica, 2010), which replace the predictable sequence numbers with full cryptographic encapsulation of the data being sent, making traditional hijacking attempts far more difficult.

Another option, as presented by Doeppner, Klein, and Koyfman (2000) is through the use of router stamping. This technique was originally developed to fight against distributed/denial of service attacks, using spoofed IP addresses - much like the spoofed addresses used when transmitting the faked packets in a session hijack attack. It involves appending to the packets a certain number of stamp slots, recording the routers and ports used for the transmission - with deterministic stamping, this is done with the IP of every router encountered, which would produce massive sizes. The suggested algorithm appends a certain probability of recording as the packet is sent, with a limited number of slots to be filled - the result is a slightly larger packet, but one whose path could be traced.

Note, however, that this technique does have a weakness with session spoofs compared to its intended use against distributed/denial of service attacks: there is usually one hijacker attempting a single assault, which means that they may be on the same 'path', router-wise, as the victim. This would result in a situation where the packets being sent would receive near identical/acceptable stamps as the genuine ones being sent by the victim - but, of course, would also require the hijacker to be in much closer proximity to the victim, potentially causing them to leave evidence behind from the connection being established.

2.2.3 - Detecting Spoofing

Detecting an incoming spoof is extremely difficult - despite heavy research, there is little to help the defender prepare for an attack in advance, due to the hijacker beginning by simply allowing traffic to flow as normal. But this doesn't mean that it's impossible, however. Frustratingly, there exists a distinct perceived lack of suggestions to improving wired networks, with the majority of the available literature focusing on wireless

networks instead, due to the vulnerabilities created from any user being able to connect from any point.

Gill, Smith, Looi, and Clark (2005) discuss some of these methods of identification in regard to wireless networks, notably focusing the then current focus on the detection of unauthorised stations and access points, which could be added by the hijacker in order to facilitate connection to the network. For session hijacking in particular, the only methods truly available to them was via detecting sudden jumps or randomness in the sequence numbering of the packets, or that the MAC of the hardware being communicated with was valid - although this latter element could be easily spoofed.

They go on to suggest two additional methods that could allow for better detection of hijackers accessing the system, namely via monitoring Received Signal Strength (or RSS), and the round trip times of the Request To Send/Clear To Send (RTS-CTS) handshake, which prepares a pathway through wireless network nodes for data to travel along, needed to prevent multiple wireless devices using the same network from interfering with each other's data. In the former case, the signal strength of the packets being received by a station can be used to calculate the distance of the source - the strength is negatively affected by intermediary barriers, such as walls and floors, allowing for a certain 'safe' strength to be set, and then in turn relying on the physical security of the building or location to prevent internal access and circumvention of this suggestion. This has the potential to foil attacks before they even happen.

With the latter suggestion, what is instead targeted is the spoof itself - an expected duration for a given handshake (the process of the two ends of communication recognising each other) is calculated. As packets are sent, if they diverge from this predicted time, it can be used as an indication that another source is then suddenly sending packets - of course, there is the issue of a bad connection suddenly cropping up, potentially limiting the usefulness of this tactic. Of note is that this could be used similarly for smaller wired networks with very few changes.

Chakravarty, Portokalidis, Polychronakis, and Keromytis (2011) inspected the use of the Tor protocol in conjunction with session hijacking. The Tor protocol works by re-routing traffic streams through volunteer-run nodes, essentially scrambling any attempts

to trace the route back to the originator - this also has the effect of causing neither end to be able to tell the IP address of the other. This, obviously, makes the process of spoofing any packets that are detected and successfully slipped into the network that much easier.

What is proposed is that, if a set of credentials or personal information is captured by a hijacker or snooper and later used, that event can be monitored - although the hijacker will, of course, be able to get in. What they suggest is to set up a dummy server and client, supplying dummy packets of false information across the network via periodic faux login attempts - that is, 'fake' login attempts meant to fail - using specific entry and exit points and password/login pairs that change with each faux login attempt. If that false information is later attempted to be used, then it can be presumed that there is a hijacker monitoring the traffic as it comes and goes - using the password and login pair, it can then be determined where on the network the hijacker is listening from.

2.3 - Online Marketplaces

Over the years, as the internet has changed and improved, so have the practices of distributing and purchasing games. Consumers now have a choice between purchasing physical copies of their games in-store or purchasing them via online digital distribution systems - or even downloading demo of something particularly interesting so that they might try before they commit. This also gives the publishers of videogames options of how to handle their content distribution, leading to the term downloadable content, or DLC, which can be used to expand a game with new options, levels, and assets.

It is hard to understate how online marketplaces have changed things for both the public and producers of videogames. According to Gartner, in an article by Jeff Beer (2012), it is expected that sales of videogames will climb to $112 billion U.S dollars from the 2011 level of $74 billion. This is despite, since 2008, sales of physical videogames have been in a steady but constant decline. The mobile market - that is, games designed and marketed towards use on handheld phones or similar small-scale multimedia devices - is similarly expected to almost double from its 2010 level of $5.6 billion to $11.4 billion by 2014. It isn't just the bigger names who're profiting from this shift in delivery either - smaller companies are able to produce and deliver their content without having to deal with the massive overhead of packaging, delivery, and dealing with distributors. To

quote Rodney Gibbs, of Ricochet Labs, “It makes businesses like mine possible” (Gaar, 2011).

This chapter will be focused around a discussion of the impact of online marketplaces on the digital distribution of videogames, focusing on and contrasting two of the largest examples to date: PC’s Steam platform, and the Xbox Live Marketplace.

2.3.1 - Steam

Steam got started on the 12th of September, 2003 (Valve, 2003). It was released, and required for use, during the beta of the 1.6 version of the then extremely popular Counter-Strike mod - that is, an add-on/expansion of their original hit game, Half-Life - garnering roughly 300,000 simultaneous users (Bode, 2003). Since then, however, Steam has grown to be the absolute market leader within digital distribution, consisting of roughly 54 million registered users with a peak concurrent rate of over five million (Suddi, 2012).

Steam has also set a benchmark in regards to the way a platform such as this can interact directly with the products it is advertising. One of Valve’s games, Team Fortress 2, is a first person, multiplayer team-based shooter with varying goals depending on which map players are currently playing on - with the exception of the Mann vs Machine mode, which involves all players fighting off increasing waves of robotic clones, the modes pit a red team against a blue team, attempting to capture specific points on the map, transfer some secretive briefcase from the enemy’s base to their own, transport a massive bomb into the enemy’s base, or simply decimate each other entirely. During all this, the players receive additional weapons and clothing to outfit the nine available ‘classes’ of mercenaries they can play as, gained via either random chance during play, by completing certain ‘achievements’, buying them directly from the in-game real-money store (which is integrated in turn with a intermediary between the player and the store purchases via Steam, the Steam Wallet), buying other games on the Steam network, or by a combination of the above - specifically, having a locked crate (which in turn contains a random item determined upon being ‘unlocked’) being randomly given to the player, which has to be unlocked by a key bought from the online store. These items have, in turn, developed into a full-fledged internal economy - centered specifically around the available hats.

The hats available in the game, introduced during the Sniper vs Spy update (Valve, 2009), do not have any effect upon the gameplay in any way, shape, or form. However, those hats that are obtained via crate unlocking have a small chance to be 'Unusual', exhibiting additional particle effects when worn. These hats command an extremely high price inside the game - and outside. These hats became so valued by the community that, in 2011, Valve went so far as to hire an economist to help continue to grow and balance their personal economy (Griffiths, 2012) - who went on to publish a paper explaining how the community had come to develop their bartering and pricing system for hats, developing a exchange rate between the differing types and levels of goods (Varoufakis, 2012). A well stocked 'backpack', which contains a player's items and is directly linked to their Steam account, can be worth tens of thousands of dollars based upon the current economy, with players developing tools to track the changing trade price of particular items - notably, backpack.tf (backpack.tf, 2012).

This is merely within a singular game - a player's account can typically contain several dozen games from multiple developers, with some players accruing lists of hundreds of games, typically picked up during the various special Steam sales. These games can at any time be redownloaded and reinstalled to any number of machines - however, a Steam account can only be logged in from one computer at a time, as a method of preventing piracy. However, Steam does allow certain games to be played via an offline mode, though it maintains an internal check of when the platform last logged in, requiring some periodic activity from the user to prevent their games from becoming (temporarily) unusable.

Steam, however, is not a completely secure system. A recent exploit was discovered within the internal browser that can be used by the player to visit web pages while playing a game - and also do various internal commands, using the "steam://" prefix. Security researchers at ReVuln found that they could use this, along with various overflow methods, to inject code into remote memory, or simply abuse in-game systems to create entire batch files from thin air (Orland, 2012).

2.3.2 - Xbox Live Marketplace

The Xbox Live Marketplace actually predates Steam - it was launched in November, 2002 for the original Xbox console, one year after the console's North American debut,

with this particular iteration continuing on until April of 2010 (Peckham, 2010). The same name was also used for the more advanced Xbox 360 version, which debuted with that particular console in 2005 - it was also extended and became able to be integrated with the Windows OS series XP through 7, under the name Games For Windows Live.

Unique amongst the distribution platforms offered on PCs and other consoles, Xbox Live offers a tiered subscription system, of two levels - Xbox Live Free (formerly Xbox Live Silver) which costs nothing, and Xbox Live Gold, which requires monthly-or-longer subscription. The player can gain access to the online marketplace with either subscription, but only the Gold accounts are allowed access to multiplayer services within games, apart from select 'Free' weekends. In contrast, the other equivalent generation consoles offer fully free online access, albeit still with some form of account/login system. It should be noted that the number of users with Xbox 360s does not directly correlate to the number of Xbox Live accounts - of the over 66 million consoles currently in use, only roughly 40 million of their users have accounts (XboxLiveTV, 2012).

As mentioned earlier, the Xbox 360 uses a 'Secret ROM' to encode its games - this is most important when it comes to the Xbox Live service, as this same Secret ROM is responsible for the generation of per-account-and-machine licenses. Licenses are maintained on the Xbox Live servers, forcing the games and media downloaded to be played only on that hard drive that the purchased was made upon, and only with the account that made the purchase in question. A user may switch the Xbox Hard Drive from one console to another, but can only use the content downloaded to that hard drive if they in turn log into their Xbox Live profile for the duration of use - however, it is possible to transfer licenses once per four months from one hard drive to another (Microsoft, 2012).

Unlike Steam in its current state, Xbox Live features far greater interconnection between the Xbox Live account and other media distribution systems, as mentioned earlier. A user's Youtube, Netflix, Hulu Plus, and Zune accounts can all be linked into the system to allow for various media streaming, downloading, and uploading options, not to mention the various other media partners that Microsoft allows access to the service - this includes region-specific partners, such as FOXTEL in Australia and BSkyB

for U.K users. Furthermore, Xbox Live features heavy social integration with the user's Facebook and Twitter accounts, and certain Live-only features, notably the Video Kinect service - allowing users to use the Kinect add-on for video chat - and Xbox Live Parties, where multiple users can join together over the service while they play the same or different games, allowing them to use voice chat with each other as they play.

Similarly to Steam, however, the Xbox Live service does not have a spotless record. After discovering that his account had been broken into, Jason Coutee discovered an exploit on the Xbox.com website - which is separate from the Xbox Live service, other than allowing for users to login to manage their accounts - that would allow hackers to attempt an infinite number of passwords using a simple script (Dyer, 2012). There have also been reports of compromised accounts being used as mules for black market 'Microsoft Points' - the currency used on Xbox Live, bought with real money - that are siphoned off the account via abuse of the Family Gold Pack, a method of transferring Microsoft Points from one account to another (Plunkett, 2012).

2.4 - The Xbox 360

This section will focus on the Xbox 360 console itself. As previously mentioned, the current user base of the Xbox 360 is around 66 million - it holds the largest market share of the current generation of consoles at the time of writing. This makes it interesting to forensic investigators, not just because of it being the most likely of the consoles to be owned, but also due to it having been developed by Microsoft - as such, several of the hardware and software components (as will be discussed below) have striking similarities to those used in conventional PCs and the Windows series of operating systems. To begin with, there will be a overview of the hardware that makes up the Xbox 360, focusing particularly on the memory storage available to it. There will be a brief overview of the software formats used by the Xbox 360, and then a similar overview of its available accessories, with particular attention paid to items of forensic interest.

2.4.1 - Hardware

There are currently two models of Xbox 360 on the market - the original 'fat' Xbox 360, and the newer 'slim' Xbox 360 S. At the moment, the available research is heavily

biased towards the original ‘fat’ hardware, and as such, this section will focus on that, followed by a brief discussion of the changes present in the Xbox 360 S.

As laid out by Andrews and Baker (2006), the Xbox 360 features several areas of interest to a forensic investigator. The overall design is centered around the GPU, consisting of a pair of memory controllers, a core specifically for 3D rendering, a video out (which in turn attaches to a analog converter chip), and ten megabytes of eDRAM - a form of dynamic random access memory (DRAM) that stores each bit within a separate capacitor, which is embedded (hence the ‘e’ in eDRAM) with the processors of the system. Both memory controllers are in turn connected to 512 megabytes of DRAM. Meanwhile, the GPU interacts with the CPU through a BIU/IO interface, feeding into a 1 megabyte level two cache on the CPU. This cache is then attached to a series of three pairs of level one caches, one of the pair dedicated for incoming streams and the other for outgoing streams - these pairs, each 32 kilobytes in size, in turn feed into the three cores that make up the main horsepower of the system. Finally, the BIU/IO interface - located between and acting as a go-between for the Bus Interface Unit (BIU), responsible for accessing memory busses, and the Input/Output (I/O) chip - in turn feeds into the I/O chip, consisting of system management controller and a Xbox 360 media audio decoder. The I/O chip, with the exception of the analog chip, controls all the connections to outside media - it is connected to the built in DVD drive via a Serial ATA (SATA) connection - a bus interface standard, designed to connect the other hardware components with a form of mass media, such as hard drives or optical drives - as well as another SATA connection for the hard drive port. There are two front USB connections available, typically used for controllers, a wireless connection for use with the Xbox 360 wireless controllers, and as well as three more USB ports located on the back - two of these are intended to be used with Xbox 360 compatible memory units, while one is a generic USB for use with accessories. The unit also includes ethernet, Infra-Red (IR), audio out, flash, and system control outputs/inputs.

Previous sections have already covered the ‘Secret ROM’ and the hard drive itself, and so this section will not repeat those particular sections. Of note, however, is a major difference between the Xbox 360 and the Xbox 360 S - the former has no wireless internet or networking capabilities without the use of a specialised Xbox 360 wifi

adaptor. The latter, however, has a built in 802.11n WiFi connection. There is otherwise a frustrating lack of exact information on the internal components of the Xbox 360 S, although there are 'deconstructions' of the hardware available for viewing online - notably, a rather in depth investigation done by Ryan Shrout (2010).

2.4.2 - Software

Other than the ubiquitous video games available for the console itself, distributed either via the Xbox Live service or on the DVD format game discs, the Xbox 360 relies heavily on its built in software to allow it to run. When a Xbox 360 is first powered on, a series of bootloaders run, beginning with a one located within the CPU itself - this is responsible for decoding the secondary bootloaders, following computation and verification of its RSA signature. At this point, if the signature is valid, the process continues down the chain. The second boot loader is responsible for decrypting and starting the Xbox 360 virtual machine, upon which the rest of the system relies on - again, this process features heavy verification to ensure that everything is valid. If so, the process moves onto the third boot loader, responsible for the decompression of the base kernel of the system. It's at this point that the system will also check for and apply any post-reboot patches it may find, using yet another boot loader - during which the third boot loader remains in memory. Once this process is complete, the user is presented with the Xbox 360 dashboard (Free60, 2012).

Within the dashboard, the user is able to access a wide variety of options. Of particular note for investigators would be the messaging system - when connected to the Xbox Live network, the user is able to send messages to other users, via the use of 'Gamertags', or their login username. These messages can consist of video, audio, or just plain text, and are also able to be used to invite other users to games or to join up into parties to allow for ease of communication.

Beyond the console itself, the system is also capable of integrating itself with Windows Media Center. This process is extremely simplified for the user - they are simply required to install the Media Center software on either a Windows 7 or XP system, and connect to a wired or wireless connection. After following some basic steps on the PC system, the two will automatically be able to connect and share media between each other, essentially turning the Xbox 360 into a very versatile media center

(Microsoft, 2012). This has a potential for abuse, however, combined with the Xbox 360's extremely well protected hard drive - it would be a prime method of storing data off a main system, as long as a user could convince Windows Media Center to transfer the data.

2.4.3 - Accessories

The Xbox 360 features a wide range of accessories. Of particular note, exclusively for the older consoles, is the Memory Unit - attached via the two slots in the front of the console designed specifically for memory units, these are essentially memory sticks of various size, ranging from 16 gigabytes up to, with some third-party version, 256 gigabytes. They are incompatible with the newer Xbox 360 S, which dropped the ports needed - furthermore, Microsoft themselves have stopped producing the units, though third parties do have their own versions available (Microsoft Support, 2012).

Another key accessory in the old model of Xbox 360 was the wireless adaptor. The original was compatible with 802.11a/b/g networks, while a later edition dropped 802.11a support for 802.11n support, to allow for multiple connections. Again, these are not widely available due to the release of the Xbox 360 S, which features the unit built in.

Finally, it should be noted that the Xbox 360 did originally feature a webcam available for it, allowing for video to be recorded and attached to its messages. This has in turn been supplanted by the Kinect, an additional piece of external hardware placed near the console, which is capable of analysing the motion of people in front of it to a very fine degree. This is typically used in games to allow users to control the games with their body, but it also can be used as a traditional webcam - some users have been known to purchase the Kinect without having a Xbox 360 in the first place, in order to use its 3D analysis systems or its motion capture capabilities for their own purposes. This was enabled due to a release of community-developed open source drivers, creating a small community centered around exploitation of the hardware for various reasons (Adafruit, 2010).

2.5 - Summary of Issues

The above sections demonstrate the many challenges involved in the forensic investigation of session spoofing, particularly as it pertains to consoles. While there is

plenty of literature available on the subject of session spoofing itself, and the Xbox 360 console, there is little overlap between them other than the subject of WiFi communication. There was also no literature found on the subject of detecting or finding evidence of a past hijack attempt, be it via wired or wireless networks.

The primary result of this is causing it to be extremely difficult for investigators to work with a Xbox 360 for forensic purposes. Overcoming the anti-piracy measures and ensuring that a copy of the hard drive and the underlying memory can be obtained will be the first hurdle, followed by attempting to identify any signs of misuse via any log information contained within the Xbox 360's other memories.

These aren't the only issues that need to be overcome, however. A brief summary of others will follow, but one particular issue of note is that of the modular nature of the Xbox 360's non-volatile (hard drive or memory-stick based) memory. It is, as has been mentioned above, that it is extremely easy to just swap out a Xbox 360's hard drive with another - as a result, and in order to reach the goal of being able to provide a full forensic perspective, it will be necessary to see if the Xbox 360 tracks the hardware that has been attached to it.

On the note of modular systems, there is also the Xbox 360's capability to associate itself with third-party services - as noted before, Netflix, Facebook, and the like. As a result, if the Xbox 360 is compromised via a session spoofing attack, there is the chance that the attacker could in turn continue the attack via spoofing these third party login systems, or obtain their details via the Xbox 360 itself - as a result, it will be vital to see what information is stored in the Xbox 360 about these third-party services, and how it uses those login details.

And finally, there is the concern that the security measures that have been implemented by Microsoft will mean that the Xbox 360 will simply be immune to a session spoofing attack. In such a case, the research to be done will switch focus from the spoofing attack and to focusing on the other points covered in this summary - that of tracking the hardware used with the Xbox 360, overcoming the anti-piracy measures in order to provide a high quality forensic examination, and how open the Xbox 360 is with the information it stores on third-party services.

2.6 - Conclusion

With this literature review, it has become apparent that there is a unique gap in knowledge surrounding spoofing a console's session. While methods exist for the forensic extraction and analysis of the Xbox 360, these may miss certain internal caches and memory beyond the unit's hard drive. Furthermore, there is the question of what protocols, if any, the Xbox Live service uses when it communicates between a user and its home servers.

The existing work done by online hacker communities, such as Free60, offer a unique perspective for investigating these holes. By using the information presented through this chapter, particularly section 2.2 and the existing session spoofing literature - both in how-to and in detection - a viable solution and set of guidelines for forensically analysing session spoofing in regards to the Xbox 360 should be able to be produced.

Chapter 3 will detail the steps required for the development of such literature - focusing first on successfully spoofing a Xbox 360 session. Then seeing what will in turn be made available to the hijacker from both the console itself and the server it's communicating with, and finally go through the act of extracting everything possible from both the victim and attacking units, to find forensically sound evidence fit to be used in a court of law.

Chapter 3

Research Methodology

3.0 Introduction

In Chapter 2, a review of current literature available on the subjects of video game console forensics and wired/wireless session hijacking was done in order to help define the current scope of the thesis, and the thesis topic at hand. In this chapter, a small number of papers that were part of the available current literature have been chosen for closer scrutiny - specifically, in their process of creating models to support their research and findings, as well as the logic behind those choices, and existing methods used in that process of selection (Section 3.1). The approaches and technical specifications will help me structure what is possible and give practical guidance on what may be achieved in developing a research methodology. These processes will in turn be used to assist in the definition of a research methodology (Section 3.2) - this will also include the development of the research questions and hypothesis, expanding on the critical areas identified in Section 2.5. From there, the data requirements, the method of selecting samples, and how those samples will be processed and presented will be the focus of Section 3.3. Finally, Section 3.4 will present the limitations of the research to be undertaken, followed by a conclusion and summary of work in Section 3.5.

3.1 Review of Related Publications

In the following subsections, three existing items of relevant literature have been chosen for closer scrutiny and to learn how others do this type of research.

The first paper, Xbox Security Issues and Forensic Recovery Methodology (Utilising Linux) by Chris Vaughn (2004), examines forensic methods and their use in regards to the original Xbox, along with the security challenges it presents. It provides a step by step breakdown of the Xbox and its features, hardware, and possible modifications that can be applied to the system - furthermore, it also provides guidelines and instructions on how these features can be analysed forensically. This will help

provide structure in how to report the findings on the forensic investigation portion of the research to be done in Section 3 and 4.

The second paper, Session Hijacking in WLAN Based Public Networks by Ørjan Bækkelund (2009), was chosen due to its similarities with what will be needed in order to test session spoofing with the Xbox 360. As part of the research presented, a setup of three systems were used - one to act as the attacker, one as a legitimate client, and the third to act as a monitor of the other two. In particular, it focuses on the wireless version of such an attack, detailing how the attacks were done and with which settings, a discussion that will be similarly necessary when it comes to Section 3.2 of this chapter.

The third paper, Passive Techniques for Detecting Session Hijacking Attacks in IEEE 802.11 Wireless Networks (Gill, Smith, Gooi, and Clark, 2005), focuses on techniques designed to discover session spoofing/session hijacking without the need for active monitoring. This will be vital for the part of this chapter focusing on identifying evidence of session spoofing as it relates to the Xbox 360 - it is important to note that the researchers once again utilised an attacker-client-monitor setup as the second chosen paper of this section.

3.1.1 Xbox security issues and forensic recovery methodology (utilising Linux)

Vaughan begins by exploring what makes the Xbox 360 unique for forensic investigators - and where it is quite familiar. Describing the Xbox 360 as ”internally at least, a legacy free PC; a computer containing a motherboard without the traditional set of ports, such as parallel, PCI or PS2” (p. 165), the paper goes on to briefly discuss what make the Xbox 360 unique compared to it’s PC compatriots and how it can be altered to act more like an actual PC; of note are software exploits to force the system to load software that has not been signed and approved by Microsoft, and hardware exploits that circumvent the hardwired security systems located in the Xbox 360. These issues are important not just because they allow the system’s user to alter the machine and perhaps hide information within the Xbox 360, but because the software/hardware security can hinder forensic investigation efforts.

The main focus of Vaughan’s work is on these security issues, how they may have already been circumvented, and how to circumvent them in a forensically sound manner. Beginning with the ATA password protection (p. 166), methods are proposed to

first see if the Xbox 360's security has been compromised and is thus of possible evidentiary value. Of the methods proposed, the quickest and most forensically safe is the simple insertion of either a Linux disc or a Evox (a unapproved replacement for the standard 'dashboard' or Xbox 360 operating system) disc. If the discs load, then it indicates that the machine has been altered to allow for unapproved software to be run. Not that these are the only potential signs in weighing the evidentiary value of a Xbox 360; visible hardware alterations are also touched on, such as non-standard hard drives, broken warranty seals, and fully-installed games that can be launched without their discs are also indications that the Xbox 360 has been altered.

From there, and presumably once the investigator has either discovered the above alterations or has been forced to use other methods to gain access to the hard drive, a simple process is given for obtaining a bit-by-bit image of the Xbox 360 hard drive (p. 168). Following this, the Xbox 360 file system is briefly explained, identifying the major sections of the file system, their size, and their starting offset, along with an explanation of their contents, albeit based off of then-current 2004 kernel and hard drive size. While not noted in Vaughan's work, it should be added that this information may not hold true for the now-current 2013 models.

A description of the FATX file system that the Xbox 360 uses is then offered (p. 169), paying particular attention to how the directory entries are formatted, along with the hexadecimal value given to 'deleted' files (0xe5) to identify them for being overridden later by new files. After this is done, a section is devoted to the method of getting the FATX format to work alongside a Linux operating system, due to FATX's proprietary nature. This is in turn followed by a series of suggestion on how to actually view the file system once it is recognisable, with a focus given to using either NASA or standard Linux loopback drivers, loopbacks being special drivers allowing for files to be treated as hard drives by the system. This will allow for the viewing of undeleted files.

The rest of Vaughan's work focuses on the act of searching through the data thus derived from the accessed image using text analysis and data carving, after a brief explanation of Evox and what information can be derived from its ini file's headers (p. 170). For string searching, it is suggested that the strings are extracted from the data using a series of Linux commands, after setting the files to read-only to prevent any

possible writing, after which they can be examined using standard keyword/grep expressions. As for data carving, due to the FATX file system in use by the Xbox 360, a non-file system reliant data carver needs to be used, with a focus given to Lazarus of The Coroner's Toolkit, although a suggestion of using the data searching tool Foremost is also given for the data thus extracted. Importantly, this last section discusses the retrieval of deleted files from the Xbox 360 hard drive used for the experiments (p. 171) using Foremost, finding that roughly ninety percent of the 'deleted' data was then recoverable, with the other ten percent presumably overwritten by the file system.

3.1.2 Session hijacking in WLAN based public networks

Bækkelund begins by giving a brief description of Wireless Trondheim, a wireless LAN that offers free internet access, in the center city area of Trondheim, a city/municipality located in the Sør-Trøndelag county of Norway. The text briefly explores how the open nature of this network leaves itself open to exploitation due to various flaws in the network's security, before explaining the goal of the thesis; the development of a system to detect session hijack attempts made with the network, and the automated logging of any such attempts.

The second chapter focuses on giving a full explanation of the Wireless Trondheim network. Two methods are available to the user in order to gain full network access. The first is a web portal that users can spend money through via SMS to gain three hours of access time via, or they can alternately login using a Norwegian University of Science and Technology (NTNU, abbreviated from the Norwegian name for the university) login. This initial access confirmation is done via an encrypted SSL connection, but the communication done thereafter is in unencrypted. The alternative access method is WPA2, offered to those who have access to the Eduroam roaming infrastructure that connections various universities across the world, using the 802.11i standard, giving it the higher security that is inherent to 802.11i connections.

From here, an overview of the network architecture is offered; various wired and wireless Access Points (APs) are filtered through Wireless LAN Controllers (WLCs), in turn feeding into a Nomadix brand gateway, which maintains a whitelist of allowed sites any user can access, redirects users who have not logged in or bought time to the Portal login, and maintains a list of MAC addresses of users who have successfully logged in and been

granted full access. This then feeds into an exploration of the wireless MAC layer used in the 802.11 standard, covering the TCP/IP layers, the MAC header format used for data transfer, and how those MAC headers may be changed to spoof where a connection is coming from.

Finally, after briefly exploring existing literature on this topic, the ways that the above information can be used together are explored, outlining three such examples of MAC address spoofing; Session Hijacking, where the MAC addresses of a client and a AP are used together, the latter to fake deauthentication messages to the client, disassociating them from the network, and the former to then claim a session under the client's MAC address; Freeloading, where the attacker simply takes on the IP and MAC address of the client without disconnecting them; and finally Waiting for Availability, where the attacker simply waits for the client to disconnect without ending their session with Wireless Trondheim, letting them then simply assume the client's identity and make use of the session.

This leads into an exploration of the countermeasures available, as well as the topic of Intrusion Detection Systems (IDSs). After briefly covering the nature of IDSs and how they detect harmful network traffic - via the use of either statistical analysis and anomaly detection, or signatures that identify patterns of attack - the various countermeasures available for use against MAC spoofing are offered. In brief, the Session ID method attaches a simple cookie that requires periodic and automated refreshing to the user once they log in to the Portal, which an attacker would then lack, and in turn be able to be detected as they try to make use of the session. Tracking MAC Frame Sequence Numbers allows for detection due to how the generation of MAC frames occurs - as an attacker sends information, their Frame Sequence Numbers would slowly go out of synch with the client's, allowing for detection in the case of Freeloading or possibly hijacking attacks. Received Signal Strength is based on the idea that, as a user connected to a AP is relatively stable, their signal strength to an AP is equally stable; in turn, an attacker connecting from somewhere else could physically have a different signal strength, and thus be identified. Finally, the RTS-CTS Handshake method is explored, where the time it takes to complete a Request To Send - Clear To Send handshake can

vary due to the physical location of the users, which in turn can be used to identify a connection apparently coming from two places at once.

Moving into the act of actually attacking in Chapter Three, Bækkelund lays out how they will fake an attack for the purposes of penetration testing, using a Client-Attacker-Monitor setup, and detailing the specifications of each machine. The software to be used is then given focus; Backtrack, a suite of penetration testing tools; Kismet, a wireless traffic analysis tool; Aircrack-ng, a wireless penetration suite which can be used to circumvent the various security implemented in wireless connections; and Wireshark, which can also gather wireless traffic but is able to extract more information from the various protocols that may be used. This chapter then continues on to explain the setup of all these tools, their uses, and the intricacies of making an attack, on both Linux and Windows systems.

The rest of the paper concerns itself with using the test attack network described in Chapter 3 alongside the various countermeasures and attacks from Chapter 2. Ultimately, it finds that the Session ID countermeasure would not be appropriate, due to needing a separate window open to maintain the connection, which might be closed due to the client not understanding why it is open. The countermeasures relying on physical information - Received Signal Strength and RTS-CTS Handshake - are also not recommended, mainly due to the difficulties in recording and comparing the information needed for these countermeasures in a central location, and the unexplored problems that might arise from them being deployed in a wide, public area. However, MAC Frame Sequence analysis is identified as being the best fit for what Wireless Trondheim has requested, due to it not requiring any information bar the frame sequence numbers, the lack of intrusion on the clients, and how it fulfils the logging requirements of Wireless Trondheim, with the caveat of uncertainty of how these numbers will be gathered.

3.1.3 Passive Techniques for Detecting Session Hijacking Attacks in IEEE 802.11 Wireless Networks

The goal of Gill, Smith, Looi, and Clark's work is simple; to investigate the current state of Wireless Local Area Network (WLAN) security with a focus on session hijacking attacks, and suggest how it may be improved.

To begin, this paper discusses the weaknesses of WLAN security, discussing the much covered MAC frame weaknesses and session hijacking in general. However, they also cover an issue with the Robust Secure Network associated (RSN) state that was added into the 802.11i state machine - when in this state, it is capable of connecting to IEEE 802.1X compliant port-based access control, specifically the state machine of said control. However, the coupling between the two state machines is apparently quite loose, resulting in potential for exploitation in regards to session hijacking.

This leads into a brief coverage of current work done in regards to Wireless Intrusion Detection Systems (WIDS), lightly touching on the methods, eventually focusing on two similar products; Snort-Wireless and WIDZ. The methods used by these two are covered, again in brief, the two capable of rogue access detection via MAC whitelists, rogue AP/wireless Station (STA) detection, MAC sequence tracking, deauthentication/disassociation flood attacks, and of course, session hijacking detection. In turn, a discussion of what room there is for improvement follows; issues are quickly identified with the majority of the above methods, due to the various methods available to fake MAC addresses, frame numbers, or pretend to be APs or STAs. Even the physical methods of rogue detection have issues, with signal strength based detection being singled out, as directional antennas may be used to essentially spoof the apparent location of the rogue.

Thus, with the limitations of the existing methods having been identified, the paper is able to then move onto describing desirable characteristics that WIDS should exhibit in order to mitigate or eliminate said limitations; namely, the unspoofable elements of the MAC protocol and physical network layer, which are also computationally inexpensive to improve response times. Furthermore, the WIDS should operate in real-time, passively, and without modification to any of the existing hardware or software in use, along with no network performance issues. And finally, of course, there needs to be as few false positives/negatives as possible.

With these goals in mind, two passive monitoring systems are suggested; Monitoring Received Signal Strength (RSS), and Monitoring Round Trip Times of RTS-CTS Handshake, both systems receiving a general overview, some brief discussion, and then implementation in various experiments; as these methods have been discussed both

in Chapter 2 and again in Chapter 3.1.2, we shall focus mainly on how the experiments were carried out.

3.2 Research Design

The experimental goals of this paper are threefold; to show that the Xbox 360 can be a victim of a session spoofing attack, to discover what information is thus available to the attacker, and finally to investigate both attacker and victim after the fact, to discover what proof remains of the attack. Consequently a five phase experimental approach was designed for the research project (see Figure 3.1).

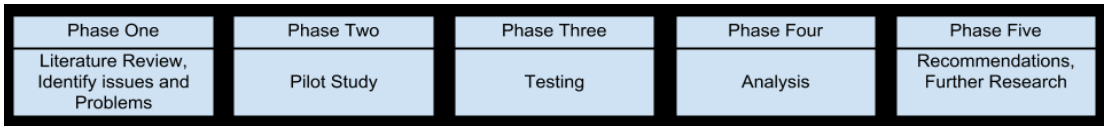

Figure 3.1: Research Phases

3.2.1 Research Questions and Hypothesis

Following both the literature reviews contained within Chapter 2 and the more in-depth analysis of the papers presented in Chapter 3.1, the following research question, sub-questions, and hypotheses were developed:

Does the Xbox 360 retain forensic evidence in the event of a session spoofing attack?

SQ1: Is the Xbox 360 vulnerable to session spoofing attacks at all?
SQ2: What information is made vulnerable via a Xbox 360 session spoof attack?
SQ3: Is evidence of an attack available on either the attacker's machine, the victim's, both, or neither?

H1: Due to the similar nature of the Xbox 360 to modern PCs, it will retain the vulnerability to session spoofing.
H2: The Xbox 360 retains a large quantity of information on the user, including a large number of social media login information - this could be retrieved.

H3: Both attacker and victim will retain evidence of an attack in some form, due to the inbuilt communications system of the Xbox 360 and Xbox Live.

3.3 Data Collection

Due to the nature of a session spoofing attack, simply attempting it requires heavy data collection; namely, in the form of the packets being sent between our target victim and their wireless router. But in order to do that, the right tools and testing set up must be prepared beforehand.

To begin with, a monitoring PC is set up running Ubuntu Linux, a fairly common free operating system. This system will be responsible for the actual packet capture as the 'victim' Xbox 360 goes about their regular use of the Xbox Live service; in order to facilitate this, the infiltration software suite Aircrack-ng (Aircrack-ng, 2013) is installed onto the monitoring PC. Airmon-ng comes with eighteen related programs, all involving the infiltration of wireless networks. Of these, four will be used to infiltrate the wireless traffic and decrypt it into a format usable both by a potential attacker and in order to gather data on what traffic is coming from the 'victim' Xbox 360. These four programs, and a brief summary of their use in the attack, are: Airmon, responsible for setting up monitoring of a target wireless access point; Airodump, which captures the packets and places them into several common packet formats, most importantly the common .cap format; Aircrack, which handles the actual cracking of a wireless network to provide its access key; and finally Airdecap, which handles the decrypting of the captured .cap format files to make them actually usable and readable. For the purposes of this test, the Aircrack program was provided with a word-list to be used in attempting to discover the access key; in this case, the word-list consisted solely of the access key, in order to speed up the process which may have otherwise taken many hours or even days.

Following the process for capturing packets from and cracking a wireless WPA2 network, as instructed by Aircrack's guide (Aircrack-ng Wiki, 2013), followed by running Airdecap on the resulting .cap file, the resulting decrypted packet file is still extremely hard to read using a common text editor. It is at this point that another piece of software, Wireshark, needs to be used; as previously mentioned in Chapter 2, this is a common piece of software that allows for the viewing and exploration of packet files.

Using Wireshark, these decrypted packets will be inspected for packets of use, in an attempt to discern how the Xbox 360 transmits data between itself and the Xbox Live servers and how these packets are identified by the Xbox Live service as genuine, particularly during the initial login of the ‘victim’ to the service.. At this point, Wireshark can be used to forge a fake packet to attempt to trick the Xbox Live servers into accepting a login from the ‘attacking’ Xbox 360 as coming from the ‘victim’. A simple diagram of this setup can be seen in Figure 3.2.

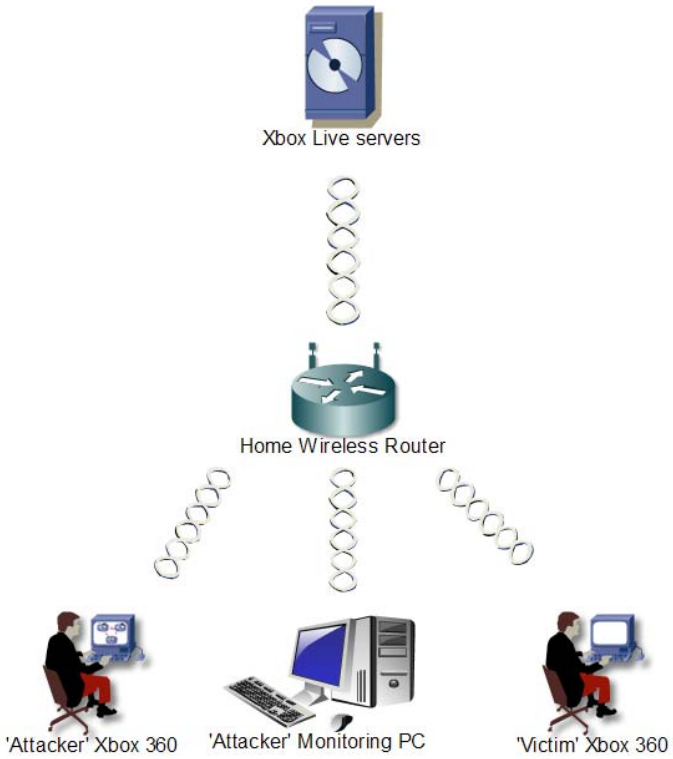

Figure 3.2: Initial Test Network Setup

In order to generate meaningful data, the ‘victim’ Xbox 360 will be monitored as it is used for various purposes on the Xbox Live service, from playing games online, performing media and game downloads, to utilising the app and web-browsing capabilities of the Xbox Live service.

3.4 Data Processing

This type of session spoofing attack provides two main avenues of data collection; the initial gathering period in order to make the attack possible, and the flow of packets between the ‘attacker’ Xbox 360 and the Xbox Live service following the deployment of

the forged login packets. In order to analyse both sets of data, Wireshark will once again be used.

The type of traffic that has been gathered will be of particular concern. While the packets will be decrypted from their WPA2 encoded state, there are still several layers of encryption that is applied to the packets during their transit between the client machines and the routers. Most still feature some sort of identifying description as part of their format, however, and may feature unencrypted data of interest encapsulating more well-protected data; the best example and prime candidate for exploration being any HTTP traffic, as that features no encrypting of data of any sort.

With this in mind, using Wireshark, there will be three stages of data processing; firstly, a casual read through the decrypted packet file to see if any identifying features or words of interest can be identified. Following this, a keyword search will be deployed, using Wireshark to attempt to filter packets containing information related to the act of logging into the Xbox Live service. Typically, these keywords are some variation on 'login', 'pass', 'user', or 'session'; even more common words such as 'hello' would be of interest, as they are used as part of the typical process of two systems identifying each other over a network. Finally, there will be an attempt to identify any patterns in the packets transmitted; focus will again orient around the process of a user logging into the Xbox Live service. All of this will be of use when it comes time to identify the weaknesses of the Xbox Live service, and what data it may leave open to a potential monitoring attacker.

3.5 Data Analysis

Once the attack has been completed, and the data from it has been collected, attention will then turn to analysing the collected data from a forensic standpoint. To begin with, both Xbox 360s will have their hard drives imaged using a forensic imager; in this case, FTK Imager. The resulting images will then be able to be placed the Encase forensic analysis software to be further inspected and to allow them to be searched using common forensic search tools and methods.

Of particular interest will be transferal of data from the 'victim' Xbox 360 to the 'attacker' system. Using Encase, information that should be unique to the 'victim' Xbox 360, such as usernames, passwords, browsing or download histories, and payment

information will be searched for and identified. Particular attention will also be paid to where information has been transferred yet was never accessed by the 'victim' during the monitoring period, such as message histories or friend lists, to see how the Xbox Live service handles other Xbox 360 machines logging into a singular account; what is stored and automatically retrieved from the 'cloud', as it were.

Finally, this data will be formatted and presented in the form of tables, detailing where the information was gathered from, what it contained or made vulnerable, and how it could then be used by an attacker for further miscellaneous activities.

3.5 Limitations

There are quite a few limitations to this experiment; first and foremost being the Xbox 360 system itself. While the Xbox 360 shares much in common with PC architecture in both system design and software design, it does make use of many unique protocols and methods of handling user interaction. What may work upon the Xbox 360 may not necessarily be immediately applicable to a PC system; furthermore, as it is possible for further renditions of the hardware to be released (as has been the case with the Xbox 360 S) or for the firmware to be patched remotely via the Xbox Live service, any exploits that are used in the process of this testing may be rendered unusable in the future.

Further on the subject of transferability, the results gathered will quite likely not be applicable to the other consoles currently available on the market. As previously discussed, other consoles on the market use quite different forms of architecture for both their software and hardware. It is quite likely that the methods used in this testing would have to be altered, perhaps quite heavily, in order to make it compatible with any other console system.

3.6 Conclusion

In this chapter, three pieces of literature were chosen from those read as a part of the preparation for the Chapter 2 literature study; these were explored and discussed, with a focus given to the methods of how their testing was prepared and handled. As a result of this, a set of research questions and hypothesis were generated, followed by a plan for a session spoofing attack on the Xbox Live using a 'victim' and 'attacker' Xbox 360. Finally, methods for dealing with the data thus gathered in order to make it actually

usable and applicable were described, as well as the potential limitations of the described experiment. In the following Chapter 4, the results of the described session spoof attack will be given, along with any variations that had to be made to make the attack successful.

Chapter 4

Research Findings

4.0 Introduction

In Chapter 3 the research methodology was decided and the data requirements specified. Chapter 4 will report the findings of the practical research.

To begin with in section 4.1, any variations to the initial research plan will be explained and justified, producing a second 'revised' condition that was used to gather data. Following any such revisions, section 4.2 will report the actual results thus gathered, in the form of information from the initial attack, and the final results produced by the revised experiment. Finally, section 4.3 will analyse these findings and explore how session spoofing may be applied to the Xbox 360, alongside any other relevant outcomes that can be gleaned from the initial or revised experiments.

4.1 Variation to Research Plan

During the investigation into the Xbox 360 and the Live service, several changes had to be made to the methodology specified in section 3.2.3. The changes to be discussed in this section were the result of the several factors, namely the degree of security implemented by Microsoft themselves, their partners/app developers, and the websites that are in turn available/accessible through the Xbox 360.

4.1.1 Experimental Design Changes

The research environment presented a number of challenges that had to be resolved. HTTPS, SSL, and TLS; these protocols were the primary cause for change in the experiment's design. HTTPS (Hypertext Transfer Protocol Secure) refers to the use of the standard HTTP (Hypertext Transfer Protocol) along with the SSL (Secure Sockets Layer) or TLS (Transport Layer Security) protocols, usually denoted within web browsers by the URL beginning with 'https://', followed by the site. SSL and TLS are both examples of asymmetric cryptography - a private key is established on both ends alongside a public key, the latter being exchanged during the 'handshaking' process, where the client and server exchange said public keys to set up a session. In this

particular case, analysis of the captured packets indicates that the Live service relies mostly upon the TLSv1 protocol (the initial version of the TLS protocol that was released, and in turn superseded by the TLSv1.1 and TLSv1.2 protocols).

This discovery made it impossible for the experiment design, as it stood, to be implemented. Part of the TLS protocol is the regular altering of the cipher being used to pass data back and forth between the client and server. This involves a three step process of re-exchanging the keys, the cipher specification being changed, and then a encrypted handshake process being completed - swiftly rendering any prior sessions established as utterly useless.

In turn, the TLS encryption meant that the data being exchanged could not be analysed. While the process of a user logging into Xbox Live could be roughly captured (see Appendix A: Xbox Live Login Capture), which packets to relay and in what order became a very rough guessing game. While there are methods that could be used to crack TLS, such as the CRIME (Rizzo, Duong, 2012) and/or BREACH (Prado, Harris, Gluck, 2013) attacks, their implementation had either reportedly been mostly defended against (CRIME) at the time of the research or simply took too much time for the researcher to implement (BREACH). The alternative, to somehow gain access to the Xbox 360's inbuilt private key, was reluctantly discarded after the heavy overhead that would have been added to this project was taken into account.

This encryption issue affected spoofing attempts against the Xbox Live process and its apps, which relied heavily on TLS for the exchange of non-vital data. As a result, focus was switched to the Xbox 360's built-in implementation of Internet Explorer. Under this new design, the concept of having an 'attacker' system and a 'victim' system swapping roles was discarded - instead, a simple monitoring of wireless traffic coming from the 'victim' (whose role was now relegated to the Xbox 360 S console, previously the 'attacker') would be recorded and decrypted, with efforts focusing on unsecured/non-HTTPS traffic, in an attempt to obtain either a session ID or session cookie variable to be used by the monitoring PC, as per a more traditional PC-vs.-PC spoofing attempt. As part of this, several new accounts were created for the "victim" to browse to during the attack, on Twitter, Facebook, Gmail, and TradeMe. The new experimental design diagram that resulted was as shown in figure 4.1.

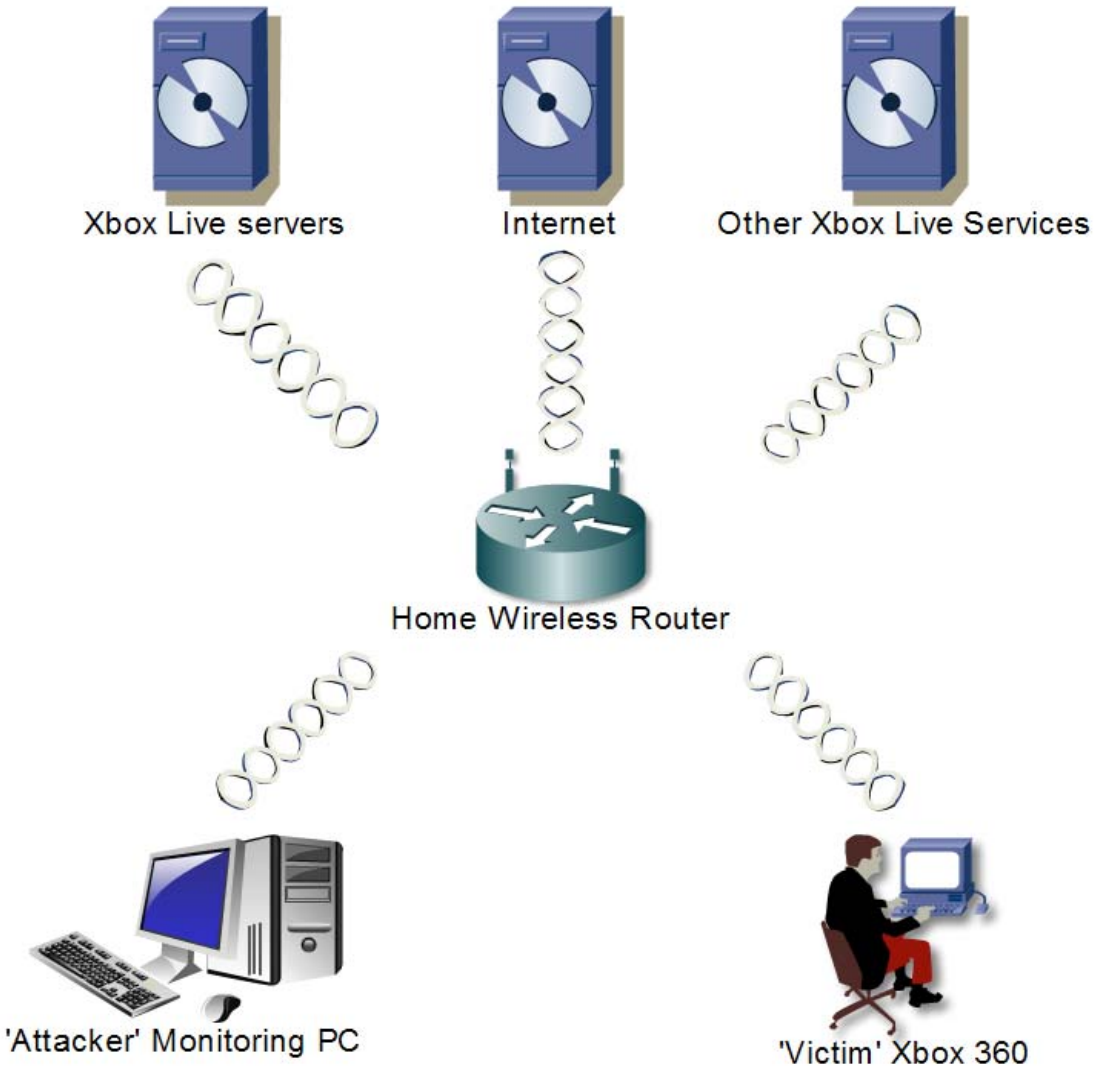

Figure 4.1: Revised Test Network Setup

In order to achieve this change in design, a piece of additional software was required; Edit This Cookie v1.2.5 (Capano, F., 2013). This Chrome specific app allows for the live viewing and editing of cookies that are currently in use by the Chrome browser; this would allow for the captured session to thus be placed into use by the attacker's browser.

4.1.2 Data Collection Alterations

Fortunately, the changes that needed to be applied to the data collection methodology were minimal. The main difficulty in this area was from that of the Aircrack-ng suite itself - specifically, what was either an unmentioned feature or simply a lingering bug in either the airmon-ng or airdecap-ng programs.

While airmon-ng would, during collection, display as having captured the WPA2 handshake, what was not mentioned was that this handshake was client and server specific; as a result, a handshake captured between the access point and the victim, for example, would be useless for decrypting the encrypted packets transferred between the attacker and the access point. This would probably not be important for an actual attacker, but it did lead to some confusion when initially reading the results.

A change was also made to what portions of the captured packets were being inspected. Where previously all the traffic generated by the Xbox 360 was inspected (see Appendix B: Successful Session Spoof Packet Dump), under the new design only those that were of unsecured HTTP data and featured cookie information were of interest. This filtering process was made easy by the use of Wireshark; specifically, its inbuilt packet filter allows for exactly this sort of situation; in this case, through the key phrase 'http.cookie'.

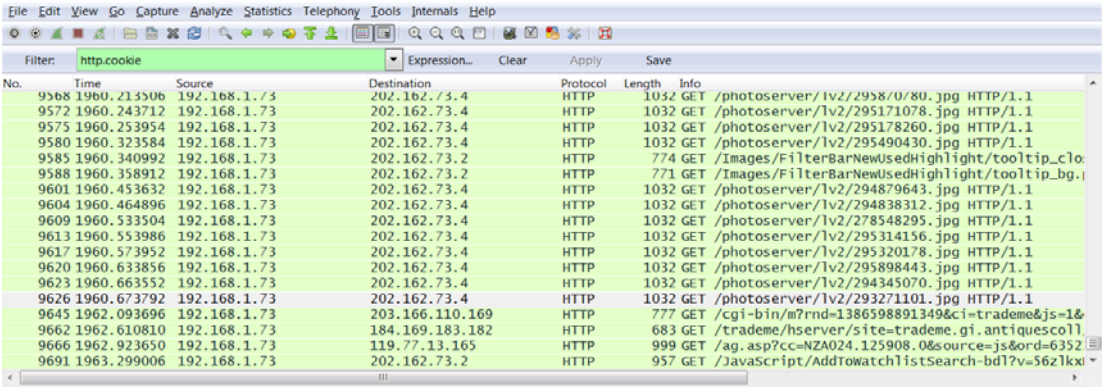

Figure 4.2: Filtered Cookie Traffic

This automatically allows for http packets with cookie information contained within them to be identified, following which a manual search for words of interest ('datr', 'session', or some form of ID) can be done, or further refinement of the filtered list can be done by adding the term 'contains' followed by the words of interest.

4.2 Results

The results gathered took the form of capture packet files (file type .cap) that were compatible with Wireshark. The first sequence of these, dump1.cap through dump17.cap, were captured under the initial experimental design. The last three were captured under the revised experimental setup. In this section, the observed results of the

initial attempt to spoof the Xbox Live service will be detailed, followed by the successful attack done using more traditional spoofing methods.

4.2.1 Xbox Live Initial Attack

Under the initial attack, the Xbox 360 'victim' console was repeatedly logged in and out of the Xbox Live service, through powering the system on and off, manually logging in and out of the service, and forcing the service to disconnect via deauthentication attacks. All three systems of attack produced similar packet numbers, typically in the two to three hundred range - the majority of these involving TLS encrypted communication between the Xbox 360 and the Xbox Live servers. Of note, however, are the times where the Xbox 360 initiates HTTP GETs for static images, typically for ads or to display box art on the screen.

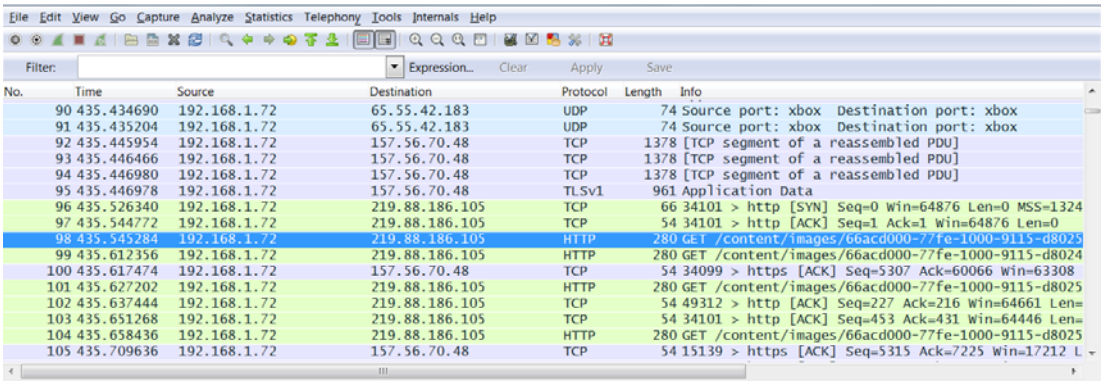

Figure 4.3: Traffic During Xbox Live Browsing

These HTTP GETs are, unlike the more secure HTTPS communication done with TLS, left utterly unencrypted - all information is made plainly visible to both the receiving and sending side. Typically, this sort of unencrypted traffic is used for obtaining similarly low-security items of importance; in this case, image files. While revealing the full address of where the images are coming from, they do not offer up anything in the way of allowing for session spoof attacks, especially as they were transmitted without any cookie information.

4.2.2 Xbox Live Revised Attack

Under the revised attack, the Xbox Live system was tested at three levels - firstly under the case of using the general features of Xbox Live, such as game downloads or account updates, secondly with the available Xbox Live apps that are made available through the

marketplace, and finally through the Internet Explorer app that is made freely available to Xbox Live users, albeit with certain caveats.

4.2.2.1 Xbox Live

Put simply, there were found to be no vectors (other than the aforementioned HTTP GETs observed during initial login/marketplace browsing for static images) that would allow for a potential spoofing attack. All traffic of interest was securely encrypted under TLSv1 (with regular and frequent alteration of the TLS cipher), or comprised of UDP/TCP packets, the use of which was non-existent due to a lack of knowledge of what the packets were actually being used for or what the console was reading from them, alongside the probability that they were, in turn, encrypted.

4.2.2.2 Xbox Live Apps

The Xbox Live, beyond its own services, allows for licensed developers to produce 'apps', or applications, that are essentially self-contained programs for use on the Xbox 360 not directly relating to playing videogames. While technically the videogames offered only via the Xbox Live service could be considered apps, they are considered to be a category of their own - this is simply a matter of categorization. An excellent example would be the Youtube app, which is custom-built to allow for the browsing and playing of videos housed on the Youtube site, without going through a browser. All the apps experimented with, however, did require that the user have a Xbox Live Gold account; as discussed in Chapter 2, this is a form of more 'advanced' membership that requires a paid, regular subscription.

What could be gleaned from the Xbox Live apps available almost directly mirrors that of the Xbox Live results - that is, nothing of immediate use - it is apparent through the network traffic that the Xbox Live apps do not appear to use the same ports or protocols for TCP traffic that Xbox Live itself does. While packets relating to the Xbox Live service itself typically use ports in the 40000/50000 region that aren't associated with typical protocols, the Xbox Live apps that were captured in action were observed to be using a wide variety of standard TCP/UDP protocols.

4.2.2.3 Internet Explorer App

The Internet Explorer app for the Xbox 360 functions as a simple web browser, albeit one with several display and navigation issues; attempts to use one of sites offered to the user to visit from the app's main menu, Twitter, fell completely flat, with attempts to sign in producing no navigation.

The websites on offer from said main menu were those typically associated with New Zealand browsing habits; famous social sites Facebook and Twitter being first offered, followed by more New Zealand specific sites such as Crunchyroll and, importantly, TradeMe. While the social sites featured HTTPS secured traffic, during capture it was observed that the TradeMe site, under the Internet Explorer app at least, was transmitting everything in unencrypted HTTP traffic - including, crucially, cookie information. A particular packet was focused on soon after logging into TradeMe, number 9626 of the decrypted packets available; this particular packet was chosen for no specific reason, other than it appeared as part of the filtered Wireshark list of packets containing http.cookie information.

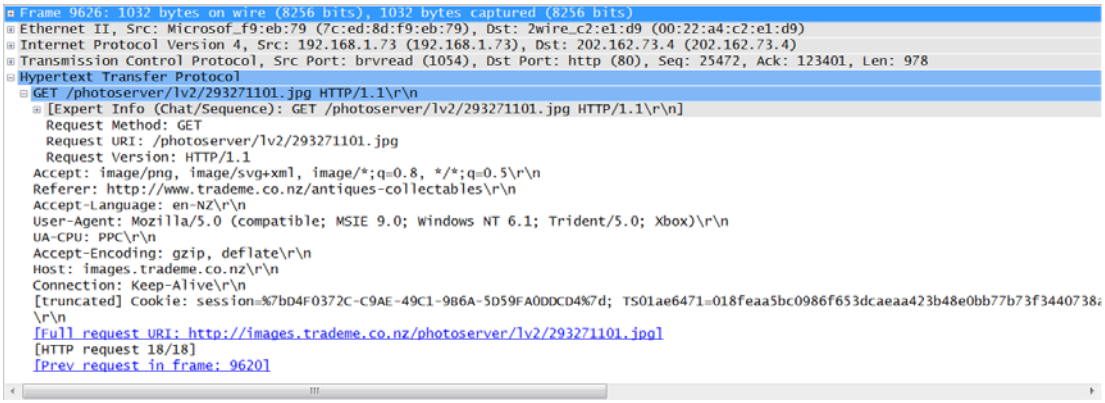

Figure 4.4: HTTP Contents of Packet 9626

At this point, the attacking PC was navigated to trademe.co.nz. As part of this navigation, TradeMe automatically generated a session ID for the attacking PC, which was then stored automatically within a cookie file. By using Edit This Cookie, this session identifier, assigned to the key name "session" was able to be located and edited to match the session identifier given by packet 9626 - after saving the edited cookie and simply refreshing the browser, this tricked the site into believing that the attacking PC was using the same login as the Xbox 360 had via the Internet Explorer app.

4.3 Research Analysis

In this section of the chapter, an analysis of the research findings will be held, focusing on the three sections of interest that were covered by the revised experiment, namely the Xbox Live service itself, the Xbox Apps in general, and then specifically the Internet Explorer app made available to the Xbox Live app which made the experiment possible and successful in the end. These sections will mainly contain observations of the each area of interest as they were discovered during the experiments, accompanied by evidence relating to these observations.

4.3.1 Xbox Live

Overall, the Xbox Live service itself withstood the attacks quite well. The use of TLSv1 to encrypt its traffic and the use of ports that were unique to the Xbox Live service ensure that the communication between Live and its servers are quite secure, at the least when it comes to session spoofing attacks. However, two items of forensic/security related interest were revealed during the investigation:

Firstly, the use of unsecured HTTP GETs to obtain static image files for display on the Xbox 360. While not exposing any information in and of themselves, they do show that the Xbox 360 has a potential for these packets to be intercepted and other images returned for display or other, more malicious means - for example, steganographically encoded images or virally compromised files, such as the Perrun virus (Knowles, 2002). What this means, exactly, depends on whether the former or latter situation occurs, and the Xbox 360's method of caching - in the former case, at the very least, it would mean that disturbing images could be transferred for display through the Xbox 360 GUI, albeit in a very roundabout way. As the images appear to be cached for later display, however, this means that they could potentially be retrieved, allowing for the Xbox 360 to act as a go-between medium in transferring such images. In the latter case, that of a viral attack, the Xbox 360 would either already need to have been infected in some way with the 'base' virus in order for the malicious code within it to run, or the image would have to make use of what is known as buffer overflow errors to allow the arbitrary code to be run (Security TechCenter, 2004).

Also of note was how the Xbox Live service reacted to deauthentication attacks done during the process of collecting the four-way wireless handshake in the initial

experiment upon the Xbox 360 - namely how simple the process was, if the attacker is able to monitor the wireless connection and execute such an attack. While not useful in terms of spoofing, this sort of attack would have ramifications in the realm of e-sports; that is, in order to force the opposing team offline at a critical moment in order to secure one's own victory. This could be used to essentially rig competitions that rely on a wireless medium for connecting multiple users, the rough equivalent of throwing dirt in a competitor's eyes.

4.3.2 Xbox Apps

While none of the apps used were observed to allow insecure, non-HTTPS traffic to be transmitted (barring the Internet Explorer app, to be discussed below), this does point to a lack of uniformity in the way Xbox Live app data packets are transmitted - not a smoking gun, but it does leave open the possibility of individual developers making poor security choices in regards to the 360's applications (app).

4.3.3 Internet Explorer App

At the very least, the Internet Explorer app shows that it is just as vulnerable as any other browser to the more simplistic session spoofing attacks, although the blame for this can just as easily be laid on the individual site designers/coders. The app, on repeated use, also shows what could easily be considered an annoyance or advantage to the user; it does not appear to cache any information back to the Xbox 360 itself, as during testing site logins and cookies would be lost once the user left the app, forcing them to log back in to any service.

Beyond the spoofing results, however, the Internet Explorer app revealed one further item of interest; that of the meta-data containing the user-agent string. The user-agent metadata is used by the receiving server to help figure out if any special tricks need to be used to render a particular web page. While largely redundant due to the standardization of web code and websites over the years, there are some unique problems that still crop up for different browsers based on how they were coded. In this particular case, the user-agent of the Internet Explorer app reads as follows:

User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0; Xbox)

Each portion of the string means something about the browser sending out the request; in this case, that the browser is compatible with Mozilla 5.0 HTML renderers, identifies as Microsoft Internet Explorer 9.0, is running on a Windows NT 6.1 based operating system, uses the Trident 5.0 engine for laying out HTML components on screen, and finally adds the tag ‘Xbox’, which is not a normally recognised portion of a user-agent string, but in this case can be presumed to just identify that the communication came from a member of the Xbox console family.

The key portion of interest here is how the browser identifies as Internet Explorer 9.0 - at the point of writing, the latest stable release of Internet Explorer is v11.0.9600.16438, two entire major releases later. This means that the Internet Explorer app has missed out on years of patches and security additions that the PC version of Internet Explorer has received; essentially, this gives an attacker a ready made list of exploits that could be used against the Internet Explorer app, simply by referring to all the security bulletins and patch notes released by Microsoft over the intervening editions of Internet Explorer.

4.3 Conclusion

Overall, the Xbox Live system provides extremely good protection for the information that is being transferred back and forth between itself and the Xbox Live servers. While this means that the Xbox Live communications are quite secure, it does not necessarily mean that the entire system is as tightly closed as it could be, with several lingering vectors of potential misuse remaining. What these vectors could potentially be used for and mean when compared with previous research will be discussed in the following chapter 5.

Chapter 5

Discussion

5.0 Introduction

Chapter 4 presented the results of the initial and revised experiments as they stood alone, this chapter is to take what has been discovered and to compare/contrast it with the theoretical work done in Chapter 2 and Chapter 3; specifically, answering the research questions derived in Chapter 3, and seeing how the experiment played out in comparison to the previous work done by other researchers in Chapters 2 and 3.

To this end, this chapter will begin with a review of the evidence to answer the research questions in section 5.1, beginning with the primary question, moving through the secondary questions, and finishing with the hypothesis. Following this, section 5.2 will compare the results found in Chapter 4 to the literature review done in Chapter 2; in turn, section 5.3 will compare the results to the three papers focused on in the Chapter 3 review. The reason for this continued scrutiny is to glean how the results compare to other, similar experiments done, as well as to identify any areas where the Xbox Live service may be improved, or the experiment itself may in turn be improved. These improvements will be expounded upon in Section 5.4, containing recommendations for future research in this area. Note that the scope of the research was reduced as the testing proceeded to fit the time requirement and the forensic investigation was not attempted. The findings are interesting and sufficient to answer the research question. The other investigation matters that may have been attempted if time permitted now better fit suggestions for further research.

5.1 Research Questions Review

In this section, the primary question of the research will be reviewed, and discussed in light of the findings produced by the experiments undertaken. References to the results found in chapter 4 will be provided to provide evidence for the answers, alongside any observations that were also recorded in the research diary.

5.1.1 Primary Question Answers

To begin a review of the research questions, it must be stated that the primary question "*Does the Xbox 360 retain forensic evidence in the event of a session spoofing attack?"* remains in an unconfirmed state. This is primarily due to the difficulty of getting a working version of the initial attack done, combined with an equal inability to properly forensically dissect the Xbox 360 following the attack. However, the secondary questions provide some more robust and substantial answers.

5.1.2 Secondary Question Answers

The final experiment shows that the answer to the first of the secondary questions, "*Is the Xbox 360 vulnerable to session spoofing attacks at all?*", is an unequivocal yes as evidenced by packet 9626, as shown in Chapter 4.2.2.3 - if only due to the vulnerabilities that are inherent to all unsecured HTTP traffic and communication. The Internet Explorer app provided via the Xbox 360 provides the clearest route of obtaining data that could be used to execute a session spoofing attack, albeit with the normal caveats that the attacker is able to obtain access to the user's traffic at a time when they are using sites that are vulnerable to a session spoofing attack.

The second of the secondary questions, "*What information is made vulnerable via a Xbox 360 session spoof attack?*", is again limited to the information that can be gleaned from the limitations of an everyday session spoofing attack; namely, that of a user's session and in turn whatever information can then be obtained from the website that is being spoofed. However, further examination of the packet data obtained during the research and shown in Chapter 4.1.2's filtered results and Chapter 4.2.2.3's packet 9626 shows that, at the very least, the user's browsing or Xbox Live habits can be monitored via the use of unencrypted the HTTP GETs and the ports being used by the various apps available over the service. While not exactly the most vulnerable of data, this can be used to develop a profile of the user's habits and potentially find individual loopholes within the protocols used by each individual app. Finally of note is the user-agent reveal that the Internet Explorer app runs on an apparently unpatched and potentially vulnerable version of the Internet Explorer 9.0 program; this single line offers up the potential for an array of exploits or weaknesses to be tested against the system.

Finally, the answer to "*Is evidence of an attack available on either the attacker's machine, the victim's, both, or neither?*" is uncertain, primarily due to the incomplete

forensic investigation that would have been held following the initial version of the experiment. That does not mean that there is no evidence of such attacks; some indication of the attacks being deployed can be detected through more standard session spoofing methods, namely router logs identifying the deauthentication attacks being deployed in an effort to gather four-way handshakes or other information necessary to decrypt the traffic as it flows. On a less technical level, this can appear simply due to the user being suddenly or repeatedly being forced off the Xbox Live service without other devices losing internet connectivity.

5.1.3 Reconciling Hypotheses with Observed Results

Two hypotheses were made as a product of the primary and secondary questions; the first stated "*Due to the similar nature of the Xbox 360 to modern PCs, it will retain the vulnerability to session spoofing*". This hypothesis turned out to be mostly correct, in that the Xbox 360 relies on standard internet communication protocols to facilitate its various services and apps doing what they are needed to. The most blatant example of this would, again, be the Internet Explorer app, which self-identifies in the user-agent string as being Internet Explorer 9.0, and allowed for the success of the revamped experiment via standard session spoofing methods. It should be noted, however, that while the Xbox 360 relies on these standards and protocols for its communication, it does make use of some of its own to obfuscate matters; namely the strict adherence to proper TLS and HTTPS communications made by its own services alongside non-standard ports in and out of the system. Its app developers also appear to follow similar guidelines when developing for the Xbox Live service and communications with their own servers, considering the heavy (albeit protocol varied) use of TLS communication there. It could be speculated that the lack of enforcement of similar standards with the Internet Explorer app may simply be due to the difficulties of getting the said app to work under constant TLS encryption, considering that most websites that a casual user might visit would have no need for more a more secure connection (other than those that require or depend upon a login or some other form of authentication, of course) and could thus have difficulties with receiving TLS traffic.

The second hypothesis, that "*The Xbox 360 retains a large quantity of information on the user, including a large number of social media login information -*

this could be retrieved", actually proved to be a mixed result. The Internet Explorer app, through repeated use, demonstrated that it did not in fact retain cookie information from session to session - once the app was closed, it dumped whatever it had been using. The Xbox Live console and its associated apps, however, did demonstrate that they would retain login information for the user, assuming the user had an account; this can be demonstrated most strongly and immediately simple by the presence of the user being able to power up the console and choose which account they wish to sign into the Xbox Live service. This information, however, is clearly and closely guarded by the system's use of encryption and proprietary security, rendering it difficult if not impossible for an attacker to obtain it from outside the system.

5.2 Comparison with Literature Review

It is not enough, of course, to simply provide a synopsis of how the experiment played out in comparison to the research questions and the associated hypotheses; how this information compares to the previous work done in the key areas identified in Chapter 2's literature review is equally important in order to expand on the work done in this field. In this section, the major area of interest directly related to this research as discussed in Chapter 2 - Session Spoofing - will be compared and discussed with the findings, along with references back to the previously done studies.

5.2.1 Session Spoofing

The session spoof that was finally undertaken as part of the revised experiment deviates from that describes by Eric Cole "Hackers Beware: The Ultimate Guide to Network Security" (2002), namely in the use of the cookie/session variable to trick the receiving server, versus the more technical method of sequence number prediction and having to take the opposing user entirely offline. It is, of course, also much less broadly applicable than the methods described by Cole; while cookie based session hijacking can only be done due to HTTP traffic transmitting what is essentially human readable values, the TCP/IP hijacking described by Cole can be utilised against many different applications that do not rely on HTTP traffic.

Of note when compared to the 'traditional' attack, however, is Dave Dittrich's (1999) observations on how a successful session spoof could be used to introduce

backdoors into a target system, and the possibility of abusing various overflow or underflow issues in how the Xbox 360 processes the images it receives via its HTTP GET requests. The same point can easily be extended to the Internet Explorer app and its apparent lack of version steps and potential for exploitation; while not directly related to session spoofing, it would allow for an attacker to potentially introduce more direct means of accessing the console.

In regards to detecting session spoofing, as discussed by Himanshu Arora (2012), and Gill, Smith, Looi, and Clark (2005), the cookie-based method that was finally used by the revised experiment essentially dodges the potential to identify this kind of session spoofing by the traffic's sequence numbers, as it does not matter to the receiving server what packets have been received when, only that the session identifier matches what it is expecting. This is only due to the trusting nature of the TradeMe servers in question, however; if every session was being fully monitored, the differing sources of the packets would be quickly identifiable, although this would put a lot more pressure on the servers in question as they track and compare the sources, sequence numbers, and travel times of the packets being received.

Finally, on the subject of defending against session spoofing, it should be pointed out that consoles provide a unique opportunity for deploying monitoring and encryption based hardware; as every console is manufactured identically, barring the rare hardware revision across a console's lifetime, applying hardware defenses could be applied straight to the console in the factory. The two defenses that would best benefit from this would be Doeppner, Klein, and Koyfman's (2000) router-stamping proposal, and Chakravarty, Portokalidis, Polychronakis, and Keromytis (2011)'s faux-login approach. The former, router-stamping, could be ideal as the console's manufacturers would control both ends of communication; client and server. This would allow for end-to-end packet and routing tracing. Meanwhile, the faux-login approach could periodically attempt to send the fake login signals, possibly in reaction to suspicious activity in a console's traffic, seeing if anyone then takes the bait offered up.

5.3 Comparison with Previous Studies

In this section, a comparison with the three papers previously presented in Chapter 3 as part of the review of related publications are conducted against the findings and observations made throughout the initial experiment and the revised experiment.

5.3.1 Xbox security issues and forensic recovery methodology (utilising Linux)

Due to the unfortunately cut portion of the experiment relating to the actual forensic investigation into a compromised machine, the application of Vaughan's (2004) research to the experiment done is largely theoretical. The most that can be said is that the paper would provide an excellent guide to performing a forensic investigation upon a Xbox 360 console following another rendition of the revised experiment; the highest potential application would be in the event of the initial experiment being completed, due to the possibility of incriminating forensic evidence existing in the Xbox 360. Due to the nature of the session spoof attacks done in the revised experiment, there is a lack of evidence that could potentially remain upon the victim's Xbox 360; the biggest potential indicator that an attack had been made would be any sudden, unexplained deauthentication attacks.

What this research would be most useful for, however, would be in the event of a Xbox 360's HTTP traffic being used to set off malicious code, as could be done via the previously theorised resource underflow/overflow methods. In this case, if a Xbox 360 had been seized in an investigation due to suspected activity, it could show that the console had been compromised by a third party.

5.3.2 Session hijacking in WLAN based public networks

The benefits of Bækkelund's (2009) research match nearly point to point with those mentioned during Chapter 2 in regards to identifying session spoofing attempts through Sequence Number inspection, albeit this time using MAC Frame Sequence Numbers instead. The difficulty inherent in spoofing a MAC Frame Sequence Number, albeit not that much removed from spoofing any other kind of traffic related sequence number, would add an extra layer of difficulty to any spoofing attacks attempted. Furthermore, the suggested method of requiring repeated logins by the client to the service being used, in order to invalidate previous sessions that may be being spoofed, could be quite simply automated as part of the Xbox Live service; comparisons have to be made, of course, between this suggestion and the existing example of the TLSv1 traffic encountered

during the experiment, which frequently alter their cipher to prevent previous traffic from being decoded. It could therefore be quite possible to use a similar infrastructure not just for the traffic, but the entire session of communication between the console and the Xbox Live service, essentially doubling the difficulties of spoofing anything thus encoded; not only would the traffic need to be constantly recaptured in a spoofing attack, but then the login itself would have to be recaptured.

Another possibility for detection, albeit one unique to wireless communication, would be the described Received Signal Strength (RSS) monitoring. Consoles, by their very nature, are extremely sedentary items of entertainment; the user is unlikely to move the console once it has been placed down until they feel the urge to redecorate. As a result, the RSS suddenly fluctuating would be a excellent indication of a more mobile user attempting to spoof the session; both console and server could be on the watch for such a thing, if the hardware was implemented.

5.3.3 Passive Techniques for Detecting Session Hijacking Attacks in IEEE 802.11 Wireless Networks

The primary items of interest that can be taken from Gill, Smith, Looi, and Clark's (2005) work on Wireless Intrusion Detection Systems (WIDS) are those relating to the desirable characteristics of WIDS. Taken with the observations made through the rest of this research, it would seem to be an ideal checklist for a small, automated system made inherent to the Xbox 360 or its descendant's hardware; as components are mass produced and, using the methods suggested in Gill, Smith, Looi, and Clark's paper, computationally inexpensive both on the client and server end. Any such changes could also cover the console in the event the user is using it via a wired connection; due to the paper's use of the physical elements and MAC related information produced by the console and the physical network layer, this could ensure that even the more difficult to pull off wired session spoofing attacks could be detected and potentially defended against.

5.4 Recommendations

In this section, recommendations will be made as a result of the findings of Chapters 4 and 5; these will be in two areas. The first subsection will deal with improvements that

could be made to the Xbox 360 and Xbox Live service; the second will deal with improvements that could be applied to the experiments done in this research.

5.4.1 The Xbox 360 Console and Xbox Live Service

A key suggestion that can be gleaned from the review of previous work done in combination with this research would be an implementation of some variation of intrusion detection system within the Xbox 360 itself; this would allow for a standardised method across the entire service of detecting, reporting, and defending against intrusion attempts against a Xbox 360.

Furthermore, in the light of the possibility of abusing the Xbox 360's reliance upon HTTP GETs for its static media and the Internet Explorer app's unencrypted communications, several suggestions can be made. The first is to ensure that any and all communication made between the Xbox 360 and the Live service is properly encrypted, even in the case of simple elements such as static media files; while it is possible to patch a console to take care of any attacks that could be made through this medium, it would be more prudent to make it as difficult for the attacks to occur in the first place. Secondly, in regards to the Internet Explorer app, it would be extremely prudent to make it regularly update to the same state as the commercially available Internet Explorer program for PC, considering the apparent similarity between the two browsers and their underlying hardware. At the very least, an attempt to have the user use HTTPS traffic whenever possible would help immensely in preventing cookie based session spoofing attacks.

5.4.2 The Experiment

The first recommendation for any future rendition of this experiment would be the application of forensic tools following a successful attack; this recommendation is entirely due to the lack of such an application in this paper itself, as more observations in this field would be of great benefit to current and future forensic investigators.

Beyond this recommendation, a future experiment following the initial design would be of great interest if combined with either some variant of the CRIME or BREACH attacks, as mentioned in Chapter 4.1.1. These attacks, which use the transmission and encryption time of packets to attempt to deduce the actual key being used to encrypt the TLS traffic, which would in turn allow for the original attack to be

successfully completed. This attack, if successful, would also in turn allow for the Xbox 360 apps and their logins to be similarly spoofed; a forensic investigator would then be able to see, after extracting an image of the Xbox 360 HDD, what would remain upon an Xbox 360 used in conjunction with such an attack.

5.5 Conclusion

Chapter 5 began with a comparison between the research questions posited in Chapter 3.2.1 and brief discussions on how the hypothesis formed in Chapter 3 compared to the actual results that came from Chapter 4. This chapter, 5.1, found that, despite the lack of a full forensic investigation at the conclusion of the revised experiment, the Xbox 360 is vulnerable to session spoofing attacks and does make information visible to an attacker/observer; this is with the caveat, of course, that this is only due to the inherent weaknesses of unencrypted HTTP traffic with a similarly lax server, both sides these weaknesses in turn only due to the Internet Explorer app offered for the Xbox Live service. Following this, chapter 5.2 featured a discussion of the findings that came from Chapter 4, comparing them to the previous work done by the literature covered in Chapters 2 and 3. Overwhelmingly, the resulting impression was of a need for some form of Intrusion Detection System inherent to the system itself, which could quite easily be deployed thanks to the mass-produced nature of modern video game consoles.

Finally, chapter 5.3 produced some recommendations for the Xbox Live service to better strengthen it (or any descendants of it) against future session spoofing attacks, alongside a few brief suggestions on how to improve the experiment done in this paper.

The following Chapter 6 will contain a summary of findings produced by this research and the recommendations for further research. As part of this, the limitations of the research done will be identified, followed by recommendations of related areas of study that could be followed by future researchers.

Chapter 6

Conclusion

6.0 Introduction

In Chapter 5, a comparison of the findings presented in Chapter 4 was done with the research questions and their associated hypothesis. Comparison was made with work previously published in the area of forensic investigation and session spoofing, particularly as it applied to video game consoles. The reduction in scope for this research was also noted so that forensic investigation concerns are now deferred to suggestions for further research.

In this chapter, a conclusion to the above research will be offered, in three sections. In section 6.1, a summary of the findings is made. Following this, the limitations of the research will be explored in section 6.2. Finally, possible areas of future research will be offered in section 6.3, before section 6.4 brings this paper to a final close.

6.1 Summary of Findings

An initial test was prepared in order to answer the research question "*Does the Xbox 360 retain forensic evidence in the event of a session spoofing attack?*", utilising two Xbox 360 video game consoles in the role of 'attacker' and 'victim', with a PC dedicated to monitoring the traffic being sent by both through a wireless router, to be utilised as part of a session spoofing attack. This initial attack would take the form of capturing packet data, isolating a login attempt from the 'victim' Xbox 360, and attempting to manufacture traffic to make it appear as though the 'attacker' Xbox 360 had logged in as the user of the 'victim' Xbox 360.

This initial test soon proved to be fruitless, due to the heavy use of TLSv1 encoded and secure traffic being passed between the 'victim' and the Xbox Live service. After repeated attempts to find a method around this obstacle, the test was redesigned to focus instead on attempting a more traditional, HTTP/cookie-based session spoof attack against the Xbox Live service.

Traffic was gathered as the Xbox 360 made use of the Live service, the apps available via this service, and in particular the web browsing app Internet Explorer. What was discovered was that, although encryption was used for the traffic originating from the Xbox Live services and the majority of its apps, the Internet Explorer app functioned as a stripped down and unpatched form of the Internet Explorer program available for PC, specifically the 9.0 version of it. This app sent and received some HTTP traffic in an unencrypted form; the session spoof attack was successful after a cookie containing a session variable was found within traffic following the user logging into the TradeMe website via the Internet Explorer app. When this session variable was transferred to the attacking PC and copied into the PC's own version of this session variable, the TradeMe site was successfully tricked into considering the PC as using the same session as the Xbox 360 user.

Many observations and findings resulted from both the initial and revised versions of this test. Of note were the use of unencrypted HTTP GETs by the Xbox Live service in order to obtain static images and other media, which could potentially be used in various underflow or overflow methods in order to force the Xbox 360 to run unauthorised code. The actual communications done by the Xbox Live service with the console, and the associated apps, otherwise made heavy use of TLS encrypted traffic, although the apps were observed to be using quite a variety of ports and protocols, indicating a possible lack of uniformity amongst the various apps. And of course, it was discovered that the Internet Explorer app identified, in its user-agent string, as a variation on the Internet Explorer 9.0 browser, indicating that exploits or bugs that had long been patched or fixed in the more recent editions and version of Internet Explorer could possibly be applied against the Internet Explorer app. Finally, the success of the session spoof attack against the Internet Explorer app showed that, at the very least, the Xbox 360 could be used as a vector for these kinds of attacks.

One unfortunately change to the test plan was the lack of any forensic investigation into what evidence of the spoofing might remain on either the Xbox 360 or the attacker's PC; this left several of the original research questions in an unanswered state.

6.2 Limitations of Research

The limitations of the research were discussed briefly in Chapter 3.5, and then again as part of the recommendations that resulted from the discussion done in Chapter 5.4.2. From these discussions, the overwhelming area of weakness in this test from a forensic perspective is the lack of forensic investigation able to be done; this would have to be of prime interest in any future extensions of this work. This is true on both the console and the router side; both need to be considered to be sources of forensic information, as well as the monitoring PC, of course. The security problems encountered in this research will have to be considered when planning further investigations and the possibility of getting more evidence evaluated.

Another limitation comes from the nature of console systems themselves. While the Xbox 360 does feature an architecture that closely resembles modern PCs from both a hardware and software design point, this is very untrue of its current competitors, the Wii and the PS3; the PS3, in particular, has had many developers complaining of its unwieldy design, although early editions of the console were capable of running variations of the Linux operating system. As a result of this observation, these results should not be assumed to be universally applicable until the individual consoles and their apps/software are properly explored in the context of this research.

Finally, of course, there is the issue of the Internet Explorer app and in its unpatched state. One of the great advantages in the modern age of PC and video game design is the possibility for developers to deploy patches to fix any issues that may result; it is quite possible that at a date beyond the release of this paper and its data that the app may be updated to prevent session spoofing of this kind from being possible.

6.3 Areas of Future Research

There are myriad ways that this research could be continued, beyond the suggested limits. In particular picking up the requirement for forensic investigation of game consoles. The aforementioned forensic investigations that could be done to discover evidence of session spoofing are a beginning. Starting with the Xbox Live service, research should be done to investigate what effects the manipulation of its use of HTTP GET sources images and media result in. Beyond the also aforementioned overflow/underflow exploits, it could be used as a vector to transfer illicit or encoded material between systems in a not so obvious method; this could be of interest for those with a steganographical interest.

On the subject of the TLSv1 encrypted traffic, the obvious road of future investigation would be the deployment of some variant of the CRIME or BREACH attack to discover what security methods within the TLSv1 encryption is used by the console. It could quite possibly be found that the key used to encrypt said traffic is part of the key used to encrypt the console as a whole; this would allow for heavy exploitation of the Xbox 360, either as a simple storage medium or as a small, self-contained PC.

How this test applies to other console systems is also of interest; could it be that all consoles share these security concerns? If so, it points to a security issue within the design of consoles at large that should be addressed. At the same time, when thought of alongside the recommendations covered in Chapter 5.4.1, it could also be that the Xbox 360 and its console brethren, through their use of a standardised design, could allow for various hardware based defenses to be deployed without the user having to manually apply them.

6.4 Conclusion

Through these chapters, the Xbox 360 and console forensics have been investigated and reported on, with a focus applied to session spoofing and its possible uses through the Xbox 360 console and the Xbox Live service, using common spoofing and traffic monitoring tools. While the test did not manage to cover much in the way of forensic information, it did manage to confirm that the Xbox 360 shares vulnerabilities to session spoofing with other web browser compatible systems. Furthermore, several observations were made in ways the Xbox 360's method of handling traffic could be further abused.

Chapter 6 thus brings this work to a close, with a final summary of the findings, a brief extension on the limitations of the research done through the chapters, and speculation on where future areas of research could be done to continue this work. Console forensics is a difficult due to its heavy use of proprietary formats and methods, making it quite difficult for a forensic investigator to do their job. Consoles are able to share in and standardise various methods of protection, and are capable of featuring some very simple weaknesses that can be exploited, potentially either by those of a villainous turn or an investigative one. It is a simple matter of knowing what to do and for which intention.

References

A note on the reliance of online resources for references; due to the very recent appearance of video game consoles within the forensic domain, the vast majority of resources are found digitally online.

Adafruit (2010). "WE HAVE A WINNER – Open Kinect driver(s) released – Winner will use $3k for more hacking – PLUS an additional $2k goes to the EFF!", *Adafruit*. Retrieved from http://www.adafruit.com/blog/2010/11/10/we-have-a-winner-open-kinect-drivers-released-winner-will-use-3k-for-more-hacking-plus-an-additional-2k-goes-to-the-eff/

Altizer, R. (n.d.). "PLAYSTATION 3 (PS3) Specifications and Details". *About.com*. Retrieved from http://playstation.about.com/od/ps3/a/PS3SpecsDetails_3.htm

Andrews, J., Baker, N. (2006). "Xbox 360 System Architecture," *Micro, IEEE* 26(2), pp(25-37), doi: 10.1109/MM.2006.45

Arora, H. (2012). "TCP Attacks: TCP Sequence Number Prediction and TCP Reset Attacks". *The Geek Stuff*. Retrieved from http://www.thegeekstuff.com/2012/01/tcp-sequence-number-attacks/

Barardini, C. A. (May 2012). "The Xbox 360 System Specifications". *Team Xbox.* Retrieved from http://hardware.teamxbox.com/articles/xbox/1144/The-Xbox-360-System-Specifications/p1

BBC News (2012). "Playstation 'master key' leaked online", *BBC News*. Retrieved from http://www.bbc.co.uk/news/technology-20067289

Beer, J. (2012). "Rise of mobile gaming surprises big video-game developers", *Canadian Business*. Retrieved from http://www.canadianbusiness.com/article/75215--rise-of-mobile-gaming-surprises-big-video-game-developers

Bellovin, S. (1996). "Defending Against Sequence Number Attacks", *Network Working Group*. Retrieved from http://tools.ietf.org/html/rfc1948

Bolt, S. (2011). "Xbox 360 Forensics: A Digital Forensics Guide to Examining Artifacts", *Xbox 360 Forensics: A Digital Forensics Guide to Examining Artifacts.* ISBN-10: 1597496235

Chakravarty, S., Portokalidis, G., Polychronakis, M., Keromytis, A. D. (2011). "Detecting Traffic Snooping in Tor Using Decoys", *Recent Advances in Intrusion Detection*, Lecture Notes in Computer Science vol 6961 (2011), pp(222-241), DOI: 10.1007/978-3-642-23644-0_12.

Chou, N., Ledesma, R., Teraguchi, Y., Mitchell, J. C. (2004). "Client-side defense against web-based identity theft", *NDSS*, 2004. Retrieved from http://www.isoc.org/isoc/conferences/ndss/04/proceedings/Papers/Chou.pdf

Conrad, S., Dorn, G., Craiger, J. P. (2009). "Forensic Analysis of a Sony Play Station 3 Gaming Console", *6th Annual Conference of the International Federation of Information Processing*. Retrieved from http://computerforensicsllc.com/computer-forensics-expert-florida-miami-palm-beach-lauderdale-dave-kleiman-forensic-training-files/Forensic_Analysis_of_a_Sony_Play_Station_3_PS3_Gaming_Console.pdf

Conrad, S., Rodriguez, C., Marberry, C., Craiger, P. (2009). "Forensic Analysis of the Sony Playstation Portable", *Advances in digital Forensics V*, IFIP Advances in Information and Communication Technology vol. 306, 2009, pp(119-129). DOI: 10.1007/978-3-642-04155-6_9.

Dittrich, D. (1999). "Session Hijack Script". *washington.edu*. Retrieved from http://staff.washington.edu/dittrich/talks/qsm-sec/script.html

Doppner, T. W., Klein, P. N., Koyfman, A. (2000). "Using router stamping to identify the source of IP packets", *Proceedings of the 7th ACM conference on Computer and communications security*. Retrieved from http://dl.acm.org/citation.cfm?id=352627

Fowler, S. (2011). "An Overview of Small Scale Digital Forensics", *Senor Honors Theses*. Retrieved from http://commons.emich.edu/cgi/viewcontent.cgi?article=1282&context=honors&sei-redir=1

free60.org, (2011). "Console Security Certificate". *Free60.org*. Retrieved from http://www.free60.org/Console_Security_Certificate

free60.org, (2013). "FATX". *Free60.org*. Retrieved from http://www.free60.org/FATX

free60.org, (2013). "Free60". *Free60.org*. Retrieved from http://www.free60.org/Main_Page

Gaar, B. (2011). “Download Distribution Opening New doors for Independent Game Developers”, *statesman.com*. Retrieved from http://www.statesman.com/news/technology/download-distribution-opening-new-doors-for-inde-1/nRZHP/

Gill, R., Smith, J., Clark, A. (2006). “Experiences in passively detecting session hijacking attacks in IEEE 802.11 networks”, *ACSW Frontiers '06 Proceedings of the 2006 Australasian workshops on Grid computing and e-research*, vol. 54, pp(221-230). ISBN:1-920-68236-8

Gill, R., Smith, J., Looi, M., Clark, A. (2005). “Passive Techniques for Detecting Session Hijacking Attacks in IEEE 92.11 Wireless Networks”, *AUSCERT* May 2005. Retrieved from http ://eprints.qut.edu.au/21169/1/21169.pdf

Grobauer, B., Walloschek, T., Stocker, E. (2011). “Understanding Cloud Computing Vulnerabilities”, *IEEE Security & Privacy*, March/April 2011. Retrieved from http://ieeexplore.ieee.org.ezproxy.aut.ac.nz/stamp/stamp.jsp?tp=&arnumber=5487489

Prado, A., Harris, N., Gluck, Y. (2013). "SSL, Gone in 30 Seconds". *Breach Attack*. Retrieved from http://breachattack.com/

Willingham, T. (2012). "Video Games: Digital vs Retail Sales [infographic]". *Daily Infographic*. Retrieved from http://dailyinfographic.com/video-games-digital-vs-retail-sales-infographic

Microsoft (2004). "Microsoft Security Bulletin MS04-28". *Security TechCenter*. Retrieved from http://technet.microsoft.com/en-us/security/bulletin/ms04-028

Aircrack-ng (2013). "Aircrack-ng". *Aircrack-ng*. Retrieved from http://www.aircrack-ng.org/

mister_x (2010). "Tutorial: How to Crack WPA/WPA2". *Aircrack-ng*. Retrieved from http://www.aircrack-ng.org/doku.php?id=cracking_wpa

Knowles, D. (2007). "W32.Perrun". *Symantec*. Retrieved from http://www.symantec.com/security_response/writeup.jsp?docid=2002-061310-4234-99

Capano, F. (2013). "EditThisCookie". *Chrome Web Store*. Retrieved from https://chrome.google.com/webstore/detail/edit-this-cookie/fngmhnnpilhplaeedifhccceomclgfbg/details?hl=en

Rizzo, J., Duong, T. (2011). "The CRIME attack". *Google Drive*. Retrieved from

https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU/edit#slide=id.g1de53288_0_16

Jess, P. (2006). "Session Hijacking in Windows Networks", *SANS Institute Reading Room Site*. Retrieved from http://www.sans.org/reading_room/whitepapers/windows/session-hijacking-windows-networks_2124

Kent, S., Seo., K. (2005). "Security Architecture for the Internet Protocol", *Network Working Group*. Retrieved from http://tools.ietf.org/html/rfc4301

Langshaw, M. (November 2011). "Software Piracy: The Greatest Threat to the Gaming Industry?". *Digital Spy*. Retrieved from http://www.digitalspy.co.uk/gaming/news/a352906/software-piracy-the-greatest-threat-to-the-gaming-industry.html

Levitan, R. (August, 2012). "The Coming Video Game Distribution Wars". *Forbes.* Retrieved from http://www.forbes.com/sites/ciocentral/2012/08/01/the-coming-video-game-distribution-platform-wars/

Meisner, J. (October, 2012). "Daily Update, Oct. 11th: Blackjack! Xbox 360 Keeps the Lead in the U.S. Console Market for 21st Consecutive Month". *The Official Microsoft Blog*. Retrieved from http://blogs.technet.com/b/microsoft_blog/archive/2012/10/11/daily-update-oct-11th-blackjack-xbox-360-keeps-the-lead-in-the-u-s-console-market-for-21st-consecutive-month.aspx

Microsoft (2013). "Xbox 360 Digital Rights Management", *Xbox Support*. Retrieved from http://support.xbox.com/en-GB/xbox-live/marketplace-and-purchasing/download-content

Microsoft Corp. (June 2011). "Xbox Live Fact Sheet". *Microsoft.* Retrieved from http://www.microsoft.com/en-us/news/presskits/xbox/docs/XboxLIVEFS.docx

Mocanita, S. (2008). "Session Hijacking", *Server-Side Magazine*. Retrieved from http://www.serversidemagazine.com/php/session-hijacking/

Nintendo, (2012). "Technical Details". *Nintendo*. Retrieved from http://www.nintendo.co.uk/NOE/en_GB/systems/technical_details_1072.html

Oxford, T. (June 2010). "The truth about PC game piracy". *Tech Radar*. Retrieved from http://www.techradar.com/news/gaming/the-truth-about-pc-game-piracy-688864

Pitt, B. (2013). "backpack.tf". *backpack.tf*. Retrieved from http://backpack.tf/

Plunkett, L. (2010). "PS3 Loses Linux Support", *Kotaku*. Retrieved from http://kotaku.com/5504123/ps3-loses-linux-support

PS3 Developer Wiki (2013). "Boot Order", *PS3 Developer Wiki*. Retrieved from http://www.ps3devwiki.com/wiki/Boot_Order

PS3Wiki (2013). "PS3Wiki – LAN.ST", *PS3Wiki*. Retrieved from http://ps3wiki.lan.st/index.php/Main_Page

PSXBrew (2013). "Welcome to PSXBrew", *PSXBrew*. Retrieved http://www.psxbrew.net/wiki/Main_Page

Reed, J., Taylor, N., & Mackay, J. (2008). "Advanced videogame consoles - a new and unrecognised threat to secure service provision?" *The British Journal of Forensic Practice, 10*(4), pp(15-18). Retrieved from http://search.proquest.com/docview/213054384?accountid=8440

Ridgewell, W. W. (2011). "Determination and Exploitation of Potential Security Vulnerabilities in Networked Game Devices", *Athabasca University*. Retrieved from http://dtpr.lib.athabascau.ca/action/download.php?filename=scis-07/open/walterridgewellProject.pdf

Rivest, R. L., Shamir, A., Adleman, L. (n.d.). "A Method for Obtaining Digital signatures and Public-Key Cryptosystems", *Association for Computing Machinery*. Retrieved from http://people.csail.mit.edu/rivest/Rsapaper.pdf

Roofus, AngerWound (2009). "Xbox Dumper", *Xbox-Scene*. Retrieved from http://forums.xbox-scene.com/index.php?showtopic=462595&st=120&p=3180153&#entry3180153

Shrout, R. (2010). "The New Xbox 360 S 'Slim' Teardown: Opened and Tested", *PC Perspective*. Retrieved from http://www.pcper.com/reviews/General-Tech/New-Xbox-360-S-Slim-Teardown-Opened-and-Tested?aid=940

Sunhede, T. (n.d.). "The 6th Generation Video Game War", *LUND University*. Retrieved from http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=1316917&fileOId=1316918

Orland, K. (2012). "Steam vulnerability can lead to remote insertion of malicious code". *Ars Technica*. Retrieved from http://arstechnica.com/security/2012/10/steam-vulnerability-can-lead-to-remote-insertion-of-malicious-code/

Bode, K. (2003). "Steam Powered". *DSLReports.com*. Retrieved from http://www.broadbandreports.com/shownews/32475

Valve (2003). "Steam Client Released". *Steam*. Retrieved from http://store.steampowered.com/news/183/

Suddi, A. (2011). "Steam Breaks 5 Million Concurrent Users Mark", *tsa*. Retrieved from http://www.thesixthaxis.com/2012/01/04/steam-breaks-5-million-concurrent-users-mark/

Touch, J., Mankin, A., Bonica, R. (2010). "The TCP Authentication Option". *Internet Engineering Task Force*. Retrieved from http://tools.ietf.org/html/rfc5925

oz_paulb, pedrospad, Andrew, huceke (2006). "Xplorer 360". *Xbox-Scene.com*. Retrieved from http://www.xbox-scene.com/xbox360-tools/Xplorer360.php

Gont, F., Bellovin, S. (2012). "Defending against Sequence Number Attacks". *Internet Engineering Task Force*. Retrieved from http://tools.ietf.org/html/rfc6528

Valve (2013). "Steam & Game Stats". *Steam*. Retrieved from http://store.steampowered.com/stats/

Ballmer, S. (2012). "CES 2012: Steve Ballmer CES Keynote Address". *Youtube*. Retrieved from http://www.youtube.com/watch?v=KlPw3t1khHs

Anderson, R. (2011). "Modern Warfare 3 sets new XBL record". *That Videogame Blog*. Retrieved from http://www.strategyinformer.com/news/15467/modern-warfare-3-sets-xbox-live-concurrent-users-record

Valve Corporation (2009). "Team Fortress 2 – The Sniper and the Spy Update", *Team Fortress 2*. Retrieved from http://www.teamfortress.com/sniper_vs_spy/day09_english.htm

Varoufakis, Y. (2012). "Arbitrage and Equillibrium in the Team Fortress 2 Economy". *Valve*. Retrieved from http://blogs.valvesoftware.com/economics/arbitrage-and-equilibrium-in-the-team-fortress-2-economy/

Griffiths, D. N, (2012). "The Value of Fun: Valve Software Appoints In-House Economist". *Forbes*. Retrieved from

http://www.forbes.com/sites/danielnyegriffiths/2012/06/15/valve-appoints-in-house-economist/

Vaughan, C. (September 2004). “Xbox security issues and forensic recovery methodology (utilising Linux)”. Digital Investigation 1(3), pp(165-172). doi:10.1016/j.diin.2004.07.006

Wii Brew (2009). "Hardware/OTP". *Wii Brew*. Retrieved from http://wiibrew.org/wiki/Hardware/OTP

Dyer, M. (2012). "UPDATE: Microsoft Addresses Xbox.com Exploit". *IGN*. Retrieved from http://www.ign.com/articles/2012/01/13/update-microsoft-addresses-xboxcom-exploit

Peckham, M. (2010). "Microsoft discontinuing Xbox LIVE for Xbox Gamers". *PC World*. Retrieved from http://www.pcworld.com/article/188625/xbox_live_discontinued.html

Plunkett, L. (2012). "Is the Xbox Live 'Hacking' Problem Worse than Microsoft Realises?". *Kotaku*. Retrieved from http://kotaku.com/5873604/is-microsofts-xbox-live-hacking-problem-worse-than-microsoft-realises

Xynos, K., Harries, S., Sutherland, I., Davies, G., Blyth, A. (2010). “Xbox 360: A digital forensic investigation of the hard disk drive”, *Digital Investigation* 6(3-4), pp(104-111), http://dx.doi.org/10.1016/j.diin.2010.02.004.

Opera (2008). "Advisory: A JPEG image with a malformed header can crash Opera". *Opera*. Retrieved from http://www.opera.com/security/advisory/852

Appendix A: Xbox Live Login Capture

No.	Time	Source	Destination	Protocol	Length	Info
2279	4354.094206	Microsof_2f:53:0d	2wire_c2:e1:d9	ARP	42	192.168.1.72 is at 00:22:48:2f:53:0d
2280	4367.681984	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2281	4374.17869	Microsof_2f:53:0d	2wire_c2:e1:d9	ARP	42	192.168.1.72 is at 00:22:48:2f:53:0d
2282	4394.117758	Microsof_2f:53:0d	2wire_c2:e1:d9	ARP	42	192.168.1.72 is at 00:22:48:2f:53:0d
2283	4395.25344	192.168.1.72	134.170.179.54	TCP	54	57789 > https [ACK] Seq=1 Ack=1 Win=15331 Len=0
2284	4395.53658	192.168.1.72	134.170.179.54	TCP	54	[TCP Previous segment not captured] 57789 > https [ACK] Seq=315 Ack=48 Win=17165 Len=0
2285	4395.58317	192.168.1.72	134.170.179.54	TCP	1378	[TCP Previous segment not captured] [TCP segment of a reassembled PDU]
2286	4395.583684	192.168.1.72	134.170.179.54	TCP	1378	[TCP segment of a reassembled PDU]
2287	4395.584194	192.168.1.72	134.170.179.54	TCP	1378	[TCP segment of a reassembled PDU]
2288	4395.584706	192.168.1.72	134.170.179.54	TLSv1	1037	Application Data
2289	4395.839682	192.168.1.72	134.170.179.54	TCP	54	57789 > https [ACK] Seq=5296 Ack=429 Win=16784 Len=0
2290	4396.04864	192.168.1.72	134.170.179.54	TCP	54	[TCP Previous segment not captured] 57789 > https [ACK]

						Seq=5297 Ack=430 Win=16784 Len=0
2291	4397.04864	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2292	4397.117762	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2293	4397.701444	192.168.1.72	65.55.42.183	UDP	108	Source port: xbox Destination port: xbox
2294	4397.75162	192.168.1.72	65.55.42.183	UDP	78	Source port: xbox Destination port: xbox
2295	4397.954434	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2296	4397.96826	192.168.1.72	65.55.42.183	UDP	369	Source port: xbox Destination port: xbox
2297	4398.368642	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2298	4398.368642	192.168.1.72	65.55.42.183	UDP	90	Source port: xbox Destination port: xbox
2299	4399.102914	192.168.1.72	192.168.1.254	DNS	80	Standard query 0x889e A beacons.xboxlive.com
2300	4399.169472	192.168.1.72	65.55.42.33	TCP	66	46169 > https [SYN] Seq=0 Win=17212 Len=0 MSS=1324 WS=1 SACK_PERM=1
2301	4399.37376	192.168.1.72	65.55.42.33	TCP	54	46169 > https [ACK] Seq=1 Ack=1 Win=17212 Len=0
2302	4399.386048	192.168.1.72	65.55.42.33	TLSv1	117	Client Hello
2303	4399.470016	192.168.1.72	65.55.42.11	TLSv1	117	Client Hello
2304	4399.591872	192.168.1.72	65.55.42.33	TCP	54	46169 > https [ACK] Seq=64 Ack=2649 Win=14564 Len=0

2305	4399.677888	192.168.1.72	65.55.42.11	TCP	54	22643 > https [ACK] Seq=64 Ack=2649 Win=14564 Len=0
2306	4399.795648	192.168.1.72	65.55.42.33	TCP	54	46169 > https [ACK] Seq=64 Ack=4530 Win=15331 Len=0
2307	4399.820226	192.168.1.72	65.55.42.183	UDP	78	Source port: xbox Destination port: xbox
2308	4399.820736	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2309	4399.837634	192.168.1.72	65.55.42.183	UDP	78	Source port: xbox Destination port: xbox
2310	4399.884802	192.168.1.72	65.55.42.11	TCP	54	22643 > https [ACK] Seq=64 Ack=4530 Win=15331 Len=0
2311	4400.004098	192.168.1.72	65.55.42.11	TLSv1	368	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2312	4400.022016	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2313	4400.037376	192.168.1.72	65.55.42.183	UDP	241	Source port: xbox Destination port: xbox
2314	4400.042496	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2315	4400.081408	192.168.1.72	65.55.42.33	TCP	54	[TCP Previous segment not captured] 46169 > https [ACK] Seq=378 Ack=4577 Win=17165 Len=0
2316	4400.103936	192.168.1.72	65.55.42.33	TLSv1	80	Application Data
2317	4400.10496	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]
2318	4400.105472	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]
2319	4400.105472	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]

2320	4400.105984	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]
2321	4400.106496	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]
2322	4400.107008	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]
2323	4400.10752	192.168.1.72	65.55.42.33	TLSv1	707	Application Data
2324	4400.208384	192.168.1.72	219.88.185.11	TCP	54	44971 > http [FIN, ACK] Seq=477 Ack=31660 Win=64625 Len=0
2325	4400.217088	192.168.1.72	65.55.42.11	TCP	54	22643 > https [ACK] Seq=378 Ack=4577 Win=17165 Len=0
2326	4400.221184	192.168.1.72	192.168.1.254	DNS	87	Standard query 0x88c2 A roamingprofile.xboxlive.com
2327	4400.227328	192.168.1.72	219.88.185.11	TCP	54	44971 > http [ACK] Seq=478 Ack=31661 Win=64625 Len=0
2328	4400.233984	192.168.1.72	219.88.185.11	TCP	54	38420 > http [FIN, ACK] Seq=475 Ack=20377 Win=64732 Len=0
2329	4400.237056	192.168.1.72	65.55.42.11	TCP	54	22643 > https [FIN, ACK] Seq=378 Ack=4577 Win=17165 Len=0
2330	4400.24781	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2331	4400.303616	192.168.1.72	65.55.42.33	TCP	66	nta-ds > https [SYN] Seq=0 Win=17212 Len=0 MSS=1324 WS=1 SACK_PERM=1
2332	4400.320512	192.168.1.72	192.168.1.254	DNS	80	Standard query 0x88cc A eplists.xboxlive.com
2333	4400.332288	192.168.1.72	219.88.186.105	TCP	54	[TCP Previous segment not captured] 22126 > http [FIN, ACK]

						Seq=679 Ack=646 Win=64231 Len=0
2334	4400.3328	192.168.1.72	65.55.42.183	UDP	78	Source port: xbox Destination port: xbox
2335	4400.337408	192.168.1.72	65.55.42.183	UDP	78	Source port: xbox Destination port: xbox
2336	4400.33741	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2337	4400.410624	192.168.1.72	65.55.42.33	TCP	54	46169 > https [RST, ACK] Seq=9002 Ack=4577 Win=0 Len=0
2338	4400.520706	192.168.1.72	192.168.1.254	DNS	80	Standard query 0x88d7 A beacons.xboxlive.com
2339	4400.536576	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2340	4400.5376	192.168.1.72	65.55.42.183	UDP	871	Source port: xbox Destination port: xbox
2341	4400.541184	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2342	4400.554496	192.168.1.72	65.55.42.183	UDP	227	Source port: xbox Destination port: xbox
2343	4400.59392	192.168.1.72	219.88.186.105	TCP	54	[TCP Previous segment not captured] 61685 > http [ACK] Seq=1584 Ack=1507 Win=64876 Len=0
2344	4400.654338	192.168.1.72	157.56.70.154	TCP	66	57282 > https [SYN] Seq=0 Win=17212 Len=0 MSS=1324 WS=1 SACK_PERM=1
2345	4400.720896	192.168.1.72	65.55.42.11	TCP	66	12679 > https [SYN] Seq=0 Win=17212 Len=0 MSS=1324 WS=1 SACK_PERM=1
2346	4400.821762	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox

2347	4400.832	192.168.1.72	157.56.70.154	TCP	54	57282 > https [ACK] Seq=1 Ack=1 Win=17212 Len=0
2348	4400.837632	192.168.1.72	157.56.70.154	SSL	117	Client Hello
2349	4400.8448	192.168.1.72	65.55.42.33	TCP	54	[TCP Previous segment not captured] nta-ds > https [ACK] Seq=96 Ack=1 Win=15331 Len=0
2350	4400.929344	192.168.1.72	65.55.42.11	TCP	54	12679 > https [ACK] Seq=1 Ack=1 Win=17212 Len=0
2351	4400.937536	192.168.1.72	65.55.42.11	TLSv1	117	Client Hello
2352	4400.93805	192.168.1.72	65.55.42.33	TLSv1	368	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2353	4400.954432	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=1 Ack=1 Win=64512 Len=0
2354	4400.955456	192.168.1.72	157.56.70.48	TLSv1	149	Client Hello
2355	4401.014848	192.168.1.72	157.56.70.154	TCP	54	57282 > https [ACK] Seq=64 Ack=2649 Win=14564 Len=0
2356	4401.123906	192.168.1.72	65.55.42.33	TCP	54	nta-ds > https [ACK] Seq=410 Ack=48 Win=17165 Len=0
2357	4401.13312	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=96 Ack=2649 Win=61864 Len=0
2358	4401.150018	192.168.1.72	65.55.42.11	TCP	54	12679 > https [ACK] Seq=64 Ack=2649 Win=14564 Len=0
2359	4401.154626	192.168.1.72	65.55.42.33	TLSv1	80	Application Data
2360	4401.15565	192.168.1.72	65.55.42.33	TCP	1378	[TCP Previous segment not captured] [TCP segment of a reassembled PDU]

2361	4401.15616	192.168.1.72	65.55.42.33	TCP	1378	[TCP Previous segment not captured] [TCP segment of a reassembled PDU]
2362	4401.156672	192.168.1.72	65.55.42.33	TCP	1378	[TCP Out-Of-Order] [TCP segment of a reassembled PDU]
2363	4401.157184	192.168.1.72	65.55.42.33	TCP	1378	[TCP Out-Of-Order] [TCP segment of a reassembled PDU]
2364	4401.157696	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]
2365	4401.157698	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]
2366	4401.158208	192.168.1.72	65.55.42.33	TCP	702	[TCP segment of a reassembled PDU]
2367	4401.340994	192.168.1.72	157.56.70.48	TLSv1	368	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2368	4401.357378	192.168.1.72	65.55.42.11	TCP	54	12679 > https [ACK] Seq=64 Ack=4530 Win=15331 Len=0
2369	4401.438274	192.168.1.72	65.55.42.11	TLSv1	368	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2370	4401.443392	192.168.1.72	65.55.42.33	TCP	54	nta-ds > https [ACK] Seq=9028 Ack=558 Win=16655 Len=0
2371	4401.45517	192.168.1.72	65.55.42.33	TCP	54	nta-ds > https [FIN, ACK] Seq=9028 Ack=558 Win=16655 Len=0
2372	4401.471042	192.168.1.72	157.56.70.154	TCP	54	[TCP Previous segment not captured] 57282 > https [ACK] Seq=378 Ack=4577 Win=17165 Len=0
2373	4401.518146	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=410 Ack=4577 Win=64465 Len=0

2374	4401.519682	192.168.1.72	157.56.70.48	TLSv1	80	Application Data
2375	4401.520194	192.168.1.72	157.56.70.48	TCP	1378	[TCP segment of a reassembled PDU]
2376	4401.520706	192.168.1.72	157.56.70.48	TCP	1378	[TCP segment of a reassembled PDU]
2377	4401.521218	192.168.1.72	157.56.70.48	TCP	1378	[TCP segment of a reassembled PDU]
2378	4401.52173	192.168.1.72	157.56.70.48	TCP	1378	[TCP segment of a reassembled PDU]
2379	4401.52224	192.168.1.72	157.56.70.48	TCP	1378	[TCP segment of a reassembled PDU]
2380	4401.522752	192.168.1.72	157.56.70.48	TCP	1378	[TCP segment of a reassembled PDU]
2381	4401.523264	192.168.1.72	157.56.70.48	TLSv1	652	Application Data
2382	4401.630274	192.168.1.72	65.55.42.33	TCP	54	nta-ds > https [ACK] Seq=9029 Ack=559 Win=16655 Len=0
2383	4401.65229	192.168.1.72	65.55.42.11	TCP	54	12679 > https [ACK] Seq=378 Ack=4577 Win=17165 Len=0
2384	4401.671746	192.168.1.72	65.55.42.11	TLSv1	80	Application Data
2385	4401.67277	192.168.1.72	65.55.42.11	TCP	1378	[TCP Previous segment not captured] [TCP segment of a reassembled PDU]
2386	4401.673282	192.168.1.72	65.55.42.11	TCP	1378	[TCP Previous segment not captured] [TCP segment of a reassembled PDU]
2387	4401.673792	192.168.1.72	65.55.42.11	TCP	1378	[TCP Out-Of-Order] [TCP segment of a reassembled PDU]
2388	4401.674306	192.168.1.72	65.55.42.11	TCP	1378	[TCP Out-Of-Order] [TCP segment of a reassembled PDU]

2389	4401.674816	192.168.1.72	65.55.42.11	TCP	1378	[TCP segment of a reassembled PDU]
2390	4401.675328	192.168.1.72	65.55.42.11	TCP	1378	[TCP segment of a reassembled PDU]
2391	4401.675328	192.168.1.72	65.55.42.11	TCP	717	[TCP segment of a reassembled PDU]
2392	4401.876098	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=7225 Win=64512 Len=0
2393	4401.877122	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=9873 Win=64512 Len=0
2394	4401.973378	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=11197 Win=64512 Len=0
2395	4402.050176	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=13845 Win=64512 Len=0
2396	4402.050178	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=16493 Win=64512 Len=0
2397	4402.05325	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=19141 Win=64512 Len=0
2398	4402.055298	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=21789 Win=64512 Len=0
2399	4402.139266	192.168.1.72	157.56.70.154	TCP	54	[TCP Previous segment not captured] 57282 > https [FIN, ACK] Seq=9019 Ack=5111 Win=16631 Len=0
2400	4402.145922	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=24437 Win=61864 Len=0

2401	4402.18893	192.168.1.72	65.55.42.36	TCP	66	14234 > https [SYN] Seq=0 Win=17212 Len=0 MSS=1324 WS=1 SACK_PERM=1
2402	4402.229378	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=32381 Win=64512 Len=0
2403	4402.229378	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=35029 Win=64512 Len=0
2404	4402.232962	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=37677 Win=64512 Len=0
2405	4402.236546	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=40325 Win=63188 Len=0
2406	4402.239106	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=42973 Win=64512 Len=0
2407	4402.239106	192.168.1.72	157.56.70.48	TCP	54	10751 > https [ACK] Seq=8978 Ack=45621 Win=64512 Len=0
2408	4402.239618	192.168.1.72	65.55.42.11	TCP	54	[TCP Previous segment not captured] 12679 > https [ACK] Seq=9012 Ack=4813 Win=16930 Len=0
2409	4402.387586	192.168.1.72	65.55.42.36	TCP	54	14234 > https [ACK] Seq=1 Ack=1 Win=17212 Len=0
2410	4402.38912	192.168.1.72	65.55.42.36	SSL	117	Client Hello
2411	4402.472578	192.168.1.72	65.55.42.36	TCP	54	14234 > https [FIN, ACK] Seq=64 Ack=1 Win=17212 Len=0

2412	4402.59136	192.168.1.72	65.55.42.36	TCP	54	14234 > https [RST, ACK] Seq=65 Ack=1 Win=0 Len=0
2413	4402.807426	192.168.1.72	157.56.70.48	TCP	66	59272 > https [SYN] Seq=0 Win=17212 Len=0 MSS=1324 WS=1 SACK_PERM=1
2414	4402.873026	192.168.1.72	192.168.1.254	DNS	80	Standard query 0x8948 A beacons.xboxlive.com
2415	4402.977472	192.168.1.72	157.56.70.48	TCP	54	59272 > https [ACK] Seq=1 Ack=1 Win=17212 Len=0
2416	4402.989762	192.168.1.72	157.56.70.48	TLSv1	149	Client Hello
2417	4402.990272	192.168.1.72	65.55.42.183	UDP	78	Source port: xbox Destination port: xbox
2418	4403.100354	192.168.1.72	184.27.91.156	TCP	66	63026 > https [SYN] Seq=0 Win=64876 Len=0 MSS=1324 WS=1 SACK_PERM=1
2419	4403.107522	192.168.1.72	219.88.186.112	TCP	54	13898 > http [FIN, ACK] Seq=566 Ack=264492 Win=64100 Len=0
2420	4403.108034	192.168.1.72	192.168.1.254	DNS	76	Standard query 0x8954 A eds.xboxlive.com
2421	4403.11469	192.168.1.72	65.55.42.11	TCP	54	18430 > https [ACK] Seq=1 Ack=1 Win=17212 Len=0
2422	4403.16333	192.168.1.72	157.56.70.48	TCP	54	59272 > https [ACK] Seq=96 Ack=2649 Win=14564 Len=0
2423	4403.19456	192.168.1.72	184.27.91.156	TCP	66	35122 > https [SYN] Seq=0 Win=64876 Len=0 MSS=1324 WS=1 SACK_PERM=1
2424	4403.197122	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2425	4403.207362	192.168.1.72	65.55.42.183	UDP	1338	Source port: xbox Destination port: xbox

2426	4403.20736	192.168.1.72	65.55.42.183	UDP	372	Source port: xbox Destination port: xbox
2427	4403.20992	192.168.1.72	184.27.91.156	TLSv1	368	[TCP Previous segment not captured] Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2428	4403.216576	192.168.1.72	184.27.91.156	TCP	54	35122 > https [ACK] Seq=1 Ack=1 Win=64876 Len=0
2429	4403.216576	192.168.1.72	184.27.91.156	TLSv1	117	Client Hello
2430	4403.25242	192.168.1.72	184.27.91.156	TCP	54	63026 > https [ACK] Seq=378 Ack=48 Win=64202 Len=0
2431	4403.254976	192.168.1.72	184.27.91.156	TLSv1	80	Application Data
2432	4403.256	192.168.1.72	184.27.91.156	TCP	1378	[TCP segment of a reassembled PDU]
2433	4403.256512	192.168.1.72	184.27.91.156	TCP	1378	[TCP segment of a reassembled PDU]
2434	4403.257026	192.168.1.72	184.27.91.156	TCP	1378	[TCP segment of a reassembled PDU]
2435	4403.257536	192.168.1.72	184.27.91.156	TCP	1378	[TCP segment of a reassembled PDU]
2436	4403.25805	192.168.1.72	184.27.91.156	TCP	1378	[TCP segment of a reassembled PDU]
2437	4403.258048	192.168.1.72	184.27.91.156	TCP	1378	[TCP segment of a reassembled PDU]
2438	4403.25856	192.168.1.72	184.27.91.156	TLSv1	979	Application Data
2439	4403.258562	192.168.1.72	184.27.91.156	TCP	54	35122 > https [ACK] Seq=64 Ack=2649 Win=63552 Len=0
2440	4403.259072	192.168.1.72	184.27.91.156	TCP	54	35122 > https [ACK] Seq=64 Ack=3276 Win=62925 Len=0

2441	4403.298498	192.168.1.72	65.55.42.11	TCP	54	[TCP Previous segment not captured] 18430 > https [ACK] Seq=96 Ack=2649 Win=14564 Len=0
2442	4403.309762	192.168.1.72	184.27.91.156	TLSv1	368	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2443	4403.378882	192.168.1.72	184.27.91.156	TCP	54	35122 > https [ACK] Seq=378 Ack=3323 Win=64829 Len=0
2444	4403.380418	192.168.1.72	184.27.91.156	TLSv1	80	Application Data
2445	4403.381442	192.168.1.72	184.27.91.156	TCP	1378	[TCP segment of a reassembled PDU]
2446	4403.381954	192.168.1.72	184.27.91.156	TCP	1378	[TCP segment of a reassembled PDU]
2447	4403.382466	192.168.1.72	184.27.91.156	TCP	1378	[TCP segment of a reassembled PDU]
2448	4403.382978	192.168.1.72	184.27.91.156	TCP	1378	[TCP segment of a reassembled PDU]
2449	4403.382978	192.168.1.72	184.27.91.156	TCP	1378	[TCP segment of a reassembled PDU]
2450	4403.38349	192.168.1.72	184.27.91.156	TCP	1378	[TCP segment of a reassembled PDU]
2451	4403.384002	192.168.1.72	184.27.91.156	TLSv1	977	Application Data
2452	4403.62413	192.168.1.72	65.55.42.11	TLSv1	368	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2453	4403.687618	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2454	4403.689154	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox

2455	4403.692226	192.168.1.72	157.56.70.48	TCP	54	[TCP Previous segment not captured] 59272 > https [ACK] Seq=410 Ack=4577 Win=17165 Len=0
2456	4403.72397	192.168.1.72	157.56.70.48	TLSv1	80	Application Data
2457	4403.724994	192.168.1.72	157.56.70.48	TCP	1378	[TCP segment of a reassembled PDU]
2458	4403.725506	192.168.1.72	157.56.70.48	TCP	1378	[TCP segment of a reassembled PDU]
2459	4403.726018	192.168.1.72	157.56.70.48	TCP	1378	[TCP segment of a reassembled PDU]
2460	4403.726018	192.168.1.72	157.56.70.48	TCP	1378	[TCP segment of a reassembled PDU]
2461	4403.72653	192.168.1.72	157.56.70.48	TCP	1378	[TCP segment of a reassembled PDU]
2462	4403.728578	192.168.1.72	157.56.70.48	TCP	1378	[TCP segment of a reassembled PDU]
2463	4403.728578	192.168.1.72	157.56.70.48	TLSv1	661	Application Data
2464	4403.804354	192.168.1.72	65.55.42.11	TCP	54	18430 > https [ACK] Seq=410 Ack=4577 Win=17165 Len=0
2465	4403.82381	192.168.1.72	65.55.42.11	TLSv1	80	Application Data
2466	4403.825346	192.168.1.72	65.55.42.11	TCP	1378	[TCP Previous segment not captured] [TCP segment of a reassembled PDU]
2467	4403.825346	192.168.1.72	65.55.42.11	TCP	1378	[TCP Previous segment not captured] [TCP segment of a reassembled PDU]
2468	4403.82637	192.168.1.72	65.55.42.11	TCP	1378	[TCP Out-Of-Order] [TCP segment of a reassembled PDU]

2469	4403.826882	192.168.1.72	65.55.42.11	TCP	1378	[TCP Out-Of-Order] [TCP segment of a reassembled PDU]
2470	4403.826882	192.168.1.72	65.55.42.11	TCP	1378	[TCP segment of a reassembled PDU]
2471	4403.827394	192.168.1.72	65.55.42.11	TCP	1378	[TCP segment of a reassembled PDU]
2472	4403.827394	192.168.1.72	65.55.42.11	TCP	717	[TCP segment of a reassembled PDU]
2473	4403.89453	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2474	4403.895554	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2475	4403.984642	192.168.1.72	184.27.91.156	TCP	54	63026 > https [ACK] Seq=9273 Ack=5868 Win=64727 Len=0
2476	4403.99181	192.168.1.72	192.168.1.254	DNS	87	Standard query 0x899b A roamingprofile.xboxlive.com
2477	4404.010242	192.168.1.72	157.56.70.48	TCP	54	59272 > https [ACK] Seq=8987 Ack=7225 Win=17212 Len=0
2478	4404.010754	192.168.1.72	157.56.70.48	TCP	54	59272 > https [ACK] Seq=8987 Ack=8070 Win=16367 Len=0
2479	4404.074242	192.168.1.72	65.55.42.33	TCP	66	30530 > https [SYN] Seq=0 Win=17212 Len=0 MSS=1324 WS=1 SACK_PERM=1
2480	4404.125954	192.168.1.72	65.55.42.11	TCP	54	18430 > https [ACK] Seq=9043 Ack=4812 Win=16930 Len=0
2481	4404.224768	192.168.1.72	192.168.1.254	DNS	90	Standard query 0x89af A socialaggregation.xboxlive.com
2482	4404.312834	192.168.1.72	65.55.42.11	TCP	54	[TCP Previous segment not captured] 18430 > https [ACK] Seq=9044 Ack=4813 Win=16930 Len=0
2483	4404.395266	192.168.1.72	184.27.91.156	TCP	54	63026 > https [ACK] Seq=9273 Ack=41334 Win=64876

						Len=0
2484	4404.39885	192.168.1.72	184.27.91.156	TCP	54	63026 > https [ACK] Seq=9273 Ack=43982 Win=64876 Len=0
2485	4404.400896	192.168.1.72	184.27.91.156	TCP	54	63026 > https [ACK] Seq=9273 Ack=46630 Win=64876 Len=0
2486	4404.402946	192.168.1.72	184.27.91.156	TCP	54	63026 > https [ACK] Seq=9273 Ack=49278 Win=64876 Len=0
2487	4404.40448	192.168.1.72	184.27.91.156	TCP	54	63026 > https [ACK] Seq=9273 Ack=51926 Win=64876 Len=0
2488	4404.407042	192.168.1.72	184.27.91.156	TCP	54	63026 > https [ACK] Seq=9273 Ack=54574 Win=64876 Len=0
2489	4404.40704	192.168.1.72	184.27.91.156	TCP	54	63026 > https [ACK] Seq=9273 Ack=55095 Win=64355 Len=0
2490	4404.412162	192.168.1.72	184.27.91.156	TCP	54	35122 > https [ACK] Seq=9271 Ack=19081 Win=64876 Len=0
2491	4404.414722	192.168.1.72	184.27.91.156	TCP	54	35122 > https [ACK] Seq=9271 Ack=21729 Win=64876 Len=0
2492	4404.414722	192.168.1.72	184.27.91.156	TCP	54	35122 > https [ACK] Seq=9271 Ack=22830 Win=63775 Len=0
2493	4404.416258	192.168.1.72	184.27.91.156	TCP	54	35122 > https [ACK] Seq=9271 Ack=24879 Win=64151

						Len=0
2494	4404.436736	192.168.1.72	65.55.42.33	TCP	54	[TCP Previous segment not captured] 30530 > https [ACK] Seq=96 Ack=1 Win=14564 Len=0
2495	4404.441346	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2496	4404.441858	192.168.1.72	65.55.42.183	UDP	78	Source port: xbox Destination port: xbox
2497	4404.589314	192.168.1.72	219.88.185.49	TCP	66	48172 > http [SYN] Seq=0 Win=64876 Len=0 MSS=1324 WS=1 SACK_PERM=1
2498	4404.610816	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=1 Ack=1 Win=64876 Len=0
2499	4404.611842	192.168.1.72	219.88.185.49	HTTP	226	GET /CONTENT/IMAGES//40/5d17949b-e5ff-4f9a-9a2a-664c72ae7040.jpg HTTP/1.1
2500	4404.61184	192.168.1.72	65.55.42.33	TCP	54	30530 > https [ACK] Seq=96 Ack=1882 Win=15331 Len=0
2501	4404.617984	192.168.1.72	192.168.1.254	DNS	77	Standard query 0x89e7 A download.xbox.com
2502	4404.626688	192.168.1.72	157.56.70.48	TCP	54	[TCP Previous segment not captured] 10751 > https [ACK] Seq=8979 Ack=60066 Win=63308 Len=0
2503	4404.628224	192.168.1.72	192.168.1.254	DNS	77	Standard query 0x89ea A download.xbox.com
2504	4404.628224	192.168.1.72	184.27.91.156	TCP	54	35122 > https [FIN, ACK] Seq=9271 Ack=24879 Win=64151 Len=0
2505	4404.636416	192.168.1.72	192.168.1.254	DNS	80	Standard query 0x89ed A mktplassets.xbox.com

2506	4404.639488	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=2649 Win=64876 Len=0
2507	4404.641024	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=5297 Win=64876 Len=0
2508	4404.64461	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=7945 Win=64876 Len=0
2509	4404.64768	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=10593 Win=64876 Len=0
2510	4404.650752	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=13241 Win=64876 Len=0
2511	4404.650752	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2512	4404.651264	192.168.1.72	184.27.91.156	TCP	54	35122 > https [RST, ACK] Seq=9272 Ack=24879 Win=0 Len=0
2513	4404.657922	192.168.1.72	65.55.42.183	UDP	225	Source port: xbox Destination port: xbox
2514	4404.66304	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=15889 Win=64876 Len=0
2515	4404.666624	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=18537 Win=64876 Len=0
2516	4404.67072	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=21185 Win=64876 Len=0
2517	4404.672256	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=23833 Win=64876 Len=0
2518	4404.675328	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=26481 Win=64876 Len=0
2519	4404.678914	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=29129 Win=64876 Len=0
2520	4404.68045	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=31777 Win=64876 Len=0

2521	4404.682496	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=34425 Win=62228 Len=0
2522	4404.691202	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=37073 Win=59580 Len=0
2523	4404.691202	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=39721 Win=56932 Len=0
2524	4404.694786	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=42369 Win=54284 Len=0
2525	4404.69632	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=45017 Win=51636 Len=0
2526	4404.700416	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=47665 Win=48988 Len=0
2527	4404.701952	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=50313 Win=63552 Len=0
2528	4404.704512	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=52961 Win=64876 Len=0
2529	4404.708608	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=55609 Win=64876 Len=0
2530	4404.70861	192.168.1.72	65.55.42.33	TLSv1	368	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2531	4404.710144	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=58257 Win=64876 Len=0
2532	4404.714752	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=60905 Win=64876 Len=0
2533	4404.716288	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=63553 Win=64876 Len=0
2534	4404.720384	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=66201 Win=63552 Len=0
2535	4404.722432	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=68849 Win=64876 Len=0

2536	4404.722434	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=71497 Win=64876 Len=0
2537	4404.726016	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=74145 Win=64876 Len=0
2538	4404.732672	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=76793 Win=64876 Len=0
2539	4404.732672	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=79441 Win=64876 Len=0
2540	4404.733184	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=82089 Win=64876 Len=0
2541	4404.733184	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=84737 Win=64876 Len=0
2542	4404.733696	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=87385 Win=64876 Len=0
2543	4404.73677	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=90033 Win=64876 Len=0
2544	4404.73728	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=92681 Win=64876 Len=0
2545	4404.741888	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=95329 Win=64876 Len=0
2546	4404.741888	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=97977 Win=64876 Len=0
2547	4404.746496	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=100625 Win=64876 Len=0
2548	4404.749058	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=103273 Win=64876 Len=0
2549	4404.749058	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=105921 Win=64876 Len=0
2550	4404.74957	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=108569 Win=64876 Len=0
2551	4404.791554	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=158881 Win=64876 Len=0

2552	4404.791552	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=161529 Win=64876 Len=0
2553	4404.792064	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=164177 Win=64876 Len=0
2554	4404.792066	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=166825 Win=64876 Len=0
2555	4404.796672	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=169473 Win=64876 Len=0
2556	4404.797186	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=172121 Win=64876 Len=0
2557	4404.797186	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=174769 Win=62228 Len=0
2558	4404.797184	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=177417 Win=64876 Len=0
2559	4404.81101	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=180065 Win=64876 Len=0
2560	4404.811008	192.168.1.72	65.55.42.36	TCP	54	45914 > https [ACK] Seq=1 Ack=1 Win=14564 Len=0
2561	4404.811008	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=182713 Win=64876 Len=0
2562	4404.81152	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=185361 Win=64876 Len=0
2563	4404.811522	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=188009 Win=64876 Len=0
2564	4404.811522	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=190657 Win=64876 Len=0
2565	4404.812034	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=193305 Win=64876 Len=0
2566	4404.812032	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=195953 Win=64876 Len=0
2567	4404.812544	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=198601 Win=62228 Len=0

2568	4404.812544	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=201249 Win=64876 Len=0
2569	4404.812546	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=203897 Win=64876 Len=0
2570	4404.814592	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=206545 Win=64876 Len=0
2571	4404.81664	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=209193 Win=64876 Len=0
2572	4404.821248	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=211841 Win=64876 Len=0
2573	4404.82125	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=173 Ack=213293 Win=64748 Len=0
2574	4404.892482	192.168.1.72	65.55.42.33	TCP	54	30530 > https [ACK] Seq=410 Ack=1929 Win=17165 Len=0
2575	4404.921664	192.168.1.72	219.88.185.49	HTTP	275	[TCP Previous segment not captured] GET /CONTENT/IMAGES//cb/36bced75-352c-4ab1-81f2-9061c24d1dcb.jpg HTTP/1.1
2576	4404.924224	192.168.1.72	219.88.185.49	TCP	66	40291 > http [SYN] Seq=0 Win=64876 Len=0 MSS=1324 WS=1 SACK_PERM=1
2577	4404.92525	192.168.1.72	65.55.42.33	TLSv1	80	Application Data
2578	4404.926272	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]
2579	4404.926274	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]
2580	4404.926784	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]
2581	4404.927298	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]

2582	4404.927808	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]
2583	4404.928322	192.168.1.72	65.55.42.33	TCP	1378	[TCP segment of a reassembled PDU]
2584	4404.92832	192.168.1.72	65.55.42.33	TLSv1	722	Application Data
2585	4404.942146	192.168.1.72	219.88.185.49	TCP	54	48172 > http [ACK] Seq=615 Ack=213723 Win=64318 Len=0
2586	4404.944192	192.168.1.72	219.88.185.49	TCP	54	40291 > http [ACK] Seq=1 Ack=1 Win=64876 Len=0
2587	4404.944192	192.168.1.72	219.88.185.49	HTTP	275	GET /CONTENT/IMAGES//b6/77deb510-4c6e-4b62-adca-a0b9b860ffb6.jpg HTTP/1.1
2588	4405.023552	192.168.1.72	65.55.42.36	TCP	54	45914 > https [ACK] Seq=1 Ack=1882 Win=15331 Len=0
2589	4405.046594	192.168.1.72	219.88.185.49	TCP	54	40291 > http [ACK] Seq=222 Ack=216 Win=64661 Len=0
2590	4405.141314	192.168.1.72	219.88.186.105	TCP	66	30973 > http [SYN] Seq=0 Win=64876 Len=0 MSS=1324 WS=1 SACK_PERM=1
2591	4405.142338	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2592	4405.162818	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=1 Ack=1 Win=64876 Len=0
2593	4405.210946	192.168.1.72	219.88.186.105	TCP	54	[TCP Previous segment not captured] 30973 > http [ACK] Seq=178 Ack=15889 Win=64876 Len=0
2594	4405.21453	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=18537 Win=64876 Len=0
2595	4405.215042	192.168.1.72	65.55.42.33	TCP	54	30530 > https [ACK] Seq=9048 Ack=2439 Win=16655 Len=0

2596	4405.221186	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=21185 Win=64876 Len=0
2597	4405.22272	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=23833 Win=64876 Len=0
2598	4405.22477	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=26481 Win=64876 Len=0
2599	4405.225282	192.168.1.72	65.55.42.33	TCP	54	30530 > https [FIN, ACK] Seq=9048 Ack=2439 Win=16655 Len=0
2600	4405.229376	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=29129 Win=64876 Len=0
2601	4405.230402	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=31777 Win=64876 Len=0
2602	4405.23245	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=34425 Win=62228 Len=0
2603	4405.239106	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=37073 Win=59580 Len=0
2604	4405.239106	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=39721 Win=56932 Len=0
2605	4405.241154	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=42369 Win=54284 Len=0
2606	4405.244738	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=45017 Win=51636 Len=0
2607	4405.24781	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=47665 Win=48988 Len=0
2608	4405.249858	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=50313 Win=46340 Len=0
2609	4405.253954	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=52961 Win=43692 Len=0
2610	4405.257026	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=55609 Win=63552 Len=0

2611	4405.257026	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=178 Ack=57583 Win=64226 Len=0
2612	4405.30925	192.168.1.72	219.88.186.105	TCP	66	17447 > http [SYN] Seq=0 Win=64876 Len=0 MSS=1324 WS=1 SACK_PERM=1
2613	4405.40909	192.168.1.72	219.88.186.105	TCP	54	[TCP Previous segment not captured] 30973 > http [ACK] Seq=404 Ack=57798 Win=64011 Len=0
2614	4405.40909	192.168.1.72	192.168.1.254	DNS	83	Standard query 0x8a86 A xperiments.xboxlive.com
2615	4405.41165	192.168.1.72	219.88.186.105	TCP	54	17447 > http [ACK] Seq=1 Ack=1 Win=64876 Len=0
2616	4405.4224	192.168.1.72	219.88.186.105	HTTP	280	GET /content/images/66acd000-77fe-1000-9115-d802555307f3/5129/boxartlg.jpg HTTP/1.1
2617	4405.432642	192.168.1.72	219.88.186.105	HTTP	280	GET /content/images/66acd000-77fe-1000-9115-d802545407e7/1033/boxartlg.jpg HTTP/1.1
2618	4405.444928	192.168.1.72	219.88.186.105	TCP	54	17447 > http [ACK] Seq=227 Ack=216 Win=64661 Len=0
2619	4405.455682	192.168.1.72	219.88.186.105	HTTP	280	GET /content/images/66acd000-77fe-1000-9115-d8024d5307e8/5129/boxartlg.jpg HTTP/1.1
2620	4405.46029	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=630 Ack=58013 Win=63796 Len=0
2621	4405.508928	192.168.1.72	134.170.178.64	TCP	66	54784 > https [SYN] Seq=0 Win=17212 Len=0 MSS=1324 WS=1 SACK_PERM=1
2622	4405.520706	192.168.1.72	219.88.186.105	HTTP	280	GET /content/images/66acd000-77fe-1000-9115-d802545a07d2/1033/boxartlg.jpg HTTP/1.1

2623	4405.52736	192.168.1.72	219.88.186.105	TCP	54	[TCP Previous segment not captured] 30973 > http [ACK] Seq=856 Ack=58228 Win=63581 Len=0
2624	4405.53453	192.168.1.72	219.88.186.105	HTTP	280	GET /content/images/66acd000-77fe-1000-9115-d802415807d5/1033/boxartlg.jpg HTTP/1.1
2625	4405.54528	192.168.1.72	219.88.186.105	TCP	54	17447 > http [ACK] Seq=679 Ack=646 Win=64231 Len=0
2626	4405.558592	192.168.1.72	219.88.186.105	TCP	54	30973 > http [ACK] Seq=1082 Ack=58443 Win=64876 Len=0
2627	4405.659456	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2628	4405.873024	192.168.1.72	134.170.178.64	TCP	54	[TCP Previous segment not captured] 54784 > https [ACK] Seq=96 Ack=1 Win=14564 Len=0
2629	4406.050178	192.168.1.72	134.170.178.64	TCP	54	54784 > https [ACK] Seq=96 Ack=1882 Win=15331 Len=0
2630	4406.125954	192.168.1.72	157.56.70.48	TCP	54	59272 > https [FIN, ACK] Seq=8987 Ack=8070 Win=16367 Len=0
2631	4406.143362	192.168.1.72	134.170.178.64	TLSv1	368	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2632	4406.326146	192.168.1.72	134.170.178.64	TCP	54	54784 > https [ACK] Seq=410 Ack=1929 Win=17165 Len=0
2633	4406.343042	192.168.1.72	134.170.178.64	TLSv1	80	Application Data
2634	4406.344066	192.168.1.72	134.170.178.64	TCP	1378	[TCP segment of a reassembled PDU]
2635	4406.344578	192.168.1.72	134.170.178.64	TCP	1378	[TCP Previous segment not captured] [TCP segment of a

						reassembled PDU]
2636	4406.34509	192.168.1.72	134.170.178.64	TCP	1378	[TCP Out-Of-Order] [TCP segment of a reassembled PDU]
2637	4406.34509	192.168.1.72	134.170.178.64	TLSv1	1037	Application Data
2638	4406.576002	192.168.1.72	134.170.178.64	TCP	54	54784 > https [ACK] Seq=5391 Ack=2310 Win=16784 Len=0
2639	4406.576514	192.168.1.72	134.170.178.64	TCP	54	54784 > https [FIN, ACK] Seq=5391 Ack=2310 Win=16784 Len=0
2640	4406.594946	192.168.1.72	192.168.1.254	DNS	80	Standard query 0x89ed A mktplassets.xbox.com
2641	4406.753154	192.168.1.72	134.170.178.64	TCP	54	54784 > https [ACK] Seq=5392 Ack=2311 Win=16784 Len=0
2642	4407.16077	192.168.1.72	65.55.42.183	UDP	74	Source port: xbox Destination port: xbox
2643	4407.294402	192.168.1.72	65.55.42.183	UDP	1098	Source port: xbox Destination port: xbox
2644	4408.596482	192.168.1.72	192.168.1.254	DNS	80	Standard query 0x89ed A mktplassets.xbox.com
2645	4409.001026	192.168.1.72	65.55.42.36	TCP	54	[TCP Previous segment not captured] 45914 > https [ACK] Seq=8965 Ack=3321 Win=17144 Len=0
2646	4409.012802	192.168.1.72	65.55.42.36	TCP	54	45914 > https [FIN, ACK] Seq=8965 Ack=3321 Win=17144 Len=0
2647	4409.219138	192.168.1.72	65.55.42.36	TCP	54	45914 > https [ACK] Seq=8966 Ack=3322 Win=17144 Len=0

Appendix B: Successful Session Spoof Packet Dump

```
No.    Time         Source       Destination  Protocol     Length
9626   1960.673792  192.168.1.73 202.162.73.4 HTTP         1032
Info
GET /photoserver/lv2/293271101.jpg HTTP/1.1

Frame 9626: 1032 bytes on wire (8256 bits), 1032 bytes captured (8256 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Dec 10, 2013 03:21:33.736268000 New Zealand Daylight Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1386598893.736268000 seconds
    [Time delta from previous captured frame: 0.005120000 seconds]
    [Time delta from previous displayed frame: 0.005120000 seconds]
    [Time since reference or first frame: 1960.673792000 seconds]
    Frame Number: 9626
    Frame Length: 1032 bytes (8256 bits)
    Capture Length: 1032 bytes (8256 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ip:tcp:http]
    [Number of per-protocol-data: 1]
    [Hypertext Transfer Protocol, key 0]
    [Coloring Rule Name: HTTP]
    [Coloring Rule String: http || tcp.port == 80]
Ethernet II, Src: Microsof_f9:eb:79 (7c:ed:8d:f9:eb:79), Dst: 2wire_c2:e1:d9
(00:22:a4:c2:e1:d9)
    Destination: 2wire_c2:e1:d9 (00:22:a4:c2:e1:d9)
        Address: 2wire_c2:e1:d9 (00:22:a4:c2:e1:d9)
        .... ..0. .... .... .... .... = LG bit: Globally unique address
(factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: Microsof_f9:eb:79 (7c:ed:8d:f9:eb:79)
        Address: Microsof_f9:eb:79 (7c:ed:8d:f9:eb:79)
```

```
        .... ..0. .... .... .... .... = LG bit: Globally unique address
(factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IP (0x0800)
Internet Protocol Version 4, Src: 192.168.1.73 (192.168.1.73), Dst:
202.162.73.4 (202.162.73.4)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-
ECT (Not ECN-Capable Transport))
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable
Transport) (0x00)
    Total Length: 1018
    Identification: 0x53fb (21499)
    Flags: 0x00
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x4d6b [correct]
        [Good: True]
        [Bad: False]
    Source: 192.168.1.73 (192.168.1.73)
    Destination: 202.162.73.4 (202.162.73.4)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Transmission Control Protocol, Src Port: brvread (1054), Dst Port: http (80),
Seq: 25472, Ack: 123401, Len: 978
    Source port: brvread (1054)
    Destination port: http (80)
    [Stream index: 594]
    Sequence number: 25472    (relative sequence number)
    [Next sequence number: 26450    (relative sequence number)]
    Acknowledgment number: 123401    (relative ack number)
```

```
    Header length: 20 bytes
    Flags: 0x018 (PSH, ACK)
        000. .... .... = Reserved: Not set
        ...0 .... .... = Nonce: Not set
        .... 0... .... = Congestion Window Reduced (CWR): Not set
        .... .0.. .... = ECN-Echo: Not set
        .... ..0. .... = Urgent: Not set
        .... ...1 .... = Acknowledgment: Set
        .... .... 1... = Push: Set
        .... .... .0.. = Reset: Not set
        .... .... ..0. = Syn: Not set
        .... .... ...0 = Fin: Not set
    Window size value: 17212
    [Calculated window size: 17212]
    [Window size scaling factor: 1]
    Checksum: 0x47ea [validation disabled]
        [Good Checksum: False]
        [Bad Checksum: False]
Hypertext Transfer Protocol
    GET /photoserver/lv2/293271101.jpg HTTP/1.1\r\n
        [Expert Info (Chat/Sequence): GET /photoserver/lv2/293271101.jpg
HTTP/1.1\r\n]
            [Message: GET /photoserver/lv2/293271101.jpg HTTP/1.1\r\n]
            [Severity level: Chat]
            [Group: Sequence]
        Request Method: GET
        Request URI: /photoserver/lv2/293271101.jpg
        Request Version: HTTP/1.1
    Accept: image/png, image/svg+xml, image/*;q=0.8, */*;q=0.5\r\n
    Referer: http://www.trademe.co.nz/antiques-collectables\r\n
    Accept-Language: en-NZ\r\n
    User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1;
Trident/5.0; Xbox)\r\n
    UA-CPU: PPC\r\n
    Accept-Encoding: gzip, deflate\r\n
    Host: images.trademe.co.nz\r\n
    Connection: Keep-Alive\r\n
```

```
    [truncated] Cookie: session=%7bD4F0372C-C9AE-49C1-9B6A-5D59FA0DDCD4%7d;
TS01ae6471=018feaa5bc0986f653dcaeaa423b48e0bb77b73f3440738ae65c45fbb9bbd7b8885
cede580a80071849b2158349a4a4bce1126c8625845f6cb5e8f1eb55bfc8df45381e3e7;
__utma=266029386
    \r\n
    [Full request URI:
http://images.trademe.co.nz/photoserver/lv2/293271101.jpg]
    [HTTP request 18/18]
    [Prev request in frame: 9620]
```