Modelling and simulation of a new cloud computing platform based on the SPEEDOS operating system

School of Electrical Engineering and Computer Science

A thesis submitted to the

University of Newcastle, NSW, Australia

For the degree of Doctor of Philosophy in Computer Science

Ala' Said Mohammad Mughaid

June 2018

THE UNIVERSITY OF
NEWCASTLE
AUSTRALIA

Abstract

Today's society is data-driven. Collecting data from people, actions, algorithms and the web has resulted in large data stores, and accommodating all these data has become a major challenge. 'Big data' tends to grow exponentially each year [1]. To handle these increasing data sizes, the concepts of shared computing, shared memory, and remote storage and access to resources have been developed. Systems such as grid computing systems, where the infrastructure combines computer resources and storage from different locations to reach a common objective; utility computing systems, a business model in which computational resources and demand are packaged as a metered service similar to electricity and the public switched telephone network; and distributed systems, which consist of either physically distributed institutions or logically related projects/groups, are examples of such concepts.

Cloud computing is a relatively recent abstraction, providing functionalities such as computation, and the sharing and storage of data for the users of computer networks. Cloud computing is attracting massive global investment [2] because of the services that it provides. However, security remains one of the top concerns for organisations and customers using cloud computing environments [3]. In fact, some security issues in cloud computing were inherited from previous computing systems, but the others were created because of its unique characteristics and architecture. Conventional security mechanisms are not sufficient to mitigate the threats in cloud systems, and new techniques are needed.

This research presents a new platform for secure cloud computing. The platform allows cloud service providers to host their clients' data in a secure environment and allows them

to operate on the services in a secure manner for transactions. The platform was designed to make the operations relatively secure and safe using a robust structure by building a general software-structuring framework to implement the cloud software resources.

The new platform provides new mechanisms that authorise only legitimate users to access data. The access to data is handled by a third-party service that checks on all of the requests by using the user's ID and authentication/authorisation details. All of the users' details and sensitive data are encrypted during transit and storage. The platform was implemented, evaluated and compared in terms its effectiveness to the existing cloud platforms over a set of criteria. The results showed that this platform worked as expected and fairly quickly as compared to the other security platforms, and provided strong security against an intruder's actions.

Keywords: security, cryptography, cloud computing, architectural model

Email: c3193420@uon.edu.au

Statement of originality

I hereby certify that the work embodied in the thesis is my own work, conducted under normal supervision. The thesis contains no material which has been accepted, or is being examined, for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. I give consent to the final version of my thesis being made available worldwide when deposited in the University's Digital Repository, subject to the provisions of the *Copyright Act 1968* and any approved embargo.

(Ala' Said Mughaid)

Acknowledgements

I would like to express my sincere gratitude to Associate Professor Frans Henskens as well as my sponsor The Hashemite University in Jordan, who made it possible for me to conduct this research on *Modelling and simulation of a new cloud platform based on the SPEEDOS operating system*. Owing to their help, I conducted extensive research and learnt a number of new things. Working under Professor Frans afforded me many benefits, including access to his wide knowledge. He provided and maintained an academic environment throughout my thesis and made this entire experience interesting and worthwhile for me. He is a great mentor, and it was an honour to conduct my research with him.

I would also like to thank my other supervisors, Dr David Paul and Dr Mark Wallis, for their belief in me and for their help throughout my study. This work would not have been possible without the time and support they provided me. I look forward to working with them long into the future. Thank you also to Right Style Editing for their copy-editing work on the final document.

To my daughter, Marlin, for the many times I have been away from you or not in the mood to spend time with you; thank you for understanding the work I've been doing and thanks for the hugs I receive every time you see me locked in my office. Being a PhD student is tiring, but you are the best reason pushing me to continue to achieve my goal. Finally, I am grateful to my wife and my parents, who have provided with moral and emotional support in my life and strength to complete this work. I am also grateful to my other family members and friends who have supported me along the way.

Table of Contents

List of Figures:

List of Tables:

Chapter 1

1.0 Introduction

In today's digital world, people are becoming increasingly adept at using convenient technology to make their work easier and to provide prompt advice and services to clients. Laptop computers, remote access to firm servers and wireless networking are just some of the expedient ways that lawyers, for example, can connect to the Internet and their own offices from around the world.

However, how secure is confidential client information when people are not actually in their offices using a desktop computer, or when they are using a laptop computer on a network set up by a network administrator? When people take advantage of networking, to use cloud computing services or to access e-mail or client documents, they may unwittingly risk the loss and/or disclosure of sensitive data and confidential client information.

Organisations need to pay more attention to data security and use protection technologies to help secure their sensitive data and confidential client information. Weak security systems may lead to data breaches and theft, resulting in significant material losses and loss of user confidence in the service provider.

1.1 Motivation

Despite the fact that cloud computing systems, which provide large-scale computer resources with minimal effort for online customers, have many merits, such as flexibility and scalability [4], [5], [6], security remains one of the top concerns for organisations and

customers in cloud computing environments [3]. Some security issues were actually inherited from previous computing systems, and the other cloud security issues were created by the unique characteristics and architecture of cloud computing.

Conventional computer systems may be desirable, but conventional security mechanisms are not sufficient against unauthorised access in cloud systems [4], and new security techniques are needed, as will be explained in Chapter 3. The motivation of this work is to build a new cloud platform using a more secure operating system: a platform that provides cloud services while reducing the security concerns and risks in cloud computing, and still allowing the overall system to process at an acceptable level.

1.2 Research objectives

This thesis proposes a radical solution to some security breaches in cloud computing. It introduces a framework that takes into consideration the effectiveness and performance of providing cloud services and discusses the development of a framework that provides an authentication/authorisation system that addresses clients' identification and data protection needs. A prototype of the proposed framework was implemented to provide a proof of concept and a base for the evaluation of the framework.

The objectives of this work include the following:

- detailed analyses of the existing cloud computing systems and conventional computer systems that host cloud software,
- analysis of the security and networking requirements of cloud systems and an evaluation of their security,

- study of a trusted operating system that was built with the aim to resolve security problems found in other operating systems,
- proposal of a unique general structure to build cloud software applications that will reduce/prevent unauthorised access from hackers and enhance cloud data protection,
- creation of a new architecture model that mitigates cloud computing threats and reduces security concerns, and
- evaluation of the research including suggested directions for future research.

1.3 Problem definition

A number of techniques and solutions have been proposed and implemented to combat security breaches on various cloud computing activities. However, breaking into private data is a concern in cloud computing [7]. It is necessary to adopt and leverage a new approach that succeeds in protecting data to determine the following:

Problem definition: Can the new authentication/authorisation framework guarantee that only authorised users access the cloud service's data?

1.4 Research questions

This research aims to address the following questions:

- Is it possible to develop a cloud environment that provides useful services while ensuring that all of the data and communications are safe from unauthorised access?
- How can a new platform prevent/reduce the most concerning attacks that threaten the existing cloud systems, while ensuring that the attackers do not have access to new weaknesses that could leave the system vulnerable?

- Could such a system perform at an acceptable rate, or would it be too slow to be practical?

1.5 Research methodology

The project started with a study of the existing cloud computing security issues. The causes of the security issues were analysed and the current solutions examined. Based on the information collected, the modelling and framework design were implemented. This framework models cloud systems and provides a definition of what it means for a system to be a part of a cloud. This will stand as a standardised model to allow the evaluation of our secure cloud platform.

The study results showed that the fundamental solution to building a secure cloud platform is to build it on top of a secure environment. The solution requires working on an appropriate environment and analysing its security features, and then considering the possibility of building a cloud computing system using the proposed model on top of this environment.

Lastly, in this required study, a prototype was designed and built on the basis of the concepts implemented in the secure operating system SPEEDOS [8]. SPEEDOS is an operating system that was built from the ground up to guarantee security [8] by using techniques to secure memory organisation and access to data, as will be explained in Chapter 3. It includes novel mechanisms to protect data that are considerably different from those in the current conventional systems. Hence, moving the cloud infrastructure from the current conventional computer systems to SPEEDOS could considerably increase the security of the cloud services.

1.6 Thesis structure

This thesis has been divided into seven chapters. This first chapter introduces the main contributions and motivations of the research. The remaining chapters are organised as follows:

Chapter 2 provides a background on computer networks and security. In particular, this chapter gives an overview of the types of threats and attacks around networks, including details of how each threat can be exploited. This chapter proposes general solutions and protections that can be used to make networks more secure. This chapter also defines the concepts and characteristics of cloud computing, examining cloud computing middleware and what each service provides. Different types of cloud computing, and how they can be used, are introduced. Finally, this chapter presents the advantages and disadvantages of cloud computing, specifically in relation to security in cloud computing.

Chapter 3 provides a background on conventional operating systems and their current security vulnerabilities that adversely affect cloud computing systems. This chapter presents and analyses some existing frameworks that attempt to solve the current system vulnerabilities. The chapter introduces the SPEEDOS operating system and kernel features, including a description of the memory structure and how modules are represented. Finally, this chapter analyses and presents what makes SPEEDOS a more secure system for hosting cloud services.

Chapter 4 provides a logical overview of the new platform's design. This includes how software applications and services are designed, how communication between clients and the server is implemented and what sort of encryption mechanisms are used to secure the

data in transit and at rest. This chapter presents a comparison between the proposed platforms and the existing frameworks.

Chapter 5 explains how the platform was implemented, particularly what steps were used to leverage the SPEEDOS features into the new platform. It explains the technologies used to implement the prototype and explains how clients communicate with the platform.

Chapter 6 describes experiments conducted to validate the proposed platform. This chapter discusses the security of the proposed platform and compares it against other frameworks. The overall performance of the platform is tested to prove its effectiveness, focusing on how the proposed platform acts against some common cloud threats and attacks. This chapter demonstrates the advantages of using a SPEEDOS system for cloud users.

Chapter 7 concludes the project and discusses future work. The research as a whole is evaluated and directions for future work are identified.

1.7 Contributions

This work has mainly contributed to finding a solution to the problem of unauthorised access to data in the cloud computing environment and reducing the effects of the attempts to gain unwanted access.

In this work, the following contributions are made:

1. Study and analysis of SPEEDOS operating system: Study of the SPEEDOS operating system to support cloud computing services. A critical review of the issues and benefits related to the adoption of cloud computing.

2. Study and development of a cloud computing framework: An investigation into the factors which influence organisational decision making for cloud adoption in technologically developing environments. A critical review of the existing frameworks and models which support cloud computing services.
3. Modelling and design of a new trusted cloud platform: Design and implementation of an authentication/authorisation cloud platform for extremely high security demands and confidentiality of the online cloud services using the implemented security technology system. Provides an efficient protection mechanism based on the capabilities and the encryption technology to protect data in transit and at rest.
4. Overview and validation of the proposed cloud platform: Overview of the platform presented and the results of a validation experiment.

Chapter 2
Computer network and cloud systems

2.0 Computer networks

A computer network can be characterised as an arrangement of nodes that are connected and associated together for the purpose of sharing assets in an anticipated and controllable manner. These nodes are associated by using an arrangement of the sets of rules known as protocols. There are many types of network systems that empower the sharing of assets, for example:

- Local Area Network (LAN): A group of nodes that are connected through a wireless link or a communication line to share resources. Typically, the nodes are connected to the server within a distinct geographical area such as a library.
- Wide Area Network (WAN): Group of nodes that are extended over a large geographical area. Organisations and governments use WANs to relay data to others and to carry out their daily function irrespective of location.
- Internet: A network of networks that provides information and communication facilities globally.

Computer networks make it easier for firms to distribute their business globally at a low cost. The Internet particularly increases trading operations where firms are targeting billions of customers. The open nature of the Internet makes it easy for people to access services from all over the world by using their own devices. Companies have developed and implemented easy access applications to provide their services over the Internet.

Cloud computing uses computer networks to control and manage services that are available online. Cloud systems have facilitated business operations for international companies and large organisations by providing them with services, such as storage systems to host their data, and others facilities, which are explained in the next section.

2.1 Cloud computing concepts

The US National Institute of Standards and Technology (NIST)'s definition of cloud computing, which is also adopted by the European Commission (European Commission 2012) [9], is as follows: 'a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (for example, networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction' [9].

The huge combination of infrastructure in cloud computing provides large-scale and affordable benefits for users, including the ability to replace their own IT infrastructure with large pools of computational and storage resources [10]. Cloud computing has the ability to handle a large load of information from different users in a capable manner and to vary the usage of computational resources on the basis of this load. Any enterprise using cloud services can dynamically scale to any number of computing resources on the basis of traffic spikes and the other changes in the environment.

Cloud computing can be divided into layers. The layers range from the physical hardware layer at the bottom to the virtualised hardware layer to Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [11].

NIST has stated that the cloud model is composed of five essential characteristics (see below), three service models (see Section 2.2) and four deployment models (see Section 2.3) [12]. In its description of essential cloud characteristics, NIST looked at the requirements for providing IT services over the Internet and determined the following:

- **On-demand self-service:** No administrators or third parties are required to order or manage services for users. All that is required is a web portal and management interface. The provision of services and resources occurs on the provider side.
- **Ubiquitous network access**: Access to cloud services, both locally and globally, is via the Internet and based on normal protocols.
- **Resource pooling:** In cloud computing, the infrastructure (computing hardware and software) is built to serve multiple consumers from different locations.
- **Rapid elasticity:** Scaling up and down of computer resources in the cloud for consumers in need of additional resources, for example, the automatic provisioning of extra storage, when required. Hence, the services are available at anytime from anywhere and are not limited in quantity.
- **Service**: Cloud providers use modern charging techniques such as pay-as-you-go to charge their customers for using the services. Charges are metered on the basis of the usage and service type. For example, the charges could be based on each user account accessing the cloud services and/or the number of resources that each user consumes.

NIST's definition framework for cloud computing, with its list of essential characteristics, has now evolved into the de-facto standard for defining cloud computing [13].

2.2 Middleware of cloud computing

Cloud computing can be divided into three different models on the basis of the services provided. These service models are described below.

2.2.1 Software-as-a-Service

Software-as-a-Service (SaaS) is a software model that provides subscribed customers with access to software services that are centrally hosted. This delivery model is becoming increasingly widespread as the underlying technologies supporting web services mature and support new approaches to development. SaaS provides the following benefits to the IT industry [5]:

- easier management,
- automatic updates and patch management,
- compatibility, and
- facilitating cooperation and overall accessibility.

Furthermore, SaaS allows consumers to use programs in the cloud, where the cloud service provider manages all of the parties. Users will have guaranteed compatibility, and collaboration is facilitated because everyone uses the same software. Companies do not have to pay additional license fees, and administrators can easily add new users on the basis of the system being used. Some examples of popular SaaS applications are Gmail and Google Apps [14].

SaaS can have many advantages as it pertains to cost savings and budgeting for an organisation. A report [15] published by Microsoft Corporation emphasises that one of the

biggest benefits of deploying applications using a SaaS model is the low initial investment on software, hardware and staff [16]. Hurwitz and associates stated in their study [15] that SaaS services offer as much as 64% cost savings over four years as compared to an on-premises solution.

Any organisation can offer SaaS by providing valuable IT assets. For example, any IT company can be a SaaS provider by exposing its own products over the Internet and allowing consumers to access them.

The SaaS solution has many advantages. However, a SaaS solution also brings concerns in areas such as availability, as most applications require an Internet connection to continue working. There are also some other concerns related to the security of enterprise data, as a platform that provides applications and services to millions or even billions of users may be more attractive to attackers than smaller systems with fewer users [10].

2.2.2 Platform-as-a-Service

Platform-as-a-Service (PaaS) is another cloud model that provides consumers with a complete environment in which to develop their own programs and the option to publish them over the Internet. PaaS offers a set of components or sub-system software used to develop a fully functional product or service, such as a web application. In general, PaaS provides the stack of the operating system, middleware and/or application database, and an application programming interface (API) to use these resources.

By using the stack provided, cloud computing developers can deploy their programs without the need to install full systems on local machines. The Oracle Cloud Platform [17] is an example of a PaaS provider.

A PaaS provider is responsible for providing users with a fully managed platform and management infrastructure that deals with low-level resources. Developers can access software tools and databases to make their jobs easier. Paul Burns, President of the analysis firm Neovise LLC, believes that PaaS services will increase the movement of the existing applications to the cloud: 'You take an existing application and you make it run in a cloud environment and take advantage of some of the underlying capabilities like elasticity' [18].

PaaS provides many benefits for programmers, such as allowing globally distributed development teams to collaborate on software development projects. Services can be obtained from multiple sources. An example of a PaaS system is Google's App Engine [19]. Users can customise these tools on their own devices. Windows Azure is another PaaS provider [20]. It provides users with a complete environment to build their software by using various languages, tools and frameworks. The users can then integrate the applications into their existing IT environments [21].

2.2.3 Infrastructure-as-a-Service

Infrastructure-as-a-Service (IaaS) offers users access to fundamental computing resources. 'The capability provided to the consumer is of provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications (for example, host fire walls)' [22].

IaaS is considered to be the existing foundation of cloud computing [3]. It supports the business service model by providing an outsourced basis for the infrastructure. IaaS typically provides hardware, storage, servers, and data centre space or network components. IaaS is also known as Hardware as a Service (HaaS) [23]. IaaS is attractive for new companies because there is no need to buy equipment, hire people to manage the hardware systems, or even deal with real estate to build a warehouse to hold the equipment. However, IaaS can also be adopted by the existing companies to reduce the costs associated with these activities, for example, when the existing in-house infrastructure reaches its end of life.

In IaaS, the user has the most control and can run any operating system and applications inside virtual machines (VMs). This control diminishes in PaaS where users are only allowed to develop applications with the resources provided to them, for example, a Java software development stack. Further restrictions are improved in SaaS where the user only has control over the configuration settings in an application offered by the cloud provider [24].

IaaS platforms work with virtualisation tools that allow users to easily scale up the resources that they are using as the need arises, only paying for the resources that they use. These systems provide a metering model that charges the users on the basis of their usage of the resources at rates negotiated beforehand.

Service-level agreements (SLAs) require the providers of IaaS to ensure that the servers continue to function correctly and are adequately maintained. The providers of IaaS are required to give guarantees from the hardware level to the software management level and

to provide security for an IaaS level. Examples of IaaS providers include Amazon EC2 and S3 [25], Terremark Enterprise Cloud [26] and Rackspace Cloud [27].

IaaS provides unique benefits but has created new security risks for the IT community. For example, IaaS provides management tools that support administrative operations on VMs (for example, delete, add and update), which provides a possible avenue of attack [25].

2.3 Cloud computing classifications

The following are the four main cloud types, based on who controls the data:

- **Public cloud:** This type of cloud is managed and operated by a third party or an organisation that sells cloud services and applications through the Internet. The services are available in this cloud type through a variety of pay models, such as pay-as-you-go and fixed-price models for the users. Google and Amazon implement examples of this cloud model.
- **Private cloud:** This cloud model is managed by a private organisation that has its own management, or the cloud can be managed by a third party. A private cloud relies on the virtualisation of an organisation's existing infrastructure [28], giving the company the opportunity to obtain the benefits of the cloud while retaining control of its system. Examples of these models are Big Data and the Chargeback tool [28].
- **Community cloud:** This cloud model shares the deployed applications between different companies to serve a specific community that has common concerns. The users can run their applications internally and globally, for instance, by using high-end security products to ensure working on dedicated clouds and compliance within certain regulations.

- **Hybrid cloud:** This is a cloud model that is made by a combination of a private and a public cloud, or more than two clouds. The cloud infrastructure is connected together by standardised technologies to enable data and application portability [9]. Developers can choose the cloud infrastructure on which a given application will run. For example, when a company leverages a number of SaaS applications, it can move data between private and data centre resources.

2.4 Virtualisation in the cloud

Virtualisation is a technique that provides users with a virtualised software environment practically identical to what they would have on dedicated hardware. Each of these environments is called a VM. Figure 2.1 shows the architecture of virtual and traditional systems, where virtual systems can provide more software environments than a traditional system by using the same hardware facilities.

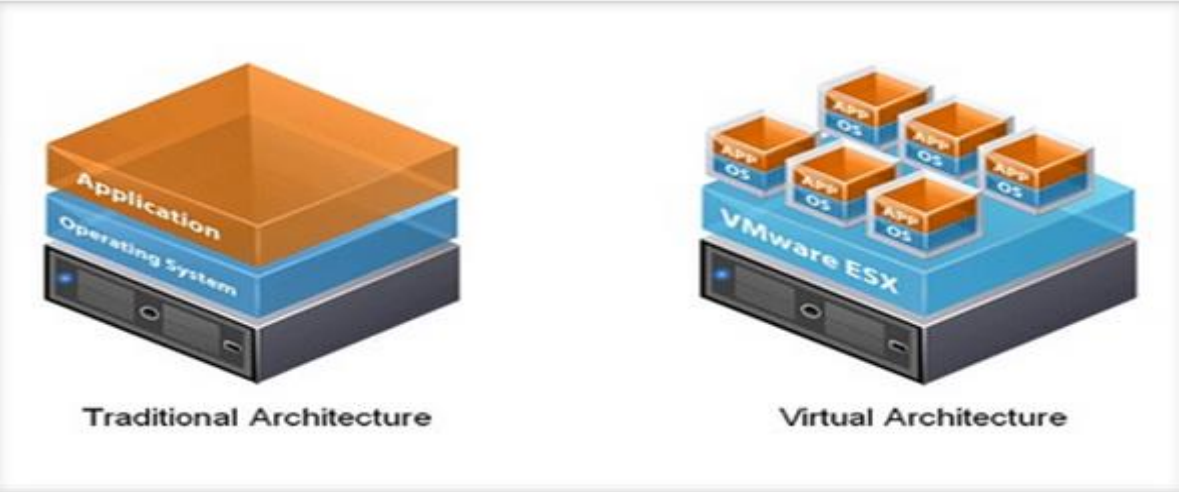


Figure 2.1: Virtual architecture and traditional architecture [80]

Virtualisation offers a set of professional management tools called a virtual machine monitor. The most common specialised software to manage VMs is a hypervisor [29].

Sabahi defines a hypervisor as ‘one of many virtualisation techniques which allow multiple operating systems, termed guests, to run concurrently on a host computer, a feature called hardware virtualisation. It is so named because it is conceptually one level higher than a supervisor’ [29].

It is usual for hypervisor layers to be used to run multiple operating system guests or VMs that are installed between the operating system and the hardware. To consider how a hypervisor virtualises hardware, we need to consider the features that hypervisors have. The most concerning features are as follows [30]:

- Base OS used: It is the operating system used to support the service.
- System updates: The system should be able to periodically apply updates on the platform to match the market and user requirements.
- System architecture: It refers to the type of architecture that the system uses, either a bare-metal or a hosted virtualisation.
- Supported virtualisation technologies: Two types of virtualisation technologies can be used by the system: (1) bare-metal virtualisation where the hypervisor has direct access to hardware resources and (2) hosted virtualisation where the hypervisor needs to be installed on the server’s operating system.
- CPU scheduling features: These refer to the way the CPU scheduler works and the functions it performs (for example, control with VM reserve, VM limit, and relative weight).
- Symmetric multi-processing (SMP) scheduling feature: It is the single scheduler system used for all CPUs.

- Memory address translation mechanism: It is the mechanism to manage the memory in the system (for example, a shadow page table is a mechanism that allows a system to run more than one type of operating system concurrently).
- Disk management features: These are the techniques used to manage the storage disks of the system.
- Network management features: These are the techniques used to manage the network of the system.
- Performance time: It refers to the time taken to perform a single job from arrival to completion for each user in a single VM in the system.
- Performance isolation provided and level of performance isolation: These refer to the mechanism used to isolate VMs from each other and to what level.

There are many different types of hypervisors, from open source to commercial. It is up to the individual administrators and enterprises to determine which type of hypervisor would best meet their service requirements, taking into consideration details such as hypervisor performance, features and price. However, most hypervisors have the same goals to provide infrastructure; only the underlying technologies vary. The operating systems that support hypervisors are mostly split between Windows and Linux products [30].

Container management tools are another type of tools that can be used in virtualisation; new virtual systems rely on facilities provided by an operating system on the underlying hardware that shares the same kernel, and do not require separate operating systems for each virtual system, as VMs do. Containers use a single storage plus smaller deltas for each layer, whereas hypervisors consume storage space for each instance [31].

VMs are normally divided into the following two major types [32]:

- System VM: It is software designed for virtualising hardware, sharing the underlying infrastructure resources among different VMs. It supports multiple operating systems for users running on the same physical machine as the underlying system. Examples include VMware, Virtual PC and XEN [33].
- Process VM: It is application software that installs on the top of an operating system/hardware combination to carry out machine instructions and system calls for a single process. Examples include Oracle's Java VM architecture and the Microsoft Common Language Infrastructure [30].

VMs have provided many benefits for the IT industry. Some of these benefits are as follows:

- VMs provide cost savings by minimising the need for physical infrastructure and consultation.
- VM environments give administrators the ability to remotely arrange backups and manage failure easily in the case of a disaster.

However, VMs can also present delivery challenges and vulnerabilities. For example, the live migration of VMs is an attractive area for many attacks such as the man-in-the-middle, denial of service and stack overflow attacks [6]. Another issue with VMs is that the performance of a VM is poorer than that of a real machine running the same software [6]. This is attributed to the fact that the VM executes commands in a common environment that may share the entire infrastructure between different consumers, which may cause a slowdown of processes when priority is given to one of the other consumers. Another

reason for the process slowdown in VMs is the need for instructions to be interpreted by the VM before their eventual execution on the real hardware; however, modern hardware mostly overcomes this limitation [6].

2.5 Cloud computing advantages and disadvantages

Cloud computing allows consumers to obtain and configure computer resources with minimal effort. Cloud computing offers an attractive environment to the IT industry which promises to reduce electricity usage and carbon emission [34]. Besides these benefits, there are numerous other benefits that cloud computing can offer, such as the following [3], [5], [35]:

- Cost saving: The partitioning of the infrastructure for providing multiple operating systems and applications plays a significant role in the cost savings due to cloud computing. Businesses have taken advantage of renting (rather than buying) their computer services and using the pay-per-use billing model facilitated by cloud computing.
- Mobility of information: A benefit of cloud systems is the ability to access data from anywhere anytime; in its simplest form, a user just needs a browser connected to the Internet.
- Ease of backup and disaster recovery: Some cloud providers offer scheduled backup (automated backup) and recovery services for user data with professional options to achieve their demands and make them comfortable in case of a disaster.
- Scalability and attractive architectures: Users can scale their cloud services or storage up or down depending on their current requirements. Devices can be added instantly to

improve performance, and clients have access to the core material of the cloud on their global devices such a PC, laptops and/or smartphones.

- Increased storage capacity: The cloud may offer the user storage space, and more storage can be provisioned at the consumer's request. This eliminates concerns about the lack of storage space and, at the same time, saves companies the need to upgrade their equipment, thus reducing the overall IT costs.
- Diversity of devices and local independence: A cloud computing service can be availed from different types of devices. The flexibility of the provided services is very attractive for cloud users to access data from anywhere anytime, for example, from their own mobile phones.

Since the inception of cloud computing systems, it has been clear [10], [36], [14], [17], [37] that cloud computing offers considerable benefits to its users. However, it is a tool that also comes with its own set of problems and shortcomings. A problematic impact of cloud computing could occur if the user wishes to change its cloud provider. It can be a huge risk and costly to transfer data from the old provider to the new provider and may not even be possible at all (for example, if the old provider has gone out of business and is no longer online). Further, it is difficult to ensure that data have been deleted from the old provider after the user has moved. There are also occasional technical difficulties that might result in downtime for cloud computing systems. Like any other system, cloud computing may face breakdowns even with the best providers; for example, Amazon Web Services has gone down several times [36]. Even if the cloud provider is operational and if the network connection to the provider is broken, no access is possible.

Another issue with cloud computing is that the underlying infrastructure of a cloud is hidden and the users have little or no control over where their data are saved. Data might be stored in countries that have restrictions and rules regarding the migration of data, which may cause issues with different privacy standards, whilst in other countries, the privacy laws or their enforcement may be comparatively lax. For example, the data stored in the USA and the EU are subject to different laws [38].

Businesses are concerned with cloud security as they might have private data and sensitive or confidential information that may not be secure in remote infrastructure. A survey [3] by Cloud Security Alliance (CSA) and Institute of Electrical and Electronic Engineers (IEEE) showed that companies across the world are willing to move to cloud computing but the security needs to be compatible with the large and growing scale of the world's data under management in cloud computing. Cloud security will be examined in more depth in Section 2.9.

2.6 Network security

Network systems have provided many advantages to organisations, enabling access to facilities and computer resources anytime from anywhere, and might be considered a technological revolution. However, organisations need to give more attention and consideration to their system security and need to guarantee that unauthorised individuals cannot access their information.

One of the most unwanted situations to occur in networks is unauthorised access. This type of access may be accomplished by an outside or inside company intruder, or both. Such access may cause considerable damage to the company's reputation by stealing its

important data, which reflects negatively on the company's commercial dealings and reduces its customer's trust in it.

A number of risks and threats exist in the operational environment of computers and networks, particularly where they can become exposed to security breaches. There could be various reasons for the vulnerability, including an incorrect installation of systems, an incorrect usage or malicious software.

In general, information sharing among two or more computers over a network may be exposed to the risk of intrusion. There are four ways of intrusion that can negatively affect the system [39]:

- Interception: An unwanted entity between the transmitter and the receiver steals the information.
- Interruption: Any unwanted entity between the conversation of two nodes stops the message and prevents it from being passed to the destination.
- Modification: An unwanted entity at the middle of conversation of two nodes changes the sender's messages, modifying their information before forwarding them to the receiver.
- Fabrication: A type of lie; here, one party is fabricated, and the other participants on the network are unaware that the messages are not from a valid participant.

2.7 Network security threats/attacks

The definition of threat can be expressed as any possible action which causes damage to a network, data, users or security goals, either in a hostile manner or accidentally. This

unwanted damage, change or security bypass in the hostile/accidental manner is known as an attack on the network.

There are two types of threats: passive and active. A passive attack is a network attack that involves getting access to a link or device and consequently, the data [40]. Active attacks involve active attempts to breach security leading to modification, redirection, blockage or destruction of data, devices or links [40].

2.7.1 Unauthorised access

A high-level term, ‘unauthorised access’, refers to multiple types of attack. The objective behind these attacks is to obtain access to resources that are not supposed to be provided to the attacker by your machine. For instance, a web server as the host should respond to anybody who makes a web page request. However, this host should not provide command shell access without being sure that the person making such a request is someone who should get it, such as a local administrator.

With this type of attack, the hacker can use many techniques, including the following:

- guessing passwords for a well-known account, for example, the administrator;
- accessing the password file and using a password-cracking program [41]; and
- using social engineering, which is often the easiest way to gain unauthorised access to the network recourses in any network; here, a hacker calls various users about a network problem, gathers information about the network and starts using this information to attack the network.

Some types of unauthorised access are discussed below.

2.7.1.1 Malicious association

Malicious association is when a cracker or hacker gains access to a company by using a wireless device via a cracking PC instead of the company access point. The cracking PC is called a ‘soft access point’ that is used by the cracker to act like the company’s access point. Various unwanted activities can occur when an attacker takes over the company access point, such as password stealing, attacks on networks and more dangerous activities, such as executing Trojans, by getting unsuspecting users to connect to the fake access point. The purpose of such an attack may not to be break into a VPN or breach security, but may be to try to take over the current clients to gain important data.

2.7.1.2 Man-in-the-middle attack

This type of attack seems to occur in the middle of a conversation between two parties in such a way that the attacker makes independent connections with the victims, to have them send messages via the attacker while believing they are using a private connection. The flow of information on the real network becomes available to the hacker. This attack and the type of access enable hackers to ‘sniff’ traffic over the network, and this can occur because of security faults. The wireless tools LANjack and AirJack [42] help hackers to launch man-in-the-middle attacks by automating various steps required to perform them.

2.7.1.3 Injections

Injection attacks refer to a wide class of attack vectors that enable an attacker to provide a trusted input to a program, which gets processed by an interpreter as part of a command or query which alters the course of the execution of this program. The following four processes are required for such an attack:

- The technology that the system is running should be identified.
- Programming languages and hardware processing issues are major instigators of an injection attack. An investigation is required to secure the network by trying common injection attacks. Various avenues can be used by the attacker to gain information, such as the footers of web pages, error on pages, viewing of the source code of the page and the use of various tools, including Nessus, Nmap and THC-Amap [43].
- All possible user inputs should be identified. There may be the identification of the usual and expected inputs, which may include an HTML page. Web applications can be accessed in many ways by an attacker. Hidden HTML code can be accessed or changed, which may be sent as an input to the system; cookies can be used by the attacker; and backend languages and scripts such as Java script can be accessed. All of the data input within HTTP or POST are considered the user input.
- Identification and detention of this type of information, which is risky for the network, should take place in a timely manner to prevent an attack. The user input becomes risky because of the applications that provide detailed information about the user input.

2.7.1.4 Eavesdropping

Secretly listening and stealing private information during conversations that are believed to be confidential, and for which such listening is unauthorised, is termed eavesdropping. This process compromises the confidentiality of the information, but its integrity remains intact.

This process can occur on wireless networks as well as wired networks. The execution of eavesdropping on a wired network is difficult, as it requires an attacker to physically connect to the network over which the data are being transmitted.

Eavesdropping can be enabled by compromisers at a number of positions in the network. A vulnerability in any node on the entire network may allow the attacker to penetrate through the network to exploit the entire network and probe through all of the packets being transmitted on this network, even if the addresses of the packets are different from the address of the attacker. In eavesdropping, all that is needed by the attacker is to be in the wireless network coverage area or have software that allows him/her to eavesdrop.

2.7.1.5 Phishing

Phishing is an attack intended to persuade the victims to give away their information such as credentials or account information. Phishing is characterised as a strategy used to misleadingly gain data, including an individual's information, managing an account and credit data, passwords and other record numbers from Internet clients, by utilising messages and sites intended to resemble a legitimate business, monetary organisation or government office. Phishing is a developing trick; with each attack comes another technique to help the phisher bait more victims. It is difficult to identify and avoid phishing attacks as they, from time to time, have qualities similar to those of legitimate requests.

2.7.1.6 Accidental association

There are a number of methods to gain unauthorised access to a company wireless and/or wired network. 'Accidental association' is one of these methods. When a user switches on a computer system and attaches it to a wireless access point of a neighbouring company's overlapping network, the user may be unaware that he/she is connected to the wrong network. However, it will cause a security breach and the proprietary information of the

company may be at risk. Links can exist, and hooking up of a laptop to a wired network can create the same issue.

2.7.2 Denial-of-service attack

An attacker tries to prevent real users from accessing information or services in a denial-of-service (DoS) attack. The attacker directly targets the real user's computer and/or network connection, or the website's computers and network that the users are trying to access. An attack of this type can result in the denied access of email addresses, websites, online accounts (for example, banking) or other services that are supposed to be provided by the affected computer. The attacker sends a large number of spurious messages which the receiver takes time to process and, while these messages are processed, valid messages cannot get through. A distributed denial-of-service attack (DDoS) is designed by attackers to lessen a server's ability to provide services. This is achieved by flooding the community with purported (but fake) clients' requests stopping the device from servicing legitimate customers (more details in sections 2.9).

The most common DoS attacks are described below.

2.7.2.1 Slowloris

Slowloris is a type of DoS attack software that has the potential to take down a web server by using a single computer [44]. The elegant nature of this attack means that, although it is simple, it requires only minimal bandwidth for execution and only the target web server can be affected, without any, or at least almost no damage, to other services and ports.

Slowloris attempts to retain many open connections to the target web server for as long as possible. It achieves this task by opening connections related to the target web server and sending a partial request. Periodically, subsequent HTTP headers will be sent, and it will add them, but the request will never be completed. Connections to the affected servers will remain, filling their maximum concurrent connection pool; eventually, additional connection requests by legitimate clients will result in service denials.

2.7.2.2 ICMP (ping) flood

An ICMP flood overloads the target resource with the ICMP echo request (ping) packets, without waiting for replies, normally sending packets as quickly as possible. Both the outgoing and the incoming bandwidth can be consumed by this type of attack, as the target's servers will try to respond with ICMP echo reply packets; this will result in a significant slowdown of the system.

2.7.2.3 SYN flood

A flood scenario in the case of a synchronisation (SYN) packet is where multiple SYN requests are sent by the requester normally, but they either result in no response to the host's synchronisation-acknowledgment or send the SYN requests from a spoofed IP address. In either case, with each of the requests, the host system waits until it receives an acknowledgement. Scarce resources result in no new connections, finally leading to the denial of service.

2.8 General principles to protect networks

To maintain the security of a company's information, the company must take precautions to prevent intruder(s) from attacking its privacy and prevent any penetration from taking place. A thorough security policy should cover the topics discussed below.

2.8.1 Firewalls

With respect to network security, a standout amongst the most usable approaches to assist such security is the firewall. Firewalls are used with a specific end goal of providing a partition between an association's intranet and the Internet. A firewall is just a gathering of parts that on the whole, shape a hindrance between two networks.

A firewall is a framework intended to prevent unapproved access to, or from, a private system. Firewalls can be executed in both equipment and software, or a blend of both. Firewalls may be used to keep unapproved Internet clients from accessing private systems associated with the Internet, particularly intranets. All of the messages entering or leaving the intranet go through the firewall, which examines each message and blocks those that do not meet the predefined security criteria.

Effective security planning assists firewalls to work more effectively, and a well-designed security policy enables it to be executed efficiently. Its effectiveness improves with connections to antivirus software, detection systems and various other tools.

2.8.2 Intrusion detection systems

An intrusion detection system (IDS) is a software or hardware device that monitors networked systems for suspicious actions and alerts the framework or system administrator

if any unusual activity is detected. The IDS may occasionally react to anomalous or malicious traffic by taking action automatically, for example, blocking the client or source IP address from getting to the system.

IDSs arrive in an assortment of ‘flavours’ and approach the objective of distinguishing suspicious movement in various ways. There are network-based (NIDS) and host-based (HIDS) intrusion detection systems. There are IDSs that identify particular signatures of known threats; for example, antivirus programming software distinguishes and secures against malware. Further, there are IDSs that contrast activity designs against a baseline and search for anomalies.

IDSs come in a variety of flavours and detect suspicious traffic in different ways.

2.8.2.1 NIDS

A network-based intrusion detection system, NIDS, detects unwanted activity, such as a DoS attack, scanning of ports or hacking into a computer by examining the traffic on a network. It attempts to identify any malicious or unwanted packets of data when the entire packet enters the network by detecting the signature of unwanted patterns that do not ‘follow the rules’.

2.8.2.2 HIDS

A host-based intrusion detection system, HIDS, requires individual hosts and devices connected to the network where IDSs execute. The incoming and the outgoing traffic from the devices is monitored by HIDS, which alerts the user about any unwanted activity that is

detected. It is an IDS that monitors and analyses the internals of a computing system rather than the network packets on its external interfaces (as an NIDS would).

2.8.2.3 Signature-based IDS

A signature-based IDS screens packets with respect to the network and compares them against a database of signatures or properties from known malicious threats. This is similar to the way most antivirus programs identify malware. The issue, however, is that there will be a lag between another risk being discovered in the wild and the signature for identifying that threat being connected to your IDS.

2.8.3 Cryptography

Cryptography is the method of writing and converting data into a format that is unreadable for unauthorised users, where only the intended recipient can reverse it into a readable format to obtain the original message. It is a tool used to hide data in both the transmission process and when stored. It is a technique that can provide virtually unbreakable protection to sensitive information.

Cryptography requires the following two main components:

1. a cipher, which means that the encryption process used to hide data during transition or/and storage; and
2. keys required to perform the encryption/decryption processes. They are the core part of cryptography operations used to transform the text into cipher.

Cryptography can be divided into two main categories as discussed below.

2.8.3.1 Symmetric encryption

Symmetric encryption is one of the most familiar cryptography techniques used to protect data. It applies a secret key to the original text of a message to change the content in a way that cannot be read by any others except those who have the key to encrypt/decrypt it (see Figure 2.2). Encryption is the process of changing data into the cipher text form that cannot be understood by anyone except the authorised parties. Decryption is the process of decoding the encrypted text and converting it back to the original text. The key could be a number, a word or a string of random letters. That key is shared between two or more parties and can be used to encrypt private information. Sharing the key is a drawback for this encryption mechanism, because anybody who knows the key can read and create messages.

Figure 2.2: Symmetric encryption [45]

Advanced encryption standard (AES) [46] is an example of a symmetric encryption algorithm. AES is a symmetric block cipher chosen by the US government to protect classified information and is implemented in software and hardware throughout the world to encrypt sensitive data. In present-day cryptography, AES is widely adopted and supported in both hardware and software. To date, no practical cryptanalytic attacks against AES have been discovered. Additionally, AES has a built-in flexibility of key length,

which allows a degree of ‘future-proofing’ against progress in the ability to perform exhaustive key searches.

2.8.3.2 Asymmetric encryption

Asymmetric encryption applies two keys to encrypt the information (see Figure 2.3). A public key is published to all the users of the system, and the private key is made available only for a specific receiver customer who will receive the information. The public key can be represented in a short sequence of bytes that simplifies it during the cryptography mechanism. The short sequence is called a fingerprint and can be certified to represent the public key. A message that has been encrypted using the public key can only be decrypted by applying the matching private key to the encryption algorithm. Any message that is encrypted using the private key can only be decrypted by using the matching public key.

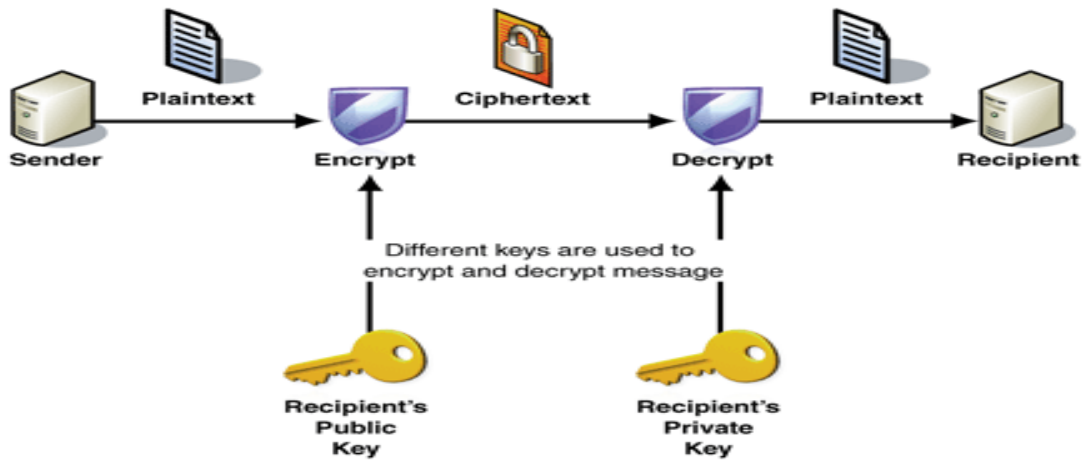


Figure 2.3: Asymmetric encryption [45]

RSA may be considered one of the most secure methods of data transmission and is named after its designers: Rivest, Shamir and Adleman [47]. RSA utilises public and private keys that are based on a pair of large random prime numbers. Its security depends on the difficulty of factorising large numbers. Public keys can be stored in any location without

compromising the security of the system, although there is an issue in determining that a public key does in fact belong to a particular person.

2.8.3.3 Hash functions

A hash function is one of the secure mechanisms used to produce a fixed-size alphanumeric string as an output without changing the original message content [48]. The string is called a hash value or message digest, as it contains a string of digits generated by a one-way function that verifies whether the input matches the hash value. It is nearly impossible to retrieve the original data from only the hash value, and unfeasible to fabricate a data block that matches a given hash value. A hash function can be used to ensure that the information sent from the sender is identical to the information received by the recipient.

SHA-2 functions

SHA-2 is a family of cryptographic hash functions designed by the US National Security Agency (NSA) [49]. The office provided a number of test vectors to check the accuracy of the capacity usage. SHA-2 encodes a byte cluster or rundown of whole numbers, or the characters of a string (utilising the default framework encoding) into a SHA-2 process as a hex string.

An SHA-1 hash produces a hash value which is typically rendered as a 40-digit-long hexadecimal number. It is no longer considered secure against well-funded opponents. SHA-2 incorporates considerable changes from its antecedent, SHA-1. The SHA-2 family consists of six hash capacities with digests (hash esteems) that are 224, 256, 384 or 512 bits: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256.

SHA-256 and SHA-512 are novel hash capacities registered with 32-bit and 64-bit words, separately. They utilise diverse move sums and added substance constants, yet their structures are generally essentially indistinguishable, varying just in the number of rounds.

- SHA-2 is one of the strongest calculations that has no known attacks for it as indicated by NSA.
- The message is broken into 1024-bit chunks.
- The introductory hash esteems and round constants are stretched out to 64 bits.
- There are 80 adjusts and not 64.
- The message plan exhibit has 80 64-bit words and not 64 32-bit words.
- The circle to broaden the message plan exhibit is from 16 to 79 and not from 16 to 63
- The round constants depend on the initial 80 primes: 2...409.
- The word measure utilised for computations is 64 bits in length.
- The attached length of the message (before pre-handling), in bits, is a 128-piece huge endian number, and the move and pivot sums utilised are extraordinary.

2.8.4 Security solution frameworks

In today's world, the volume of information is increasing every second and threats proceed to change and develop weakening security systems to respond to attacks. Many solution frameworks have been implemented to empower security and to protect people's data. The security architectures that help to enhance the defences are as follows:

2.8.4.1 Spring security core

Spring security core is a framework that provides a security base to build secure applications. It focuses on authorisation/authentication to secure data [50]. It attempts to allow basic and advanced usage, securing URL, views and methods.

Spring security supports access list control (ACL) to authorise the users of applications. An ACL is a list that informs the system which user can exercise which privileges over a particular system object. The list has an entry for each system user along with the corresponding access privileges. The privileges allowed include read, write and execute. A read access right allows the items stored in the computer memory to be read. A write access right allows the items stored in the memory to be changed. An execute access right allows the locations of the memory to be fetched as instructions.

2.8.4.2 Security assertion markup language

Security assertion markup language (SAML) is another framework that exchanges authorisation and authentication between parties. It was developed by OASIS Security Services Technical Committee to define roles amongst applications. Typically, the main roles of the framework are the principal, which is typically a user, the identity provider and the service provider [51].

SAML works by the users requesting authentication from an identity provider through an interface module. Then, the identity provider, which has the certificate fingerprint of the service provider, retrieves the user authentication details and validates it by using the certificate fingerprint. The identity of the user is established, and the user is provided with app access.

2.8.4.3 Kerberos

Kerberos is a computer network authentication protocol that provides a mechanism for authentication between clients and a server, or between servers. It provides tickets to permit nodes communicating over an unsecure network to prove the nodes' identities to each other in a secure manner [52]. The Kerberos authentication protocol builds on symmetric key cryptography and may optionally use public-key cryptography during certain phases of authentication. It requires a third party to check the clients' requests and the issuing of tickets to permit the access on the basis of their authorisation details.

A client verifies itself to the authentication server (AS), which advances the username to a key distribution centre (KDC). The KDC issues a ticket-granting ticket (TGT), which is time stamped, and encodes it using the TGS's mystery key and returns the scrambled outcome to the client's workstation. This is done rarely, normally at client login; the TGT expires sooner or later in spite of the fact that it might be re-established by the customer's session manager while he/she is signed in. At the point when the customer needs to speak with another node ('principal' in Kerberos parlance), to some service on this node, the customer sends the TGT to the ticket-granting service (TGS), which normally shares the same host as the KDC. Each service must be enrolled at TGT with a service principal name (SPN). The customer uses the SPN to ask for access to this service. Subsequent to confirming that the TGT is legitimate and that the client is allowed to access the requested service, the TGS issues the ticket and the session keys to the customer. The customer at this point sends the tickets to the service server (SS) alongside its service ask.

2.9 Cloud computing security

Security is a major concern in cloud computing [53] because resources are held by a third party with little oversight from any individual users or group. Services such as the global sharing of computing resources offered by the cloud are attractive to attackers from around the world [36], [21].

Cloud users can upload and download their data from anywhere anytime. The data might include private information, such as credit card details or medical information, or political data for large countries. Countries might feel that the cloud is the safest place to save their data because it has many options for security, recovery and backup. However, it is still a system that can be broken into, as evidenced by examples that will be discussed later in this section, which refer to cloud systems that have already been attacked.

Security and privacy are issues that remain major concerns and prevent business deployment to cloud systems. With online data storage services, a new method of breaking the privacy of data has become available. For example, countries may use some services offered by cloud providers to obtain information about others. Recent reports [54] show that some governments have been extracting files from the Internet to obtain information about other countries. Therefore, online data storage may put the data at risk, possibly without any guarantee of their safety. NCC Group showed in its 2014 research [55] that 72% of respondents are afraid of using cloud computing in the future because of concerns over the data not being backed up and issues of disaster recovery and of privacy and security. The research surveyed CIOs from financial services firms with more than 1,000 employees.

Further, the increasing number of incidents of malicious attacks was a reason for such a high percentage in the cyber security study in 2011 [56]. For example, a large number of Twitter users leaked their own private data to a French social engineer in 2009 [57].

Google's transparency report shows that the company receives requests from governments and courts from around the world to gain access to user data, or to remove content that overreaches the government roles [54]. It also shows the numbers of these requests over six months for each government. One of the largest percentages of requests concerns privacy and the security of others.

Cloud computing faces challenges to ensure that the security of applications and the infrastructure are solid, while still completing the tasks required by the users. Some of the challenges, from the eyes of the cloud consumers [58], are attributed to the fact that the critical data of different consumers are stored in the same data centre. Furthermore, companies participating in the cloud may lose control over data because these data are managed by third parties. Another factor may be that not all providers offer the same security guarantees in their agreements with customers.

CSA has recently released research that focuses on threats to the cloud [59]. Based on this study, it provided the needed context to assist organisations in making educated risk management decisions regarding their cloud adoption strategies. The seven main threats listed through this research are as follows [59]:

- Abuse and nefarious use of cloud computing: Some cloud systems offer customers unlimited free trial registrations to their services, or anybody with a valid credit card may be able to register and use the cloud services. By abusing the relative anonymity

behind these registrations, attackers have been able to conduct their activities with relative immunity.

- Insecure APIs: The security and vulnerability of cloud services are dependent upon the interfaces that the cloud providers expose, to allow their customers to interact with their services. Additionally, third parties and organisations may use these interfaces to build their own interfaces to offer value-added services to their customers. The new interface layer adds more complexity to the cloud providers' security and might increase the risks as it typically requires the organisations to release their confidentiality to third parties to enable their agency.
- Malicious insiders: The level of access granted to an adversary because of the lack of hiring standards could give the chance to such an adversary to gain confidential data or take control over the cloud services and perhaps cause a financial impact and/or large amounts of damage and losses.
- Shared technology vulnerabilities: The lack of strong isolation between architecture tenants that use the same infrastructure could enable guest operating systems to gain a certain level of control over the underlying platform.
- Data loss/leakage: Compromising data could be achieved because of a few reasons, such as changing or deleting records without having a backup for the original contents or unauthorised access to sensitive data from different parties. This has increased recently with the use of cloud computing and become more dangerous [59]. This could be attributed to the operational characteristics or the architectural environment of the cloud.

- Account, service and traffic hijacking: Attackers may use their common methods to hijack services and reuse the credentials and passwords to achieve further steps such as controlling service availabilities. If an attacker gains access to credential data that contains a cloud service supplier, then the attacker might use the supplier service instances to become a base for him/her. Furthermore, the attacker may leverage the power of supplier reputation to launch subsequent attacks.
- Unknown risk profile: Cloud computing providing services supply hardware, software, security and even maintenance which allow companies to focus on their business growth. However, security details and internal compliance are not known for the cloud customers, and they may have an unknown risk profile. The obscurity in cloud security might result in unknown exposure.

These threats should be taken into consideration by organisations that are planning to move into the cloud. By studying the environment of the threats, organisations can select the appropriate security controls, allowing them to reduce the risks of potential attacks.

With all the challenges and threats around cloud computing, security incidents have continued to increase [59]. Over the last ten years, there have been many security issues related to cloud computing.

A few examples of the relevant recent exploits are described below.

The Twitter social network was attacked by French attackers by using stolen administrator privileges. Twitter internal business documents and personal information were leaked to the technological website ‘Tech Crunch’ [60]. A 25-year-old successfully used social engineering to illegally gain access to the former US President Barack Obama’s Twitter

account, as well as some other users' accounts, including the accounts of well-known entertainers Britney Spears, Ashton Kutcher and Lily Allen. The attacker put 13 screenshots of Twitter's admin panel under the handle of 'Hacker Croll' and commented that 'The images were taken from the admin area that was secured with hatches' [61].

Another data theft was inflicted on Vodafone, a cloud provider and one of the biggest mobile phone and broadband providers [62]. Over two million German Vodafone customers' names, addresses, dates of birth and bank details were compromised by this theft. The attack was perpetrated by a malicious ex-administrator who used to work for the Vodafone Company [63].

White Lodging Service Corp., the management firm for many hotels including Marriott, Holiday Inn and Sheraton, said it detected a breach in 2013 in the guests' credit card data at certain hotels [64]. The company said that this breach took place between 20 March and 16 December 2013 at the food and beverage outlets of 14 hotels, including some operated under the Westin, Renaissance and Radisson names. The names and numbers on consumers' debit or credit cards, security codes and card expiration dates were stolen by the cyber criminals.

Peter G. Neumann is Senior Principal Scientist at Computer Science Lab, SRI International, and a moderator of the ACM Risks Forum. He considered the existing and the new types of risks inherent in cloud services in his publication in *Communications of the ACM* [65]. He stated that 'Cisco Systems had a private crypto key embedded in their VOIP manager that allows unauthorised control of sensitive messaging gear', and 'Dropbox's sharing services

were hacked, resulting from a security hole in its link-sharing scheme. The exploits were also disseminated by the perpetrators'.

Amazon EC2 is a large provider of cloud services. It was designed and built to provide the best environment for consumers and to prevent outages. However, Amazon servers went down for 15 min, affecting many of its customers across the US and Canada. Users only got an error message when trying to access some popular shopping websites that use the EC2 provider [66].

Another instance happened to the Sony company when an attacker used Amazon's EC2 service to rent a server to attack Sony's Play Station Network [67].

Another victim of data theft was a company called Epsilon, where a large number of details from its user databases were stolen [68].

Stratford is another company that had information stolen. More than 860,000 user names and passwords and 75,000 credit card numbers were stolen [69].

From the previous descriptions of the cloud and the examples given above, it is clear that hackers can benefit from cloud computing services to gain unauthorised access to users' critical data and information. Attackers have numerous techniques to illegally access specific objects or use the large scale of clouds to spread attacks. There are different types of attacks discussed in cloud computing, but the most concerning, which will be examined below, are the malicious insiders attack, denial-of-service attack, cross VM side-channel attack, attacks targeting the shared memory, and phishing attacks. Sometimes, an attack might be a combination of two or more different types [70], [71].

An insider attack is a type of a malicious attack that is committed to a network or computer system by a malicious insider. A malicious insider has been defined by CERT Insider Threat Centre as a ‘former or current employee, contractor, or other business partner who had or has authorised access to an organisation’s system, data, or network and intentionally misused or exceeded that access in a way that negatively affected the organisation’s information availability, integrity or confidentiality’. This new definition of malicious insider differs from the old definition of CERT in that it considers contractors and business partners to be potential malicious insiders who have access to the organisation’s systems [72].

CERT considers a contractor to be a potential malicious insider because the contractor could be granted access to the information of the clients and their systems and networks. Furthermore, organisations often relax nontechnical control for contractors more than for employees [73]. Further, because trusted business partners have been responsible for numerous attacks, they are also included in the definition. The authors of [73] have discussed insiders and malicious insiders’ concepts in relation to cloud computing.

Insider attacks are one of the most harmful and serious risks to cloud computing. According to the 2011 Cyber Security Watch Survey conducted on 607 businesses, government executives, professionals and consultants, 21% of the cyber attacks were caused by insiders. Further, 33% of the respondents thought that the insider attacks were more costly and damaging than the other attacks to organisations [56].

A denial-of-service (DoS) attack, or distributed denial-of-service (DDoS) attack, is another common example of an attack in the cloud. While the DoS attack type needs only one

attacker to achieve its goals, the DDoS attack needs two or more attackers [74].This type of attack stops servers from performing their duties because they are too busy handling the spurious requests. Hence, DoS or DDoS attacks can result in a loss of revenue for the provider and its clients. The damages of this type of attack can affect numerous users who rely on the systems under attack. Companies have reported losses ranging from $10,000 to $100,000 per hour during a DDoS attack [75]. DDoS attacks are now frequent and constant [74], and attackers keep changing their methods to keep ahead of methods that thwart these types of attacks [74].

One of the most concerning attacks in cloud computing is the VM side-channel attack [76]. To perform such an attack, the attacker uses software to break out of one VM and into another by gaining unauthorised cryptography secrets from the lower infrastructure. A cloud company that executes VM on shared physical infrastructure with public clients is the typical target for this type of attack. As each VM can run a different operating system, each may have its own vulnerabilities. This gives these VM attacks more opportunity to abuse the shared resources [76], [77]. The attacker gains tenancy on the same physical infrastructure as the target in an attempt to steal critical data over a period of time. Even if the cloud provider offers a perfect service, sharing per-core CPU information such as L1 data and instruction caches may allow the theft of cryptographic secrets [77]. In fact, some hypervisor developers have realised the risks of these attacks and applied protection using the conventional access control mechanisms to logically isolate the VMs [77]. An example of a security vulnerability to achieve a VM side-channel attack is VENOM, where ‘VENOM is a security vulnerability in the virtual floppy drive code used by many computer virtualisation platforms. This vulnerability may allow an attacker to escape the

confines of an affected VM guest and potentially obtain code-execution access to the host. Without mitigation, this VM escape could open access to the host system and all other VMs running on this host, potentially giving adversaries significantly elevated access to the host's local network and adjacent systems' [78].

The basic concept of cloud computing is to deliver shared infrastructure for several clients, potentially in different regions. Some attackers use this feature to target the shared memory that might have critical data for some other users [79], making this environment a good target for an attacker. Once the attacker gains access, then he/she can transmit valuable information to his/her device [76].

In a phishing attack, the attacker circumvents the system security for a user by sending messages pretending to be from trusted companies, banks or governments to trick users into giving their private information (for example, credit card details, email addresses and passwords). Once these data are obtained, they can be used to achieve the scammer's goals [80]. Another technique that phishers could use to obtain data from victims is to send warning emails with known titles (bank, university and so on) to the victim warning him/her of losing important data. Therefore, the victim follows up with the emails that might have a link to solve the problem. When the victim clicks the link, he/she is taken to a fake website that looks like a legitimate source, which gathers specific private data from the user. Phishing is very common [81]; attackers make use of the large scale of cloud networks to widely send their messages. A Gartner study claims that $3.2 billion of losses in 2007 can be attributed to these types of attacks [82]. More recently, phishing attacks targeted Google services. Chinese hackers created Gmail accounts masquerading as senior

US and South Korean government officials, military personnel, Chinese activists and journalists to be used in phishing goals [83].

Researchers in the academia and the industry have developed logical and physical solutions to solve the vulnerabilities affecting cloud computing. Some of these attempts include the following:

- Providing physical intrusion detection systems in data centres. Jakub and his team [79] introduced a physical cyber defence mechanism that uses the actual physical features of the data centre (for example, cameras, motion sensors and electronic locks on the door) to determine the time an attacker would spend to gain access to the equipment physically. Based on the time between attack detection and an actual attack, certain defences or response actions can allow the protection of data at rest.
- Jay et al. [80] in the paper titled "Improving stored data security in cloud using Rc5 algorithm" introduced a new encryption technique using the Rc5 algorithm. The user can use this method during data transmission to produce cipher text, which is the result of data that have been encrypted. Therefore, even if the data are stolen, they are unreadable as only the user knows the keys to decrypt the data. This encryption algorithm helps to avoid the traditional problems with encrypted data, such as efficiency of operation. By using this encryption technique, the user need not worry about the security of data in cloud storage as the data are encrypted and nobody else has the encryption keys. This technique provides a new method to encrypt data during transmission. However, it is still vulnerable to threats such as malicious insider attacks, as the keys are available on the client side, making them accessible to malicious insiders [84].

- Some cloud providers may break SLAs for financial reasons or convenience by placing files on arbitrary physical hard disks instead of placing the files on particular hard disks as agreed in the client's SLA. To save users from such negligence, a 'Chinese wall' was introduced by Zhan et al. [85]. This system aims to isolate each VM from accessing the data storages of other VMs, even if they are located on the same server, by investigating the risks of VMs residing on the same server and developing a technique for a 'concrete verification to detect other occasionally appeared co-resident VMs on a reserved physical machine'. However, while this commercial policy might be able to solve one side of having VMs residing on the same physical disk, it does not solve all the risks around VMs, such as live migration or insider attacks [85].
- Trusted Computing Group (TCG) proposed a structured cloud computing model through several technologies to achieve security in the cloud [86]. To provide a trusted platform, TCG introduces the trusted platform module (TPM) based on a hardware protected cryptograph. TPM is a hardware technology that was built by the group on the basis of international standards to implement trusted computing. The chip has several features such as the ability to secure certificates, keys, platform state information [17] and passwords and cryptographic capabilities to create fingerprints for systems and an integrity check of systems and authentications. It has been adopted by many computer manufacturing companies such as Intel, Microsoft and Boot Guard [87]. These manufacturers have released two other technologies to support cloud security, namely trusted network connect to provide secure networks and trusted storage to provide a fully encrypted disk [88].

Cloud computing is an application platform that either sits on top of software programs or at least uses the functionality of software programs. Hence, any known software security holes can be used to attack cloud systems that use these software programs. As Fleece stated in his research (March 2015) [4], 'The cloud computing implementation is still based on hardware and application infrastructure—all the stuff IT people have worked with for decades'.

It is becoming more difficult to consider cloud computing to be safe, particularly after many famous cloud service providers have suffered from being hacked in different countries and from different hackers as described above. Conventional security measurements are not sufficient to mitigate the threats in cloud systems.

Many IT groups and academic researchers have been trying to fix the security holes in cloud computing, but their solutions address the existing problems without completely solving the main cause, as discussed in the previous sections. Therefore, it is time to think of a new environment that can provide cloud services and new mechanisms of security to perform normal operating system functions securely. The aim of this research is to build a cloud platform based on a secure operating system to run cloud services. The base operating system chosen to build on is Secure Persistent Execution Environment for Distributed Operating Systems (SPEEDOS), [89] an operating system with security as the primary goal.

Chapter 3
Overview of computer systems

3.0 Introduction

An operating system (OS) is a software program that manages and sets the standards and operations for all of the application programs that perform the actual work required by the users [7]. Application programs can send requests to the OS by passing parameters, and the OS decides whether to proceed with these requests by alternately sending back the operation results, or if a request is not proceeded with, another return value such as a system error to indicate that an error occurred during the processing. In addition, the user can interact with the OS by typing commands or using a graphical user interface (GUI), which can be considered a part of the OS. OSs perform the following functions [7]:

- **User interface:** communication between a user and the computer by different interfaces such as commands, windows, menus, or some other method.
- **Job management:** The OS manages the applications that are executed in the system typically controlled by using a process scheduler, which handles the allocation of processes to the CPU, pre-emption of processes from the CPU and the selection of another process on the basis of a particular strategy. Process management is a part of all multitasking systems that support multiple concurrent processes or tasks, and this feature exists in all major OSs today.
- **Memory management:** The OS manages each memory location in the system, and deciding which processes have access to which parts of the memory, which locations are allocated or free and when memory is allocated, which memory is assigned**.**

- **Data management:** The OS tracks all of the data on disk or other storage devices.
- **Device management:** The OS controls peripheral devices such as printers and keyboards by sending them commands that they can understand and interprets messages sent back by the hardware.
- **Security:** The OS provides security to the user. For example, providing password protection to prevent unauthorised access, and backup and recovery techniques to save data from being lost because of hardware, software or environmental mishaps.

3.1 Computer memory

Computer memory can be classified as either main memory (volatile) or secondary memory (non-volatile). The volatile memory maintains information while the power is on. An example of this memory is RAM, which is designed and implemented to store data in. This data are lost when the power is lost because the memory requires power to save/maintain data. However, conventional systems also have secondary memory (for example, disk or flash memory) that provides the basis for file systems and can store data even after the power is switched off. Each type of memory has its own memory management system. These management systems continually track all of the memory locations to monitor which locations are allocated and which ones are free.

The CPU is a machine that performs the main processing work, under the control of the OS, by using the main memory of the computer that stores the code that the CPU executes, and the data used by or produced by the executed code. The system continually performs the following steps:

- Fetch data (an instruction and required operands) from the main memory.

- Decode the data that have been fetched from the main memory.
- Execute the decoded instruction.
- Store the result back into the main memory.

This machine implements the execution by repeatedly cycling over these four steps very quickly.

3.2 Memory management

The demand for larger computational memory and the high cost of RAM have led to the concept of virtual memory, [8] which exploits a part of the secondary memory to be used as an extension of the main memory, thus creating the illusion of directly addressable (that is, computational) memory by using a combination of (directly addressable) RAM and (file system managed) disk storage (see Figure 3.1 below).

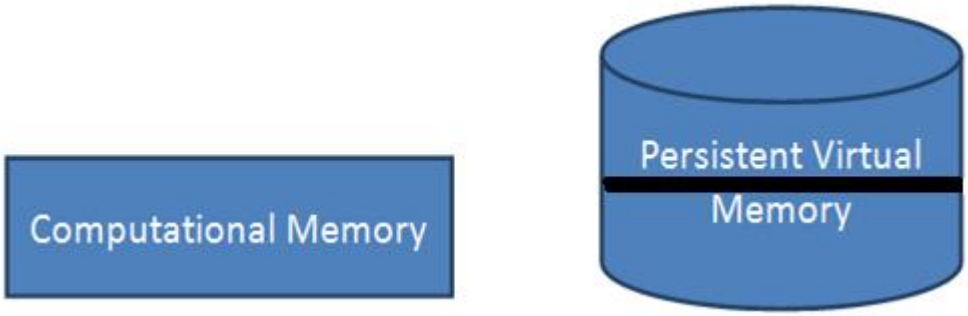


Figure 3.1: Representation of conventional memory management [8]

This potentially means separating the storage of the currently running programs and their data across two media: all or part in the main memory and again all or part in the secondary memory. This mechanism is achieved by having the OS fetch and store information in the

disk-based extended main memory (termed ‘swap space’ or ‘paging area’) and utilising hardware to check whether the bytes in the executing program and/or its data are located in the main memory, and if not, to take further actions to move the bytes from the extended memory into the main memory.

Prior to virtual memory management, the management of the main memory was based on the reservation of memory by programmers, by dividing the memory into variable-size blocks called segments (lumps of memory that contain logically related data of a program and some protection mechanism) as requested by the applications. Programmers were provided with the ability to reserve memory for their applications and add restrictions to protect their memory. At times, this resulted in an executing program accessing a memory location outside the segment bounds. Another drawback of this allocation strategy was that there was no guarantee that an allocated segment was directly adjacent to a currently free block of memory, which ended up creating a situation in which the available memory became increasingly fragmented over time. The fragmentation of memory wastes memory space by using it inefficiently and results in the free memory blocks being too small for any request [90].

The consequence of fragmentation is that although the total amount of free memory is sufficient to provide more storage, there is no single contiguous block of an appropriate size. A consequence of virtual memory management is to defer the impact of fragmentation, usually long enough to negate its impact. The act of moving data between the computational and the secondary storage on demand is called ‘paging’ or ‘segmentation’. Conventional paging uses a fixed-size unit of transfer between the main

memory and the virtual memory (the virtual page) whose size is determined on the basis of the support afforded by the processor architecture and by the fine-tuning decisions made by the OS designer.

This management model has two separate memory management systems: one that manages the data stored in the main computational memory, and the other managing the data stored in the non-volatile (disk-based) extension to the computational memory. This leads to duplication of the memory management techniques and infrastructure, and thus more complexity in the system.

The currently used processors typically implement only the paged virtual memory; some of them mention segmentation in their specifications for only a group of pages. Today, segmentation is almost extinct. The protection benefits gained through the use of segments have taken a back seat, with the current emphasis on performance [89].

Here are some of the current memory management problems [8], [89]:

- There are two separate memory management systems.
- Hackers can take advantage of the interface between the file system and the computational memory.
- It is difficult to prevent subjects (people and programs) from accessing information which they are not supposed to access.
- It is difficult to prevent subjects from passing on the information to unauthorised third parties.
- The file systems are the most hacked parts of conventional systems.
- The central password files are a major target for hackers.

3.3 Shared memory

A conventional computer system provides services to users through its support for processes. In such systems, a process can be defined as a set of tasks linked to a specific virtual memory space to implement the system's intended functionality [89]. This, together with the pre-emptive process scheduling provides the illusion of there being more than one process executing on the machine simultaneously even if the machine has a single call (that is, multitasking). A process may use the OS functions to exchange data with other processes in the same system. However, support for concurrent multiple active processes creates problems with the sharing of information between processes without going through the OS for every shared access [89].

The operating system features mentioned above, together with some others such as shared memory that allows the same physical memory to be mapped into the virtual execution environment of multiple processes, are supported in the common operating systems in use today, such as Windows, Mac OS, Linux and UNIX, IBM I and DOS.

3.4 Problems with currently used operation systems

The currently used standard operating systems that are utilised for desktop computing, for example, Windows and Mac OS X-based systems, prove to be unsatisfactory when it comes to security where the data can still be accessed without the correct authorisation [4]. The security features in OSs have been proven to have weak implementation, in the last decade [4]. This may be attributed to many reasons, including the appearance of Internet services that connect different computers and users from multiple regions and countries. Such services host data from users from all around the world that are stored in one or more

physical places. However, these places host a considerable amount of the clients' data on behalf of the clients who access them regularly. Using such services on a large scale brings security challenges to OS programmers, who are responsible for implementing security mechanisms to prevent users from illegally or inappropriately gaining access to other users' valuable data.

In a typical OS such as Windows or Linux, the core networking code that includes network card drivers and various protocol stacks (802.11 and TCP/IP) runs in the kernel. This means that a potential bug in the network code, when exploited by an attacker, can result in a full system compromise.

One of the major problems with the currently used systems is their failure to provide effective isolation between different programs running on one machine [91]. If the client's web browser gets compromised (because of a bug misused by a malicious site), the OS is typically unable to protect the other clients' applications and information from also being compromised. Thus, if certain system core components get compromised, for example the Wi-Fi driver or stack, none of the above-mentioned systems can defend themselves from a complete compromise of all of the applications and data.

The above circumstance is an immediate consequence of certain architectural design decisions, which include overly complex OS APIs, insecure GUI design and monolithic kernel architecture [92]. Systems that depend on such an insecure design cannot provide strong security. The reactive approach, which many vendors utilise today, tries to fix each and every known security bug. In any case, such an approach does not scale well, particularly for clients who require more security. Fixing can only address the known and

popular attacks, and offers no security against new, or less-prevalent, more-targeted threats. Therefore, a different approach is needed to build a secure system.

3.5 SPEEDOS architecture

Professor J. L. Keedy and his team redesigned the way the common OSs are structured and the support they provide for executing programs. They took a radical approach to achieve better mechanisms, most recently in an innovative operating system called SPEEDOS [89]. Some ideas included in the SPEEDOS system are based on previous research such as Keedy's Monads [93] and his collaborator's Grasshopper [94] projects.

SPEEDOS is a unified system that was designed and built 'from the ground up' to achieve a very high level of security; it aims to solve the protection problems found in other operating systems and to meet the challenges faced by the widespread usage of software systems [8] in today's highly connected computing world.

The term SPEEDOS is an acronym for Secure Persistent Execution Environment for Distributed Operating Systems. The SPEEDOS design provides only one level of memory abstraction, used for both computation and long-term data storage. This is contrary to conventional computing, which has two memory abstractions: one non-volatile memory for long-term storage, and the other computational memory that loses its content on re-boot or when the power is switched off. SPEEDOS provides a single persistent [89] virtual memory, so all of the data persist as a natural property of its existence.

Interestingly, then, a fundamental computational assumption in SPEEDOS is that all the data exist and continue to exist unless the user explicitly deletes the data, which is in

contrast to conventional systems that assume that data are not required beyond the life of the program that generated them, unless the user explicitly acts to save these data. The single persistent virtual memory model brings challenges because it does not provide the between-program protection mechanism inherent in the conventional ‘separate virtual memory environment per process (or group of threads)’ model.

SPEEDOS uses conventional computational memory features (for example, segmentation and paging) in a novel way (for example, memory management and protection) [89] to achieve the persistent memory features that support its virtual memory. Segmentation and paging models are orthogonally applied in SPEEDOS.

3.6 SPEEDOS kernel

Data security is very important in today’s technology. The design philosophy of the SPEEDOS kernel is to create a complete security environment that provides new techniques to protect the entire system and all of the user modules (modules represent the processes that run in the SPEEDOS system) and contains the interaction information hidden in the modules according to the software engineering ideas originally proposed by Parnas [95]. The kernel focuses on all of the aspects required by the OS to meet the challenges that are faced by the widespread use of software applications, including the following:

- Protection: the mechanisms by which the system avoids the possibility of danger, for example unauthorised outsider access to data or the inability to access memory addresses beyond the end of a data structure.

- Security: the policies (implemented using the protection mechanisms and other techniques) that identify the user rights and permissions to access data, define what can be achieved during the permitted access, ensure the integrity of messages, report anything unusual and protect from the loss of valuable data. This protects the data from illegal actions.
- Uniformity: the basis ensured to achieve all of the system functions in a formal manner to improve efficiency for both the users and the system.
- Flexibility: the ability of a system to deal with potential internal or external changes affecting its value delivery, in a timely and cost-effective manner, such as extending the memory allocation if needed.
- Extensibility: the ability of the system to be upgraded via software or hardware to deal with the fluctuations in the user needs.

The SPEEDOS kernel provides execution environments on the basis of new principles that are considerably different from the principles found in conventional operating systems. Some of these were inherited from and built on those implemented in previous systems such as Monads and Grasshopper [89]. The SPEEDOS kernel can access well-defined data structures stored in the persistent virtual memory to implement the basic support for modules and processes. All of the SPEEDOS modules are assumed to be represented in the virtual memory, and all of the data maintained in the kernel are copies of the data stored in the persistent virtual memory. This implies that each process in SPEEDOS describes itself, including maintaining its own mapping table for the memory; it uses the same stack to save data and keep it persistent (processes are self-contained and not maintained by the kernel).

The kernel uses the process's own internal data to manage the object rather than having the kernel maintain the data stored in the kernel.

These features allow the kernel to work on persistent data without the kernel itself being persistent. Having the kernel not be persistent in SPEEDOS eliminates the need to represent it in the virtual memory, removing the challenge experienced in Grasshopper, for example, in which the kernel data persistently create a recursion problem when implementing stability by saving data to the non-volatile memory. The stability of a persistent storage requires that when an error occurs, the storage can recover to a previously recorded consistent state [89].

3.7 Memory in SPEEDOS

The memory available for the combination of computational and long-term data in SPEEDOS is an enormous space, as each user's process stack size is up to 2^{64} bytes.

The SPEEDOS memory management improves the uniform persistent virtual memory by implementing an orthogonal segmentation and paging model (explained below) to eliminate the duplication of the memory protection mechanisms found in most of the other operating systems (see Figure 3.2). The SPEEDOS kernel is designed to support the large persistent virtual memory that SPEEDOS has, and a very high level of security, which means that it might be possible to solve the protection problems encountered by the memory management in other OSs.

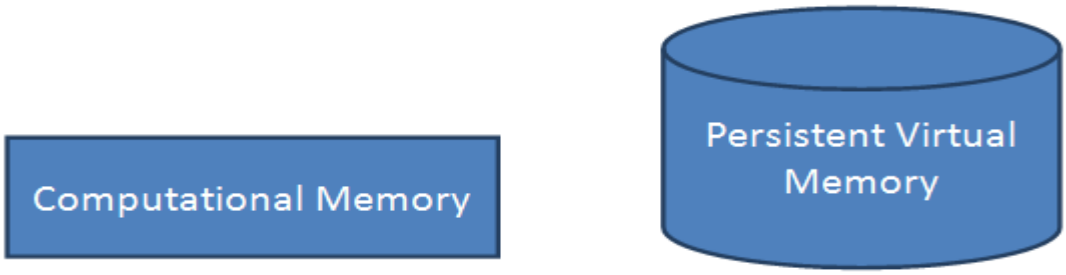


Figure 3.2: Representation of memory in SPEEDOS system [8]

3.7.1 Paging

The SPEEDOS model has an enormous virtual memory with each container belonging to a user (a container is a large unit of virtual storage that can be compared to files in conventional systems) assigned up to 2^{64} bytes. The paging model is used in SPEEDOS to support such a large data size and to map between the virtual memory and the physical memory space. Each page's virtual memory address has a unique container number and an offset within this container.

Each page of the virtual memory is managed by the kernel and, when possible, is represented in the main memory through a main memory page table (MMPT). The MMPT is a global resource for all of the processes in the SPEEDOS system and is updated indirectly by the modules that implement the virtual memory. The kernel implements the translation look-aside buffer (TLB), where the TLB is a fast cache used to store the recent translations of virtual addresses to physical addresses. If a page fault occurs in the TLB, the kernel passes a module capability to the container directory, which is responsible for resolving the page fault in cooperation with the page table manager by using a pre-determined method [89].

3.7.2 Segmentation

A segment is a lump of memory that has the logically related data of a program and some associated protection mechanisms. Segments are orthogonal to pages and may be of different sizes as compared to a page. Segments are rarely used in conventional systems but are common in SPEEDOS. One of the reasons for their use in SPEEDOS is that they can be used as a protection unit. To clarify this, when data have been stored in a segment and the system is aware of the segment and aware of its length, then if the execution of a code which uses a segment attempts to access an address outside the range of the segment, the system will not allow the access. This stops hackers from being able to access memory that they should not have access to, for example when they take advantage of the buffer overflow.

The SPEEDOS system is aware of each segment's length. Each of these segments consists of three parts:

- Normal data: Data structures that represent the usual programming language types such as integers, characters, arrays or more complicated data structures.
- Pointers: A reference form that is used to link segments with the segment capability that describes the segment, including the access rights within a container. All pointers are references to segments in the same container to ensure that the containers remain isolated from each other.
- Module capabilities: These specify how a segment is permitted to relate to other containers and can only be accessed by the kernel. They represent references to other containers and follow the information-hiding principle that was first proposed by Parnas

[95]. By this we mean that these references refer to modules and not directly to the segments stored in any module's container.

The pointer parts of the segment capabilities are references to data in the same container only. This enforces the related data to be placed in a single container, which ensures the information-hiding principle [95]. It also encourages the application of similar protection requirements to an entire group of information. Another benefit of this mechanism is that it limits where direct memory references may be stored, which may make garbage collection (automatic reclamation of memory for reuse) in a distributed shared memory system feasible [82].

The representation of a segment capability that describes a single segment has the length of the data and several fields as shown in Figure 3.3. The offset identifies a segment capability for a segment within the container. Segment capabilities are only unique within a single container. The next three parts specify the length of the segment, the number of pointers and module capabilities that the segment contains, respectively. The pointers and the capabilities can only be accessed by the kernel to modify their content to prevent forgery.

Offset in Container	Length of Data	Number of Pointers	Number of Mod Caps	Access Rights

Figure 3.3: Representation of a segment capability

The access rights field determines the access rights to a segment. The access rights to segment areas are basic access rights, allowing the reading, writing and execution of the

normal data to be controlled. For the other parts, only read and write access rights can be specified.

This representation of segments shows the size of the elements; their content is not directly accessible to protect the boundaries between all the parts from obtaining unauthorised access to memory locations outside the boundaries of a segment. This design also prevents any program from going further to access the pointers and module capabilities that are stored in the registers of the main memory (a register is a high-speed storage location in the CPU to store data while a program is being executed).

3.7.3 Containers

A container is a large unit of virtual storage that is generated from a persistent medium (it can be compared to files in conventional systems). Its realisation in practice as a physical area of memory on the disk, with a length of up to 2^{64} bytes, makes it potentially very large.

Each disk is divided into multiple containers (note that the allocated space needs to store only the populated part of each container), and each user can have one or more containers. Each container has an address table that tracks the allocated pages and the free pages. All the containers in SPEEDOS have worldwide unique addresses and can be located in any SPEEDOS system connected to the network.

There are three different types of persistent containers held in the SPEEDOS system:

1. Process containers: These belong to a user, are located where the user saves his/her capabilities and information, and contain threads (processes in SPEEDOS are divided into threads which represent the concurrent activity of the application modules).

2. Data container: This holds the system data of a module.
3. Code: This holds the code and the constants associated with a module. In other words, it contains the read-only data and the code.

The code has been separated from the data container to allow its use by different processes. For example, a banker does not need to repeat the code for each bank account file in the data; there is one copy of the code, and the data are separated to protect the individual bank accounts.

In a single container, the SPEEDOS kernel does not specify how several implementations of different programs can be related to each other. However, it is up to the OS designer or user to follow one of the several strategies mentioned below to bundle different codes into a single container (or to determine an acceptable alternative) [8]:

a. Placing the code of several related types of modules that are typically used together in a single container. They share a single page table that describes the allocated pages in the container.
b. Sharing the code segment amongst several related types. For example, the implementation of a double-ended queue can be used to implement a stack simply by constructing an appropriate entry point list that only uses one ‘half’ of the double-ended queue.
c. Allowing the same type or modules to share some methods by placing their implementation in a single container.

The container creation is the duty of the container directory which sets up the initial content of a container such as a co-module manager. The container directory is the only module

that has the authority to create module capabilities in SPEEDOS [89]. If a page fault occurs a container directory module has to be invoked to cooperate with the page table manager module to resolve it by using a capability module from the kernel from which the fault occurred.

3.8 Applications in SPEEDOS

In conventional computer systems, a program starts when its program code is loaded into the main memory, after which the information that is held in the user's data files is accessed to carry out the program's defined task. Typically, the user rights determine who is permitted to access the data files of a program, and these access rights are typically restricted to either 'no access', 'read only' or 'read-write'. The representation of programs and the appropriate method invocation mechanisms are the core of the SPEEDOS kernel, which follows the in-process model (explained in Section 3.9) and implements modules. Each module has its own routines that represent the access forms and are managed by the user, who gives authority to other system modules.

In SPEEDOS, each application or program is represented as a module (composite module) [89], and each composite module consists of co-modules belonging to a particular application. Co-modules are intended to represent the OS functionality associated with an application, not as a structuring mechanism for the actual application functionality [89]. All the co-modules belonging to the same application module are represented in a composite module in the same container, and their data are separately installed from other co-modules. Co-modules perform tasks that are closely associated with an application module and provide security functions to assist the kernel. Each co-module consists of a reference to the

co-module table, which represents the references between the private data and the implementation code for each co-module. The co-module table allows mixing and matching the implementations of different co-modules, managed by a special module called the co-module manager.

When a composite module is created, it conceptually contains one co-module table with a reference to the co-module manager code. It is the task of a central OS module to perform the necessary resource allocation. Initially, when a new co-module is created, similar to the constructors in many object-oriented programming languages, it is first rooted as a segment with a null value in the file container (the co-module table contains a null reference). Then, the co-module invokes a well-known method of the new co-module to return the required lengths for the normal data, pointer and module capability areas of the root segment [89]. The segment manager is used by the co-module manager to allocate the requested size for the segment and records it as the root segment for the new co-module in the co-module table that maintains the properties of the other modules. The new co-module can be used after the allocation of the root segment.

Each co-module can be qualified by a list of qualifying modules [89], where the qualifier list manager for the co-module is usually represented as a complete composite module stored in a field in the respective co-module entry. The content of the qualifier list is controlled by the qualifier list module and is referenced by the co-module manager. Each qualifier list module contains a list of identities that should no longer have access. The information-hiding principle applies to any co-module interface method, providing a fine-grained, freely programmable access control system. With respect to security, this

mechanism can be used to monitor accesses, add time-restricted access or deploy decoys to the co-module.

The access control for SPEEDOS modules is enforced by a mechanism called capabilities [8]. This is considerably different from the technique used in most conventional computer systems, which typically use access control lists (ACLs [8]) of entities such as users who have access (read and/or write) to objects in the system. A capability includes the identity (in SPEEDOS, this is the virtual address) of a specific module, the access rights provided by the capability, and a definition of what the owner of the capability is permitted to do with the capability itself (for example, provide a copy to another user). Capabilities are controlled at the user level and can be passed to any other system module to access the content of the specific module referenced by the capability.

A capability also contains information including which routines can be invoked and other protection-related fields, such as whether the module or capability can be deleted or copied or how many times the capability can be used. Capabilities are fundamental system entities; they cannot be created, changed, stored, accessed or, in any way, manipulated by programs. The only thing a program can do with a capability is to notify the system kernel that it wishes to use it. Then, the kernel retrieves the capability and matches it with the called destination's module capability. The kernel then performs the desired action(s) if the permissions allow it to do so. In this mechanism, it is not possible to, nor is there a need to, directly access the information hidden/encapsulated in the data file of the module as is the case in conventional systems. In summary, in SPEEDOS, a module cannot be accessed by

any other module unless it has the authority specified by a capability controlled by the kernel.

The other technique which can be used in SPEEDOS to control the access to modules is called a qualifier [8], which is a special module with its own code, data and routines that mainly check and qualify the calls between modules with the help of the kernel and can, for example, prevent access to a qualified module from the unauthorised calling modules. Each co-module has its own qualifier list stored in a specific field in the respective co-module. There are different types of qualifiers that can be placed around the modules, including the following:

- **Blocking Qualifier:** This qualifier has its own code to misinform the unauthorised module or block it from calling, as shown in Figure 3.4. For example, the user can use this qualifier module to feed the spy client with harmless information, as a disinformation mechanism.

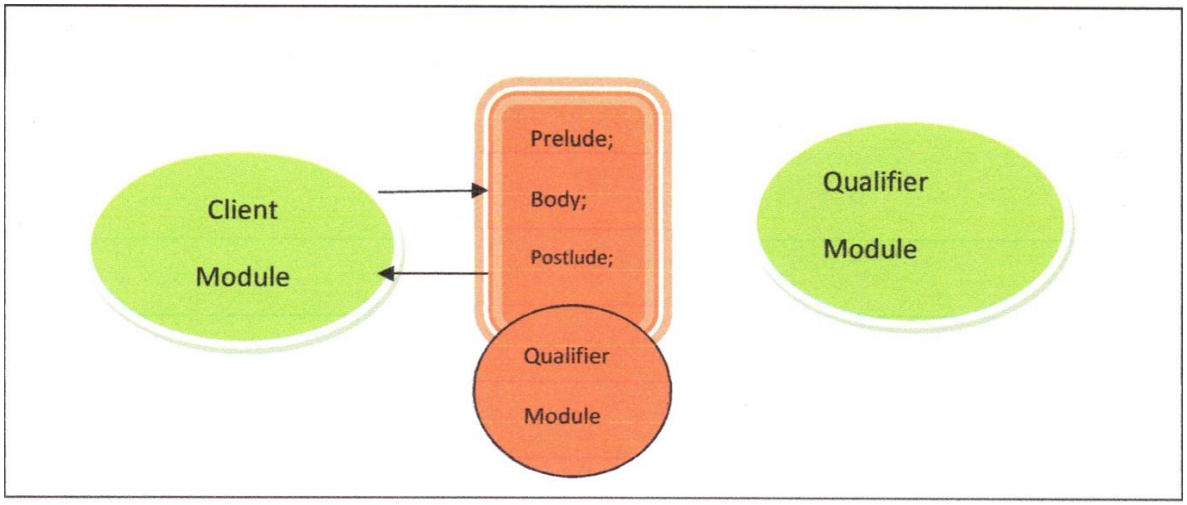


Figure 3.4: Representation of blocking qualifier.

- **Body Instruction:** It refers to the addition of some instructions to the qualifier code that tell the kernel to allow the inter-module call to continue, as shown in Figure 3.5, for example, to achieve a normal instruction.

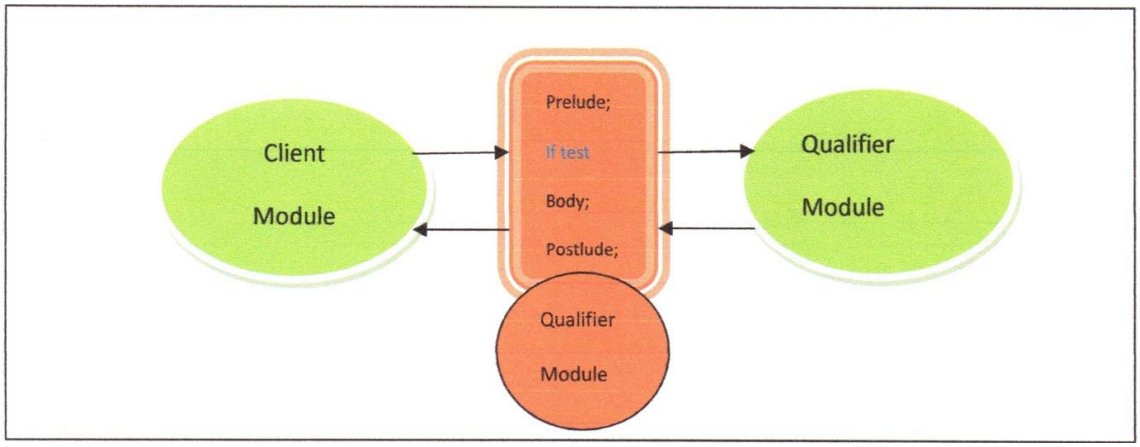


Figure 3.5: Representation of body instruction qualifier.

- **Augmenting Qualifier:** This is a qualifier model that has many uses such as the synchronisation management between modules and the maintenance of a log of all the accesses to the target module.
- **Testing Qualifier:** The instruction body in the qualifier can hold a conditional statement to enable the qualifier to allow the call to continue only if it determines that the module should have the corresponding access (see Figure 3.6), for example, to provide additional security to a particular module.

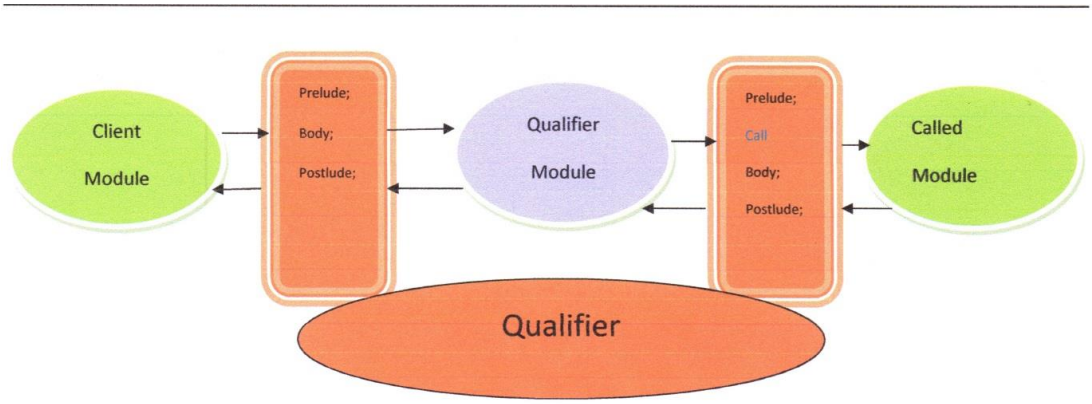


Figure 3.6: Representation of testing qualifier.

- **Callout Bracket Methods:** In this case, the qualifier calls another routine of a third module to access the information that it requires (see Figure 3.7), for example, to log all of the outgoing calls or block all of the outgoing calls to a specific module.

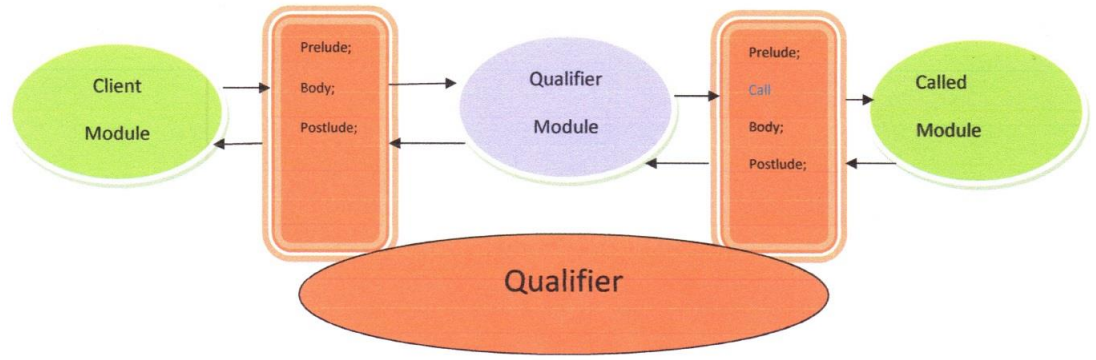


Figure 3.7: Representation of callout bracket method qualifier.

A qualifier is different from the targeted module; it qualifies the caller's access calls. However, each qualifier type has its own method definition where it defines the type of call that it qualifies [89], so that it can, for example, implement synchronised access to an otherwise unsynchronised module.

3.9 Processes in SPEEDOS

A process can be defined as ‘a major computational unit belonging to a user’ [89]. In conventional computer systems, when the system starts up, a number of processes are executed in parallel to provide services that support the execution of user-level processes. However, these services are intended to serve multiple users’ processes, or in the case of single-user systems, to serve multiple (generic) user-level programs invoked by the user. A user process typically sends a message to the system process from which it needs a service; such messages are passed to the single process module that services these requests for all the user-level programs. There are two models of organising processes: the out-of-process model (sometimes called message-oriented) [89] and the in-process model (sometimes called procedure-oriented) [89].

3.9.1 Out-of-process mechanism

The out-of-process model [89] is very widely used in conventional computer systems. It refers to exchanging messages between processes in a system. When one object calls another object, the process scheduler implements this call as a message which goes through a queue to the receiving object, implying that the receiving object can receive multiple messages from the same source or from different sources and may execute multiple processes in parallel. This mechanism has a serious problem as the users may have many objects with many processes, which in the end creates a software traceability nightmare. In addition, each object might have multiple processes executing its code, all of which makes the communication between processes difficult to follow.

3.9.2 In-process mechanism

The in-process model [89] is another mechanism for organising processes and executing process instructions. Basically, it is a way to call objects in the context of the current process, in which the same process that makes the call executes the called method. This approach is used to implement the SPEEDOS processes, where a process can invoke other interface codes during its execution. This implies that all of the access permissions are already set up for this process.

Initially, when a user logs into a SPEEDOS system, an existing dormant process associated with the user is activated, or a new associated process is created to perform the user's current processing. This user process has the usual data structures, including its own stack to store the temporary user data. Irrespective of what the user works on, the system can put this information in the user's stack. The system has a single process for each user that implements all the software programs that the user runs in parallel and within one stack. Each part of the process is represented as a 'thread'. Threads are entities that represent concurrent activity within an application module in the container [89]. These threads use the user stack to save all the invocation records. Threads communicate with each other by using the shared memory in SPEEDOS. As the different threads stored in a container are isolated from each other, there are generally no references between the representations of different threads, thus avoiding the sharing of problems with the stack segments.

The SPEEDOS kernel carries out all the module calls, first checking whether a call is valid before continuing with the call. The caller thread should present the capability that shows the rights to invoke the semantic routines of the destination co-module and the number of

the entry point to be called. The kernel checks the presented capability by comparing it to the capabilities list of the permitted semantic routines. If the call is valid, the kernel locates the co-module called from the unique container identifier in the module capability and then, sets up a segment register (which is made accessible to the called module) that allows the calling thread to access the data section of the root of the persistent data in the called co-module. The other parts of the segment cannot be directly accessed. If the call is invalid, then the kernel initiates a bracket routine or a qualifier module to block this call. This organisation is efficient because the process scheduler has only the information about application processes as there are no system processes performing in-line activities for the applications.

3.10 Logging in and out

In a conventional system, when a user logs off, the processes that he/she was running will be deleted, and when he/she logs back on, all of these processes will have to be started again. The logout and login procedure in SPEEDOS is considerably different from that in a conventional system as the memory is persistent and nothing is lost between the logins. The procedure works by having an active module in the user stack that is invoked when the user logs out. This module is responsible for calling the kernel instructions that put a user's processes in the dormant state. In this sense, the user stack that is saving information for the user processes can exist as long as the user exists, even when the system is shut down. In summary, when the user wants to log out, the kernel of the system calls the user's logout routine, which deactivates his/her processes until the next log in, when the user will go back to exactly where he/she left off. The efficiency of this organisation is that when the

user logs in again, the process scheduler carries on processing from where it left off when the user logged out. Therefore, as a user, one will not need to recreate the information that was left before one logged out.

An excellent security advantage of this organisation is for an administrator or professional user to have the ability to add his/her own authentication code in any way that he/she chooses. The logout module that is activated by the kernel when the user logs in again can use a custom code to check the authorisation of the users. Technically, the first thing that the logout module does after being activated is to check the authenticity of the user in any way that it is programmed. Thus, instead of just having a normal password that may be guessed by somebody else, it could be programmed to require an answer to a mathematical problem every time the user logs in, or even to only accept the wrong answer to the problem.

The processes in SPEEDOS are in the dormant state until they are invoked by the user stack when the user logs back in. This part of the SPEEDOS design introduces difficulties where the processes need to react to environmental changes and changes in the I/O devices when they are re-attached to by the user [95]. This difficulty is worsened in the currently used commodity OSs that support a wide range of hardware devices.

3.11 Significance of SPEEDOS system

The SPEEDOS system includes new ideas to resolve the protection problems encountered in other OSs, particularly the memory management problems described in Section 3.4. This system has re-redesigned the implementation of operations within the current conventional system to ensure stronger security. It has significantly improved security approaches that

allow system administrators and/or users to protect their resources and critical information by a number of provided mechanisms. These make us think that SPEEDOS will be an excellent platform for a cloud system. The reasons for this decision are discussed in the following sections.

3.11.1 Minimal kernel

The SPEEDOS minimal kernel is one of the significant features of the system. All the information mentioned in the kernel is either a copy from the persistent memory or can be derived from it by directly accessing certain well-defined data structures that represent the core building blocks for the construction of software systems in SPEEDOS.

Each kernel instruction represents an elementary, atomic operation, which implies that the intermediate results are not visible to the user-level modules and processes. This is important for instructions that access and/or modify protected data structures such as capabilities, which must always be consistent.

The state of the current processes is not modified at all in the SPEEDOS kernel; therefore, interrupts are handled logically before the kernel instruction. In distributed systems, the SPEEDOS kernel works only on the local data and does not need to communicate with the other kernels to exchange data about modules and processes because communication is achieved through the user-level modules.

Such a kernel design provides the following:

- a single persistent virtual memory;
- no delay in starting new programs;

- a minimal kernel that is easier to understand and to prove secure;
- a process scheduler which only has to have information about application processes, because there are no system processes performing in-line activities for the application; and
- a considerably simplified kernel task as it can activate processes without being concerned about checking identities.

3.11.2 Memory management

SPEEDOS supports the orthogonality of segmentation and paging. Each of these schemas has unique addresses in SPEEDOS. Segments are used to represent a protection mechanism on the data and can start at any address within a container. Pages are changeable as part of the virtual memory management and are not used for protection at all. A page could have many segments or could be part of a single segment, and each segment in the same page could have different content (code segment, data segment and so on). In SPEEDOS, at any one instant, there can be many read-only copies of a page in the network, but only one copy of a page can be in the memory with the read/write access, and if so, none with the read-only access (multiple readers/single writer consistency protocol) as specified in [96] and [97].

This system's memory management provides the following:

- The same security technique can be applied in all situations.
- All the data and programs both on the disk and during the Internet transfers are easily encrypted.

- The number of security mechanisms is decreased, and their value is enhanced simultaneously.
- There is a single persistent virtual memory.

3.11.3 Representation of applications in SPEEDOS

A composite module can be understood as a cluster of information-hiding modules that hold data in a container owned by the user who created the application module. The co-modules are intended to represent security functions to assist the kernel and the data stored in the same container as that used for storing the data of the application module. The kernel uses co-modules that have a predefined set of the basic semantic routines. Each co-module in a composite module can be qualified by one or more qualifier modules to check the calls to the other co-modules. All the co-modules are represented in a co-module table that is managed by a special co-module called the co-module manager. The co-module manager has all the properties of the other modules, and its methods can be accessed through a module capability that is passed to the kernel in an in-process call. This representation structure of the modules in SPEEDOS has many security advantages besides the benefit of isolating the module's data.

This design for applications means that there is no standard way of logging into the process because of the principle of brackets used in SPEEDOS; therefore, a hacker attempting to log into someone else's process has no idea what he needs to do to activate the process. The code in the logout module can check the user's authenticity in any way that the user chooses.

3.11.4 Module capabilities

In the SPEEDOS system, the kernel has more control over modules and no module is allowed to access another module without permission (capabilities). Capabilities are stored in a segment in the environment of the subject, who can organise them as desired. The capabilities module cannot be accessed by a program; only the kernel has access to it and can modify their content. The three main types of module capabilities allow access to file modules, code modules or threads in SPEEDOS. There are also some other types of capabilities used for special purposes, such as kernel capabilities that are normally used in the context of OS modules.

Capabilities give access rights that determine the interface methods to be invoked. They are usually created with all access rights set, and these access rights can be restricted during their distribution to actual clients and cannot be changed using kernel calls. The bracketing system provides the ability to prevent execution from gaining access to a module outside of the access rights structure, which adds more security and minimises the target for attackers from gaining access to the data modules.

This type of module representation provides the following:

- There is controlled access to modules. Hence, in SPEEDOS, there is only one way to access a module, that is, by invoking a semantic routine; this is only possible if one can present a capability with the appropriate authority.
- The access control is undertaken by the security kernel and by some further checks such as the qualifier modules.

3.11.5 In-process communication protocol

The kernel only provides basic support to the processes and does not have any processes representing its activity. All the functions and system calls in SPEEDOS follow the in-process model. The form of the in-process communication protocol (objects can be called in the context of the current process) has a number of efficiency advantages. For example:

- processes exist in a persistent (but dormant) form while a user is logged out, and
- the process scheduler has to have only the information about application processes, because there are no system processes performing in-line activities for the application.

3.12 SPEEDOS security benefits for cloud

The SPEEDOS system provides many security advantages over conventional systems. Such a system can support a secure environment to provide cloud services, particularly as it has already solved most of the memory management and security concerns of the other OSs (as discussed in Section 3.4). Some security benefits that the SPEEDOS features provide for the cloud are as follows:

- The module structure used to represent applications/processes in SPEEDOS is designed to prevent unauthorised access, malicious attacks and Trojan horse programs. As explained in the previous sections, each module has to present a capability to call another module, and this call is then carried out by the kernel, which can request further checks by the qualifier system.
- The use of the container concept, with its memory size, to store a module's data and code, is persistent until explicitly deleted. Moreover, it keeps the different thread stacks

isolated from each other to prevent the problems of using shared memory that the cloud users face.

- The orthogonality of paging and segmentation used in the SPEEDOS memory enhances the isolation between the cloud users' data. This is attributed to the fact that all the data at rest (that is not in transit between hosts) are stored in segments completely separated from the page boundaries. Moreover, the segment length in a SPEEDOS system is defined, and each segment is designed to be accessed via the presentation of a capability that prevents the access process from going past the segment's end to possibly access data in another adjoining segment. This design prevents hackers from being able to access memory that they should not have access to.
- Each segment belonging to a user process has a segment capability that prevents any one from accessing it without presenting the appropriate access rights. This will solve the problem of unauthorised extraction of data from the owners of the cloud. As explained in Chapter 2, some governments have been extracting data using cloud services to obtain information that belongs to the cloud users; in SPEEDOS, the only one who can access data from a module is the owner of the module or people who are authorised by the owner.
- The providers can enhance security by using the qualifier modules. These can be used to qualify each call between the modules and record how many times they allow making the call. As the qualifier module is a unique facility provided by the SPEEDOS system to provide enhanced access control; this mechanism may help prevent some regular cloud attacks from occurring.

- The use of a logout module in SPEEDOS enables the providers to custom-design the login authentication to their systems. The providers can use many authentication techniques such as a math question or a password or even accept a wrong answer to these qualification techniques. Therefore, no one can log in without providing the correct authentication response. This design, known only by the provider, will assist in preventing others from gaining unauthorised accesses to the VMs.
- The SPEEDOS system provides a single persistent virtual memory. Such a persistent virtual memory will save the cloud users' data automatically and isolate their data from those of the other users.

To give a brief picture to the cloud clients who will be running their accounts on the new SPEEDOS platform, some of the connections between the security features that SPEEDOS provides and the accompanying benefits that they provide will be shown to the clients of the cloud. Some of these benefits are outlined below.

- The clients will be provided with the same environment that is currently provided by other cloud providers, but will be given the use of SPEEDOS to implement this environment to add more security, compared with the conventional provision. As explained in this chapter, the SPEEDOS system separates the application data from each other and provides additional methods to check on the calls on the data in a way that the messages passing between modules cannot be accessed by unauthorised users. Typically, this environment can only be accessed by clients through their use of client-server interactions, or a browser, or a thin client application that provides a user interface.

- The clients can design their own bracket routine to represent their login/out to their VMs in any way that they like (for example, giving a wrong answer to a password or a math question) to make it more difficult for the hackers to access their data or other resources.
- The client's data will not be lost during execution, as the persistent virtual memory of the SPEEDOS host will save the data persistently; even if the power is lost, nearly all of the data are saved.
- All the VM data at rest are saved in segments, protected by the capabilities and the SPEEDOS kernel designed to protect them. These segments cannot be accessed unless a matching capability is provided.

The SPEEDOS system can be used as a secure operating system to host cloud computing that will enhance the protection of data. It has been proven to be a robust design that contains a solution for most of the security issues faced in cloud computing. Therefore, building a cloud platform on top of SPEEDOS will achieve the objectives of this project, provide the opportunity to overcome the challenges presented, and allow an evaluation of the data protection in the new environment.

Chapter 4
Designing cloud computing over SPEEDOS

4.0 Introduction:

Cloud computing systems rely on conventional systems that have been proven to be breakable [4] as witnessed from the examples discussed earlier in Section 2.9. The academic and the IT industries have introduced a number of solutions to fix these issues [79], [80], [85], [86]. However, the offered solutions are for the known attacks and based on using conventional systems that hackers have been working on for many years, which makes it easy for them to break it again [4].

SPEEDOS is a system that can be considered a secure OS because of the fact that it has been built from the ground up to fix all the protection problems found in the other OSs, taking into account the advent of the Internet that connects multiple computers and users from different regions [8]. The SPEEDOS architecture comes out with a number of security/efficiency benefits for cloud customers and organisations, as described in Section 3.12. Therefore, having the cloud computing running over SPEEDOS will definitely enhance the cloud computing protection system and secure the cloud's data from intruders' actions.

The SPEEDOS OS has been written in the Timor language that supports the SPEEDOS concepts [98]. Cloud computing uses a mix of conventional languages, such as C++, C#, Java and Oracle to provide its services [99]. To get the clouds working with SPEEDOS, we

need to find a way to restructure all the cloud services to operate in the SPEEDOS environment.

Building the cloud computing on top of an OS that has never been used to provide cloud services requires a deep analysis of how to restructure all the cloud components to work with the SPEEDOS system, taking into account the efficiency, cost, effort and time. This chapter discusses/analyses the opportunities of designing the cloud system in the SPEEDOS environment, considering the calculations required for reducing/preventing security issues and threats discussed earlier in Section 2.7. This chapter covers all the requirements to move the cloud to the SPEEDOS system [100], proposes solutions and discusses the effects of using the SPEEDOS protection system on cloud computing.

4.1 Representation of cloud modules in SPEEDOS environment

Applications are represented in conventional systems as programs and data files in separate units [101]. The SPEEDOS system replaces the data file with information-hiding, object-oriented abstract data types that create the opportunity of basing file protection and security on the semantic operations associated with the file [101]. Hence, a program is represented as a module with multiple code entry points and a hidden persistent data structure.

Applying the SPEEDOS modelling architecture to cloud modules requires the following three steps [89]:

- application of the information-hiding principle,
- application of the capabilities linking mechanism and
- application of the qualifiers system.

4.1.1 Application of information-hiding principle to cloud modules

The application of the information-hiding principle to any cloud module requires the re-structuring of the module to have multiple semantic functions and hiding the data, thereby preventing certain parts of a class or a software component from being available to its client or expressing an arrangement to protect different parts of the program [95], [102]. The following example explains how to apply the information-hiding principle to a conventional bank system.

Example: modelling a bank system

A conventional bank system structure is shown in Figure 4.1 [101]; here, each user has an interface of semantic functions used for interacting with a bank database. The interfaces have to be previously defined in a list file structure to permit use by the users, distinguishing which users can access these interfaces and which cannot.

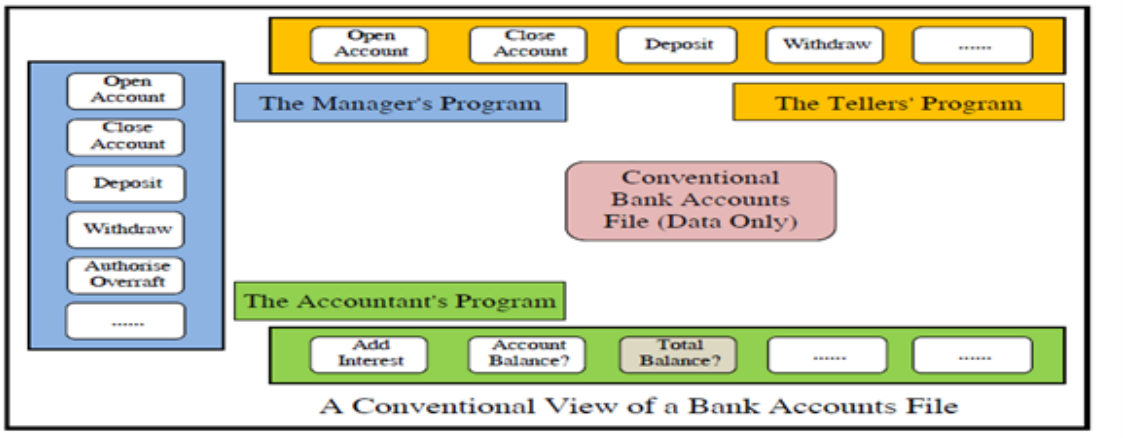


Figure 4.1: Conventional bank system example [101]

The same operation often appears in several interfaces for multiple bank modules with different protection requirements. In conventional systems, the semantic operations

associated with a particular human activity or work role are collected into a single program because the protection mechanism applies to the entire program not to the program interfaces. In this approach, programs are designed not to have control over semantic operations. However, this is not true in the modelling design presented by SPEEDOS.

Banking systems have historically undergone a series of technological changes [101], and each of these technologies has mainly copied the same interface functions to interact with the bank file data. The separation of software into monolithic programs and data files is not adequate as a structuring tool [101].

To address such issues, the information-hiding principle was used to build modules in SPEEDOS. The nature of SPEEDOS approaches associates files with their semantic operations, which have separately defined code entry points. Together with an internal data structure (see Figure 4.2), this arrangement implies that the data are hidden and no one is allowed to access them without presenting the access rights to the appropriate interface routine.

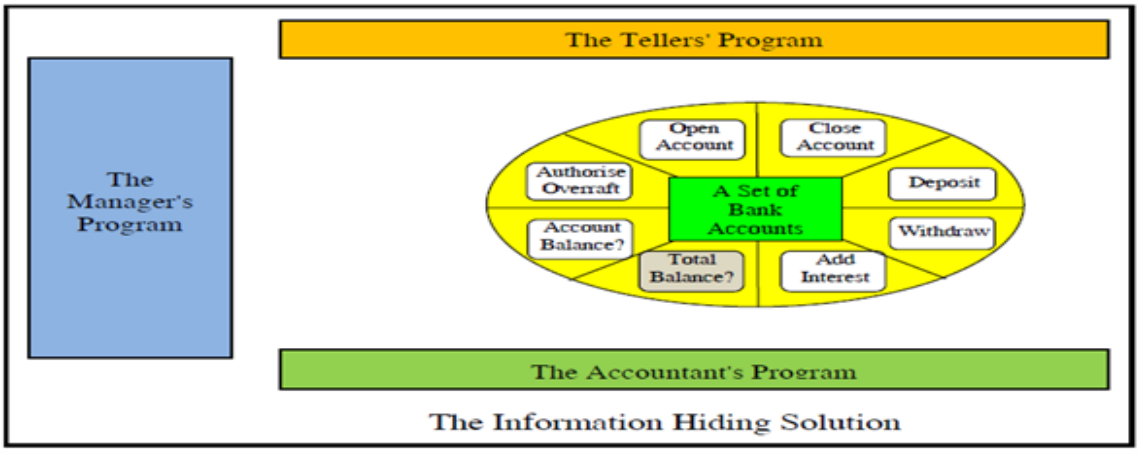


Figure 4.2: SPEEDOS bank system example [101]

This design structure can be used to implement cloud software resources, such as a module with multiple code entry points and a hidden persistent data structure. This implies a strong protection that is not defined from the perspective of all the applications, but in terms of the right to call the operations of the modules.

The access right of invoking a semantic operation of another module, and its right to make the call, is checked by the SPEEDOS kernel. The implementation of these checks can be based on capabilities rather than traditional access control lists.

In practice, to apply the information-hiding concept to cloud computing applications, we first need to restructure each application and apply the concept to the application classes, methods and variables. Certainly, the current programming languages support information hiding; Figure 4.3 shows an example of how to apply the information-hiding principle to a bank account at the method level. Here, the ‘CloseAccount’ method is open and alternate methods are protected. Any calling code can instantiate ‘BankAccountProtected’, yet it can just call the ‘CloseAccount’ strategy. This provides the programmer with protection from somebody invoking the behaviour of the object in unwanted ways. However, the SPEEDOS implementation applies the information hiding at a system level to all the modules including the OS modules that represent the OS services.

```
//Sample information Hiding princple program
using System;

class BankAccountProtected
{
    public void CloseAccount()
   {
        ApplyTax();
        Widthdrawal();
        Deposite();
   }
    protected virtual void ApplyTax()
   {
     //deduct from account
   }
    protected virtual void Withdraw()
   {
      //deduct from account
   }
   protected virtual void Deposit()
   {
      //deduct from account
   }
}
```

Figure 4.3: Information-hiding code example

4.1.2 Application of capabilities linking mechanism

A capability as explained in Section 3.7.2 indicates the access rights that represent what the capability holder can do on a module. SPEEDOS uses the capability mechanism to represent the appropriate authority to invoke a semantic routine of a module. These authorities can be used by the SPEEDOS kernel to accept/reject the requests. The kernel uses the authorities to perform several checks on their validity and privileges that help in making decisions on the calls.

A call to a cloud module's semantic routine can be restricted to a valid capability by re-structuring the cloud module to apply the information-hiding principle and the access rights to any semantic routine of this module represented in a capability. A good programming language can provide such a mechanism at the method level; Figure 4.4 shows how to apply the capability of the bank account shown previously in Figure 4.3 to some semantic routines.

```
void main(){
        deposit(50);
        withdrawal(10);
    }
boolean getAccountInfoByCapability(username,account){
        if(hasCapability(username)){
            return account;
        }
        return null;
    }
 protected void withdrawal(Double amount) {
        account = getAccountInfoByCapability(username)
        if (account) {
            account.balance -= amount;
        } else {
            Console.WriteLine("False Capability");
        }
    }
    protected void deposit(Double amount) {
        account = getAccountInfoByCapability(username)
        if (account) {
            account.balance += amount;
        } else {
            Console.WriteLine("False Capability");
        }
    }
```

Figure 4.4: Application of capability access rights programming language

Figure 4.4 provides an example of a bank account restructured to have its semantic routines accessed only if a valid capability is presented. This mechanism determines who will be able to execute the withdrawal and deposit semantic routines.

SPEEDOS applies information hiding and capability to modules at the system level where all the SPEEDOS modules hide data and prevent access to the data without a valid capability. The SPEEDOS kernel plays the main role in this design as it checks the capabilities of each call and performs decision-making on the basis of these checks.

4.1.3 Application of qualifier system

The qualifier system, as explained in Sections 3.8, is a module that contains a list of qualifying methods that can be used by the SPEEDOS users to apply an additional check on the access to their module data. SPEEDOS supports several forms of qualifiers, as

discussed in Section 3.8. In some forms, the SPEEDOS users can feed a semantic routine of a qualifier with specific instructions to indicate the rights to access their module. Here, when their module gets called by any other module, the callers are required to pass these instructions to get their call granted. However, the use of qualifiers for modules that might contain sensitive data is optional for the users.

Cloud modules can apply this mechanism to add security against granting unwanted access to their module data. Here, the content of the module cannot be accessed without providing the right answers to the checking queries in the qualifier module. In the current computer programming languages, qualifiers can be applied at the application level by redirecting the calls of a method to another method that contains instructions that perform authority checking on the executions. Figure 4.5 shows an example of how to qualify the user information for the previous bank account example shown in Figure 4.4 before executing a specific routine, where the qualifier contains instructions that check the user identity by using certain questions and returns the access approval to the application. The withdrawal and deposit functions are restricted to users who only pass the qualifier instructions and have a valid capability.

The qualifier system can enhance the security for cloud applications as the cloud applications share the memory with many other applications in the data centre. Therefore, skilled users of other applications might use the shared memory to gain access to a cloud application and access certain private data. However, the application of qualifiers might prevent the users from gaining access to the application routines that the qualifier applied to.

```
void main(){
        deposit(50);
        withdrawal(10);}
    //to check if operation is secure depends on user info & some questions & capability
    boolean checkUserInfo(){
        boolean answer1 = userQuestion1.answer;
        boolean answer2 = userQuestion2.answer;
        boolean answer3 = userQuestion3.answer;

return answer1 && answer2 && answer3
    }
 boolean getAccountInfoByCapability(username,account)
 {
        if(hasCapability(username))
        {
            return account;
        }
        return null;}
protected void withdrawal(Double amount) {
        if (checkUserInfo() == true) {
            account = getAccountInfoByCapability(username)
            if (account) {
                account.balance -= amount;
            } else {Console.WriteLine("False Capability"); }}
    else {Console.WriteLine("Unauthorised User");} }
protected void deposit(Double amount) {
        if (checkUserInfo() == true) {
            account = getAccountInfoByCapability(username)
            if (account) {
                account.balance += amount;
            } else {
                Console.WriteLine("False Capability");
            }
        }else {
            Console.WriteLine("Unauthorised User");}
    }
```

Figure 4.5: Application of qualifier and capability access rights programming language

However, SPEEDOS builds the qualifiers on the basis of a system level where modules can be referenced to their qualifying lists and the SPEEDOS kernel that links them, and actions are taken on the basis of the results of the qualifier instructions along with the capabilities.

4.2 Communication with cloud modules in SPEEDOS environment

Modules require services from other modules to accomplish their tasks [103]; SPEEDOS represents modules as a module and multiple semantic routines, as explained in Sections 3.8 and 4.1. Each routine's data are isolated from the other routines' data and protected by the capability and possible qualifiers. For modules to communicate with other modules, SPEEDOS was structured/designed to conduct in-process communication, as explained in Section 3.9.2. As requests from applications for services are handled in the calling process,

each application has a stack process that contains information related to many modules, to invoke the modules' routines while the application process is running.

The application of the SPEEDOS communication architecture at the system level for cloud computing requires the restructuring of all of the cloud applications to follow the information-hiding principle, capabilities and in-process mechanism that performs in-process invocation among modules. However, this requires having a special kernel to schedule the in-process mechanism and apply security at all bases as the SPEEDOS kernel does.

4.3 Storage of cloud data in SPEEDOS memory

SPEEDOS approached another way to structure the computer's main memory, by applying orthogonal paging and segmentation, as described in Section 3.7 [89]. Segments and paging are independent of each other in the program layout, as each segment's start address, length and access rights are recorded in a segment table that supports the prevention of overriding accesses and reduces fragmentation [89]. As described in Section 3.7, the segment tables are stored in dedicated registers called segment registers to be loaded by the SPEEDOS kernel. Each group of related segments is paged and held in an entity called a container [89]. Each container has its own module table and module manager with all the properties of the other modules, and access to their methods is via a capability passed to the SPEEDOS kernel in an in-process call.

The scenario of having cloud data running on the SPEEDOS memory mechanism is similar to that of each cloud application stored in a segment or multiple segments, on the basis of the size of the container in which they are paged and stored. Access to a cloud's segment

data from a module semantic routine requires presenting a valid capability that is checked and matched with the requested segment capability stored in the segment registers by the SPEEDOS kernel in order to grant access.

4.4 Protection of cloud data in SPEEDOS memory

Data protection is applied in the SPEEDOS architecture during communications and storage at the system level; the SPEEDOS protection mechanisms can be summarised as follows [89]:

- The SPEEDOS kernel architecture manages all the required communicating modules and stores their data in the SPEEDOS memory.
- The SPEEDOS modules apply the following:
    - information-hiding principle,
    - capability linking mechanism and
    - in-process invocation mechanism.
- The memory architecture presented by the orthogonal segmentation and paging model is applied to a single persistent memory.
- The implementation of the segment registers addresses the segment details and the segment capability.

The integrated protection system provided by the SPEEDOS system provides a strong defence against the current conventional operation system security problems, described in Section 3.4. SPEEDOS provides better security mechanisms to prevent/reduce all the

network security threats/attacks discussed in Sections 2.7. Therefore, enabling cloud computing over SPEEDOS will provide a secure environment for the cloud applications.

4.5 Provision of cloud computing in SPEEDOS environment

The cloud software implementation was written to have data storage support for the VMs stored in the current conventional operation systems such as the LINUX, UNIX and WINDOWS environments [104]. SPEEDOS applications have been written in their own language Timor.

A considerable amount of software is available to help to build the cloud over SPEEDOS systems; one approach would be to rewrite all the cloud applications in Timor and have all the cloud services execute in the SPEEDOS system, but SPEEDOS is not a general-purpose operating system and we cannot have every program from the cloud running on SPEEDOS. Furthermore, building the cloud applications over a new language never been used to build cloud applications could introduce the possibility of the following effects [105], [106], [107], [108]:

1- complexities of developers' learning the Timor language;
2- introducing more new bugs, troubleshooting and bug fixes;
3- small changes resulting in slower performance, higher resource consumption and more frequent failures and crashes;
4- hardware/driver problems;
5- interface problems/learning curve for cloud customers;

6- System administrators having to learn how to re-manage cloud software resources over SPEEDOS; and
7- the considerable cost and time required.

4.5.1 Complexities developers face while learning Timor

Unfortunately, the current conventional programming languages do not support many of the main concepts of SPEEDOS [98]. Consequently, Keedy and his research team at the University of Ulm decided to design a new programming language Timor [98]. This language was developed to provide a notation for the SPEEDOS security mechanisms [98]. Timor provides an entirely different structural framework for writing programs from that provided in the current conventional object-oriented programming languages [98]. Therefore, cloud developers need to learn the structural framework of Timor and write sufficient useful and practical code to develop a style that takes advantage of the features of this language. In practice, mastering a language is difficult and takes a long time; some languages even take years to master [109]. In contrast, convincing developers to use Timor and prove that it has advantages over the other programming languages that developers have been using for many years increase the related complexity [110].

4.5.2 Introduction of more new bugs, troubleshooting and bugs fixing.

Timor has its own framework to build code and programs as explained in Section 4.5.1; therefore, developers have to learn and work on Timor for a long time to be able to build the vast number of services offered by the cloud at present. As stated previously, learning

Timor may be complex for developers; therefore, certain defects in the code will be introduced and increased [111].

Many factors should be considered as the cause of the defects in a code, such as the following [111]:

- **Human factor:** Nothing can be perfect as long as it made by a human, as humans tend to make mistakes.
- **Communication failure:** Code methods and parties need to be linked to each other to complete their tasks; however, lack of communication or incorrect communication or miscommunication may occur when the requirements are incomplete or indistinct.
- **Unreal development timeframe:** A code may undergo a few tests at different stages, and the testers might be a third party offered the test job with limited time. However, this could result in having the testers work with less understanding of the code and therefore lead to bad-quality and defective services.
- **Poor design login:** Because each project is bound by a time frame, a poor design login can be caused by the time factor as not each programmer may be given sufficient time to follow the improvements of the software developments coming up every day.
- **Poor coding practices:** An error of missing code because of faulty debuggers or poor tools may lead to more defects in the code. Furthermore, such a situation will make it difficult for programmers to detect the error and even more difficult to solve it.

- **Lack of version control:** A number of software systems help the programmers track changes in the code base; however, the programmers have to be sure that the tracking software is up to date; otherwise, new bugs may remain in the code.
- **Buggy third-party tools:** Third-party tools are often used to meet the development process requirements; however, these tools might have their own bugs, which might be reflected in the code and in turn cause defects in the current software.
- **Lack of skilled testing:** A sufficient skill base is required of testers to protect the code from future defects.

Code faults may not be displayed on the wall for the programmers to discover; however, troubleshooting and bug fixing has to be part of each coding process to accomplish defect-free code. The identification of bugs might require very highly skilled programmers with strong knowledge and experience of the coding language. Therefore, the bugs discovered can be solved on the basis of the programmer's experience or attempt to write a different code without changing the main task and restore the product or process to its working state. For now, building a cloud over Timor with no doubts about its complexity for the programmers will introduce a number of bugs; therefore, more time and effort are needed to troubleshoot these bugs and fix them. In contrast, this will add more difficulty to run the cloud over the SPEEDOS system.

4.5.3 Small changes resulting in slower performance, higher resource consumption and more frequent failures and crashes

Cloud computing is built on conventional computer systems that are designed and managed to provide cloud services to cloud costumers online [112]. Running the cloud system over SPEEDOS might affect the performance of some cloud services as the programmers may not be Timor experts and might have written a very buggy code.

In general, software applications filling up permanent storage, using up the temporary storage and using the CPU are poor implementations and could slow down the computer resources and result in a higher consumption of these resources [113].

System failures and crashes might range from hardware failures to software problems. However, SPEEDOS has never been used to provide cloud services; therefore, the cloud hardware and software may be incompatible with the SPEEDOS system. The configuration of the cloud's hardware and software to work with SPEEDOS depends on the programmers and software engineers who need to work with Timor. As there is a possibility that the programmers are not Timor experts, the chances of errors in their programming are high; there is also a possibility of system failures and crashes.

4.5.4 Hardware/driver problems

Cloud computing provides a range of services for users from all around the world, in which they use their own devices to access the cloud services. Users' devices might contain several types of hardware/drivers that are configured for use with conventional OSs.

Changing the cloud environment to work over SPEEDOS requires having an environment that supports all the driver programs and user devices. Computer drivers' are manufactured by companies from all around the world in different languages, and the drivers' communicate with the computer's OS to function properly [113]. Therefore, all computer drivers' need to reconfigured to work with the SPEEDOS OS, or a system library that can communicate with all of the currently used conventional drivers' programs and the SPEEDOS OS should be built. This will add more complexity to having the cloud run over SPEEDOS and might introduce more bugs and incompatible hardware/software, thereby deteriorating the performance of some devices.

4.5.5 User interface problems/learning curve for cloud customers

Computer users rely on menus, control panels or other devices or programs to interact with their machines and tell them what jobs to do for them [114]. With the appearance of the cloud, menus have become larger and are designed more in line with the requirements of the online users [115]. Moving information technology services to the cloud has imposed new ways of interaction with the users [115]. The cloud users are provided with well-designed user interfaces (UIs) that make their job easier and more professional [115].

Obtaining IT services online requires bringing the very intuitive UI of the work area to the web program, while approaching these services pervasively requires dynamic perception and a common connection with the UI. Hence, building a reasonable UI for cloud services is viewed as a critical success in the achievement of the cloud [115]. However, to tackle the challenges of implementing such highly demanding user interfaces, cloud programmers are

required to develop an interactive UI in the SPEEDOS environment that is very clear, simple and easy to use, particularly in critical situations that need quick responses. These designs must meet all the needs of the users and be commensurate with the level of their learning of the new environment.

4.5.6 System administrators' requirement to learn re-management of cloud software resources over SPEEDOS

At present, the administrators of the cloud are working with many resource management software programs that can help their businesses reach new levels of productivity and utilisation [106]. The cloud provides many utilities which can be limited to the SaaS, PaaS and IaaS, as discussed in Section 2.2. These services are provided to the cloud customers through companies such as Amazon [25], Google [14] and Microsoft [30]. These firms have implemented online services for their customers to remotely schedule their business equipment and software resources.

However, restructuring the cloud over SPEEDOS requires the administrators of the current cloud firms to learn how to manage the cloud resources in the new environment. Consequently, the administrators have to understand how SPEEDOS works and put a considerable amount of effort into providing the same cloud utilities in the new environment. In contrast, this will introduce a high-cost investment and a considerably large time requirement.

4.5.7 High cost and time requirements

Cloud computing might not be visible for its reliance on providing services online; however, customers of the cloud are dealing with a range of cloud companies such as IBM [116], Rackspace [117] and GoDaddy [118]. These firms have big data centres that consist of a considerable amount of software and hardware equipment to provide the cloud services to millions or even billions of customers. Such data centres may be very expensive to build and require many years and considerable effort to build. The construction of data centres should consider the following [119], [120]:

- **Site Selection:** The facility location should be chosen carefully and not be vulnerable to any natural dangers such as floods or earthquakes.
- **Utilities:** A decent site that is close to multi-fibre and power providers should be selected.
- **Building Design:** The building design should consider protection from any possible physical attacks from humans or even animals. Furthermore, the building should be able to withstand heavy rains, wind and snow and be constructed from a durable material that can withstand loads exceeding the normal design load.
- **Mechanical/Electrical Systems:** The data centres should be surrounded by multiple power suppliers. However, the equipment of the data centres has to be protected against surges, sags, high-energy transients, brownouts and blackouts, which depends on the conditions provided by the power supplier.

- **Facility Security/Monitoring:** Multiple security systems have to be installed in the data centres to monitor them 24/7 from the tracking of calculations and logging activities to the backup of every movement.
- **Construction Monitoring:** The data centres have to be monitored during the construction stages to ensure that they meet all of the requirements and that there is no defect in the construction or any bugging devices or security breaches.
- **Cleaning:** The data centres must be kept free of dust and dirt.
- **Planning for the worst:** Plans for the future should be considered in the construction of data centres, such as accommodation in emergency situations and storage of adequate quantities of food, water and medicine for emergencies.

In view of all of these requirements and the high costs of building a data centre, we can conclude that it is very costly and difficult to restructure these centres and design their equipment to accommodate the SPEEDOS environment that might require the addition of different facilities.

4.6 Proposed solution

In a previous study on the cloud computing, mentioned in Chapter 2, the provision of cloud computing using the current OSs resulted in many security issues discussed in Section 2.7. This had a negative impact on the cloud users, as illustrated by the examples given in Section 2.9. It also showed that the cloud systems based on the currently used OSs provided hackers with great opportunities to obtain user account information and thus, use these accounts to access important data. A long study of the cloud systems has clarified that the

protection systems of the existing operating systems are unable to protect the important data in cloud computing. Consequently, the SPEEDOS system has been introduced as it is a system built from the ground up to solve all of the current problems found in the conventional system that the cloud system uses to provide its services.

Therefore, the SPEEDOS system was studied and all of the protection systems provided by this system were analysed in Chapter 3. SPEEDOS ideally introduces a powerful solution to the problems of the currently used OSs that host the cloud systems and provides additional security features to cloud users. The potential mechanisms for rebuilding cloud systems over SPEEDOS were discussed in Sections 4.1, 4.2, 4.3 and 4.4. However, the result showed that the cloud transfer to operate in the environment of SPEEDOS might result in more bugs, higher costs and a considerably large time requirement, as discussed in Section 4.5.

Consequently, an effective approach would be to merge the conventional Cloud application implementations by using a secure service. The secure service itself would be developed on top of the SPEEDOS system. Therefore, the security and performance of this cloud service will give results similar to those obtained if the entire cloud service were implemented in SPEEDOS.

4.7 Designing a cloud security service using SPEEDOS architecture

The design of the new cloud security service is based on a previous study on the SPEEDOS system. Therefore, some parts of the SPEEDOS will be adapted from this previous study to design the new security service and to avoid the software engineering difficulties discussed

in Section 4.5. As this design will be based on the protection concepts followed in the SPEEDOS environment and the fact that the design will not change the composition of cloud computing and affect its services.

The new security service design using the SPEEDOS security concepts requires meeting the cloud consumers' requirements and ensuring the protection of their own data and the privacy of their communications. Therefore, the design of the security service applies the SPEEDOS applications model to cloud applications and restricts access to their data to enhance security.

The security service design was built by leveraging the main security features of the SPEEDOS system, in which the cloud applications were restructured without re-implementing them to avoid the software engineering re-implementation difficulties. The main SPEEDOS features built in this design were as follows:

1. design of a cloud application model that applies the information-hiding principle to cloud applications in which a valid capability is required to access the data,
2. design of the capabilities that identify users, user actions, access types and capability metarights that indicate whether a capability can be used more than once or not,
3. design of a qualifying system that grants applications data access, and
4. design of an encryption/decryption mechanism to add more protection to the data during transition and storage.

The security service design is based on the design of a suitable environment to allow for secure communication between the cloud modules and to provide a secure memory that holds all of the private data. The important data will be stored and protected by a memory

that is encrypted and mapped to the security service. Hence, the access to these data is provided only to the new security service and this access is restricted by a strong encryption key.

4.8 Proposed security service design

The proposed security service applies the SPEEDOS security features to provide services more securely and to establish an environment to secure a data exchange amongst the cloud modules and store the communication results in a secure memory.

The new design has a matching environment that can be provided by any cloud system, where users access a UI through web channels to access the data stored somewhere in the cloud. However, the security service design reaches different ways of permitting access to data, where it conducts several checks before granting the access. The access checks of data require a valid capability and for the important data to pass all the queries of a specific qualifier module. Therefore, the data access is based on confirming the users' IDs and the capabilities that indicate their actions. The users' sensitive data will be stored in a memory that can be accessed only by the security service. The security service is designed to apply the qualifying principles on the data of the cloud applications. Each application's modules contain several routine functions that can only be accessed by providing a valid capability. However, the design provides another level of checks on a user's identity before granting the call on a specific routine, by redirecting the user's call on a semantic routine to a qualifier that requires the user to answer several instructions that indicate his/her ID; the calls are approved on the basis of the instruction results. The qualifier modules can be useful in some cases when, for example, the module contains very important data and requires an additional check for the data access. The security service is designed to simulate

a cloud environment that is basically represented in the cloud applications connected to a client interface. However, the security service restructures the cloud applications to provide access only to the data within the scope of the SPEEDOS concepts. Here, the client accesses the interface and requests data from a specific cloud application routine to which the application passes the requests to be checked by the security service. Each client must have a valid capability to grant its call, where the security service checks the call authorisation on the basis of its capabilities. The security service provides the cloud applications with data protection and storage for the user data. It holds access to sensitive data including client authentication/authorisation details. The authentication details consist of identifying capabilities that contain the client information provided when the clients first sign up/register to the cloud applications. The authorisation details are represented by the authorisation capabilities that consist of a client's access permissions to a cloud application's data where these capabilities are restricted by the capabilities' metarights that indicate the validity of the capabilities. The cloud applications and the security service communicate with each other through private secure connections to check the clients' authentication/authorisation details and to update them.

4.8.1 Authorisation/authentication flow

The proposed authentication/authorisation flow is explained below:

1. A client requests connectivity through a cloud application that passes the request to the security service.
2. After validating the client's details, the security service provides an identifying capability to the cloud application that verifies the parameters to permit access.

3. The client obtains authenticated access to the cloud application and starts calling operations of the cloud application. The cloud application receives the call, checks the request and passes it to the security service to determine the authorisation capability.
4. The security service checks whether the client has a matching capability to perform this request. If so, the security service grants the call and passes the approval permission to the cloud application to fulfil the request.

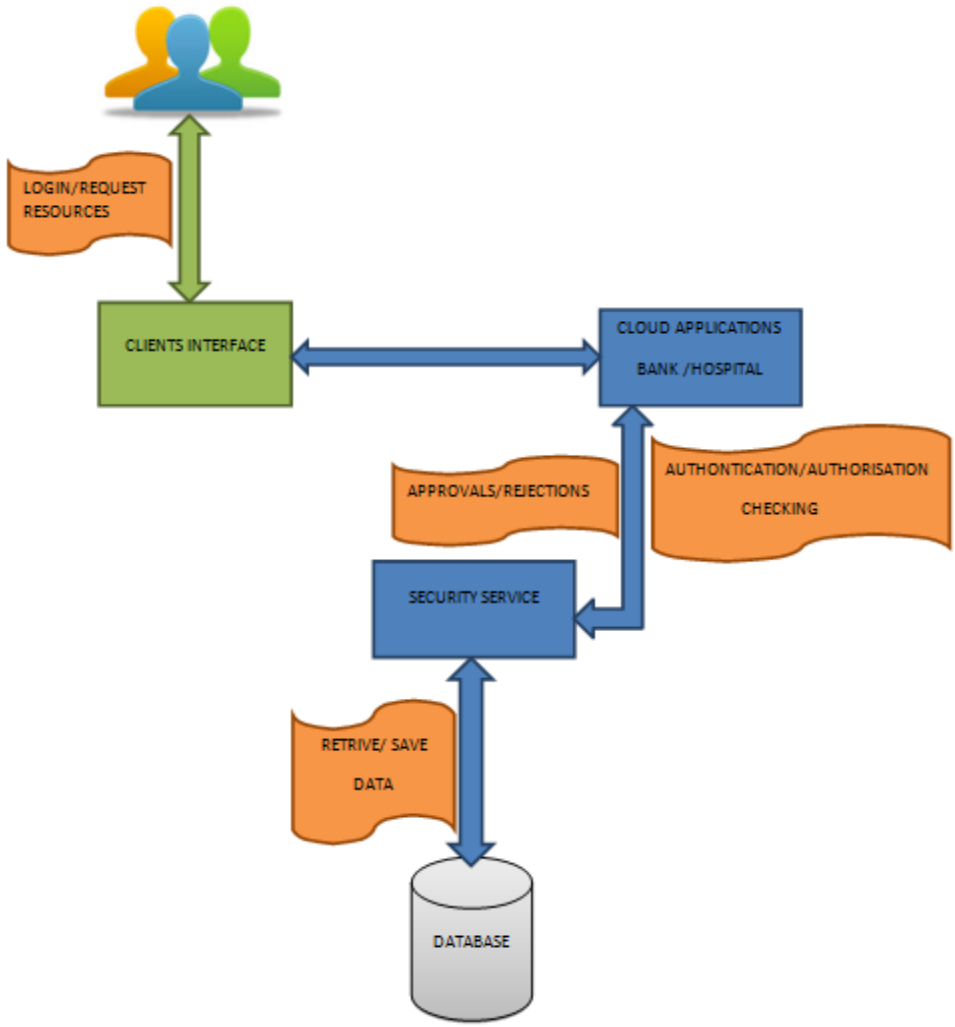


Figure 4.6: Platform diagram

Figure 4.6 shows the overall design and communication flows between a client, the cloud applications and the security service. The client obtains access to a cloud application by providing the login details; the cloud application formulates these particulars and passes the request to the security service. The security service checks the details provided, and if they match with the details stored in the database for this client, it provides a capability that identifies the client to the cloud application. The cloud application checks the identifying capability and obtains authentication to the client. The identifying capability identifies the client each time he/she makes a call to the cloud application.

Once the authentication is obtained, the client must have a capability that permits his/her request of the cloud application. Every time the client requests a resource of the cloud application, the request gets transferred by the cloud application to the security service by the client identifying capability and the request details are checked against the client authorisation capabilities. The cloud applications and the security service communicate through private channels to check the client's authentications/authorisations and to update the corresponding details.

The proposed security service is a general software framework that uses the SPEEDOS security features to implement any cloud software application without re-implementing it, in terms of ensuring protection to the application's routine data during transit and storage. The design implements multiple robust security structures, including the following:

- user authentication,
- user authorisation,
- ACL and capability implementation,

- encryption of critical data, and
- design of cloud modules to achieve flexible security, including the following:
    - capability revocation,
    - access control lists,
    - access monitoring, and
    - restricting allowable parameters

The security service is designed in such a way that it provides an environment that supports the services offered by the cloud systems, while preventing the attacks and threats that might affect the efficiency of the services provided, taking into account the cloud security threats described in Chapter 2.

4.8.2 Proposed security scheme

The security service is inherently designed and implemented using the SPEEDOS architecture to control the cloud module transactions and secure the sensitive data. The design implements a distributed capability-based system which restructures the cloud applications to provide a secure authentication/authorisation environment to the cloud providers and clients. The authentication/authorisation is represented by the capabilities that are stored and encrypted in the database mapped only to the security service.

The design assumes that the clients/cloud applications communicate through a secure, third party. The security service is the mediator between these parties and strictly controls the users' orders from the cloud applications on the basis of their capabilities.

4.8.3 Security service architecture

The security service architecture is designed to apply security techniques that protect all aspects of the cloud applications' data. This is represented by the structuring of the cloud applications to apply the SPEEDOS design model to applications by following the information-hiding principle, capabilities and qualifiers for important data. The design of the cloud applications in this approach requires having an intermediary that confirms the identity of the caller to these applications and is based on the capability. The security service plays an intermediary role by checking each access to the applications' data and confirming the caller's identity against the details stored in the database.

The security service checks all the calls to the data and confirms the identity of the caller by using the identifiers represented as the capabilities that identify the caller and the permission details. The security service receives calls from cloud applications to validate the clients' requests and match their authentication/authorisation details with the capabilities encrypted in the database. If the access is granted, the security service allows the client to call the cloud application function until the capability metarights indicate that this capability is not valid for more usage (specified in the capability) and then stores the changes to a capability in the database. If the check fails, the security service takes actions to reject the call.

The security service architecture design re-manages the cloud applications to apply access to the module routine data by providing only a valid capability. These capabilities are all encrypted, and only the security service has access to them. The security service architecture is used for designing cloud applications, capabilities, qualifiers and the

encryption/decryption mechanism, and for determining the general responsibilities of the security service.

4.8.3.1 Design of cloud applications

The cloud data within this solution are abstracted as modules so as to follow the information-hiding design explained in Section 4.1.1. The modules contain operational functions and data, which are hidden and can only be accessed if the right capability is presented. Calls from clients come straight to a specific function of a module, and the acceptance or rejection of this request is based on the security service checking the stored capabilities for the client.

Cloud applications such as social media, hospitals and banking systems can be built using this design by applying a checking mechanism that accepts executions on a page only when a valid capability is provided. This design provides a robust structure and control over the semantic operations of the cloud applications. For example, bank security is enhanced by ensuring that the right to call some semantic interfaces of a bank account file does not depend (only) on the identity of the calling user but also (or only) depends on the identifier of the code module accessing the interfaces, thus ensuring that these calls are from a legitimate program, rather than that being used by a hacker.

This design will also facilitate the application of other types of protection; for example, a protection system could be applied to an application routine that examines the identity of the callers before executing their request. The protection system can direct the caller of the application routine to other routines and contains instructions that confirm the caller ID and on the basis of the results, grants the call. This can be applied to modules to check the

access permissions, in which each call to the module routine gets qualified before granting the request.

4.8.3.2 Design of capabilities

Capabilities are the main features used to provide security in the proposed approach. As explained in Chapter 3, capabilities are a collection of keys given to users that allows them to operate on the module functions for a particular number of calls. All the capabilities are accessed only by the security service to check the users' authorisation/authentication.

The capabilities are used to identify users and their permitted actions, where the user information is stored in the identifying capabilities and the user authorisations are stored in the authorisation capabilities. The identifying capabilities contain all the user details provided when the users first register to the cloud application, such as usernames and email addresses. The authorisation capability structure defined in the security service represents the users' authorisation permissions. It consists of a protected data structure that provides access to the cloud application data.

Figure 4.7 shows the authorisation capability structure that consists of the actual user, user actions, access type and capability metarights that permit the user actions on a specific module function for a particular number of calls.

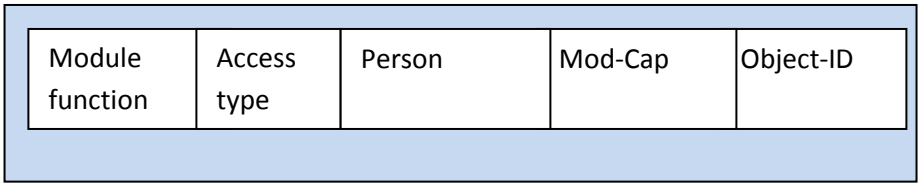


Figure 4.7: Structure of capability

- **Module function** represents the module function name
- **Access type** represents the permitted action (read, write or read-write)
- **Person** represents the actual user
- **Mod-cap** represents the capability metarights (capability revocation)
- **Object-ID** represents a unique module ID in the database

The capability is designed to have a mod-cap that controls the permission and is controlled only by the data owner. The mod-cap metarights indicate whether the capability can be used for once or more than once or is not valid anymore; it also allows capability revocation which might be difficult to achieve in other solutions. The metarights can be set to be invalid by the data owners anytime. The data owners can benefit from this design as they can provide capabilities to clients and revoke them after any sign of suspicious activity.

Capabilities cannot be transferred to other users without the data owner's permission. Obviously, when users receive a capability, they are allowed to use it for the particular metarights specified by the original owner of the capability, but they cannot extend its metarights or transfer it to the other users without obtaining permission from the same owner. Each capability within the system contains a unique ID used by the security service every time the user makes a call. If the user invokes the transfer capability method, the transfer only gets approved by the security service if the owner of the capability has been authorised by the original capability owner to transfer the capability.

Every time a particular capability is used, the metarights are decremented in the security service. When the metarights are invalid and not renewed by the data owner after a certain

period of time, the security service erases the database of garbage keys to save the database from garbage collection. The erasing date/time can be set up to suit the requirements of each cloud application that uses the security service.

4.8.3.3 Design of encryption/decryption mechanism

The authentication/authorisation details and all the other sensitive data are represented as the capabilities encrypted and stored in a separate table within the security service. These important pieces of data do not have to be stored in the cloud applications database, which may be accessed by many applications. Instead, they are stored in the security service database that is only accessed by the security service. The users cannot access the security service; they only interact with their cloud application that passes their requests to the security service to provide the application with a confirmation of the clients' authentication/authorisation details.

The security service applies a robust encryption mechanism to all of the user data for both storage and transmission as will be explained in detail in Chapter 5. The encryption/decryption mechanism is designed to rely on a key that performs the encrypt/decrypt process; however, this key is generated and hashed using a one-way hash function implemented in the security service.

Each user of the security service has a unique key. This key is generated as a random value on the basis of the users' table by using an algorithm that uses the user details and additional specific keywords. These details are hashed in a 56–72-character-long string and stored in a separate data table that is accessible only by the security service. Each time the

security service demands to retrieve/edit/save to a capability, the security service retrieves the key and decodes it for use in the encryption/decryption process.

Here, the same hash function is used to sort all the security service users' passwords, to which no access is granted to the security service without providing the matching password. As one key of these passwords would be difficult to guess (72^26 = 1.9527422e + 48, that is, if only letters are used), the addition of a specific keyword to the hashed details makes it very difficult to break the password as explained in Section 6.2.6.

4.8.3.4 Security service responsibilities

The primary responsibilities of the security service are as follows:

- **Generate authentication capability**

User authentication is part of the security service implementation. Each user is provided with a registration form that is used to permit access to cloud applications on the basis of alternate authentication mechanisms such as username/password or question/answer. These details are provided by the cloud applications to be kept encrypted in the security service database, and only the security service can access them.

- **Assigning authorisation capability**

The security service is responsible for creating authorisations for users by providing capabilities on the basis of valid user IDs. The capability is only awarded to clients by the data owners where the data owners call a specific method to generate a new capability to authorise another user on their module for a limited number of uses specified in the

capability metarights. The data owners need to provide details that identify the capability and the user. These features are as follows:

- **Person:** the actual user receiving the capability
- **Module function:** the operation function that the person is being provided with access to
- **Object ID:** the data owner module ID
- **Access type:** identify the access type (Write, Read or Write-Read)
- **Mod-Cap-Number:** metarights access details

Once all of these details are provided, the cloud application passes these particulars to the security service to create a new capability for the user to be used on the data owners' account for the particular metarights specified and can only be extended on the basis of the data owners' approval.

Only data owners can provide users with a set of capabilities to interact with their module functions. Furthermore, users who have had a capability for a specific module function cannot use this capability after the capability metarights indicate invalid unless the owners renew it. This approach to authorisation gives a chance to data owners to create and control their access list, listing the users who have been given access to their modules' data.

- **Disable/transfer/enable capabilities**

A data owner has more control features available to perform on the capabilities provided to other users, such as the ability to disable/enable a capability or to transfer a capability. The disable/enable capability features are only available to the data owners. These options allow

the data owners to block users from accessing their data or enable users who have had a capability to access their data disabled to use it again.

If the users wish to update their capability to be used again, it needs to be actioned by the data owners. If the security service receives the owners' approval, the user capability will be modified for use for the particular metarights specified by the data owner.

The transfer capability feature can be called from any user who has the permission from the data owner to transfer a capability. When a user has the permission from the data owner to transfer a capability to another user, he/she has to provide some details to complete the process, such as capability ID and the ID of the person receiving the new capability. The capability ID is a primary key generated in the database for a capability, and the capability owner can retrieve this key anytime.

- **Disable all capabilities**

This method is used by data owners to disable all the users' capabilities from operating on their modules. When this method is called, the security service checks the user details sent by the cloud application, and if they are a match with the owner details, then it prevents the users from operating on the owner module immediately. This option can be useful for reducing malicious damage, where the data owners can use this option if they detect any suspicious activity.

There are more operations available for data owners to perform on the capabilities. The data owner can use these features to do the following:

- **Enable users to assign capabilities**

This process can be called by a data owner to allow another trusted user to assign capabilities on behalf of the data owner. Here, the user can provide authorisation

capabilities to clients for use on the data owner's modules. Such an option could be helpful in some cases as this security service is meant to support all types of cloud applications, for example, bank or hospital software as the users might have health difficulties and need a trusted person to control their accounts. However, this option could be revoked anytime as explained below.

- **Disable users' assigned capabilities**

This operation disables the users who have been given authority to assign capabilities on behalf of a data owner, and this process can be called only by the data owner who owns the module.

- **Check capability validation**

A user must have the capability to call any application operations. The cloud application passes calls coming from the users to be checked by the security service. The security service checks all the requests and matches them with the user authorisation capabilities stored in the database.

The cloud application provides the security service with the caller details and the details of the call. The security service checks whether this information matches with the user authorisation capabilities or not, whether the user owns this capability, whether the capability is active and whether the capability metarights to perform this call are valid. On the basis of these checks, the security service sends the call approval or rejection to the cloud application.

- **Passing requests between cloud applications and security service**

When the user calls one of the cloud application methods from the users' interface, a specific service carries out this call with the user details and transmits all aspects into a byte stream. An HTTPS channel is then generated to pass the transmitted data to the security service. The security service receives the request represented as a byte stream and rewrites it in the original object's format to process the request. The security service matches the user details and the operation requested with the authorisation capabilities stored for the user in the database. If the checks are successful, the security service returns the approval through the HTTPS channel to the cloud application to perform the requested call. Figure 4.8 depicts the communication flow.

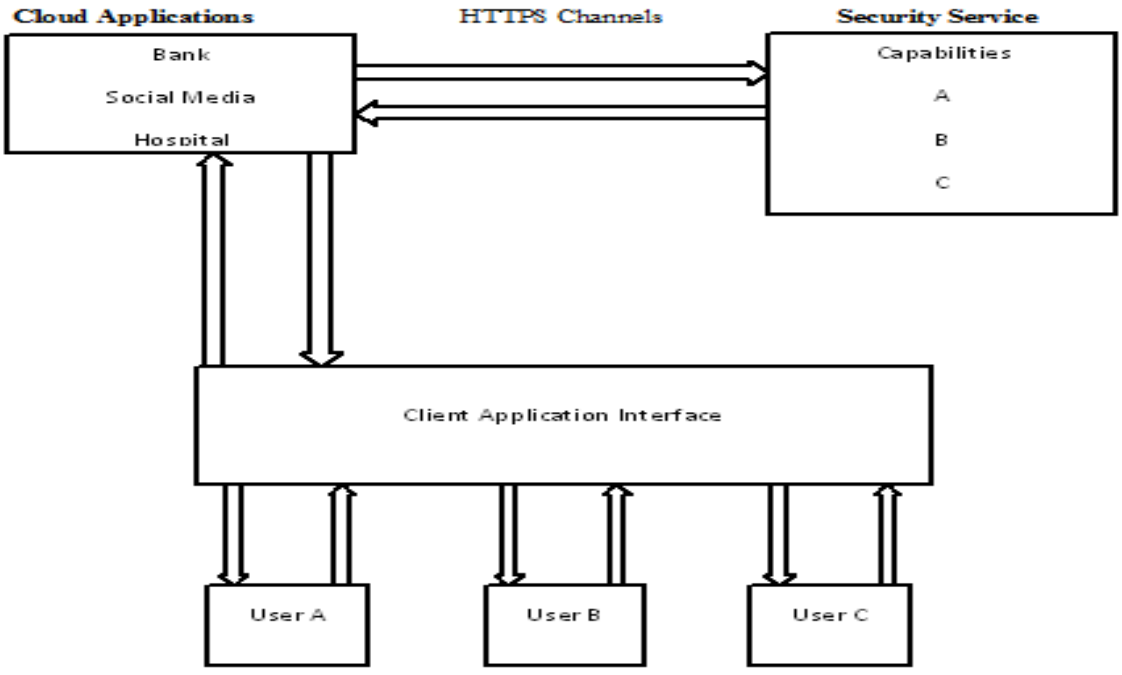


Figure 4.8: Platform communications diagram

4.8.4 Client application

The client platform contains an interface for the clients to access and interact with cloud applications. Clients make requests for their cloud application which passes the calls to the security service to be validated. The cloud application passes identifiers to the security service; these define the caller, cloud application and the application function to be called. If the clients have authorisation for such requests, the security service replies with the approval of the cloud application that responds to the client by performing the call and replying with the results. If the client is not authorised, an error is returned instead. If the security service returns a positive result, then it is the cloud application's responsibility to ensure that the capability is only used to grant the end user access to the requested function. This mapping of the capability to the function provides a level of protection similar to that provided by the capability to memory mapping functions of SPEEDOS, which ensure that the buffer overflows and other memory-based attacks are not feasible [89].

All the cloud applications are designed to perform calls only for users who have valid capabilities. The caller has limited access to application functions if the access is granted. Only the data owner, or whoever has the appropriate capability, can access the data behind the application functions.

If a client wishes to authorise another client to operate on their account in one of the cloud applications, they must provide them with a capability. The capability can be provided by calling the assigning capability method, as described in Section 4.7.3.4. This method contains some details that are required to be completed by the client giving the capability. Once this is done, the other client can start using the new capability to perform a specific

operation on the client account for the particular metarights. See the example described below.

- **Authorisation example**

In this example, we have a bank system that uses the security service. The bank has two users (A, B), and they are permitted to perform bank operations such as withdrawals, deposits and balance requests. If user A wishes to give user B the authority to withdraw $100 from user A's account three times, then the following operations are performed:

1. User A calls the 'assigning capability method' on the client interface and provides the required data to assign a capability for user B to withdraw $100 out of their account up to three times.
2. The information provided will be sent out to the security service by using the HTTPS-generated channel writing all the data into a byte stream format.
3. The security service rewrites the information received and checks the identity of user A against that stored for the user in the database, determining whether this user can pass the capabilities or not to the account. If the check passes, the security service will create a new capability for user B for the particular metarights specified for user B, which in this example is three times.
4. The security service encrypts the new capability and saves it for user B in the database. Now, user B has the capability from user A to withdraw $100 from user A's account for the particular metarights provided by user A.
5. If a request is received from user B on the account, the bank cloud application sends user B's request to the security service that checks it and approves it to the bank. The security service decrements the metarights of the capability that can be used by user B.

6. User B can continue working on user A's account until the capability metarights become invalid.

The example above shows an authorisation design which can provide a secure method for Internet shoppers using the security service to purchase products without the need to provide their credit card details. Instead, they provide the seller the permission to withdraw a specific amount of money from their bank account once.

The clients who have had a capability cannot use it after the capability metarights become invalid as it needs to be reauthorised by the data owner. The data owners can disable any capability or all the capabilities that are given to the other users on their accounts anytime they want to in case of a malicious activity.

4.9 Main methods of security service

The following table shows the platform methods and parameters:

Table 4.1: Platform Methods, Data Inputs, and Outputs

Security service methods	***Return value***	***Parameters***	***Method job***
disableCapability	*Boolean (returns yes if authorisation on requested check is successful and if not, returns no)*	*(Long ID)*	*Stops users from accessing specific module function data*
transferCapabilities	*Boolean*	*(Long ID, Long person ID)*	*Transfer authorisation capability to other parties*
enableCapabilities	*Boolean*	*(Long ID, Integer number)*	*Enable all previous users' authorisations on specific module*
disableAllCapabilities	*Boolean*		*Disable all users'*

			authorisations on specific module
hasCapabilityAccess	*Boolean*	*(Object Related ID, Long person ID, String module Name, String function, Access Type access Type)*	*Check an authorisation's validity for a specific user on a specific module*
searchCapabilitiesBean	*Paged List (serialised object)*	*(Map params)*	*Search authorisation information for specific user*
getCapabilitiesBean	*Capabilities Bean?*	*(Long id)*	*get the primary key that authorises user on an account*
saveCapabilitiesBean	*Capabilities Bean?*	*(Map params)*	*Save updated information for user authorisation*
Method to identify module and module function			
getModuleBean	*Module Bean*	*(Long ID)*	*Return called module*
getModuleFunctionByName	*Module Function Bean*	*(String name)*	*Return called module function name*
getModuleByName	*Module Bean*	*(String name)*	*Return called module name*
searchModuleBean	*List<Module Bean>*	*(Map params)*	*Search for module info*
searchModuleFunctionBean	*List<Module Function Bean>*	*(Map params)*	*Search for module function info*
Method to identify users			
getPersonBean	*Person Bean*	*(Long ID)*	*Return specific user id from database*
getPersonBeanByUserName	*Person Bean*	*(String username)*	*Return specific user info by name*
searchPersonBean	*List<Person Bean>*	*(Map params)*	*returns users list info from the database*
getUserInfo	*Person Security Bean*	*(String username)*	*Returns users info from database*
More methods in the client platform to control access			
isObjectOwner	*Boolean*	*(String module, String object Unique Id)*	*Check on ownership of specific user*
setObjectAdminAssignCapabilities	*Boolean*	*(Long object Unique ID, Long*	*Allow admin to assign capability*

		module ID, Boolean can Assign = Boolean.TRUE)	*to specific account*
LoadObjectByCapabilityAccess	*Object*	*(System Module Operation module Operation, Object object Unique ID, Long person ID, Access Type access Type)*	*Check on any request from the server based on capability.*

4.10 Why this system should be more secure than a traditional system

The security service represents a middle ground for the cloud environment. However, the security service design is inherently secure for the following factors:

- All the databases are encrypted and stored in a memory that can be accessed only by the security service.
- The decryption keys are hashed and stored in an industrial database, in which only the security service can decode/access the keys.
- If the security service is secure, then the data is secure. The security service prevents interceptions by communicating only with the cloud applications through private channels, and these data are encrypted during the transition process so that even if the data get compromised, no benefit of these data can be taken without decrypting them.
- The security service design applies the SPEEDOS security mechanism on cloud applications without re-implementing them, which presents no more bugs, risks and troubleshooting.

- The security service provides the ability for users to control access to their data in the cloud environment.
- The security service providing the ability to the users to perform capability revocation on their data for any suspicious activity.
- The security service provides an additional checking mechanism on important data by qualifying the caller ID before granting the access.
- The security service provides a secure trading method between cloud customers as they will be able to provide to other parties a capability indicating certain activities on their account without providing any private details that can be misused later.

The security service simulates the SPEEDOS environment for cloud applications. It provides a mechanism to apply the information-hiding principle to any cloud application, in which the modules are designed to have a linking mechanism so that access to the module operations is only permitted when a valid capability is provided. The capability specifies the caller ID, caller actions, type of activities and the metarights of the capability that indicate whether the capability can be used more than once or not. Users who want to perform any request on a cloud application must have the capability to permit such a request, and without this capability, the request will be rejected. The capabilities and the user details are encrypted using a secure encryption/decryption mechanism, and only the security service has access to the decryption keys.

The security service was built to check all the requests coming from the users and to validate their claims with the capabilities stored in the database. There is no way for the client calls to be permitted without having a valid authorisation capability. All the

parameters passed between the cloud applications and the security service are transmitted in a format that is only readable to the security service through private connection channels. The system provides an additional protection level for the cloud data by guaranteeing that only authorised users obtain access to the data. The system has proven to be a reliable mechanism that protects the cloud data against modern security risks, as discussed in Chapter 6.

4.11 Comparisons with other security frameworks

Now, we will examine the presented platform against a number of existing alternate systems. In cloud applications, authorisation and authentication can be provided to users by the Spring framework, as explained in Chapter 2. The Spring framework uses a list that holds the users' permissions defining which users are allowed to operate on which modules. This file is stored in the conventional computer memory, accessible by multiple applications. A malicious application user might take advantage of this access to obtain other users' permissions on the modules and the sensitive data.

In the platform presented in this thesis, the access mechanism provided is entirely different as the access rights are stored as capabilities within the security service. The subjects need to have access rights (capabilities) each time they call an object. Checking the validation of the capabilities determines whether the access is granted or not.

The capability is the key to the system that determines the access validation for users to access system objects. This capability only provides the authorisation for users to operate on the module function for a particular count, and without a valid capability, access cannot be permitted. This offers more confidentiality to data owners, knowing that no one can

operate on their accounts without obtaining their permission. Furthermore, this platform provides data owners with the ability to build access to their data.

SAML is another framework that provides authentication between parties (as described in Chapter 2). SAML authenticates a user on a module of a service provider, passing the user details to an identified third party that is already authorised to contact the service provider. The third party is given the authority to authenticate the user on behalf of the service provider.

In comparison to the mechanism presented in this thesis, this platform includes the authentication/authorisation mechanisms that rely on the capabilities encrypted and stored in the security service database. The applications communicate with the security service to obtain user details when the user logs in, and on the basis of these circumstances, the applications determine the users' authentication/authorisation rights. The capability controls the authorisation to the data by specifying who can access the data and what actions are allowed, and only for a particular count. In comparison, our platform differs from SAML by controlling the access to the applications and the data, limiting the access to the data and all of the users' sensitive details encrypted/stored in the security service database, and the authorisation to the data by individual data owners. The mechanisms used in this platform allow the data owners to build their access list on their data and manage the access to it in a manner that enhances the protection and prevention of unauthorised access.

Finally, Kerberos is an authentication protocol that provides users with tickets that identify users and allow them to communicate with each other over non-secure networks. As explained in Chapter 2, Kerberos's primary goal is to prevent unencrypted passwords from

being transmitted across a network, where users' authorisations for the data are stored within the applications database that is accessed by multiple different users, which in turn puts the sensitive data at the risk of being stolen by non-authorised users. However, our platform provides an authentication/authorisation mechanism represented as the capabilities that authorise the users' actions on applications for a particular count. These user details are kept encrypted in the security service database, and the applications only communicate with the security service to check the user permissions. Furthermore, the capabilities provide more confidence to data owners as they can revoke any or all capabilities on their accounts anytime they want. Access revocation might not be easily guaranteed in other cloud applications, which gives further credit to the proposed security service design.

Chapter 5

Implementation of the cloud platform

5.0 Introduction

Cloud computing systems provide services to users, and these services can be accessed from all over the world by using the infrastructure provided by the cloud. Many clients are using such services and uploading/downloading data to the cloud. This provides a significant opportunity for the data to be misused by malicious users or attackers.

The SPEEDOS system, the module structure and the approach to the communication between the module functions and the methods to protect them were described in Chapter 3. As described in Chapter 3, SPEEDOS provides a secure structure for modules by using capabilities for communication between the module functions. However, providing cloud computing over SPEEDOS has introduced a number of software engineering problems, as described in Section 4.5.

This chapter focuses on the implementation of the proposed security service described in Chapter 4. This security service provides a restructuring design for cloud applications while reducing or preventing the attacks and threats affecting the applications operating in the cloud. The security service is designed to be installed on a SPEEDOS system to offer more security features and protected database storage. However, as SPEEDOS is not a complete system, the main security features of SPEEDOS are designed to be used in the new security service by using an open source database. To show that the security service is secure, we

built it using a conventional programming language (Grails) on the currently used OSs and evaluated it against the cloud threats/attacks to prove that it is inherently secure.

5.1 Grails

The platform can be implemented in practically any language such as C# or Java, but it is implemented using Groovy and mainly the Grails framework [121]. The reasons for doing so will be explained in the following section.

Grails is an open source web framework that was built on the basis of other technologies such as Java, Spring, Groovy, Hibernate and Sitemesh. All of these structures, platforms and database engines are made available inside the Grails platform without any need for completing additional configuration steps.

Grails is a model-view-controller (MVC) framework that contains models, views and controllers to separate concerns. A domain in Grails is the heart of the application and business model concept, and is linked to the database using the Grails object mapping implementation (GORM) that can be used by programmers as a mini language to describe the persistent objects in their application. The domain can be created using the Grails command and is saved in the Grails domain directory. Grails uses controllers to implement the behaviour of web pages, which are central to a Grails application. It can efficiently marshal requests, deliver responses and delegate views. This includes the placement of the title, logo and any additional style sheets. The service contains business logic that can be reused across a Grails application. In Grails, a service is a class with a name that ends in the convention ‘Service’ and lives in the Grails-app/services directory.

Grails security was constructed on the basis of the security included in Java Servlets [121]. Grails provides answers to a broad set of issues and is, to a great extent, invulnerable to URL abuses [121].

5.2 Plugins

The Grails framework is the core collection of the essential plugins in the Grails framework. It represents the primary feature of the Grails application from the beginning. In fact, Grails is essentially an aggregator of its plugins.

Our platform uses the benefits of Grails plugins as discussed below.

- The Spring security core plugin is responsible for the security of the Grails applications. It is based on the Spring security library and has a number of companion plugins for other aspects of security, for example, Spring security REST plugin for REST API security [121]. The Spring security plugin simplifies the integration of Spring security libraries into Grails applications. The plugin provides sensible defaults with many configuration options for customisation. Nearly everything is configurable in the plugin and in Spring security itself, which makes extensive use of interfaces. This plugin supports the authorisation and authentication of any user that exists in the system.
- ACL plugin adds domain object security support to a Grails application that uses Spring security. It depends on the Spring security core plugin. The core plugin and other extension plugins restrict access to URLs via rules that include checking a user's authentication status, roles etc. The ACL plugin extends this by adding support for limiting access to individual domain class instances. The entrance can be very fine-grained and can define which actions can be taken on an object. These typically include

read, create, write, delete and administer. However, it is free to determine the measures required.

- Remote plugin: This plugin makes it easy to expose your Grails services to remote clients via RMI [122], Hessian [123], Burlap [123] and Spring's HTTP Invoker protocol [124]. Further, it is possible to access remote services easily via the same set of protocols.

  The HTTP Invoker [124] protocol is used to carry all the data between parties in this platform. The data are formulated into a digital format that can be carried out as an HTTP request, where the receiver rewrites that digital data back into a readable format to perform the requested operation and return a response. The following two points explain the techniques used to exchange data between parties in the platform.

1- Serialise/de-serialise objects: All the transmitted data are serialised before the transport process [125]. To serialise an object means to convert an object state into a sequence of bytes. The object type and the types of data stored in the object transform into a stream of bytes that can be reverted (de-serialised) into a copy of the object when needed.

Serialisation is used for lightweight persistence and communication via sockets or Java remote method invocation (Java RMI) [122]. The default encoding of the objects protects private and transient data and supports the evolution of the classes. A class may implement its external encryption and is then solely responsible for the external format.

2- HTTP Invoker protocol: HTTP Invoker [124] is used in this platform to post calls between the cloud application and the security service. HTTP Invoker is solely used for convenience, and any remote procedure call (RPC) frameworks such as web services can be

used during the implementation. It is a Spring remote technology that enables RPC over HTTP. It is achieved by passing outbound method invocation represented as serialised invocation by using Java standard serialisation (as explained in the previous section) through an HTTP post to the destination exporter. Then, the targeted system invokes the methods requested and returns the results as a serialised value to the HTTP response. Spring must be used on the server and the client platforms to perform the serialisations.

5.3 Database

Grails automatically provides a database to store the project data. However, the space provided might not be sufficient for big projects. Grails provides the ability to use other database sources such as MySQL [126], Oracle database [127] and H2 [128].

In this project, we chose one of the most advanced free source databases called PostgreSQL [129]. It is a robust database that provides full support for foreign keys, joins, views, triggers and stored procedures (in multiple languages). It includes most SQL:2008 data types, including INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL and TIMESTAMP [129]. It also supports the storage of large binary objects, including pictures, sounds and video. It has native programming interfaces for C/C++, Java, Net, Perl, Python, Ruby, Tcl and ODBC.

5.4 Implementation

The Grails language needs to be set up to start implementing the project. All the steps to install Grails are available on its official website [130]. The databases need to be linked to Grails for us to start developing projects.

After setting up the requirements to build projects on Grails, the implementation process of this project progressed in parts as explained below.

5.4.1 Security service implementation

The security service application contains an implementation of modules, capabilities and an encryption mechanism. The security service is implemented to provide authorisation for client calls to cloud applications. The capabilities model the linking mechanism that authorise clients on the cloud applications. Private channels are implemented to allow communication between the cloud applications and the security service, and the data are encrypted at all the stages.

In this study, the implementation process started with a few configurations to provide our security model. First, a number of plugins were installed that provided the core security features. The two main plugins were installed as described below.

- **Spring plugin**: As described in Section 5.1.2, it is a security core plugin used to support authorisations beside capabilities [121]. To apply this plugin to any application, follow the instructions available at [131]. The plugin is easy to configure, as described later.
- **ACL plugin**: This plugin is used to check the URL access. This plugin is available at [131].

The desired configuration was deployed to the application to link the access to the database and URLs. The following configuration dependencies compiled the plugins in the platform:

Compile'org.Grails.plugins:Spring-security-core:3.1.1'

Compile 'org.Grails.plugins:Spring-security-acl:3.1.0'

Compile"org.PostgreSQL:PostgreSQL:9.4.1211.jre7" To link the platform with PostgreSQL database

Sample users' were configured on the platform, and their authentication/authorisation rights were set up during the application boot strap (first code instruction started when the application was run). The users' created in this platform were as follows:

1. **Admin-Cap user:** security service administrator
2. **Admin user:** cloud application administrator; assigns capabilities on behalf of data owners
3. **User1, User2:** cloud applications clients

A random key was generated for each of these users to be used in the encryption/decryption mechanism. The keys were encoded into the users' table. No access was given to this table except to the security service. The encoding technology used the username details with other specific keywords and applied the standard Bcrypt Password Encoder [132] to encode them. This password hashing system tries to thwart off-line password cracking using a computationally in-depth hashing algorithm, primarily based on Bruce Schneier's Blowfish cipher [132].

Other classes were generated to support the Spring/ACL plugins used. Each of these classes provided authorisation/authentication services as explained below:

- Authority: includes authority for users
- Persistent login: includes login logs
- Person: sets up platform users and keys for encryption
- Person authority: creates authorisation for users
- Request map: stores each authority's authorisation details in a function

The security service is the central core that provides security in our platform as it controls the clients' access to the platform and the data that it protects. The clients request the resources of the applications, and the security service determines whether to permit access or reject it on the basis of the capability details stored for the clients.

5.4.1.1 Implementing modules

The module structure of the platform was designed on the basis of the robust model explained in Section 4.7.3.1. An arrangement that works for all cloud applications and their functions was implemented. The structure that defines modules contains two classes as follows:

- **Module:** a Groovy class consisting of a string name that identifies the platform modules
- **Module Function:** a Groovy class that consists of a module name defining the function owner, a string name that identifies the functions of the module, and an index of these functions in the module database. The index is defined as an integer type and the module name as a string type.

Each module function has command lines that implement specific operations on the module's data. When a client calls one of the functions of a module, a Groovy class that contains each module function's details as a package directs and links the call to the correct module function.

As a proof-of-concept, a bank application was implemented according to our module design, using the capability mechanism to communicate between the clients with bank operations. The bank application provided the principal operations to clients such as

withdrawals, deposits and balance check. The clients could use these operations on the basis of their capabilities for a limited number of iterations.

A social media application is another cloud application implemented as a proof-of-concept to validate the security service protection. The application was designed to provide clients with more options to control their private data with respect to who could view these data by providing the relevant capabilities.

5.4.1.2 Implementing capabilities

The capabilities were applied in this platform as a linking mechanism that connected the clients and the cloud applications, where the clients used the capabilities to authorise their requests for a specific function of the application. The capabilities consisted of a list of functions that were used to define the module, module function, type of access and capability count that specified the count permits using the capability. The capability functions were as follows (see Figure 5.1):

- **Module function:** a Groovy class function consisting of the details of the object routine such as String name, Integer index and Module name.
- **Access type:** a Groovy Enum function that contains the access permitted values such as Read and Write.
- **Person:** a Groovy class function that includes user details used by the security service such as String username, String password and String key.
- **Mod-cap:** a Java Integer value that represents the permission count and the control capability revocation
- **Object-ID:** a Java String value that represents a unique module ID in the database

```
import capabilities.security.Person

class Capabilities {

   ModuleFunction moduleFunction//uniqueModuleIdentifier
   AccessType accessType
   Person person
   Integer modCapsNumber
   String objectRelatedId

   static constraints = {
   }

   static mapping = {
      objectRelatedId type: EncryptedString
      modCapsNumber type: EncryptedInteger
      person type: EncryptedPerson
      moduleFunction type: EncryptedModuleFunction
      accessType type: EncryptedAccessType
   }

}
```

Figure 5.1: Capability code structure

The capability functions were stored in a separate table of the security service database. They were mapped to an interface class called ‘user-type’ that provided a mapping between the Java database connectivity interface (JDBC), which connected and executed a query with the database, and the Grails Hibernate library, which managed all the queries to the database. Each of these functions represented a client authorisation on a specific account of a cloud application. Such important client details were encrypted in the security service database and were only accessible by the security service.

Controllers were responsible for handling web requests in Grails. A number of controllers were created and implemented in this proof-of-concept. The capability controller was the primary controller that controlled all the activities and requests for the capabilities. It managed requests to the capability class allowing operations such as creating, disabling, enabling or transferring capabilities.

5.4.1.3 Encryption/decryption implementation

Encryption/decryption was used on this platform to keep the authentication/authorisation details secure from unauthorised access. The encryption/decryption methods could be called only by the security service when it needed to check the user authorisations or to generate/update them. All the sensitive details were encrypted and stored in a separate database for transmission and at rest, and only the security service had access to the unencrypted data. The encryption/decryption mechanism used in this platform was the AES algorithm [46] and SHA-2 to hash data [49].

AES is one of the most reliable encryption mechanisms and is capable of protecting sensitive government information well into the next century according to the NIST [46]. It is a symmetric encryption mechanism that requires a key to code the data and decodes it. Each user has a unique key generated in the users' list table (person) and encoded to be used anytime by the security service.

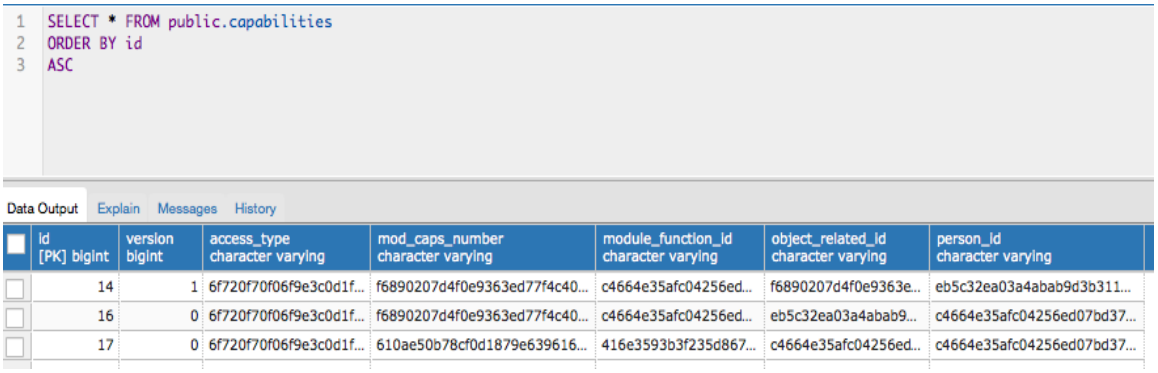
The security service encrypts each capability key separately into a class, as each one of the capability keys has a different value (Integer, String etc.). The capability values are mapped into the user-defined interface 'userType', and each of these values is retrieved and prepared by the 'nullSafeSet/nullSafeGet' method to be either encrypted or decrypted. The 'nullSafeSet' method passes the prepared values to the encryption method to get it encrypted. The 'nullSafeGet' method passes the prepared values to the decryption method to be decrypted.

The encryption starts by applying what follows on each key value:

1) The encryption begins by calling the AES algorithm. The SHA-2 provider is called to generate a hash for the raw data. This hash function requires a key, and here, the security service administrator key was used. SHA-2 generates a digest message that returns the array of bytes that result from the hash calculation. The array has a size of 16 bytes to reduce database utilisation and to improve performance.

2) The array of bytes is initialised in a form that is understandable to the AES algorithm. The AES encryption method encrypts the data passed and returns the resulting value to be stored in the database (see Figure 5.2 that shows the capability encryption results).

In the decryption process, the same steps are followed, but the AES method is changed to the decryption mode, which returns the original raw value.



```
1  SELECT * FROM public.capabilities
2  ORDER BY id
3  ASC
```

Data Output | Explain | Messages | History

	id [PK] bigint	version bigint	access_type character varying	mod_caps_number character varying	module_function_id character varying	object_related_id character varying	person_id character varying
	14	1	6f720f70f06f9e3c0d1f...	f6890207d4f0e9363ed77f4c40...	c4664e35afc04256ed...	f6890207d4f0e9363e...	eb5c32ea03a4abab9d3b311...
	16	0	6f720f70f06f9e3c0d1f...	f6890207d4f0e9363ed77f4c40...	c4664e35afc04256ed...	eb5c32ea03a4abab9...	c4664e35afc04256ed07bd37...
	17	0	6f720f70f06f9e3c0d1f...	610ae50b78cf0d1879e639616...	416e3593b3f235d867...	c4664e35afc04256ed...	c4664e35afc04256ed07bd37...

Figure 5.2: Capabilities encryption representation in the database

A complete prototype of the security service has been implemented; it contains controllers over the main classes and applications (Bank and Social Media). See Figure 5.3.

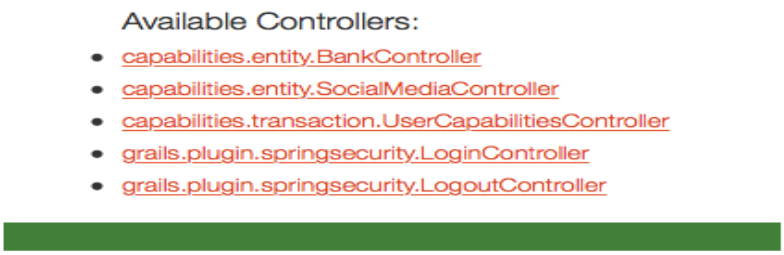


Figure 5.3: Server application

5.4.2 Client application implementation

The client application contains a login interface that the clients use to access the cloud applications. It includes two demonstration applications implemented as part of this work: Bank and Social Media. Several client accounts were generated and provided with authentication/authorisation details to use these modules. These clients could use their login details to enter the system; see Figure 5.4.

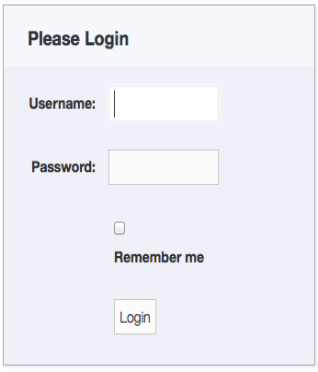


Figure 5.4: Client platform authentication

When the clients log in to their accounts, they are permitted to complete operations on the cloud application on the basis of their capabilities for a particular count, as explained in Section 4.7.3.2. Two sample client applications were implemented and connected to the security service:

1. Bank system

The bank system prototype was developed to use our platform and provides general bank operations such as withdrawal, deposit and balance inquiry. The authorisations to perform any of these operations depend on the capability. Clients need to have the right capability to call any of the bank operations. The capability authorises the client to complete a transaction on an account for a particular count. The check on the authentication/authorisation held by the security service and all the bank client details are kept encrypted and are saved in the capabilities within the security service.

Figure 5.5 shows how to create a new bank account by using the platform containing the following:

- can-admin-assign-capability (allows the Admin to assign capabilities to the account)
- balance
- account name
- person (owner of the account)

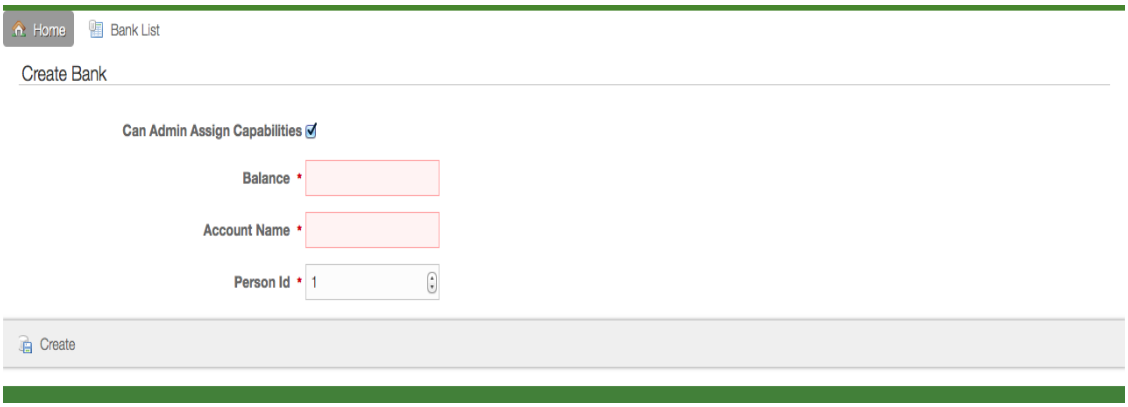


Figure 5.5: Creating bank accounts in the system

Data owners have a useful option when they create a new account that allows the admin to assign capabilities to the other users. This option gives the administrator the authority to generate capabilities for this account on behalf of the data owner. This authority can be stopped or revoked anytime by the same owner. This can help to reduce or prevent the admin from using this option maliciously. A call can be made to disable the ability of the administrator to assign a capability on behalf of the owner, as explained in Section 4.7.3.5.

2. Social media system

The social media system was implemented using the security service to control the private data of the social media clients. The system consists of primary social operations such as Like, Comment, Share, Show photos and Show videos. The security service gives any social media account holder the ability to assign a capability to the other users to perform any of these operations on the account holder for a particular count. If any user wishes to complete one of the operations, they need to first obtain an authorisation capability from the account holder. Any client requesting to see pictures or videos belonging to the account holder must have the right capabilities to permit access, or the request will be rejected by

the security service. This design can help to prevent unauthorised access to private pictures and videos and control the access to them.

To create a new account on the social media system, the following data are needed; see Figure 5.6:

- can-admin-assign-capability (gives the admin the ability to assign capabilities to the other users on this account)
- account nickname (account name)
- person ID (account owner)
- content (friends)

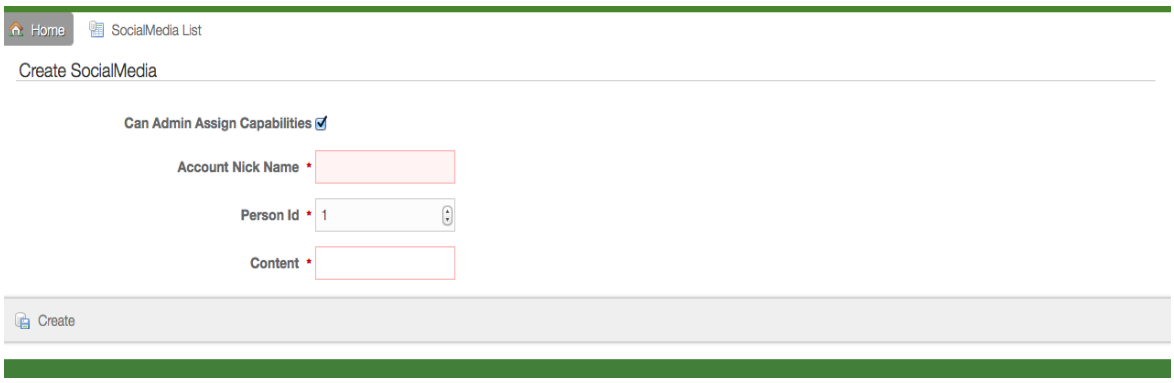


Figure 5.6: Creating new social account process

The client application contains a number of remote services that communicate with the security service to post the client operations. They handle the client's requests and pass them encrypted through private channels to the security service.

Common service: This is a custom service that contains the primary functions of the security service, as shown in Figure 5.7.

Remote service: This receives requests from the common service, encrypts the values and passes them by using the HTTP remote function to the security service in the server (see Section 5.6.3).

Custom detail service: This provides the remote service with custom details on users when they make requests.

The implementation of the client interface is based on the security service implementation, which it contains actions for managing the cloud applications and their clients. The central security service actions provided by this interface are outlined below.

```
package capabilities

interface ICapabilities {
    Boolean disableCapabilities(Long id)
    Boolean transferCapabilities(Long id,Long personId)
    Boolean enableCapabilities(Long id,Integer number)
    Boolean disableAllCapabilities()
    Boolean hasCapabilityAccess(Object objectRelatedId,Long personId,String moduleName,String function,AccessType accessType)
    PagedList searchCapabilitiesBean(Map params)
    CapabilitiesBean getCapabilitiesBean(Long id)
    CapabilitiesBean saveCapabilitiesBean(Map params)


    ModuleBean getModuleBean(Long id)
    ModuleFunctionBean getModuleFunctionBean(Long id)
    ModuleFunctionBean getModuleFunctionByName(String name)
    ModuleBean getModuleByName(String name)
    List<ModuleBean> searchModuleBean(Map params)
    List<ModuleFunctionBean> searchModuleFunctionBean(Map params)


    PersonBean getPersonBean(Long id)
    PersonBean getPersonBeanByUserName(String username)
    List<PersonBean> searchPersonBean(Map params)


    PersonSecurityBean getUserInfo(String username)


}
```

Figure 5.7: Capability interface methods

5.2.3 Implementation of client and server communications

The implementation of client and server communications is designed to secure data during the transmission process and pass them through private channels to the security service. The cloud application forwards the client requests through the remote service to the security service in a serialised form to be de-serialised only by the security service. The serialised data contain specific details that are only understandable to the security service. These data are represented as a Java Bean class (a class that encapsulates many objects into a single object) that contains the details of the user, object name and interface details. The security service receives the request, checks it against the clients' details and replies with the authorisation permission or rejection (True or False). Figure 5.8 shows a snapshot of the code implementing the communication between the client interface and the server.

The communication mechanism used in the prototype relies on the transmission of the serialised objects over the network. A package of data is sent out to the security service; it contains the user details and the invoked interface method details as a serialised bean. The security service receives this object, de-serialises it to the original details, checks the user details and checks the request against the user authorisation capability. If the security service grants the call, it will return an approval message to the application confirming the user request; otherwise, the call will be rejected.

In case a request from the security service requires a list of data to be sent back to the cloud application, the Grails parameter map is designed to receive millions of records that can be represented in a page list. The security service carries out the request by checking and returning the results as a page list of serialised beans, where each record of the list is an individual serialised bean.

```
@ansactional
class RemoteService implements ApplicationContextAware {

    static transactional = true
    def applicationContext

    final String SERVER_IP_Address = "localhost:8090/Capabilities";


    void setApplicationContext(ApplicationContext applicationContext) {
        this.applicationContext = applicationContext
    }

    def capabilitiesProxy

    public void capabilitiesProxySetup() {
        if (!capabilitiesProxy) {
            try {
                HttpInvokerProxyFactoryBean httpInvokerPerson = new HttpInvokerProxyFactoryBean()
                httpInvokerPerson.setServiceInterface(ICapabilities)
                httpInvokerPerson.setServiceUrl("http://" + SERVER_IP_Address + "/httpinvoker/CommonService")
                httpInvokerPerson.afterPropertiesSet()
                capabilitiesProxy = (ICapabilities) httpInvokerPerson.getObject()
            } catch (ConnectException ex) {

                log.error("Failed to connect to the Server", ex)
            }
        }
    }
}
```

Figure 5.8: HTTP call method in the remote service

5.5 Capability service plug-in

To hook up a cloud application to the security service, it has to be published online for the cloud providers. The Grails framework provides the option to build a plugin for web services to be posted online. The Grails plugin as described in the Grails official site ‘is like any application but has a plugin descriptor and can be packed as a plugin and be installed into other applications; it is considered a way to modularise large Grails applications’ [131].

The capability plugin contains everything related to the platform, such as primary classes, services, controllers and views. These define the metadata of the plugin and hook to the plugin extension points. The cloud providers can call an external service that allows them to use the security service. This requires several configurations made by the security service; some entity properties need to be added and changed to supply the plugin concept.

There is a need for some configuration for each specific application that wishes to join our security service, however, there is an easy way to join. The application owners have to transfer specific details for their clients to be stored in the security service database. In contrast, the security service provides Authentication/Authorisation control for those clients is based on provided criteria from the application owners. Therefore, based on those criteria provided by the owners, the clients will be distinguished according to who is able to create/disable/transfer each capability (and more) as discussed in the security service features in sections 4.8.3.4.

5.6 Implementation of security scheme

The security service is implemented to represent the SPEEDOS protection environment over cloud computing, in which the cloud applications are restructured without reimplementing each cloud application so that no software engineering difficulties introduced. It provides a model to apply the information-hiding principle to the cloud applications and a linking mechanism that controls access to the applications' function data by presenting the right capability. This design supports another protection mechanism for the data, in which the application security depends on not only the protections applied to the applications but also the protection mechanisms applied to the methods.

The authentication/authorisation details for users represented in the capabilities are encrypted in a separate database accessed only by the security service. The encryption/decryption processes rely on the encrypted keys that are held in a separate data table referenced to be used only by the security service. Each key has a long string of up to

72 characters generated by a standard ‘one-way encryption method’ that is approved to be protected against breaches.

The cloud application communicates with the security service through private channels. The data get encrypted before they are transferred and decrypted by only the security service that holds the decryption format for these data, to prevent interceptions. The users do not interact with the security service at all; they only communicate with their cloud applications through a client interface. Thus, the communications are designed to reduce/prevent the opportunities of compromising the security service as it is not published to all the cloud users but only to the cloud providers through private channels by using encrypted data.

The security service provides an additional level of checks on the function data, in which the users can control the access to their data. The users are provided with an ability to revoke access on their data anytime, and the access to data is granted only by providing a valid capability that indicates the caller’s actions and for a limited number of uses. The cloud users will be more comfortable having control over their data as the cloud applications are available online and many breaches of data have been reported in the past, as discussed in Section 2.9.

The security service was designed/implemented to have maximum security for the cloud applications and the user data. However, the evolution of the protection mechanisms provided by the security service will be presented in the next chapter to show that the security service’s inherent design/implementation is secure against the current cloud threats/attacks discussed in Section 2.7.

Chapter 6

Security overview and experimental results

6.0 Introduction

In the previous chapter, the new security cloud platform was implemented by leveraging the SPEEDOS security features. The new platform can be used to build and design cloud applications in a way that provides better security, utilisation of resources and flexibility in deployment.

This chapter presents an overview of the security aspects of the introduced platform, demonstrates that the platform accurately reduces/prevents the computing risks of the cloud model, and shows that the linking mechanism system and module structure proposed to provide security benefits over the current transaction standards in the cloud. These scenarios were used to perform experimental and functional comparisons to show that the platform offers advancements over the existing cloud platforms.

6.1 Security overview and experiment

The platform was designed from scratch with the primary goal of preventing/reducing unauthorised access to the cloud computing application. The novel SPEEDOS security technique was implemented using the Grails framework to provide cloud services.

To evaluate the overhead of the implemented security techniques with respect to running the new cloud platform on ‘bare metal’, two types of experiments that exercised the entire system were used to characterise performance and security.

The first experimental environment measured the security strength of the platform against authorisations by testing calls between applications and the security service and measured the authenticity of callers. The second experiment tested how quickly the system responded to requests and the overall performance as compared to the results of a standard ACL implementation.

6.2 Overview of attacks

The following provides an overview of some attacks/threats, mentioned in Chapter 2, to which the existing cloud platforms are vulnerable, and how the new platform helps to prevent or mitigate these vulnerabilities.

6.2.1 Malware or malicious software

Referring to the attack described in Section 2.7.1.1, malware consists of an attack that focuses on a variety of forms of intrusive software, such as computer viruses, worms, Trojan horses, ransomware, spyware, adware and scareware.

This attack aims to reach sensitive data and copy them to the threat actor. All the privileges and confidential data related to the new platform users are encrypted and stored within the security service protected by the capabilities. The security service checks each incoming call, and the caller needs to have a capability to authorise the request. The capability identifies the caller and what it can do on the system and for how many times. There is no

way to access the data without providing a valid capability. Furthermore, having the security service installed on top of the SPEEDOS system enhances the prevention of such an attack because of the many data protection features provided, as discussed in Chapter 3.

If the client application is compromised by malicious software, it can hijack the capabilities assigned to the clients using this software. However, our platform provides the clients with an option to stop their capabilities if they identify suspicious activities on their accounts, which can reduce the damage of such an attack. This option can be applied to a mobile application or any form that suits the clients.

6.2.2 Man-in-the-middle and eavesdropping

The use of a secure connectivity mechanism to communicate between clients and modules over private channels can reduce and prevent the attacks described in Sections 2.7.1.2 and 2.7.1.4. The platform contains two applications where the security service is installed in a separate database, and the cloud application calls the security service from anywhere. There is a risk that the calls between the cloud application and the security service could be hacked. If a man-in-the-middle attack or eavesdropping occurs somehow, there are few options for the attacker to see the data or alter the requests. However, all the data transmissions take place only between the cloud applications and the security service through an encryption mechanism as the transmitted data get coded first in the client interface and changed to a serialised format that can only be read by the security service. This type of attack cannot benefit from the data without having them decrypted. The proof of the encryption strength will be presented in Section 6.2.6.

6.2.3 Test for web storage SQL injection

With reference to the attacks described in Section 2.7.1.3, the platform was built to mitigate the injection attacks. Attackers often attempt to use injection avenues to alter their privileges in the system by using an SQL injection. In a traditional web application, if the web application has a SQL injection flaw anywhere within its code, then it is possible for the attacker to access the application database that stores all the authentication/authorisation information. In the presented platform, the users' privileges are encrypted and stored as capabilities within the security service.

An SQL injection experiment that requests data from an ACL-based application that holds the user details in a local database was performed along with another experiment requesting data from the bank application running on our security service. An SQL injection query was used to retrieve data from both the databases. The results are shown below (see Figure 6.1):

ACL application query: result= db.rows("select* from acl_entryent, acl_sidsi where si.id=ent.sid and si.sid='"paramInjection+="'")

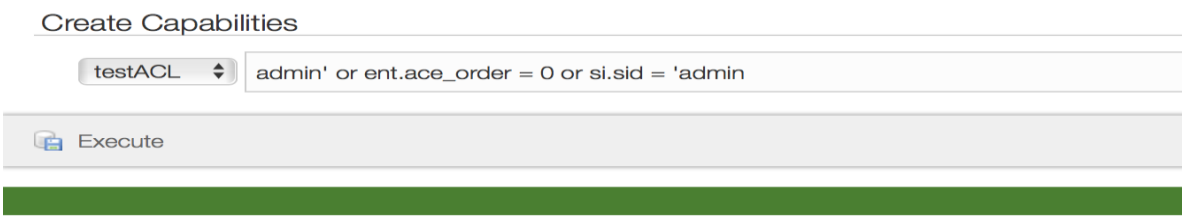


Figure 6.1: Requesting admin details of ACL application

The snapshot below shows that the previous request was approved and the user data were accessed. The SQL query was able to retrieve data from the users' database, as shown in Figure 6.2 below.

ACL-SQL injection output

[['id':5, 'ace_order':0, lacl_object_identity:5, 'audit_failurel:false, laudit_success'ffalse, 'granting':true, 'mask':2, 'sid':'adminCap', 'principantrue], ['id':6, lace_order':0, 'acl_object_identity':5, laudit_failuret:false, laudit_success':false, 'granting':true, 'mask':2, 'sid':'admin', 'principantrue], ['id':7, lace_order':0, 'acl_object_identity':5, 'audit_failure'ffalse, 'audit_success':false, 'granting':true, 'mask':2, 'sid':'user', 'principantrue], ['id':5, lace_order':0, 'acl_object_identity':6, laudit_failurel:false, 'audit_successl:false, 'granting':true, 'mask':2, isidl:tadminCap', 'principantrue], ['id':6, lace_order':0, lacl_object_identity:6, 'audit_failure'ffalse, 'audit_success':false, 'granting':true, 'mask':2, 'sid':'admin', 'principantrue], ['id':7, 'ace_order':0, lacl_object_identity:6, 'audit_failurel:false, laudit_success'ffalse, 'granting':true, 'mask':2, 'sid':'user', 'principantrue]]

Figure 6.2: Administrator details from ACL application

The same attack was applied to the bank application that uses the security service to check whether the SQL injections can retrieve any user data or not. The SQL query and the result of the experiment are shown below (see Figure 6.3).

Security service application query: result= db.rows("select* from capabilities where person_id=""+paramInjection+""")

Security service -SQL injection inquiry

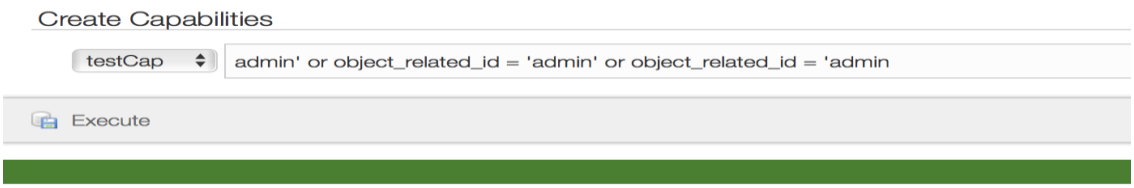


Figure 6.3: Requesting administrator details from the security service

Figure 6.4 shows the result of a previous query request to the security service. The SQL query could not retrieve data from the security service database that holds the user data.

Security service-SQL injection output

Figure 6.4: Administrator details from the security service

A tool was used to automate the experiment mentioned above to provide more evidence for the strength of our platform against SQL injection attacks. SQLMAP is an open source testing tool that robotises the way toward recognising and abusing SQL injection defects and taking over control of the database servers. Figure 6.5 shows the SQLMAP queries used on the ACL application database shown below:

ACL-SQLMAP injection inquiry

```
python sqlmap.py --
uri="http://localhost:8090/Capabilities/bank/executeSqlInjection?type=testACL2&paramInjection=admin&execute=Execute"
```

Figure 6.5: Requesting admin details of ACL-bank system

Figure 6.6 presents the result of the SQLMAP execution on the ACL application. It shows that the tool could retrieve the user data from the ACL application database.

ACL-SQLMAP injection output


```
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers as
sume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 21:50:43


[21:51:20] [WARNING] GET parameter 'execute' does not appear to be dynamic
[21:51:20] [WARNING] heuristic (basic) test shows that GET parameter 'execute' might not be injectable
[21:51:20] [INFO] testing for SQL injection on GET parameter 'execute'
[21:51:20] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[21:51:20] [INFO] testing 'MySQL >= 5.0 boolean-based blind - Parameter replace'
[21:51:20] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[21:51:21] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[21:51:21] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[21:51:21] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[21:51:21] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[21:51:21] [INFO] testing 'MySQL inline queries'
[21:51:21] [INFO] testing 'PostgreSQL inline queries'
[21:51:21] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[21:51:21] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[21:51:21] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[21:51:21] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[21:51:21] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[21:51:21] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[21:51:22] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[21:51:22] [INFO] testing 'Oracle AND time-based blind'
[21:51:22] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[21:51:23] [WARNING] GET parameter 'execute' does not seem to be injectable
sqlmap identified the following injection point(s) with a total of 371 HTTP(s) requests:
---
Parameter: paramInjection (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: type=testACL2&paramInjection=admin' AND 7839=7839 AND 'dZPV'='dZPV&execute=Execute
---
```

Figure 6.6: Administrator details from the ACL-bank system

The same tool was applied to the bank application that uses our security service and the same SQL query, as shown in Figure 6.7 below.

Security service-SQLMAP injection query

```
python sqlmap.py --
uri="http://localhost:8090/Capabilities/bank/executeSqlInjection?type=testCap2&para
mInjection=admin&execute=Execute"
```

Figure 6.7: Requesting admin details of security service-bank system

The security service refused to provide any user data to the SQLMAP tool as these details were protected by the capabilities and could be accessed only by the security service.

Figure 6.8 shows the result of the experiment; the rejection message is highlighted.

Capability-SQLMAP injection output

Figure 6.8: Administrator details from the security service-bank system

6.2.4 Shared memory threats/unauthorised access to the security service

The security service is protected against the shared memory threats described in Section 2.9; that is, if the security service database is stored in a shared memory in the cloud, all of its data are encrypted. The security service provides a strong encryption mechanism to the database that holds all the users' authentication/authorisation capabilities. Furthermore, the security service relies on encryption keys generated from the users' details by using a standard algorithm that joins these details to specific keywords and then encrypts them by using a standard one-way encryption method in a string of up to 72 characters to be stored in a separate data table.

The security service users' details are encrypted using a standard one-way encryption method that remains resistant to brute-force search attacks even with increasing computational power. The attackers might use a password-cracking program [41] to access the users' passwords and therefore, use their privileges to achieve further data. However, the security service encrypts all the user passwords by using very long hashed passwords approved against such attacker programs. The passwords are hashed using an algorithm that uses specific keywords added to the passwords and stores these details in a string of up to 72 bytes by using 10 iterations to slow down the unauthorised encryption of these details. However, decrypting keys without permission by guessing would require the whole time universe. Furthermore, all the encryption keys are encrypted using a specific keyword with the encryption method to be stored in a separate data table that is linked only to the security service in which no root flow is provided to any other party. When the number of unauthorised accesses to the security service exceeds a certain number, the login gets blocked and a notification is sent to the administrator to take the necessary action. In the

following experiment, a sniper tool was installed on the server that holds the security service trying to capture data while the security service is running.

Dumping the security service memory

Jvisualvm is a tool that captures the data of Java applications during a running process. It uses the data retrieved by the Java developer kit (JDK) and organises them to be represented through a user interface [133], where the data are presented in a way that enables us to view the data on multiple functions.

The Jvisualvm tool was used to capture the object actions on the security service memory and to prove that all the data were protected. Figure 6.9 shows the result of a capability SQL query from the security service memory where all the retrieved capability data are encrypted.

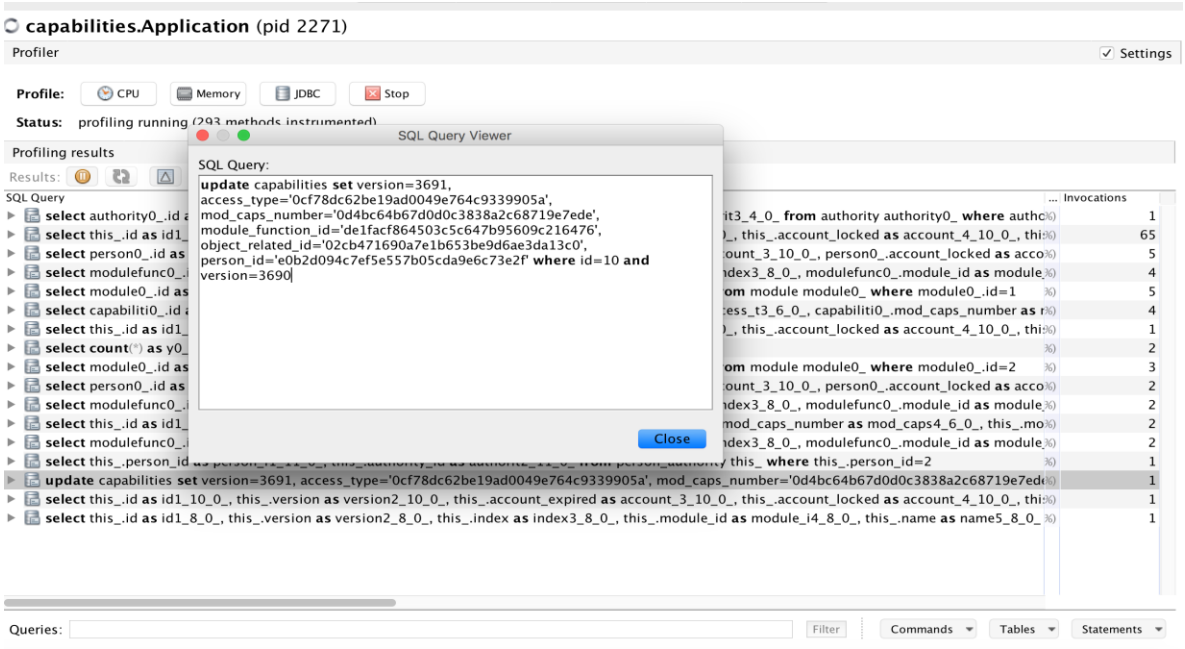


Figure 6.9: Capability SQL results

Further, Jvisualvm was used to capture the users' passwords as Figure 6.10 shows that all the passwords and encryption keys were encrypted and protected. The keys used to encrypt the passwords and the capabilities were generated using a robust structure as explained in Section 6.2.6, to protect them against any hacking tools.

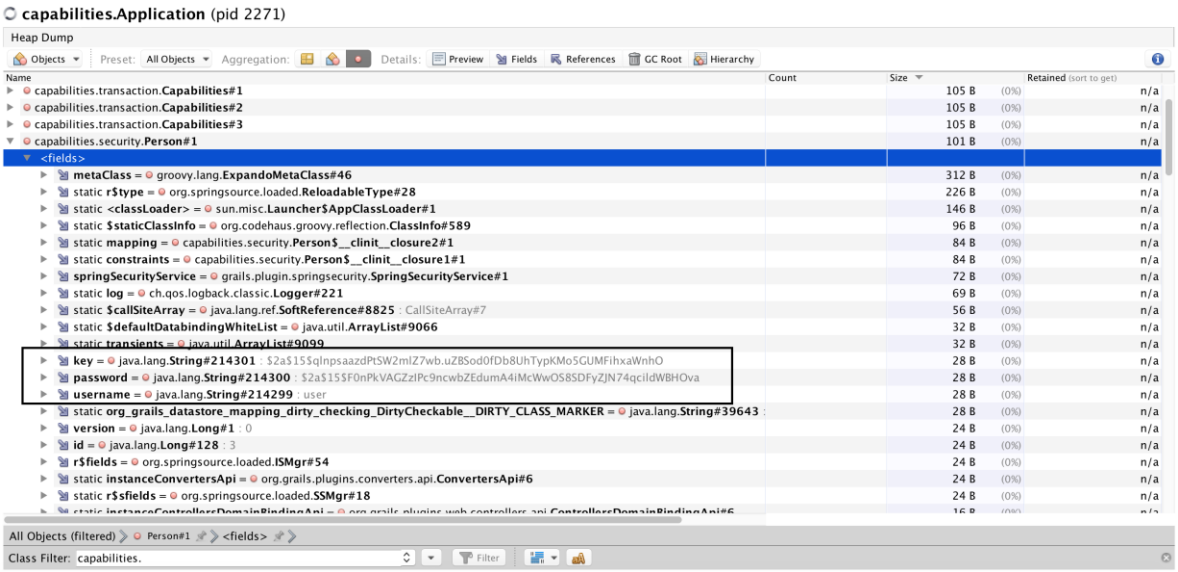


Figure 6.10: Passwords and encryption keys of the security service

6.2.5 Client application threats

Basically, the introduced solution provides a user interface that includes cloud applications where the users access the interface to interact with their applications account. The security service is designed to receive calls from the applications to check on the authorisations/authentications for their users, and on the basis of these applications, grant the users' call. Private channels are provided to bypass the data between the legitimate applications and the security service; however, let us assume that the data are from a fake legitimate application or the legitimate application gets compromised and starts calling the security service; this affects the security service.

The security service is implemented to deal with cloud applications, and as there are a number of applications out there in the cloud, there is a chance that a fake application is generated to gain access to the security service and tries to take control over the security service. The security service is implemented in response to the authentication/authorisation requests of only the applications. As described in Section 5.6.3, the call carries out the module ID, function ID, user ID and user action where the security service validates these details of the caller with those stored in the database and then provides answers; if the details do not match, the call gets rejected. If a user/application wishes to access any other application, he/she/it must be authenticated first; otherwise, the call will be rejected by the security service. Furthermore, the user interface is very limited and if the user/application tries to reach more data that are not included in the user interface, then the call will be rejected by the security service. The following experiments show how the security service acts on certain malicious activities such as

- malicious user: The activities of a malicious user include the following:
    (1) Intruder trying to gain access to the capabilities by using the server details.
    (2) Logged-in user trying to gain access to the capabilities by using the server details.
    (3) Logged-in user trying to generate the capabilities on other applications.
- compromised user/application.

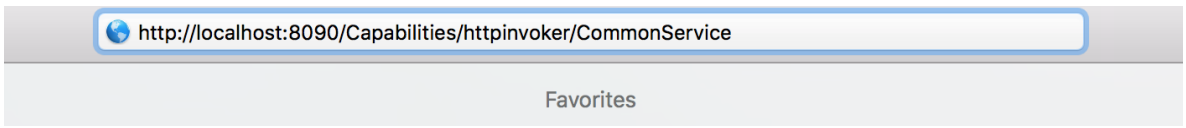
6.2.5.1 Malicious user

One of the cloud computing threats is that the cloud can be used to retrieve public and private information about its users, as discussed in Section 2.9. Recently, the Facebook

platform was used by a third party to uncover the users' identities and collect information on most of its 2 billion users worldwide [134]. However, the security service structure restricts access to each function's data in the applications in which access to any function is granted only to a legitimate user trusted by the data owner.

The security service designed to force unwanted access to application functions by granting calls to someone with a valid capability. In the social media application designed by the security service, if a user wishes to retrieve the social media user's data (pictures, videos and locations), he/she must have a valid capability of the data owner to have his/her request granted. The security service restricting access to data per user and per function provides more protection to user data in social media applications than extracting data from others.

A few experiments were conducted to prove that the user cannot retrieve data from others without having the right capability in the security service and that each user must be authenticated. The following experiments show the system actions to a skilled user who knows the techniques used in the security service from which he/she stole some details that could be used to invoke the security service. The first technique used was to call the security service by using the stolen details such as (service URL) assuming that the skilled user knows these details; however, the system replays with no further data and requests the authentication of the user, as shown in Figure 6.11.



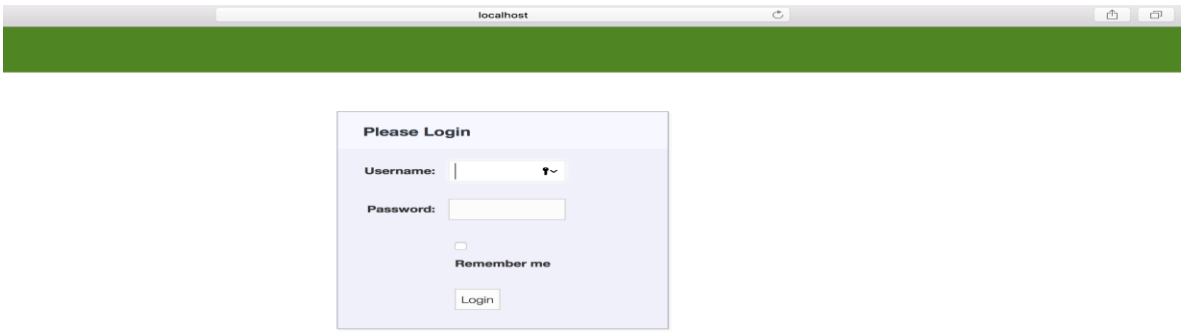
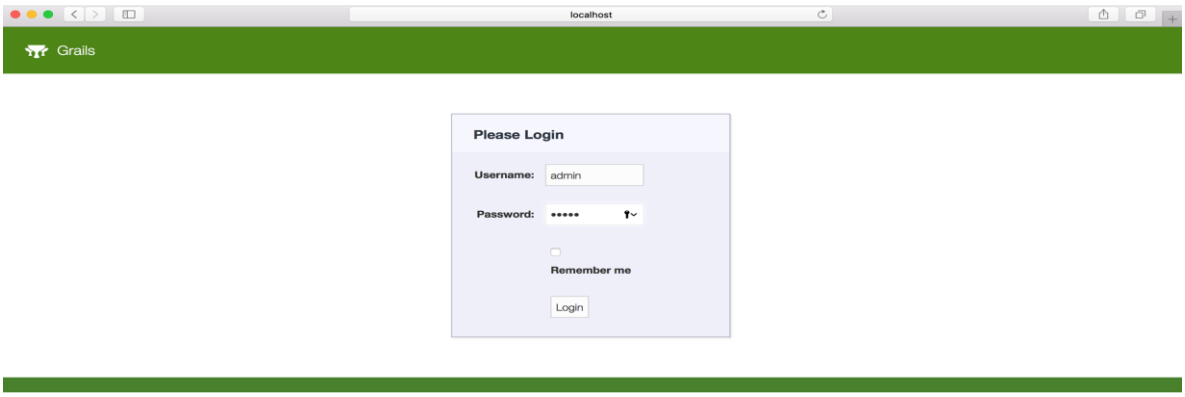


Figure 6.11: User trying to retrieve data using capability URL through web browser

The second option was that a logged-in user used the service URL to retrieve any further data of the security service through a normal web browser. Figure 6.12 shows the experimental results, where the security service requested the authentication details of the user before accepting any calls; however, even if this user is authenticated, the security service returns an error for the request as it is not the formal way of users' requesting data from the security service.



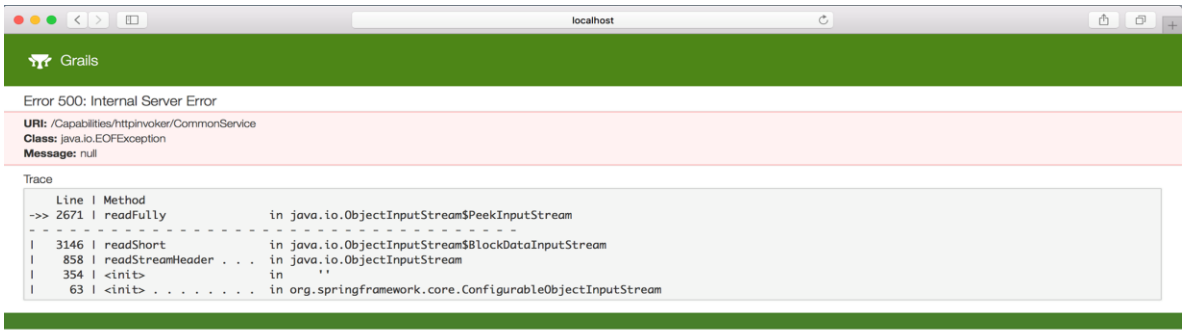


Figure 6.12: Logged-in user trying to retrieve further data using URL

The last option was that a logged-in user to the bank system attempted to generate a capability on other applications, 'Social Media, he/she was not authenticated to. Figure 6.13 shows the results of the experiment and that the security service returns the 'log-in page' as it checks the authentication of the users on the other application before permitting access to any data.

```
def testOtherApplicationCapability(){
    //Logged in user to the Bank system and trying to get capability of the social media application
    Long personId = springSecurityService?.principal?.id
    Long bankId = "3"
    String module = SystemModule.SOCIAL_MEDIA.value()
    String function = "like"

    def resultData = commonService.hasCapabilityAccess(personId,bankId,module,function,AccessType.WRITE)

    render resultData
}
```

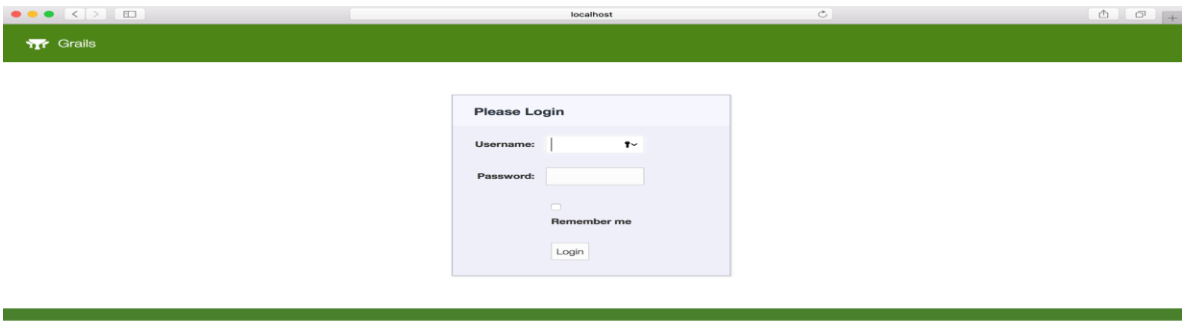


Figure 6.13: User trying to retrieve data of other applications by using valid capability details

6.2.5.2 Compromised user/application

The users' data in the security service are all encrypted and protected by hashed keys that are only accessed/decrypted by the security service as proved experimentally. Therefore, there will be no opportunity for the users or applications to gain access to the capabilities; only their requests to their accounts get approved. If any application account gets compromised by a hacker, the security service will take an action in which the security service administrator can revoke all the capabilities for the specific application account and block it from generating any capability in order to reduce the damage caused by the malicious attack. Figure 6.14 from the server shows the revocation option available for the security service administrator to perform on any account in the case of misuse.

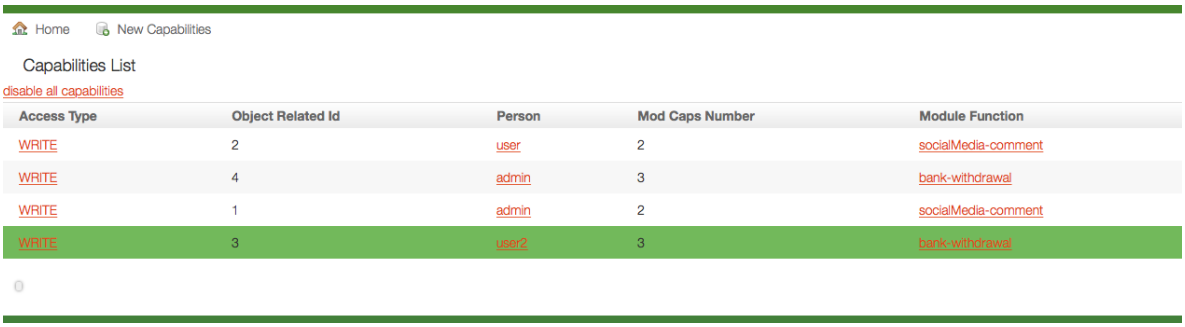


Figure 6.14: Revocation capabilities option

6.3 Test analysis report

A meter and a performance tool were configured to scale, analyse and stress the new platform. The tool was used to build the test and scale the test ability to run hundreds of services in parallel to produce thousands of concurrent users, from up to 25 locations around the world and from multiple cloud providers. The results were viewed and analysed in real-time. Before the tests were run, a baseline of a small decrease in the performance

was acceptable, but it was deemed essential to have a system that could still be used in the real world.

A bank application system was implemented using the new cloud platform. The new bank application has features one would expect to be provided to bank clients in the current bank systems and offers the unique security techniques proposed in this work. The experiment verified the performance of the new bank application by using authentic multi-user traffic to perform operations on the application and tracking the corresponding authorisation methods. It compared the timing results of the evaluation platform with that of a similar banking system implemented using an ACL mechanism.

The tool ‘JMeter’ provided a valid and useful mechanism to measure the performance of the new platform. The key performance metrics measured were the average response time that it took to complete several jobs starting from their arrival and the average number of tasks completed in a certain specific period. The experiment was performed on a bank application implemented using the ACL mechanism and the bank application implemented using the security service; each application was run through the experiment as described below.

Bank application using ACL mechanism experiment

The JMeter tool was provided with two users ‘admin, admin-acl’ to perform the test on the ACL-based application that did not use the security service. The two users were allowed to request different operations from the application and the JMeter tool created. For each real bank user, another five dummy users that requested 5,000 requests were created, as shown in Figure 6.15.

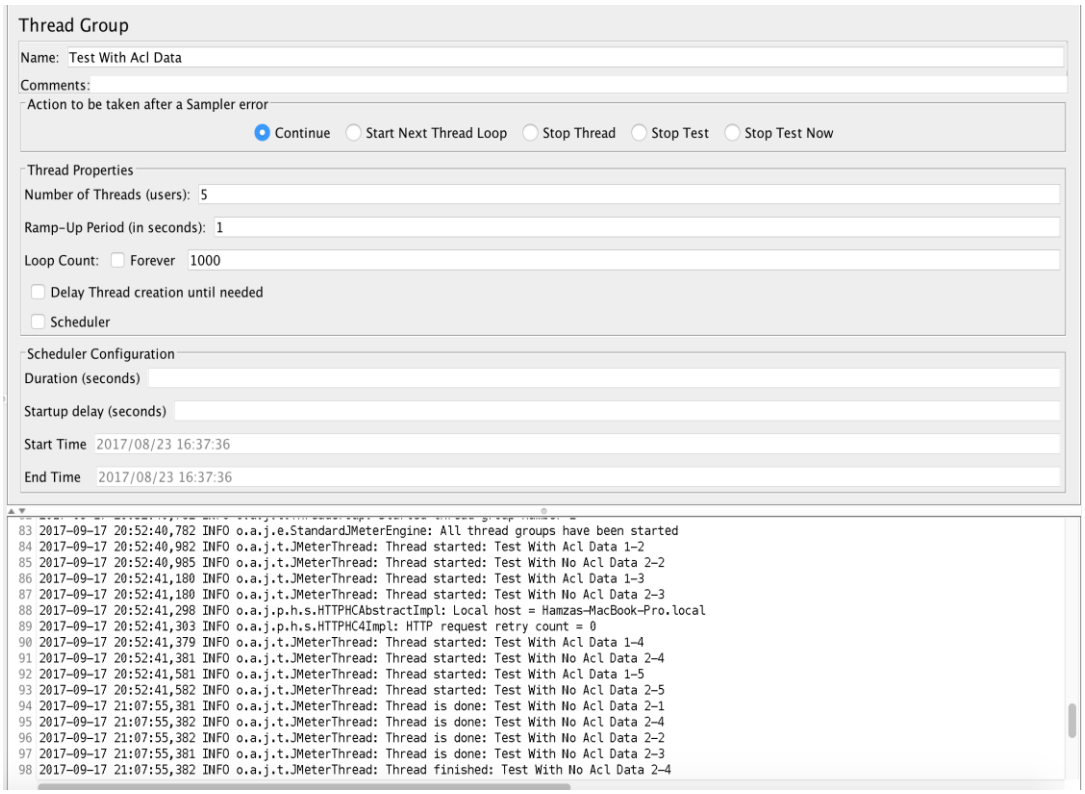


Figure 6.15: Creation of users and requests

All the users requested access to the ACL application, assuming that the ‘admin user’ and the five dummy users had ACL access and the admin_acl user with another five dummy users had no ACL access. The users with access and the users without access were created to re-create a realistic operating environment for the applications, where the users requested operations from the application with and without access. Figure 6.16 shows the server responses to the requests of the users with the ACL access.

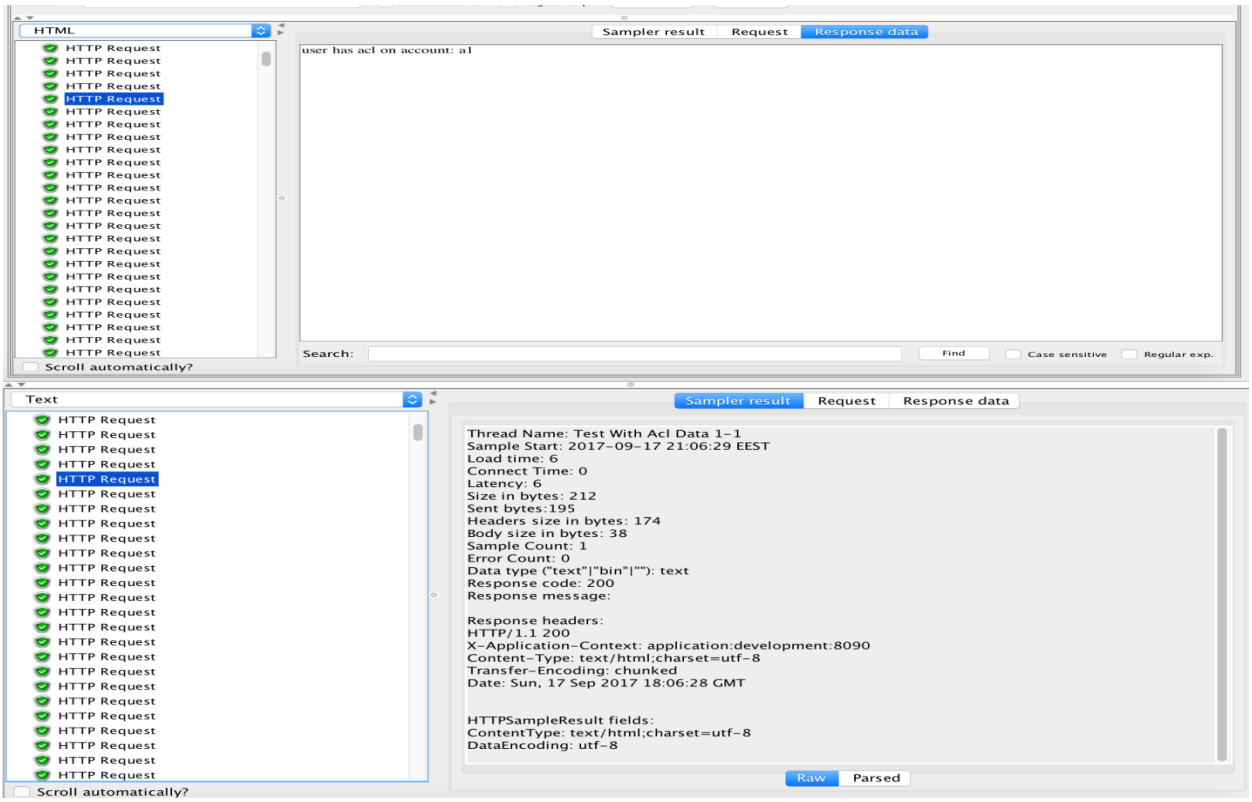
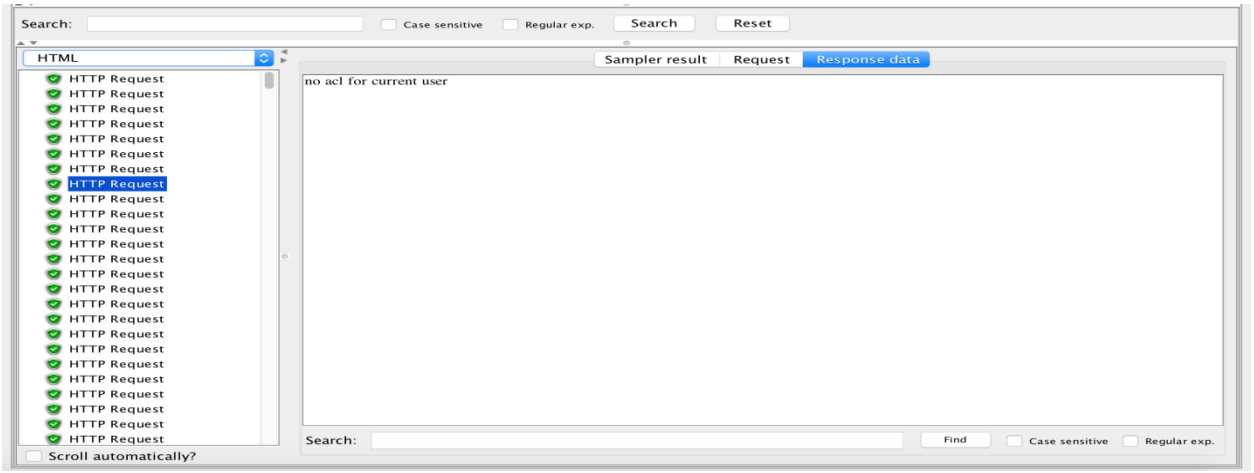


Figure 6.16: Server responses to users with ACL access

Figure 6.17 shows the server responses to the requests of users without the ACL access.



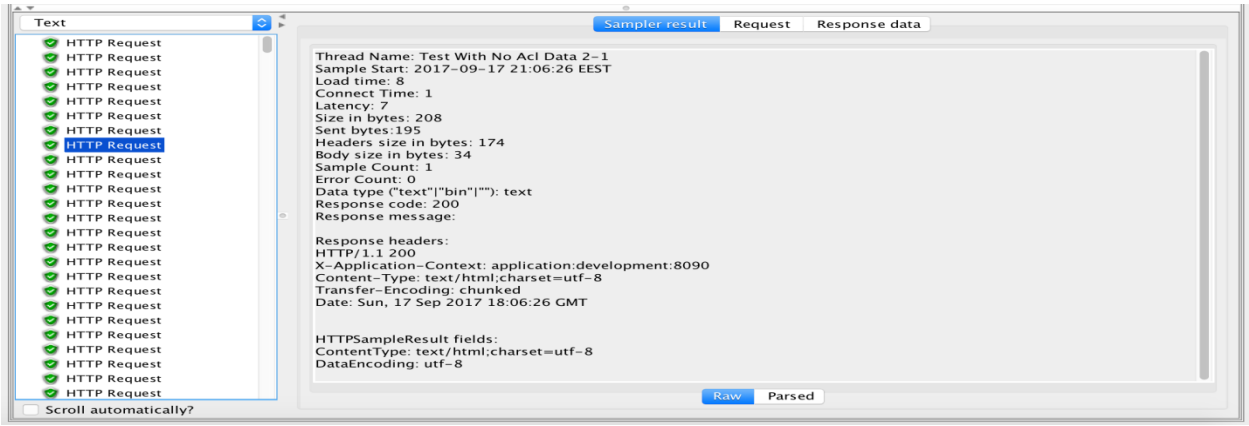
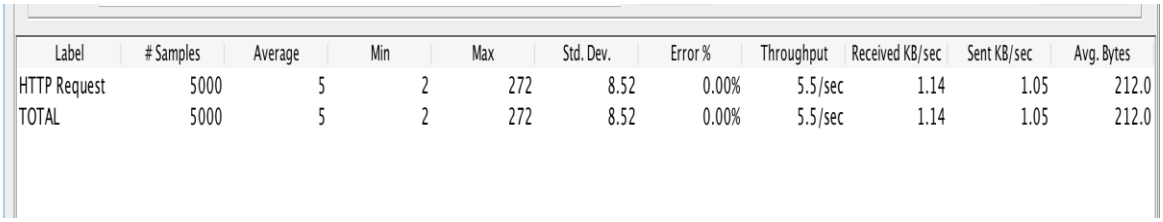


Figure 6.17: Server response to users without ACL access

Figure 6.18 shows an analysis of the experimental results. It shows that no errors occurred and all the responses were stable. Further, Figure 6.19 shows the server responses graph for users with the ACL access, and Figure 6.20 shows the server responses graph for users without the ACL access.



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	5000	5	2	272	8.52	0.00%	5.5/sec	1.14	1.05	212.0
TOTAL	5000	5	2	272	8.52	0.00%	5.5/sec	1.14	1.05	212.0

Figure 6.18: Analysis of results of ACL experiment

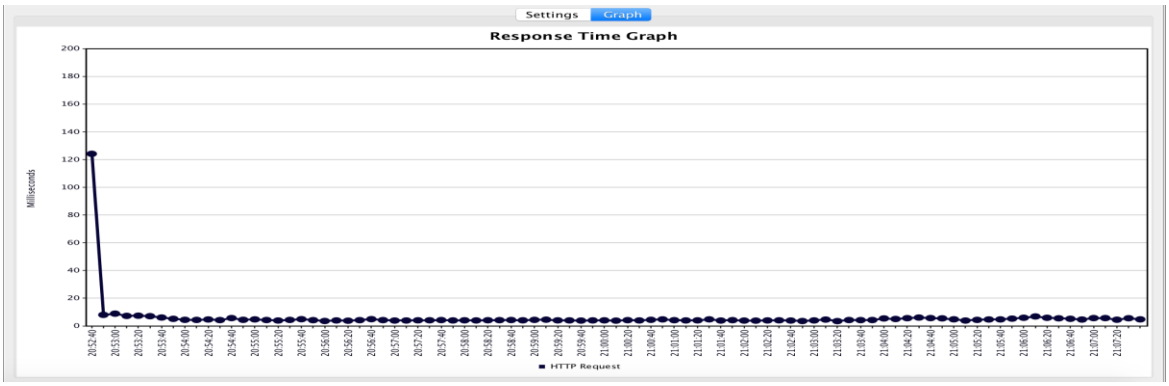


Figure 6.19: Data analysis for requests of users with ACL access

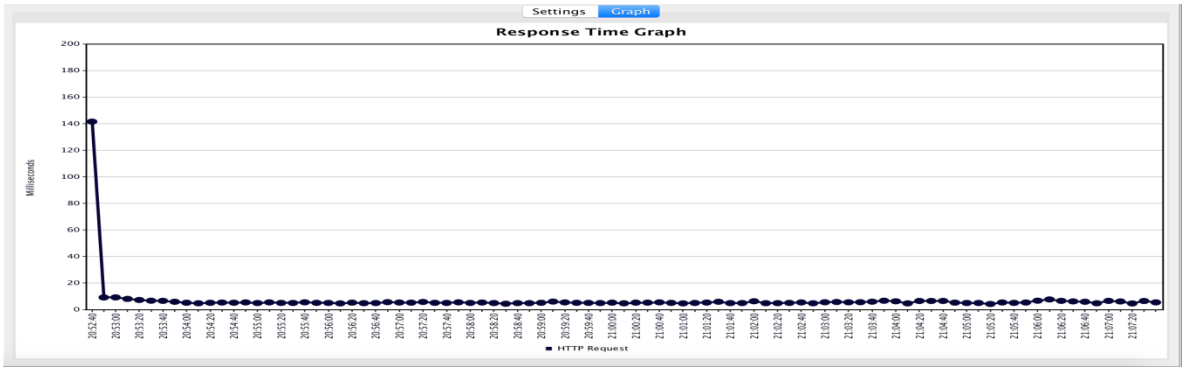


Figure 6.20: Data analysis for requests of users without ACL access

The experiment took 15 min 15 s on average, as shown in Figure 6.21.

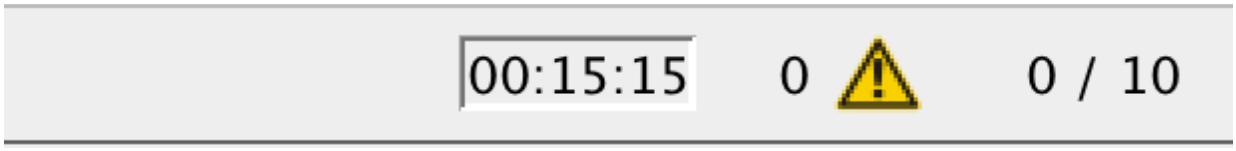


Figure 6.21: Average total time of the experiment

Bank application using security service experiment

The same tool and the same steps were used to run another test on the bank application which was implemented using the proposed system, the security service. The experiment

was run using two actual users (admin, user), and five dummy users that requested 5,000 requests for each real user were created, as shown in Figure 6.22.

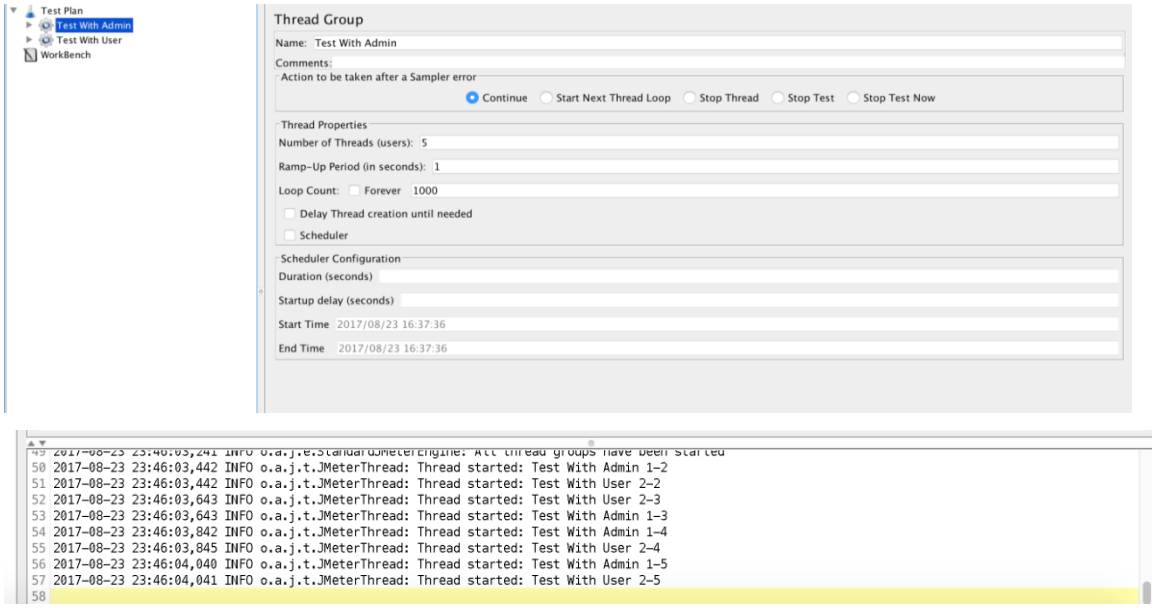


Figure 6.22: Creation of users and requests

The ‘admin’ and five dummy users were given the capability to request a withdrawal on a specific bank account, and the ‘user’ with another five dummy users to request a deposit on the same bank account, for testing the application in a multi-operation access privilege environment.

The requests of the admin and the five dummy users were performed successfully, as shown in Figure 6.23.

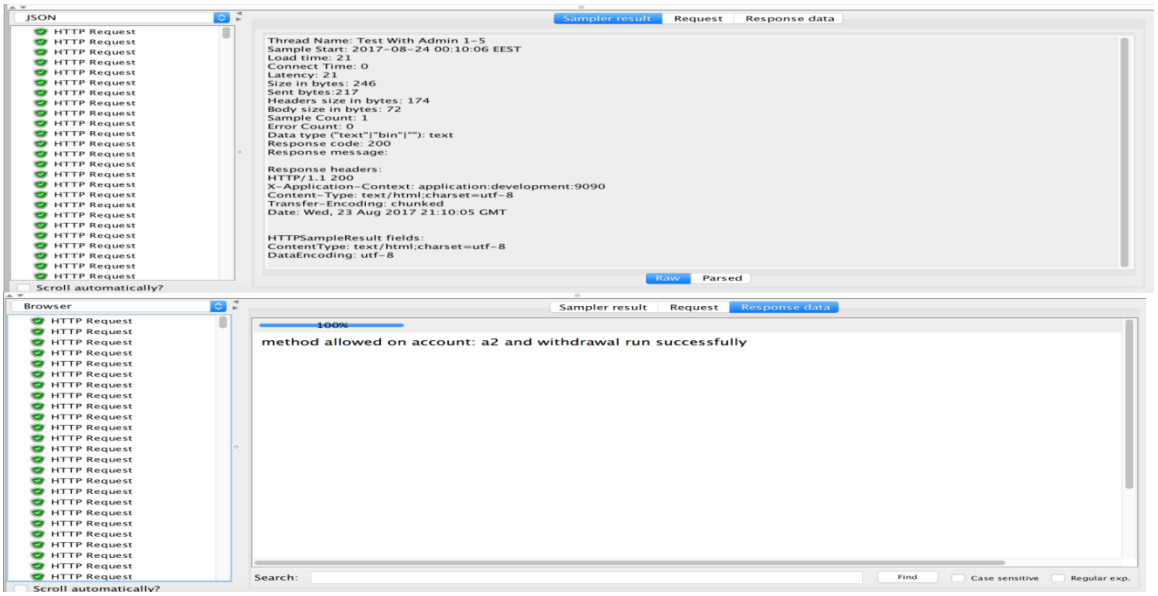


Figure 6.23: Admin requests successful

The requests of the real user and the five dummy users were also performed successfully, as shown in Figure 6.24.

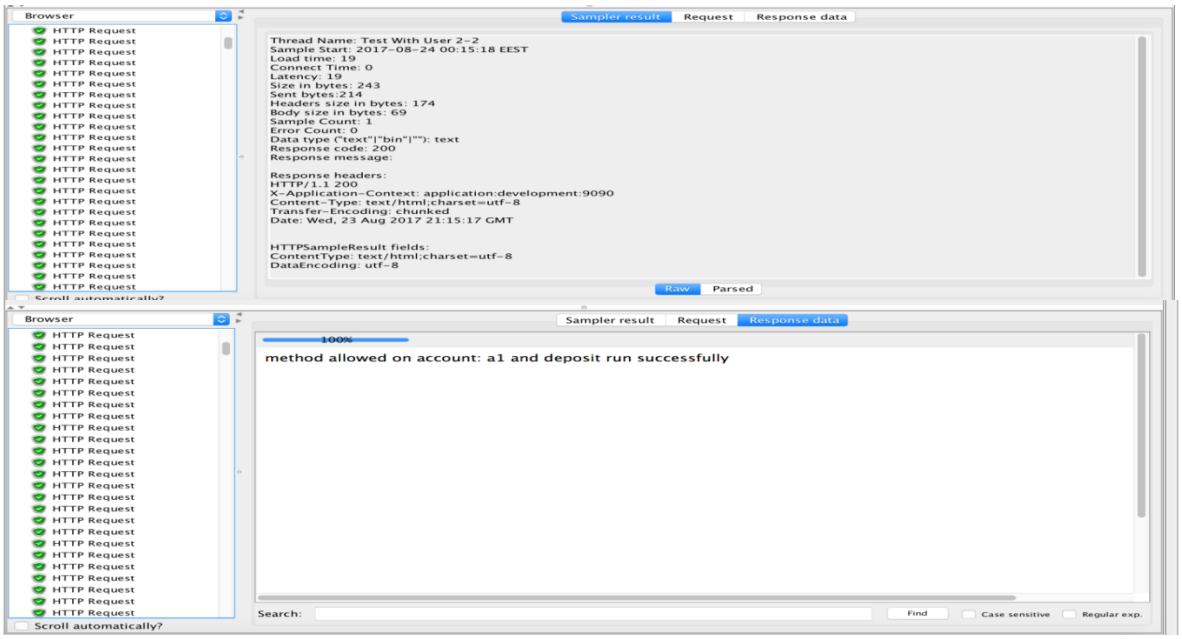


Figure 6.24: User requests successful

Figure 6.25 presents the analysis of the experiment. It shows that no errors occurred and all the responses were stable. Further, Figure 6.26 shows the server responses graph for the admin users, and Figure 6.27 shows the server responses graph for the other users.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	5000	22	14	132	5.39	0.00%	5.4/sec	1.23	1.14	232.7
TOTAL	5000	22	14	132	5.39	0.00%	5.4/sec	1.23	1.14	232.7

Figure 6.25: Analysis of experimental results

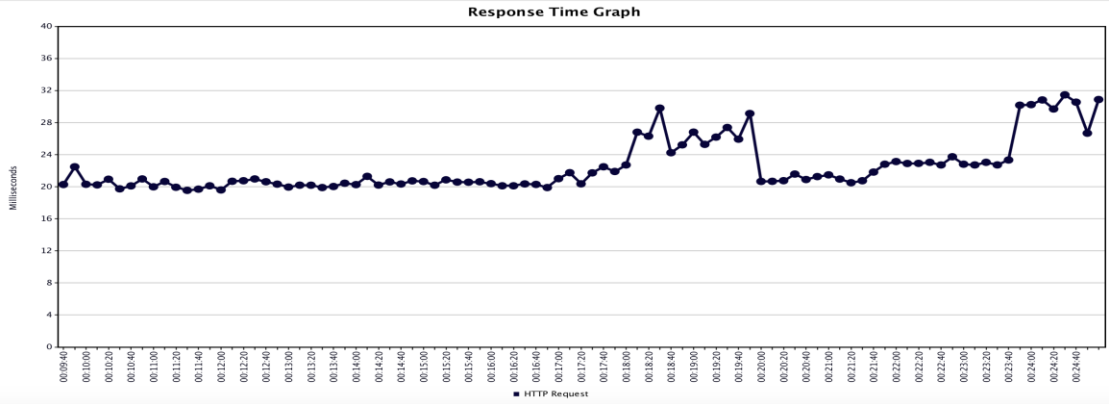


Figure 6.26: Data analysis for admin requests

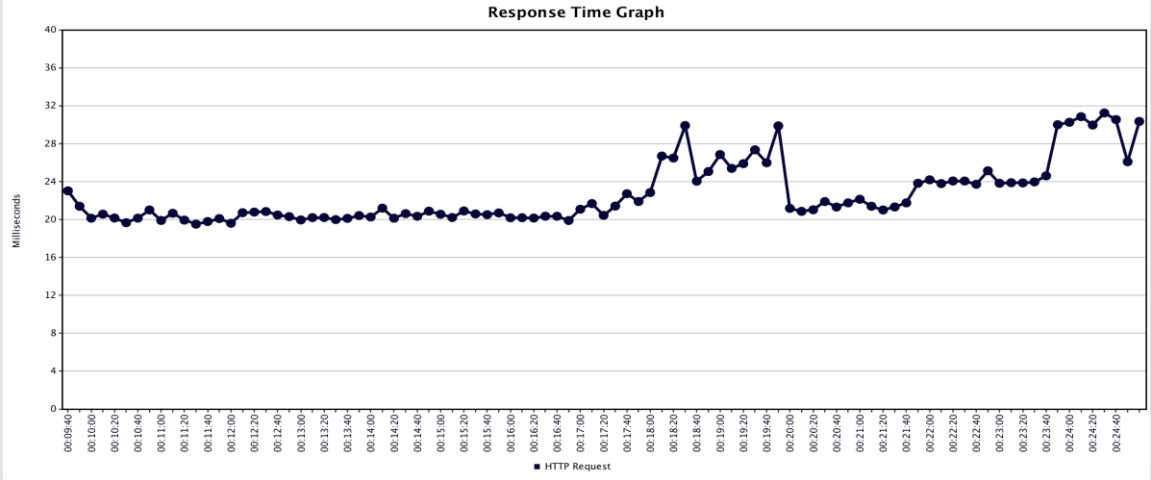


Figure 6.27: Data analysis for user requests

The average time to request operations for all the users was 15 min 30 s, as shown in Figure 6.28, which proved that the system ran smoothly and presented an acceptable level of performance when compared to the experiment performed on the ACL-based application.

Figure 6.28: Average total time of the experiment

Figure 6.29 illustrates the results of the two experiments side-by-side, where the top two graphs represent our security service approach to the bank application and the bottom two graphs represent the ACL approach. The graphs of the security service show an acceptable increase in the response time as compared to that in the ACL experiment. This was

attributed to the use of encryption.

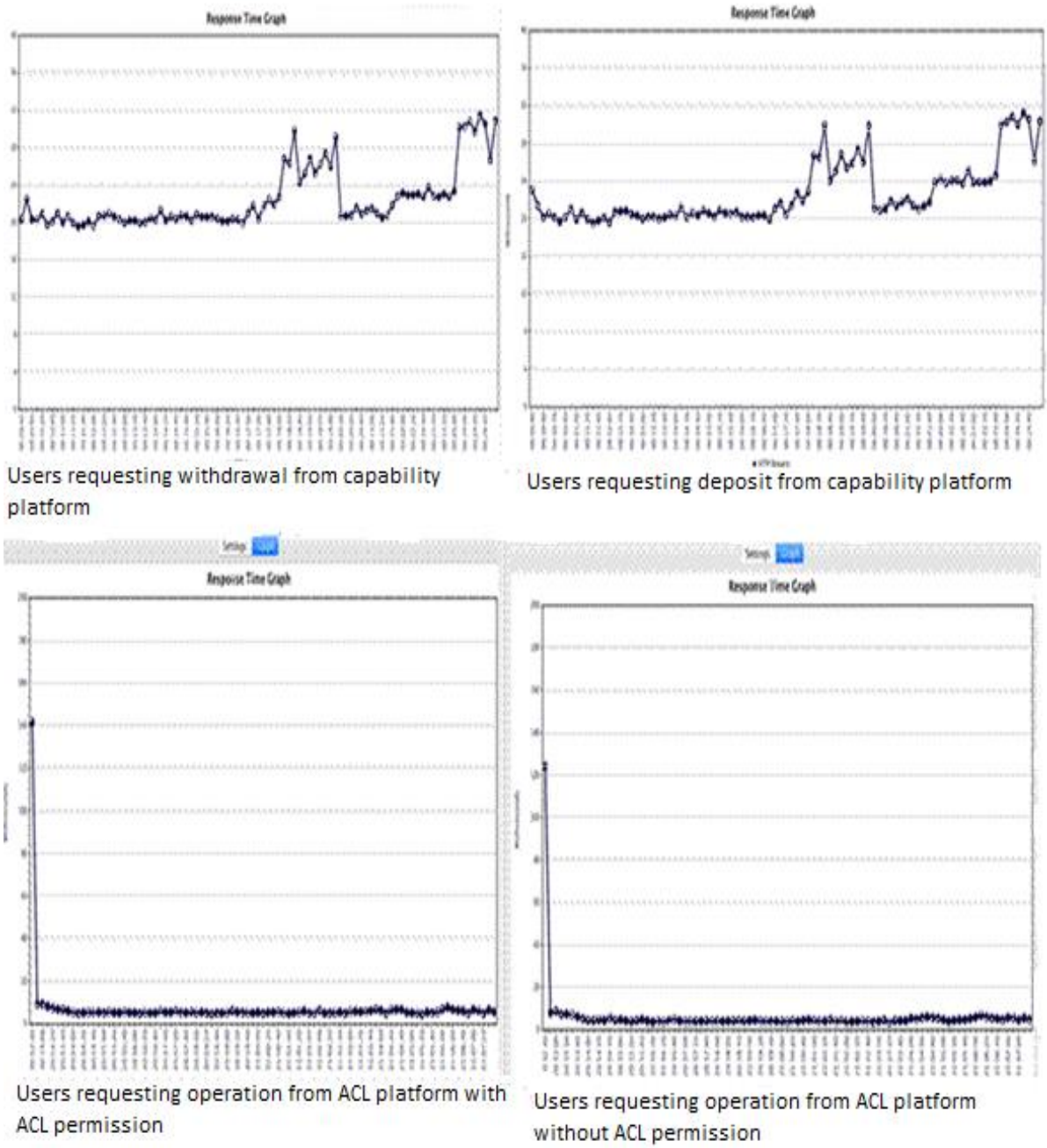


Figure 6.29: Comparison between the two experiments

Chapter 7

Conclusion and future work

7.0 Introduction

The primary focus of this thesis was the development of an access control scheme for users when accessing applications. The study developed a module structure that separated the functions and the data and proposed and implemented a new linking mechanism to combine the modules and the users. An API that realised the research objectives was designed and implemented in this thesis. The objectives of this research are explained in Table 7.1.

	Objective	**Investigation method**	**Chapter(s)**
1	To analyse in detail the existing cloud computing systems and the conventional computer systems that host the cloud	Review the existing literature and industry documents	2
2	To analyse the security and networking requirements for cloud systems and evaluate the relevant security, resilience authentication, disruption and attacks	Review the existing literature on security and networks	2
3	To study and propose a trusted OS that was built with the aim to resolve all protection problems found in other operating systems	Review the existing literature on computer systems and the SPEEDOS system	3
4	To propose a unique general structure to build cloud software applications that will reduce/prevent unauthorised access from hackers and enhance the cloud clients' data protection	Create a system design logical overview	4

	Objective	Investigation method	Chapter(s)
5	To propose an encryption mechanism to encrypt/decrypt data during transition and rest	Review the system design on protection methods	4
6	To create a new architecture model that mitigates cloud computing threats and to reduce security concerns	Use the logical design overview	4 and 5
7	To validate and evaluate the research, and suggest directions for future research	Summarise the findings of the research	6 and 7

Table 7.1: Summary of research objectives

The proposed scheme's objectives were realised through the new platform. The platform also addressed the following research questions introduced in Section 1.4.

7.1 Research question 1

Is it possible to develop a cloud environment that provides useful services while ensuring that all the data and communications are safe from unauthorised access?

This research showed that it is possible to achieve a secure cloud environment by introducing a better designed structure for modules and module communication. The modules represented as multiple routines that get successfully called only from whoever has the right capability to permit the call.

The protection mechanism used in the platform added a new security level to data, by providing the users with the capability to initiate their actions on the modules. The

capability identifies user ID, data access, access count and access type. Capability revocation was considered to reduce the malicious attacks.

The mechanism was developed by structuring the main modules into multiple routines. Calls came into a specific routine, and the authorisation of the call was implemented by the security service. Actions were taken to prevent unauthorised access on the basis of the capabilities and validation of the users IDs. The actions can be rejecting the call, granting the call or progress to further checks on the basis of the cloud provider's requirements.

The caller calls a function of a cloud application that passes the call onto the security service which confirms the user details and the requested operation with the capabilities of this caller from the database. If the details match, the security service sends the cloud application the approval for this call. If the details do not match, the call will be rejected. All the capabilities are stored in the security service database and can only be accessed from the security service. The capabilities are kept encrypted in transit and at rest, with no method to decrypt the capabilities without providing the encryption keys.

7.2 Research question 2

How can a new platform prevent/reduce the most concerning attacks that threaten the existing cloud systems, while ensuring that the attackers do not have access to new weaknesses that could leave the system vulnerable?

As described in the previous chapter, the implemented platform could be used to design a security solution that ensured that only legitimate users accessed the data. The platform provided protection against some of the most dangerous attacks in the cloud environment.

Experiments and system analyses were used to justify and test the system security against malware, malicious software, eavesdropping, man-in-the-middle and SQL injection attacks. The platform showed a very strong level of protection against these attacks, and the results of the experiments clarified that there were no data breaches identified during the testing.

7.3 Research question 3

Could such a system perform at an acceptable rate, or would it be too slow to be practical? Experimental comparisons implemented as part of this work demonstrated that the performance of the new platform was as expected, even in the presence of a malicious workload.

7.4 Future work

Despite the fact that the characterised goals set in Section 1.3 were met by the work introduced in this thesis, a few likely outcomes exist for future extension. The future direction regarding extending this research work is described below.

1 Implement the full SPEEDOS memory environment to provide an IaaS cloud service

The platform was built using the main security features of SPEEDOS to offer PaaS/SaaS services for the cloud system. However, the SPEEDOS memory is secure for the IaaS provision. Therefore, it is a valuable plan to implement an approach to memory that has the SPEEDOS memory features deployed as an IaaS service over the Internet.

2 Enhance the platform performance in the long run

The platform was tested on two cloud applications: the bank application and the social media application. However, more applications need to be connected to the new platform in a more complex environment to prove the effectiveness of the security service over time.

3 Implementing qualifier modules

The qualifier modules explained in the SPEEDOS system (Section 3.8) provide an additional level of checks on the calls. Future work may include the application of this concept to the modules in this platform. This will increase the security of calls in our platform and potentially prevent suspicious activities.

4 Enhancing the capabilities of the security service to be more effective in protecting the client application from malicious activities.

The security service's ability to protect against malicious calls from client applications is currently a risk. In the future, mechanisms providing increased protection will be built on the basis of a statistical analysis of calls, to detect when a client application is hacked. For example, on average, if an application makes five calls a day to the security service and suddenly the same application starts making 5000 calls, then the security service will pro-actively flag the client application as a security threat and inform the application administrators about the situation.

Bibliography

1. Manyika, J.C., Michael Brown, Brad Bughin, Jacques Dobbs, Richard Roxburgh, Charles Byers, Angela Hung McKinsey Global Institute, Big data: The next frontier for innovation, competition, and productivity. 2011.

2. Tripathi, A.M., Abhinav. Cloud computing security considerations. in Signal Processing, Communications and Computing (ICSPCC), 2011 IEEE International Conference on. 2011. IEEE.

3. Subashini, S.K., V, A survey on security issues in service delivery models of cloud computing. Journal of Network and Computer Applications, 2011. 34(1): p. 1-11.

4. Fleece, J., The Top 3 Reasons Cloud Computing Implementations Fail, [Online] Available: http://blog.sungardas.com/2015/03/the-top-3-reasons-cloud-computing-implementations-fail/ . Last accessed 7/5/2016.

5. Mathur, P.N., Nikhil. Cloud computing: New challenge to the entire computer industry. in Parallel Distributed and Grid Computing (PDGC), 2010 1st International Conference on. 2010. IEEE.

6. Anala, M.S., Jyoti Shobha, G. A framework for secure live migration of virtual machines. in Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on. 2013. IEEE.

7. Tanenbaum, A.S. and A.S. Woodhull, Operating systems: design and implementation. Vol. 2. 1987: Prentice-Hall Englewood Cliffs, NJ.

8. Keedy, D.L., The SPEEDOS computer archetecture, [Online] Available: http://www.speedos-security.org/. Last access 21/9/2017.

9. Mell, P. and T. Grance, The NIST definition of cloud computing, [Online]. Available: http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf. 2011

10. Liu, D.K., Utkarsh Lung, Chung-Horng. A light weight SLA management infrastructure for cloud computing. in Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on. 2013. IEEE.

11. Firdhous, M.G., Osman Hassan, Suhaidi. A trust computing mechanism for cloud computing. in Kaleidoscope 2011: The Fully Networked Human?-Innovations for Future Networks and Services (K-2011), Proceedings of ITU. 2011. IEEE.

12. Hogan, M., et al., Nist cloud computing standards roadmap. NIST Special Publication, 2011. 35: p. 6-11.

13. Grobauer, B., T. Walloschek, and E. Stocker, Understanding Cloud Computing Vulnerabilities IEEE Security & Privacy, 2011. 9(2): p. 50-57.

14. Rajan, S.J., Apurva. Cloud computing: The fifth generation of computing. in Communication Systems and Network Technologies (CSNT), 2011 International Conference on. 2011. IEEE.

15. Sanjeev Aggarwal, P., Laurie McCabe, Partner, The TCO Advantages of SaaS-Based Budgeting, Forecasting & Reporting. A Hurwitz wHite PAPer, 2016.

16. Gianpaolo Carraro, F.C., Software as a Service (SaaS): An Enterprise Perspective [Online] Available: http://msdn.microsoft.com/en-us/enus/library/aa905332.aspx. Last accessed 23/2/2014.

17. Zhang, S.C., Xuebin Zhang, Shufen Huo, Xiuzhen. The comparison between cloud computing and grid computing. in Computer Application and System Modeling (ICCASM), 2010 International Conference on. 2010. IEEE.

18. Riglian, A., PaaS system benefits go beyond just freeing up developers' time, [Online] Available: http://searchcloudapplications.techtarget.com/feature/PaaS-system-benefits-go-beyond-just-freeing-up-developers-time. Last accessed 18/7/2015.

19. Google cloud platform, [Online] Available: https://cloud.google.com/appengine/docs. Last accessed 14/6/2016.

20. Azure., M., https://azure.microsoft.com/en-us/overview/what-is-paas/. Last accessed 9/11/2018.

21. Chou, T.-S., Security Threats On Cloud Computing Vulnerabilities. International Journal of Computer Science & Information Technology (IJCSIT) Vol, 2013. 5.

22. Kulkarni, G.G., Jayant Patil, Tejswini Dongare, Amruta. A security aspects in cloud computing. in Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on. 2012. IEEE.

23. Gong, C.L., Jie Zhang, Qiang Chen, Haitao Gong, Zhenghu. The characteristics of cloud computing. in Parallel Processing Workshops (ICPPW), 2010 39th International Conference on. 2010. IEEE.

24. Rocha, F.G., Thomas van Moorsel, Aad. Defense-in-depth against malicious insiders in the cloud. in Cloud Engineering (IC2E), 2013 IEEE International Conference on. 2013. IEEE.

25. Amazon cloud computing [Online]. Available: http://aws.amazon.com/ec2/. Last accessed 7/3/2017.

26. Verizon cloud solutions [Online] Available: http://www.verizonenterprise.com/solutions/cloud/. Last accessed 5/1/2018.

27. Rakspace Cloud, [Online] Available: https://www.rackspace.com/en-au/login. Last accessed 23/3/2018.

28. Reese, G., Cloud application architectures: building applications and infrastructure in the cloud 2009: " O'Reilly Media, Inc.

29. Sabahi, F. Virtualization-level security in cloud computing. in Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on. 2011. IEEE.

30. Hwang, J.Z., Sai Wu, FY Wood, Timothy. A component-based performance comparison of four hypervisors. in Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on. 2013. IEEE.

31. FERRO, G., Basics – Docker, Containers, Hypervisors, CoreOS, [Online] Available: http://etherealmind.com/basics-docker-containers-hypervisors-coreos/ . Last accessed 9/5/2016.

32. Smith, J.E.N., Ravi, The architecture of virtual machines. Computer, 2005. 38(5): p. 32-38.

33. Barham, P.D., Boris Fraser, Keir Hand, Steven Harris, Tim Ho, Alex Neugebauer, Rolf Pratt, Ian Warfield, Andrew, Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 2003. 37(5): p. 164-177.

34. Garg, S.K.B., Rajkumar, Green cloud computing and environmental sustainability,. Harnessing Green IT: Principles and Practices (2020): p. 315-340

35. Aljabre, A., Cloud computing for increased business value,. International Journal of Business and Social Science, 2012. 3(1): p. 234-239

36. Catteddu, D., Cloud Computing: benefits, risks and recommendations for information security, in Web application security. 2010, Springer. p. 17-17.

37. Singh, A.K., et al., A review of cloud computing open architecture and its security issues. International Journal of Scientific & Technology Research, 2012. 1(6): p. 65-67.

38. Pasquier, T.P., J. Expressing and Enforcing Location Requirements in the Cloud using Information Flow Control. in International Workshop on Legal and Technical Issues in Cloud Computing (Claw'15). IEEE. 2015.

39. Pfleeger, C.P, Is There a Security Problem in Computing? , INFORMIT [Online] Available: http://www.informit.com/articles/article.aspx?p=680830&seqNum=2. Last access 7/2017

40. KP, V.R.A., ND Srikantan, AV, Client Authorization and Secure Communication in Online Bank Transactions. International Journal of Scientific and Research Publications Retrived 2017: p. 455

41. WONDERHOWTO, Extracting Password techniques [Online] Available https://www.wonderhowto.com/search/extracting-password/. Last accessed 9/1/2017.

42. AirDefence, What hackers know that you dont know[Online]. Available: http://davidhoglund.typepad.com/integra_systems_inc_david/files/wlan_security_what_hackers_know_that_you_dont.pdf. Last accessed 4/8/2017.

43. SECTOOLS, Top 125 Network Security Tools, [Online] Available: http://sectools.org. Last accessed 3/7/2016.

44. DDOS PROTECTION CENTER [Online] Available: https://www.incapsula.com/ddos/ddos-attacks/. Last accessed 5/4/2015.

45. Asymmetric keys and encryption method [Online] Available: http://codingatschool.weebly.com/asymmetric-keys-and-encryption-methods.html. Last accessed 2/7/2017.

46. National Institute of Standards and Technology, Advanced Encryption Standard, [Online] Available: https://csrc.nist.gov/publications/detail/fips/197/final. Last accessed 9/5/2017.

47. AMSI Encryption, [Online]. Available: http://www.amsi.org.au/teacher_modules/pdfs/Maths_delivers/Encryption5.pdf. Last accessed 21/5/2017.

48. Rogaway, P. and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. in International workshop on fast software encryption. 2004. Springer. Last accessed 9/11/2018.

49. Madhuravani, B. and D. Murthy, Cryptographic Hash Functions: SHA Family. International Journal of Innovative Technology and Exploring Engineering (IJITEE). 2(4).

50. Alex, B. and L. Taylor, Spring Security Reference Documentation. Accessed on the world wide web on, 2008. 8(06): p. 2008.

51. Hallam-Baker, P., Security assertions markup language. May, 2001. 14: p. 1-24.

52. Migeon, J.-Y., The MIT Kerberos administrators how-to guide. 2008. 6.

53. Zissis, D.L., Dimitrios, Addressing cloud computing security issues. Future Generation Computer Systems, 2012. 28(3): p. 583-592.

54. Google Transparency Report, [Online] Available: http://www.google.com/transparencyreport/removals/government/?hl=en_GB. Last accessed 2/2/2016.

55. NCCgroup, Impact on cloud take up in financial services sector, [Online] Available: https://www.nccgroup.trust/en/newsroom/news/2014/10/cloud-fears-having-impact-on-cloud-take-up-in-financial-services-sector/. Last accessed 6/2/2014.

56. secure., C.W.S.O.n.m.s.c.p.t.s., [Online]. Available: https://www.sei.cmu.edu/news/article.cfm?assetid=52441&article=031&year=2012. Last accessed 5/7/2017.

57. Tan, X.A., Bo. The issues of cloud computing security in high-speed railway. in Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on. 2011. IEEE.

58. Alfath, A.B., Karim Baina, Salah. Cloud computing security: Fine-grained analysis and security approaches. in Security Days (JNS3), 2013 National. 2013. IEEE.

59. Hubbard, D. and M. Sutton, Top threats to cloud computing v1. 0. Cloud Security Alliance, [Online]. Available: https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf., 2010: p. 1-14.

60. Arrington, M., "In our inbox: Hundreds of confidential twitter documents. July 2009.[Online]. Available: http://techcrunch.com/2009/07/14/in-our-inbox-hundreds-of-confidential-twitterdocuments, .Last accessed 10/2009.

61. Danchev, D., ZDNET: french hacker gains access to twitter's admin panel. April 2009.[Online]. Available: http://www.zdnet.com/blog/security/french-hacker-gains-access-to-twitters-adminpanel, 2009. 3292.

62. Vodafone, vodafone-launches-cloud-and-hosting-in-germany [Online] Available: http://cloud.vodafone.com/cs/ch/resource_centre/2014-03-09-vodafone-launches-cloud-and-hosting-in-germany.jsp. Last accessed 1/9/2014.

63. Express, Data theft hits Vodafone customers,[Online] Available: http://www.express.co.uk/news/world/428819/Data-theft-hits-Vodafone-customers. Last accessed 15/9/2014.

64. Reuters, Credit card data breach targets Marriott, Sheraton, other hotels,[Online] Available: https://www.reuters.com/article/us-whitelodging-databreach/credit-card-data-breach-targets-marriott-sheraton-other-hotels-idUSBREA1300120140204. Last accessed 10//9/2014.

65. Neumann, P.G., Risks and myths of cloud computing and cloud storage,. Communications of the ACM, 2014. 57(10): p. 25-27

66. Skariachan, D., Amazon Down Briefly In U.S. and Canada,[Online] Available: https://www.huffingtonpost.com/2013/08/19/amazon-down_n_3781362.html. Last accessed 5/1/2015.

67. Alpeyev, P., J. Galante, and M. Yasu, Amazon. com server said to have been used in sony attack, [Online] Available: https://www.bloomberg.com/news/articles/2011-05-13/sony-network-said-to-have-been-invaded-by-hackers-using-amazon-com-server Bloomberg, May, 2011. 14.

68. Armerding, T., The 15 worst data security breaches of the 21st Century,. COS Security and Risk, [Online]. Available: http://www.csoonline.com/article/2130877/data-protection/the-15-worst-data-security-breaches-of-the-21st-century.html, 2012.

69. Mills, E., Hackers release credit card, other data from Stratfor breach,[Online] Available: http://www.cnet.com/news/hackers-release-credit-card-other-data-from-stratfor-breach/. CNET News, December, 2011.

70. Khorshed, M.T.A., ABM Shawkat Wasimi, Saleh A. Trust issues that create threats for cyber attacks in cloud computing. in Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on. 2011. IEEE.

71. Shetty, J., M. Anala, and G. Shobha, A survey on techniques of secure live migration of virtual machine ,[Online] Available: http://research.ijcaonline.org/volume39/number12/pxc3877305.pdf. International Journal of Computer Applications, 2012. 39(12): p. 34-39.

72. CERT Insider Threat Center [Online] Available: http://www.cert.org/insider-threat/index.cfm. Last accessed 1/6/2017.

73. Claycomb, W.R.N., Alex. Insider threats to cloud computing: Directions for new research challenges. in Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual. 2012. IEEE.

74. Preimesberger, C., DDoS attack volume escalates as new methods emerge,[Online] Available: http://www.eweek.com/security/slideshows/ddos-attack-volume-escalates-as-new-methods-emerge.html. Eweek, 2014.

75. Bao, X., et al., NSFOCUS DDoS Threat Report 2013, [Online] Available: http://en.nsfocus.com/2014/SecurityReport_0320/165.html. NSFOCUS Information Technology, 2013.

76. Ristenpart, T.T., Eran Shacham, Hovav Savage, Stefan. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. in Proceedings of the 16th ACM conference on Computer and communications security. 2009. ACM.

77. Zhang, Y.J., Ari Reiter, Michael K Ristenpart, Thomas. Cross-VM side channels and their use to extract private keys. in Proceedings of the 2012 ACM conference on Computer and communications security. 2012. ACM.

78. Geffner, J., VENOM: Virtualized Environment Neglected Operations Manipulation. [Online]. Available: http://venom.crowdstrike.com, 2015.

79. Wagenseil, P., Sneak Attack: Android Apps Can Ambush Each Other, [Online] Available: http://www.tomsguide.com/us/android-gui-sneak-attack,news-19371.html. 2014.

80. Norton, Spear Phishing: Scam, Not Sport, [Online] Available: http://us.norton.com/spear-phishing-scam-not-sport/article. 2016.

81. Phishing Attack, [Online] Available: http://en.wikipedia.org/wiki/Phishing. Last accessed 9/1/2016.

82. McCall, T., Gartner survey shows phishing attacks escalated in 2007,. More Than $3 Billion Lost to These Attacks, [Online] Available: http://www.gartner.com/newsroom/id/565125, 2007.

83. NEWS™, B.A.D.I., Phishing Attacks Keep Hitting Gmail: Be careful, [Online] Available: http://www.themountainsentinel.com/documents/2011/08-22-11.pdf. 2011.

84. Singh, J.K., Brajesh Khatri, Asha. Improving stored data security in Cloud using Rc5 algorithm. in Engineering (NUiCONE), 2012 Nirma University International Conference on. 2012. IEEE.

85. Wang, Z.S., Kun Jajodia, Sushil Jing, Jiwu. Disk storage isolation and verification in cloud. in Global Communications Conference (GLOBECOM), 2012 IEEE. 2012. IEEE.

86. Strasser, M.S., Heiko, A software-based trusted platform module emulator, in Trusted Computing-Challenges and Applications. 2008, Springer. p. 33-47.

87. Trusted computing Group, Protect Your Data and Enhance Security, [Online] Available:

http://www.trustedcomputinggroup.org/resources/protect_your_data_and_enhance_security. Last accessed 1/2/2016.

88. Jog, M.M., M. Cloud Computing: Exploring security design approaches in Infrastructure as a Service. in Cloud Computing Technologies, Applications and Management (ICCCTAM), 2012 International Conference on. 2012. IEEE.

89. Espenlaub, K., Design of the SPEEDOS Operating System Kernel, [Online] Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.127.1238&rep=rep1&type=pdf. 2005, Universität Ulm.

90. Johnstone, M.S. and P.R. Wilson. The memory fragmentation problem: solved? in Acm Sigplan Notices. 1998. ACM.

91. Keedy, D.L., computer security an ongoing problem, Private communication and part of book in preparation. Last accessed 5/8/2017.

92. Feske, N., Operating System Framework, [Online] Available: http://genode.org/documentation/genode-foundations-15-05.pdf. Last accessed 8/2/2017.

93. Keedy, D.L., The Monads Project ,[Online] Available: http://www.monads-security.org/. Last accessed 10/2017.

94. Dearle, A., et al., Grasshopper: An orthogonally persistent operating system. Computing Systems, 1994. 7(3): p. 289-312.

95. Parnas, D.L., Information distribution aspects of design methodology,[Online] Available: https://link.springer.com/chapter/10.1007/978-3-642-59412-0_25. Last accessed 8/ 2016.

96. HENSKENS, F.A., A CAPABILITY-BASED PERSISTENT DISTRIBUTED SHARED MEMORY,[Online] Available: http://www.cs.newcastle.edu.au/~henskens/papers/acsc91.pdf. 5/1991.

97. Delp, G. S. "The Architecture and Implementation of Memnet: a High-Speed Shared-Memory Computer Communication Network", University of Delaware, Udel-EE Technical Report Number 88-05-1, 1988.

98. Keedy, D.L., A SHORT INTRODUCTION TO TIMOR, [Online] Available: http://www.timor-programming.org/. Last accessed 8/1/2018.

99. TechTarget, Cloud applications: A programming languages background, [Online] Available: https://searchitchannel.techtarget.com/feature/Cloud-applications-A-programming-languages-background. Last accessed 8/1/2018.

100. Techbeacon, 5 steps to building a cloud-ready application architecture, [Online] Available: https://techbeacon.com/5-steps-building-cloud-ready-application-architecture. Last accessed 9/2/2018.

101. Keedy, J.L., A model for security and protection in persistent systems. Microprocessors and Microsystems, 1993. 17(3): p. 139-146.

102. O'Hearn, P.W.Y., Hongseok Reynolds, John C, Separation and information hiding. ACM Transactions on Programming Languages and Systems (TOPLAS), 2009. 31(3): p. 11.

103. Puppet, Module fundamentals Puppet, [Online] Available: https://puppet.com/docs/puppet/5.3/modules_fundamentals.html. Last accessed 4/1/2018.

104. Bardhan, N. and P. Singh, Operating System Used in Cloud Computing, [Online] IJCSIT) International Journal of Computer Science and Information Technologies, 2015. 6(1): p. 542-544

105. Adamalthus, Cloud Computing and Complexity, [Online]. Available: http://www.adamalthus.com/blog/2013/06/05/cloud-computing-and-complexity/. Last accessed 21/1/2018.

106. Sheng Liang, C., and Peder Ulander, CMO, Cloud.com, requirements-for-building-your-cloud-infrastructure, [Online] Available: https://www.cio.com/article/2412506/cloud-computing/7-requirements-for-building-your-cloud-infrastructure.html. 2010.

107. Ghrab, A., et al., Towards A Standards-Based Cloud Service Manager, [Online] Available: https://orbi.uliege.be/bitstream/2268/160750/1/Towards%20A%20Standards-Based%20Cloud%20Service%20Manager.pdf. Last accessed 6/2017.

108. Brebner, P.L., Anna. Modeling cloud cost and performance. in Proceedings of Cloud Computing and Virtualization Conference (CCV 2010), Singapore. 2010. Citeseer.

109. Sitepoint, the-3-myths-of-learning-programming-languages, [Online] Available: https://www.sitepoint.com/the-3-myths-of-learning-programming-languages/. Last accessed 2/3/2018.

110. Selecting the optimal programming language, [Online] Available: https://www.ibm.com/developerworks/library/wa-optimal/. Last accessed 2/ 2018.

111. 8 FACTORS THAT CAUSE SOFTWARE BUGS, [Online] Available: https://blog.testfort.com/automated-testing/8-factors-that-cause-software-bugs. Last accessed 15/3/2018.

112. Mansuri, A. and P. Rathore, Cloud computing: A new era in the field of information technology applications and its services. American Journal of Information Systems, 2014. 2(1): p. 1-5.

113. Explaining computers, [Online] Available: http://www.explainingcomputers.com/hardware.html . Last accessed 6/3/2017.

114. Lelli, V.B., Arnaud Baudry, Benoit. Classifying and qualifying GUI defects. in Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on. 2015. IEEE.

115. Kurdi, H.A.H., Safwat Khalifa, Amal. Towards a friendly user interface on the cloud. in International Conference of Design, User Experience, and Usability. 2014. Springer.

116. IBM cloud, [Online]. Available: https://www.ibm.com/cloud-computing/au/en/. Last accessed 14/4/2018.

117. Rakspace Cloud, [Online] Available: https://www.rackspace.com/en-au/login. Last accessed 23/3/2018.

118. GoDaddy cloud, [Online] Available: https://au.godaddy.com/. Last accessed 18/4/2018.

119. Knowledge, D., Refining Your Criteria for Data Center Site Selection, [Online] Available: http://www.datacenterknowledge.com/archives/2013/07/24/refining-your-criteria-for-data-center-site-selection. Last accessed 8/4/2018.

120. LIFELINE, 5 FACTORS TO CONSIDER WHEN BUILDING A GLOBAL DATA CENTER, [Online] Available: https://lifelinedatacenters.com/data-center/building-global-data-center/. Last accessed 5/4/2018.

121. GRAILS Security, [Online] Available: https://grails.org/wiki/version/Security/1. Last accessed 1/4/2018.

122. Java T Point, RMI (Remote Method Invocation), [Online] Available: https://www.javatpoint.com/RMI. Last accessed 4/3/2018.

123. Shah, M., Spring Remoting: Hessian, [Online] Available: http://www.studytrails.com/frameworks/spring/spring-remoting-hessian/. Last accessed 18/3/2018.

124. Grails Remoting Plugin, [Online] Available: https://grails.org/plugin/remoting. Last accessed 9/4/2014.

125. Microsoft, How Serialization Works, [Online] Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/serialization/. Last accessed 9/3/2018.

126. MySQL Database, [Online] Available: https://www.mysql.com/. Last accessed 9/8/2017.

127. Oracle database, [Onlien] Available: http://www.oracle.com/technetwork/database/enterprise-edition/overview/index.html. Last accessed 9/11/2017.

128. H2 database Engine, [Online] Available: http://www.h2database.com/html/main.html. Last accessed 5/12/2017.

129. Postgresql database, [Online] Available: https://www.postgresql.org/. Last accessed 1/2/2018.

130. Download Grails, [Online] Available: http://grails.org/download.html. Last accessed 1/3/2018.

131. Grails plugins, [Online] Available: https://grails.org/plugins. Last accessed 1/4/2018.

132. BCrypt Encryption, [Online] Available: https://docs.spring.io/spring-security/site/docs/4.2.4.RELEASE/apidocs/org/springframework/security/crypto/bcrypt/BCrypt.html. Last accessed 2/3/2018.

133. JVisualVM, [Online] Available: https://blog.fastthread.io/2016/06/06/how-to-take-thread-dumps-7-options/. Last accessed 1/2/2018.