Using virtualisation to create a more secure online banking infrastructure

by

Jaco Louis du Toit

DISSERTATION

Submitted in the fulfilment of the requirements for the degree

MASTER OF SCIENCE

in

COMPUTER SCIENCE

in the

FACULTY OF SCIENCE

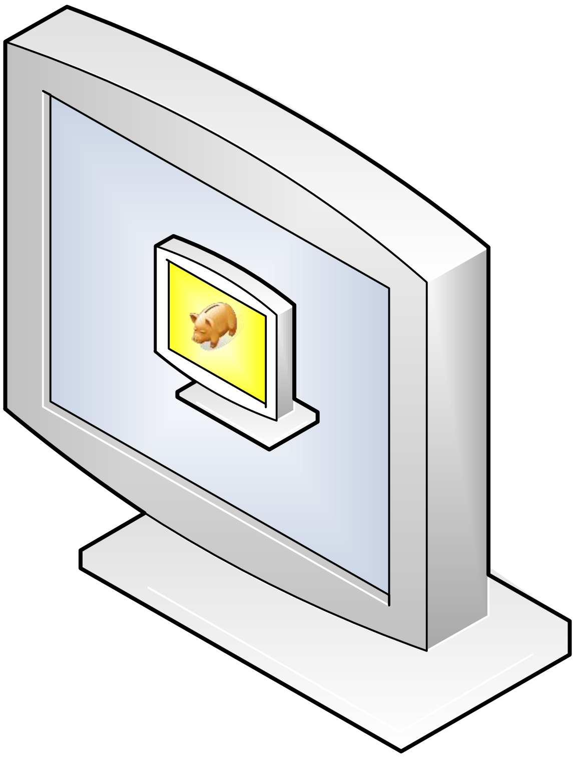
at the

UNIVERSITY OF JOHANNESBURG

SUPERVISOR: PROF. B. LOUWRENS

CO-SUPERVISOR: PROF. M. COETZEE

May 2013



Using virtualisation to create a more secure online banking infrastructure

2013

Acknowledgements

There are a number of people that had a big influence in getting me to write this dissertation. I would like to thank the following special people that assisted me and ensured that I completed this work:

- My wife, Ilse du Toit. She is my inspiration and my friend and my partner in life. Without my partner I would not be the person that I am today. Thank you for being the person you are and helping me with this endeavour.
- My supervisor, Professor Louwrens. Thank you for your commitment to the academic community and specifically for your assistance in getting my dissertation to its final form. Thank you also for giving me ideas for making the prototype work better.
- My co-supervisor, Professor Coetzee. Thank you for your advice at the beginning of this endeavour. Thank you also for all the work from your side that made it possible for me to get the dissertation into an academically accepted standard.
- My "unofficial" mentor, Professor von Solms. Thank you for persuading me to do this. Thank you also for always being there when I needed to discuss a "new idea".
- The University of Johannesburg, for giving me the opportunity, resources and time to complete my dissertation.

I would also like to thank all the other people not specifically mentioned in here that helped me make this a reality.

Abstract

Sim swop, Phishing, Zeus and SpyEye are all terms that may be found in articles concerning online banking fraud. Home users are unsure of how the configuration of their computers affects the risk profile for conducting online banking. Software installed by a home user on their computer may be malware designed to steal banking details. Customers expect banks to provide a safe online banking system. The challenge that banks have is that they cannot control the configuration that exists on a client operating system.

The V-Bank system was designed to determine whether virtualisation can be used as a means to increase the security for online banking. The V-Bank system uses a virtual machine that is run from a guest that is single purpose, read-only and fulfils the configuration requirements that the bank has for a client system. The V-Bank system also utilises public and private key encryption for identification, authentication and authorisation mechanisms in the online banking system.

The architecture of the V-Bank system defines online banking as an end-to-end system. It approaches online banking as a system that consists of three major components. The three major components is a client-side component, network and server-side environment. The V-Bank system gives banks the ability to provide customers with a system that is controlled from the client, through the network to the server.

The V-Bank system demonstrates that virtualisation can be used to increase the security of online banking.

Comment on writing style and numbering system

To promote easy reading the author uses the third-person singular personal pronouns “he”, “him”, “himself” to refer to a person of any gender.

References to specific sections are in the following format Chapter a:b.c.d. This refers to Chapter a, heading b.c.d

Example: Chapter 2:3.1.1 refers to Chapter 2, heading 3.1.1

Whenever not specifically referring to another source the figures were created by the author.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1: INTRODUCTION

1 Introduction

Doing banking over the Internet has grown rapidly in the last decade. Goldstuck reported a 28% growth in online banking in South Africa in 2003 (Goldstuck, 2004). Organised crime has taken the growth in online banking as an opportunity for exploitation using techniques such as malware and spamming to gain access to legitimate users' bank accounts allowing money mules to get access to the actual funds. (Florêncio & Herley, 2010).

Banks implement various controls to minimise online banking risks. One area, over which they have little control, is the user's computers. Banks have to depend on the user's security awareness and computer habits to ensure a safe end-to-end banking environment. (Karim, et al., 2009).

A number of studies have been conducted that shows that a virtual machine on the client can increase the security of computer systems (Wang, et al., 2010). A virtual machine increases the security of the computer system by taking control from the user. Security is also enhanced by isolating the online banking environment from the rest of the computing environment.

This chapter considers the problem that the study addresses and highlights the objectives that needs to be met by the study. The research methodology and research contribution are discussed. The chapter concludes by defining the structure of the dissertation.

The next section looks at the problem which the dissertation addresses.

2 Problem Statement

In 2010 CNet reported that the Zeus Trojan was used to steal more than $1 million from UK banking customer accounts (Mills, 2010). In an Internet security report from Symantec it was mentioned that over $9.5 million was stolen from bank accounts using the Zeus Trojan. In the same report a new Trojan called SpyEye was mentioned as the latest bank stealing Trojan (Symantic Enterprise Security, 2011).

Computer users at home have full control over their own computers. They can change the configuration and install software without realising the security implications for online banking. Users expect banks to provide them with a safe environment to conduct online banking.

Unfortunately banks cannot control what is installed on the computers of home users and cannot control the configuration of these computers.

The problems that this study attempts to resolve is that banks cannot control what users load on their computers, nor can they control how users' operating systems are configured. Many users are not security conscious and do not know what effect a specific piece of software has on the overall security of their computer. This leads to an environment where the operating system of a normal desktop computer is in an unknown, potentially insecure state. To conduct online banking in this unknown or insecure environment is a problem for both the banking customer and the bank.

To better understand the security problems faced by an online banking environment, a typical online banking session is described in the next section

3 Overview of Online Banking

Consider a typical online banking session:

John needs to pay AA-Electrical for some maintenance work conducted on his garage door. Before John starts, he ensures that he has the banking details for AA-Electrical.

Step 1: Pre-session start-up

John starts his computer in his study which runs Microsoft Windows 7 (Microsoft Corporation, 2013). John uses an ADSL Wi-Fi router to connect to the Internet. During the computer start-up the computer loads necessary device drivers so that the operating system can function properly with his computer hardware.

One of the device drivers that loads is for a printer that John has. The device driver was installed a few months ago after John downloaded it from the Internet.

Once the device drivers have loaded, the operating system starts a number of software applications automatically. One of them is his Anti-Virus program, whose license has expired and has not been updated in a while. Another application, which assists his children to choose between Internet based games, also starts in the background.

Figure 1:1 is a diagram depicting John's computer with the various peripherals attached to the computer, with the operating system and the Anti-Virus program and Internet Games Assistant program.

Once the computer has given John control of his mouse and keyboard John is ready to pay AA-Electrical.

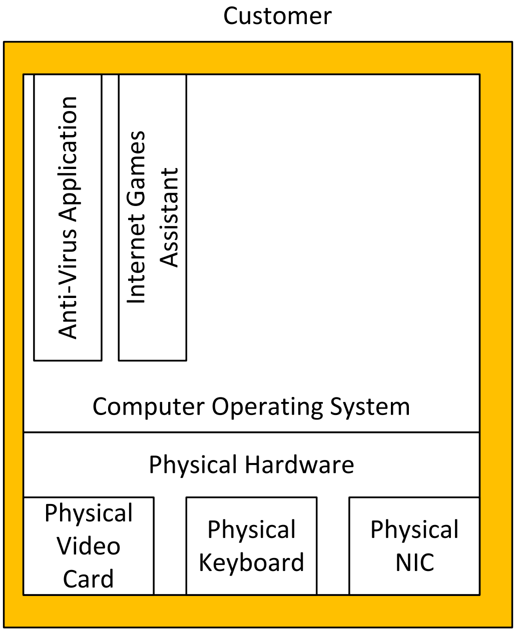


FIGURE 1:1 PRE-SESSION START-UP ENVIRONMENT

Step 2: Online banking security channel establishment

John starts his favourite web browser and selects the online banking site from one of his saved bookmarks as he does not like to type the online banking address every time.

The browser opens a connection to the Internet server at the bank's hosting facility. The web browser establishes an encrypted channel with the web server, by connecting to https://johnsbank.com/bank. The certificate on the web server ensures the identity of the server and helps to establish the encrypted channel. John does not scrutinize the information in the certificate that is displayed for him on the browser.

Figure 1:2 shows that there is now a web browser that is loaded on John's computer. The web browser established a secure channel with the bank's web server.

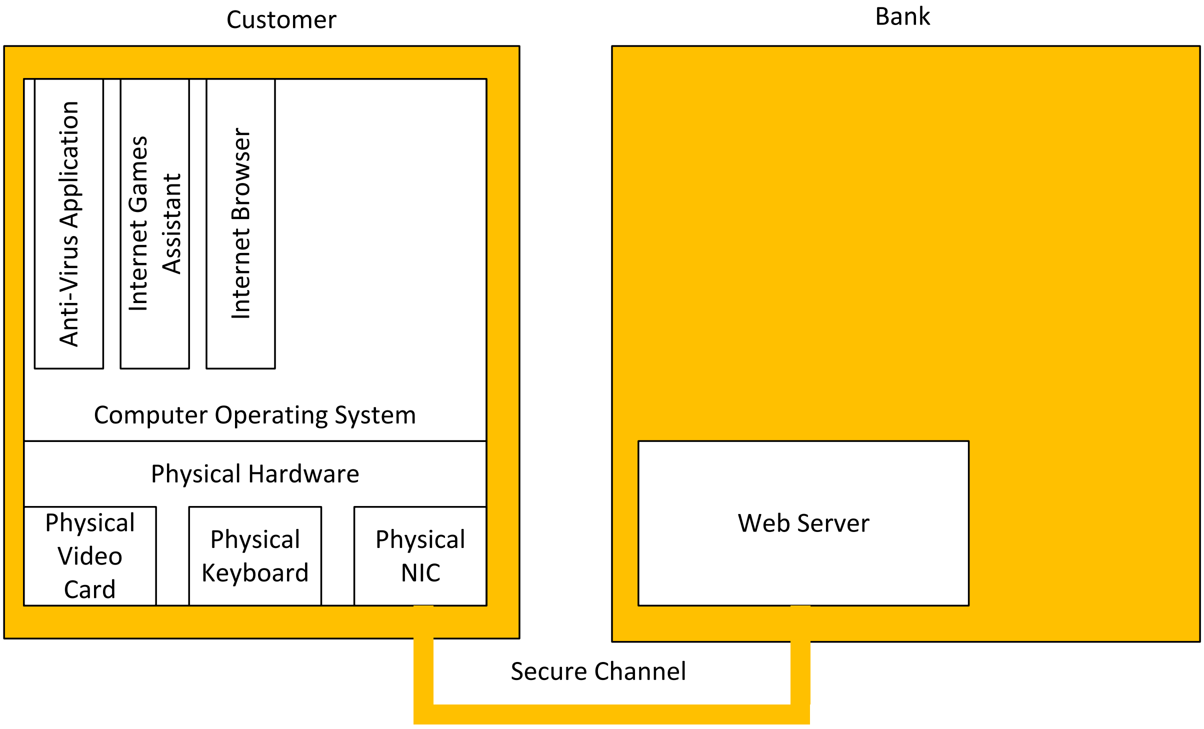


FIGURE 1:2 JOHN'S COMPUTER ESTABLISHES SECURE CHANNEL WITH BANK

The bank's logo is displayed on the web browser, together with a login area where John needs to type in his account number with a pin and password.

Step 3: Login authentication

John types in his account number, his pin and his online banking password in the fields provided for him. He uses the keyboard that is connected to his computer to do the actual typing.

The keyboard driver captures the keys as John types and presents the keys to the operating system. The Internet browser's software interface receives the keys from the operating system and stores in the login screen's elements that currently exist on the browser.

John clicks on the submit button to submit his login credentials to the bank. As John clicks on the submit button the browser automatically encrypts the keys typed in by John and posts it to the bank's online banking server. The online banking server receives the encrypted keys, decrypts it and sends it to the online banking system's authentication system.

The online banking system's authentication system verifies the given pin and password against the stored credentials for John's account number. The credentials match and the authentication system informs the web server that John has been authenticated. The web server establishes a connection to the bank's transaction server and receives general account information that is displayed to John on his web browser.

Figure 1:3 shows the Authentication Service as an extra component at the bank that authenticates login credentials supplied by online banking customers.

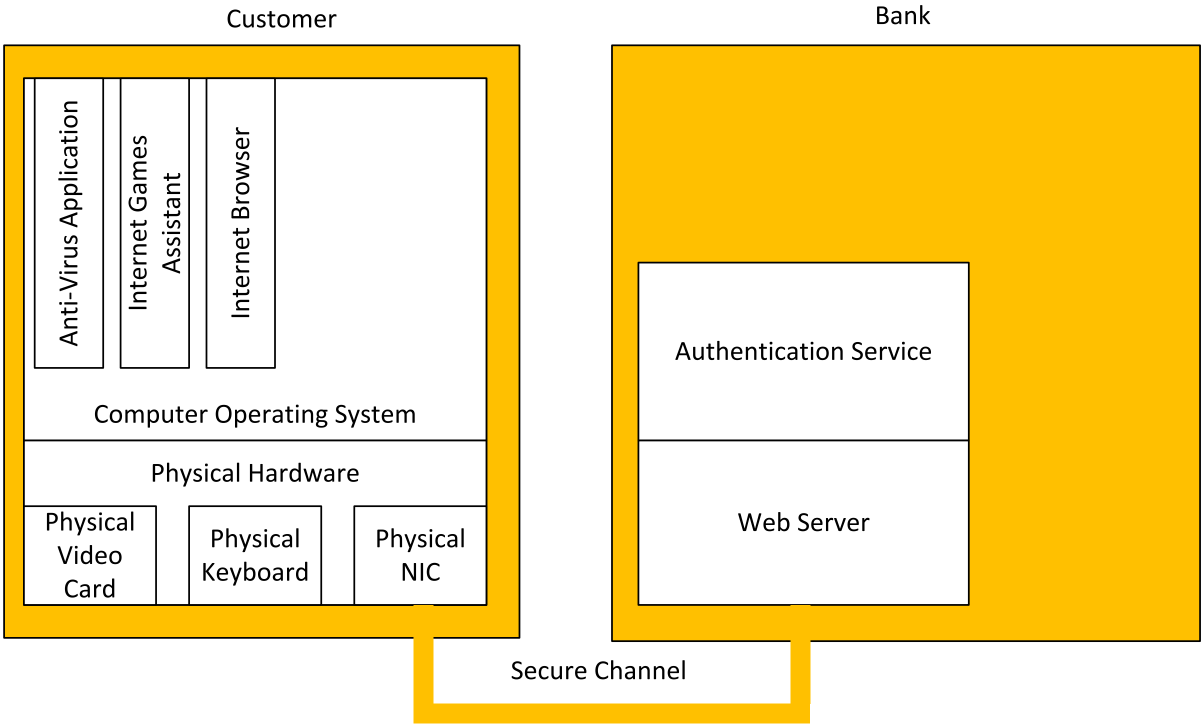


FIGURE 1:3 THE BANK AUTHENTICATES JOHN'S CREDENTIALS.

The bank's web server does not have access to John's computer and cannot determine whether John has a valid Anti-Virus program loaded, or that there may be potential malware installed on the his computer.

Step 4: Review banking statement

Before John pays AA-Electrical, John requests a statement of his cheque account. John can only see details of his cheque account and not his wife's cheque account, because the bank has only authorised John access to his own account.

Figure 1:4 depicts the Execution Service that ensures that account information is displayed to online banking customers. The execution service is authorised to only give information for the current logged in account and no one else.

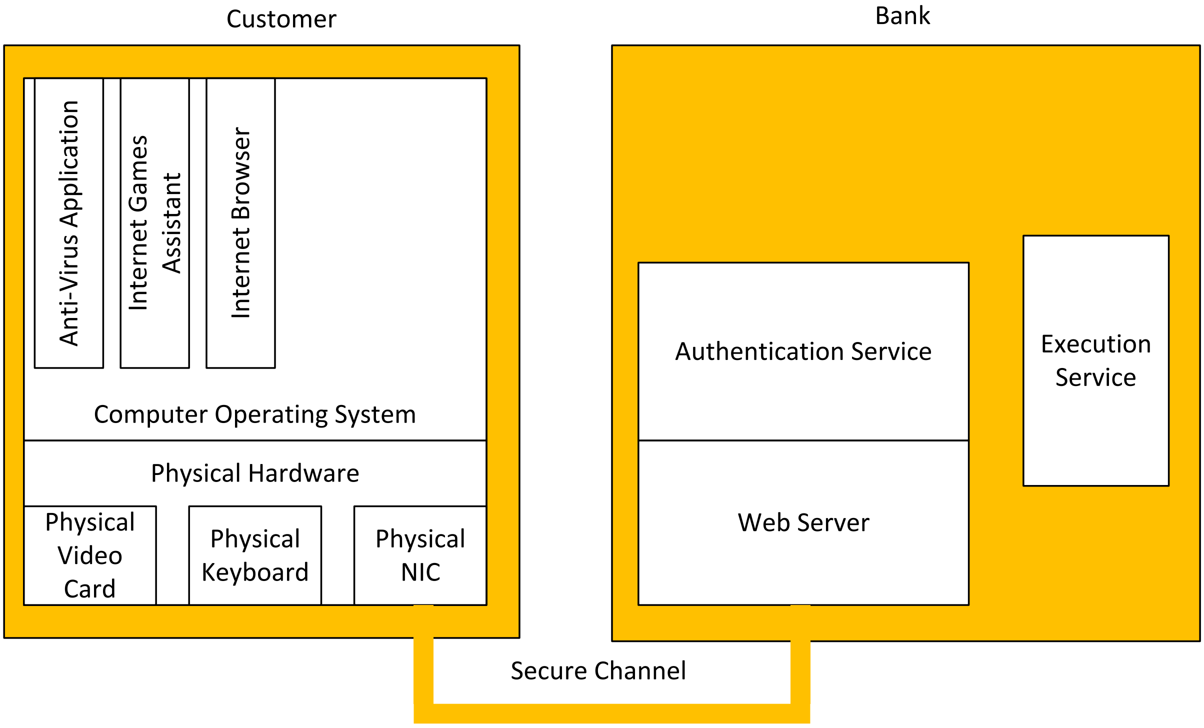


FIGURE 1:4 EXECUTION SERVICE PROVIDES ACCOUNT INFORMATION

Step 5: Pay third-party

John creates a beneficiary for AA-Electrical. The bank flags this type of action as a special case requiring additional authorisation. The bank has a policy that states that any transaction used to pay a new third party needs authorisation before being approved. This minimises the risk of enabling an attacker to transfer funds from the bank's customer account without authorisation.

After John types in the banking details of AA-Electrical, the bank sends a random authorisation number to John's mobile phone via a SMS (Short Message Service). John types the random authorisation number into the special field provided by the banking system on John's browser. The browser posts this number back to the bank and the bank verifies the number and authorises the creation of the new beneficiary.

John proceeds to pay the new beneficiary by providing the proper information such as, from which account, to which beneficiary, as well as other information like the amount to be paid. As soon as John clicks the submit button the bank verifies that enough funds exist in John's account and that he has not exceeded his daily payment limit. The bank then creates a transaction that is sent to the bank's transaction server so that the transaction can be fulfilled between John's bank and AA-Electrical's bank.

Figure 1:5 depicts the transaction service at the bank that manages transactions, which includes the payment of beneficiaries.

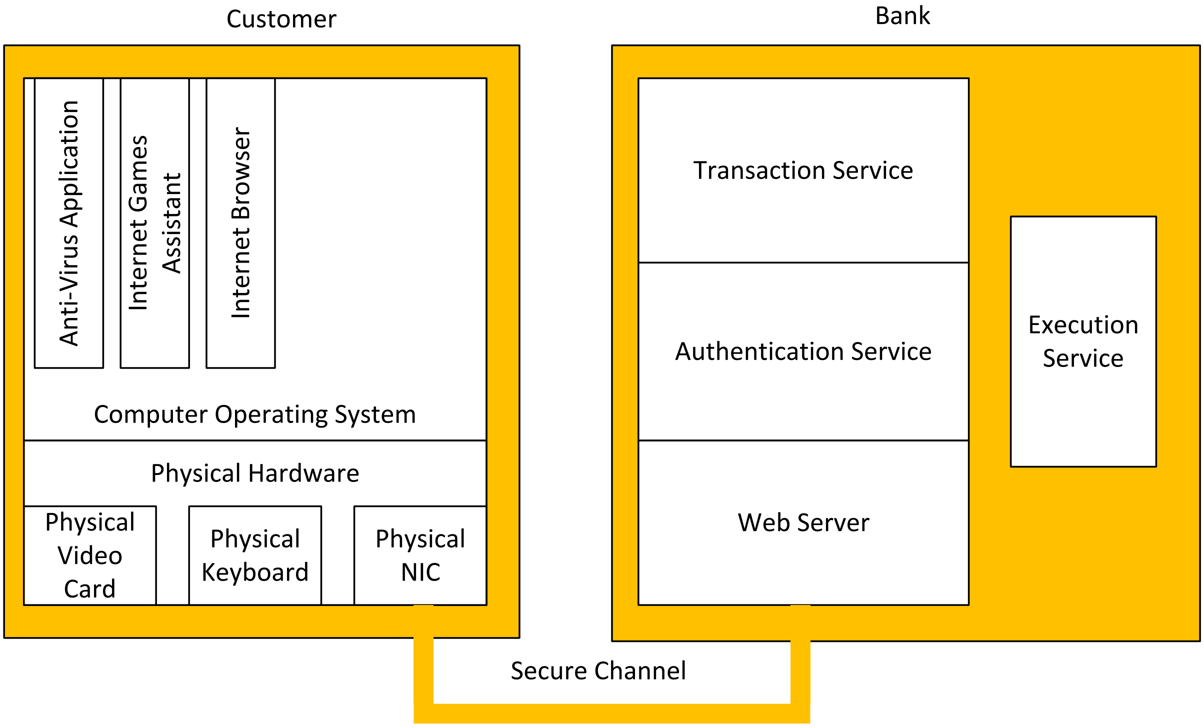


FIGURE 1:5 TRANSACTION SERVICE MANAGES PAYMENTS TO BENEFICIARIES

John requests a proof of payment from the bank and prints the proof of payment on his printer attached to his computer. John faxes the proof of payment to the accounting department at AA-Electrical to show that John submitted the payment transaction with his bank.

Step 6: Close

John concludes the banking session by closing the Internet browser.

The overview described in this section gives the reader an idea of what typically happens during an online banking session. The overview highlights the online banking environment that is used as a reference when evaluating the risks of online banking (Chapter 2) and the information security principles used in online banking (Chapter 3).

There are many components working together to enable the transfer of money between accounts. There are also many possible entry points to be used to compromise the online banking session. John's computer is not as secure as a bank controlled and configured computer.

The focus of this research is to comprehensively detail the security risks faced by the home user doing online banking. The applicability of using virtual machines to increase the security of online banking for home users is researched.

4 Hypothesis

This study hypothesises that banks can protect a home user by creating an online banking virtual machine application that can be configured to be in a known state. This known state can be evaluated and changed to conform to an accepted security configuration. This virtual machine, with its security configuration, can be locked down so that it cannot change. The virtual machine can then be given to a customer of bank as a system in which the user must do online banking that is more secure than his normal computer.

The next section considers what objectives need to be achieved in this dissertation.

5 Research Objective

In the previous section the research hypothesis was defined. It was stated that research is conducted in the use of virtualisation as a means to increase the security of online banking.

There are existing studies that consider virtualisation for safe Internet browsing (Wang, et al., 2010) (Cox, et al., 2006). This dissertation uses the idea of virtualisation, but applies it specifically to secure online banking from a user's computer point of view. To determine whether virtualisation is a viable option for secure online banking, a proof of concept system is produced.

The primary objective of the dissertation is to develop a proof of concept system using virtualisation that increases the security of online banking from an end-user's perspective. This proof of concept system is called the V-Bank solution.

Primary Objective:

Develop a proof of concept system using virtualisation to increase the security of online banking from an end-user's perspective

A number of secondary objectives for the solution are identified. These objectives include:

- Describe the security risks faced by online banking.
- Describe information security mechanisms used in online banking environments.
- Describe the security controls included in an online banking system.
- Identify the security advantages provided by virtual machines.
- Describe the architecture of an online banking system.
- Describe the isolation features that enhance the security of the online banking session.

The primary and secondary objectives act as drivers while conducting research and producing the V-Bank prototype solution.

The objectives of the study help us to define the deliverables of the study. The contribution of the research is defined next.

6 Research Contribution

It was mentioned in the research objective that a number of existing studies consider virtualisation as a solution that enables safe Internet browsing (Cox, et al., 2006) (Wang, et al., 2010).

The isolation properties of virtual machines are also used by England and Manferdelli to describe a model where a virtual machine is used to secure a host operating system (England & Manferdelli, 2006). Rowan defined a virtual machine that a user can use at any time for secure computing using a USB memory stick (Rowan, 2008).

Cox, et al. and Wang, et al. describe systems that use virtual machines for Internet browsing. The research by England and Manferdelli and Rowan describes general secure computing environments not just for Internet browsing or online banking.

The research in this dissertation considers the above mentioned studies, but applies it specifically for a typical home user that conducts online banking. It also includes not just the client-side virtual machine, but considers the network and server-side environment as part of the solution.

To achieve the objectives of the research, a specific research methodology was followed, which is defined in the next section.

7 Research Methodology

The methodology for the research follows the approach as described by Olivier (Olivier, 2009).

A literature survey is conducted, concentrating specifically on topics regarding security in online banking, online banking risks, virtualisation and the effect of existing research for online banking.

A list of design requirements for the prototype is established during the literature survey. An architectural framework for the prototype is defined that fulfils the design requirements. From the framework, the prototype is built and implemented and the prototype is evaluated against the stated hypothesis.

8 Structure of Dissertation

This dissertation is structured to support the research methodology shown in Figure 1:6. The dissertation has two main parts. The first, which is described at the top of Figure 1:6 is the literature survey. Chapters two to five form part of the literature survey.

The output of the literature survey is a list of design requirements for an online banking system.

The next major area in the dissertation revolves around the prototype. This includes chapters six to eight. In Figure 1:6, these chapters are marked in a square called: “Prototype design and implementation”

The dissertation ends with the conclusions and possible further research.

The last chapter in the dissertation is the bibliography.

Each of the chapters is discussed.

In chapter 1 the problem and the objectives of the research are defined. The hypothesis is defined and the reader is introduced to a typical online banking session

Chapter 2 discusses the information security risks that exist while conducting online banking. The chapter identifies the risks. To address a risk, a design requirement is formulated which must be met by an online banking system to make it a safer environment. A minimum set of design requirements is formulated from the existing controls that banks implement.

In chapter 3 the information security concepts in online banking are described. The chapter looks at information security concepts that apply to online banking. It includes the controls used by banks to implement these security concepts. The controls are evaluated and the existing set of design requirements is amended with the required controls.

Chapter 4 considers virtual machines. The different types of virtual machines are identified and describe the differences between virtual machines typically used on servers and desktops. The chapter also identifies some of the virtualisation options that exist. The chapter concludes with the security aspects that must be considered when implementing or designing for virtual machines. These considerations are defined as requirements that are added to the already established list of design requirements.

Some of the research that exists which address the security of Internet browsing by using virtualization are discussed in chapter 5. The chapter looks at a study that created a system called the "Tahoma" system. The chapter also looks at "SafeFox: A lightweight virtual browsing environment", "The secure virtual computer on your keychain" and "A Novel Security Scheme for Online Banking Based on Virtual Machine". The solutions described by these studies are evaluated against the design requirements that were created in chapters two to four. The existing design requirements are amended to include applicable design principles from the existing studies.

Chapter 6 establishes a framework architecture against which the V-Bank prototype is built. The chapter revisits the problem statement and discusses the approach that was taken to define the architecture. The chapter discusses the detail of each architectural component, establishing the link between the component and the design requirements that were the result of chapters two to five.

In chapter 7 the implementation of the framework architecture is described. The chapter takes each component that was defined in chapter 6 and see how it is implemented to establish the V-Bank prototype.

Chapter 8 describes and evaluates the prototype as it is implemented. The chapter describes how the prototype works and highlights the security features. The system is then evaluated against the design requirements that were defined in chapters two to five. The chapter then highlights the areas of the solution where it did not fulfil specific design requirements.

Chapter 9 ends with a review of the problem statement, review the objectives and look at the solution that was described. It lists areas where further research may be required.

Chapter 10 contains the bibliography of references used in the dissertation.

The dissertation ends with an appendix that describes implementation steps and settings that are used in creating the V-Bank prototype.

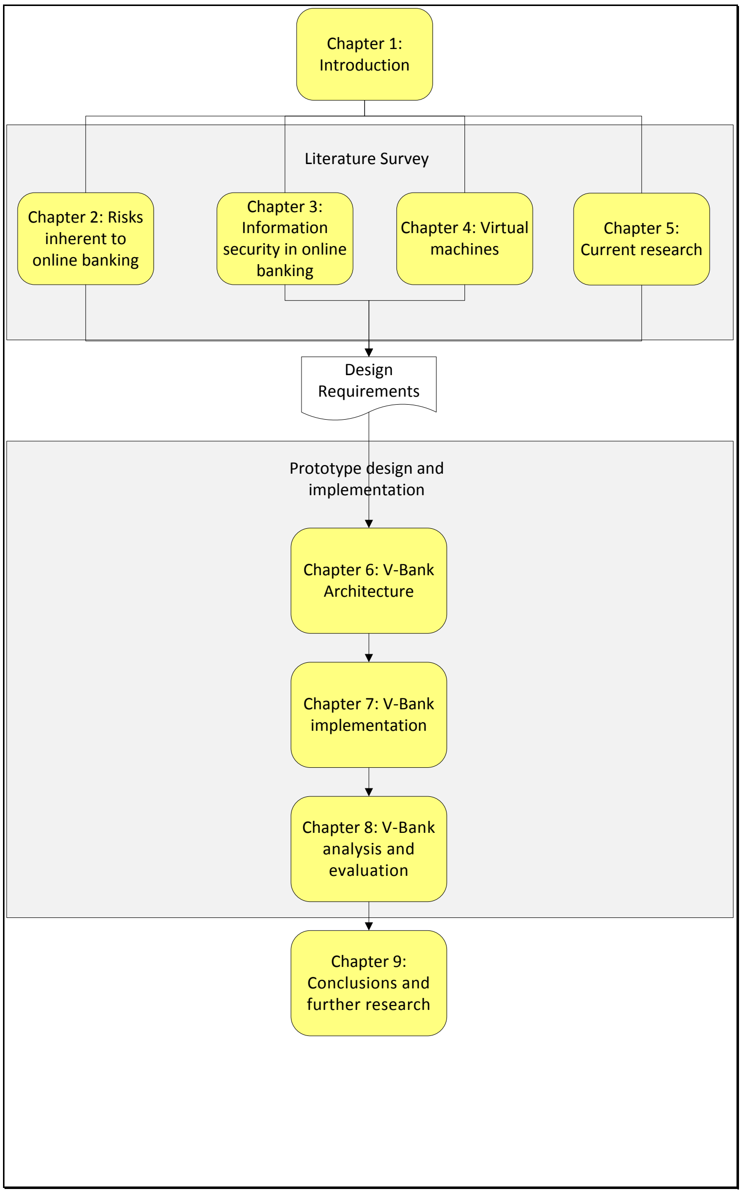
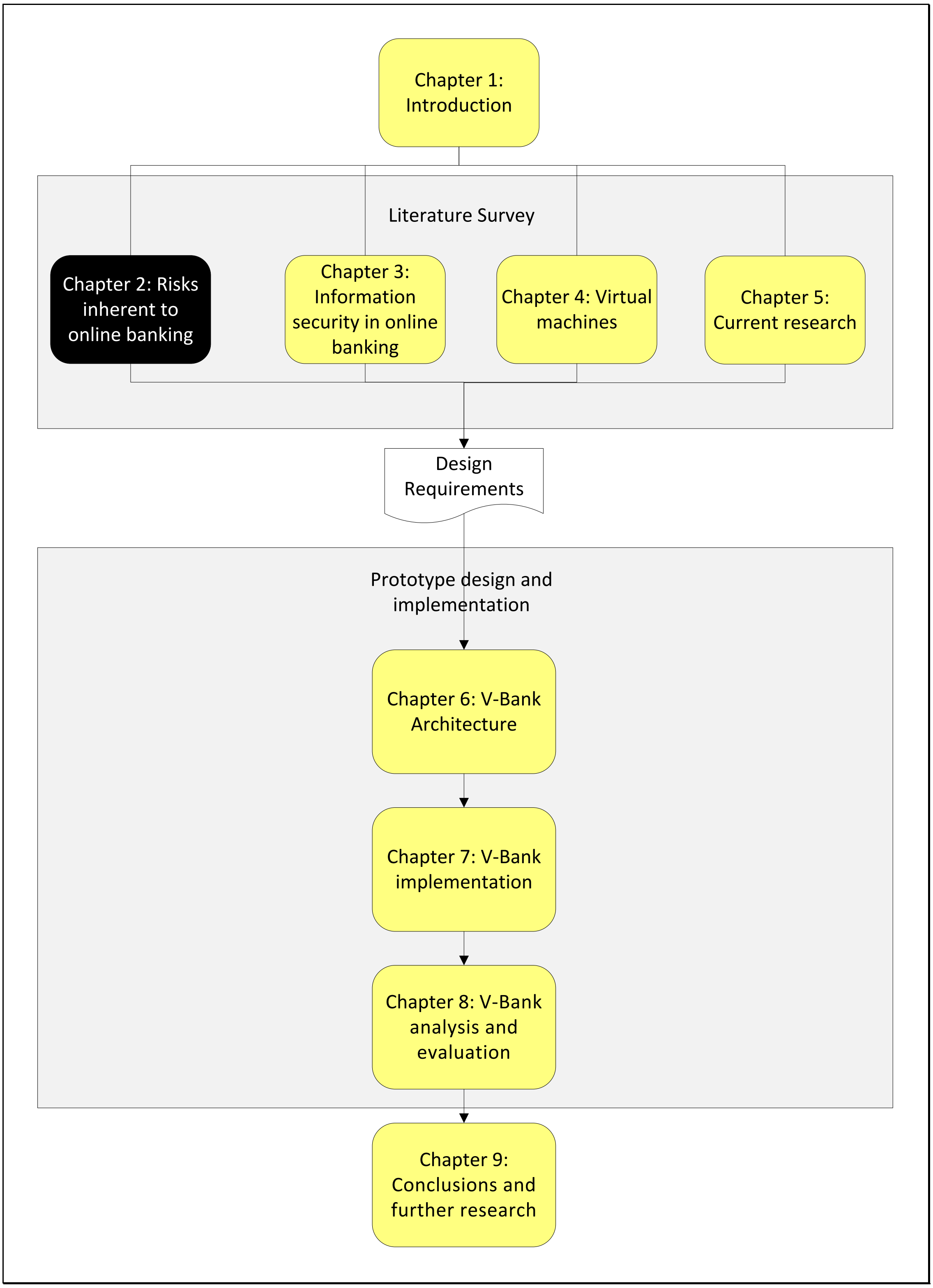

Chapter 1: Introduction
Literature Survey
Chapter 2: Risks inherent to online banking
Chapter 3: Information security in online banking
Chapter 4: Virtual machines
Chapter 5: Current research
Design Requirements
Prototype design and implementation
Chapter 6: V-Bank Architecture
Chapter 7: V-Bank implementation
Chapter 8: V-Bank analysis and evaluation
Chapter 9: Conclusions and further research

FIGURE 1:6 CHAPTER LAYOUT

Before the beginning of each chapter Figure 1:6 is repeated, with the current chapter highlighted, so that the reader gets a quick overview where in the structure of the dissertation the chapter exists.


Chapter 1: Introduction
Literature Survey
Chapter 2: Risks inherent to online banking
Chapter 3: Information security in online banking
Chapter 4: Virtual machines
Chapter 5: Current research
Design Requirements
Prototype design and implementation
Chapter 6: V-Bank Architecture
Chapter 7: V-Bank implementation
Chapter 8: V-Bank analysis and evaluation
Chapter 9: Conclusions and further research

Chapter 2: Risks Inherent To Online Banking

1 Introduction

In the previous chapter an overview of online banking was described. A number of areas vulnerable to attacks exist in the online banking process.

This chapter evaluates the information technology risks associated with online banking by approaching it from the view point of possible attack vectors. During the risk investigation, a list of requirements is identified for designing a secure online banking system.

The chapter is structured as follows. The chapter first defines what online banking fraud is. It then describes some of the terms and concepts that are used during the investigation of attack vectors. Once the terms and concepts have been defined, a number of attack vector categories are identified. Each attack vector category is described and a design requirement to minimise the effect of the attack vector is identified. After all the attack vectors have been described the list of attack vectors identified is consolidated to be used as design requirements for a secure online banking system.

2 Online Banking Fraud

The Merriam-Webster dictionary defines fraud as the intentional perversion of truth in order to induce someone to part with something of value (Merriam Webster, 2012).

Akopyan and Yelyakov (2009) define cybercrime as: "A process, in which a malefactor uses a computer to gain unauthorized access to other computers or control computer systems with the aim of gaining some benefit".

This research defines online banking fraud as a type of cybercrime, but where the malefactor specifically wants to gain unauthorised access to a banking customer's online banking account to either gain knowledge of the financial status, or to transfer funds from the bank account.

Online banking fraudsters use various methods to gain access to the customer's bank account. These methods are called attack vectors (TechTarget, 2012). Hansman and Hunt (2005) define an attack vector as: The method by which an attack reaches its target.

3 Terms and concepts

In Chapter 2:4 the terms for the different attack vectors are described. In this section of the research certain concepts allow the reader to better understand the impact and the specific attack vector. For these reasons this section evaluates the following concepts and terms, but in context of an online banking session:

- A computer with operating system
- Banking system environment
- Rootkits
- Keyloggers
- Command-and-control servers

- Money mules.
- Man-in-the-browser attacks
- Man-in-the-middle attacks
- Bot nets

3.1 A computer with operating system

John makes use of his computer at home to open up a web browser and then establishes a connection to the Bank's online banking website. When John uses his computer in this way, different components of the computer are involved.

Figure 2:1 shows the layers of the computer operating system. The computer consists of primarily two layers. These layers are known as user mode and kernel mode (Russinovich & Solomon, 2005).

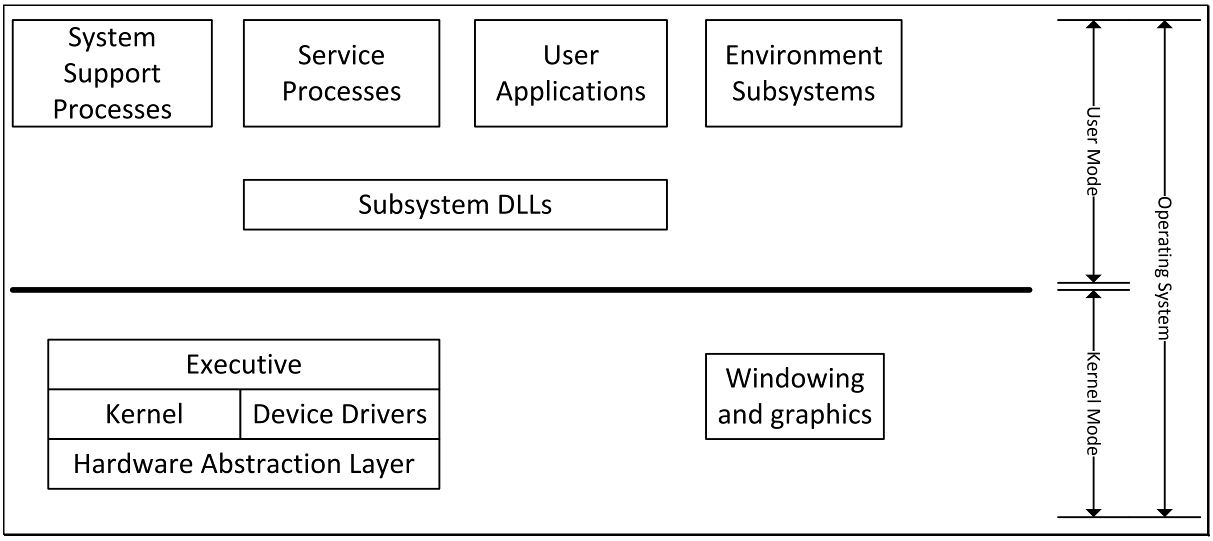


FIGURE 2:1 OPERATING SYSTEM COMPONENTS.

3.1.1 User mode and kernel mode

Processors like the Intel x86 or x64 family of processors has four privilege rings, numbered zero to three. The privilege levels of the rings provide protection for system code and data to be overwritten either inadvertently or maliciously (Russinovich & Solomon, 2005).

An operating system like Windows XP (Microsoft, 2013) uses ring zero and ring three. Code executed in ring zero is also called kernel mode code and code executed in ring 3 is called user mode code (Russinovich & Solomon, 2005).

3.1.2 Kernel mode

The kernel mode components include the following:

- The Windows executive contains the base operating services, which includes services like memory management, process and thread management, security and I/O.

- The kernel handles multi-processor synchronisation, interrupt and exception management and thread scheduling
- The device drivers translate user I/O calls into device specific I/O, as well as file system and network drivers.
- The hardware abstraction layer abstracts the executive, kernel and device drivers from platform-specific hardware.
- Windowing and graphics are responsible for the drawing and display of windows.

Devices connect to the computer through specific hardware interfaces. In the Kernel of the operating system, device drivers accepts and make connections to the operating system's Input \ Output System (Russinovich & Solomon, 2005).

From a security perspective any code that executes in kernel mode has full access to system space memory and can bypass Windows security to access objects (Russinovich & Solomon, 2005). This implies that malicious code that executes in kernel mode can gain access to system wide objects and memory. This type of malicious code is also known as rootkits. See Chapter 2:3.3 for a description of rootkits.

3.1.3 User mode

There are four types of user mode processes. They are (Russinovich & Solomon, 2005):

- **System support processes** are processes that perform Windows processes, but always start with the operating system. They include the logon process and the session manager.
- Windows **service processes** are logon independent processes that can run whether or not the user logs onto the computer or not. This includes processes such as the spooler service and the task scheduler service.
- **User applications** are applications that automatically start when a user logs on and manually executes the application. This includes applications such as word processors or web browsers.
- Windows XP (Microsoft, 2013) allows different system applications from executing in the operating system. This includes Windows 32bit applications, Windows 64bit applications, MS-DOS applications, Windows 3.1 16bit applications, OS/2 and Posix applications. Support for these applications is handled by the **environment subsystems**.

User mode applications each have their own virtual memory address space and execution environment. One application cannot access the memory and objects belonging to another process running in user mode (Russinovich & Solomon, 2005).

Applications that interact with the hardware, such as the file system, generate calls to subsystem DLLs, which in turn access kernel mode objects in the Windows executive that call device driver specific libraries, which in turn access the hardware via the hardware abstraction layer (Russinovich & Solomon, 2005).

Even though user applications are in general protected from each other, malicious applications can still influence other applications through the security level of the executing user. Sagiroglu

& Canbek (2009) demonstrate how various user mode components can be affected to allow a keylogger from reading keystrokes on the operating system.

3.1.4 Compromised operating systems

A compromised operating system may have various components embedded into the operating system that change the expected behaviour of the system.

Figure 2:2 refers to the architecture under discussion, but shows John's computer with two malware items. The two malware items on the computer is a malicious browser plugin and a keylogger. The malicious browser plugin is a plugin installed in the web browser, which executes in user mode. The keylogger was part of a device driver and executes in kernel mode.

Both items were installed on John's computer while John updated a printer driver on his computer using a driver John downloaded and installed from the Internet. The possible effects that these items may cause, are John's online banking password and pin can be intercepted by the keylogger. Another effect is that false information about his banking status can be displayed by the browser because of the browser plugin.

Even though the malicious browser plugin only has access to the memory and objects belonging to the browser process, it is still enough to display false information.

The keylogger that is installed in kernel space, as part of a device driver has full access to the system memory and can easily record, store and transmit keystrokes entered from the keyboard.

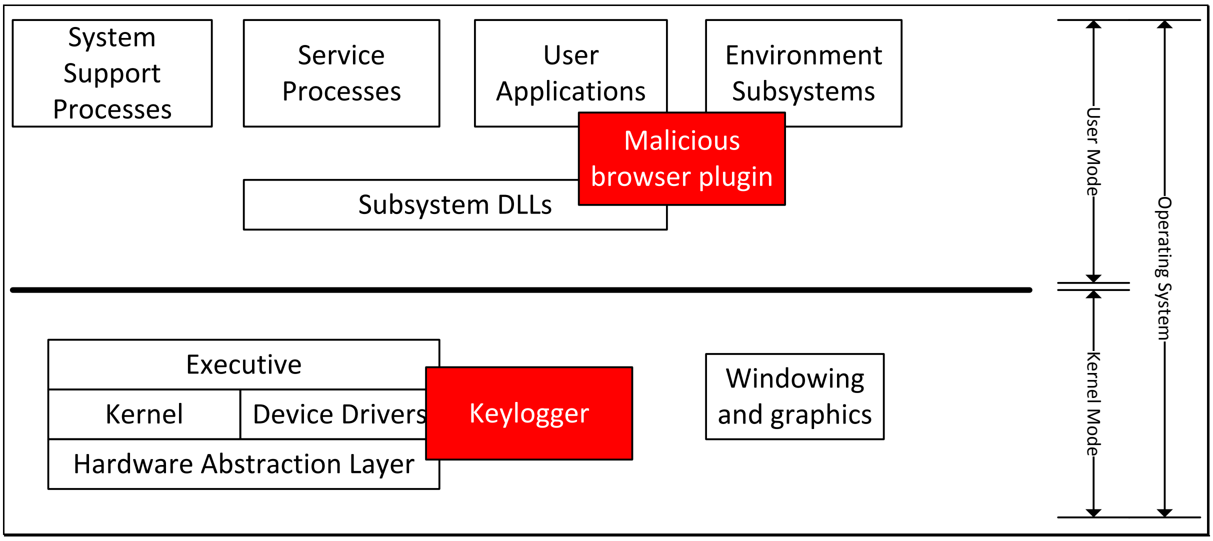


FIGURE 2:2 COMPROMISED OPERATING SYSTEM

3.2 Banking system environment

Four different functionalities can exist in a banking system. These functionalities can be implemented as four distinct services (Claessens, et al., 2002)

1. Interface Server
2. Authentication Server

3. Transaction Server
4. Mainframe

In some banking environments the Mainframe is replaced with an Execution Server.

Figure 2:3 depicts the various components described by Claessens, et al. (2002). It shows the Interface Server on the left that accepts connections from customers. To the right of the Interface Server is the Authentication Server whose function it is to authenticate users. The transaction server queues transactions to be executed on the mainframe of the execution server.

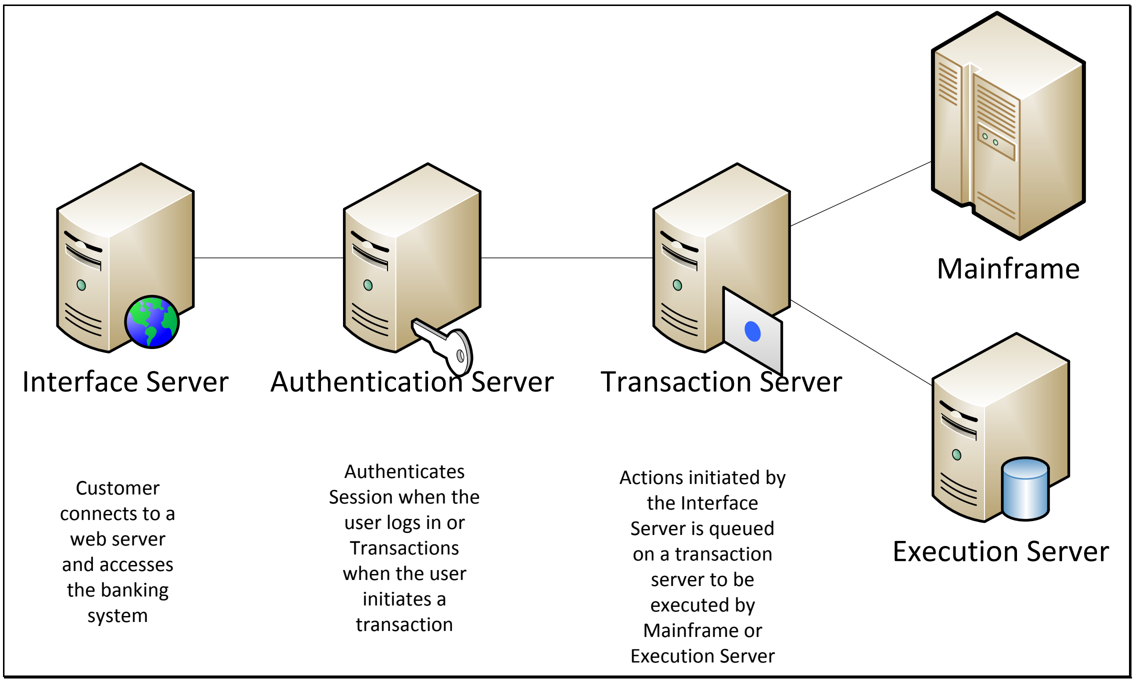

Mainframe
Interface Server
Authentication Server
Transaction Server
Execution Server
Customer connects to a web server and accesses the banking system
Authenticates Session when the user logs in or Transactions when the user initiates a transaction
Actions initiated by the Interface Server is queued on a transaction server to be executed by Mainframe or Execution Server

FIGURE 2:3 BANKING SYSTEM ENVIRONMENT

The environment has now been described. Before describing the attack vectors, the rest of the terms and concepts used in the attack vectors are now described.

3.3 Rootkits

A rootkit is a program that specialises in hiding itself or other malware from a system administrator. The program's main purpose is to perform certain functions that cannot easily be detected or undone (Microsoft, 2011).

An example of a known rootkit is Win32/Rustock (Microsoft, 2011). This rootkit is a family of Trojans that open backdoors on computers, but used to specifically distribute spam email (Microsoft, 2011).

If John's computer was infected with a rootkit that acted like the Zeus Trojan then it would be very difficult for an anti-virus program to detect the rootkit. In this example John may realise a problem with his bank account only after he received a statement from the bank, or he tried to pay another beneficiary and no funds were available.

3.4 Keyloggers

A keylogger is a piece of software, either in the form of a worm or Trojan, or it can be a piece of hardware. The job of a keylogger is to capture keystrokes from the computer keyboard and then depending on the implementation of the keylogger store or send this information on to a third party. (Sagiroglu & Canbek, 2009)

Keyloggers are either hardware or software based (Grebennikov, 2011).

Hardware keyloggers are small electronic devices designed to capture the data in between the keyboard device and the I/O port. Software keyloggers track systems that collect keystroke data within the target operating system, store them on disk or in remote locations, and send them to the attacker who installed the keylogger. (Sagiroglu & Canbek, 2009).

Figure 2:2 showed John's computer that had a keylogger installed. The keylogger was installed when John installed a printer driver that he downloaded from the Internet. The keylogger logs his keystrokes and stores his login details for his online banking profile.

3.5 Command-and-control servers

A command-and-control server operates in the manner of an army general. This general can receive information and provide instructions to its troops. The troops can be pieces of Malware like worms or Trojans. These troops are also called botnets (Feily, et al., 2009).

The command-and-control server can receive data from malware regarding banking pins and passwords. It can also issue information to the malware to inject information into a banking session, such as banking details about third-parties to whom payments can be made to (Sherstobitoff, 2011).

The command-and-control software can also be installed on any computer on the Internet that has vulnerabilities on it that enables the attacker to open a backdoor into the system. It can then utilise different techniques to control the different botnets, like internet relay chat (IRC), web-based, peer-to-peer (P2P) and domain naming server (DNS) services (Ianelli & Hackworth, 2005)

The captured banking information on a command-and-control server can also be used by people to try and gain access to customer banking details (Sherstobitoff, 2011).

Figure 2:4 depicts a scenario where online banking information gets transferred to and from a command and control server. In one instance the account number and pin information for the online banking profile is sent to the command and control server. In another instance the command and control server sends a command to an infected computer with instructions to pay another person instead of the intended recipient.

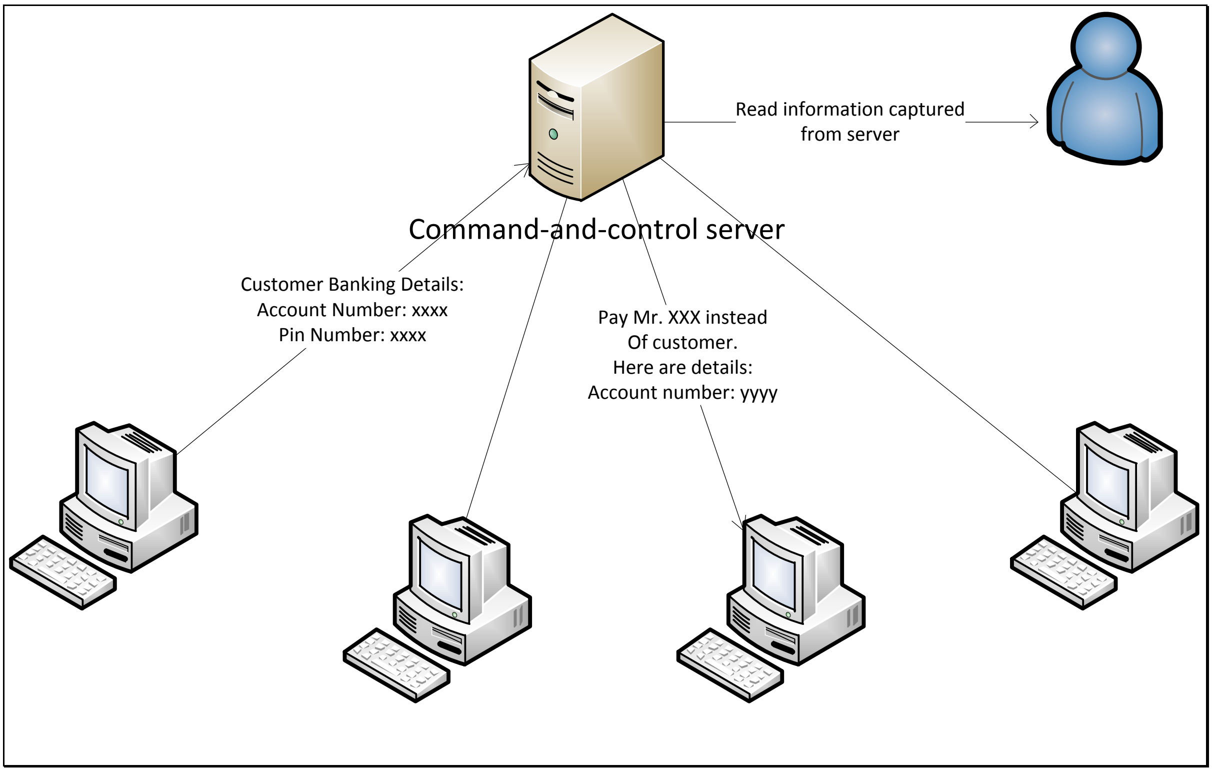


FIGURE 2:4 COMMAND AND CONTROL INFORMATION.

3.6 Money mules

To enable the successful transfer of money from a compromised bank account to the attacker's bank account, money mules are used. When a person's online bank account gets compromised the attacker transfers money from the compromised bank account into a mule's bank account. The mule in turn transfers the money into a bank account owned by the attacker (Florêncio & Herley, 2010).

Banks can trace the transfer of money between various accounts. If an attacker should transfer money directly from a compromised account into the attacker's own account, then the bank could trace the transaction and the authorities could apprehend the attacker. Attackers employ mules which act as middle men. Money gets transferred from a compromised account into a mule's account. The mule transfers the money to a foreign account, which is not traceable (Florêncio & Herley, 2010).

Florêncio & Herley (2010) argues that it is ironic that the person losing money most of the time, is not the bank or the compromised bank account, but the money mule itself.

3.7 Man-in-the-browser

Figure 2:2 depicts a computer with a browser plugin that affects the functionality of the Internet browser. The browser plugin runs as part of the Internet browser that executes in user space. The browser plugin is an example of one of the components that comprises the Zeus Trojan (Sherstobitoff, 2011).

Kernel based keyloggers typically gets installed as part of a Trojan or rootkit. Keyloggers have the ability to capture any input from keyboards (Sagiroglu & Canbek, 2009). This allows them to capture banking details as a user types it into the browser when he visits the banking website and then visit the web site independently and use the customers banking details to gain access to the customer's funds.

User space malware installed in the browser can create, what is called, a Man-In-The-Browser attack. The browser plugin effectively intercepts any code to the web browser and makes changes to it, making the customer believe he sees something, while actually doing something else (Gühring, 2006).

Example: The Zeus Trojan (which is a man-in-the-browser type of malware, also have a keylogger) may get installed onto the customer's computer, through a zero day, or existing vulnerability of the customer's browser. Once Zeus has been installed, it contacts a command and control server and retrieves a configuration file. The configuration file tells Zeus to stay dormant until the customer visits a specific banking web site. Some variants of the Zeus Trojan acts in the following way (Sherstobitoff, 2011):

1. The user logs into his bank
2. The login information is captured and sent to a command and control server
3. It then captures the account balance and sends it to the command and control server
4. It waits until the user goes to an electronic fund transfer page
5. It waits for the user to click on the "Submit" button to do an electronic fund transfer
6. The malware captures the intended amount
7. It freezes the session and does not allow the transaction to go to the banking server
8. Injects a fake page, making the user thinks the page is taking longer to load than normal
9. Makes a call to the Command and Control server and finds information about an appropriate mule.
10. It injects the mule's account detail and an appropriate amount into the transaction
11. It releases the session allowing the modified transaction to be sent to the online server
12. It injects a fake balance into the Account Balances and Account Summary pages

3.8 Man-in-the-middle attacks

A Man-in-the-middle attack is very similar to the Man-in-the-browser, except that the interception of the banking session does not take place on the customer's computer, but on an Internet facing server that makes the connection on behalf of the customer (Shirey, 2000).

The server that makes the connection on behalf of the customer's computer to the banking web site is also referred to as a proxy for the customer's computer.

The server that makes the connection is now "in the middle" of the customer and the bank. This server makes it seem to the customer as if the customer computer is connected to the actual banking web site. In reality the customer's computer talks to this server, thinking it is connected to the bank and the server then connects to the bank, using information gathered from the connecting customer computer.

This type of attack is dependent on the fact that the server disguises itself and makes it look and seem like the banking web site. The bank's web site on the other hand assumes the connections from the server are actually from the legitimate banking customer. This assumption from the banking web site is made because the server can get authentication details from the customer's computer, because the customer's computer is giving it to the server.

3.9 Bot nets

Attacks using unknowingly compromised computers, employed for malicious internet attacks, using an extensive array of these compromised systems are known as a bot net, or robotic network. (Carrow, 2007)

Figure 2:5 depicts a network of infected systems that work together as an array to launch an attack on a computer.

These attacks make use of unknowingly compromised users' computers or corporate resources. These infected systems act as conduits for malevolent attacks redirected against individual users, websites or network domains. (Carrow, 2007)

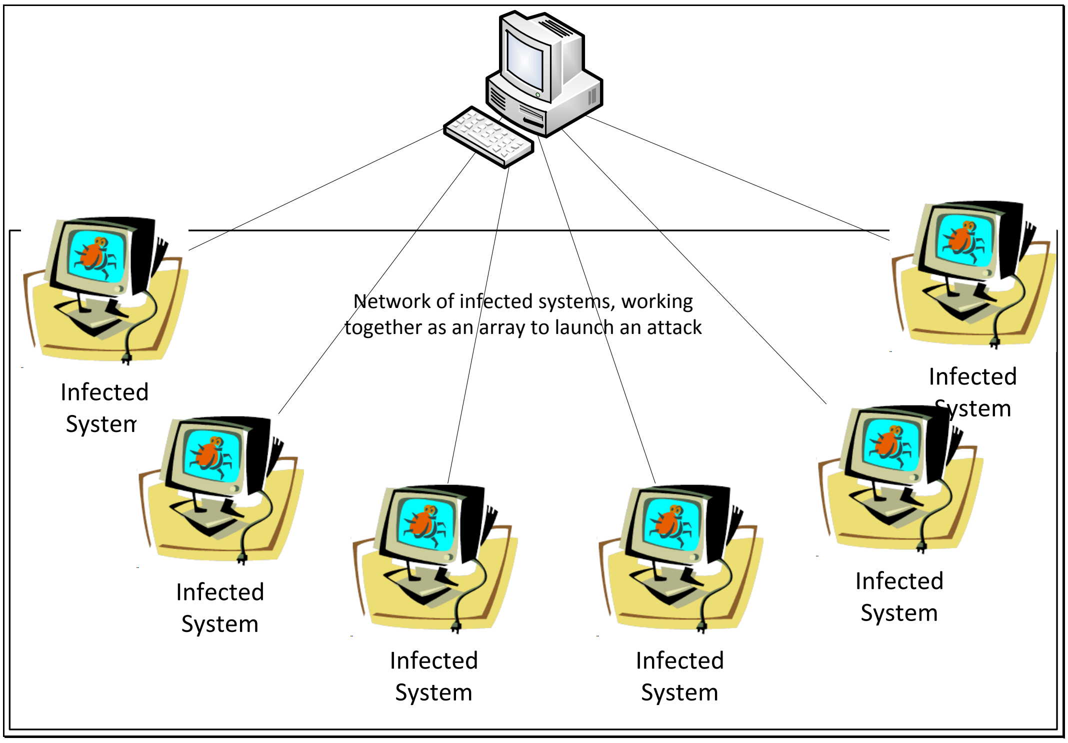


FIGURE 2:5 BOT NET ATTACK

The main purpose of a botnet is to steal information about cards or other bank details to gain access to money in bank accounts. (Akopyan & Yelyakov, 2007)

4 Attack Vectors

To understand possible attacks that can take place during online banking, is it necessary to understand the online banking system and the various components involved. This study evaluates online banking from a perspective of a home computer belonging to a customer. The study excludes the attack vectors faced by customers while doing online banking from mobile devices.

This chapter assigns a section for each attack vector. While investigating the specific attack vector, the attack target is described. The attack targets are limited to the areas derived from the case study described in chapter 1 and defined by Hutchinson and Warren (2003). They are the following and are displayed in Figure 2:6 (Hutchinson & Warren, 2003).

1. A customer operating environment on the left
2. The network and Internet environment at the bottom
3. The banking system environment on the right

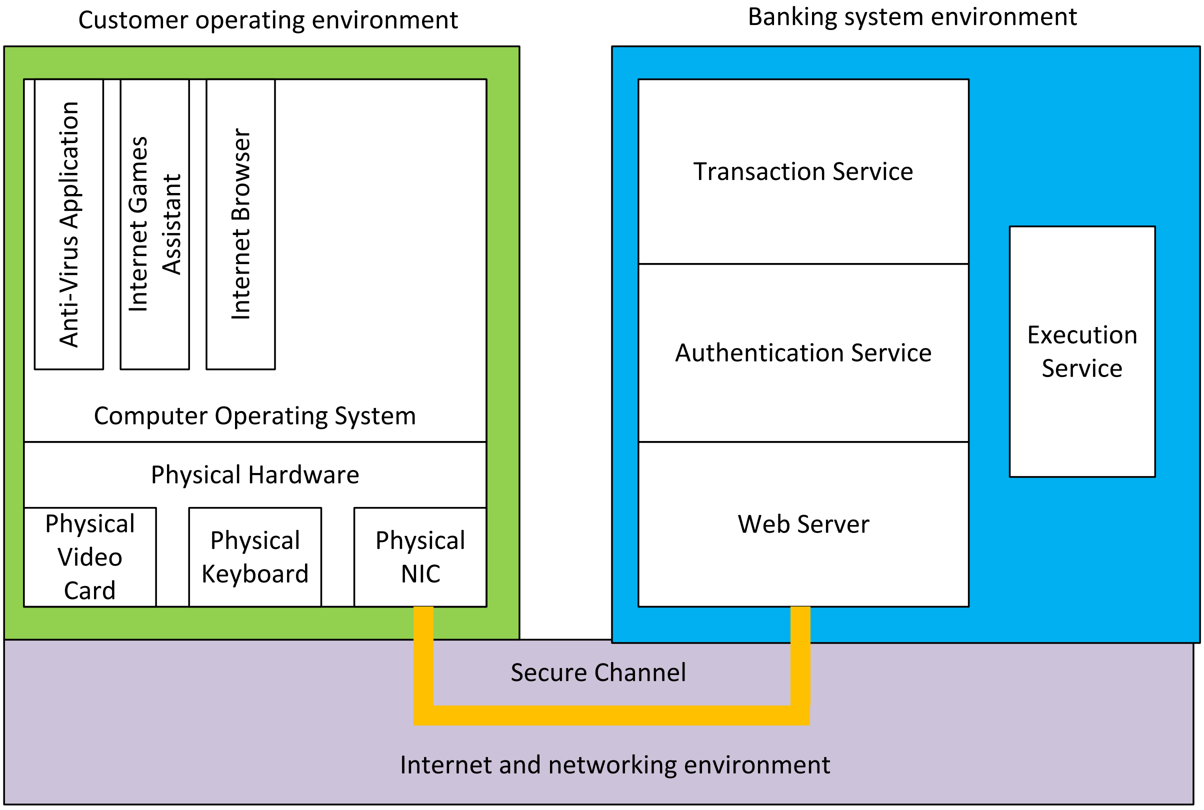


FIGURE 2:6 AREAS IN AN ONLINE BANKING SESSION.

Each of these areas is called an "environment". They consist of various components that make it possible for the online banking session to be established. (Russinovich & Solomon, 2005)

Hansman and Hunt (2005) define the following categories of attack vectors:

1. Viruses.
2. Worms.
3. Trojans.
4. Buffer overflows.
5. Denial of service attacks
6. Network attacks.
7. Physical attacks.

8. Password attacks.
9. Information gathering attacks.

4.1 Viruses

A virus is type of malware that reproduces and attaches itself to a file. Viruses require human interaction to spread. Viruses are not inherently programmed to spread to other hosts, instead they only spread to another computer through the interaction of a human. When a person copies or emails an infected file to another computer, and the user of the other computer opens or executes the file, only then does the virus infect that computer (Shirey, 2000).

Viruses have the ability to damage computer files, making the computer inoperable.

Viruses can attack any computer system. This means that a virus can attack both the customer operating environment and the banking system environment.

Six months earlier John's computer became infected with Win32.Neshta.a virus (ESET, 2006). The virus caused his web browser to become infected. This in turn caused his computer to start malfunctioning while trying to connect to the Internet and his online banking system. John had to download and install a new Internet browser after he cleaned his computer using his anti-virus program.

The infection of the computer virus caused John's computer to malfunction and disallow him access to online baking. If the virus infected the banking system environment then it may have caused the whole banking system to stop functioning. This would not only have affected John, but all online banking customers.

A virus is a type of malware, and the online banking system should minimise the chances of malware getting installed into the operating system.

<u>Design Requirements</u>

1. An online banking system should minimise the chances of malware getting installed into the operating system at the client and allowing fraudsters to change the online banking session

It is critical that online banking systems, both the client and the server be protected from viruses.

<u>Design Requirements</u>

2. The online banking system should be protected from viruses, by making the changing of files on the computer impossible.

4.2 Worms

Worms are programs that replicate themselves, but without using infected files (Hansman & Hunt, 2005). It is common for worms to propagate through network services on computers or through email (Hansman & Hunt, 2005).

Worms may utilise email, existing network connections, chat applications and even Internet servers to spread. It makes use of these mechanisms through some kind of vulnerability or implicit design flaw.

A worm is also a standalone program and does not require another file to exist (Symantec, 2009).

Apart from the worm's ability to spread itself and run as a standalone program, worms usually cause some type of damage by executing its "payload". The payload may be to deface a web site, open a backdoor into the computer or steal personal information from the infected host. The Win32/Conficker.e.dll (Microsoft, 2011) worm terminates many anti-virus programs, it blocks access to some web sites that helps to clean your computer from viruses and it distributes and receives commands through a built in peer-to-peer (P2P) network (Microsoft, 2011).

If John's computer became infected with a worm, it may be possible for a hacker to use the worm to steal his online banking login details.

Worms can target both the customer operating environment and the banking system environment. Since worms can effectively run any payload on an online banking system, the online banking system should be protected from worms. Both the customer operating environment and the banking system environment should be protected from worms.

<u>Design Requirements</u>

3. An online banking system should be protected from worms, minimising the chances that a worm can cause backdoors into the client computer to steal vital banking details.

4.3 Trojans

A Trojan is a piece of malware, but it is disguised as something else. It may be defined as a computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes the program. (Shirey, 2000)

A Trojan may for example be seem to be a friendly game, or screensaver, freely available on the Internet, that when downloaded, starts monitoring keystrokes. (Danchev, 2005)

When John's children installed a gaming helper program, the helper program may have been a Trojan. If the Trojan was the Zeus Trojan (Symantec, 2010), then it would have been possible for attackers to gain access to John's bank account and transfer money from his account to an attacker's account.

The Zeus Trojan (which is a man-in-the-browser type of malware, also have a keylogger) may get installed onto the customer's computer, through a zero day, or existing vulnerability of the customer's browser. Once Zeus has been installed, it contacts a command and control server and retrieves a configuration file. The configuration file tells Zeus to stay dormant until the customer visits a specific banking web site. Some variants of the Zeus Trojan acts in the following way (Sherstobitoff, 2011):

1. The user logs into his bank
2. The login information is captured and sent to a command and control server
3. It then captures the account balance and sends it to the command and control server
4. It waits until the users goes to an electronic fund transfer page
5. It waits for the user to click on the "Submit" button to do an electronic fund transfer
6. The malware captures the intended amount
7. It freezes the session and does not allow the transaction to go to the banking server
8. Injects a fake page, making the user thinks the page is taking longer to load than normal
9. Makes a call to the Command and Control server and finds information about an appropriate mule.
10. It injects the mule's account detail and an appropriate amount into the transaction
11. It releases the session allowing the modified transaction to be sent to the online server
12. It injects a fake balance into the Account Balances and Account Summary pages

In the example where John conducts online banking, the Zeus Trojan allows an attacker to steal money from his bank account into another person's account.

Trojans can cause harm in both client operating environment and the banking system environment. Users of a system can download and install Trojans on their home computers without realising the risk or effect the Trojan may have on their online banking sessions. An online banking system should be protected from Trojans.

Design Requirements

4. An online banking system should be protected from Trojans. Users should not be allowed to download and install anything beyond what is already in the system.

4.4 Buffer overflows

A buffer overflow attack gains control, or crashes another process, by overflowing the buffer of the other process (Hansman & Hunt, 2005).

The SQL Slammer Worm, or W32.SQLExp.Worm (Symantec Corporation, 2013), is a famous worm that replicates itself by using a buffer overflow vulnerability that existed in Microsoft SQL

Server 2000. The worm spreads itself by connecting to UDP port 1434. It overflows the buffer of the SQL Server Resolution Service, effectively allowing the worm's code to execute in the security context of the SQL Server service. It then randomly generated IP addresses and tried to spread itself to these IP addresses on port 1434 (Symantec Corporation, 2013).

The SQL Slammer Worm caused widespread network performance degradation. It affected ATM and voice over IP networks (Symantec, 2003).

Buffer overflow attacks are one of the oldest and most pervasive attack techniques, exploiting vulnerabilities that exist in systems (Piromsopa & Enbody, 2006). After the SQL Slammer Worm was detected, Microsoft had to write software patches to fix the vulnerability that existed in SQL Server 2000.

Buffer overflow attacks can target a system using the internet and networking environment, but it can also attack directly using a cleverly placed file on the customer operating environment (Piromsopa & Enbody, 2006).

In an online banking scenario the client side operating environment needs to be protected from the internet and networking environment. Network connections to the client side operating environment should only occur to and from the banking system environment. Users should also be prevented to accidentally change the files in the customer operating environment that may cause a buffer overflow attack.

Design Requirements

5. An online banking system should be protected from buffer overflow attacks, by controlling access to clients from the network and not allowing clients to change the files in the customer operating environment.

4.5 Denial of service attacks

A denial of service (DoS) attack cripples a network service by flooding the service with many requests. The service is usually Internet based, providing some type of service to Internet clients. The flooding can be from either a single source or from multiple sources. In the case where the flooding originates from multiple sources the denial of service attack is known as a distributed denial of service (DDoS) (Hussain, et al., 2003)

Denial of service attacks originate on the network and can affect a destination network, or service on a network. In an online banking session the areas that may be affected by a DoS attack are the internet and networking environment and the banking system environment.

In a banking environment a denial of service attack can be targeted at the DNS system (Domain Naming System), which may cause customer computers from not resolving the IP address of the bank web site, or the banking web server itself. On August 16 2012 AT&T suffered an outage from a distributed denial of service (DDoS) attack against their DNS servers. This caused some of their customers to experience intermittent disruptions on their services (Williams, 2012)

An attack on the banking web server can be achieved by keeping a web site so busy servicing authentication requests, that the system does not get enough time servicing legitimate banking customers. Certain system vulnerabilities may also allow certain buffers on the server to overflow, which can sometimes cause the system to crash, effectively stopping the whole system. On April 20 2012 the official Formula 1 web site was taken down through a DDoS attack from the hacker collective called "Anonymous" in response to violent protests in Bahrain (Wert, 2012)

Design Requirements

6. An online banking system should discard connections from systems trying a Denial of Service attack.

4.6 Network attacks.

Hansman and Hunt (2005) defines a network attack as an attack that focusses on attacking a network of the users on the network. This is accomplished through the manipulation of network protocols. According to Hansman and Hunt's taxonomy, even if the attack is a DDoS attack, but it floods the network, then the attack should be classified as a network attack.

Hansman and Hunt also classify the following examples as network attacks. They include (Hansman & Hunt, 2005):

- Spoofing
- Session hijacking
- Web application attacks, like cross site scripting
- Database attacks
- Hidden field manipulation
- DNS Contamination

Network attacks use the network to attack the client operating environment. The client operating environment should not allow connections to the client's computer from fraudsters.

Design Requirements

7. An online banking system should not allow connections to the client's computer from fraudsters.

Each of these network attacks is now described.

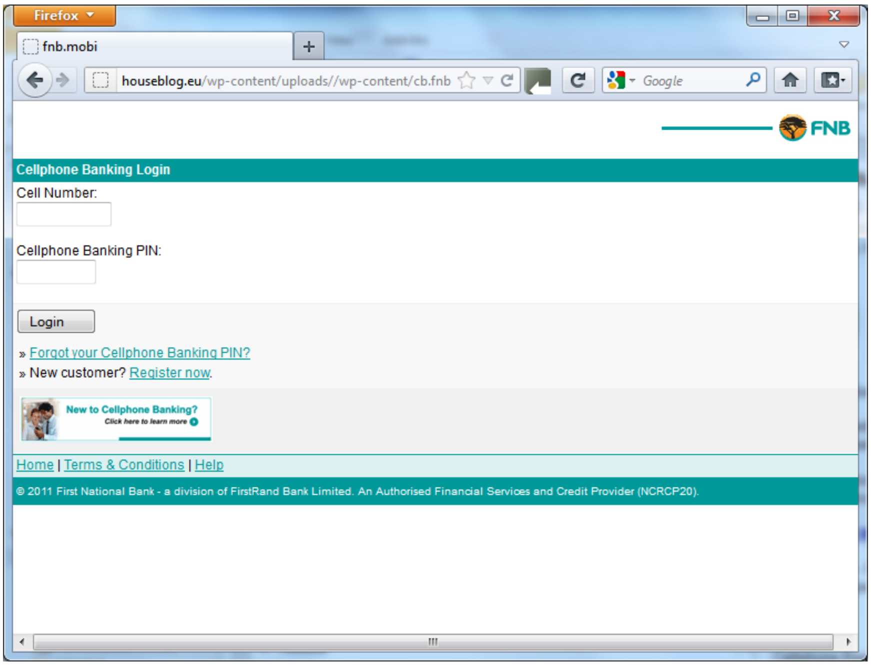
4.6.1 Spoofing

Spoofing is defined as the act of an unauthorised entity gaining access to a system by posing as an authorised entity (Shirey, 2000). In an online banking environment spoofed web sites pose a real threat to online banking customers.

Spoofed web sites are web sites that look and feel the same as the bank's actual web site. It usually exists inside another website, which in itself may have been compromised.

The reason for a spoofed web site is to build web pages, that look and feel exactly like the real web site. Attackers can use a spoofed web site to capture information from users, who thinks, they are connected to the real web site. If the spoofed web site is a bank's web site then the spoofed web site can steal a user's online banking details (Standard Bank, 2005).

Figure 2:7 is an example of a malicious spoofed web site that was loaded while clicking on a phishing e-mail. The site may seem legitimate, but the address does not belong to the real bank.



houseblog.eu/wp-content/uploads//wp-content/cb.fnb

Cellphone Banking Login

Cell Number:

Cellphone Banking PIN:

Login

» Forgot your Cellphone Banking PIN?

» New customer? Register now.

New to Cellphone Banking? Click here to learn more

Home | Terms & Conditions | Help

© 2011 First National Bank - a division of FirstRand Bank Limited. An Authorised Financial Services and Credit Provider (NCRCP20).

FIGURE 2:7 EXAMPLE OF SPOOFED INTERNET BANKING WEB SITE

Design Requirements

8. An online banking environment should minimise the chances of the Internet browser to go to a spoofed malicious banking web site.

4.6.2 Session hijacking

Session hijacking is a term that describes an attack where an attacker takes control away between two parties that exchange information over a network. In an online banking environment it is possible for an attacker to hijack the SSL session established by a client and server, allowing the attacker access to the online banking session for the client (Cheng, et al., 2010).

The man-in-the-middle attack and man-in-the-browser attack are examples where the session between a banking customer and the online banking system gets hijacked and manipulated. Man-in-the-middle and man-in-the-browser attacks were described in Chapter 2:3.7 and Chapter 2:3.7.

An online banking system should be protected from man-in-the-middle and man-in-the-browser attacks. The first requirement that minimises this risk is for the online banking system to not allow communication to any system outside the online banking system.

Design Requirements

9. An online banking system should not allow communication to any system except the online banking system.

The second thing that can be done to minimise the chances of a man-in-the-browser attack is to disallow the internet browser used in an online banking session from changing.

Design Requirements

10. It should not be possible to modify the Internet browser used by an online banking system.

4.6.3 Cross site scripting

Cross Site Request Forgery, or “Session Riding”, allows an attacker to use an existing secure session established from the customer’s web browser, to a secure site, but from another web site, which may be opened from within the same web browser, but in another tab (Jovanovic, et al., 2006).

This type of attack happens without the customer knowing about it. The reason why this type of attack is possible is because a session with the web server is stateless. This means that the HTTP protocol employed to establish the session is unable to distinguish a series of requests to a particular customer. The web server employs techniques such as cookies or URL re-writing to distinguish specific sessions. (Jovanovic, et al., 2006)

These techniques unfortunately enjoy little protection and can easily be hijacked by a script or code executed from a different web site inside the same web browser.

Example (Jovanovic, et al., 2006):

1.) A user logs into an Internet Banking web site, using secure credentials
2.) The web server stores the successful authentication in a session variable that identifies the user as having correctly authenticated.
3.) Every time the web site now gets a request from the user’s web browser, it looks at the session ID and determines that the user already authenticated
4.) The user temporarily gets distracted and does not log off from the Internet Banking web site, meaning the session is still authenticated according to the server.
5.) The user opens another tab in the web browser and goes to a blogging web site. On the blogging web site the user notices an interesting link and clicks on it.
6.) The link in HTML terms looks something like this:

```
<          href=’www.bigbank.com/transfer.php?amount=10000?to=7777’>Click
here</a> for something really interesting
```

7.) The link effectively executes a command to transfer money into a bank account on the Internet Banking web site.
8.) The Internet Banking web site accepts the request coming from the customer’s web browser, because the user is already logged in the session between the browser and the web server is already authenticated.

The above example is very simplistic and alert people may notice the URL as it is written into the address bar of the web browser. The same, however, can be accomplished by hiding the action inside a JavaScript application that may automatically start when the user navigates to the page.

Design Requirements

11. Online banking systems should not allow the Internet Browser from opening multiple pages and the transaction component in an online banking system should re-authenticate the user, before committing important transactions.

4.6.4 *Database attacks*

Information that is entered or displayed on a banking system is entered and retrieved from a database. The entering and retrieval of information is handled using an industry standard query language called “structured query language” (SQL).

If a web site does not filter the input provided by a customer, then it is possible for the input to transform the intended SQL command to retrieve or enter information into something else.

An example of what SQL injection may look like is as follows (Friedl, 2007):

When a user log in, there may be a field where the user should enter his username and another field where the user must enter his password. From these two fields the system can then construct a query that searches the database for a record with the corresponding username and password. The SQL code for such a query may look as follow.

```
Select * from users where username = ‘John’ and password = ‘p@ssw0rd’
```

If any rows were found, then the system may know that the username and password entered were correct. However, if the system does not filter the input from the user, the user can enter the following into the username field, and anything else into the password field.

```
“John--“
```

The SQL query using the above username now looks like:

```
Select * from users where username = ‘John’--and password =
‘Gobbledygook’
```

The single quotation mark in SQL, as used around the words John and Gobbledygook in the above example, is the character that includes text type fields and the “--“ is the SQL syntax for a comment in the code. Anything following the “--“ is interpreted as a comment and is ignored. The resultant SQL command effectively searches for any record, where the username is “John” and ignores the password field. An attacker can now gain access to John’s information without knowing the password for John.

The above example is a very simple form of SQL injection, more sophisticated SQL injection commands can even allow an attacker to execute system-level commands on the database server, which may give the attacker administrative permissions or even return all usernames and passwords stored on the system (Litchfield, et al., 2005).

Design Requirements

12. An online banking system should evaluate any input and allow only approved syntax in input fields.

4.6.5 Hidden field manipulation

Web developers sometimes require content from a different web site to be displayed in their own web site. This content may be advertisements, or news items. The developer uses a component called an Inline Frame (IFrame) to display the content from the remote web server on the current web site.

In the case where the IFrame on a web-site references an advertisement on a different web site, the attacker will target the web site with the advertisement on. If this web site is compromised the attacker embeds script code into the advertisement then when the user clicks on the advertisement the code will execute on the client computer, which may download and install malware.

Figure 2:8 depicts a banking web site with an advertisement. The advertisement references another web site, which is compromised. The script downloads and installs malware through a possible vulnerability in the web browser.

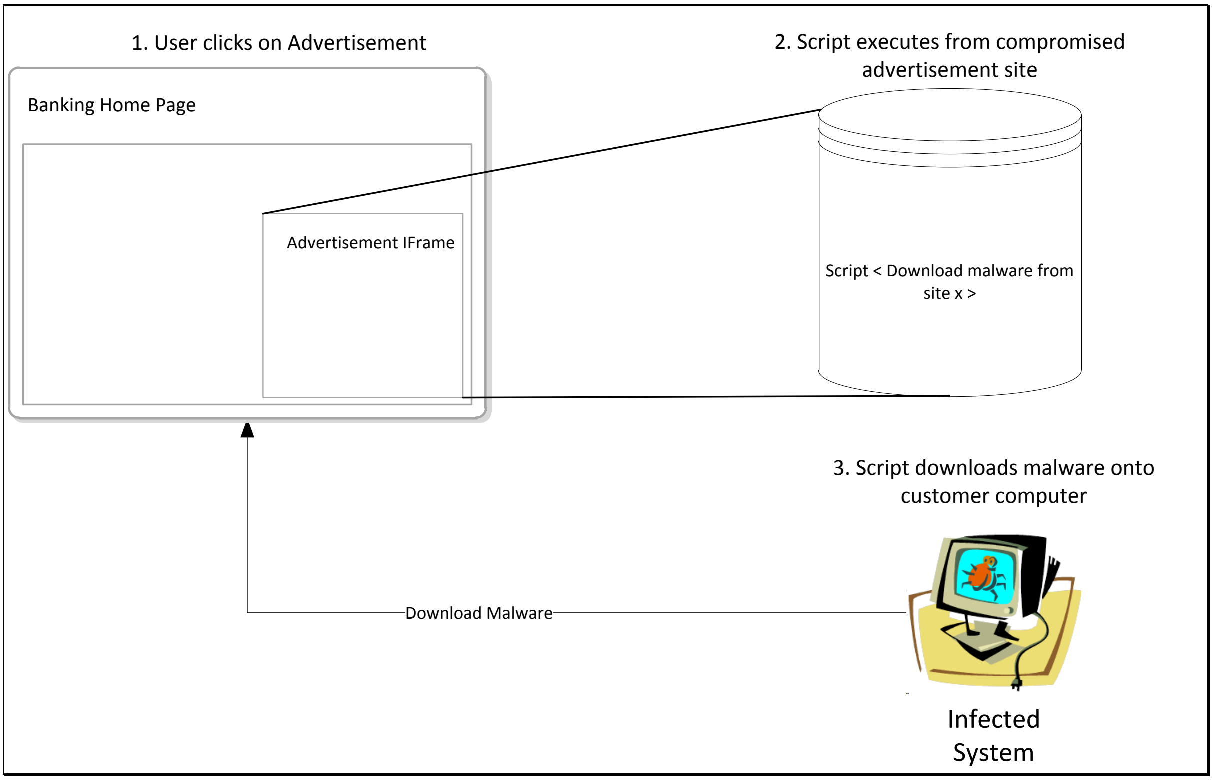

1. User clicks on Advertisement
2. Script executes from compromised advertisement site
Banking Home Page
Advertisement IFrame
Script < Download malware from site x >
3. Script downloads malware onto customer computer
Download Malware
Infected System

FIGURE 2:8 IFRAME VULNERABILITY EXAMPLE

Design Requirements

13. The online banking system should not contain any IFrames, nor should it reference code from any site except its own.

4.6.6 DNS Contamination

A man-in-the-middle attack is usually initiated via some type of phishing attack. This means the user may receive an e-mail that seems as if it is from his\her bank, with a hyperlink to the "supposed" banking web site. This hyperlink however is actually a link to the server in the middle of the man-in-the-middle attack.

There is however also another way for the user's computer to go to a spoofed web site, even though the user typed in the correct banking URL. This is through DNS Contamination (ACM US Public Policy Council, 2012).

DNS (Domain Naming Services) is the service on the Internet that translates names to IP addresses. Communication on the Internet is done by connections made to and from various IP addresses. IP addresses are difficult to remember and may change. DNS can take a friendly name, such as www.bank.com and translate that to the actual IP address of the banking servers (Shirey, 2000).

Computers and other DNS servers cache frequently used IP address mappings to ensure fast name resolution.

DNS contamination is the ability of a malicious person or program to change the cached information of these names to IP address mappings. This will redirect a web browser to a spoofed web site, even if the customer typed in the correct banking address (ACM US Public Policy Council, 2012).

In an online banking environment the client operating environment should only connect to the online banking system. Name resolution should not affect the connection of the online banking client.

<u>Design Requirements</u>

14. Name resolution should not affect the connection inside an online banking system

An online banking client should only be able to connect to the actual online banking server. The online banking server should verify connections to ensure it accepts only connections from approved clients.

<u>Design Requirements</u>

15. An online banking system should only allow connections to and from approved clients and servers.

4.7 Physical attacks.

A physical attack is where an attacker damages physical components of a computer or network (Hansman & Hunt, 2005). In an online banking environment physical attacks may cause a denial of service to customers, if the attackers can physically gain access to the customer's computing environment or the banking system environment. This study does not take into account physical attacks as an attack vector because of the lack of reported cases where it affects online banking.

4.8 Password attacks.

Password attacks are aimed at gaining knowledge of a user's password for a system (Hansman & Hunt, 2005).

A keylogger is a method used by attackers to steal passwords from users. Keyloggers were described in Chapter 2:3.4. Keyloggers affect the client operating environment in an online banking system. Attackers can gain access to a customer's online banking environment, if the environment is only dependent on passwords. For this reason online banking systems should be protected from keyloggers.

Design Requirements

16. A keylogger should not be allowed inside an online banking system or should minimize the effect it may have if present.

4.9 Information gathering attacks.

Information gathering attacks do not do any direct harm to computer systems or equipment. Attackers gather information regarding type of equipment, user names, access rights or personal information to be used in an attack at a later stage (Hansman & Hunt, 2005).

Examples of an information gathering attack are:

- Social engineering attack
- Phishing

These attacks are now described.

4.9.1 Social engineering attacks

Social engineering attacks make use of human interaction to gain information about the customer's online banking details. The user may either disclose this information to the attacker using a telephone conversation, whereby the attacker disguises himself as a bank employee or agent who have to confirm banking details, or makes use of phishing attacks. (McDowell, 2004)

Example of how a social engineering attack can be staged:

John	Imposter
	Hello there, I work for the bank. We are rolling out a new security system. Can you please confirm for me a few of your particulars?
	Can you please confirm for me your account number, ID number, Address and Home telephone number?
Yes sure. I want to help. My account number, ID number and other particulars are …	

Imposter	Bank Customer Service
Hello there, my name is John. I forgot my online banking password. Can you please reset it for me?	Yes sure. For security purposes, can you please confirm the following for me? What are your ID number, and your physical address?
Thank you. My ID Number is …	Thank you sir. Your new password is: xxxx. You will be asked to reset it the first time you log in.
Thank you very much. I will now be able to access my online banking again!	

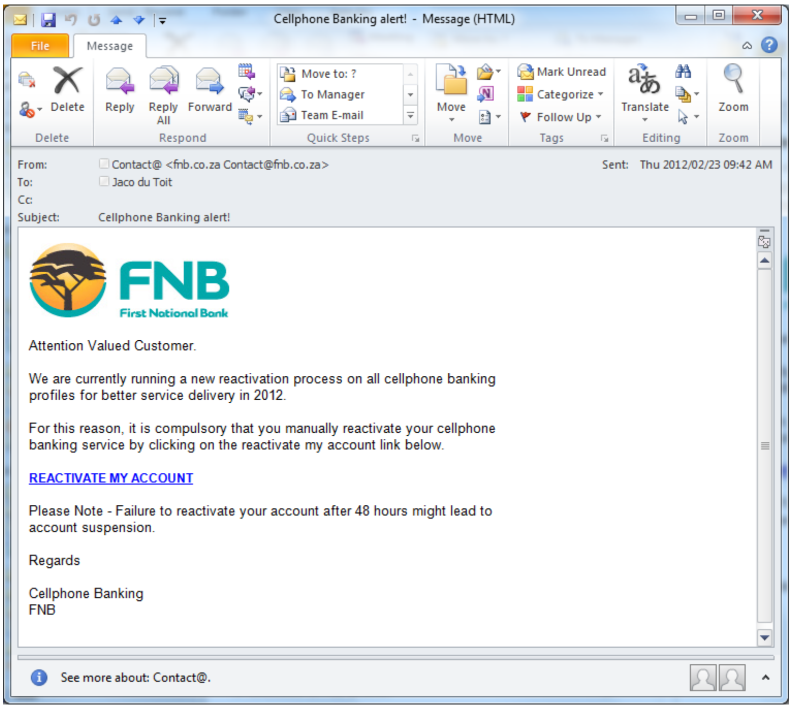
With the rise of the social networking, like Facebook and Twitter, social engineering has taken on new dimensions for compromising secure information. According to the Symantec Internet Security Threat Report for 2011 one of the five identified recurring themes in Internet Security was the use of social networks to launch a targeted social engineering attack (Symantic Enterprise Security, 2011).

The second area is a specific type of social engineering attack and is specifically discussed because of its effectiveness.

4.9.2 Phishing

Phishing attacks are a type of social engineering attack. It takes the form of an email that is received by the customer and fools the customer into connecting to a fraudulent banking web site, where the customer discloses his banking details on the web site (McDowell, 2004).

Figure 2:9 shows a screenshot of an actual phishing attack that was detected on the author's computer.



From: Contact@ <fnb.co.za Contact@fnb.co.za>
Sent: Thu 2012/02/23 09:42 AM
To: Jaco du Toit
Cc:
Subject: Cellphone Banking alert!

FNB
First National Bank

Attention Valued Customer.

We are currently running a new reactivation process on all cellphone banking profiles for better service delivery in 2012.

For this reason, it is compulsory that you manually reactivate your cellphone banking service by clicking on the reactivate my account link below.

REACTIVATE MY ACCOUNT

Please Note - Failure to reactivate your account after 48 hours might lead to account suspension.

Regards

Cellphone Banking
FNB

FIGURE 2:9 AN EXAMPLE OF A PHISHING ATTACK EMAIL

Design Requirements

17. An online banking environment should minimise the chances of an attacker using stolen information to gain access to a customer's online banking profile.

5 List of design requirements

During the risk investigation in this chapter a number of design requirements were identified that must be taken into account while designing a secure online banking system. The list of the initial 17 requirements is listed in Table 1.

Many of the requirements can be grouped together into a more generic requirement. An evaluation of the requirements is done to see which requirements can be grouped together.

Design requirements one to four concern themselves with minimising the effect of some type of malware, which includes viruses, Trojans and worms. Requirements one to four has a common recurring theme that states that the client side operating environment should not be allowed to change. Requirements ten and sixteen also fall in this category.

Number	Requirement
1	An online banking system should minimise the chances of malware getting installed into the operating system at the client and allowing fraudsters to change the online banking session
2	The online banking system should be protected from viruses, by making the changing of files on the computer impossible
3	An online banking system should be protected from worms, minimising the chances that a worm can cause backdoors into the client computer to steal vital banking details.
4	An online banking system should be protected from Trojans. Users should not be allowed to download and install anything beyond what is already in the system.
5	An online banking system should be protected from buffer overflow attacks, by controlling access to client from the network and allowing clients to change the files in the customer operating environment.
6	An online banking system should discard connections from systems trying a Denial of Service attack.
7	An online banking system should not allow connections to the client's computer from fraudsters.
8	An online banking environment should minimise the chances of the Internet browser to go to a spoofed malicious banking web site.
9	An online banking system should not allow communication to any system except the online banking system.
10	It should not be possible to modify the Internet browser used by an online banking system.
11	Online banking systems should not allow the Internet Browser from opening multiple pages and the transaction component in an online banking system should re-authenticate the user, before committing important transactions.
12	An online banking system should evaluate any input and allow only approved syntax in input fields.
13	The online banking system should not contain any IFrames, nor should it reference code from any site except its own.
14	Name resolution should not affect the connection inside an online banking system
15	An online banking system should only allow connections to and from approved clients and servers.
16	A keylogger should not be allowed inside an online banking system or should minimize the effect it may have if present.
17	An online banking environment should minimise the chances of an attacker using stolen information to gain access to a customer's online banking profile.

TABLE 1: INITIAL DESIGN REQUIREMENTS

If the client operating environment cannot change, then viruses cannot change files in the system, worms and Trojans cannot install and attack the system also software keyloggers are not able to install in the system and log keystrokes and the Internet browser is not able to change, for a man-in-the-browser attack. These requirements can be rewritten as a common requirement that states: ***The client operating environment should be persistent. It should***

not be possible to change any file or configuration in the client operating environment. This is the new requirement one.

Requirements seven to nine minimise attacks directed from the network to the client operating environment. These requirements are fulfilled if the client operating environment can be isolated on the network to only allow connections from the client operating environment to the banking system environment. Requirement seven to nine are redefined as follow: ***The client operating environment should be isolated on the network and only be allowed to connect to the banking system environment***. This is the new requirement two.

Requirement five concerns itself with buffer overflow attacks to a client from the network. Requirement six concerns itself with denial of service attacks to the client from the network. If the client can be truly isolated on the network, allowing only connections to the server-side environment and stopping any connections that are not the server-side environment then both requirement five and six are fulfilled. These requirements are fulfilled by fulfilling the requirement mentioned above, that states: ***The client operating environment should be isolated on the network and only be allowed to connect to the banking system environment.*** This is the new requirement three.

Requirement 15 also falls in this category, but is a special case in that it also states that only approved clients should be able to connect. To ensure a client is approved some type of validation should occur to confirm the client is approved. For this reason requirement 15 stays a special case and is defined as it is, but is now renumbered to be requirement number 8.

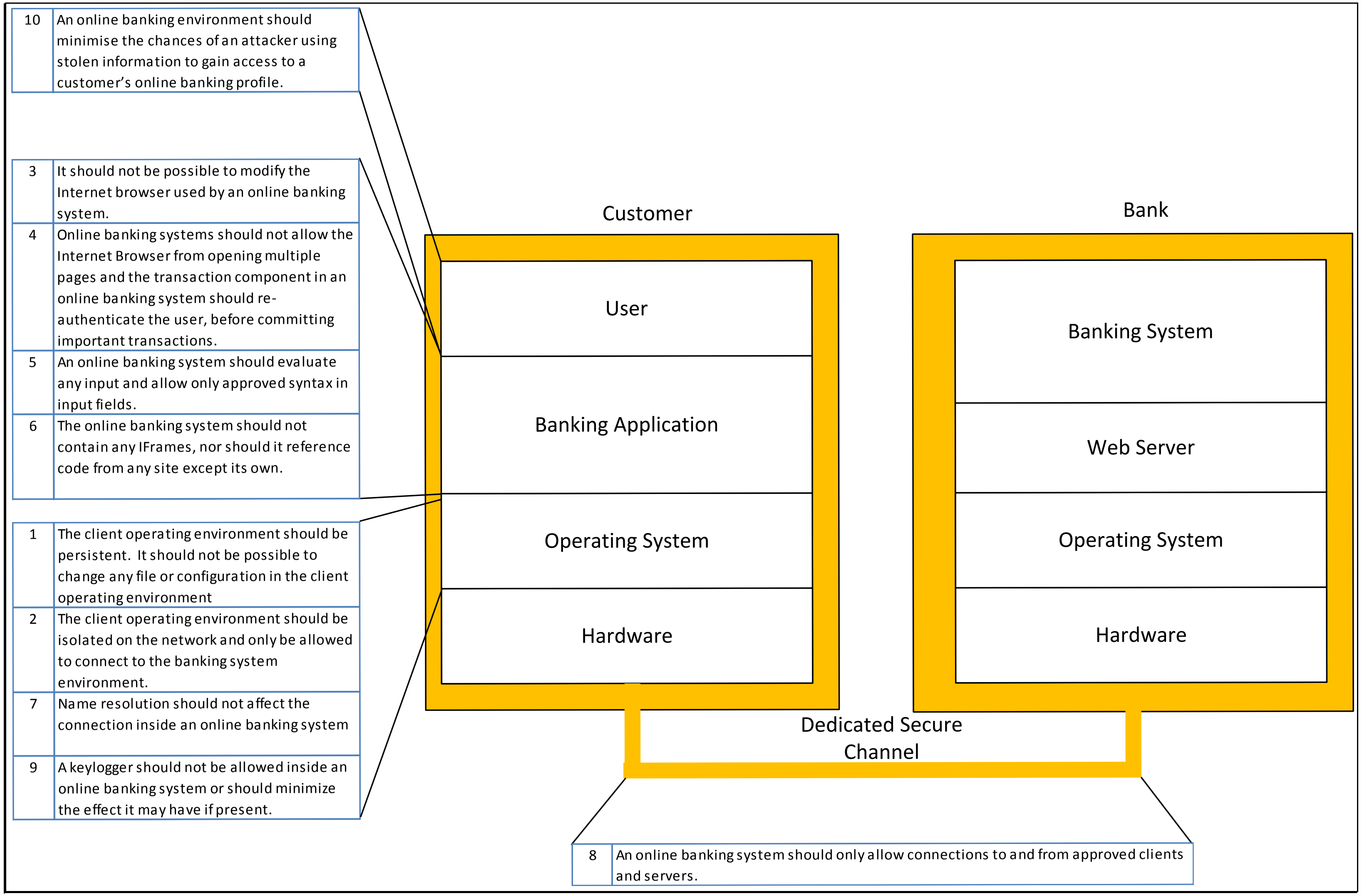
It has been mentioned that requirement 16 can be categorised together with requirements one to four. Since keyloggers can be hardware, no software system disallows a hardware keylogger from trying to log keystrokes. For this reason the requirement stays and is specifically defined, but is now renumbered to be requirement 9

The rest of the requirements are all individual and cannot be grouped together. Apart from the redefining of the requirements, the requirements are also grouped together according to three main categories customer, dedicated secure channel and bank. The requirements listed in Table 1 are now grouped together, renumbered, combined and described in Figure 2:10.

Figure 2:10 shows the Client environment on the left hand side with the various computer architectural layers, starting with hardware at the bottom, followed by the operating system, with the application running on the operating system, with the user using the application.

On the right hand side of Figure 2:10 is the server-side bank environment. It follows with the same architecture as on the client, with the hardware followed by the operating system, but in this case the web server acts as translation layer between the banking system and the operating system. In between the client and the server is the dedicated secure channel, which enables the communication between the client and the server.

Figure 2:10 is used as a basis to explain changes in the design requirements throughout the dissertation and is redefined after each chapter up to chapter 6.



10 An online banking environment should minimise the chances of an attacker using stolen information to gain access to a customer's online banking profile.

3 It should not be possible to modify the Internet browser used by an online banking system.

4 Online banking systems should not allow the Internet Browser from opening multiple pages and the transaction component in an online banking system should re-authenticate the user, before committing important transactions.

5 An online banking system should evaluate any input and allow only approved syntax in input fields.

6 The online banking system should not contain any IFrames, nor should it reference code from any site except its own.

1 The client operating environment should be persistent. It should not be possible to change any file or configuration in the client operating environment

2 The client operating environment should be isolated on the network and only be allowed to connect to the banking system environment.

7 Name resolution should not affect the connection inside an online banking system

9 A keylogger should not be allowed inside an online banking system or should minimize the effect it may have if present.

8 An online banking system should only allow connections to and from approved clients and servers.

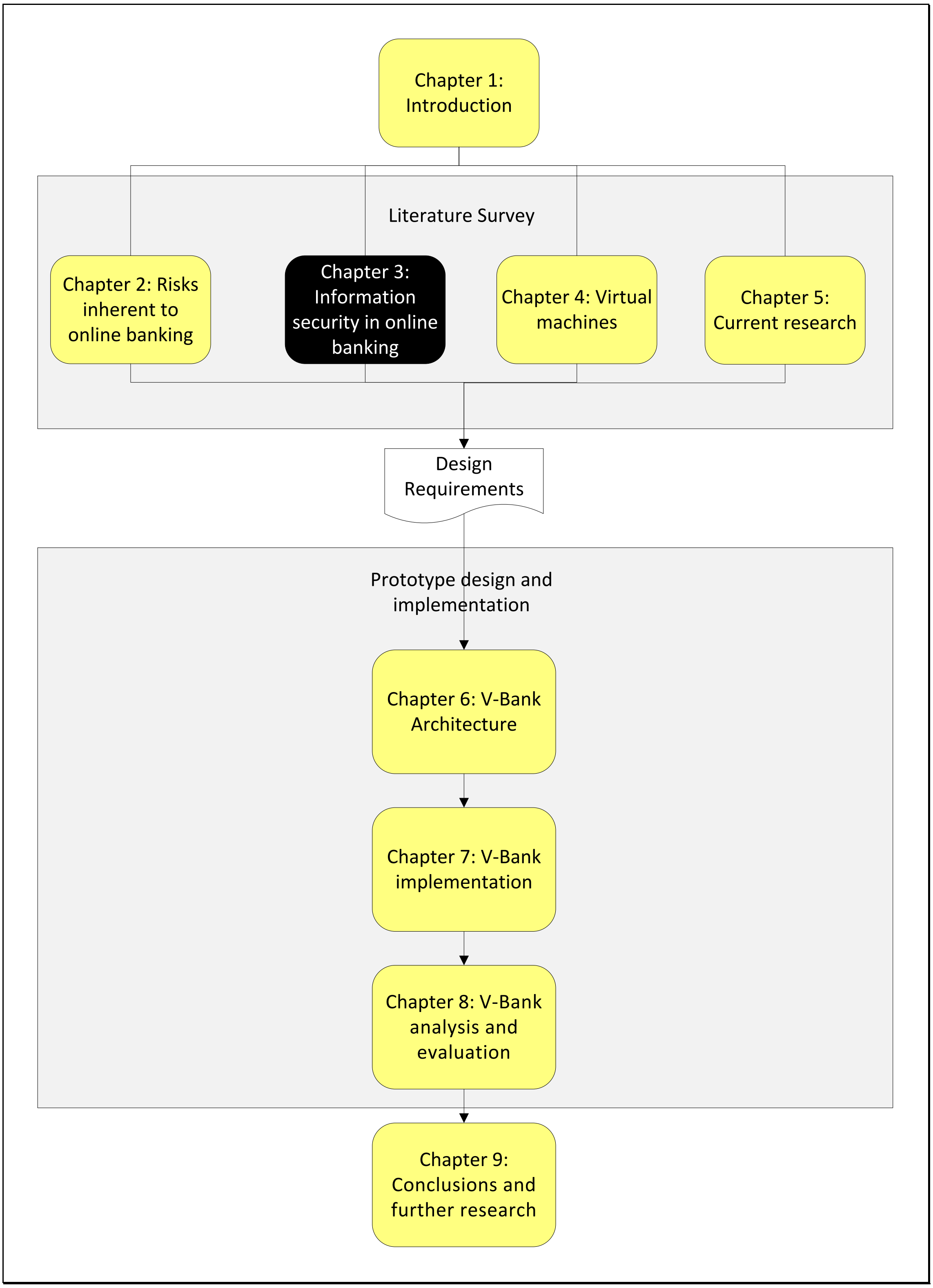
FIGURE 2:10 DESIGN REQUIREMENTS

6 Conclusion

An online banking environment consists of the client environment, the Internet or networking environment and the server-side environment. Each of these environments is susceptible to different types of attacks that affect online banking.

In order to design a safe online banking system, like the V-Bank system, a list of design requirements are defined to minimise the attacks on the online banking environment.

The next chapter evaluates the security of online banking environments. Some of the existing controls used by banks will be evaluated to determine whether it should be a requirement of a new online banking system that uses virtualisation.


Chapter 1: Introduction
Literature Survey
Chapter 2: Risks inherent to online banking
Chapter 3: Information security in online banking
Chapter 4: Virtual machines
Chapter 5: Current research
Design Requirements
Prototype design and implementation
Chapter 6: V-Bank Architecture
Chapter 7: V-Bank implementation
Chapter 8: V-Bank analysis and evaluation
Chapter 9: Conclusions and further research

Chapter 3: INFORMATION SECURITY IN ONLINE BANKING

1 Introduction

Chapter 2 identified some of the risks faced by a home user when conducting online banking. These risks are known by banks and they have implemented a number of controls to mitigate the risks (Hutchinson & Warren, 2003).

This chapter evaluates the controls used by banks to mitigate the risks associated with online banking. The structure and context of the chapter reflects the case study discussed in Chapter 1:3. During each step, the controls that mitigate the specific risk are evaluated and mapped against the five information security services categories defined in ISO 7498-2 (ISO, 1989).

The evaluation of each control may result in additional requirements to be added to the design requirements for our secure online banking system, V-Bank, defined in chapter 2.

Before the case study evaluation a brief overview of the five security services categories follow:

2 ISO 7498-2 Security Services

ISO 7498-2 defines five categories of information security services. These services are used as a standard against which controls can be mapped.

The ISO 7498-2 information security services are used as a basis to determine whether the existing controls used by banks fulfils the specific information security service requirement. Example: In the case study, John logs into his online banking profile. The login process verifies the username and password of John in order to authenticate him. The assurance of identity is known as authentication and is one of the ISO 7498-2 security services.

The five information security services identified by ISO 7498-2 are (ISO, 1989):

- Authentication
- Access control
- Data confidentiality
- Data integrity
- Non-repudiation (non-deniability)

A brief overview of each term follows.

2.1 Authentication

Authentication is the information security service that ensures the identity of an entity. Before an entity can gain access to a resource, the entity has to identify him to the security system of the resource (ISO, 1989).

To ensure that the entity is who he claims to be, the entity has to provide some type of secret that is only known to the entity and resource. The information security service that guarantees the identity of the entity is known as authentication.

2.2 Access control

Access control is the information security service that controls what type of access the entity has to a resource. The implemented security system can use various mechanisms to control access on the resource. Some mechanisms include, discretionary access control, mandatory access control and role based access control (ISO, 1989).

2.3 Data confidentiality

Data confidentiality is the information security service that ensures that the resource can only be seen by the entity and cannot be eavesdropped or intercepted. This information security service ensures data is kept confidential between the requesting entity and the resource system (ISO, 1989).

2.4 Data integrity

Data integrity is the information security service that ensures that the resource does not change while being transmitted. The data should not be tampered with, or any tampering of the data should be detected (ISO, 1989).

2.5 Non-repudiation

Non-repudiation is the information security service that ensures that the entity cannot deny creating or modifying the resource after the fact (ISO, 1989).

3 Information security controls applied by online banking systems.

Banks have implemented a number of controls to mitigate risks in online banking systems. Some of the controls are recommended, like anti-virus programs, and others are implemented and controlled by the bank (Federal Financial Institutions Examination Council, 2011).

The next section refers to a case study where a user needs to pay a company for work done by the company for him. The case study describes five steps that John goes through. The case study looks at each step and identifies the controls used in the different steps. The five steps that are part of the case study are:

- Step 1: Pre-session start-up
- Step 2: Establish secure channel
- Step 3: Login user
- Step 4: Review banking statement
- Step 5: Pay third party

The controls that are described are:

- Step 1: Anti-Virus programs (Standard Bank, 2012) (First National Bank, 2012)
- Step 1: Software updates (Standard Bank, 2012) (First National Bank, 2012)
- Step 2: SSL \ TLS (Zoller, 2011)
- Step 2: Trust anchors (Claessens, et al., 2002)

- Step 3: Fixed passwords (Claessens, et al., 2002)
- Step 3: Dynamic passwords (Claessens, et al., 2002)
- Step 3: Challenge \ response (Claessens, et al., 2002)
- Step 3: Digital signatures (Claessens, et al., 2002)
- Step 3: Hardware tokens (Markovic, 2007)
- Step 3: Smart cards (Shirey, 2000)
- Step 3: Random authentication codes (Goring, et al., 2007)
- Step 4: Discretionary access control (Samarati & De Capitani di Vimercati, 2001)
- Step 4: Mandatory access control (Samarati & De Capitani di Vimercati, 2001)
- Step 4: Role-based access control (Samarati & De Capitani di Vimercati, 2001)
- Step 5: Out-of-band verification (AlZomai, et al., 2008)
- Step 5: Challenge questions (Federal Financial Institutions Examination Council, 2011)
- Step 5: Private key signing (Claessens, et al., 2002)

These controls are now described and evaluated in context of an online banking session.

4 Case Study: Online banking security

In order to create a context for evaluating existing online banking security controls, the case study is extended in order to highlight the information security risks.

This section of the study looks at the process that John follows to conduct his online banking payment. The process starts with John switching on his computer and finishes when John pays the contractor. The process is broken up in five steps. The steps are:

- Step 1: Pre-session start-up
- Step 2: Establish secure channel
- Step 3: Login user
- Step 4: Review banking statement
- Step 5: Pay third party

Each step is then further separated in the following structure:

3.1 Step N: Description of the online banking process
3.1.1 Information security control 1
3.1.2 Information security control 2
3.1.3 Affected information security services
3.1.4 Identified design requirements

The structure this section starts with a description of the online banking process. The existing information security controls used by banks are then listed. Each information security control is described under its own sub-heading. Once all the information security controls are described, the affected ISO 7498-2 information security services are identified and evaluated. After the information security services are identified and evaluated the information security controls are investigate to see if new design requirements for V-Bank are identified.

The first step in the online banking process is when John switches on his computer.

John is a father of two teenagers. He works for a mining company as an accountant. He has a computer at home that is used by himself, his wife and his two teenage boys.

John uses the computer primarily for paying his bills and reading his e-mail. His wife uses the computer for online shopping and assisting the kids in their school projects. The two teenagers use the computer to buy and download songs for their music players, play games and do school projects.

4.1 Step 1: Pre-session start-up

John arrives home from work and has to pay AA-Electrical. John switches on his computer. The computer boots into Microsoft Windows 7 and presents John with the desktop interface. During start-up John's computer loads various device drivers and start-up programs.

One of the programs that starts is an Anti-Virus program. The Anti-Virus program helps to protect the computer from various forms of malware. An anti-virus program is a control that mitigates the risk of an attack from malware.

Another component that starts in Windows 7 is the Windows Update service. The Windows Update service contacts the Microsoft Windows Update infrastructure to determine if there are any security updates that need to be installed. The security updates typically patch operating system functionality from known vulnerabilities. The Windows Update service is an implementation of software updates that is a control that mitigates risks against known vulnerabilities in the operating system.

Banks cannot control the installation of anti-virus software or ensure that customers keep their computers up to date with security patches, but they recommend both controls (Standard Bank, 2012) (First National Bank, 2012) (Nedbank, 2012).

The two controls identified that mitigate online banking risks are:

- Anti-Virus Programs
- Software Updates

A short description of each of these controls follows.

4.1.1 Anti-Virus programs

To minimise the risks of malware hijacking online banking sessions, banks recommend customers to install an Anti-Virus program and to keep it up to date.

The biggest disadvantage of anti-virus programs is that most of them are reactive in terms of threats, and are limited in proactive protection. Proactive protection offered by anti-virus software does not always detect suspicious behaviour, or generates false negatives when real behaviour is identified as suspicious (Wu & Yu, 2011).

Anti-Virus programs can however help in protecting most Internet users provided the users keep their anti-virus software up to date and configured properly.

4.1.2 *Software updates*

Vulnerabilities in software can allow attackers unauthorised access to a specific system or communication channel.

Most vendors patch their software when vulnerabilities are identified. To ensure that the vulnerability cannot be exploited the system has to be updated regularly.

Banks advise their customers to update their software when new updates become available to minimise the risk of attackers exploiting a specific vulnerability in the system.

4.1.3 *Affected information security services*

The following information security service is affected:

- Authentication

Anti-Virus or updating of software is not required to ensure successful online banking transactions. Banks recommend both controls, which imply that it increases the security of online banking. Given that anti-virus programs protect a computer from malware, which can steal online banking login credentials, an anti-virus program helps to secure the authentication process.

Software updates protect the computer from software vulnerabilities. Software vulnerabilities allow malware to be executed on a computer, which in turn allow for the interception of online banking login details. Ensuring that a computer is up to date with the latest security patches helps to secure the authentication process.

Table 2 summarises the two controls present during step 1 of the case study. It shows that both Anti-Virus and software updates minimise the risks associated with authentication.

	Authentication	Access Control	Data Confidentiality	Data Integrity	Non-repudiation
Step 1					
Anti-virus	x				
Software updates	x				

TABLE 2: CONTROLS FULFILLING SECURITY SERVICES - STEP 1

With the identification of the controls in relation to the ISO 7498-2 security services, an investigation is necessary to determine whether the controls should be a requirement for an online banking system.

4.1.4 *Online banking design requirement*

In Chapter 2:3.1 a number of risks were listed that can be categorised as malware. During the evaluation of each malware-type a design requirement was generated to mitigate the risk for that specific malware type. A proper anti-virus program could theoretically fulfil the requirements mentioned, however traditional anti-virus technology is not 100% effective (Wu & Yu, 2011).

For the above mentioned reason an online banking system cannot rely on anti-virus technology as a requirement, instead the first requirement determined in Chapter 2 is the only requirement required related to anti-virus technology. The requirement states: “The client operating environment should be persistent. It should not be possible to change any file or configuration in the client operating environment”.

Software vulnerabilities need to be addressed using software updates, because it may allow attackers to execute code remotely or intercepting data between the online banking client and server. This results in a new design requirement. This new design requirement is number 11 because the previous chapter regrouped the design requirements in that chapter to only ten.

Design Requirements

11. Online banking systems must have the ability to be updated when software vulnerabilities are detected.

During step 1 of the case study, it was determined that anti-virus and software updates are controls that mitigate online banking risks. The case study now continues with the user establishing a connection to the online banking web site.

4.2 Step 2: Establish secure channel

Now that John has switched on his computer and logged into the operating system, he starts his web browser to connect to the online banking system. John double clicks the web browser icon that he has on his desktop and selects the online banking system link from his bookmarks in the web browser.

The bank ensures an encrypted connection with John’s web browser using Secure Socket Layer (SSL) encryption. The bank also presents a digital certificate to John’s web browser that identifies the web site to the web browser and user.

SSL encryption is also known as transport layer security (TLS). SSL and TLS are described in Chapter 3:4.2.1.

Apart from the technical information stored in the digital certificate, the digital certificate also acts as a trust anchor. It allows the banking client to visually verify the identity of the online banking web site (Claessens, et al., 2002).

In this scenario there are two controls that mitigate online banking risks. The two controls are:

- SSL \ TLS.
- Trust anchors.

The next two sections provide an overview of these two controls.

4.2.1 SSL \ TLS

Secure Socket Layer (SSL) was originally initiated by Netscape. Transport Layer Security (TLS) is the latest version and implementation of SSL and provides cryptographic support that SSL cannot offer (Zoller, 2011).

Even though there are various versions of the protocol, they all provide a secure communications channel between the bank and the client. The secure communications channel means that data between the client and the server is kept secret (confidentiality) and tampering of the data is detected (integrity).

SSL authenticates at least the banking side of the channel and banks usually implement a separate client authentication channel on top of the existing secure channel (Claessens, et al., 2002).

SSL does not provide non-repudiation. This means banking transactions are not signed and clients and banks can still repute specific transactions if a bank did not implement a separate non-repudiation mechanism (Claessens, et al., 2002).

SSL v2 was developed by Netscape in 1996 and in 2012 the standard is 16 years old. The system suffers from vulnerabilities and is no longer supported by most major Internet browsers.

SSL v3 added many features that were not present in SSL v2. After the development of SSL v3 TLS 1.0 was developed and has since been revised up to TLS 1.2.

The various versions are evaluated and adopted by the Internet Engineering Task Force (IETF).

To enable a specific version of SSL or TLS, both the client and the server has to support that version. In May 2012 only the Opera browser and Internet Explorer 7 and above supported TLS 1.2. On servers only Microsoft IIS 7.5 (The web browser that comes with Windows Server 2008) and web servers that support NSS (National Security Services) supported TLS 1.2 (Zoller, 2011)

In all the cases TLS 1.2 is not enabled by default and both the server and client have to be configured to support TLS 1.2.

Because vendors are slow to support the latest TLS versions, most banks can only configure their servers to support SSL v3 and TLS 1.0, this leaves most electronic banking systems open to vulnerabilities that exist in these protocols.

4.2.2 Trust Anchors

A trust anchor is defined by Claessens, et al. as a visual indicator that the user is communicating to the correct banking web site. Most Internet banking web sites make use of the green lock, displayed by the Internet browser. This green lock is displayed by the web browser when the certificate by the online banking web site passes a few tests.

1. Is the certificate given by the banking web site trusted? This is tested by looking at the certificate chain of signatures to ensure each entity that signed the certificate is from a trusted entity. These root trusted entities are defined by trusted root certificates installed in the operating system.
2. Is the certificate valid in terms of dates? Any certificate is only valid between certain dates. A browser does not trust a certificate when the current date is outside the valid date range of a certificate.
3. Is the name of the entity in the certificate the same of the name of the web site? If the certificate was given to a specific entity (web site) and displayed by another web site, then the browser does not trust the certificate.

This certificate trust link however can be changed and Internet browsers displaying incorrect trust information when an attacker can install its own trusted root certificate into the operating system. Any user who then visually looks for green lock in their browser will find a green lock even if the web site they are connected to, is not actually their banking web site, even though it may look the same.

Because of the above reason, some banks also include a secret identifier, defined by the user during their first banking session. If the user does not see this secret identifier, after he\she logged in, then the user should know that the banking web site is not the correct web site.

These trust anchors help ensure that the communication channel is established to a trusted web site.

4.2.3 *Security Services*

SSL and TLS ensure that data is encrypted between the client and the online banking system. It also ensures that tampering is detected and authenticates the server to the client.

Trust anchors identify the server to the client. The client verifies the identity visually, which acts as authenticating the server's identity.

Table 3 summarises the controls mapped against the security services they fulfil. SSL\TLS keeps data confidential, and ensures integrity and server identification. Trust anchors ensure server identification.

	Authentication	Access Control	Data Confidentiality	Data Integrity	Non-repudiation
Step 2					
SSL\TLS	x		x	x	
Trust Anchors	x				

TABLE 3: CONTROLS FULFILLING SECURITY SERVICES - STEP 2

4.2.4 Online banking design requirement

The data transferred between an online banking client and the server is private and should be kept confidential. Clients and banks would not like potential attackers from gaining access to login information or personal banking information. Since there are already identified vulnerabilities in SSL v3 and TLS 1.0, the latest version of TLS is required, which is TLS 1.2.

Design Requirements

12. Online banking systems must ensure TLS 1.2 channel security

Banking customers should also have the ability to ensure that they can successfully identify the banking web site that they are connected to, is the correct web site. This requirement may not be necessary if requirement four, in Figure 2:10 can be met that states: "The client operating environment should be isolated on the network and only be allowed to connect to the banking system environment". The existing requirement however does not give any visual indication to the user that the online banking system connects to the correct server-side system.

Design Requirements

13. Online banking systems must have a visual trust anchor.

John's computer started, with the browser connected to the online banking web site. John is now ready to access his personal profile on the online banking system.

4.3 Step 3: Login user

Before John can gain access to his personal profile in the online banking system, John has to identify himself to the online banking system and the online banking system has to verify the identity of John.

John's bank requires John to type in his account number, as identifier and also a personal identification number (PIN) and a password. Only John should know his PIN and password linked to his account number. If John types in the PIN and password correctly, the bank assumes that it is in fact John accessing his bank account.

By providing an account number, with correct PIN and password, John authenticated himself to the bank.

Authentication is the process that ensures that a person is actually who he\she claims to be. The first step in the authentication process is where a user identifies him and then shares something with the other party that authenticates himself.

This sharing can take the form of something they know, they have or they are (von Solms & Eloff, 2004).

Something they know is typically a pin number, or password. Something they have can be in the form of a smart card, or hardware token. Something they are is some type of biometric authentication, like a fingerprint (von Solms & Eloff, 2004).

The controls most commonly used for authenticating users according to Claessens, et al (2002) are:

- Fixed Passwords.
- Dynamic Passwords.
- Challenge \ Response.
- Digital signatures.
- Hardware tokens.
- Smart Cards.
- Random authentication codes.

The next few sections provide an overview of these controls.

4.3.1 Fixed Passwords

A fixed password is a secret combination of letters and\or numbers that is known only by the user. It is fixed, because the user initially creates the password and only changes when the user changes the password.

It is possible for a system to force a user to change his password, but the password is still considered fixed, since it doesn't change dynamically.

The problem with passwords is that they are subject to brute force attacks. Brute force attacks are the process where an attacker starts guessing passwords and continues guessing, until the password is found.

To minimise the effectiveness of brute force attacks, banks recommend that a user chooses a complex password. Complex passwords in general refer to a password that contains upper case, lower case, numbers and special characters. The longer the password is, the more difficult it is to guess.

Banks may also implement controls where it locks an account after a certain number of failed attempts. This stops brute force attacks from continually trying to guess a password, but opens up the accounts to denial of service attacks.

Most banking systems use fixed passwords, simply because it is a low cost security control that can be effective.

4.3.2 Dynamic Passwords

A dynamic password is also called a one-time password. Dynamic passwords change from one session to the next and can only be used once (Claessens, et al., 2002).

Dynamic passwords can be implemented by using scratch lists. A scratch list is a piece of paper with a list of passwords on. When a user used a password, the user must scratch the password from the list and use the next one (Claessens, et al., 2002).

Dynamic passwords can also be implemented using extra software. The extra software must exist on a computer, or it can exist on mobile phones. Some hardware tokens also generate one time passwords.

4.3.3 Challenge \ Response

Challenge \ Response is a scheme where the user identifies him to the bank by responding to some type of challenge. The bank challenges the client with something, which changes from session to session. The client responds to the challenge in an appropriate method.

The advantage of this system is that the response that travels over the communication channel changes from session to session.

An example of a challenge \ response scheme is where the bank issues a challenge to the client. Example: The bank sends a random number of characters to the client. The client then signs this challenge with his private key. This signed response is different with each session, because the challenge is different with each session. The bank ensures the signed challenge was signed by the client, by using the client's public key (Claessens, et al., 2002).

4.3.4 Digital signatures

In a SSL\TLS communication channel the client can identify himself to the server by signing a hash of the previously exchanged handshaking methods. The server verifies the signature by using the client's public key.

Individual transactions can also be signed to ensure non-repudiation on the transactions.

Private keys can be stored on hardware tokens, but it can also be stored in software. Private keys stored in software however is fairly vulnerable (Shamir & van Someren, 1999).

4.3.5 Hardware tokens

A hardware token is usually a small piece of hardware that is given to the user when they register for Internet banking. Hardware tokens can generate one-time passwords in the case where a one-time password is required by the Internet banking system. These one-time passwords are usually used before authorising a banking transaction (Markovic, 2007).

4.3.6 Smart Cards

A smart card is a credit card sized device that contains a chip, with its own processor, memory and input – output channel (Shirey, 2000).

Smart cards can contain cryptographic information, such as the private key in a public \ private key pair. Access to the private key is usually controlled through a pin number or password.

The advantage of a smart card is that it combines two of the three authentication components mentioned in Chapter 3:4.3. The components are something you have (smart card, with private key) and something you know (PIN number or password).

The disadvantage of smartcards is that there is a cost associated with the actual card, and not all computers have the native ability to read smart cards. This makes the deployment of smart cards as an authentication device difficult for banks to implement.

4.3.7 Random authentication codes

Some banks implement two steps of authentication during an online banking session. The first step is in the form of the identification and authentication using something like a pin. The second step is then a prompt for a password, but only certain characters of the password are asked for during the second step.

The main reason for only asking for a few characters for the password is to minimise the chance of a keylogger capturing the user's full password in a single session.

It should be noted that implementation of the random authentication codes affects the effectiveness of the control. It has been argued that if the system asks the same random authentication characters, until a successful login, that an attacker can gain the knowledge of the full authentication password. (Goring, et al., 2007)

Another disadvantage of the system is that the full password must be stored on the banking system. Many security discussions concerning the storing of passwords recommend that passwords should never be stored in a system, only a hash of the password. In the case of implementing random authentication codes, the full password has to be stored, because the authentication system should know the exact characters of the password, not just the hash of the full password (Guimaraes, 2006)

The random authentication codes however can still be combined with another authentication mechanism to minimise the chances of an attacker using a keylogger to gain knowledge of the user's login details.

4.3.8 Security Services

John logged onto the online banking system in the case study. The various controls listed in this section are all controls that enable the act of authenticating the user to the system.

All of these controls provide authentication to the online banking system and very little of the other ISO 7498-2 security services.

Table 4 summarises the different controls and highlights the ISO security service fulfilled by the specific control.

	Authentication	Access Control	Data Confidentiality	Data Integrity	Non-repudiation
Step 3					
Fixed Passwords	x				
Dynamic Password	x				
Challenge \ Response	x				
Digital signatures	x				
Hardware tokens	x				
Smart Cards	x				
Random authentication codes	x				

TABLE 4: CONTROLS FULFILLING SECURITY SERVICES - STEP 3

4.3.9 Online banking design requirements

Online banking systems contain personal and confidential information. It is the bank's responsibility to ensure that the information is protected from unauthorised access. Any online banking system must have some type of authentication process. Users of the system must prove their identity to the system using some type of authentication control.

Design Requirements

14. Online banking systems must properly authenticate users before allowing access.

John has logged into the online banking profile. In the next section it is made clear that John can only access information that he has been granted access to.

4.4 Step 4: Review banking statement

After logging into the online banking system, John is presented with a number of features that exist in the online banking system. One of the features is for John to retrieve an account statement. John uses the information in the account statement to verify that he has enough funds to pay AA-Electrical.

John's wife Lisa also has an account with the same bank and also uses online banking to gain access to her information. Even though John can see information about his account, he cannot view the information about Lisa's account.

The bank controls access to their customers' accounts and ensure that only the account belonging to the specific customer can be accessed by that same customer. The customer cannot give access to another customer to his account information.

The banks not only control access to the account information, but also control what can be done with the bank account. The bank can control whether a specific bank account can be used to make payments, or the customer can only view a statement for the bank account.

The controlling of access can fall into a number of categories. Samarati & De Capitani di Vimercati (2001) lists three main classes of access control policies. They are:

- Discretionary (DAC).
- Mandatory (MAC).
- Role-based (RBAC).

The next three sections provide an overview of these three classes of access control policies.

4.4.1 Discretionary

In discretionary access control policies, access is granted to an object based on the identity of the requestor and a set of access rules. The word "Discretionary" in the term means that in this type of policy users can be given the ability to give other users the same privileges as they have on the objects.

Discretionary access control policies are usually implemented in operating systems. Microsoft adopted the DAC policy for their Windows NT operating system when it was first developed. This policy has since been used in their operating systems like Windows XP and Windows 7 (Russinovich & Solomon, 2005).

In the Windows NT – type operating systems objects are assigned an access control list (ACL), with a number of access control entries (ACE). Users gain access to the operating system by logging in. The user accounts contain a security identifier (SID) that identifies them to the system. The SID, together with what type of access the user has on the object gets added to the ACL as an ACE. When a user wants to access the object, the operating system determines the access granted by evaluating the ACL for access control entries that the user belongs to (Russinovich & Solomon, 2005).

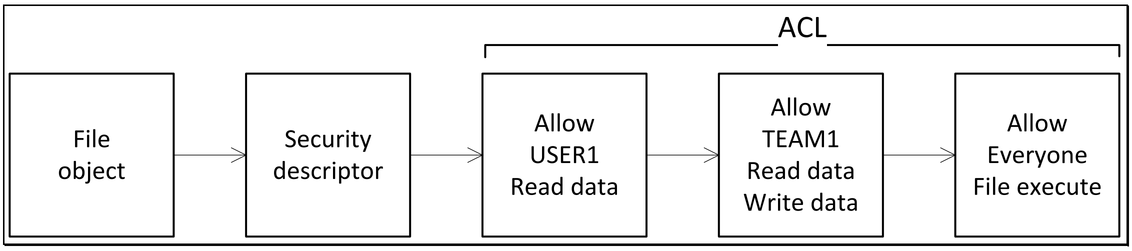


FIGURE 3:1 DISCRETIONARY ACCESS CONTROL LIST

Figure 3:1 is a simplified picture of a file with a file object and the ACL (Russinovich & Solomon, 2005). It shows that USER1 can read data from the file, but cannot write changes back to the file. It also shows that members of the TEAM1 group can read and write data to the file and that everyone can execute the file.

The discretionary access control policy defines three important aspects. The first is the object. This is the object that needs to be secured. In an operating system this can be a file. In a relational database management system this can be table or view (Samarati & De Capitani di Vimercati, 2001).

The second important aspect of the discretionary access control policy is the subjects. Subjects are the "who" that wants to access this object. This can be user accounts, but can also include group accounts or processes (Samarati & De Capitani di Vimercati, 2001).

The third important aspect is the actions. Actions can include things like read access, write access, but may also include actions like ownership and control (Samarati & De Capitani di Vimercati, 2001).

Another important aspect of DAC is the principle of the administrative policy. The administrative policy defines who is allowed to change the allowed accesses. A number of administrative policies exist (Samarati & De Capitani di Vimercati, 2001):

- Centralised. Only one authoriser exists in the system that can grant and revoke authorisation to users.
- Hierarchical. One authoriser assignes responsibilities to other administrators so that they can grant and revoke authorisation.
- Cooperative. More than one person is required to change the authorisation of an object.
- Ownership. When a user creates an object, the user typically becomes the owner of the object. The owner can change the access of the object.
- Decentralised. Decentralised means that the owner of the object can further delegate rights to users to change authorisations on the object.

The mandatory access control policies control the flow of information through the use of labels.

4.4.2 Mandatory

Mandatory access control policies are mandated from a central authority. It allows access through a multi-level classification of subjects and objects. Subjects are active entities that request access and objects are passive entities that stores information (Samarati & De Capitani di Vimercati, 2001).

Subjects and objects get classified according to different levels of access. The most common known categories are Top Secret (TS), Secret (S), Confidential (C) and Unclassified (U), where the levels of the categories are TS>S>C>U (Samarati & De Capitani di Vimercati, 2001).

The secrecy based model implements the mandatory access control policy different from what Microsoft implemented. The secrecy based model, first defined by Bell and LaPadula, uses the concepts of No-Read-Up and No-Write-Down (Bell & LaPadula, 1973).

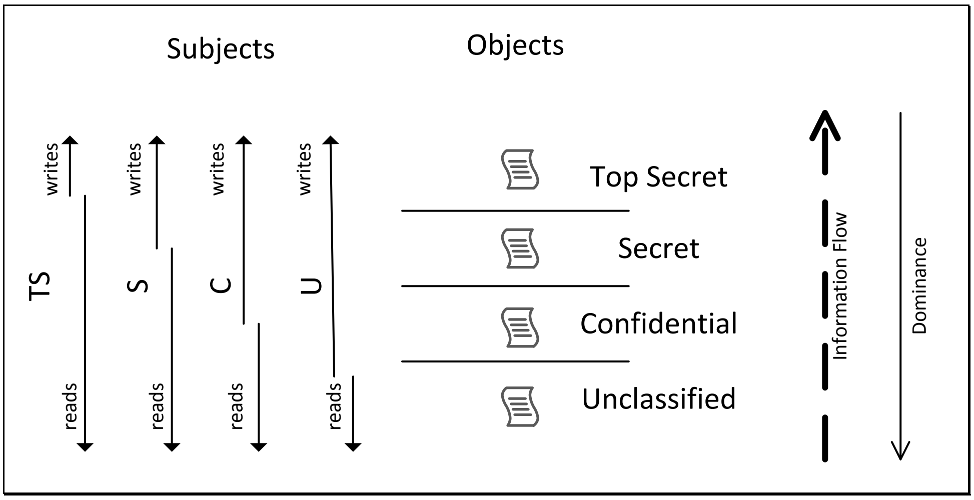


FIGURE 3:2 INFORMATION FLOW FOR SECRECY

No-Read-Up means subject with a lower level category does not have read access to objects with a higher classification category.

No-Write-Down means subjects cannot write information into objects with a lower classification category.

The No-Read-Up and No-Write-Down principle ensures that information gets created and flows upwards to subjects in higher or the same classification categories. Subjects however can only read information stored in objects that are on classification levels the same or lower. This ensures someone with a Secret classification cannot read information on objects on the Top Secret classification category, but can produce information accessible by people on the same or higher classification category.

Figure 3:2 summarises the Bell & LaPadula model (Samarati & De Capitani di Vimercati, 2001). On the rights hand of the figure it describes that information flows upwards from the lower classification levels to higher classification level. The rest of the figure explains that Top Secret (TS) subjects can write only TS objects, but can read all objects in lower classification levels.

Biba (1977) proposed an implementation of MAC, but which ensures integrity. Biba proposed his model to ensure that low level classification subjects cannot make changes to objects in systems on a higher classification level. This ensures integrity in the system. Biba also suggested classification levels, but suggested a No-Read-Down and No-Write-Up concept (Biba, 1977).

An example of the classification levels in Biba's model can be Crucial (C), Important (I) and Unknown (U). In this example Biba's model means that a subject with Important classification cannot write information to an object with a Crucial classification level, but can read information that is Important and Unknown.

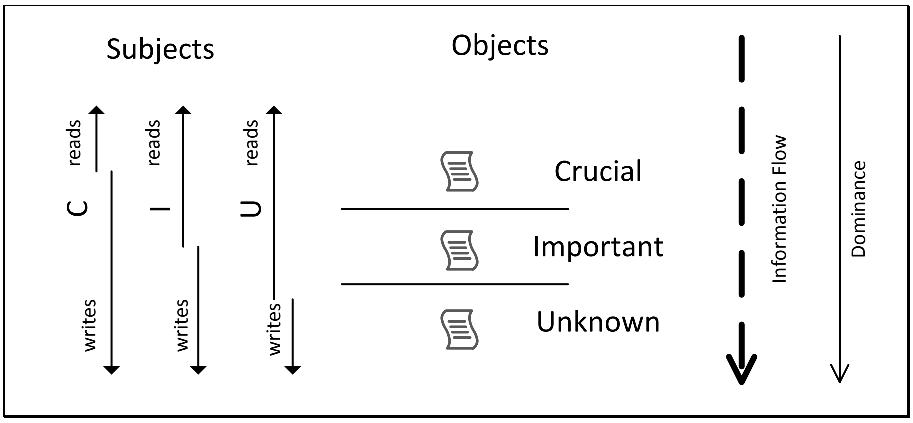


FIGURE 3:3 INFORMATION FLOW FOR INTEGRITY

Figure 3:3 summarises Biba's model. In this model information flows from higher classification levels to lower levels (Samarati & De Capitani di Vimercati, 2001). Higher level classification subjects can make changes to objects in the system on lower levels. Lower level subjects, which may be from untrusted environments, won't have the ability to make changes to objects in higher classification levels.

From Windows Vista onwards Microsoft implemented a form of Biba's model into the operating system called mandatory integrity control. Microsoft also defines different categories for subjects and objects. They are system, high, medium and low. Microsoft however only uses the MAC when a subject tries to write to an object. A subject can write to an object if the object's level is the same or lower than the subject's level (Riley, 2006). This helps protect operating system users from opening a virus attached to an e-mail (low level), which may infect operating system files (medium or higher level).

4.4.3 Role-based

The last access control policy class defined by Samarati & De Capitani di Vimercati is role-based access control policies (RBAC).

RBAC concerns itself not so much with individual users being given specific rights in a system, but base access rights on roles. Many implementations of RBAC has been suggested. There is however one single principle and that is users are assigned to a role or roles. The roles then have a specific right or access in a system (Samarati & De Capitani di Vimercati, 2001).

RBAC is very flexible in its implementation. RBAC allows policies to define implementation where users can be a member of only one role at a time, or multiple roles at a time. It can even allow roles to be members of other roles. It can also define rules where certain roles are mutually exclusive, meaning a user cannot be member of the one role and a specific other one. RBAC allows for very specific policy implementation in a system, based on responsibilities, instead of individual user names (Samarati & De Capitani di Vimercati, 2001).

Many commercial systems have adopted RBAC policies as the access control system implemented (Samarati & De Capitani di Vimercati, 2001). An online banking system can adopt a RBAC policy and implement this to ensure access in the system is controled and adheres to specific system wide policies.

DAC, MAC and RBAC are all classes of access control policies. Any system that implements access control adopts some type of access control policy. Access control is one of the ISO7498-2 security services.

4.4.4 *Security Services*

After John logged into the online banking system, John only has access to his own bank account information. John is also assured that no other banking users have access to his bank account. This assurance is implemented using an access control policy defined by the bank. The access control policy of the bank also controls what actions John can do on his bank accounts. John's bank controls his home loan account, so that John can only transfer money to his home loan account, but cannot transfer money out of his home loan account. John's cheque account however allows John to transfer money into and out of the account.

The three different classes of access control policies that exist can be implemented in a system, using different implementation mechanisms. Table 5 summarises the policies described in this section. All the policies if implemented fulfil the access control security service, but MAC can also add integrity capabilities to a system.

	Authentication	Access Control	Data Confidentiality	Data Integrity	Non-repudiation
Step 4					
DAC		x			
MAC		x		x	
RBAC		x			

TABLE 5 ACCESS CONTROL POLICIES - STEP 4

4.4.5 *Online banking design requirements*

Online banking systems contain critical and personal information. Banks have the obligation and legal requirement to protect individual user account information. Banks must ensure that access to individual user account information is controlled.

Design Requirements

15. Online banking systems must implement a RBAC policy to ensure access to user information is controlled.

Now that John has verified that he has enough funds is he ready to pay AA-Electrical.

4.5 Step 5: Pay third party

John verifies that AA-Electrical is not an existing beneficiary and click on the function to create a new beneficiary. John enters AA-Electrical's banking details into the new beneficiary fields and presses the OK button.

AA-Electrical is now a beneficiary in John's online banking profile. John clicks on the button in the online banking system that starts the function to pay a beneficiary. The pay-beneficiay-function asks John which beneficiary he would like to pay, as well as from which account he would like to pay the beneficiary and the amount of money that John would like to pay. John fills in all the fields to pay AA-Electrical and presses the OK button.

As soon as John presses the OK button his mobile phone receives a short message service (SMS) message from the bank with a random number. The online banking system prompts John to type the random number he just received in the SMS message into the online banking system. The online banking system verifies the random number John typed with the one they sent and confirms the payment transaction.

John prints out the payment transaction for his own records and faxes the printed out copy to AA-Electrical to let them know that he initiated payment of the account. John logs out of the online banking system.

The Federal Financial Institutions Examination Council (FFIEC) in the United States identified that there are different levels of risk in online banking systems, dependent on the client. In general consumer customers have a different risk profile than commercial clients. Consumer customers in general transfer smaller amounts less frequently than commercial clients. The FFIEC recommends that banks implement a layered security program (Federal Financial Institutions Examination Council, 2011).

The FFIEC recommends two controls that can be used for authorising transactions. They recommend out-of-band verification and challenge questions (Federal Financial Institutions Examination Council, 2011). The author would also like to add the possibility of using private key signing to verify transactions. An overview of these three controls follows.

4.5.1 Out-of-band verification

Out-of-band verification refers to the process of re-authenticating a user before a transaction is approved using another type of authentication control not using the same communications channel as the existing communications channel. A popular control that is being implemented by banks is one-time-passwords sent to the user using SMS (AlZomai, et al., 2008).

When the banking client creates a beneficiary to receive a payment, or when the banking client transfers an amount of money to another account, the bank sends an SMS with a one-time-password. The client must then enter the one-time-password into a field during the transaction on the online banking system.

If the one-time-password that was entered by the user, matches the one-time-password that was sent, then the bank knows that the user have received the SMS on his personal phone and makes the assumption that the transaction is authorised by the user.

The advantage to the SMS authorisation scheme is that it is a fairly cheap way to implement extra security into the online banking system.

The disadvantage to the system is that it is susceptible to a SIM swop. A SIM swop is where an attacker approaches a mobile phone provider and claims the identity of the user. The attacker then informs the mobile phone provider that he lost or damaged his SIM card that is in his phone. The bank then swops the SIM card with a new SIM card, but with the same number. From this point onwards any telephone calls and message is relayed to the new SIM card, instead of the old card. Figure 3:4 shows a simplified example of how such a SIM swop scam can be used to gain access a customer's online banking system (Jordaan & von Solms, 2011).

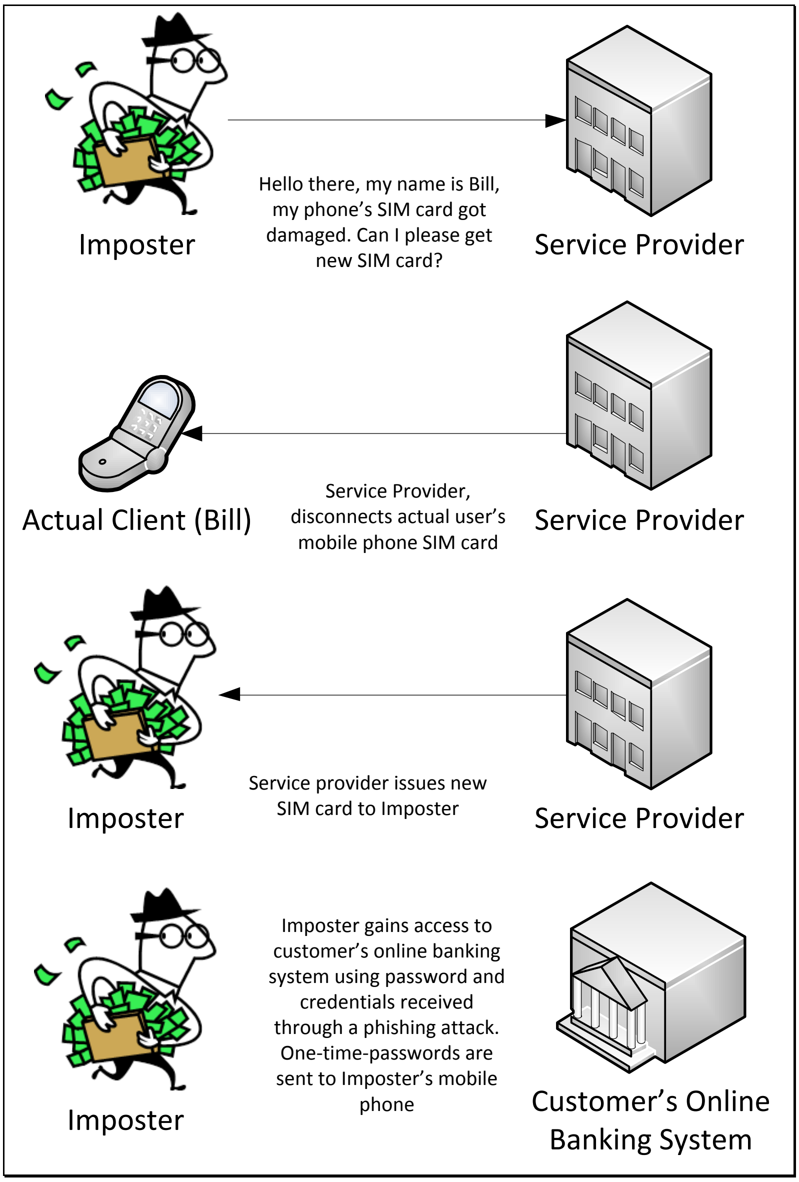

Imposter
Hello there, my name is Bill, my phone's SIM card got damaged. Can I please get new SIM card?
Service Provider
Actual Client (Bill)
Service Provider, disconnects actual user's mobile phone SIM card
Service Provider
Imposter
Service provider issues new SIM card to Imposter
Service Provider
Imposter
Imposter gains access to customer's online banking system using password and credentials received through a phishing attack. One-time-passwords are sent to Imposter's mobile phone
Customer's Online Banking System

FIGURE 3:4 SIMPLIFIED EXAMPLE OF A SIM SWOP SCAM

Another disadvantage to the SMS authorisation system is that it is also vulnerable to a man-in-the-middle or man-in-the-browser attack. It is possible for a spoofed web site to intercept, any passwords from the user (including the one-time-password sent via SMS), inject their own account and amount information, but displays false information to the user.

4.5.2 *Challenge questions*

The FFIEC recommends challenge questions as an effective component in a layered security program. The FFIEC lists challenge questions as a possibility for authorising, or re-authenticating a user, during banking transactions (Federal Financial Institutions Examination Council, 2011)

Security questions present the user with a question whose answer is only supposed to be known by the user and the bank. Different types of security questions exist. They are:

- Sensitive security questions. These questions are sensitive in nature and should be known by only the bank and the user. This can be social security numbers, bank account numbers or ATM PIN number (Rabkin, 2008).
- Personal verification questions. The questions ask personal things about the user's background, such as the user's mother's maiden name (Rabkin, 2008).

Users of the system are able to select the questions and provide answers initially during first time usage of a system. The FFIEC also recommends that the questions provided to the user may also include a "red herring", which a legitimate user will notice as nonsensical, but a fraudster not (Federal Financial Institutions Examination Council, 2011).

4.5.3 *Private Key signing*

When a user of an Internet banking system creates a transaction, the user can utilise his private key to sign the transaction. The signing of the transaction authorises the bank to continue with the transaction. The signing of the transaction also authenticates the user at that time, since only the user should have access to his private key.

If the bank uses the content of the transaction and signs the content, then they also have a non-repudiation record. This signature can then be used to verify the content of the transaction.

There are a number of challenges to overcome before private key signing can be used in an online banking system. The first challenge is the distribution of the private keys to the users. This can be in the form of some type of token or smart card.

The second challenge is that only the Netscape browser includes a JavaScript mechanism to sign forms. Any other browser must use secondary programs, like Java applets or standalone applications to enable the signing of forms (Claessens, et al., 2002).

4.5.4 *Security Services*

The three controls that may apply during the payment of a third party are:

- Out-of-band verification
- Challenge questions
- Private Key signing

These controls fulfil various areas in the ISO 7498-2 security services.

Table 6 maps the three controls against the relevant security services. All three controls authenticate the user. This can be through one-time-passwords, security questions or access to the user's private key. It can also be argued that re-authentication of the user adds some type of non-repudiation against the transaction. The author argues that re-authenticating the user does not provide non-repudiation, since authenticating a user does not imply non-repudiation. Strict non-repudiation only occurs in the case of the Private Key signing mechanism. The Private Key signing mechanism ensures that the integrity of the data through the signature, as well as provide non-deniability because of the signature.

	Authentication	Access Control	Data Confidentiality	Data Integrity	Non-repudiation
Step 5					
Out-of-band verification	x				
Challenge questions	x				
Private Key signing	x			x	x

TABLE 6: AUTHORISATION CONTROLS - STEP 5

4.5.5 *Online banking design requirements*

The FFIEC recommends the implementation of a layered security program. The layered security program implies that that type of information security control implemented is dependent on the risk profile of the user. This recommendation is applied to the online banking design requirements.

Design Requirements

16. Online banking systems must implement a layered security program.

Now that John has completed the payment to AA-Electrical using online banking is it time to review the impact the investigation regarding the various controls used by banks has on the requirement of an online banking design.

5 Design requirements and Information security services

This chapter evaluated the information security controls in an online banking environment. The evaluation was conducted to determine whether some of the controls or security services should be part of an online banking system.

Table 7 lists the various controls used by banks during an online banking session and map it against the ISO 7498-2 information security services. The table shows that most controls used by banks concern themselves with authentication. This observation can also imply that the area of authenticating the user is the area where banks experience the biggest risk. Banks want to put as much controls in place to ensure the identity of the user using the online banking system.

	Authentication	Access Control	Data Confidentiality	Data Integrity	Non-repudiation
Step 1					
Anti-virus	x				
Software updates	x				
Step 2					
SSL\TLS	x		x	x	
Trust Anchors	x				
Step 3					
Fixed Passwords	x				
Dynamic Password	x				
Challenge \ Response	x				
Digital signatures	x				
Hardware tokens	x				
Smart Cards	x				
Random authentication codes	x				
Step 4					
DAC		x			
MAC		x		x	
RBAC		x			
Step 5					
Out-of-band verification	x				
Challenge questions	x				
Private Key signing	x			x	x

TABLE 7: SUMMARY OF INFORMATION SECURITY SERVICES

Table 7 also shows that some information security services, like data confidentiality and non-repudiation have very few implemented information security controls. Looking a little bit closer however shows that data confidentiality uses SSL\TLS security channels to encrypt the data. As described SSL and TLS has gone through various versions and banks rely on the technology to ensure that data stays confidential.

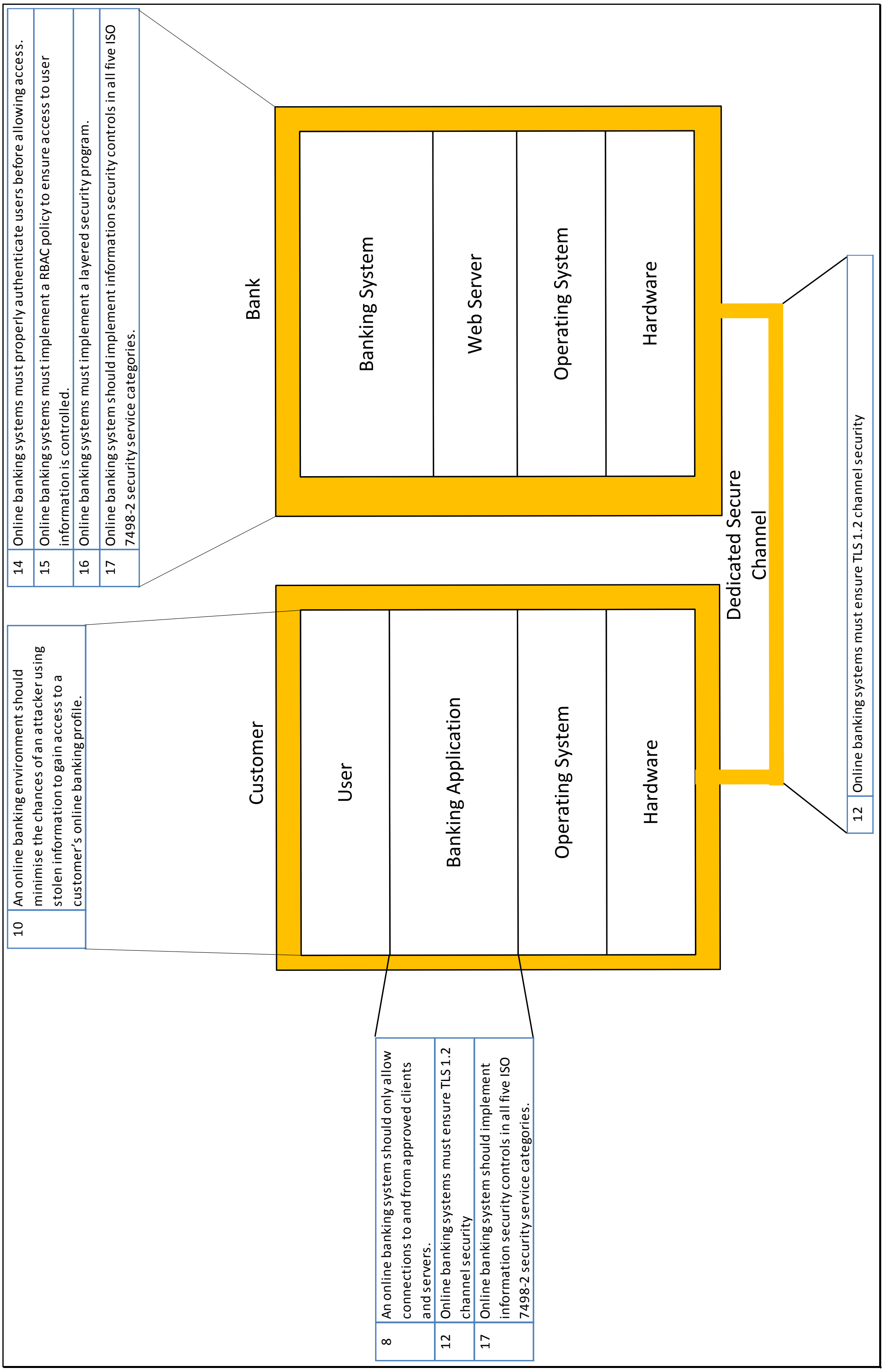
Non-repudiation is however a different problem. It seems that very little emphasis is placed on non-repudiation. It can be argued that some authentication information security controls ensures some form of non-repudiation. The author argues that an information security authentication control does not imply non-repudiation. Instead some type of electronic signature is required to ensure non-repudiation.

Before a summary of the design requirements are given, a new design requirement is defined. Online banking systems should, ensure controls fulfilling all five ISO 7498-2 information security services are implemented in an online banking system. By not implementing information security controls for each ISO 7498-2 security service, the bank opens itself to risks in that specific category.

Design Requirements

17. Online banking system should implement information security controls in all five ISO 7498-2 security service categories.

In this chapter nine new design requirements were identified. The nine requirements are design requirement number 11 to 17 and are shown in Figure 3:5. The new design requirements have been categorised under each of the case study's main process steps. The last design requirement was given the category "ISO 7498-2 information security services", because it compels banks to ensure online banking systems implement international standards on their online banking systems.



10	An online banking environment should minimise the chances of an attacker using stolen information to gain access to a customer's online banking profile.

14	Online banking systems must properly authenticate users before allowing access.
15	Online banking systems must implement a RBAC policy to ensure access to user information is controlled.
16	Online banking systems must implement a layered security program.
17	Online banking system should implement information security controls in all five ISO 7498-2 security service categories.

8	An online banking system should only allow connections to and from approved clients and servers.
12	Online banking systems must ensure TLS 1.2 channel security
17	Online banking system should implement information security controls in all five ISO 7498-2 security service categories.

12	Online banking systems must ensure TLS 1.2 channel security

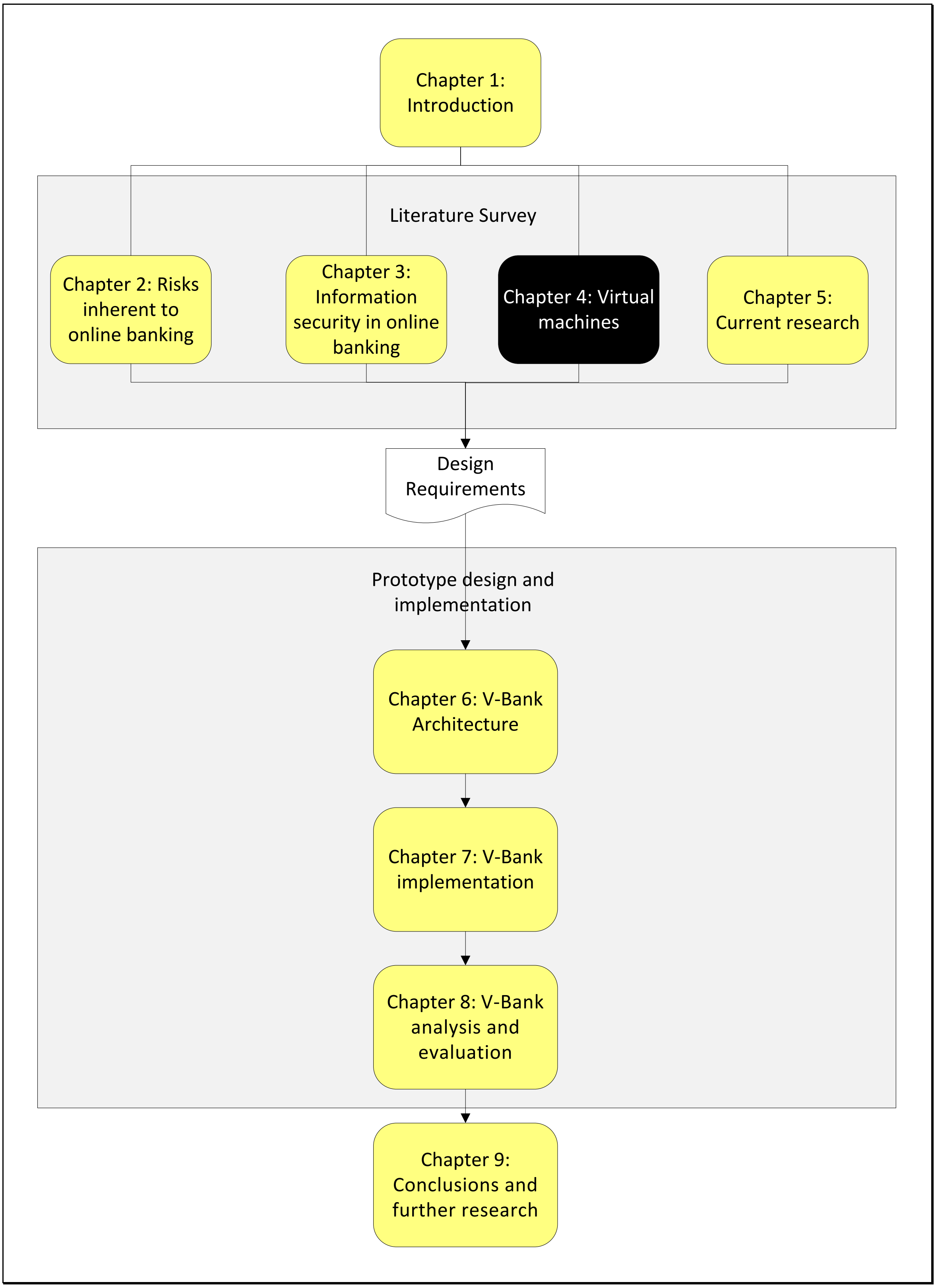
FIGURE 3:5 DESIGN REQUIREMENTS

6 Summary

In this chapter some of the controls used by banks to minimise online banking risks have been evaluated. The controls have been mapped against the ISO 7498-2 security services to see what role the specific control plays in information security.

The case study that was used described the online banking environment from the start where the user switches on his computer, opens the connection to the online banking system, logs in, accesses information and finally pays the third party.

The chapter showed that some controls recommended by banks, like anti-virus and software updates, cannot be controlled which influence the security of the online banking system. In the next chapter virtual machines will described as an introduction that may give banks the ability to control some of the items that are currently not in their control.


Chapter 1: Introduction
Literature Survey
Chapter 2: Risks inherent to online banking
Chapter 3: Information security in online banking
Chapter 4: Virtual machines
Chapter 5: Current research
Design Requirements
Prototype design and implementation
Chapter 6: V-Bank Architecture
Chapter 7: V-Bank implementation
Chapter 8: V-Bank analysis and evaluation
Chapter 9: Conclusions and further research

Chapter 4: VIRTUAL MACHINES

1 Introduction

In the previous chapter the existing security controls used by online banking systems were described. The list of design requirements for a secure online banking system, V-Bank, was expanded to include some of the controls implemented by today's online banking system.

In the previous chapters a number of design requirements were identified for a secure online banking system. Design requirement 2 states that the client operating environment should be isolated on the network and only be allowed to connect to the banking system environment. This chapter will show that one of the key features of virtual machines is its isolation property.

Other design requirements also allude to the fact that certain configuration items such as the Internet browser should be in a specific known state. This chapter also shows that a virtual machine can be configured according to a set configuration and be used to help secure online banking.

A recurring idea that surfaces is that banks cannot control what customers install on their computers. A solution to this problem could be that the bank can setup and give each online banking customer their own computer. This computer could be controlled and owned by the bank, giving them full control over this computer. Giving away a computer to each banking customer is fairly expense, so how can a bank give away a computer, but without the costs associated with the computers?

Virtualising the computer may be a viable option, which allows the bank to still define the configuration of the computer and stay in control of the computer, even though the customer operates the computer.

This chapter is a literature study of virtual machines, which consists of the following sections:

- **Background**. The history of virtual machines are briefly discussed
- **Types of virtual machines**. The two major types of virtual machines are identified and described.
- **Server and desktop virtualisation**. The reasons for virtualising computer differ depending on whether a computer is virtualised for desktop use or server use. Since this dissertation is interested in desktop virtualisation, virtualisation of servers is only briefly discussed. Desktop virtualisation highlights a number of features that is useful for securing the online banking environment.
- **Information security vulnerabilities and virtual machines**. Virtual machines are not without security risks. The vulnerabilities of virtual machines are described to take into account when designing and implementing a solution for secure online banking.
- **Virtualisation options**. There are a number of computer virtualisation implementations available today. A number of the computer virtualisation solutions are described, which gives a background on how different vendors implement virtualisation.
- **List of design requirements**. The chapter concludes again with the new design requirements that are identified during the research into virtual machines.

After the virtual machine types have been described, virtualisation of servers and desktops are discussed. This chapter is an introduction to the various virtualisation options as a possibility for an online banking solution.

2 Background

To minimise ambiguity the following terms are defined for this dissertation:

- Virtual machine: The original term for a virtual machine is an abstraction of a computer's hardware in software. These days it can also refer to a number of other abstraction types, like "Java Virtual Machine" (Rosenblum, 2004). If not stated otherwise the term virtual machine refers to the abstraction of a computer's hardware in software in the dissertation.
- Virtual Machine Monitor (VMM): A VMM is a software abstraction layer that partitions a physical computer into one or more virtual machines (Rosenblum & Garfinkel, 2005). Examples of VMMs are Microsoft's Hyper-V or VMware's ESXi. The VMM is also called a hypervisor.

"Virtualisation is the process of presenting something that is genuine, when in fact it isn't" (Perez, et al., 2008). Virtualisation techniques are used in many areas of information technology. Areas where virtualisation is used include virtual memory, virtual networks, data virtualisation and hardware virtualisation (Rosenblum, 2004).

Hardware virtualisation enables a computer to run more than one virtual machine, each with its own operating system simultaneously. IBM pioneered virtualisation on its VM/370 mainframe system in the 1960s. This enabled the mainframe to be partitioned into multiple systems (Creasy, 1981).

In the 1990s VMware introduced virtualisation to the x86-platform. VMware exported the hardware of the x86 to its own virtualisation layer, also called a virtual machine. This made it possible for a VMware virtual machine to run all the Microsoft operating systems of that era, which included DOS, Windows 3.1, Windows 95, Windows 98, Windows NT, Windows XP and Windows 2000. It also made it possible to run other operating systems like FreeBSD and Linux (Rosenblum, 2004).

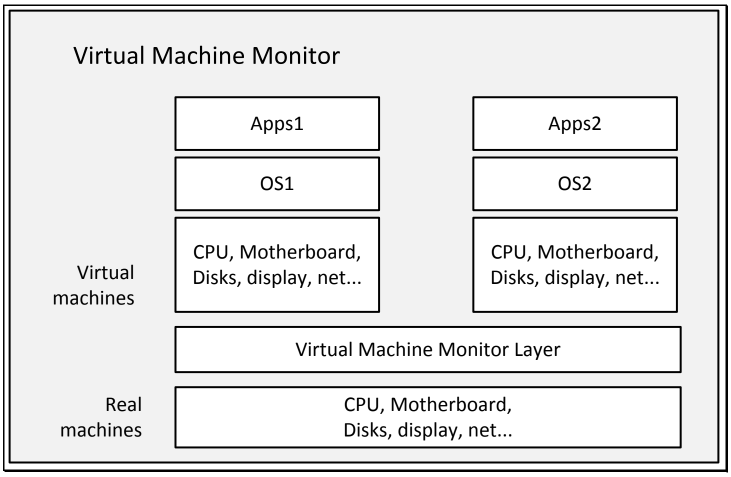

Virtual Machine Monitor
Apps1
Apps2
OS1
OS2
CPU, Motherboard, Disks, display, net...
CPU, Motherboard, Disks, display, net...
Virtual machines
Virtual Machine Monitor Layer
Real machines
CPU, Motherboard, Disks, display, net...

FIGURE 4:1 HARDWARE-LEVEL VIRTUALIZATION

Figure 4:1 shows a hardware-level virtualization layer sitting right on top of the hardware. In this case the virtual machine looks like the underlying hardware (Rosenblum, 2004). Starting at the bottom, the first layer is the host computer's hardware. The VMM gets installed on the physical hardware and allows a number of virtual machines to be installed on the host computer. The virtual machines depicted in Figure 4:1 runs as separate entities, each with its own hardware, operating system and applications.

It was stated that virtual machines can also refer to other abstraction types. To minimise confusion when referring to virtual machines an overview of the different virtual machine types follow.

3 Types of virtual machines

Smith and Nair (2005) identify two categories of virtual machines. They are "process" and "system" virtual machines. We now describe process and system virtual machines briefly.

3.1 Process Virtual Machines

A process virtual machine exists only for the process being executed. A process requires memory allocation and an instruction set and only receives access to hardware through the operating system (Smith & Nair, 2005). An example of a process virtual machine is Sun Microsystem's Java VM architecture and Microsoft's Common Language Infrastructure, which is the basis of the Microsoft .NET platform (Smith & Nair, 2005).

A compiled Java application can run on any processor platform and operating system, as long as a Java virtual machine exists in the operating system (Lindholm & Yellin, 1999)

Figure 4:2 shows a compiled Java application that runs inside a Java virtual machine. The Java application and Java virtual machine is a process virtual machine type. Java applications can easily run on multiple operating systems providing the operating systems have a Java virtual machine.

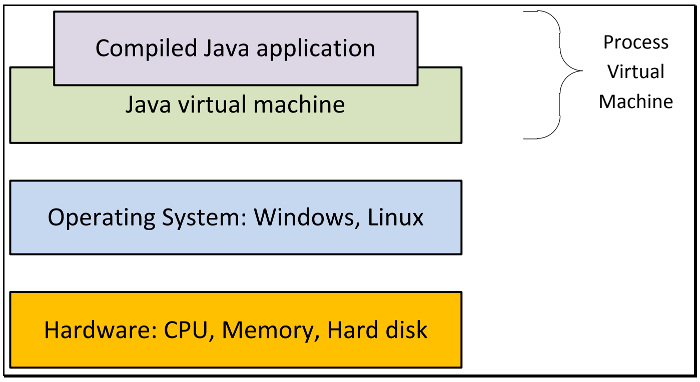

Compiled Java application
Java virtual machine
Process Virtual Machine
Operating System: Windows, Linux
Hardware: CPU, Memory, Hard disk

FIGURE 4:2 JAVA VIRTUAL MACHINE WITH JAVA APPLICATION.

3.2 System Virtual Machines

A system virtual machine on the other hand virtualizes a whole system consisting of operating system, user based processes, processor, networking and other IO components (Smith & Nair, 2005). Figure 4:3 is an example of a system virtual machine. The computer system, in which the system virtual machine is installed, is referred to as the host. This is the bottom most layer in Figure 4:3. The virtual machine running in the system virtual machine is called the guest. In Figure 4:3 the top three layers, showing hardware, operating system and application, is the guest.

An example of a system virtual machine is Oracle VM VirtualBox. The Oracle VM VirtualBox installs inside an existing operating system, like Windows, MacOSX, Linux or Solaris. The installed Oracle VM VirtualBox can host many operating systems, which include, but are not limited to various Windows, Linux and MacOSX server versions (Oracle Corporation, 2011)

This dissertation concentrates on system virtual machines further and not process virtual machines. System virtual machines allow for the virtualisation of hardware. The virtualisation of an actual computer allows a bank to install, configure and control a computer from which a customer can conduct online banking. A system virtual machine also allows the virtual machine to be isolated from the host computer, giving the customer an isolated, protected environment in which he can conduct online banking. Process virtual machines do not virtualise hardware and as such is not applicable to the solution required.

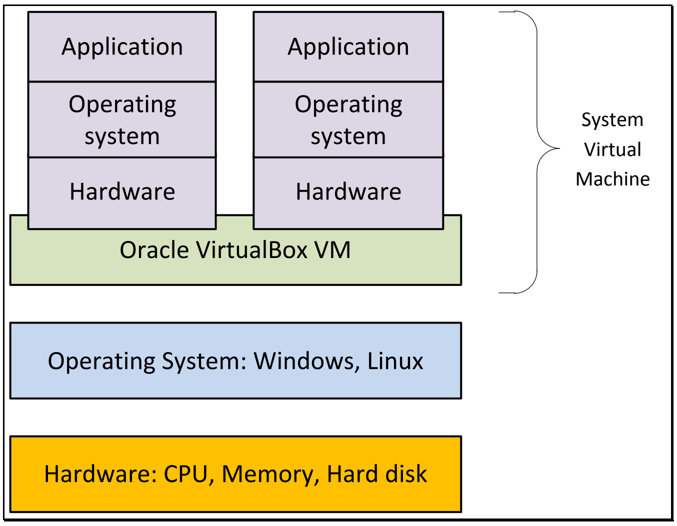


FIGURE 4:3 ORACLE VIRTUALBOX VM IS A SYSTEM VIRTUAL MACHINE

During the introduction it was mentioned that Rosenblum (2004) identified hardware-level virtualization, originally defined by IBM, is when the virtual machine monitor is directly on top of the hardware layer. He then further defined the case where the virtual machine monitor is hosted inside the operating system of the real hardware. Rosenblum (2004) refers to this type of virtualisation as hosted virtualisation. The hosted virtual machine is just another type of system virtual machine, defined by Smith and Nair (2005).

The term system virtual machine also refers to the term hosted virtual machines in the dissertation.

This dissertation focusses on online banking in a home environment. This implies that a home user cannot use a virtual machine on a server, but rather on his home computer. It is important to note that the reasons for using virtualisation on desktops and servers are different. In the next section the reasons for using virtualisation on desktops and servers are discussed.

4 Server and Desktop Virtualisation

The reasons why people implement virtualisation on servers are different than on desktops. In this section the reasons for implementing virtualisation on desktops and servers will be discussed. Once the reasons are understood, it will be simpler to understand why virtualisation can be used for online banking.

Server virtualisation allows for (Agarwal, et al., 2012):

- An increase in server utilisation rates. Multiple underutilised systems can be grouped together to better utilise existing hardware.
- Consolidate multiple operating systems and applications. Applications that require their own dedicated operating systems can utilise many hardware devices if each operating system is installed on its own server hardware. Virtualisation allows the server hardware to be consolidated, but still allows each application to run within its own operating system.
- Deploy new applications in minutes. New virtual machines can be installed and configured in minutes on existing hardware where capacity exists. There is no need to first procure new hardware, physically install the hardware and then install and configure the operating system.

Some of the driving factors for running virtualisation on existing desktop systems are described in the next section, they are (Rosenblum, 2004):

- Application compatibility. With the upgrading to newer or different operating systems, virtualisation allows applications only compatible for a specific operating system to be run on any system.
- Program testing and development. To ensure application compatibility in a developed application, virtualisation allows the developer to test the application on multiple systems, without having to physically have access to these systems.
- Accelerated application deployment. Preconfigured virtual machines with specific applications can be deployed as is and allows the system to be used without complex installation and configuration tasks per client.
- Data isolation. The strong isolation properties of virtual machines can be used to segregate data within one system. This allows a system that need to access a secured network to obtain data without other virtual machines having access to the data retrieved by this system

One of the factors not mentioned by Rosenblum (2004) is that it allows an open source operating system, like Linux, to run on nearly any other platform which is supported by the System virtual machine software. This enables an organisation to develop and configure a solution based on an open source Linux, without having to worry about software licensing to a company like Microsoft when the solution is distributed to potential customers.

In the next two sections the reasons for virtualisation on servers and desktops are discussed. These reasons lead to a better understanding of why virtualisation can be used as a solution for safer online banking.

4.1 Server virtualisation

Virtualising systems on servers means that a server is used to run virtual machines on. These virtual machines provide various services to the organisation.

The reasons for virtualising systems on servers are:

- Increase hardware utilisation (Vogels, 2008)
- Hardware consolidation (Vogels, 2008)
- Ease of deployment of applications (Microsoft, 2012)

The dissertation does not go into details for the reasons for server virtualisation because the to be developed online banking solution lends itself more towards virtualisation on the desktop.

4.2 Desktop virtualisation

In the previous sections the virtualisation of servers were discussed. Virtualisation of systems on desktops are also gaining popularity, but is being accomplished for different reasons than server virtualisation.

The reasons why systems are being virtualised on desktops are:

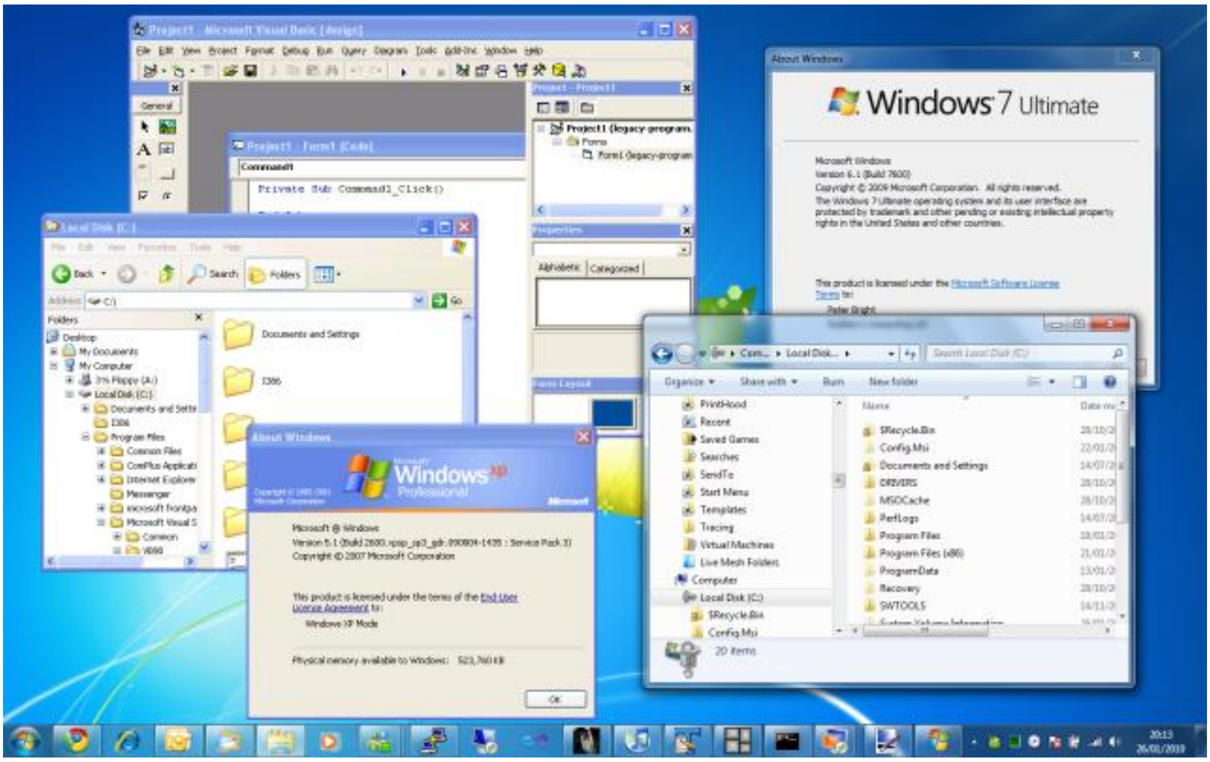
- Application compatibility (Rosenblum, 2004)
- Program testing and development (Rosenblum, 2004)
- Accelerated application deployment (Microsoft Corporation, 2012)
- Data isolation (Rosenblum, 2004)

4.2.1 Application compatibility

One of the features of Windows Vista and Windows 7 Professional is Windows XP Mode. Windows XP Mode is essentially Microsoft Virtual PC that allows Windows XP to run inside Windows Vista or Windows 7. Microsoft released this so that users could still run applications that were not compatible with Windows Vista or Windows 7 inside their latest operating system.

Figure 4:4 shows Windows XP running as virtual machines in Windows 7 (Microsoft Corporation, 2012). The figure shows two “About”-windows. On the left is the window showing the version of Window as being Windows XP and on the right is the window showing Windows 7 as the operating system.

The above feature was made possible through Microsoft Virtual PC. VMware was one of the first vendors to allow virtual machines to be installed on a desktop type operating system like Microsoft Windows XP (Rosenblum, 2004)



About Windows

Microsoft Windows
Version 6.1 (Build 7600)
Copyright © 2009 Microsoft Corporation. All rights reserved.
The Windows 7 Ultimate operating system and its user interface are protected by trademark and other pending or existing intellectual property rights in the United States and other countries.

This product is licensed under the Microsoft Software License Terms to:
Peter Bright

About Windows

Microsoft ® Windows
Version 5.1 (Build 2600.xpsp_sp3_gdr.090804-1435 : Service Pack 3)
Copyright © 2007 Microsoft Corporation

This product is licensed under the terms of the End User License Agreement to:
Windows XP Mode

Physical memory available to Windows: 523,760 KB

FIGURE 4:4 WINDOWS XP MODE IN WINDOWS 7

Virtual machines make it possible to run a Windows application inside Mac OS X or Linux using Virtual Machine software like VMware or Oracle VirtualBox. This focus of this research is in the Windows operating system.

It will be possible for a bank to design a virtual machine with a specific operating system, give it to a user and have the user run it inside his preferred operating system.

4.2.2 Program testing and development

Application developers are sometimes faced with a challenge when they have to ensure that the application that they are developing can run on various versions of operating systems.

Keeping a computer for each operating system or configuration setting is expensive and wastes time. Software engineers can keep a library of virtual machines with different software configurations to develop or test against.

System administrators, responsible for application deployment within an organisation can also keep various virtual machines, one with a previous version, current version and maybe the next version for compatibility testing in the specific environment.

Another feature that makes virtual machines ideal for application development and testing is the ability for a snapshot to be taken of a virtual machine. A snapshot is a point in time reference of a specific system. Any changes to a system from the time a snapshot was taken can easily be undone to ensure the virtual machine is in exactly the same state as it was before (Oracle Corporation, 2011).

Figure 4:5 show that a snapshot was taken of a version 1.0 state and configuration. Changes were made to the application or configuration changing the version to 1.1. Subsequent testing

showed that version 1.1 was not successful and could easily be rolled back to version 1.0 because of the snapshot that was taken of the version 1.0 system and configuration.

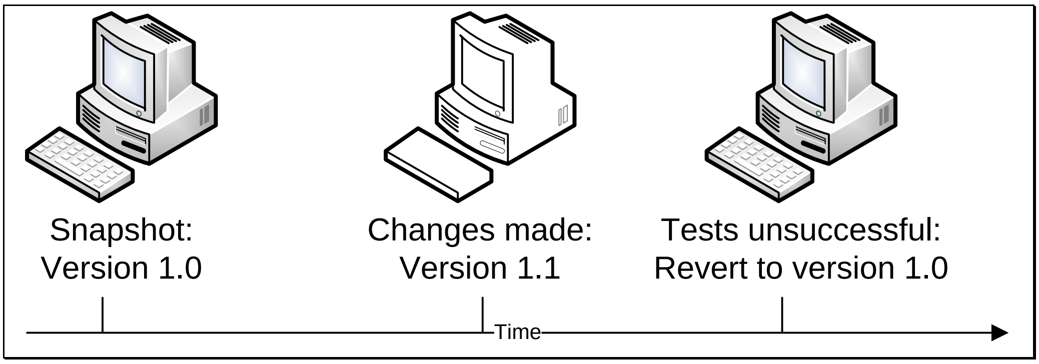


FIGURE 4:5 TESTING ON VIRTUAL MACHINES ALLOWS FOR QUICK RECOVERY TO PREVIOUS WORKING STATE.

The testing of applications in a virtual environment does not have direct influence on the end user of a virtualised online banking application.

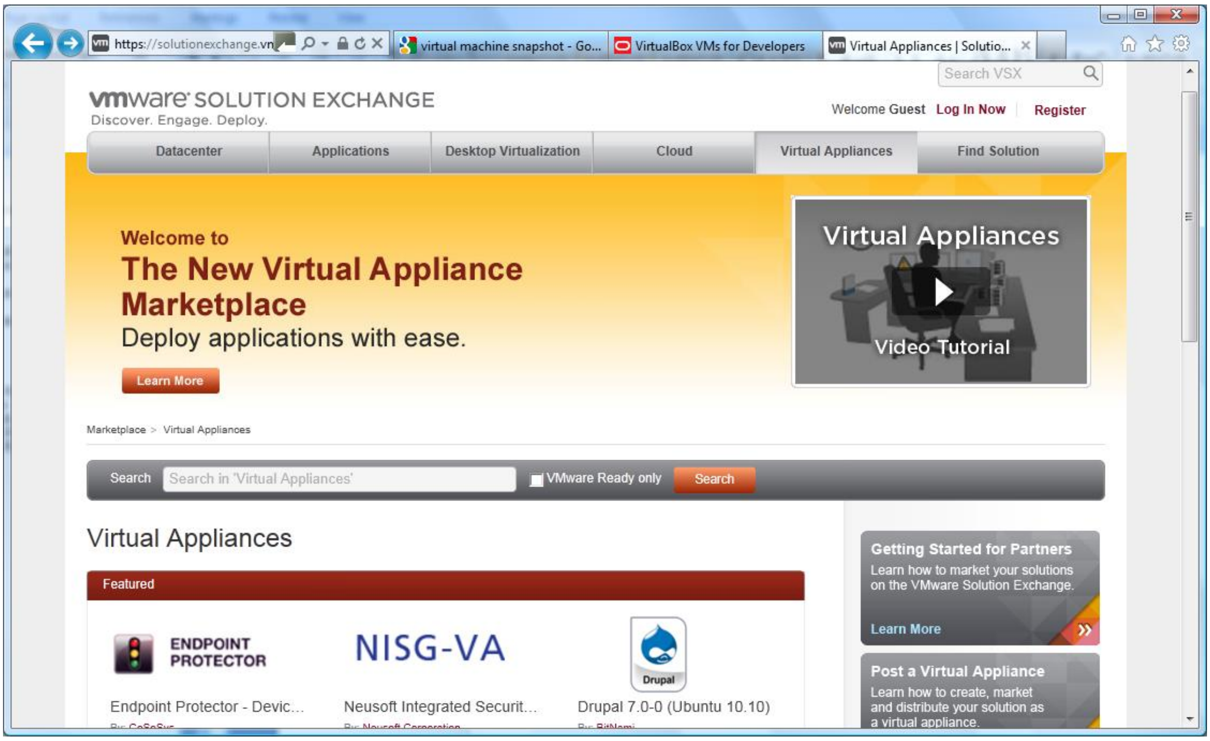
4.2.3 Accelerated application deployment

Oracle has preconfigured virtual machines with specific pieces of software (Oracle Corporation, 2012). VMware also has an area which has, what they call, Virtual Appliances. The virtual appliances and preconfigured virtual machines are virtual machines that have a number of specific applications and operating systems preinstalled and configured on them so that it allows a user to only download and run the specific virtual machine on their own computer (VMWare Inc, 2012).

An example of these preconfigured virtual machines is the “Bloombase Spitfire Message Security Server”. The Bloombase Spitfire Message Security Server is a high-performance mail proxy server that encrypts and signs emails on-the-fly at corporate messaging servers (VMWare Inc, 2012).

Just as third parties publish these preconfigured systems for commercial use, is it possible for IT departments to preinstall and configure a system and deploy to multiple corporate computers without having to go through complex installation and configuration steps per deployed computer.

VMware even made available an online market place for vendors to sell preconfigured virtual machines for use by potential customers. Figure 4:6 shows the virtual appliance marketplace from VMWare (VMWare Inc, 2012). This marketplace allows clients to download a virtual appliance and run it on their existing environment.


https://solutionexchange.vn
virtual machine snapshot - Go...
VirtualBox VMs for Developers
Virtual Appliances | Solutio...
Search VSX
vmware SOLUTION EXCHANGE
Discover. Engage. Deploy.
Welcome Guest Log In Now Register
Datacenter
Applications
Desktop Virtualization
Cloud
Virtual Appliances
Find Solution
Virtual Appliances
Video Tutorial
Learn More
Marketplace > Virtual Appliances
Search
Search in 'Virtual Appliances'
VMware Ready only
Search
Featured
ENDPOINT PROTECTOR
NISG-VA
Drupal
Endpoint Protector - Devic...
Neusoft Integrated Securit...
Drupal 7.0-0 (Ubuntu 10.10)
Getting Started for Partners
Learn More
Post a Virtual Appliance

Welcome to
The New Virtual Appliance Marketplace
Deploy applications with ease.

Virtual Appliances

Learn how to market your solutions on the VMware Solution Exchange.

Learn how to create, market and distribute your solution as a virtual appliance.

FIGURE 4:6 VMWARE'S VIRTUAL APPLIANCE MARKETPLACE

Banks can use the same concept to deploy an updated version of the virtual machine to their online banking customers.

It is also possible for banks to develop an online banking virtual appliance that adheres to a specific configuration, also known as a known configuration. With this known configuration the bank knows exactly how and who connects to the banking system.

The bank can also make the virtual appliance available as a LiveCD (PC Magazine, 2012), or image that can be run. This image can be made read only, so that the user of the appliance cannot change it. This read-only image is also known to be persistent. When a new version of the appliance becomes available the whole persistent image can be updated. Persistent images are in many cases used for demo purposes (PC Magazine, 2012).

4.2.4 Data Isolation

Rosenblum (2004) identifies isolation capabilities as one of the attributes of a virtual machine monitor.

The United States of America’s National Security Agency (NSA), created a system called, NetTop. The NetTop system comprises two virtual machines. One virtual machine is used to access public networks, like the Internet, the other virtual machine is used to access a classified network. The NetTop system is an example where the isolated character of virtual machines is exploited to isolate data (Rosenblum, 2004).

Data isolation is a powerful driver for the use of virtual machines on desktops to ensure integrity and security of data and system. This specific characteristic is used as part of the V-Bank solution to increase security in online banking systems.

Data isolation ensures that the memory space and processes of the system that is used by the client to conduct online banking, be isolated from the rest of the user's computing environment. This isolation puts the online banking environment in a sandbox that is not accessible by software running on the user's operating system.

The reasons for virtualising systems on desktops and servers were discussed and the advantages of desktop virtualisation were described. Desktop virtualisation is an isolated, fully configurable environment for online banking.

Even though virtual machines isolate the virtualised environment and are protected, virtual machines also have vulnerabilities and these vulnerabilities are explained in the next section.

5 Information security vulnerabilities and virtual machines

Virtual machines can have positive security implications to a system. It allows not just users and processes to be isolated, but full operating systems can be isolated (England & Manferdelli, 2006).

There are also some negative security implications such as the security management burden that comes with the extra layers of technology. Some virtualisation systems also allow multiple virtual machines to share information between each other. This can act as an attack vector (Scarfone, et al., 2011)

This section discusses the security implications for using virtual machines in a computing environment. The security implications described are the ability for isolating guest operating systems from each other and the host operating system. It also describes the ability for the VMM to monitor the guest operating system and then lastly the image and snapshot capabilities of the VMM (Scarfone, et al., 2011).

5.1 Guest operating system isolation

The inherent characteristic of virtual machines is that they virtualise a computer, with its hardware, operating system and applications. This may lead to an assumption that, just like physical computers, virtual machines are isolated from each other. This is however not true. This section shows that even though virtual machines seem to be isolated, that there are a number of security considerations to take into account.

Scarfone, et al. (2011) identifies the following items as a security concern with operating system isolation:

- Partitioning. When a VMM assigns physical resources to each guest operating system it is called physical partitioning. When a VMM assigns resources to a guest operating system from a pool of resources sharing the same security characterisation level is it called logical partitioning.
- Sandboxing. When guest operating systems are isolated from each other on the same host, each with its own resources.

- Side-channel attacks. The ability of a guest operating system exploiting the physical properties of the hardware to reveal information about another guest operating system.
- Escape. When a guest operating system tries to gain access to the VMM in order to affect other guest operating systems or host operating system.
- Guest operating tools. In hosted VMM implementations the VMM can install certain tools on the guest operating system, allowing the guest operating system to share information with the host operating system or other guest operating systems.

Each of the security concerns are now be briefly described

5.1.1 Partitioning

The VMM is responsible for assigning physical resources to guest operating systems. These resources are things such a CPU, memory and storage (Smith & Nair, 2005).

The VMM assigns the resources from the host system to the guest operating systems. The resources are partitioned to ensure that one guest operating system cannot encroach on resources assigned to another guest operating system. Resources may be partitioned physically or logically (Scarfone, et al., 2011).

In physical partitioning, the physical resources assigned to a virtual machine are disjoint from other virtual machines. Physical partitioning gives a high degree of isolation to a system. Neither software problems nor hardware failure on one partition can affect the resources on another partition (Smith & Nair, 2005).

In logical partitioning the underlying hardware are time-multiplexed between the different partitions. This allows for better hardware utilisation between systems (Smith & Nair, 2005).

Physical partitioning provides stronger security and reliability than logical partitioning (Scarfone, et al., 2011).

5.1.2 Sandboxing

Partitioning ensures that physical hardware is separate between guest operating systems. VMM also has the ability to control access from the different guest operating systems. The VMM can for instance disable the network in one guest operating system, while enabling the sharing of a file system between two operating systems (Scarfone, et al., 2011).

Isolating each guest operating system and controlling what resources they can access and what access is allowed is also known as sandboxing (Scarfone, et al., 2011).

5.1.3 Side-channel attacks

Side –channel attacks use the physical characteristic of computer equipment to gain information about the software running on the system. This can include things like electro-magnetic emissions or computation time (Green, 2013).

Recent studies have shown that it is very difficult, but theoretically possible, for one virtual machine to gain knowledge, like cryptographic keys belonging to another virtual machine through side-channel attacks (Green, 2013).

Running multiple virtual machines on the same hardware also mitigates traditional side-channel attacks by creating noise (Green, 2013).

5.1.4 Escape

If an attacker can break out of a virtual machine into the VMM, then the attacker can potentially compromise the VMM and gain access to other guest operating systems. When an attacker breaks out of a virtual machine, it is known as escape (Scarfone, et al., 2011).

Wang and Jiang (2010) argues that the VMM is a complex and big structure and cannot always be trusted, which allows virtual machines to exploit vulnerabilities in the VMM and escape.

5.1.5 Guest tools

Many hosted virtualisation solutions allow guest operating systems to use mechanisms called guest tools to exchange information between each other or the host operating system. These guest tools can include the functionality to share files or directories, copy and paste buffers and other resources. These communication mechanisms can act as attack vectors for attackers. Hardware level virtualisation does not have these types of sharing mechanisms (Scarfone, et al., 2011)

Because these guest tools can be used as a potential attack vector the bare minimum guest tools should be enabled. This then expands our online banking design requirements with a new requirement.

Design Requirements

18. The bare minimum guest tools should be enabled in a client-side virtual machine environment.

5.2 Guest operating system monitoring

A VMM is fully aware of the state of a guest operating system running on it. The VMM has the ability to monitor each guest operating system as it runs. This ability is called introspection (Scarfone, et al., 2011).

Introspection allows the VMM to run security applications, such as anti-virus, intrusion detection systems and policy checkers. Unfortunately it also allows attackers to use introspection as a way to gain access to information in the virtual machine (Pieters, et al., 2009)

Introspection is also a problem related to confidentiality of virtual machines. It allows the VMM to see inside the virtual machine (Pieters, et al., 2009)

In an online banking environment this can be problematic as an attacker can gain knowledge and information for authenticating, confidentiality and non-repudiation mechanisms used by the online banking system.

Pieters, et al. (2009) recommends that the VMM only has limited introspection capabilities. This leads to an online banking design requirement.

Design Requirements

19. The VMM used by the online banking system should have limited introspection capabilities

5.3 Image and snapshot management

A feature that can be provided by a VMM is to rollback actions of a virtual machine. This is accomplished by taking a snapshot of the VMM at a specific time. This snapshot is then an image of the virtual machine as it was at that specific date (Pieters, et al., 2009).

The biggest security concern with snapshots is that they contain sensitive data. Snapshots however contains more than just what is normally part of a virtual machine image, it also contains what was in RAM at that specific time (Scarfone, et al., 2011)

In an online banking environment, the client side online banking environment should not have the capability of having snapshots taken. Snapshots of a client-side operating virtual machine image should not be possible. This may allow attackers to evaluate the snapshot and potentially gain access to passwords as they are loaded in RAM.

Design Requirements

20. Snapshots of a client-side operating virtual machine environment should not be possible

A virtual machine is implemented as an image that is loaded by the VMM. Images can be deployed as virtual appliances that can expire. This means the image is read only and cannot change. The image is also normally distributed using read only media. If it is not managed as an appliance then malware on an untrusted host can make changes to the image. (Scarfone, et al., 2011).

Guest operating systems running on host operating system cannot be fully protected. The only way to guarantee security is to bypass the host operating system altogether. This can be accomplished by booting into a removable hypervisor with the guest operating system (Scarfone, et al., 2011).

Virtual machines allow isolation features that increase security of virtual machines. Virtual machines also have their own security concerns that must be taken into account while designing a system that uses virtualisation. In the next section the various virtualisation options are described. An evaluation of the virtualisation options assists with the implementation of our virtual machine based online banking solution.

6 Virtualisation Options

There are many notable virtualisation solutions available. Virtualisation solutions can be broadly grouped into commercial and open source categories.

The virtualisation options that are discussed include:

- OpenVZ (Parallels IP Holdings GmbH, 2013)
- Xen (Xen Project, 2013)
- KVM (an., 2013)
- VirtualBox (Oracle Corporation, 2011)
- VMware (VMware, 2012)
- Microsoft Virtual PC (Microsoft Corporation, 2012)

When evaluating the various virtualisation options the following criteria are looked at:

1. Can it run on Windows, Apple and Linux?
2. Does it support a Linux or Windows guest?
3. What are the licensing implications?
4. Does it support certain host integration features like printing to host printers?

6.1 OpenVZ

OpenVZ is developed as an open source community project, but is supported by Parallels. OpenVZ creates multiple Linux servers on the same server hardware (Kolyshkin, 2012).

OpenVZ can only be used as a Linux host and supports only Linux guests. There is very little overhead for the OpenVZ system and is used to provide virtualised Linux servers.

There is limited host integration features for printing to USB printers connected to the host server.

6.2 Xen

Xen Hypervisor is also an open source community project. Xen is in essence a VMM, but utilise existing Linux distributions as the Domain0 Guest. Figure 4:7 shows the Xen Hypervisor is installed directly on the server hardware (Fraser, 2009).

The Domain0 Guest is a special guest operating system that has direct access to the hardware on the host and controls all the other guest operating systems on the hardware (Fraser, 2009)

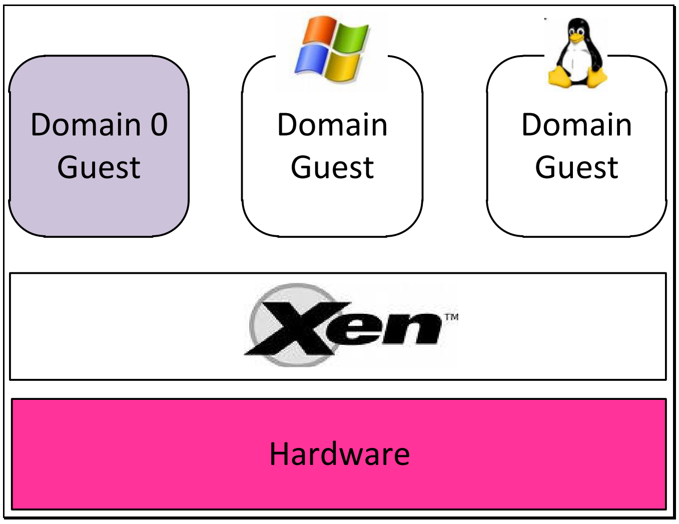


FIGURE 4:7 XEN HYPERVISOR ORGANISATION

Xen was first developed by the University of Cambridge Computer Laboratory, but is currently maintained and developed by the open source community. It is licensed under the GNU General Public License (GPLv2). Xen was at the writing of this dissertation maintained and supported by Citrix (Citrix Systems, Inc, 2012)

Xen supports both Linux and Windows as Guest operating systems, but only Linux as the Domain 0 Guest (Host). There is also limited support for USB printing available for guest operating systems.

6.3 KVM

KVM (Kernel-based Virtual Machine) is currently being developed by the open source community with support from Red Hat, Inc. The KVM project consists of various components each with a different license, but in general the project is an open source license.

KVM works with a Linux host, but can support multiple guest operating systems, including Windows and Linux, but can only run in conjunction with a Linux host.

There is limited printing support for USB printers inside the KVM configuration.

6.4 VirtualBox

Oracle VirtualBox VM was developed initially by Innotek GmbH, which was bought out by SUN and is currently being developed by Oracle. The system is open source and binaries exist for various platforms, including Linux, MacOS and Windows. It also supports many different guests including Linux, Windows, DOS and even OS/2.

There is no printing integration support from the host operating system to the guest. Printers have to be defined per guest, inside the guest operating system.

The base VirtualBox package is distributed under the GNU General Public License V2 (Oracle Corporation, 2012).

6.5 VMware

VMware Workstation or VMware Player is software developed by VMware that enables virtualisation on desktop operating systems. It can be installed on Windows and Linux and supports many guest operating systems, including Linux and Windows.

The Guest additions available in VMware allow for the automatic creation of any printer that has been created and defined in the host operating system. This means that there is no need to recreate the printers inside the guest operating system, if it has already been created in the host operating system.

VMware Player is freely available for testing and personal use. VMware Player may not be used for commercial use (VMware, 2012). VMware Workstation may be used for commercial use, but requires a license from VMware.

6.6 Microsoft Virtual PC

Microsoft has two Virtual PC products: the one is called Windows Virtual PC and the other Virtual PC 2007. Windows Virtual PC is available for Windows 7 and allows Windows XP, Windows Vista and Windows 7 guest operating systems (Microsoft, 2011). Virtual PC 2007 is available for Windows Vista and Windows XP and supports most Microsoft operating systems, but not Linux (Microsoft, 2007)

Microsoft Virtual PC is available without extra costs to Windows 7, Windows Vista and Windows XP users.

Virtual PC automatically creates printers in the Guest operating systems that exists on the host operating system

6.7 Summary

Table 8 shows a summary of the virtualisation options discussed in the above sections.

		Virtual PC	Vmware	VirtualBox	KVM	Xen	OpenVZ
Developed by		Microsoft	VMware	Oracle	Red Hat Inc.	University of Cambridge Computer	Open source community supported by
Compatible with	Host	Windows	Linux, Windows	Linux, Windows, MacOSX	Linux	Linux	Linux
	Guest	Windows	Linux, Windows	Linux, Windows, MacOSX	Linux, Windows	Linux, Windows	Linux
License type		Freely available	Vmware product license	GPLv2	GPLv2	GPLv2	GPL
Host printing integration		Fully Integrated	Fully Integrated	None	Limited	Limited	Limited

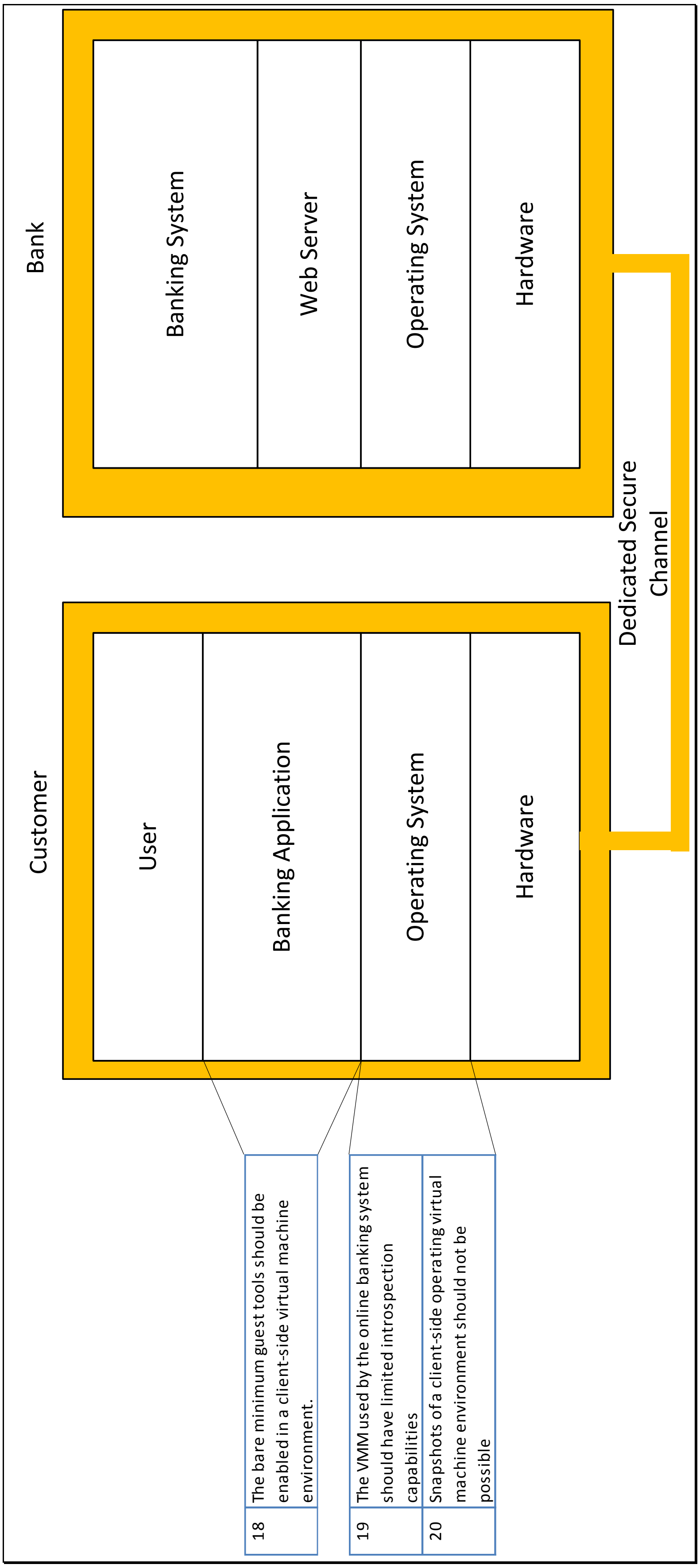
TABLE 8: SUMMARY OF VIRTUALISATION TECHNOLOGIES

7 List of design requirements

While describing the information security concerns in virtual machines three new design requirements were identified. The three new design requirements are requirements number 18 to 20. Figure 4:8 shows the new design requirements that were identified while describing virtual machine security concerns.

Requirement 18 states that the bare minimum guest tools should be enabled in a client-side virtual machine environment. The only guest tool that may be required by an online banking system running as a virtual machine is the printer integration. The printer integration makes is more user friendly when clients would like to print information from their online banking system.

While describing image and snapshot management, it was mentioned that virtual machine images should be managed as virtual machine appliances that expire. This requirement was not specifically highlighted, as it already complies with requirement one, which states that the client operating environment should be persistent. It should not be possible to change any file or configuration in the client operating environment.



18	The bare minimum guest tools should be enabled in a client-side virtual machine environment.
19	The VMM used by the online banking system should have limited introspection capabilities
20	Snapshots of a client-side operating virtual machine environment should not be possible

FIGURE 4:8 DESIGN REQUIREMENTS

8 Conclusion

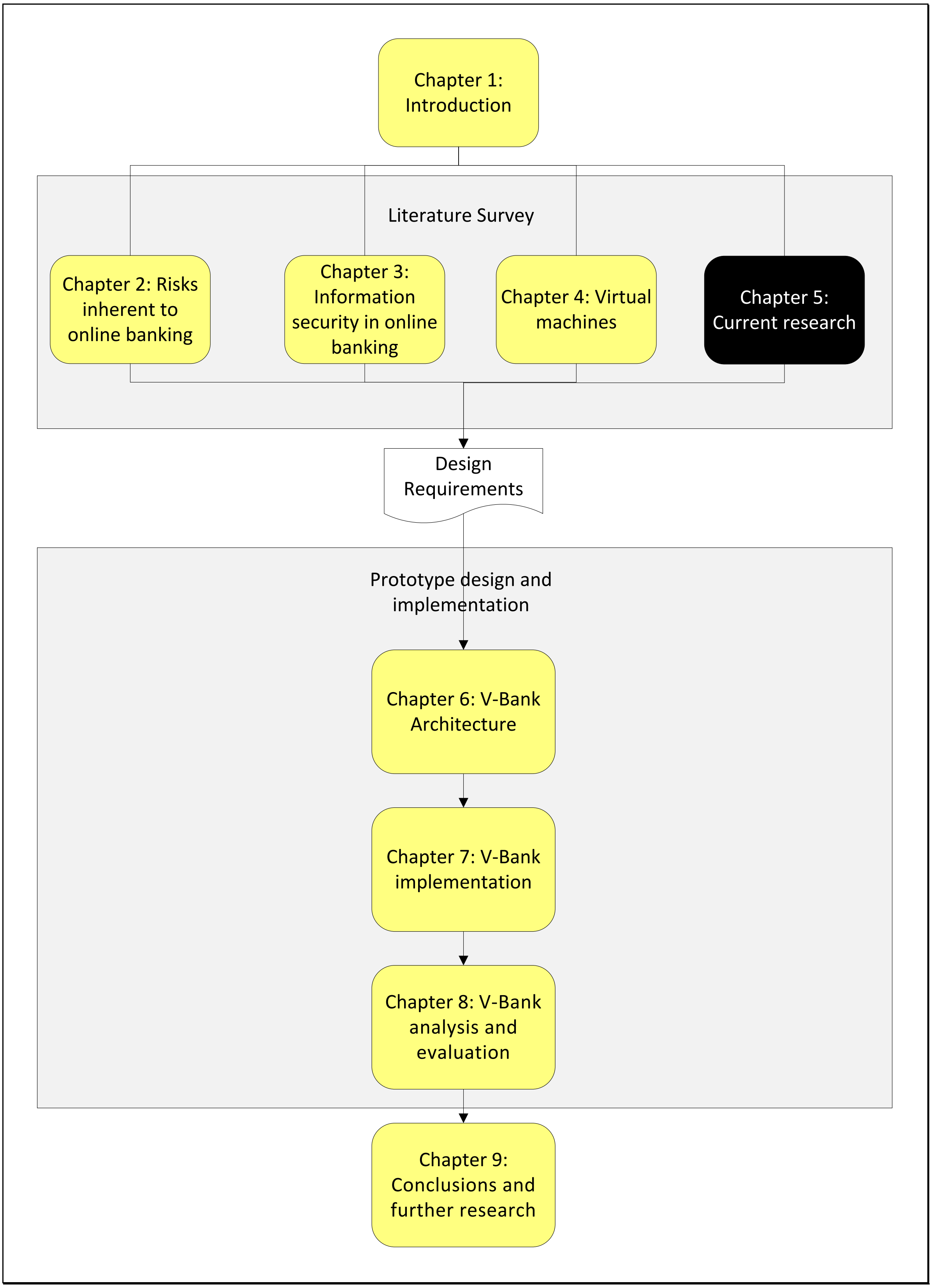
Virtual machines have important properties like isolation and accelerated application deployment that makes ideal solutions for isolating online banking environment from user operating environments.

Part of the requirements for an online banking system is that the banking system should be persistent with a known configuration that is also protected from the rest of the user's operating environment. Virtual machines can be preconfigured, made persistent and given to banking clients to increase the security of an online banking system.

It was shown that virtualisation can increase security of systems, but a number of information security concerns should be taken into account while designing an online banking system that uses virtualisation. Some of the virtualisation options that are available for implementing an online banking system were described. The chapter concluded by showing how the design requirements for an online banking system are affected by the security concerns of virtual machines.

From this chapter we can now also state that one of the design requirements for our prototype system is to make use of virtualisation. In Chapter 7 the implementation of the prototype is discussed. One of the aspects that are discussed in that chapter is the virtualisation solution used by the prototype.

The list of design requirements has now received input by evaluating the risks inherent to online banking. It was further expanded by evaluating the existing information security controls used by banks. The last input to the list was by describing some of the security concerns with virtual machines. The next chapter looks at some of the existing research conducted using virtual machines to increase security for networked systems. This is used as input to further expand the design requirements.


Chapter 1: Introduction
Literature Survey
Chapter 2: Risks inherent to online banking
Chapter 3: Information security in online banking
Chapter 4: Virtual machines
Chapter 5: Current research
Design Requirements
Prototype design and implementation
Chapter 6: V-Bank Architecture
Chapter 7: V-Bank implementation
Chapter 8: V-Bank analysis and evaluation
Chapter 9: Conclusions and further research

Chapter 5: CURRENT RESEARCH

1 Introduction

The idea of using virtualisation to increase the security in an environment is not novel. A number of research papers have been published that evaluate the ability of virtual machines to increase the security of online systems.

This chapter evaluates some of the solutions described in the studies. The solutions are evaluated to identify the advantages and disadvantages of the systems and address them as design requirements for our online banking system.

Four research papers were chosen to be evaluated. The research papers were chosen for the following reasons:

- The solution must make use of virtualisation.
- The solutions must increase the information security of the applied environment.
- The research papers must be published a few years from each other to see how advancements in technology affected the solutions.

An overview of the four research papers chosen for evaluation is described in the next section

2 Background

Cox, Hansen, Gribble and Levy published a paper in 2006 with the title “A safety-oriented platform for Web applications” (Cox, et al., 2006). This paper is the first of the four studies that are evaluated.

The second study that is examined is called “SafeFox: A Safe Lightweight Virtual Browsing Environment” and was written by Wang J., Huang Y. and Ghosh A (Wang, et al., 2010).

The third study is called “The secure virtual computer on your keychain” proposed by Tom Rowan (Rowan, 2008)

The last study is called “A Novel Security Scheme for Online Banking Based on Virtual Machine” (Guan, et al., 2012).

In this section each research topic is evaluated by describing what the problem was that the research topic wanted to solve. The structure for each research topic then describes the following:

- Design goals. Each piece of research wanted to solve a specific problem and defined a number of design goals. These design goals are similar to the design requirements that have been used as a deliverable for the various chapters in this dissertation. The design goals of each research topic must be understood so that it can be evaluated within the boundaries of the defined design goals.
- Key architecture. The architecture for each research topic is described as this can lend valuable insight into the way in which the problem was addressed.

- Implementation. Since the goal of this dissertation is to implement a prototype of a secure online banking system, using virtualisation, is it valuable to understand how other researchers implemented their systems.
- Evaluation. After the research topic has been described, the research is evaluated.

In this chapter the term "LiveCD" is used. A LiveCD is generally referred to as a full operating system that gets installed on a compact disk (CD). The user of a computer then boots the computer with the LiveCD and the LiveCD starts the operating system that is installed on the CD. The system is usually a persistent system in that it is impossible for files to change on the LiveCD, since it is a read-only medium (PC Magazine, 2012).

3 Study 1: A safety-oriented platform for Web applications

The study performed by Cox, et al., 2006 and published as "A safety-oriented platform for Web applications" is evaluated in this section of the dissertation. The evaluation starts by looking at what problem the research addressed. Once the problem is understood the design goals for the proposed solution is described.

Once the design goals are understood, the architecture of the solution is examined after which the implementation of the solution is discussed.

An evaluation of the solution is then conducted highlighting any shortcomings and difficulties. These shortcomings and difficulties are listed to see if it is possible for the V-Bank solution to address the shortcomings.

When the first Mosaic web browser was released in 1993 the web was perceived as a vast repository of interconnected, passive documents. Since then the web has evolved into a collection of independent, dynamic applications.

Web browsers are unfortunately not adequately protected for their new role. Despite attempts to "retrofit" isolation and security, the original roots of a web browser stays evident.

The problem focusses on the fact that web browsers cannot cope with the demands and threats of today's web applications and infrastructure.

With the above in mind Cox, et al. designed the Tahoma browsing architecture.

3.1 Design goals

Cox, et al. defined three key principles to which the Tahoma architecture adheres to. They are:

1. Web applications should not be trusted. Web applications and users should be protected from the various threats that exist on the web. Web applications should be contained within a sandbox environment, to mitigate potential damage.
2. Web browsers should not be trusted. Web browsers can be fairly complex and can even become more complex because of third-party add-ons. This complexity and add-ons is vulnerable against online threats. Browsers should be isolated from the rest of the system to minimise the potential impact of an exploited vulnerability.
3. Users should be able to identify and manage downloaded Web applications. Web applications should be easily identifiable with their servers to the users.

The above design goals gives the user of an implemented Tahoma architecture not only a sandboxed environment for the Internet browser, but also protects Web applications (server-side included) protection from each other.

3.2 Key architecture

Before we discuss the architecture of the Tahoma system, the following terms are defined (Cox, et al., 2006).

Tahoma Term
Web documents: static and active content (e.g. HTML or JavaScript) that a browser fetches from the Web service
Browser: Client side software (e.g. Firefox or Internet Explorer) that interacts with the user to fetch, display and execute Web documents
Virtual machine: a Virtual x86 machine that provides the sandboxed execution environment for an instantiated browser.
Browser interface: A browser executing in a virtual machine; A client has one browser instance per executing Web application
Web application: The union of a client-side browser instance and a remote Web service that cooperate to provide the user with some application function (e.g. , online banking, web mail).

The Tahoma architecture is defined with six key features (Cox, et al., 2006).

1. There is a trusted layer, called the Browser Operating System (BOS) on top of which Browser instances can run.
2. It provides explicit support for Web Applications. A Web Application is defined as both the client side component (Web Browser) and server-side component (Web Services and Web Sites) that makes up a specific Web application. Example: The browser instance that runs any client side JavaScript or Java Apps when connecting to an HTTPS session that is created on the Web server on an Internet banking system.
3. It enforces isolation between Web applications. One Web application cannot spy nor have access to any resources belonging to another Web application executed from the Tahoma system.
4. It supports an enhanced window interface for browser instances. The BOS multiplexes multiple instances onto the physical screen and authenticates the Web application for the users. (This gives the user the assurance that he is not connected to a spoofed web site).
5. It enforces policies, defined by the Web service (from the server). The policies control to which set of web sites the browser can connect to for this specific Web application. The policies are contained in a component called a manifest, which is given to the client when the initial session to the web site is created.
6. It provides resource support to browser instances, such as window management, network communication, bookmarking and execution of Web applications.

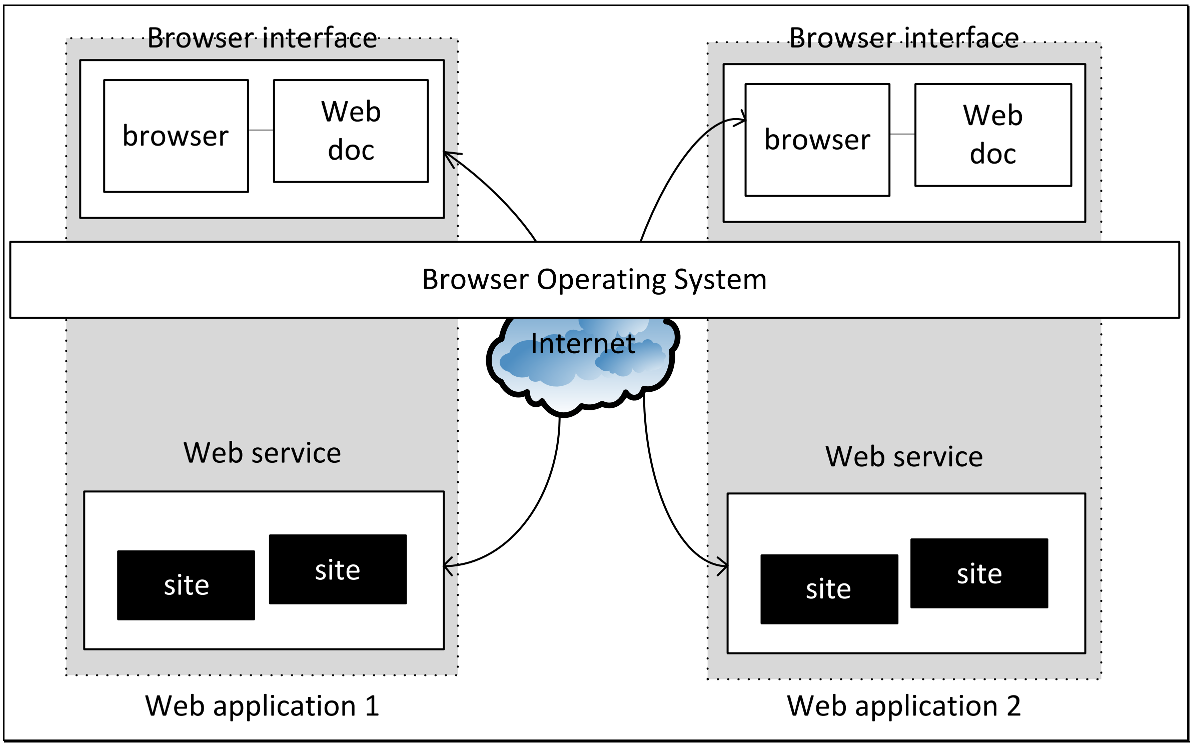


FIGURE 5:1: THE TAHOMA ARCHITECTURE.

Figure 5:1 shows a simplified high-level overview of the Tahoma architecture (Cox, et al., 2006). It shows two Web applications (Web application 1 and Web application 2). Each web application shows that it consists of a Web service.

A Web service may consist of more than one web site.

A Web application also consists of a Browser interface. The Browser interface in turn consists of the actual web browser, and the web document that gets loaded from the web site. The web document may consist of static or dynamic content that partially gets executed on the client. This Browser interface gets executed and given the necessary resource by the trusted Browser Operating System (BOS)

Each Web application is isolated from each other and controlled through a manifest provided by the Web service.

Web documents and manifest information gets transferred between the Browser interface and Web service via the Internet.

3.3 Implementation

The Tahoma system was implemented using Xen Virtual Machine Monitor (VMM). The Xen VMM makes use of Domain0 through DomainN (Where N is variable number greater than 0).

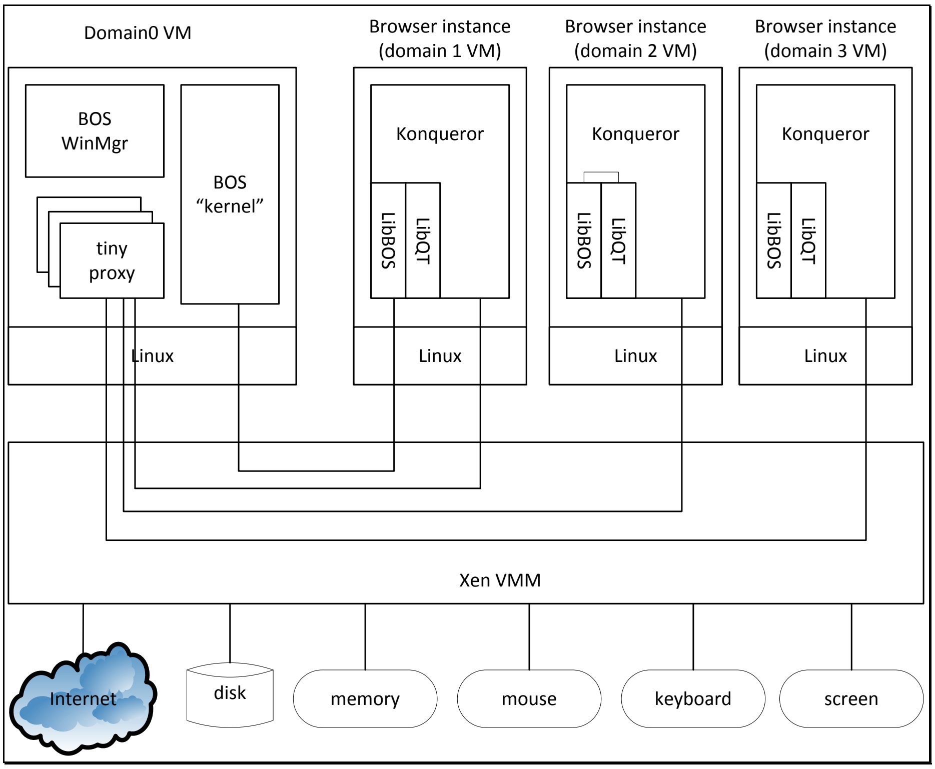


FIGURE 5:2 THE TAHOMA IMPLEMENTATION.

Figure 5:2 is a simplified view of how the Tahoma system is implemented (Cox, et al., 2006). It shows that the XEN VMM isolates browser instances by sandboxing them in virtual machines.

The Browser Operating System (BOS) executes in Domain0 and controls the creation of all the other domains when a browser instance is launched. There are three main BOS processes that execute. They are a reverse proxy, a firewall and a window manager. The reverse proxy and firewall enforce the policies that are received from the Web services. The window manager controls the windows in which the various browsers execute.

Each virtual machine uses Konqueror as the web browser for browsing the web. Inside each virtual machine are two libraries that assist the virtual machine to have access to the BOS. The two libraries are called LibBOS and LibQT. LibBOS gives the virtual machine access to functions that interface with the system functions in the BOS. LibQT gives the virtual machine access to functions that interface with graphical functions in the BOS.

The browsers that are created by the BOS execute in DomainN and do not have direct access to the hardware and must route everything through the BOS before it can get access to the Internet or hardware.

3.4 Evaluation

This section of the dissertation evaluates the solution described by Cox, et al.

The Tahoma system describes an architecture which includes the client and the server. In this way the server can help control access to the client.

A disadvantage of the Tahoma system is that it is still a generalised system, not specifically built for online banking.

Running the system from a LiveCD eases deployment and implementation, since most people do not have a separate computer to only do their banking on. It may even be possible to transfer the whole system to run in a virtual machine that the user can run on their normal desktop operating system environment, but from within a virtual machine.

The major architecture concepts that can be taken from the Tahoma system are:

1. The system includes the server and the client.
2. The server controls the access of the client through approved configuration.
3. Sandboxing of the browser helps to protect the operating system.

Applying these concepts to online banking lead to the following design requirements:

An online banking system should be designed to include not just a managed server environment, but also a managed client environment. The Tahoma system took into account special web services that can only be used by the Tahoma client-side system, but this gives it all the advantages that the Tahoma system was designed for.

Design Requirements

21. An online banking system should be designed to include not just a managed server environment, but also a managed client environment.

The Tahoma server-side environment controls what the client can do on the web site using policies. These policies are transferred to the client through a web service. This concept can be expanded so that it may be possible for the server to evaluate the client configuration. This can be made possible by running various applications pushed from the server on the client. This allows the server to test various configuration settings on the client before the client is granted access.

This requirement however is in line with our existing design requirement eight. Design requirement eight states: An online banking system should only allow connections to and from approved clients and servers. Instead of specifying a new design requirement that might have stated that the online banking system should evaluate the client against approved configuration settings, the existing design requirement eight is used.

The Tahoma system sandboxes each web browser in its own virtual machine. In the case of single purpose system that is designed for only online banking then the system should be sandboxed from the rest of the client operating environment. The sandboxing ensures that the system is isolated from the rest of the client operating environment. A new design requirement is defined.

Design Requirements

22. The client-side online banking system should be sandboxed from the rest of the client operating environment

The next study that dealt with virtualisation is called SafeFox.

4 Study 2: SafeFox: A Safe Lightweight Virtual Browsing Environment

The SafeFox browsing environment was described by Wang, et al. 2010.

In this section we look at the problem that SafeFox wanted to address. We then determine the design goals of the system and see how the architecture supports these goals.

After the architecture has been described the implementation of the system is discussed.

Wang, et al. describe the problem with Internet browsing as the ability of attackers to use the web browser as an attack vector to the operating system. The problem also includes the abilities of web sites to gain access to sensitive web sessions, such as banking sessions, though cross site vulnerabilities.

4.1 Design goals

To help protect the operating system from attacks through the web browser the SafeFox system identifies three design goals that must be fulfilled.

They are (Wang, et al., 2010):

- Host protection
- Session protection
- Low overhead

Host protection is defined as protecting the operating system that hosts the web browser. The web browser is not supposed to be able to affect the host on which it is running.

Session protection is defined as protecting one Internet session from another session. Sessions running on the same host should not be able to affect each other.

The last goal is that the system should not take up significant overhead on the host system.

4.2 Key architecture

The SafeFox architecture uses a customised Linux kernel that acts as the host operating system for its virtual environments (VE). It has a number of Linux daemons that help control the creation and configuration of the virtual environments.

A virtual environment is created each time the user clicks on the Firefox icon. Each virtual environment is a virtual machine with its own resources assigned to it. The resources includes IP address, process namespace and file system.

A browsing session can either be a general browsing session or a secure bookmarked session. A secure bookmarked session is a user defined web site that the user deems as critical. This means that the user can decide that the online banking web site is a secure bookmarked site. The user then creates a secure bookmark specifically for that site. Each time the user visits the site, the site is automatically created in its own virtual environment and never runs from the general virtual environment.

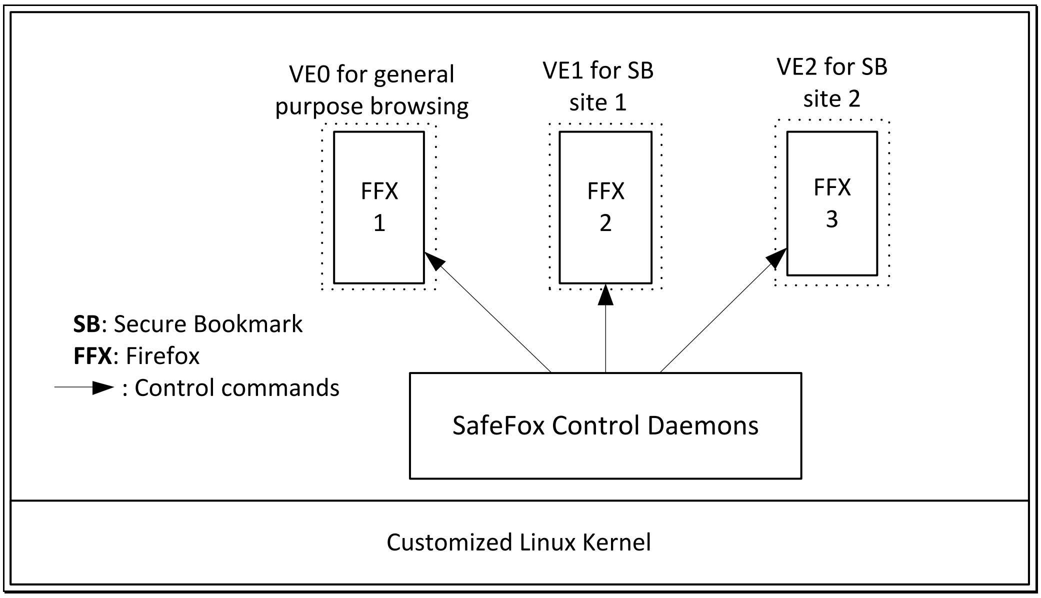


FIGURE 5:3 THE SAFEFOX BROWSING ENVIRONMENT

Figure 5:3 shows the SafeFox environment with a user having established a general purpose browsing session and two secure bookmarked sites (Wang, et al., 2010). The general browsing session is opened in a virtual environment and each secure bookmarked site in its own virtual environment.

4.3 Implementation

The SafeFox implementation uses OpenVZ as its virtualisation solution running in Red Hat Linux.

When the user clicks on the Firebox browser icon, the Firefox browser gets loaded in Master Mode. Master Mode is a browser instance that has full Firefox features, including bookmarking, cookies and stored passwords and so on.

The browser running in Master Mode is used for general web browsing. If the user clicks on the Firefox icon again, it opens a second tab in the existing browser and runs in the same virtual environment as the first browser instance.

In the initial Firefox is a bookmark folder called “Secure IC Bookmarks”. By selecting any of these bookmarks, the SafeFox environment creates a separate virtual environment for the new site and new Firefox web browser. This browser now runs in Secure-bookmarked mode.

The secure-bookmarked mode browser is a locked down instance of Firefox. All top menus are disabled and the navigation bar is read-only. Any metadata from the secure-bookmarked site is stored in a separate folder structure and any traffic to and from the web site is redirected to the new Firefox virtual environment. It should be noted that the secured bookmark site, does not stop the user from clicking inside the current site and going to a new site, thus leaving the secure-bookmarked site, but still running in the virtual environment created for the secure-bookmarked site.

Closing a secure-bookmarked browser does not destroy the metadata that may have been created when the virtual environment was created or downloaded through the secure-bookmarked session. The next time the user initiates a session to the secure-bookmarked site the site reuses the already created resources.

The last mode in which the browser can run is called Private mode. Private mode is initiated by clicking on a separate icon on the desktop called "Private Firefox". Private mode is essentially the same as secure-bookmarked mode, except that it destroys any resources created for it during the browsing session when the user closes the session.

4.4 Evaluation

SafeFox gives an architecture which does not require any changes to online banking systems or web services. It focusses on the web browser and creates an architecture which does not take up significant overhead.

Even though SafeFox does not fulfil all the online banking design requirements, is it still a more secure environment for conducting online banking than from a normal computer.

SafeFox can be implemented as a virtual machine running on Windows, making it possible for banks to deploy the system to their customers, without having to change anything on the online banking system.

The biggest disadvantage of the SafeFox system is that it was never designed for a secure online banking system. It thus does not specify any requirements for the server and allows the server to still accept connections from any type of client, whether it is a SafeFox client or not. The V-Bank system must incorporate an architecture that addresses both the client and the server and the communication channel in between. This has also been stated in Chapter 5:3.4 as design requirement 21.

5 Study 3: The secure virtual computer on your keychain

The stealing of an employee's laptop is a problem for IT directors and security officers. Apart from the fact that there are costs involved in the hardware, the biggest concern is the loss of the data stored on the hard drive and VPN access credentials that may fall into the wrong hands (Rowan, 2008).

To minimise the risk of stolen data and credentials in the event of stolen laptops, Rowan suggests that it is possible to create a computer on a universal serial bus (USB) storage device that can be started in a virtual environment hosted on a borrowed or paid-for host computer.

In this section we describe the design goals of the proposed solution. After the design goals have been identified, the architecture of the solution is described after which the implementation of the solution is described. This gives the reader a clear understanding of the solution.

The solutions are then evaluated in the case of online banking. In the general comments section the architectural components is evaluated to see which components can be adapted as a requirement for our online banking system.

5.1 Design goals

The following design goals are identified in the research paper (Rowan, 2008):

- A laptop alternative
- Security and portability
- Agile workforce
- Access control

The primary design goal for the research was to give the mobile workforce an alternative to using laptops and storing sensitive data on the laptop. The research showed that it may be possible to use a USB storage device to store a complete operating system that can be carried around and used, instead of owning a laptop.

The USB storage device would be used on a host computer. The USB device would start a virtual machine on the host computer. This means that the user will be able to work on any host computer, be it his home computer, or another computer, without leaving any trace of corporate activities. The USB device is also small and can be carried easily by user.

The third design goal was to address the problem faced by corporations that allow many of their employees not to have to come into the office. This gives the corporations options to temporarily use people, give them the virtual machine they would require for the duration of their work and use it on their own computer at home.

The last design goal was to have the network of the virtual machine protect itself from the underlying operating system.

5.2 Key architecture

Rowan (2008) does not give a very clear explanation of the architecture utilised by the research. Figure 5:4 shows the components mentioned in the research article.

The area on the left of Figure 5:4 shows the secure virtual computer, running a virtual machine on a host operating system. The area on the right depicts the corporate network. The corporate network is access control protected using network access control policies and virtual private network access control.

The line combining the virtual machine on the host computer with the corporate network shows that the virtual machine accesses the corporate network through a virtual private network. Before the virtual private network is established network access control policies at the corporate network interrogates the virtual machine hardware and software and determine whether access is granted.

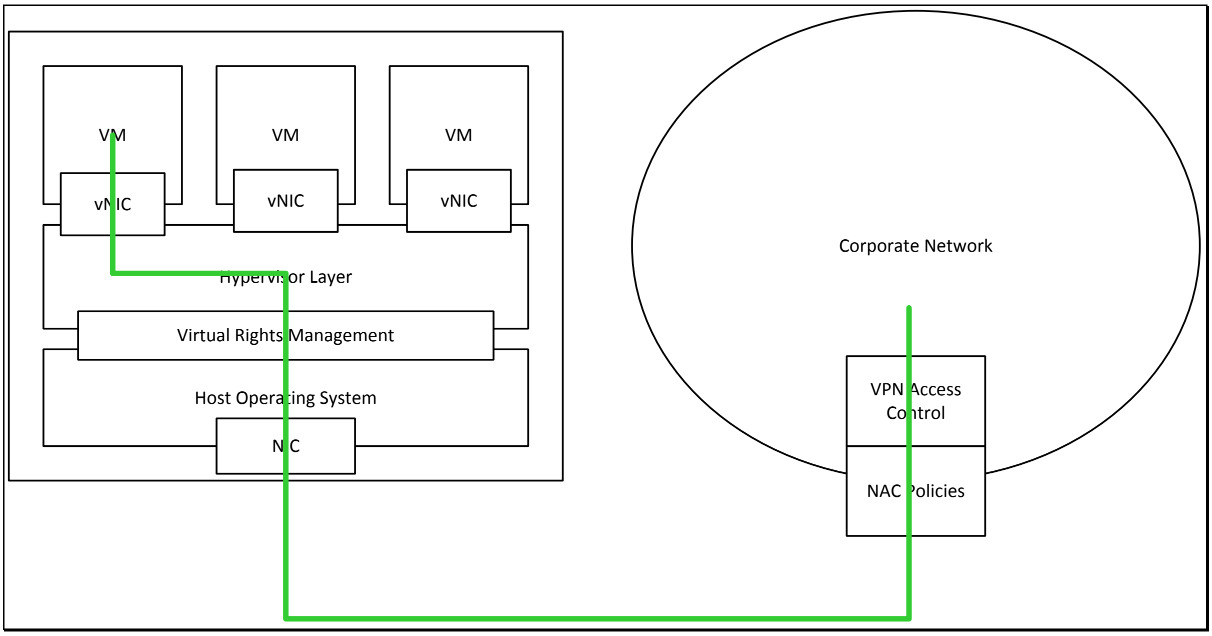


FIGURE 5:4 SECURE VIRTUAL COMPUTER ARCHITECTURE

On the left hand side of Figure 5:4, running, as a component in the hypervisor layer, is the virtual rights management (VRM) (VMWare Inc., 2005) component. The VRM component controls what can be done with the virtual machine. It has the ability to ensure that the virtual machine gets executed from a specific USB memory stick. It ensures copy protection of the virtual machine. It has the ability to password protect the virtual machine startup and can expire virtual machine images (VMWare Inc., 2012).

5.3 Implementation

In the research article, Rowan (2008) does not mention any prototype that was implemented. The implementation described in this section is recommendations from the research.

The secure virtual computer on your keychain is implemented as a hosted virtual machine, running Windows XP. The Windows XP image is configured with limited hardware support. This cuts down on the operating system image and boot-up time.

The hypervisor must be installed on the host operating system, allowing the hypervisor access to run in processor level ring zero, giving the hypervisor direct access to the hardware on the operating system.

The image of the virtual machine is stored on a USB storage device. The USB storage device is a high quality fast storage device. This helps to increase the speed of the virtual machine.

The USB storage device may also include hardware encryption, tamper resistance features, water proofing and tough metal cases (Rowan, 2008).

The VRM may be implemented using VMware ACE (VMWare Inc., 2012).

The VPN are not specified and any VPN solutions can be used to ensure network privacy.

5.4 Evaluation

The secure virtual computer on your keychain was researched specifically to solve the problem of employee laptops getting stolen. The research did not set out to solve the problem of ensuring a secure online banking environment using virtual machines.

The problem that the research article addressed was primarily to solve the problem where laptops get stolen thereby putting the corporation whose data is also stolen at risk. In this section the solution is evaluated to determine whether it can be used in our safe online banking environment. This is done using the defined design requirements that has been created in the dissertation. Each design requirement is listed and measured against the proposed solution from Rowan.

Despite the fact that the research did not concentrate on online banking, it fulfilled nearly all the online banking design requirements.

The disadvantage of the research is that it did not describe the implementation of a prototype and also did not do a critical evaluation of the proposed solution.

The solution described in the research is based very closely on VMware ACE (VMWare Inc., 2012).

The biggest disadvantage of the solution is the requirement of using Microsoft Windows. This requires that every banking customer must be issued with an operating system license, which in turn may make the solution too expensive.

An advantage of the system is the VRM, which ensures the validity of the virtual machine and the access rights of the host system in the virtual machine.

6 Study 4: A Novel Security Scheme for Online Banking Based on Virtual Machine

Guang, et al. (2012) describes a system they call Domain Online Banking. The system uses virtualisation on the client computer to isolate the online banking system from the user's operating system applications.

This section starts by describing the design goals of the solution. It then describes the architecture and then how it was implemented. The section concludes with a brief evaluation of the solution.

6.1 Design goals

The Domain Online Banking solution was designed for the following requirements (Guan, et al., 2012):

- Any compromised component in the user's operating system or applications may not gain access to the memory address space of the Domain Online Banking system
- Sensitive information from the user may not be leaked to third parties.

- The user cannot access fraudulent web-sites that look like the authorised online-banking web site.

The research also specifically wanted to make use of virtualisation as a possible solution and protect the online banking environment from the following attacks

- Phishing scams
- Pharming
- Malicious software attacks
- Man-in-the-middle
- Bank keylogger

The next section describes the architecture that enables the design goals.

6.2 Key architecture

Figure 5:5 shows the architecture that is used for the Domain Online Banking system (Guan, et al., 2012).

The architecture consists of primarily two components, the client computer, on the left of the figure, and the banking server on the right. The client computer consists of two items (Guan, et al., 2012):

- User's computer. This consists of the hardware, operating system and applications on the user's computer
- Online banking smart USB-key. The smart USB-key, depicted in the dashed box, consists of the Xen Hypervisor, the Linux Dom0 restricted operating system and the Online Banking DomU virtual machine.

Communication between the Online Banking DomU virtual machine and banking server is encrypted using a SSL\TLS channel.

The Xen hypervisor consists of a vTPM module that hashes the Online Banking DomU virtual machines image and is compared with the hash of the virtual machine stored on the vTPM database on the banking server. This is indicated with a 'b' in the figure. The vTPM module effectively ensures the integrity of the Online Banking DomU virtual machine.

The Online Banking DomU virtual machine makes use of Virtualisation Technology for Directed I\O that ensures that the virtual machine can get exclusive use of certain hardware PCI devices, like the keyboard and screen. This is indicated with an 'a' in the figure.

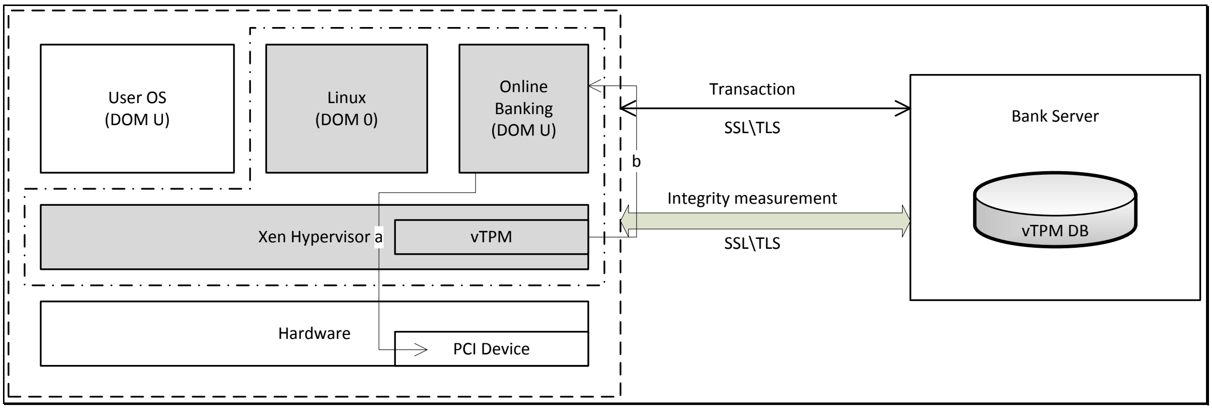


FIGURE 5:5 DOMAIN ONLINE BANKING ARCHITECTURE

6.3 Implementation

The online banking smart USB-key gets pre-burned with the user's private key, certificate, Xen Hypervisor and Online Banking DomU virtual machine. The URL of the banking site is pre-configured in the Online Banking DomU virtual machine, ensuring the virtual machine can only open a connection to the online banking site.

The Online Banking DomU virtual machine is implemented as Windows XP, with the browser defined as the shell of the operating system. This means that when the virtual machine starts, the web browser opens and automatically makes a connection to the online banking web site.

When a user receives the online banking smart USB-key from the bank, the user boots his computer from the smart USB-key. The Xen Hypervisor starts and automatically detect the user's operating system, which starts as a DomU virtual machine.

It also starts the Online Banking DomU virtual machine, but labels the browser window with [DOBank], so that the user knows that he is using the Online Banking DomU virtual machine to access the online banking web site.

Whenever the user uses the Online Banking DomU virtual machine the virtual machine gains exclusive rights to the keyboard, video graphics adapter (vga) and mouse.

6.4 Evaluation

The Domain Online Banking system stops the five major attacks mentioned in the design goals. The bank server will only accept connections from authorised clients whose integrity has been checked. Even if an attacker gains knowledge regarding the user's login details, the attacker will not be able to access the online banking system, without the user's online banking smart USB-key, which stores a copy of the user's private key.

Software keyloggers are not able to capture information from the Online Banking DomU virtual machine, because the Online Banking DomU virtual machine has exclusive rights to the keyboard.

The Online Banking DomU virtual machine can only navigate to the banking web site, which means that it minimises the man-in-the-middle attack.

If the banking server determines that the integrity of the Online Banking DomU virtual machine is not intact, then it can automatically download an approved Online Banking DomU virtual machine image to the client.

The Domain Online Banking system treats the online banking system as a combination of both the client and the server, the same as the Tahoma system did. The client-side banking environment is totally isolated from the rest of the operating environment, keeping it safe from an infected user operating system.

There may be some implementation challenges with the compatibility of the Xen hypervisor and the different hardware configurations that may exist on the client computers. If such a compatibility issue exist then a normal user may not be able to use the system.

7 Comparative evaluation

The four studies mentioned in the dissertation all use virtualisation to enhance the security of web services. The latest study, Domain Online Banking system, is probably the closest to fulfilling all the design requirements that have been defined in the dissertation so far. It also enhances the latest design requirements that were identified in Chapter 5:3.

Table 9 lists the various design requirements and mentions whether the solution for the study fulfils the requirements, is not applicable or does not fulfil.

Requirement number	Requirements	Study 1: A safety-oriented platform for Web applications	Study 2: SafeFox: A Safe Lightweight Virtual Browsing Environment	Study 3: The secure virtual computer on your keychain	Study 4: A Novel Security Scheme for Online Banking Based on Virtual Machine
1	The client operating environment should be persistent. It should not be possible to change any file or configuration in the client operating environment	✓	✓	X	✓
2	The client operating environment should be isolated on the network and only be allowed to connect to the banking system environment.	✓	X	✓	✓
3	It should not be possible to modify the Internet browser used by an online banking system.	✓	✓	X	✓

4	Online banking systems should not allow the Internet Browser from opening multiple pages and the transaction component in an online banking system should re-authenticate the user, before committing important transactions.	✓	✓	X	✓
5	An online banking system should evaluate any input and allow only approved syntax in input fields.	N/A	N/A	N/A	N/A
6	The online banking system should not contain any IFrames, nor should it reference code from any site except its own.	✓	N/A	N/A	N/A
7	Name resolution should not affect the connection inside an online banking system	X	X	X	✓
8	An online banking system should only allow connections to and from approved clients and servers.	X	X	✓	✓
9	A keylogger should not be allowed inside an online banking system or should minimize the effect it may have if present.	X	X	✓	✓
10	An online banking environment should minimise the chances of an attacker using stolen information to gain access to a customer's online banking profile.	X	X	✓	✓
11	Online banking systems must have the ability to be updated when software vulnerabilities are detected.	✓	✓	✓	✓
12	Online banking systems must ensure TLS 1.2 channel security	✓	N/A	✓	✓
13	Online banking systems must have a visual trust anchor.	✓	N/A	N/A	✓
14	Online banking systems must properly authenticate users before allowing access.	✓	N/A	✓	✓
15	Online banking systems must implement a RBAC policy to ensure access to user information is controlled.	N/A	N/A	✓	N/A
16	Online banking systems must implement a layered security program.	N/A	N/A	N/A	N/A
17	Online banking system should implement information security controls in all five ISO 7498-2 security service categories.	N/A	N/A	✓	✓
18	The bare minimum guest tools should be enabled in a client-side virtual machine environment.	✓	✓	N/A	✓
19	The VMM used by the online banking system should have limited introspection capabilities	✓	✓	X	✓

20	Snapshots of a client-side operating virtual machine environment should not be possible	✓	✓	N/A	✓

TABLE 9: COMPARITIVE EVALUATION

From the table we can deduce that systems that are designed for not just the client but include specific controls on the server, like study 1 and study 4 provides a safe environment for online banking. In these two studies the server helps with allowing clients certain permissions and rights on the server. The server specifies a policy which has to be adhered to by the client.

All the solutions have the ability to be updated, which is important in an ever changing security landscape where new vulnerabilities are constantly being identified.

Requirement six is only fulfilled by the first study. The Tahoma system controls exactly how the web service is configured on the server. It prohibits one site displaying content from another site. The other studies did not define this as a requirement.

Study 3 has the VMware ACE policies that help control access to the server. These policies are defined using roles from the server. The other studies did not specify role based access control as part of the scope.

The only study that can cope with name resolution problems, as defined in requirement seven, is the study four, which controls where the client connects to.

Requirement five is not applicable to any of the studies as it is generally assumed that user input should be evaluated on the web service.

Requirement sixteen states that online banking systems must implement a layered security program. The research articles concentrates mostly on the client and communication areas of the online banking environment. The number authentication and authorisation steps necessary for the online banking system are not specified. The assumption from the research is that the server system is adequately protected and secured.

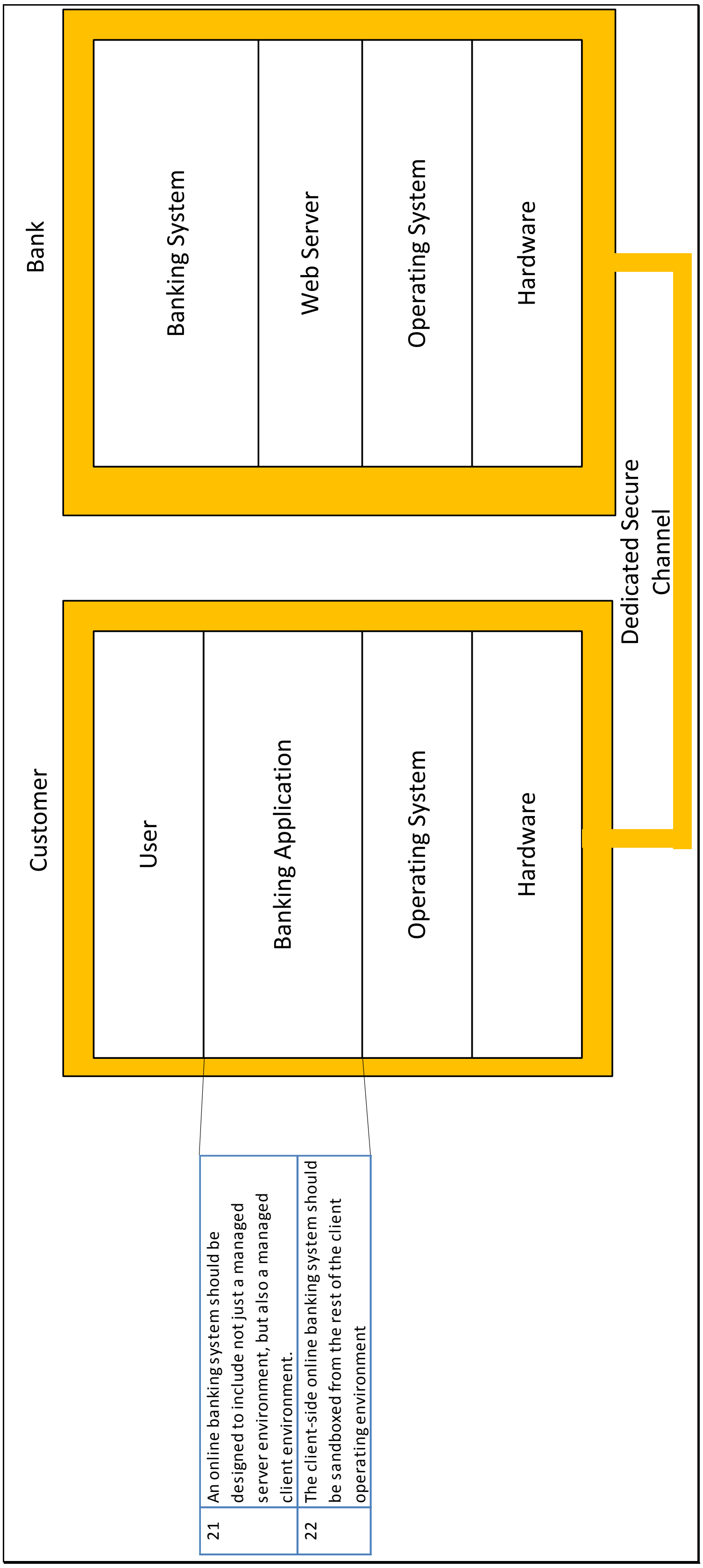
The table shows that the studies do not fulfil or is not applicable to a requirement if it did not include that specific design requirement as a requirement for the study. This highlights the importance of defining a list of proper design requirements before designing a secure online banking system.

It is clear that as technology progressed, that it became easier to enable secure, isolated online banking sessions.

8 List of design requirements

In the evaluation of the Tahoma system and the Domain Online Banking system, two new design requirements were identified. The two new requirements are added to the other design requirements identified in the dissertation, but under a category called, current research.

Figure 5:6 shows the new design requirements identified in this chapter. Both requirements apply to the banking application on the client.



21	An online banking system should be designed to include not just a managed server environment, but also a managed client environment.
22	The client-side online banking system should be sandboxed from the rest of the client operating environment

FIGURE 5:6 DESIGN REQUIREMENTS

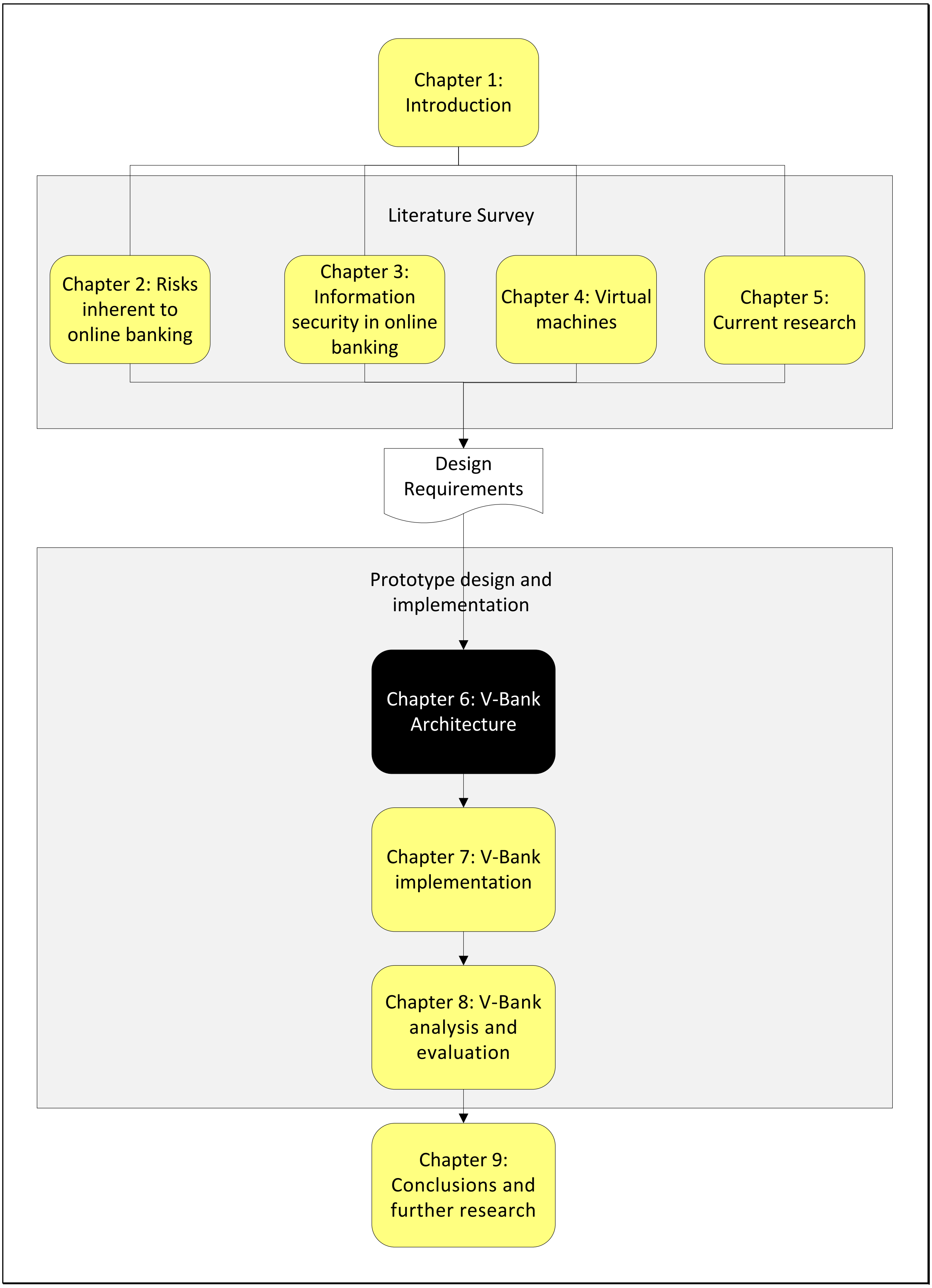
9 Conclusion

This chapter looked at some of the current research that utilised virtualisation to enable a safer online browsing experience. The research succeeded in their design goals and in part fulfilled the requirements for a secure online banking system. The design goals of a secure online banking system were extended to include some of the principles described in the research.

Virtualisation technologies are continually developed and enhanced. These developments enable solutions that increase the security of online banking sessions, as virtualisation technologies evolve better and more secure methods become available for online banking environments.

Chapters two to five produced a list of design requirements for a secure online banking system. This list of design requirements act as input in designing our V-Bank prototype.

In the next chapter the architecture of the V-Bank system is described. The architecture is defined to fulfil the requirements that are defined in this research.


Chapter 1: Introduction
Literature Survey
Chapter 2: Risks inherent to online banking
Chapter 3: Information security in online banking
Chapter 4: Virtual machines
Chapter 5: Current research
Design Requirements
Prototype design and implementation
Chapter 6: V-Bank Architecture
Chapter 7: V-Bank implementation
Chapter 8: V-Bank analysis and evaluation
Chapter 9: Conclusions and further research

Chapter 6: V-BANK – ARCHITECTURE

1 Introduction

In the previous chapter current research that utilised virtualisation was evaluated. Each research paper has a specific architecture that supports the design goals for that specific solution.

This chapter describes the architecture for the V-Bank solution specifically pertaining to design specifications that has been developed in the dissertation.

The next chapter looks at how the architecture was implemented in the V-Bank solution.

This chapter starts with describing the design requirements that are identified in the dissertation. Once the design requirements have been described the overall architecture of the V-Bank solution is described.

The chapter concludes with an overview of the architecture.

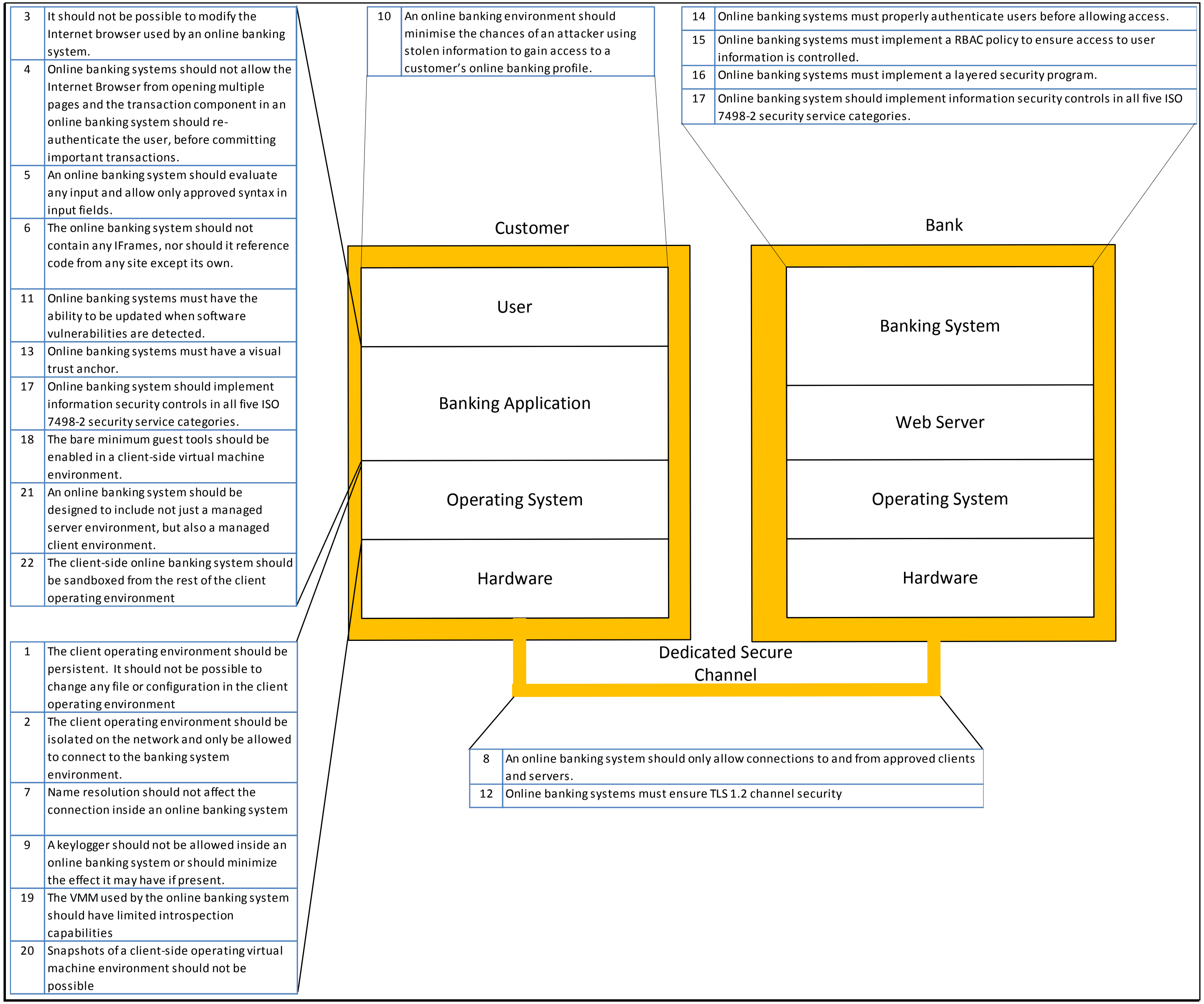
2 Design requirements

The architecture has been designed to fulfil the design requirements that have been defined in the dissertation.

The design requirements are grouped according to generally accepted architecture of a computing environment.

Figure 6:1 shows the computing architecture with the design requirements. The design requirements are grouped under the following architectural areas:

- User
- Banking application
- Operating system
- Hardware
- Dedicated secure channel
- Banking system



3	It should not be possible to modify the Internet browser used by an online banking system.
4	Online banking systems should not allow the Internet Browser from opening multiple pages and the transaction component in an online banking system should re-authenticate the user, before committing important transactions.
5	An online banking system should evaluate any input and allow only approved syntax in input fields.
6	The online banking system should not contain any IFrames, nor should it reference code from any site except its own.
11	Online banking systems must have the ability to be updated when software vulnerabilities are detected.
13	Online banking systems must have a visual trust anchor.
17	Online banking system should implement information security controls in all five ISO 7498-2 security service categories.
18	The bare minimum guest tools should be enabled in a client-side virtual machine environment.
21	An online banking system should be designed to include not just a managed server environment, but also a managed client environment.
22	The client-side online banking system should be sandboxed from the rest of the client operating environment

1	The client operating environment should be persistent. It should not be possible to change any file or configuration in the client operating environment
2	The client operating environment should be isolated on the network and only be allowed to connect to the banking system environment.
7	Name resolution should not affect the connection inside an online banking system
9	A keylogger should not be allowed inside an online banking system or should minimize the effect it may have if present.
19	The VMM used by the online banking system should have limited introspection capabilities
20	Snapshots of a client-side operating virtual machine environment should not be possible

10	An online banking environment should minimise the chances of an attacker using stolen information to gain access to a customer's online banking profile.

14	Online banking systems must properly authenticate users before allowing access.
15	Online banking systems must implement a RBAC policy to ensure access to user information is controlled.
16	Online banking systems must implement a layered security program.
17	Online banking system should implement information security controls in all five ISO 7498-2 security service categories.

8	An online banking system should only allow connections to and from approved clients and servers.
12	Online banking systems must ensure TLS 1.2 channel security

FIGURE 6:1 DESIGN REQUIREMENTS

3 Overall Architecture

Figure 6:2 shows the architecture of the V-Bank solution. The architecture consists of the user's home computer and the V-Bank system. The user's home computer consists of the computer's hardware, operating system and applications in the host operating system. The host operating system and hardware are the white areas in the figure on the left.

The V-Bank solution consists of:

- A Virtual machine monitor. The virtual machine monitor is installed in the host operating system and is situated below the trusted banking operating system. The virtual machine monitor has diagonal shading in the figure.
- Trusted banking operating system (TBOS). The TBOS runs as a virtual machine in the virtual machine monitor and contains the integrity check (IC) module, which is explained later in the chapter. The TBOS also has diagonal shading.
- Banking system. The banking system is located on the bank's servers and has an integrity check (IC) database. The whole integrity check module and database is explained later in the chapter.

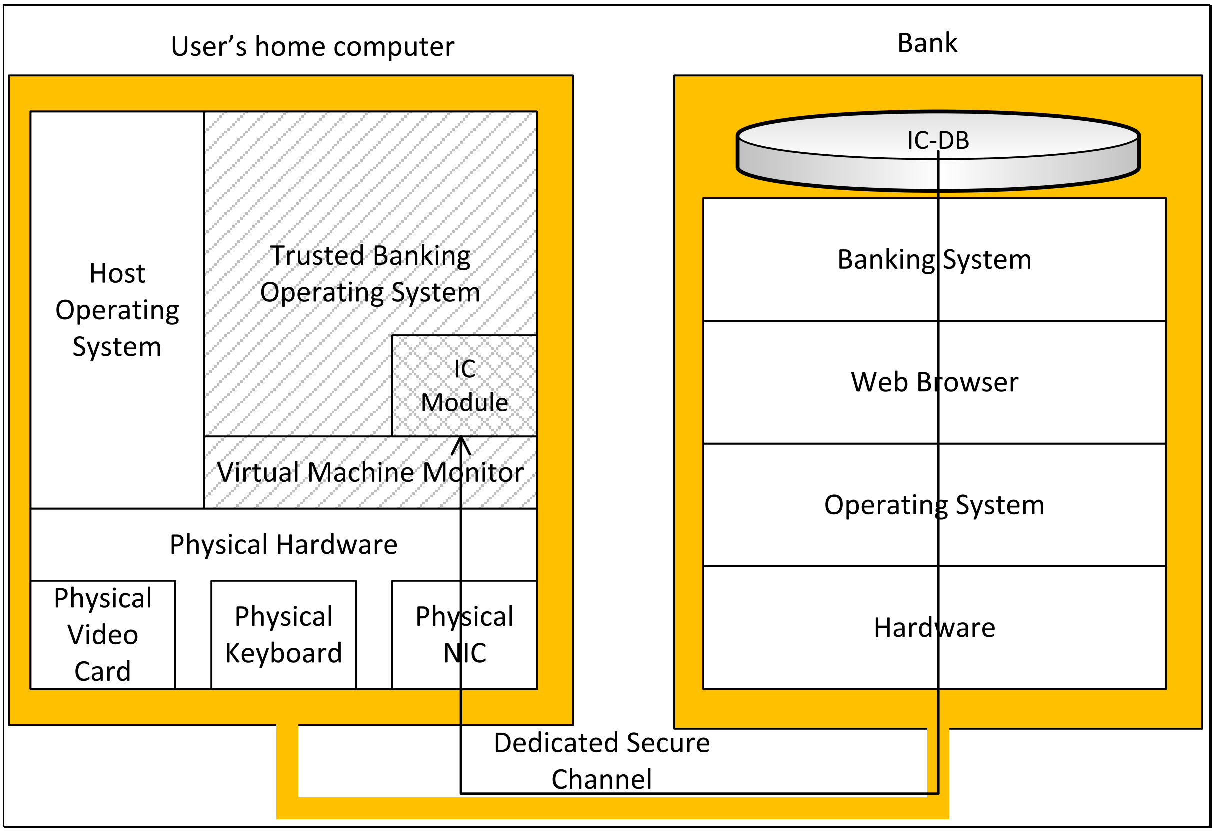


FIGURE 6:2 V-BANK ARCHITECTURE

A little bit more detail about the TBOS follows.

3.1 Trusted banking operating system

While researching virtual machines, it was realised that virtual machines have special isolation properties, that isolates the virtual machine from the rest of the user operating system. This property is used for the TBOS, by implementing it as a virtual machine.

It was also seen that virtual machines can be configured as a read-only virtual appliance. Using these properties the TBOS is defined with the following features:

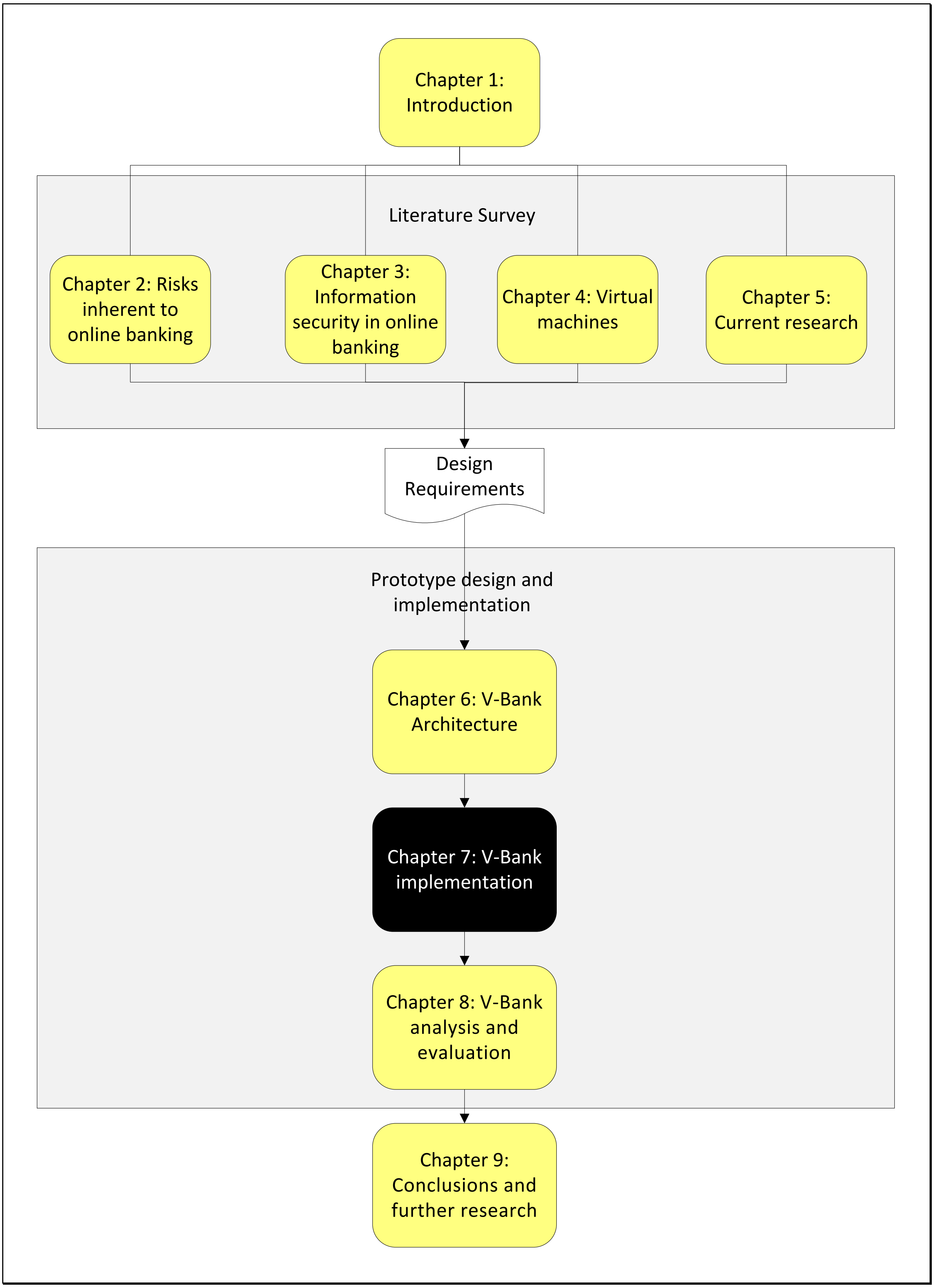
- **Persistent**. The TBOS image is read only and cannot change
- **Known configuration**. The bank has full control over the configuration of the TBOS.
- **Mutual identification and authentication**. The TBOS identifies itself to the banking system using a certificate stored on the client and the server identifies itself to the client using a certificate stored on the server.
- **IC Module**. The integrity of the TBOS is evaluated by the banking system, using the IC module and the IC database on the banking server. Before access is granted by the server to the client, the server interrogates the client and confirms the configuration of the client by evaluating key configuration items on the TBOS. The key-items are hashed and compared with the IC database on the server.
- **Private key signatures**. Banking transactions are signed using the client's private key stored in the TBOS. The electronic signature ensures non-repudiation of banking transactions.
- **Dedicated secure channel**. The client opens a TLS 1.2 channel to the banking server, known as the dedicated secure channel. Data between the client and the server is encrypted. The dedicated secure channel is also enforced using a firewall on the client. The firewall on the client ensures that the communication channel can only be established between the client and the server and no-one else.

4 Conclusion

The architecture of the V-Bank system is similar to the architecture described by Guang, et al. (2012). The big difference between the V-Bank system and the Domain Banking System is that the virtual machine from where the online banking is conducted is not running on a Xen hypervisor, but instead runs on a virtual machine monitor installed in the host operating system.

The implication of this is that the virtual machine monitor can be used on any hardware configuration, as long as the host operating system is supported by the virtual machine monitor. The V-Bank solution is more compatible than a system based on a VMM installed directly on the hardware and controlled from a Domain0 virtual machine, as in the case of Xen.

The next chapter describes how the architecture is implemented in the V-Bank system.


Chapter 1: Introduction
Literature Survey
Chapter 2: Risks inherent to online banking
Chapter 3: Information security in online banking
Chapter 4: Virtual machines
Chapter 5: Current research
Design Requirements
Prototype design and implementation
Chapter 6: V-Bank Architecture
Chapter 7: V-Bank implementation
Chapter 8: V-Bank analysis and evaluation
Chapter 9: Conclusions and further research

Chapter 7: V-BANK – COMPONENT DESIGN

1 Introduction

The dissertation so far has been working towards the design and creation of a V-Bank prototype. In the previous chapter the architecture that was defined for the V-Bank prototype is defined.

This chapter describes the various architectural components and explain how our V-Bank system is designed.

The next section describes how the components are organised into groups and where they are discussed in the chapter. The next sections describe each component. The chapter concludes with a summary of the design requirements and which components fulfilled the design requirements.

2 V-Bank Components

The architecture defines some components that need exclusive implementation on either the client or the server. Some components, however, require changes on both the client and the server, such as the dedicated secure channel.

There are also some components, such as the integrity check (IC) system on the server, that has a direct influence on the client, and even go as far as executing components on the client.

The chapter groups the components together in three categories. They are:

- Client side components
- Client and server-side components
- Server-side components

Each of these categories describes a specific component. Table 10 shows each of the identified components and mechanisms for each category. The components and mechanisms are chosen to fulfil specific design requirements, which are mentioned in each section dedicated for the specific component.

Client-side	Client and server-side	Server-side
Trusted Banking Operating System	Authentication mechanism	Web Server
Virtual Machine Monitor	Dedicated secure channel	Data Objects
Persistence	Integrity check-system	Access Layer
Operating system configuration		Business Layer
Operating system authentication		Data Layer
Private key component		

TABLE 10: COMPONENTS FOR EACH CATEGORY

The first category that requires attention is the client side components.

3 Client-side components

The client-side components that are discussed are:

- Trusted Banking Operating System (TBOS)
- Virtual Machine Monitor
- Persistence
- Operating system configuration
- Operating system authentication
- Private Key component

3.1 Trusted Banking Operating System

The trusted banking operating system (TBOS) is the operating system that is used on the client running as a virtual machine.

There are a number of aspects that must be taken into consideration when deciding on the TBOS. Each of the considerations is described in their own sub-sections. They are:

1. Operating system license costs
2. Flexibility
3. Bottom up approach or top down approach

This section concludes with an evaluation of the design requirements that are fulfilled by this component.

3.1.1 Operating system license costs

Even though the V-Bank system is a prototype, the costs of implementing the system in a commercial environment are taken into consideration. The two categories that are considered are:

- Microsoft licensed operating system
- Open source operating system

When a commercial institution deploys a single purpose operating system the cost per deployed operating system can get quite large, if the institution has to pay a license per copy deployed.

For the above reason, it was decided that the V-Bank system uses an open source operating system that does not require a license per deployed copy of the operating system.

3.1.2 Flexibility

The operating system has to be very flexible to allow changes in its functionality. The operating system must have the ability to change, has to have a small footprint and be built for a single purpose.

An open source operating system allows a bank to make changes to the source code of the operating system to ensure it fulfils the configuration requirements of the bank.

3.1.3 Bottom up approach or top down approach

The third item that requires consideration is to decide whether the open source operating system is built from scratch (bottom up approach), or use an existing Linux distribution.

Both the bottom up and top down approach are discussed

a.) Bottom up

The advantage of this approach is that the bank has full control over all aspects of the operating system. Because the bank knows the environment in which the operating system is executed, the operating system can be built to only support the virtual hardware or the virtual machine. Unnecessary software, normally included in operating systems, do not have to be included, such as calculators, or writing pads or multi-media players.

The major disadvantage of this approach is that it takes considerable development time to create an operating system of high assurance. Banks usually are not software development or even operating system development companies and may spend considerable money in developing such an operating system, even if the bank takes existing Linux operating system components to build the system.

b.) Top down

The top-down approach is where the bank decides to use an existing Linux distribution and then goes through a process of removing unnecessary items.

The most important advantage to this approach is that the work leverages on what is already implemented by the open source community. These distributions are also generally better supported and maintained from a security point of view. Vulnerabilities in the software identified by the community are patched to ensure a more stable, secure operating system.

A disadvantage of this approach is that the operating system can be bloated, taking up unnecessary space.

During the implementation of the V-Bank system, both approaches were tested. At first the recipes published as part of the Linux From Scratch (LFS) distribution to build a Linux distribution was used (Beekmans, 2012). An operating system was successfully built up to a level with a graphical user interface and a very basic Internet browser.

The top down approach was then tested. An Ubuntu Linux distribution went through the process of removing unnecessary items from the operating system.

Two working copies of Linux were produced. The top down approach, was faster and required less skill to accomplish. The end-product for both approaches was nearly the same size. The Linux From Scratch operating system failed to work as a read only bootable media, running in a virtual machine. This does not mean that it is impossible to get it to work on read only media.

From this discussion it became clear to use the top down approach with the Ubuntu based Linux as a working copy for the TBOS.

3.1.4 Design Requirements

Table 11 summarises the design requirements fulfilled by the TBOS. The numbers in brackets define the design requirement number fulfilled.

(11) The TBOS is based on the Ubuntu Linux distribution. Continual development and support on the distribution ensures that the operating system is patched when vulnerabilities are identified.

(13) The TBOS looks different from the rest of the operating environment and a user can easily identify the online banking system from normal web sites. It also displays a X.509 certificate on the web site that assists the user in identifying the banking web site.

(21) The TBOS is a managed component directly accessible by the bank.

(22) The TBOS runs as a virtual machine which means that the TBOS is isolated or sandboxed from the rest of the client operating environment.

Number	Requirement
11	Online banking systems must have the ability to be updated when software vulnerabilities are detected.
13	Online banking systems must have a visual trust anchor.
21	An online banking system should be designed to include not just a managed server environment, but also a managed client environment.
22	The client-side online banking system should be sandboxed from the rest of the client operating environment

TABLE 11: DESIGN REQUIREMENTS FULFILLED BY THE TBOS

3.2 Virtual Machine Monitor

The virtual machine monitor software chosen is based on a number of criteria:

1. **Compatibility**. Must be compatible to a wide range of host operating systems, but primarily Microsoft Windows based operating systems.
2. **License costs**. The distribution of the virtual machine software should not be very expensive if there are to be costs involved.
3. **Printing**. Some of the features that people already expect in an online banking system are to be able to print out statements or payment information. The virtual machine software must make the virtualisation of existing printers on the host easy.

Each of the criteria items are discussed next, followed by the design requirements fulfilled by the virtual machine monitor.

3.2.1 *Compatibility*

The virtual machine software must be compatible on Windows and Linux based computers and must support Linux guest operating system. The software should also be simple to install.

Using Chapter 4:6 as a guide the options for virtual machine software are narrowed town to Oracle VirtualBox and VMware Workstation or VMware Player.

If the requirements were extended to also include Apple computers then only Oracle VirtualBox was an option.

It may be possible to use different virtual machine software on different platforms. Testing the V-Bank system in these configurations is not be part of this study.

The next criterion to discuss is license costs

3.2.2 *License costs*

By using the compatibility requirement only Oracle VirtualBox and VMware are taken into account when considering the terms of license costs.

Oracle VirtualBox is freely distributable without any end user license costs. VMware Player is available freely for personal use (VMware, 2012). VMware workstation requires license costs.

VMware however has a special license programs for enterprise wide distribution programs which makes the license cost per installed instance cheaper.

3.2.3 *Printing*

One of the features that users are already familiar with in existing online banking systems is the ability to print out information. This information can be bank statements, or payment information.

From a commercial point of view this may be one of the most important aspects to consider when deciding which virtual machine software must be used.

Only VMware currently has the ability to automatically virtualise existing printers from the host operating system into the guest operating system. All of the other systems require the user to create the printer. This may be problematic since we don’t want users who may be unfamiliar with Linux to struggle with creating a printer. The persistent nature of the V-Bank system also means that the user has to create the printer every time the system starts. This may cause an aversion to using the system by users.

The only aspect left to discuss for virtual machine software is which one is the best option.

3.2.4 *Virtual machine monitor software conclusion*

Using the criteria that was defined for choosing virtual machine software is it clear that there are two options:

- Oracle VirtualBox can be used on Windows, Linux and Apple host computers. It is freely available and can be distributed to a bank’s customers without extra costs. Oracle

VirtualBox however does not automatically create virtualised printers in the guest operating system.

- VMware Player or VMware Workstation can be installed on Windows and Linux host computers. There are license costs involved if used for commercial purposes. The software however automatically creates printers inside the guest operating system that already exists on the host operating system, making the system very user friendly.

It was decided to use VMware Player as the virtual machine software to build the V-Bank prototype in. The printing criteria were the deciding factor in the decision between Oracle VirtualBox and VMware.

3.2.5 Design Requirements

Table 12 summarises the design requirements that are fulfilled by the virtual machine monitor.

(18) The virtual machine monitor is configured to only allow the printing guest tool. (19) VMware player does not have introspection abilities. It cannot control the working of the guest operating system, without the help of special guest tools. (20) VMware Player cannot create snapshots, like VMware Workstation, or Oracle VirtualBox.

(22) VMware Player creates an environment in which the TBOS can run as a virtual machine, isolated from the rest of the client operating environment. This creates a sandboxed environment in which the client conducts online banking.

Number	Requirement
18	The bare minimum guest tools should be enabled in a client-side virtual machine environment.
19	The VMM used by the online banking system should have limited introspection capabilities
20	Snapshots of a client-side operating virtual machine environment should not be possible
22	The client-side online banking system should be sandboxed from the rest of the client operating environment

TABLE 12: DESIGN REQUIREMENTS FULFILLED BY THE VIRTUAL MACHINE MONITOR

The next aspect of implementation that is discussed is the persistent characteristic of the TBOS.

3.3 Persistent image

Chapter 4:4.2.4 described that a virtual machine can be configured as a virtual appliance that can be made read only. The read only appliance can be implemented as a LiveCD. The advantage of the read only appliance is that it cannot be infected by malware. The read-only nature of the LiveCD, means it is persistent. Persistent images reset to their original state every time the system is started. No data and settings can be changed.

A LiveCD is a bootable image of a fully configured operating system with pre-defined applications and configuration. A LiveCD runs in the computer's memory and does not require space on the hard drive of the computer.

This section describes how the LiveCD is implemented and concludes with the design requirements that the persistent nature of the TBOS fulfils.

3.3.1 LiveCD implementation

For the V-Bank implementation it was decided that the TBOS will be distributed as an image of a LiveCD. This image is stored on a Universal Serial Bus (USB) memory stick which the user can then connect to his computer, from where the virtual machine software can then read the LiveCD image and boot the TBOS.

Because the LiveCD executes as an image, the image cannot change after it is created, without fully destroying it and then recreating it again.

The LiveCD image file is called "image.iso". This file can be mounted by the virtual machine software as a CD image. No virtual hard drive is necessary for the virtual machine configuration.

3.3.2 Design Requirements

Table 13 summarises the requirements fulfilled by the persistent nature of the TBOS.

(1) Because the TBOS runs as an image of a LiveCD, the configuration of the operating system cannot change.

(3) It is also not possible to change the Internet browser used by the V-Bank system.

(9) No software can be installed in the persistent image of the TBOS this includes malware, like keyloggers.

(11) The persistent image of the TBOS cannot be dynamically updated to patch identified vulnerabilities. Manual updates of the persistent image of the TBOS is necessary, although it may be possible to design a system to update the image deployed to clients, but this is outside the scope of this research.

Number	Requirement
1	The client operating environment should be persistent. It should not be possible to change any file or configuration in the client operating environment
3	It should not be possible to modify the Internet browser used by an online banking system.
9	A keylogger should not be allowed inside an online banking system or should minimize the effect it may have if present.
11	Online banking systems must have the ability to be updated when software vulnerabilities are detected.

TABLE 13: DESIGN REQUIREMENTS FULFILLED BY THE PERSISTENT NATURE OF THE TBOS

The next aspect that is discussed is the configuration of the operating system and how the bank can know that this is the configuration that is currently being used when a client connects to the V-Bank system.

3.4 Operating system configuration

The banks have the luxury of deciding exactly how the TBOS looks and functions. The TBOS is specifically configured for only online banking. This configuration is called the known configuration. The configuration that is defined by the banks includes configuring the following aspects:

1. **Single purpose operating system**. The Linux operating system has to be modified and configured so that it can only be used for online banking purposes with the V-Bank system.
2. **Least user account rights**. Any user using the operating system need only have basic permissions so that they can start the web browser to connect to the V-Bank system, provide login information and unlock their private keys when required.
3. **Security hardened operating system**. The operating system also goes through a process of hardening the security in the operating system.
4. **Locked down Internet browser**. The Internet browser that is used is locked down so that it does not allow browsing to sites other than the V-Bank system.

Each of these aspects are described in more detail next. This section concludes with the design requirements fulfilled by the known configuration of the operating system.

3.4.1 Single purpose operating system

The operating system goes through a process that ensure the system gets a new user interface that automatically starts the web browser and not allows the normal user to do anything else.

The single purpose operating system is configured through the following components:

- **Blackbox window manager** (Anon., 2006). The window manager that comes with Ubuntu is changed. The window manager is a component that handles the user interface screens once the user has logged in. The window manager that is used is the Blackbox window manager. The Blackbox window manager ensures the system has a custom menu in the operating system, allowing the user only access to the web browser. Blackbox window manager starts a script when the user logs in that start the Internet browser (Anon., 2006).
- **iDesk** (Anon., 2005). iDesk places icons on the Blackbox background so that the user can execute the Internet browser from there. Two icons are created, one icon starts the Internet browser and the second icon is used to shut down the operating system.
- **Gnome Display Manager** (GDM) (The Gnome Project, 2005). The login manager is the system that is first seen when the user starts up the Ubuntu operating system. It gives the user the ability to choose a username and type a password. To help strengthen the security, the login manager is modified so that it does not list the usernames that is available on the system. The user has to first type in his username and then the password, instead of selecting his username and then typing the password. The login manager is changed to the GDM instead of the Ubuntu login manager.

- **Oracle Java Runtime** (Oracle, n.d.). Another piece of software that is installed is the Oracle Java Runtime environment. This is required so that some Java applets can be initiated by the V-Bank system.

The above four components ensures the operating system starts an Internet browser when the user logs in.

The next area of configuration is to configure the rights a use has on the Linux operating system.

3.4.2 Least user account rights

In the previous section it was described how the operating system can only start an Internet browser when the user logs in. To further secure the operating system environment, the user accounts are configured so that unnecessary rights are removed.

The Linux operating system has two user accounts created in the system. The first user account is an account the bank uses to configure the operating system. This user account has root permissions on the TBOS and has a complex pass phrase to protect the user account. This user account name and password changes for each version of the V-Bank TBOS deployed.

The second account that is available on the system is an account that is given to the banking user. The account is based on the user's name and a password is generated and given to the user.

The account used by the banking user has limited permissions on the TBOS. Permissions such as the mounting of CD-ROMS or the using of sound cards are removed.

The next section discussed the security configuration of the operating system.

3.4.3 Security hardened operating system

Security hardening a Linux distribution means that the default operating system's security is increased to minimize the chances of attackers from exploiting a Linux system.

Bastille (Anon., 2013) is used to security harden the Ubuntu Linux distribution. Bastille makes it easier to security harden various aspects of the Linux operating system.

It should be noted that the firewall configuration that was generated by Bastille is only used as a guide to generate the IPTables.

Security hardening the operating system using Bastille ensures the Linux system files are protected from unauthorised access. It locks down user accounts to only have access to their home directories and improves the security of the Linux operating system significantly.

This section described how the security of the Linux operating system is enhanced. In the next section the configuration of the Internet browser and how it is implemented in the TBOS is described.

3.4.4 Locked down Internet browser

At the time of writing this dissertation the only two Internet browsers that supported TLS 1.2 were Microsoft Internet Explorer 9.0 and above and Opera 11 (Opera Corporation, 2013) and above.

Only Opera is a viable option to be installed on the TBOS Linux operating system. It was decided to use Opera 12 as the Internet browser for the V-Bank system.

Opera 12 has many features which supports a locked down experience.

When the Opera browser is started using the following command, the browser starts by default in a locked down mode:

```
opera -kioskmode -kioskbuttons -nocontextmenu -nodownload -nokeys -
nomail -nomaillinks -nomenu -nosave -resetonexit
```

The meaning of the switches are as follow:

-kioskmode	Starts the web browser in full screen mode, does not allow the user to close the browser
-kioskbuttons	Allows some buttons, like Home on the browser address bar
-nocontextmenu	Does not allow context menus in the browser
-nodownload	Cannot download anything using the browser
-nokeys	Does not allow special keys to be used in the browser, like Ctrl+Alt+Delete
-nomail	Does not enable the mail client functionality of the Opera browser
-nomaillinks	Does not follow mail links in hyperlinks
-nomenu	Does not create a menu
-nosave	Does not allow the user to save pages
-resetonexit	Resets the session information when the browser closes

Over and above this, the system also use a feature built into Opera which allows only certain web sites to be visited from the web browser. This is done by creating an urlfilter.ini file, which is made available to the client. Opera is then further configured to use this file. Appendix A describes how this is configured and installed.

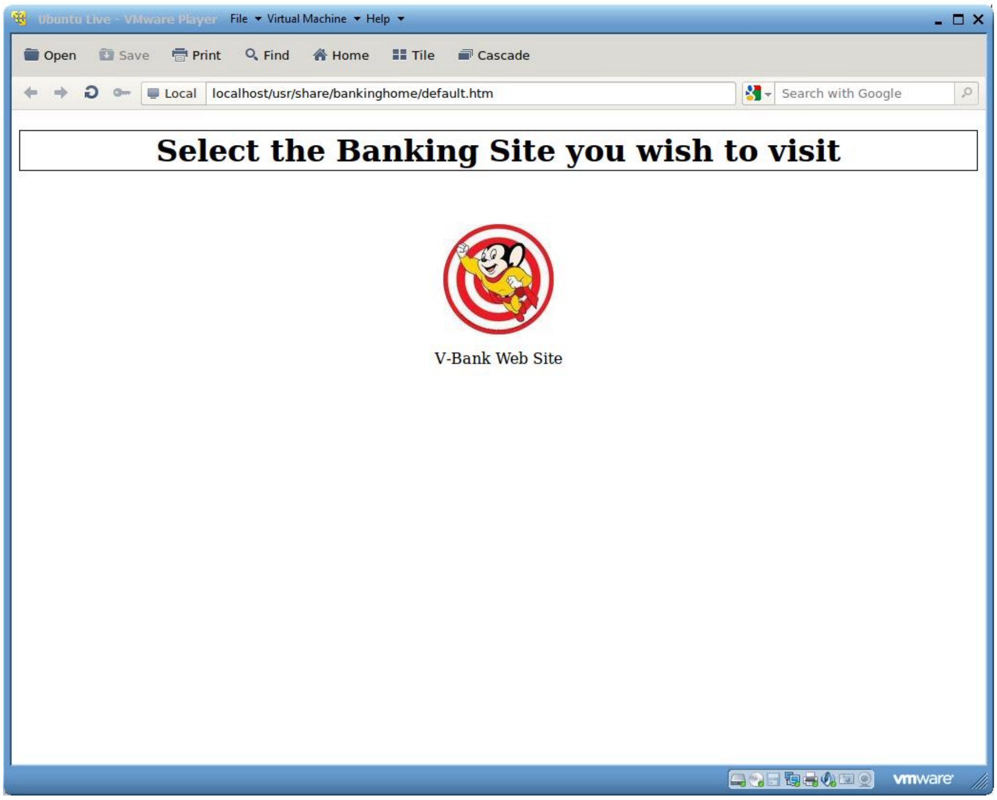

Ubuntu Live - VMware Player
File
Virtual Machine
Help
Open
Save
Print
Find
Home
Tile
Cascade
Local
localhost/usr/share/bankinghome/default.htm
Search with Google
Select the Banking Site you wish to visit
V-Bank Web Site
vmware

FIGURE 7:1 OPERA BROWSER RUNNING KIOSKMODE GOING TO THE CUSTOM LAUNCH PAGE

The home page of the V-Bank system is set to a file that is stored in the computer. This file is used as a launch pad to access the banking application on the server-side. Figure 7:1 shows how the Opera browser looks when it runs in a locked down mode. Note that there is no menu, but there are some buttons that exists. The only way the user can exit the application is when he closes the whole virtual machine. The user cannot gain access to any other operating system command or application.

The operating system is specifically configured to only allow access to the V-Bank server-side environment. The requirements are fulfilled by configuring the operating system in this way is described in the next section.

3.4.5 Design Requirements

The operating system is configured specifically for online banking in the V-Bank system. Table 14 summarises the design requirements that are fulfilled by the configuration of the operating system.

(4) The Internet browser is locked down to only open the V-Bank system web site. It cannot open multiple pages.

(9) The operating system is security hardened, so that users cannot change important system files in the operating system. The security hardened operating system will not allow a user to install a keylogger in the system.

(16) The hardening of the security of the operating system means one of the security layers of a layered security program is implemented.

Number	Requirement
4	Online banking systems should not allow the Internet Browser from opening multiple pages and the transaction component in an online banking system should re-authenticate the user, before committing important transactions.
9	A keylogger should not be allowed inside an online banking system or should minimize the effect it may have if present.
16	Online banking systems must implement a layered security program.

TABLE 14: DESIGN REQUIREMENTS FULFILLED BY THE OPERATING SYSTEM CONFIGURATION

The second last client-side component under discussion is the operating system authentication.

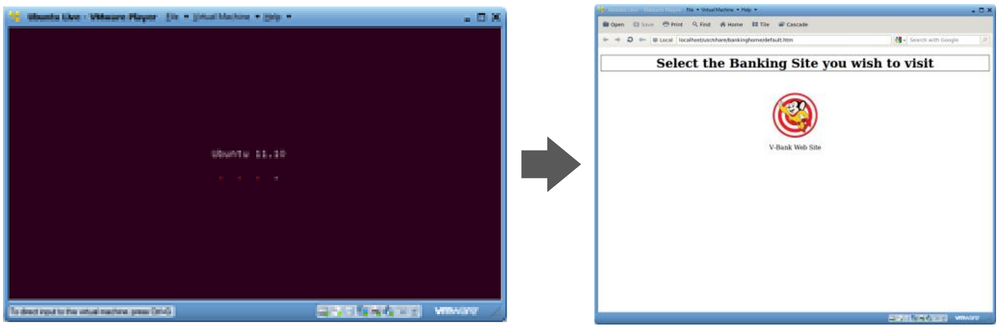
3.5 Operating system authentication

The previous sections described which operating system is used for the TBOS, which virtual machine monitor software is used, how the TBOS persistence is enabled and how the TBOS is configured. This section described how the TBOS authenticates the user.

The section concludes with the design requirements fulfilled by the operating system authentication.

3.5.1 Login process

The TBOS can be configured to either automatically log a client into the operating system as shown in Figure 7:2 or to request a username and password before the Opera browser can be used to open the connection to the V-Bank system.



Select the Banking Site you wish to visit

FIGURE 7:2 AUTOMATIC LOGIN OF THE TBOS

An advantage of automatically logging the client into the TBOS is that the client does not have to remember a username and password before the client can start using the V-Bank system. The V-Bank system still requires the client to provide a password to unlock his private key and password for the random authentication code of the V-Bank system. The disadvantage of this is that someone that steals the V-Bank client side system and knows the password for the private key and the password to the random authentication code can potentially access the V-Bank system.

The other option is for the client to type the TBOS username and password before the user can start using the V-Bank system. Figure 7:3 shows that the TBOS will ask the user for a username and password before the Internet browser starts. The advantage of this is that it makes it more difficult for an attacker that has physical access to the client side TBOS to gain access to the user's banking profile.

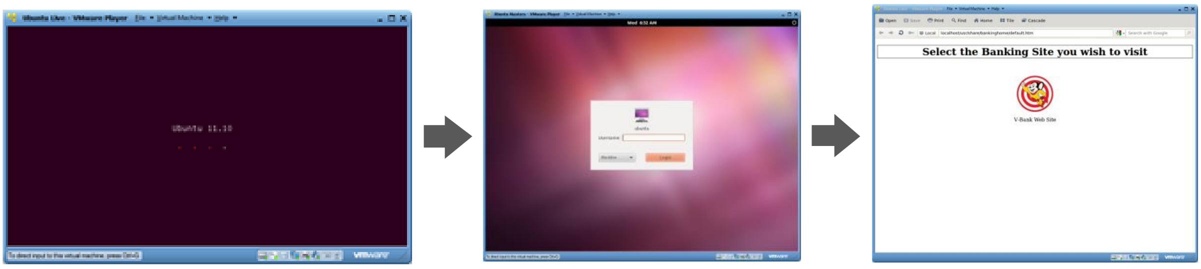

Ubuntu Live - VMware Player
Ubuntu 11.10
vmware
Username
Login
localhost/usr/share/bankinghome/default.htm
Select the Banking Site you wish to visit
V-Bank Web Site

FIGURE 7:3 TBOS OPERATING SYSTEM AUTHENTICATION.

A disadvantage of having to prompt the user for a username and password before access is granted to the TBOS is that the user has to remember another username and password.

The V-Bank prototype requires the user to type a username and password before access is granted to the TBOS.

3.5.2 Design Requirements

Table 15 summarises the design requirements that are fulfilled because the operating system authenticates the user.

(10) One of the items that must be stolen from a user is both his operating system login account name and password.

(16) The operating system authentication is one of the layers of the security program.

(17) The operating system authentication fulfils one of the identification and authentication ISO 7498-2 security service.

Number	Requirement
10	An online banking environment should minimise the chances of an attacker using stolen information to gain access to a customer's online banking profile.
16	Online banking systems must implement a layered security program.
17	Online banking system should implement information security controls in all five ISO 7498-2 security service categories.

TABLE 15: DESIGN REQUIREMENTS FULFILLED BY THE OPERATING SYSTEM AUTHENTICATION

In the next section we describe the areas where the user's private key is stored.

3.6 Private Key component

In the previous section we have seen that the user must provide a username and password before the user can gain access to the TBOS. This section describes what the use of the user's private key is and how it is stored in the TBOS.

The section concludes with which design requirements are fulfilled by implementing a private key for the user in the TBOS.

3.6.1 Private Key storage

The user's private key is used in a number of areas of the V-Bank system. They are:

- Dedicated secure channel
- During transaction authorisations in the authentication service.

During the session establishment with the V-Bank server the client must take part in a mutual authentication handshake enforced by the server. For the client to establish this connection the client utilises his private key. The private key must be loaded inside the user's Web browser. The web browser prompts the user to determine which private key should be used.

Another area where the private key is used is during the authorisation of banking transactions. During the authorisation process, the client signs the hash of the transaction details with his private key. The Java applet uses a password presented by the system, to unlock the private key that is stored in a file on the TBOS. This private key is not part of the browser setup and requires that the system unlocks the file.

The biggest disadvantage to this private key stored in a file, is that it may be possible for an attacker to get access to the private key file. The attacker can then use some type of brute force attack on the file to try and extract the user's private key (Ellison & Schneier, 2000).

The user is issued a private key using a certificate authority. The private key is exported to a file, which is password protected. This file is loaded into the TBOS system per user and then imported into the browser and then stored in a folder in the user's home directory.

3.6.2 Design Requirements

Table 16 summarises the design requirements fulfilled by implementing a private key for the user.

(9) Even if a keylogger is present on the host system, access to the V-Bank system is dependent on the certificate hosting the user's private key in the browser.

(16)(17) The private key component acts as identification and authentication of the user to the V-Bank system. This enables another security layer in the layered security program and it fulfils the authentication security mechanism defined by ISO 7498-2.

(17) The private key is also used to sign banking transactions. This ensures non-repudiation of the banking transactions. Non-repudiation is another security service of ISO 7498-2.

Number	Requirement
9	A keylogger should not be allowed inside an online banking system or should minimize the effect it may have if present.
16	Online banking systems must implement a layered security program.
17	Online banking system should implement information security controls in all five ISO 7498-2 security service categories.

TABLE 16: DESIGN REQUIREMENTS FULFILLED BY IMPLEMENTING A PRIVATE KEY

The implementation of the private key component is the last client side component that is discussed in this section. In the next section we discuss the components that have implementation areas in both the client and the server.

4 Client and Server-side components

Some of the components implemented in the V-Bank system has to be implemented on both the client and the server.

The components that fall in this category are:

1. Authentication mechanism
2. Dedicated secure channel
3. Integrity check system

Each of the components will form the same structure as the pervious sections where the component is described and also include the system requirements fulfilled by the component.

4.1 Authentication mechanism

The V-Bank system uses mutual authentication.

Mutual authentication means the V-Bank system authenticates the client before access is granted, but the client also authenticates the server before trying to communicate.

The authentication mechanism is enabled in a number of areas, they are:

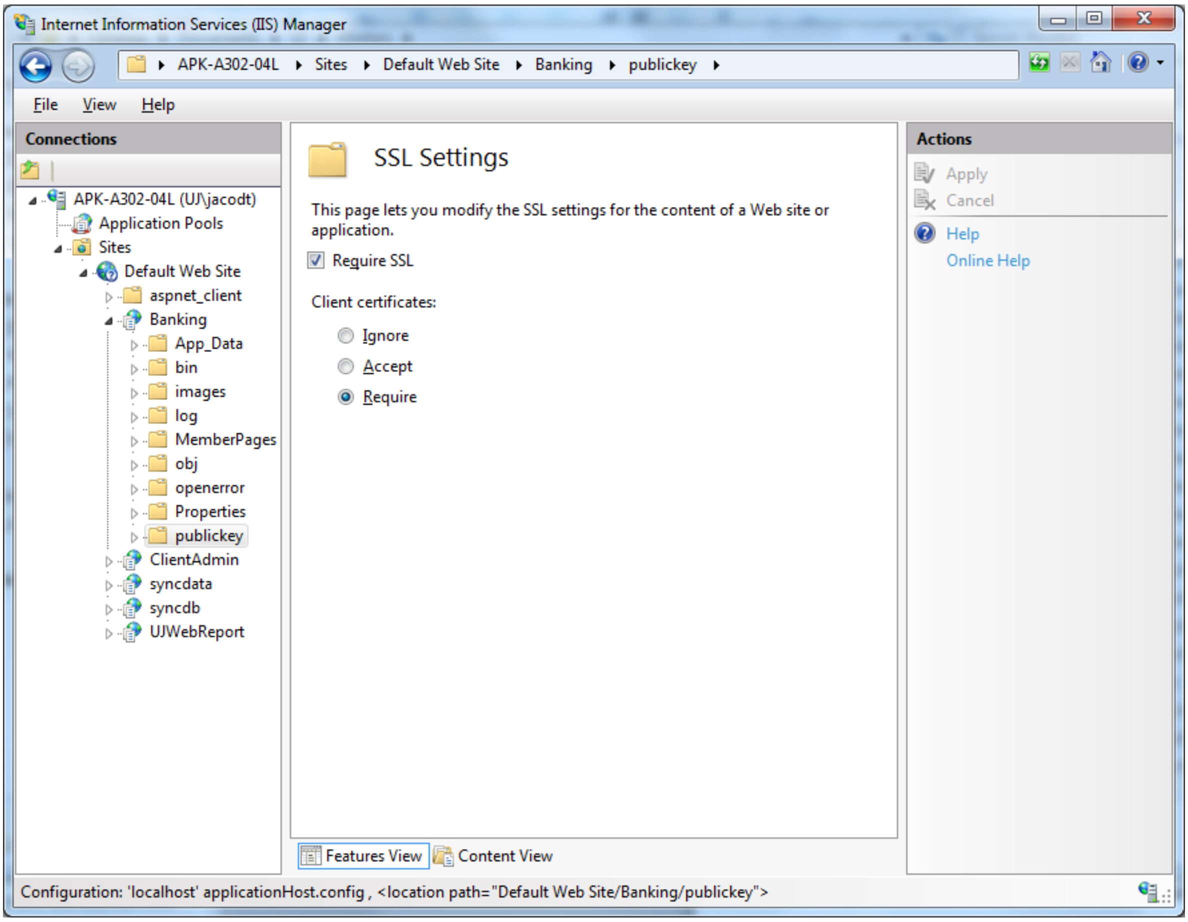
1. The web server
2. The Internet browser
3. The Java runtime environment

The configuration of these areas is now described and concludes with the design requirements fulfilled by the authentication mechanism.

4.1.1 The web server

The web server has to be configured so that it identifies itself to the client. The web server is also configured to not only identify itself to the client, but also so that the client identifies itself to the server.

The authentication mechanism is applied on the virtual directory of the web server. The configuration is set so that the banking system requires encryption.



SSL Settings

This page lets you modify the SSL settings for the content of a Web site or application.

☑ Require SSL

Client certificates:

○ Ignore

○ Accept

● Require

Configuration: 'localhost' applicationHost.config , <location path="Default Web Site/Banking/publickey">

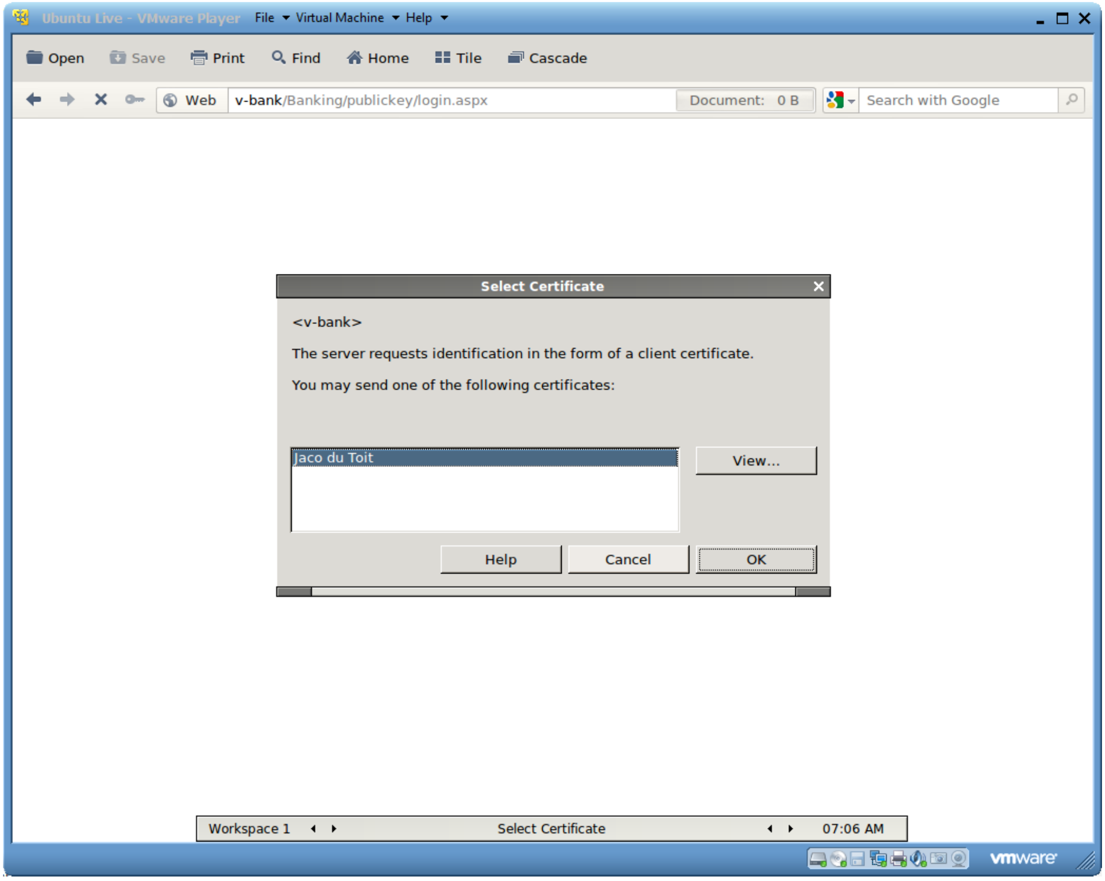
FIGURE 7:4 IIS 7.5 SSL SETTINGS PAGE

The mutual authentication is enabled using Microsoft Internet Information Server (IIS) 7.5 built-in functionality as shown in Figure 7:4. The server's certificate is loaded on the IIS server. The banking system's SSL settings are enabled in IIS to require SSL encryption and require client certificate mapping. This means that the server present himself to the client using an industry standard X.509 certificate, but the client must also identify itself to the server, using a certificate stored on the client.

The certificate on the server ensures that the browser on the client encrypts traffic to and from the server using the server's public key.

4.1.2 The Internet browser

The client is asked to present a certificate with a private key shown in Figure 7:5. This certificate is loaded in the client's browser.



<v-bank>

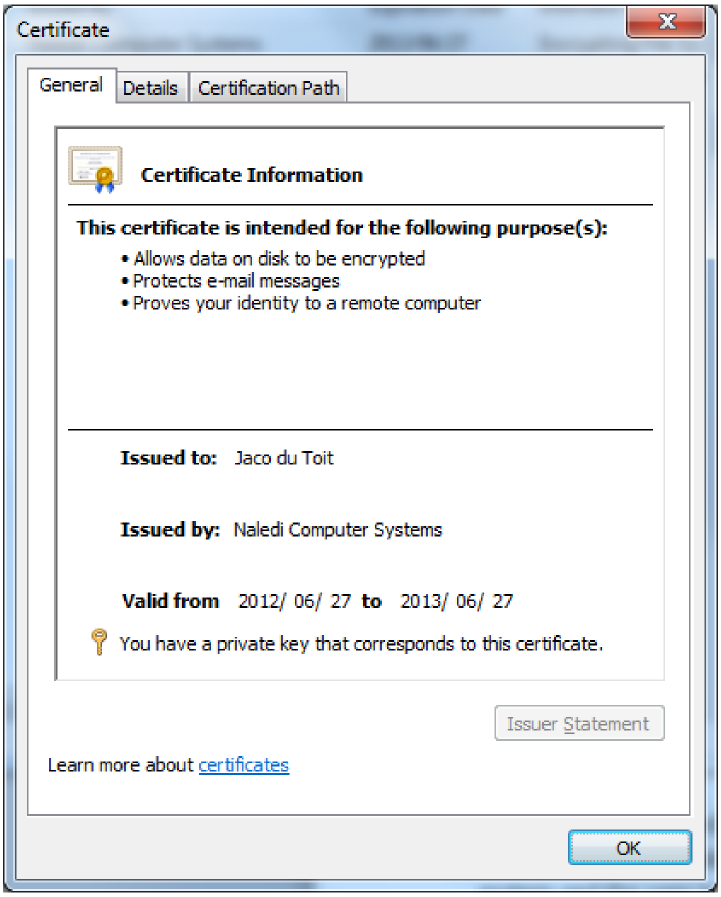
The server requests identification in the form of a client certificate.

You may send one of the following certificates:

Jaco du Toit

FIGURE 7:5 CLIENT AUTHENTICATION TO THE SERVER.

The V-Bank system uses the certificate that belongs to the user to extract the public key. The public key is then used to identify the user to the banking system after which the client is presented the V-Bank login screen.



Certificate Information

This certificate is intended for the following purpose(s):

- Allows data on disk to be encrypted
- Protects e-mail messages
- Proves your identity to a remote computer

Issued to: Jaco du Toit

Issued by: Naledi Computer Systems

Valid from 2012/ 06/ 27 to 2013/ 06/ 27

You have a private key that corresponds to this certificate.

Learn more about certificates

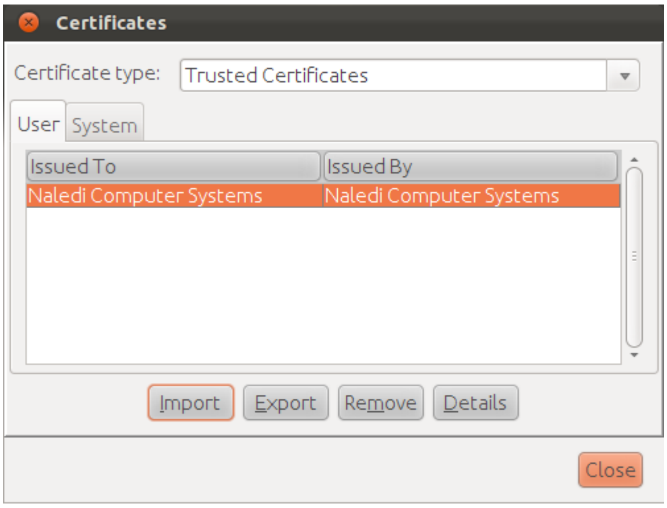
FIGURE 7:6 X.509 CERTIFICATE BELONGING TO A CLIENT

Figure 7:6 shows an example of an X.509 certificate that belongs to a client that is loaded in the certificate store that is used by the server to identify the client, when the client connects.

The Internet browser is one of the components that connect to the server to ensure that traffic is encrypted. Another component that opens connections to the server is the Java runtime environment that is installed in the TBOS.

4.1.3 The Java runtime environment

The Java runtime environment executes Java applets presented by the server. These applets are transferred to the client. Before the Java applet executes on the client the Java runtime client verifies whether the Java applet has been signed by a trusted source.



Certificate type: Trusted Certificates

Issued To	Issued By
Naledi Computer Systems	Naledi Computer Systems

FIGURE 7:7 JAVA CONFIGURATION FOR TRUSTED CERTIFICATES.

In this case every Java applet that is produced must be signed by a special code-signing V-Bank private key. The X.509 certificate for this code-signing private key must be trusted by each and every Java runtime client that is installed on the TBOS clients. This is accomplished by using a code signing certificate that is issued by a trusted certificate authority. Figure 7:7 shows the Java runtime client's control panel where the administrators of the V-Bank system will prepopulate the runtime client with trusted certificates.

Whenever this verification takes place the browser opens the Java key store on the client and the Java runtime verifies the signature according to the certificates loaded in the Java key store. This process identifies and authenticates the code from the server to ensure that the code is authorised to run on the client.

If the Java runtime client does not trust the signature in the Java applet then the Java applet cannot execute on the client.

The authentication mechanism requires a number of configuration settings to be applied per client. This process needs to be automated to ensure the V-Bank support department does not spend unnecessary time per client when issuing a V-Bank system to a client. The automation of this process is not part of this study and may require further study.

The V-Bank online banking system makes extensive use of Java. At the time of writing the dissertation new security vulnerabilities have been identified in Java (United States Computer Emergency Readiness Team, 2013). The V-Bank solution is susceptible to these known vulnerabilities provided that an attacker can access the TBOS over the network. In the next section it will be made clear that the TBOS is protected from network connections not originating from the V-Bank system. This mitigates the risk of these vulnerabilities.

Authentication is further described to understand how the server authenticates the client during login and then re-authenticates and authorize transactions during a banking session. This authentication mechanism is server-side only and is described in Chapter 7:5.1

4.1.4 Design Requirements

Table 17 summarises the design requirements that are fulfilled by the authentication mechanism that has to be configured on the client and the server.

(9) Because the authentication mechanism uses certificates stored on the client and server and is not depended on keyboard input, information stolen by a keylogger cannot be used to gain access to the V-Bank system.

(13) The certificate identification from the server creates a visual certificate icon in the browser of the user, giving the user a visual trust anchor.

(14) The V-Bank system requires mutual authentication by using certificates stored on the client and the server. Access will not be granted to the user if the user does not present a public key stored in a certificate to the server. This means that the user is authenticated before access is granted to the V-Bank system.

(16) The mutual authentication mechanism is one of the layers in the layered security program implemented by the V-Bank system.

(17) The authentication mechanism described in this section is an identification authentication security service as described by ISO 7498-2.

Number	Requirement
9	A keylogger should not be allowed inside an online banking system or should minimize the effect it may have if present.
13	Online banking systems must have a visual trust anchor.
14	Online banking systems must properly authenticate users before allowing access.
16	Online banking systems must implement a layered security program.
17	Online banking system should implement information security controls in all five ISO 7498-2 security service categories.

TABLE 17: DESIGN REQUIREMENTS FULFILLED BY THE AUTHENTICATION MECHANISM

The next section describes how to implement the dedicated secure channel.

4.2 Dedicated secure channel

The dedicated secure channel is responsible to ensure that traffic to and from the TBOS can only go to the V-Bank server components. The dedicated secure channel consists of two configuration items. They are:

1. The firewall on the client and server
2. TLS 1.2 encryption

This section concludes by describing the design requirements fulfilled by the dedicated secure channel.

4.2.1 *The firewall on the client and server*

The firewall on the client ensures network traffic can only flow between the client and the server. The firewall on the server minimises any other types of traffic to the server.

The implementation of the client-side firewall is in the form of Linux IPTables. IPTables is a firewall system that is included in Ubuntu Linux distributions.

In the V-Bank implementation, the firewall allows only HTTP and HTTPS traffic to and from the IP address of the V-Bank server.

DNS traffic is not allowed to or from the TBOS. The hosts file, which a helps identify IP addresses to host names, is populated with the name of the V-Bank system before the time. This is done to ensure that DNS contamination will not cause traffic from the client to flow to another host on the Internet.

The following entry is one of the entries in the IPTables configuration file, which allows HTTPS traffic to the V-Bank system given the IP address of the server as 10.0.0.14

```
NEW,RELATED,ESTABLISHED -m tcp --sport 1024:65535 --dport 443 -j ACCEPT
-A OUTPUT -d 10.0.0.14/32 -o eth0 -p tcp -m state –state
```

This entry may need to be changed depending on the number of Interface servers the bank has, but for the prototype only one entry is required to one IP address.

The firewall on the server-side is enabled to accept HTTPS traffic from any client. For the prototype the normal Windows firewall is used, but in practice another type of firewall with dedicated intrusion detection mechanisms can be used.

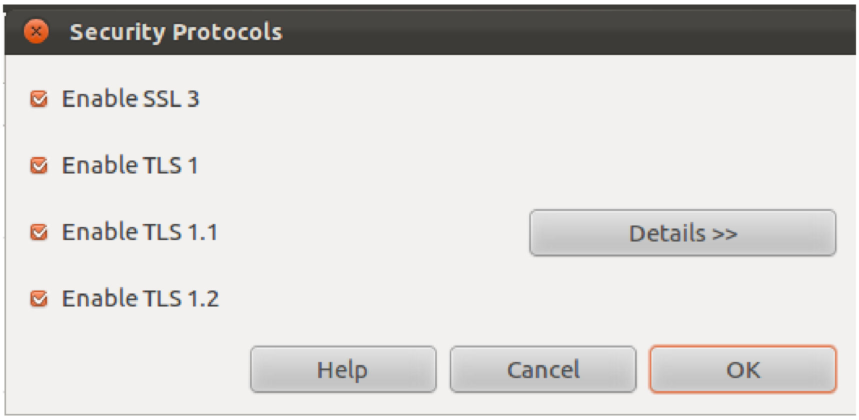
Apart from the firewall that helps minimise the chance of traffic going to and from any other host except the V-Bank system, the V-Bank system also implements TLS 1.2 as the transport layer security standard.

4.2.2 *TLS 1.2 encryption*

TLS 1.2 encryption is used to encrypt traffic between the client and the server.

TLS 1.2 is enabled on the Opera browser, as well as the Java runtime client. The IIS server is also enabled to use TLS 1.2.

The Opera browser's setting is part of the advanced settings in the browser's Preferences menu as shown in Figure 7:8.


Security Protocols

☑ Enable SSL 3

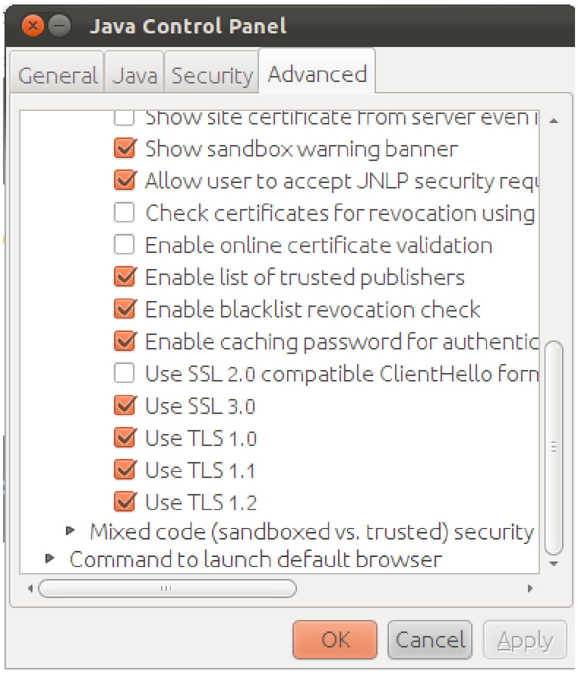
☑ Enable TLS 1

☑ Enable TLS 1.1

☑ Enable TLS 1.2

FIGURE 7:8 ENABLE TLS 1.2 IN OPERA

Figure 7:9 show the Java runtime client's TLS 1.2 is set using the Java control panel.


Java Control Panel

□ Show site certificate from server even i
☑ Show sandbox warning banner
☑ Allow user to accept JNLP security requ
□ Check certificates for revocation using
□ Enable online certificate validation
☑ Enable list of trusted publishers
☑ Enable blacklist revocation check
☑ Enable caching password for authentic
□ Use SSL 2.0 compatible ClientHello forn
☑ Use SSL 3.0
☑ Use TLS 1.0
☑ Use TLS 1.1
☑ Use TLS 1.2

▸ Mixed code (sandboxed vs. trusted) security
▸ Command to launch default browser

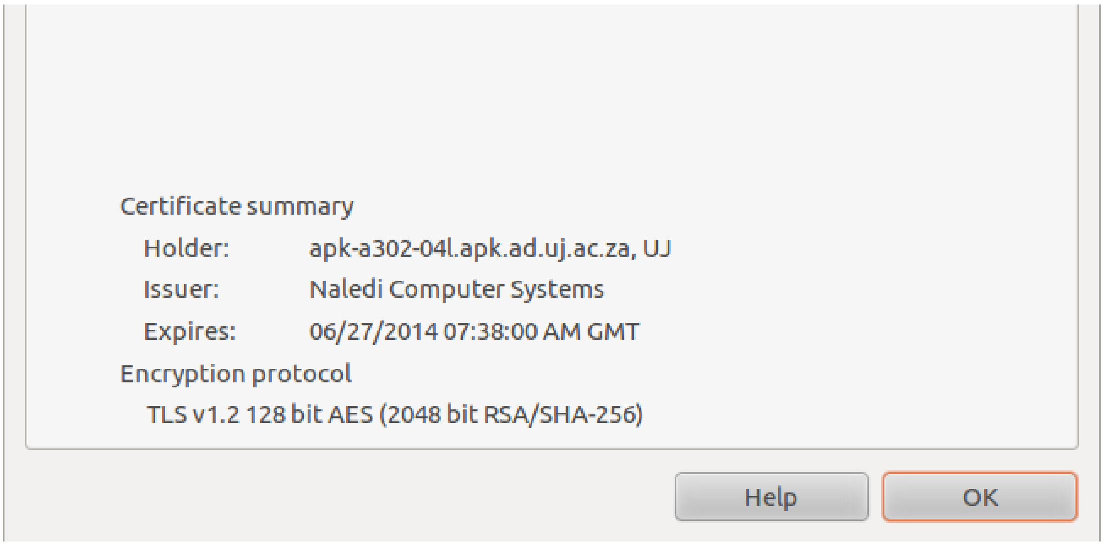
FIGURE 7:9 ENABLING TLS 1.2 IN JAVA

For IIS special registry settings have to be configured to ensure TLS 1.2 is enabled. The registry keys on the IIS 7.5 server are:

```
HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols
\TLS 1.2\Server
```

See the Appendix A for a Windows PowerShell script that can be executed on the server that creates the necessary registry keys and registry settings.

Once all the settings are made the Opera browser can be queried to determine whether the session is a TLS 1.2 session as shown in Figure 7:10.



Certificate summary

Holder: apk-a302-04l.apk.ad.uj.ac.za, UJ

Issuer: Naledi Computer Systems

Expires: 06/27/2014 07:38:00 AM GMT

Encryption protocol

TLS v1.2 128 bit AES (2048 bit RSA/SHA-256)

FIGURE 7:10 TLS 1.2 ENCRYPTION VERIFICATION

4.2.3 Design Requirements

Table 18 summarises the design requirement fulfilled by the dedicated secure channel.

(2) By using the firewall on the client, network traffic can only flow between the client and the V-Bank system. Traffic to and from any other host is not allowed.

(7) The firewall is configured with the IP address of the server. This ensures that name resolution does not affect the connection of the TBOS.

(9) In the remote chance that a keylogger is installed in the TBOS then the keylogger cannot send the captured keys to a 3rd party, as only network traffic between the TBOS and the V-Bank server-side system is allowed.

(16) The firewall and the TLS 1.2 encryption is one of the layers in the layered security program implemented by the V-Bank system.

(12)(17) The TLS 1.2 encryption ensures HTTP traffic is encrypted between the client and the server. Encryption is one of the security services defined by ISO 7498-2.

Number	Requirement
2	The client operating environment should be isolated on the network and only be allowed to connect to the banking system environment.
7	Name resolution should not affect the connection inside an online banking system
9	A keylogger should not be allowed inside an online banking system or should minimize the effect it may have if present.
12	Online banking systems must ensure TLS 1.2 channel security
16	Online banking systems must implement a layered security program.

17	Online banking system should implement information security controls in all five ISO 7498-2 security service categories.

TABLE 18: DESIGN REQUIREMENTS FULFILLED BY THE DEDICATED SECURE CHANNEL

The dedicated secure channel helps protect the client from most network based attacks on the TBOS, since it does not allow traffic to the client from any other host except the V-Bank server-side system.

4.3 Integrity check system

To ensure that the integrity of the TBOS stays intact, an integrity check (IC) system is employed. The IC system is implemented in client and server-side components. The IC system is enabled as an integrity check module (IC Module) on the client and integrity check database (IC Database) on the server.

This section describes the IC system and how it is configured on the V-Bank system.

4.3.1 IC System

When the TBOS connects to the server-side system, the server requests that the client provide some configuration settings. This request is implemented using a number of Java Runtime applets. The configuration settings checked by the server is the Opera browser and the Java version.

The server confirms that the client is currently using the approved Opera browser version and Java version before access is granted to the banking site.

When the client logs in to the server, the server initiates a Java applet that initiates the integrity checker module. The integrity checker module does the following:

1. It retrieves a list of all the processes that currently have a network connection open from the client.
2. The actual file that's causing the process is then determined
3. The file is then found on the TBOS file system and a hash is calculated for the binary value of the file.
4. The hashed information is sent back to the server
5. The server confirms whether the names of the processes and hash values of the processes currently connected to the network is the correct values as defined by the bank.

Figure 7:11 shows an example of a client that opened the approved version of the Opera browser and the approved version of Java. The Java applet running on the client looked for the application files on the hard drive and produced a hash value of each. These hash values are then sent to the server. The server compares the hash values and the names of the processes with its approved configuration and grants access accordingly.

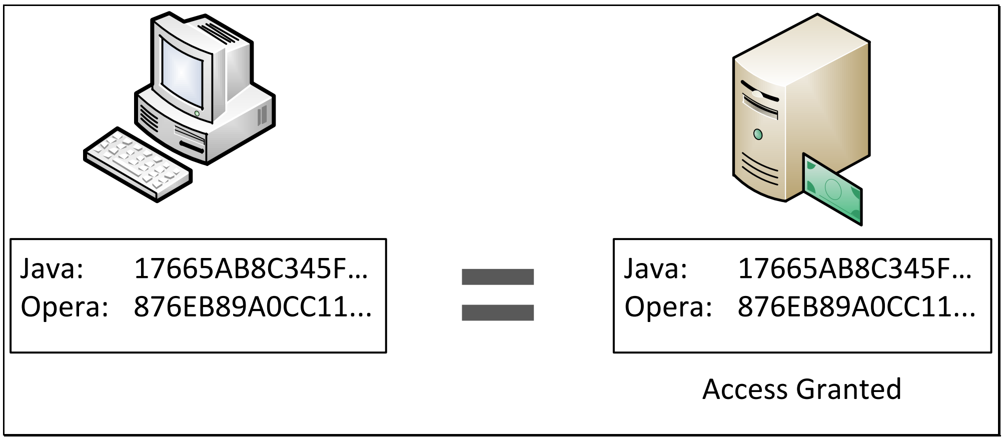


FIGURE 7:11 IC MODULE.

The configuration verification process can be enhanced to verify many client side configuration items to see whether the client system integrity is intact.

4.3.2 Design Requirements

Table 19 summarises the design requirements fulfilled by the IC System.

(8) The server confirms the configuration on the client by executing integrity checks. This means that clients that do not have the approved configuration will not be able to connect to the server.

(10) If an attacker steals the user's username and password then the attacker will still not be able to gain access to the V-Bank system, without the TBOS generated for the client. Any other configuration will not be allowed.

(16) The IC System is one of the layers in the layered security program implemented by the V-Bank system.

(17) The IC System is another form for authenticating and ensuring integrity of the client. The authentication and integrity is security services defined by ISO 7498-2.

(21) The IC System ensures that the client is a managed component, just as the server can be managed by the bank.

Number	Requirement
8	An online banking system should only allow connections to and from approved clients and servers.
10	An online banking environment should minimise the chances of an attacker using stolen information to gain access to a customer's online banking profile.
16	Online banking systems must implement a layered security program.
17	Online banking system should implement information security controls in all five ISO 7498-2 security service categories.

21	An online banking system should be designed to include not just a managed server environment, but also a managed client environment.

TABLE 19: DESIGN REQUIREMENTS FULFILLED BY THE IC SYSTEM

The IC System is the last component that requires configuration on both the client and the server. The next section describes the components that exist on the server.

5 Server-side components

It is assumed that the server-side environment is highly secure and is beyond the scope of the research.

A number of components require implementation on the server. For the V-Bank prototype only a simulated banking system is created. This section describes the various components that are implemented in the V-Bank system and the technologies that are used to fulfil the design requirements.

The system is designed according to the following high level programming architecture as shown in Figure 7:12.

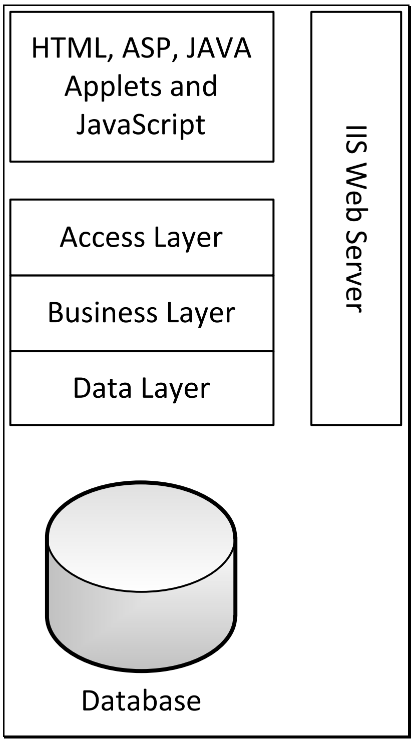


FIGURE 7:12 SERVER-SIDE ARCHITECTURE IMPLEMENTATION

There are three layers. The three layers are:

- **The Access Layer.** The Access Layer consists of a number of Hypertext Markup Language (HTML), Active Server Pages (ASP), JavaScript and Java Applets that are presented via HTTPS to the clients. The Access Layer creates a number of data objects that is then given to the Business Layer where business rules are applied on the data object. Data objects are a defined programming structure with specific properties. The access layer uses data objects instead of data tables. Using data objects mean that the object can only contain certain types of information, minimising the chances of SQL injection attacks.

- **The Business Layer.** The Business Layer interfaces with the data store via a Data Layer to ensure that the Business Layer does not have to worry about what type of data store is implemented on the V-Bank system.
- **The Data Layer.** The Data Layer transform data objects to relevant SQL queries, which may update, insert or select the data stored on the SQL server.

The Database utilised by the V-Bank prototype was Microsoft SQL Server 2008.

The Access Layer, Business Layer and Data Layer are all implemented as an IIS ASP.NET application. Some of the Access Layer components are protected by the IIS forms authentication and SSL encryption.

The server-side components are:

- Web Server.
- Data Objects.
- Access layer.
- Business layer.
- Data layer.

The component that requires special implementation and configuration is the web server.

5.1 Web Server

The web server is primarily responsible for accepting Hypertext Transfer Protocol (HTTP) connections over port 80 from the clients. The web server requires special configuration to ensure web traffic is handled correctly by the V-Bank system.

The next section describes how the user is given authorised access to different areas in the V-Bank system depending on the authentication.

5.1.1 Layered authorisation

The web server presents pages in three areas of the V-Bank system. The first area is unprotected pages, which means anyone can connect to these pages. The second area that the web server presents are protected SSL encrypted pages. The last area presented by the web server, are the SSL encrypted forms authenticated pages as shown in Figure 7:13.

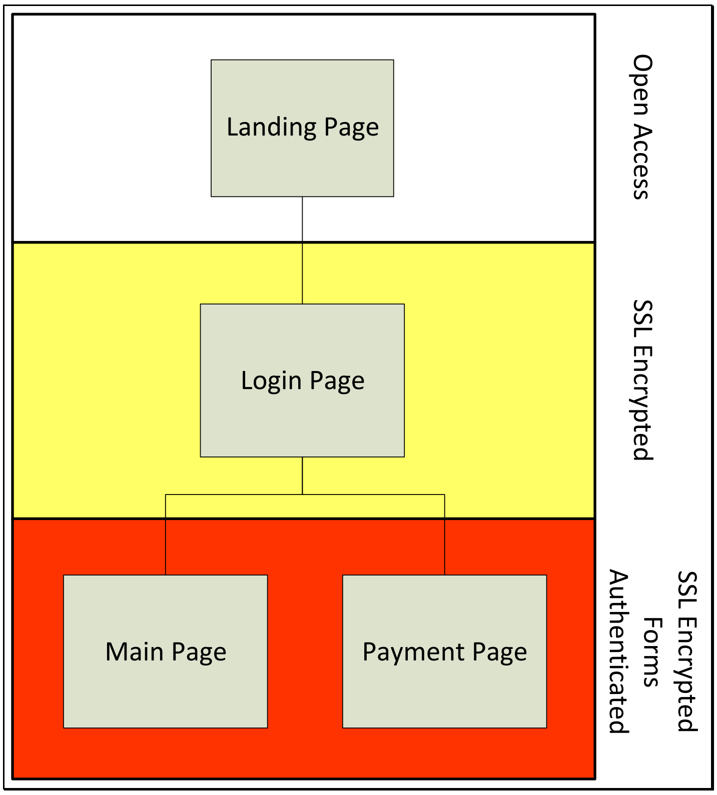


FIGURE 7:13 THE THREE SECURE AREAS OF THE V-BANK SYSTEM

For the V-Bank prototype the SSL encryption is established using a certificate signed by a self-signed certificate authority (CA). The root key of the self-signed certificate authority is specifically installed in the TBOS as a trusted certificate authority. The SSL encrypted pages are also configured to force a web browser to present a client certificate during connection. This client certificate is used by the V-Bank system to identify the connected client.

Once the client is authenticated, then the client can access the SSL Encrypted Forms. A successful login verifies two things:

1. Correct password characters during random authentication code, given the clients public key. The system verifies the password characters with the stored password for the users in the V-Bank system.
2. Integrity check from the clients. The system verifies the integrity of the V-Bank clients to ensure that the correct applications are connecting to the V-Bank system.

If the clients are successfully logged in then the web server is told by the system that the clients passed their forms authentication. Once this is done, the clients are given access to the other areas of the web server.

The web server that was used to implement the V-Bank system was Microsoft Internet Information Server (IIS).

5.1.2 Design Requirements

The authorisation to different areas in the web server is one of the layers in the layered security program. This design requirement is shown in Table 20.

Number	Requirement
16	Online banking systems must implement a layered security program.

TABLE 20: DESIGN REQUIREMENTS FULFILLED BY THE WEB SERVER

5.2 Data Objects

The V-Bank system has been designed to use different classes of objects. Instead of using data variables or entries from a table, the data objects ensure that data integrity is intact when it is created or modified.

The properties of objects are parsed to ensure valid input when the object is created or modified. The use of data objects ensures that the chances of SQL injection is minimised.

The types of objects that can exist in the system include administrative users, banking transactions, approved configurations, beneficiaries, payments, accounts and user profiles.

The data objects can be used by any of the three layers. The first layer that interacts with the user is the Access Layer.

5.2.1 Design Requirements

Table 21 shows that before an object is created in the V-Bank system, (5) the properties of the object is verified to ensure proper syntax was used in the input fields.

(16) The data objects and the syntax checking of the objects implemented is another layer in the layered security program.

Number	Requirement
5	An online banking system should evaluate any input and allow only approved syntax in input fields.
16	Online banking systems must implement a layered security program.

TABLE 21: DESIGN REQUIREMENTS FULFILLED BY THE DATA OBJECTS

5.3 Access Layer

The Access Layer is responsible for presenting information to the user. It is also the access layer against which the user interacts.

The functionality in the Access Layer is achieved through the use of Hypertext Markup Language (HTML), Active Server Pages (ASP), JavaScript and Java Applets.

Apart from the client using the access layer to input information into the system, the access layer also interacts with the IC module on the client. This interaction is described next.

5.3.1 *Interaction with the client*

By default the web services running on a web server do not have access to the resources of a computer belonging to a client. This means that the web application cannot determine whether the computer complies with the configuration standard defined by the bank. The web server also cannot access the user's private key stored on the user's computer so that the client can sign banking transactions created by the user.

Fortunately Java has a way that allows Java applets that has been digitally signed, by a trusted authority, to be executed on the client computer and gain access to resources on the client computer.

Two Java Applets have been created that enables the Access Layer to gain access to the resources on the client computer to enable the digital signing of transactions and the gathering of configuration information for compliance verification. The two Java applets are

- **BankingInitialCheck**. BankingInitialCheck runs a "netstat –tuwp" and "ps –aux" commands on the TBOS. The "netstat –tuwp" lists all the processes that currently has connections open to the network. Once the process that have connections open on the network has been determined it finds the binary files of those processes on the TBOS by executing the "ps –aux" command. Once the files have been found, hashes of the files are created. The name of the file, together with the hash value of the file is then sent to the server. The server can now determine whether these files are in compliance with the banking system's approved configuration.
- **BankingSign**. BankingSign opens the file that contains the user's private key on the client computer. It then uses the private key to sign the transaction details of the client.

The access layer acts as an interface between the client and the business logic of the system. The access layer constructs data objects received from the client via the Java applets or user input.

5.3.2 *Design Requirements*

Table 22 summarises the design requirement fulfilled the access layer.

(6) The access layer presents the ASP pages from the web server to the client's browser, without making use of IFrames.

(16) The access layer is one of the layers implemented by the layered security program.

Number	Requirement
6	The online banking system should not contain any IFrames, nor should it reference code from any site except its own.
16	Online banking systems must implement a layered security program.

TABLE 22: DESIGN REQUIREMENTS FULFILLED BY THE ACCESS LAYER

5.4 Business Layer

The business layer in the system works with the data that was captured by the Access Layer and applies some business rules onto it.

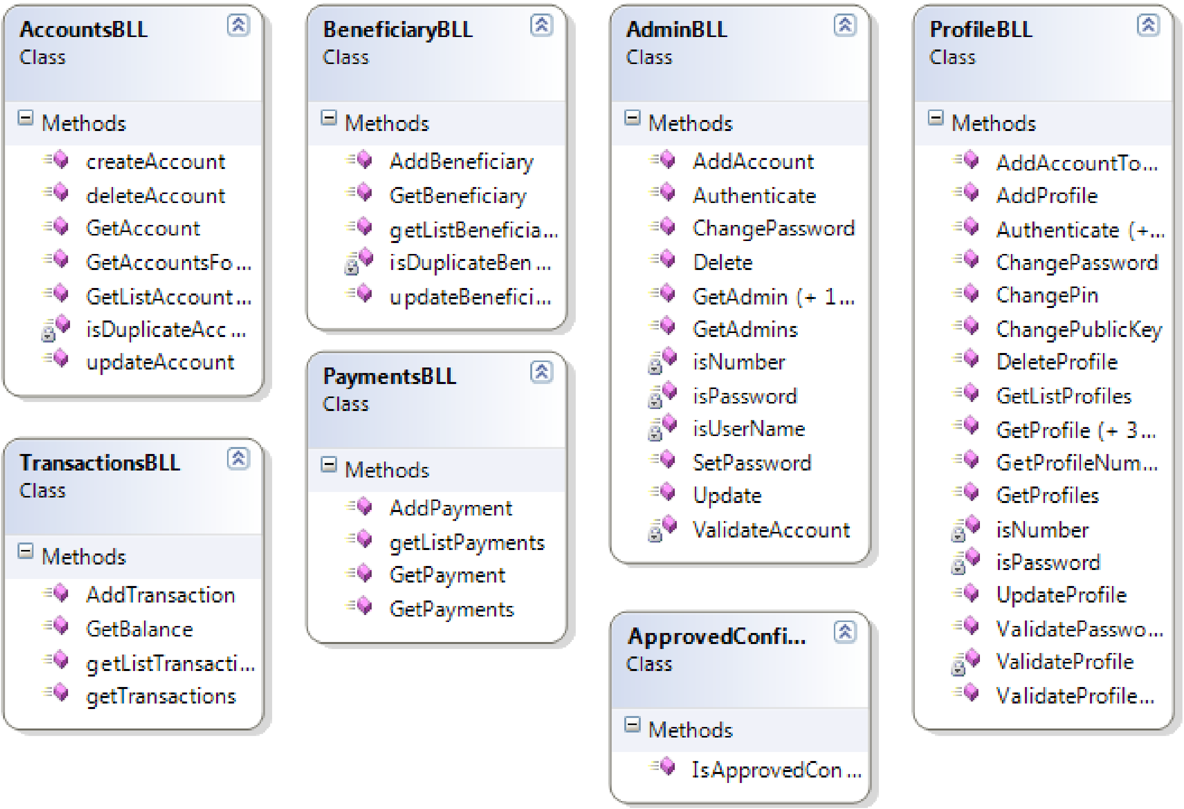


FIGURE 7:14 CLASSES USED BY THE BUSINESS LAYER.

Figure 7:14 is a screenshot of the classes that are in use in the business layer. The access layer uses the business layer to act on the data that is captured in the access layer, or the access layer uses the business layer to display the necessary data to the user.

Example: The ProfileBLL class is used by the access layer to authenticate a user. The access layer first constructs a profile object for the user. This includes at least the user's public key, as presented by the web browser to the web server. The access layer goes through the process of building a random authentication password using the ProfileBLL class. The user types the missing characters for the password and the access layer uses the ProfileBLL class to authenticate the user. The Authenticate method in the ProfileBLL class returns a true or false value that shows whether the user has been authenticated.

The business layer fulfils a very strong role as Execution Service, Authentication Service and Transaction Service. Many of the methods either enable the execution of data tasks against a user's profile, or it authenticates the user or enables transactions to occur between accounts.

The business layer verifies user input to minimise the chances of SQL injection attacks.

5.4.1 Design Requirements

Table 23 summarises the design requirements fulfilled by the business layer.

(5) The business layer verifies properties of data objects before they are added to the data store via the data layer. This ensures the syntax of input fields are double checked by the V-Bank system.

(15) The business layer ensures that access to information in the system is granted according to the role based access control. Administrators will have access to profiles in the banking system, but not the accounts. Users will have access to their own accounts, but not another's, nor can they get access to the profiles defined in the system.

(16) The business layer implements one of the layers of the layered security program.

(17) The authorisation components in the business layer are one of the security services defined by ISO 7498-2.

Number	Requirement
5	An online banking system should evaluate any input and allow only approved syntax in input fields.
15	Online banking systems must implement a RBAC policy to ensure access to user information is controlled.
16	Online banking systems must implement a layered security program.
17	Online banking system should implement information security controls in all five ISO 7498-2 security service categories.

TABLE 23: DESIGN REQUIREMENTS FULFILLED BY THE BUSINESS LAYER

The business layer classes used by the access layer do not understand the type of data store used by the system. The business layer utilises the data layer to retrieve, insert or update data into the data store. The next section describes the data layer.

5.5 Data Layer

The business layer uses a number of generic data layer classes to retrieve, insert or update different data items in the V-Bank system. These generic classes utilise ASP.NET specific classes like table adapters and data sets to interface with the Microsoft SQL Server database.

The data layer is the final layer that makes up the implemented architecture of the V-Bank prototype server-side components.

The data layer acts as a support layer for the Authentication Service, Transaction Service and Execution Service and indirectly to the Interface Service via the business layer.

The data layer does not fulfil a specific design requirement. The data layer is mentioned here to complete the picture of the server side components even though it has no direct effect on the design requirements.

It is also assumed that the data base system of an online banking system is highly secure, with proper auditing and access control. Securing of database systems are outside the scope of this dissertation.

6 Design requirements evaluation

Table 24 summarises the design requirements identified in the dissertation and maps it against the components implemented in the V-Bank prototype that fulfils the requirements.

The components are grouped together in a very light grey that depicts the client-side components. The little darker grey in the middle of the table groups together the components configured on the client and server. The darkest grey, on the right hand side of the table is the server-side components.

A quick scan of the table, from top to bottom, verifies that all the design requirements are fulfilled by at least one of the design components. In some cases many components fulfil the design requirements. This is the case for the design requirement for the keylogger, layered security program and ISO 7498-2 security services.

The layered security program and ISO 7498-2 security services require multiple components because of the nature of the requirement. The fact that the keylogger requirement is fulfilled by multiple components proves how important protection against keyloggers is in the design of the system.

A scan from left to right shows that all the components help to fulfil the design requirements, except for the data layer. As already mentioned, the data layer itself is only mentioned in the component configuration because this is the component where all the data of the system is stored. It is also assumed that the data layer is highly secure and outside the scope of the dissertation.

The only design requirement that is not properly fulfilled is requirement 11, that states that an online banking system must have the ability to be updated when a vulnerability is identified. The V-Bank system can be updated when a vulnerability gets identified, but the current prototype still uses a manual method to update the TBOS image. In a commercial environment this requirement needs to be better fulfilled to include the requirement that the update should happen automatically and securely.

Requirement number	Requirements	Trusted Banking Operating System	Virtual Machine Monitor	Persistent image	Operating system configuration	Operating system authentication	Private Key component	Authentication mechanism	Dedicated secure channel	Integrity check system	Web Server	Data Objects	Access Layer	Business Layer	Data Layer
1	The client operating environment should be persistent. It should not be possible to change any file or configuration in the client operating environment			✓											
2	The client operating environment should be isolated on the network and only be allowed to connect to the banking system environment.								✓						
3	It should not be possible to modify the Internet browser used by an online banking system.			✓											
4	Online banking systems should not allow the Internet Browser from opening multiple pages and the transaction component in an online banking system should re-authenticate the user, before committing important transactions.				✓										
5	An online banking system should evaluate any input and allow only approved syntax in input fields.											✓		✓	
6	The online banking system should not contain any IFrames, nor should it reference code from any site except its own.												✓		
7	Name resolution should not affect the connection inside an online banking system								✓						
8	An online banking system should only allow connections to and from approved clients and servers.									✓					
9	A keylogger should not be allowed inside an online banking system or should minimize the effect it may have if present.			✓	✓		✓	✓	✓						
10	An online banking environment should minimise the chances of an attacker using stolen information to gain access to a customer's online banking profile.					✓				✓					
11	Online banking systems must have the ability to be updated when software vulnerabilities are detected.	✓		✓											
12	Online banking systems must ensure TLS 1.2 channel security								✓						
13	Online banking systems must have a visual trust anchor.	✓						✓							
14	Online banking systems must properly authenticate users before allowing access.							✓							
15	Online banking systems must implement a RBAC policy to ensure access to user information is controlled.													✓	
16	Online banking systems must implement a layered security program.				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
17	Online banking system should implement information security controls in all five ISO 7498-2 security service categories.					✓	✓	✓	✓	✓				✓	
18	The bare minimum guest tools should be enabled in a client-side virtual machine environment.		✓												
19	The VMM used by the online banking system should have limited introspection capabilities		✓												
20	Snapshots of a client-side operating virtual machine environment should not be possible		✓												
21	An online banking system should be designed to include not just a managed server environment, but also a managed client environment.	✓								✓					
22	The client-side online banking system should be sandboxed from the rest of the client operating environment	✓	✓												

TABLE 24: SUMMARY OF DESIGN REQUIREMENTS AND COMPONENTS

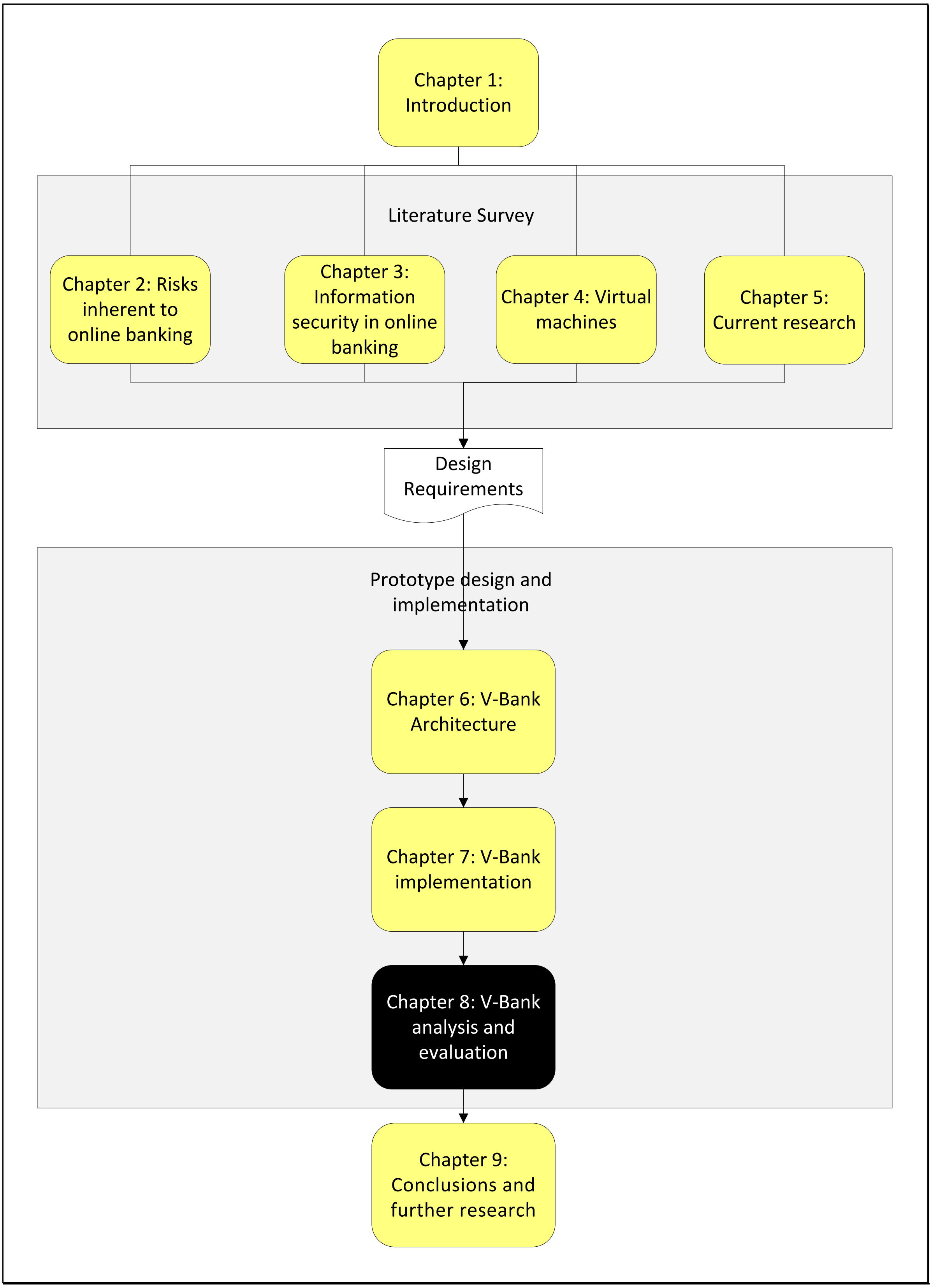
7 Conclusion

This chapter explained how the V-Bank system was implemented. The implementation of the V-Bank system happens in different components. The different components can be grouped into client-side, client and server-side and server-side components.

The components were selected and implemented to ensure the design requirements identified by the research have been fulfilled.

More details of how specific components were configured can be found in the Appendix A.

In the next chapter the V-Bank prototype system is evaluated and highlights the strengths and weaknesses of the system.


Chapter 1: Introduction
Literature Survey
Chapter 2: Risks inherent to online banking
Chapter 3: Information security in online banking
Chapter 4: Virtual machines
Chapter 5: Current research
Design Requirements
Prototype design and implementation
Chapter 6: V-Bank Architecture
Chapter 7: V-Bank implementation
Chapter 8: V-Bank analysis and evaluation
Chapter 9: Conclusions and further research

Chapter 8: V-Bank Prototype Analysis and Evaluation

1 Introduction

This chapter evaluates the prototype and where applicable try and determine if a weakness is the result of a design requirement or implementation problem.

The chapter starts by describing how the prototype works. The prototype description is conducted by using use cases for the bank, the banking user and an attacker.

The chapter then moves on to comment on the viability of commercially implementing the V-Bank architecture

2 Prototype analysis

This section describes how the V-Bank prototype behaves in the following three areas:

- **Bank**. The bank starts by creating the TBOS image and gives it to the online banking user.
- **Banking user**. The online banking user uses the V-Bank system for online banking. It starts with explaining how the banking user configures his host computer and continues with use cases up to the point where the banking user pays a beneficiary.
- **Attacker**. Three attacks are discussed. The attacks include phishing attacks, online attacks against the TBOS and using stolen information to try and gain access to the V-Bank system.

A video demonstration of the V-Bank prototype accompanies the dissertation on an accompanied CD-ROM. The video demonstration can also be downloaded from https://docs.google.com/file/d/0B8o50-4OdA6fNGp5NzhBSGJTN1U/edit?usp=sharing . The file downloaded is videos.zip, which can be extracted into a folder. The videos can then be viewed by opening Index.htm from the extracted source.

It took the author of this dissertation about 380 hours to design, develop and configure the V-Bank prototype. The area that took the most time was the testing that was done to decide between the bottom-up approach, or the top-down approach for developing the TBOS.

2.1 Bank

There is only one use case necessary to describe what the bank does to generate the TBOS image. The use case is defined in Figure 8:1

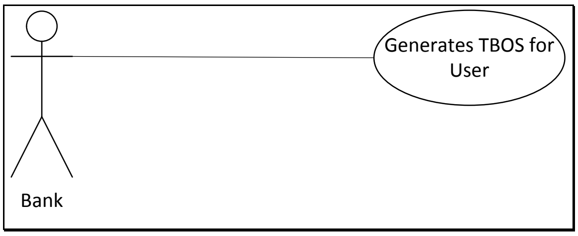


FIGURE 8:1 USE CASE FOR BANK

2.1.1 Generates TBOS for User

This section will not describe the detailed steps of creating the TBOS. Details of the steps can be found in Appendix A.

The basic steps for creating the TBOS from the bank are:

1. Create the banking profile, which includes the profile number and initial password for the user.
2. Create X.509 certificate for user.
3. Link the public key in the X.509 certificate for the user to the banking profile.
4. Load the certificate in the web server for mutual authentication.
5. Create username and password for the TBOS.
6. Install X.509 certificate in TBOS Internet Browser and Java control panel.
7. Store user's private key as file on the TBOS file system.
8. Generate LiveCD image of the TBOS.
9. Load LiveCD image on USB memory stick.

The outcome of the process is that the bank gives the banking user a memory stick with a LiveCD image file.

The bank can also copy the LiveCD image of the TBOS on a protected USB memory stick. A protected USB memory stick can be protected so that the LiveCD image cannot be copied from the LiveCD image. This was not implemented as part of the V-Bank prototype.

2.2 Banking User

The banking user fetches the USB memory stick from the bank and has to prepare his computer to run the TBOS. Figure 8:2 shows the use cases for the banking user. The use cases are described next.

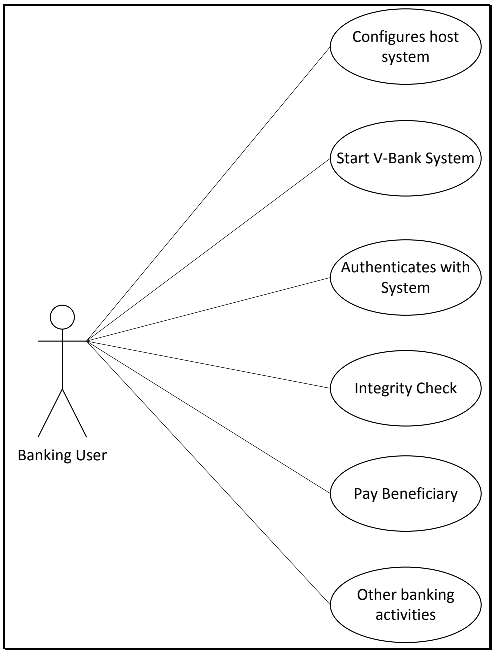
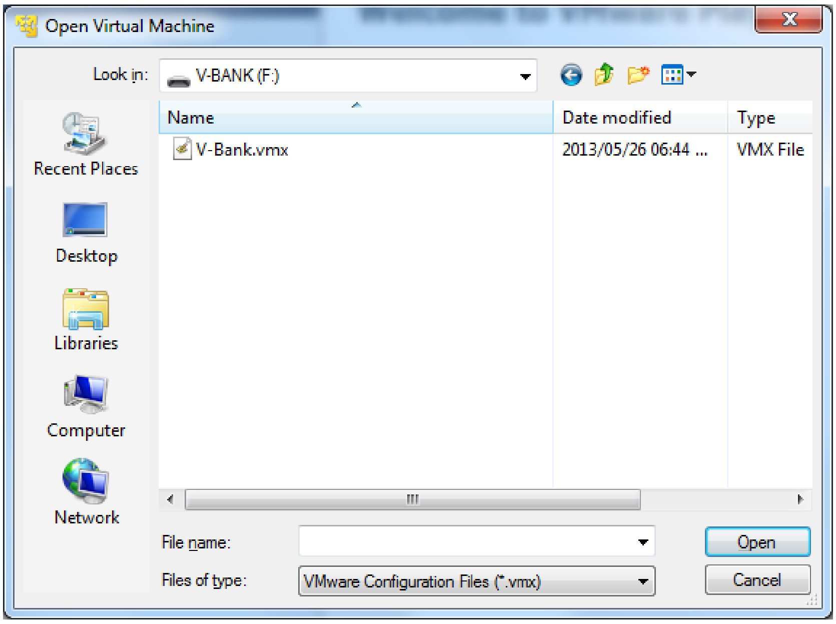


FIGURE 8:2 USE CASES FOR BANKING USER

2.2.1 *Configures host system*

The user has to install VMware Player in his host operating system. The source files can be downloaded from the VMware website and the installation is very easy. The host operating system requires a reboot once the VMware Player is installed. This section will not describe the installation of the VMware Player.

To configure the host system further, the user inserts the USB memory stick into a USB port. The banking user starts the VMware Player. On the VMware Player, the banking user clicks on the File Menu and selects the Open a Virtual Machine. In the file navigation box, the user has to navigate to the USB memory stick and select the V-Bank.vmx file. Figure 8:3 shows the V-Bank.vmx file in the USB memory stick.



Open Virtual Machine

Look in: V-BANK (F:)

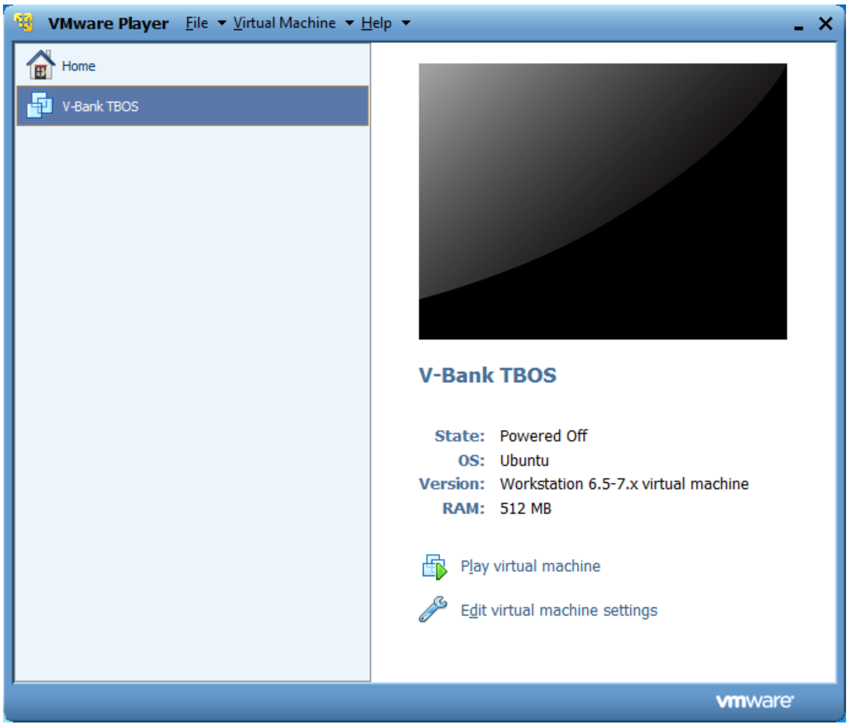
Name	Date modified	Type
V-Bank.vmx	2013/05/26 06:44 ...	VMX File

File name:

Files of type: VMware Configuration Files (*.vmx)

FIGURE 8:3 SELECT V-BANK.VMX FILE FROM USB

Figure 8:4 shows how the VMware Player will look once the TBOS configuration has been imported into the VMware Player.



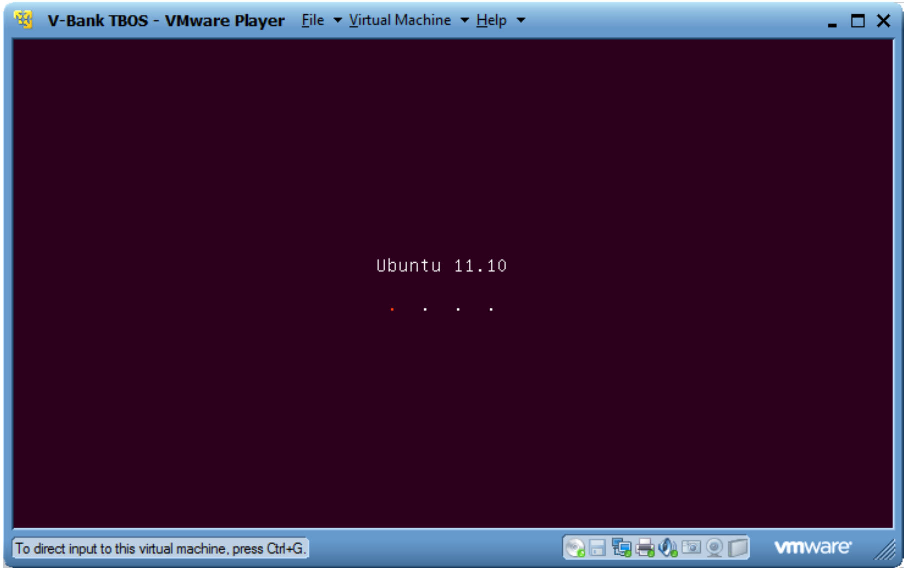
State: Powered Off
OS: Ubuntu
Version: Workstation 6.5-7.x virtual machine
RAM: 512 MB

FIGURE 8:4 THE CONFIGURED VMWARE PLAYER

2.2.2 *Start V-Bank System*

The banking user starts the V-Bank system by ensuring the USB memory stick is inserted in a USB port on his host computer. The user open the VMware Player and click on the "Play virtual machine" for the V-Bank TBOS virtual machine.

Once the user has clicked on the "Play virtual machine", the TBOS starts up. Figure 8:5 shows how it will look while the TBOS is starting up.



Ubuntu 11.10

To direct input to this virtual machine, press Ctrl+G.

FIGURE 8:5 TBOS STARTING UP

Once the TBOS has started the TBOS will display a screen for the user to type his username. Figure 8:6 shows how the screen will look once the TBOS has started.

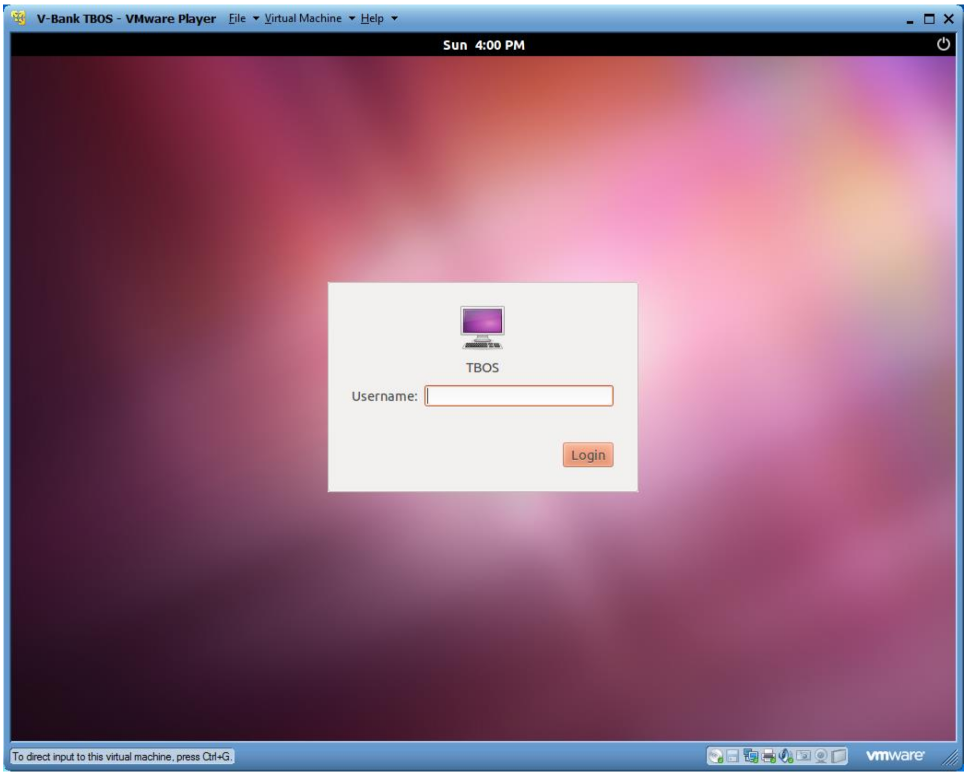

V-Bank TBOS - VMware Player
File
Virtual Machine
Help
Sun 4:00 PM
TBOS
Username:
Login
To direct input to this virtual machine, press Ctrl+G.
vmware

FIGURE 8:6 TBOS LOGIN SCREEN

2.2.3 *Authenticates with system*

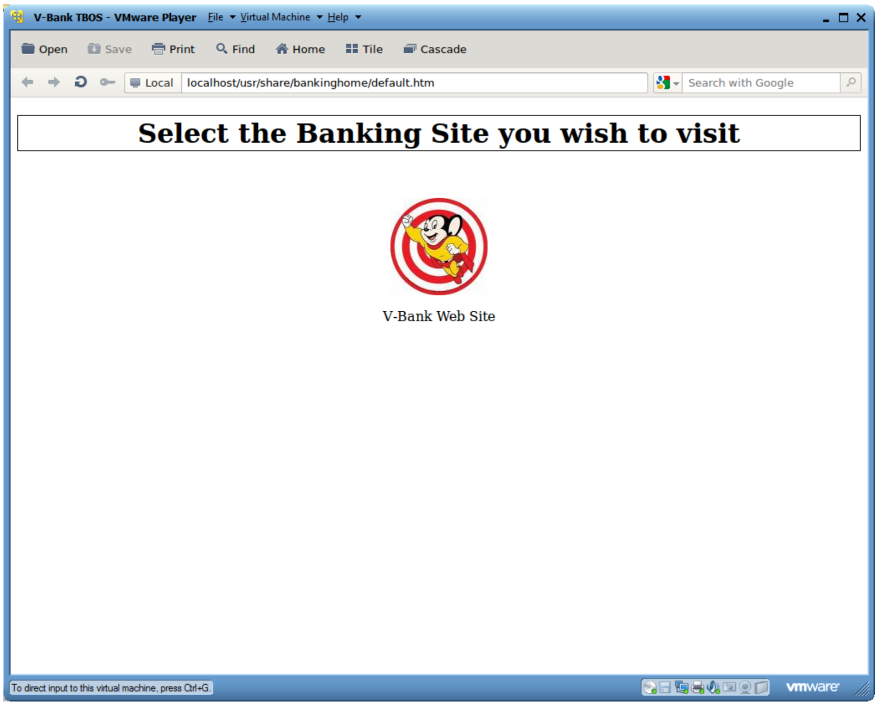
The user authenticates with the V-Bank system in a number of areas they are:

- Linux Login
- V-Bank user authentication

Each of these areas are now briefly described

The user has to type the TBOS username and password given to him by the bank to log into the Linux operating system

Once the user has logged in, the TBOS opens a landing page. For the prototype the landing page was a file located on the TBOS. Part of the tests, not included for the dissertation was trying to use the TBOS to connect to other commercial banks. The landing page contains links to the various banks. For prototype testing and demonstration purposes only the V-Bank system was left as a valid link. The V-Bank landing page is shown in Figure 8:7.

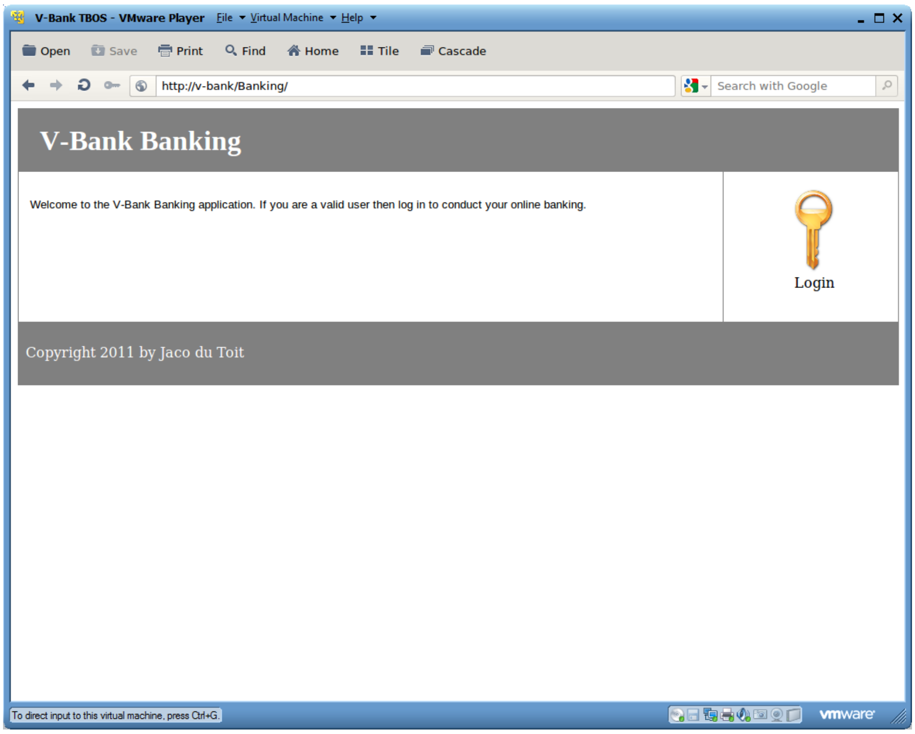

V-Bank TBOS - VMware Player
File
Virtual Machine
Help
Open
Save
Print
Find
Home
Tile
Cascade
Local
localhost/usr/share/bankinghome/default.htm
Search with Google
To direct input to this virtual machine, press Ctrl+G.
vmware

Select the Banking Site you wish to visit

V-Bank Web Site

FIGURE 8:7 V-BANK LANDING PAGE

The user selects the V-Bank Web site from the landing page and gets redirected to the server-side V-Bank web site. The V-Bank Web site is by default an unsecured web page as shown in Figure 8:8.



V-Bank Banking

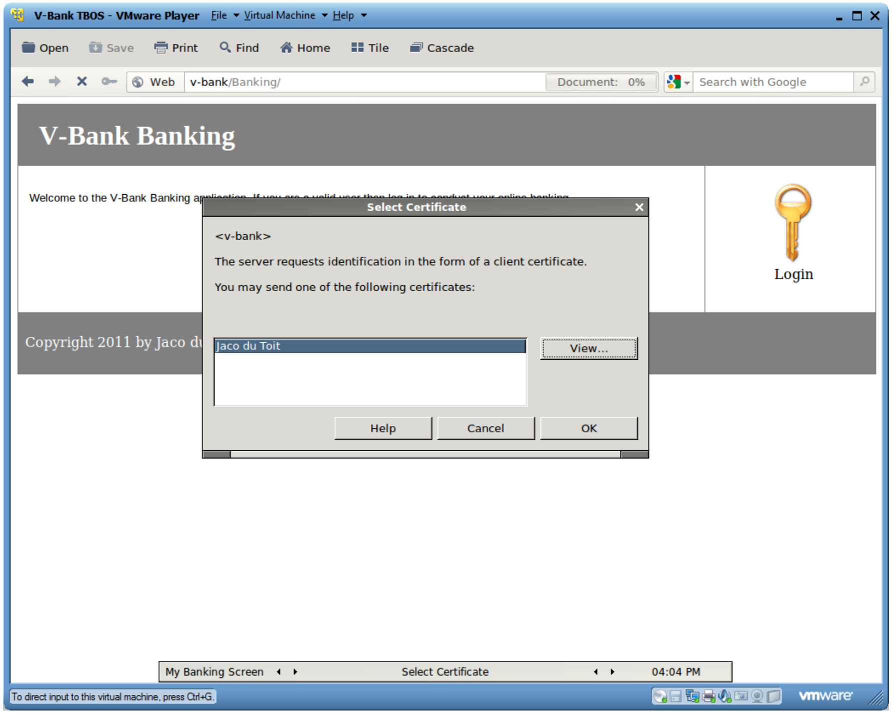
Welcome to the V-Bank Banking application. If you are a valid user then log in to conduct your online banking.

Login

Copyright 2011 by Jaco du Toit

FIGURE 8:8 UNSECURED V-BANK WEB PAGE

The user must now click on the Login icon on the web page. The web browser is now redirected to a secure web page. The first thing the user experience is that the user must select the user's certificate stored in the TBOS Internet browser. Figure 8:9 shows the certificate presented to the user that is preloaded into the TBOS Internet browser.



v-bank/Banking/

V-Bank Banking

Welcome to the V-Bank Banking a

Copyright 2011 by Jaco d

Select Certificate

<v-bank>

The server requests identification in the form of a client certificate.

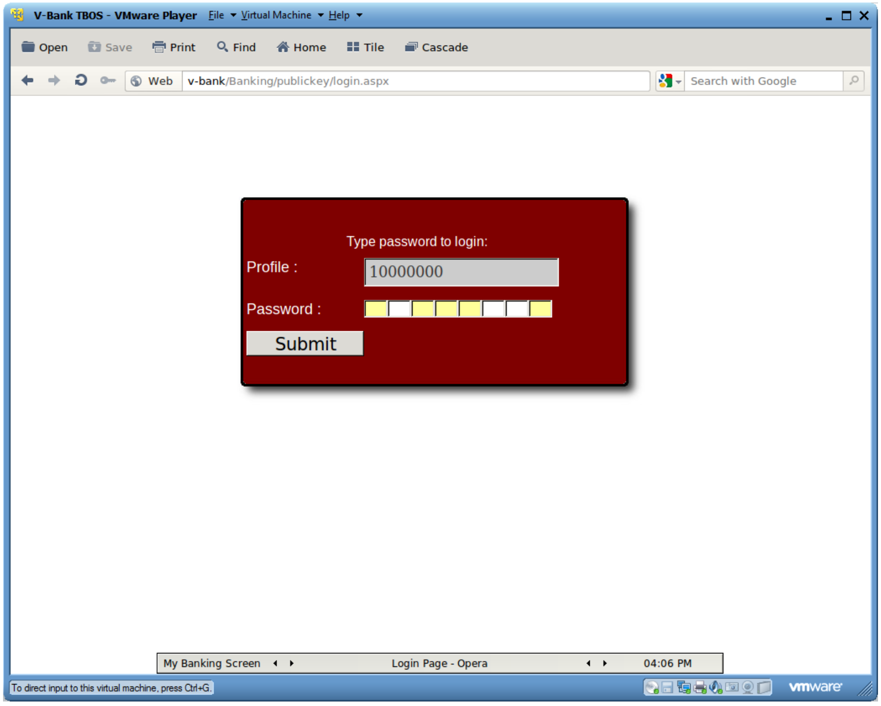
You may send one of the following certificates:

Jaco du Toit

View... Help Cancel OK

FIGURE 8:9 CERTIFICATE SELECTION

The certificate for the client identifies the client to the server and the V-Bank system automatically loads the user's profile number into the system. Figure 8:10 shows the V-Bank login page. The login page identified the user from the user's certificate as profile 10000000.



v-bank/Banking/publickey/login.aspx

Type password to login:

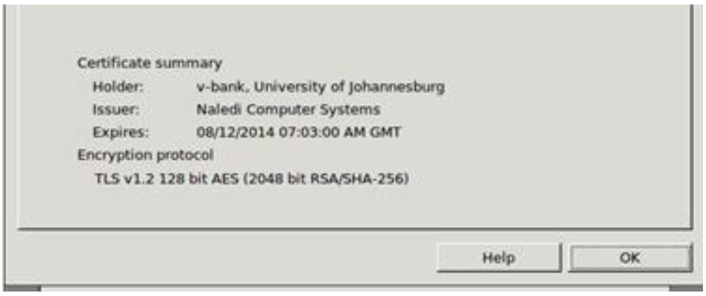
Profile : 10000000

Password :

Submit

FIGURE 8:10 THE V-BANK LOGIN PAGE

All the pages from this point onwards in the V-Bank system is encrypted using TLS 1.2. Figure 8:11 displays a summary of the connection information displayed by the Opera browser on the TBOS. It confirms the TLS 1.2 encryption protocol.



Certificate summary
Holder: v-bank, University of Johannesburg
Issuer: Naledi Computer Systems
Expires: 08/12/2014 07:03:00 AM GMT
Encryption protocol
TLS v1.2 128 bit AES (2048 bit RSA/SHA-256)

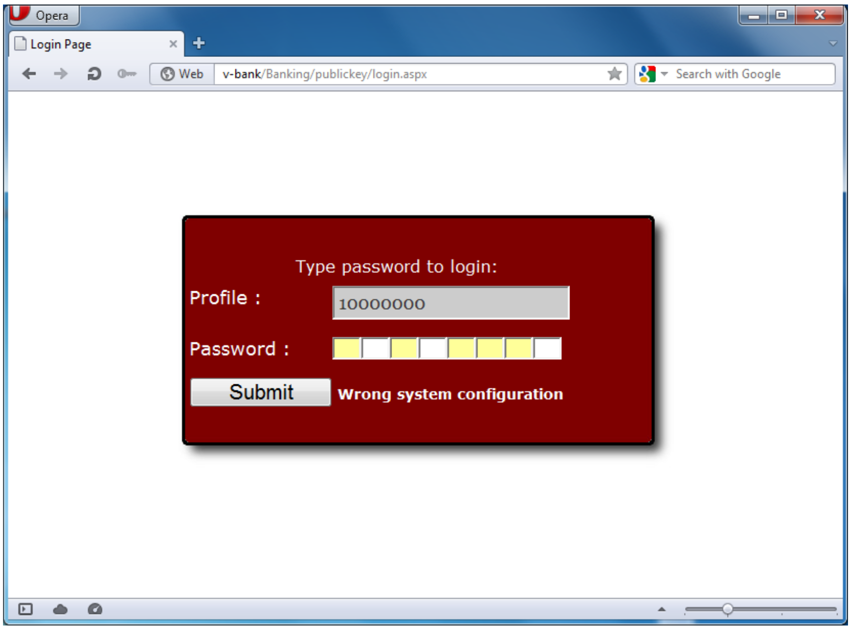
FIGURE 8:11 SECURE CHANNEL USING TLS 1.2 ENCRYPTION

The user now enters the random authentication code. This means he has to type certain characters for the V-Bank password. The random authentication code minimizes keylogger attacks on the system.

2.2.4 *Integrity check*

Once the user has typed the missing characters for his password, the system goes through the IC module. The system does this quietly and cannot be discerned from the user interface. The IC module confirms the browser and Java configuration of the TBOS and any other processes that have a connection open to the TBOS. If there is an unauthorized process that has a connection open to the V-Bank system then the IC module will fail and the user will not be allowed access to the V-Bank system.

Figure 8:12 shows how the screen will look if a connection is opened by an unauthorised configuration. A message stating that access is denied because of wrong system configuration is displayed to the user.



Type password to login:
Profile : 10000000
Password :
Wrong system configuration

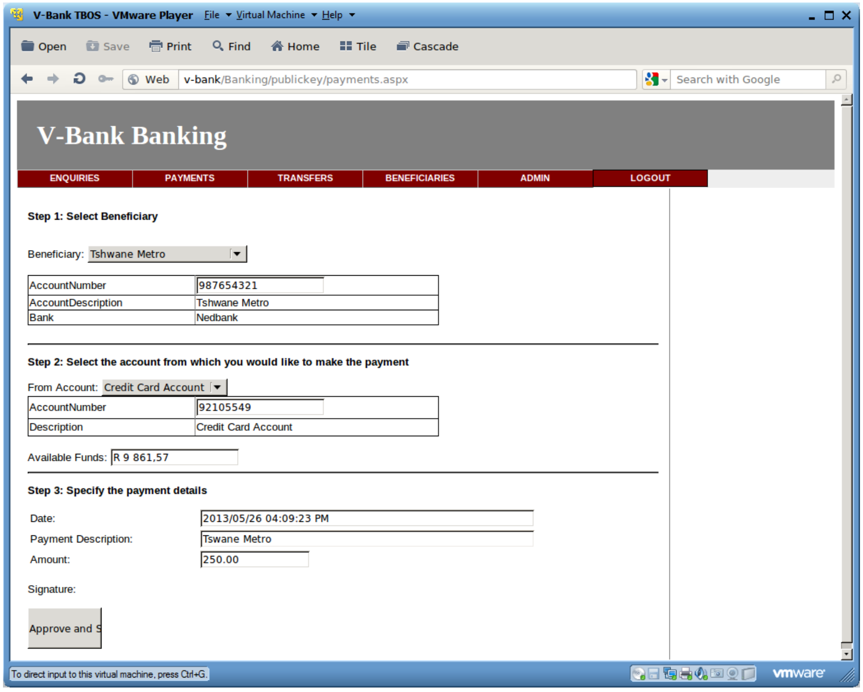
FIGURE 8:12 IC MODULE

2.2.5 *Pay beneficiary*

After the client authenticated with the V-Bank system and the V-Bank system confirmed the integrity of the TBOS, then the use the system as a normal online banking system. A special case is where the banking user pays a beneficiary.

To ensure another layer of security the banking user signs beneficiary payment transactions using his private key. This ensures non-repudiation and ensures that only the client can sign the transaction using his private key.

Figure 8:13 is the beneficiary payment screen. The banking user fills in the relevant detail and selects the beneficiary.



v-bank/Banking/publickey/payments.aspx

V-Bank Banking

Step 1: Select Beneficiary

Beneficiary: Tshwane Metro

AccountNumber	987654321
AccountDescription	Tshwane Metro
Bank	Nedbank

Step 2: Select the account from which you would like to make the payment

From Account: Credit Card Account

AccountNumber	92105549
Description	Credit Card Account

Available Funds: R 9 861,57

Step 3: Specify the payment details

Date: 2013/05/26 04:09:23 PM

Payment Description: Tswane Metro

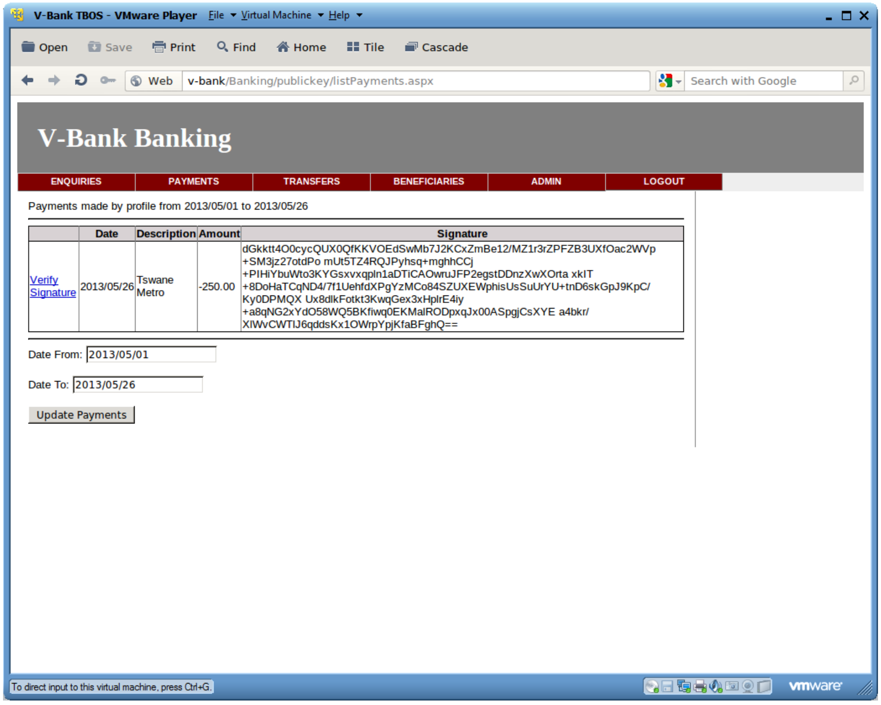
Amount: 250.00

Signature:

FIGURE 8:13 USER GENERATES A PAYMENT TO A BENEFICIARY

If the banking user is happy with the information entered on the page, he clicks on the Approve and Sign button. The V-Bank system downloads and starts a Java applet that opens the private key for the user, uses the information in the banking transaction to generate a hash and signs the hash. The process happens quietly without the user knowing that it is happening. The password to open the private key is embedded in the Java applet and stored on the server.

Once the banking transaction is signed the user is presented with the information in Figure 8:14.


V-Bank TBOS - VMware Player File Virtual Machine Help
Open Save Print Find Home Tile Cascade
Web v-bank/Banking/publickey/listPayments.aspx
Search with Google
V-Bank Banking
ENQUIRIES PAYMENTS TRANSFERS BENEFICIARIES ADMIN LOGOUT
Update Payments
To direct input to this virtual machine, press Ctrl+G.
vmware

Payments made by profile from 2013/05/01 to 2013/05/26

	Date	Description	Amount	Signature
Verify Signature	2013/05/26	Tswane Metro	-250.00	dGkktt4O0cycQUX0QfKKVOEdSwMb7J2KCxZmBe12/MZ1r3rZPFZB3UXfOac2WVp +SM3jz27otdPo mUt5TZ4RQJPyhsq+mghhCCj +PIHiYbuWto3KYGsxvxqpln1aDTiCAOwruJFP2egstDDnzXwXOrta xkIT +8DoHaTCqND4/7f1UehfdXPgYzMCo84SZUXEWphisUsSuUrYU+tnD6skGpJ9KpC/ Ky0DPMQX Ux8dlkFotkt3KwqGex3xHplrE4iy +a8qNG2xYdO58WQ5BKfiwq0EKMalRODpxqJx00ASpgjCsXYE a4bkr/ XIWvCWTIJ6qddsKx1OWrpYpjKfaBFghQ==

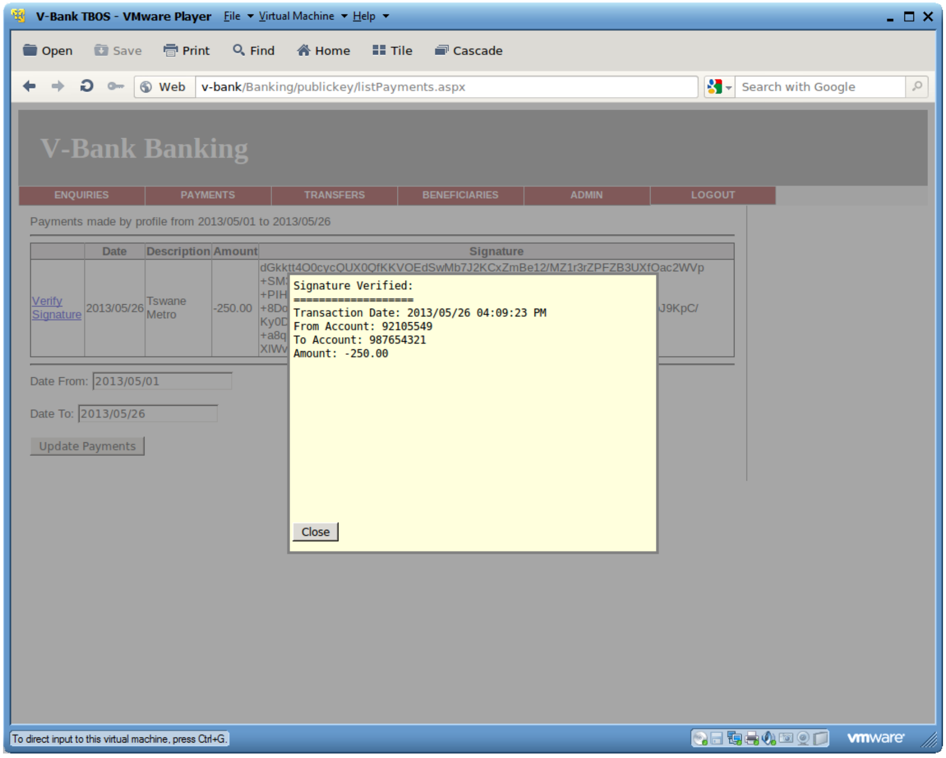
Date From: 2013/05/01

Date To: 2013/05/26

FIGURE 8:14 A LIST OF PAYMENTS

The user can get a list of payments by manipulating the Date From and Date To fields. The banking user can verify the signature on a payment by click on the Verify Signature link.

If the signature on the payment transaction is verified the user is presented with a screen shown in Figure 8:15.


V-Bank TBOS - VMware Player File Virtual Machine Help
Open Save Print Find Home Tile Cascade
Web v-bank/Banking/publickey/listPayments.aspx
Search with Google
V-Bank Banking
ENQUIRIES PAYMENTS TRANSFERS BENEFICIARIES ADMIN LOGOUT
Update Payments
Close
To direct input to this virtual machine, press Ctrl+G.
vmware

```
Signature Verified:
====================
Transaction Date: 2013/05/26 04:09:23 PM
From Account: 92105549
To Account: 987654321
Amount: -250.00
```

FIGURE 8:15 VERIFIED SIGNATURE ON PAYMENT

2.2.6 Other banking activities

The previous use cases for the banking user described the V-Bank prototype and the security features built into the prototype. The prototype has also been extended to include some basic online banking functionality, but is not developed to be a full blown online banking system with all the possible features.

The goal of the prototype is to test the security features and design requirement defined in the dissertation. This section mentions the other functionality that was built into the prototype, but is not enhanced because it does not add value to the design requirements of the dissertation.

The basic functionality includes:

- The viewing of statements. From the enquiries menu the banking user has the option to view a statement for one of his linked accounts. The banking user can change the time period to display transactions.
- Creating of beneficiaries. Creating of beneficiaries has not been extended to include out of band authorisation by the banking user. Creating of a beneficiary is a simple as providing the beneficiary name, bank account number and the description.
- Changing of password. The banking user can change his random authentication code password from the Admin menu.

2.3 Attacker

The previous sections described how the V-Bank prototype would be used without malicious intent. In the research conducted by Guan, et al. (2012), five important banking service attacks are identified. For each attack a use case is defined and discussed in context of the V-Bank prototype. Figure 8:16 shows the use cases for the five important banking service attacks.

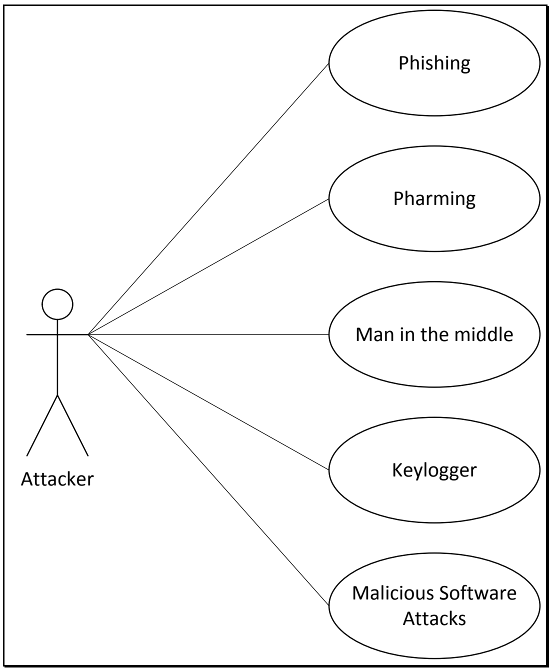


FIGURE 8:16 USE CASES FOR ATTACKER

Each of these use cases are discussed next.

2.3.1 Phishing

In a phishing attack an attacker generates an e-mail with a URL in order to steal the user's online banking credentials (Nilsson, et al., 2005).

The V-Bank system is protected from phishing attacks because the V-Bank system is dependent on the banking user providing a certificate before authentication so that the V-Bank system can identify the user.

The certificate is also only available in the TBOS this means that even if a fraudulent web site prompts the user for a certificate, the user will not be able to provide the certificate from his host operating system, from where he opened the link.

The TBOS is also protected through a firewall to only allow connection to the V-Bank site, which means that the TBOS cannot provide the certificate information to any site other than the V-Bank site.

2.3.2 Pharming

Pharming redirects user's to phishing sites through DNS contamination (Li & Schmitz, 2009).

The TBOS is protected from pharming attacks because of the firewall allowing network traffic only to and from the V-Bank server. The TBOS also does not rely on external DNS to determine the IP address of the V-Bank server.

If the host operating system is infected then the TBOS will still connect to the V-Bank system because of the TBOS independence of DNS.

2.3.3 Man-in-the-middle

With a man-in-the-middle attack, an attacker captures information sent from the client the online banking system, by intercepting and sending things on behalf of the user.

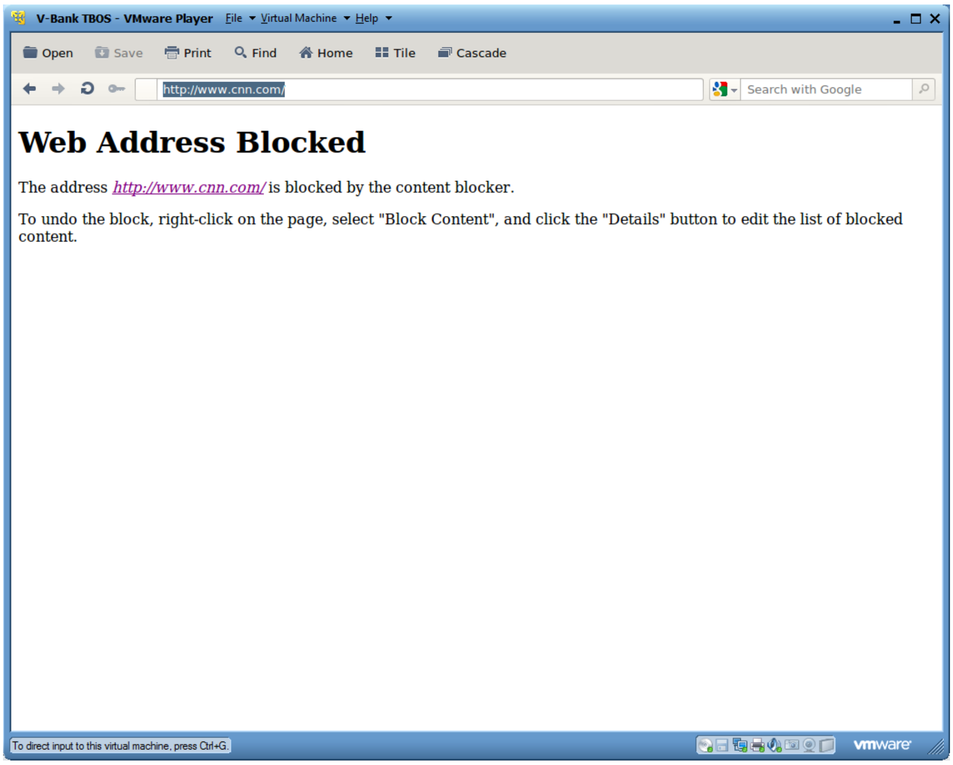
Cheng, et al. (2010) mentions that two-way authentication is an effective method to prevent HTTPS hijacking. The V-Bank system requires both the server and the client authenticates in the TLS 1.2 encrypted channel.

2.3.4 Keylogger

The V-Bank system was tested in the case of a keylogger in the TBOS and in the host operating system.

The TBOS does not allow a keylogger to be installed. The user does not have rights to install the keylogger, nor is it possible for the user to download anything to the TBOS.

Figure 8:17 shows how the browser stops the user from navigating to any other site, other than the V-Bank site. The firewall in the TBOS also stops any network attacks from connecting to the TBOS, potentially exploiting a vulnerability in the TBOS to install the keylogger.



Web Address Blocked

The address http://www.cnn.com/ is blocked by the content blocker.

To undo the block, right-click on the page, select "Block Content", and click the "Details" button to edit the list of blocked content.

FIGURE 8:17 SINGLE PURPOSE OPERATING SYSTEM

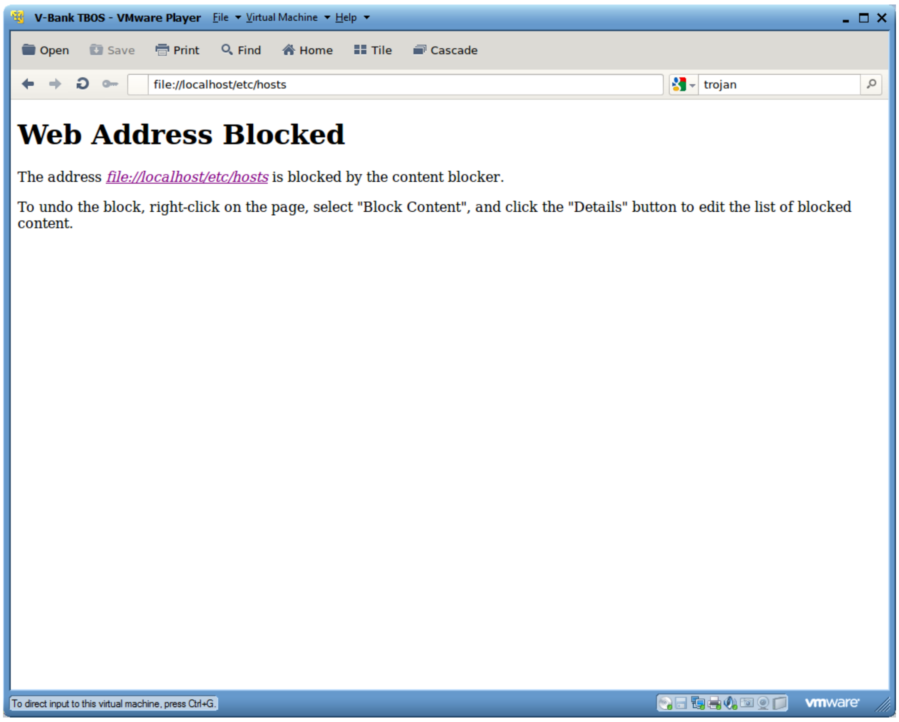
In the case where the keylogger is either a hardware keylogger, or is installed on the host then the keylogger could capture different login information. The first information that was recorded was the Linux username and password typed by the banking user. The second set of information was the missing characters in the random authentication code.

If we assume the attacker has this information then the attacker will still require the actual TBOS on the USB memory stick, before he can gain access to the user's online banking system.

2.3.5 Malicious Software Attacks

Malicious software attacks can target the host operating system and the TBOS. It has been shown that the TBOS is very well protected against malicious software attacks. The first line of defence in the TBOS is the firewall that only allows connections to the V-Bank site.

The second line of defence is the locked down interface and user rights. Users cannot install new software in the TBOS, either accidentally, or maliciously. Figure 8:18 shows that a user tried to access the hosts file on the local system through the user interface, but was denied access.


V-Bank TBOS - VMware Player
File
Virtual Machine
Help
Open
Save
Print
Find
Home
Tile
Cascade
file://localhost/etc/hosts
trojan
To direct input to this virtual machine, press Ctrl+G.
vmware

Web Address Blocked

The address *file://localhost/etc/hosts* is blocked by the content blocker.

To undo the block, right-click on the page, select "Block Content", and click the "Details" button to edit the list of blocked content.

FIGURE 8:18 LOCKED DOWN INTERFACE

The last line of defence is the persistent nature of the TBOS. If for some reason the TBOS gets infected by malicious software, then the infection will be removed once the system is rebooted.

It has also been shown that even if the host operating system is infected that it has little to no effect on the TBOS, because the TBOS is isolated from the host operating system.

This section evaluated and demonstrated how the prototype works for the bank, banking user and in some attacks.

The next section describes some general comments regarding the V-Bank system

3 General comments

The previous section described how the V-Bank system functions. This section looks at the strengths and weaknesses that is identified. The strengths is first mentioned and then the weaknesses.

3.1 Strengths

The following strengths were identified for the V-Bank prototype

- IC System
- Known TBOS configuration
- Public \ Private key encryption
- Firewalled communication channel
- Virtual machine implementation

Each of these strengths are discussed in the following sections

3.1.1 IC System

The bank can determine for each client that connects to the banking site whether the integrity of the client's software modules is intact.

3.1.2 Known TBOS configuration

In the V-Bank prototype the bank is always aware of the exact configuration of the TBOS that connects to the server-side system. The bank can determine how this configuration should look like and what software is part of the banking session when the user and system has been authenticated.

The known configuration also gives banks full control of what can execute on the TBOS during a banking session. This gives the banks flexibility to increase the configuration verification that runs on the V-Bank prototype.

3.1.3 Public \ Private key encryption

The secure communication channel utilises public\private key encryption. The biggest advantage to this is that the user is identified using his private key and not an identifier that can be spoofed by an attacker.

The public\private key encryption ensures that the access is controlled through "something the user has", instead of just "something the user knows". Without physically having access to the V-Bank TBOS, which contains the user's private key, an attacker cannot gain access to a user's banking information.

3.1.4 Firewalled communication channel

The firewalled communication channel helps ensure that the TBOS connect only to the V-Bank server-side system. This minimises the chances of a man-in-the-middle, or even man-in-the-browser attacks.

3.1.5 Virtual machine implementation

The TBOS runs as a virtual machine, isolating and giving a level of protection to the system. The TBOS can also be easily deployed, because it is a virtual machine and does not require very complex setup instructions.

3.2 Weaknesses

Now that some of the strengths of the system have been highlighted the weaknesses are discussed. The weaknesses, as identified by the author are:

- Software based private key
- Update unfriendly
- Static password
- Administrative overhead
- Less user friendly

Each of these weaknesses are discussed in the following sections

3.2.1 Software based private key

The user's private key is embedded inside the key store of the TBOS browser. This makes it possible for an attacker that gains access to the user's TBOS and have the user's or administrator's password to export the user's key and import it into his own browser.

Once this has occurred, the attacker is able to gain access to the user's banking information, provided that the attacker also has knowledge of the user's password that is part of the random authentication code and has a copy of the approved TBOS configuration.

3.2.2 Update unfriendly

The persistent nature of the TBOS makes it difficult for the bank to update the system if new security patches have to be deployed to the client's TBOS.

3.2.3 Static password

The persistent nature of the TBOS makes it impossible for the user to change his password for the TBOS. Any password changes for the TBOS are lost when the TBOS have rebooted.

If the bank also ships each TBOS with the same administrative password then an attacker that steals the password for the one administrator account can gain access to all the user's TBOS systems.

3.2.4 Administrative overhead

Each banking customer must get a custom built TBOS. This means that if the bank cannot automate the customisation required for each TBOS, it becomes too cumbersome and too expensive in human resources.

Figure 8:19 shows the high-level steps required in the process of creating a TBOS for a client. The steps described in the yellow section are required for each customer. The green areas are user independent and are the same for all of customers. A system needs to be developed that can automate the steps and link them together to quickly and easily create a TBOS per customer. The creation of such an automation system is not part of this study.

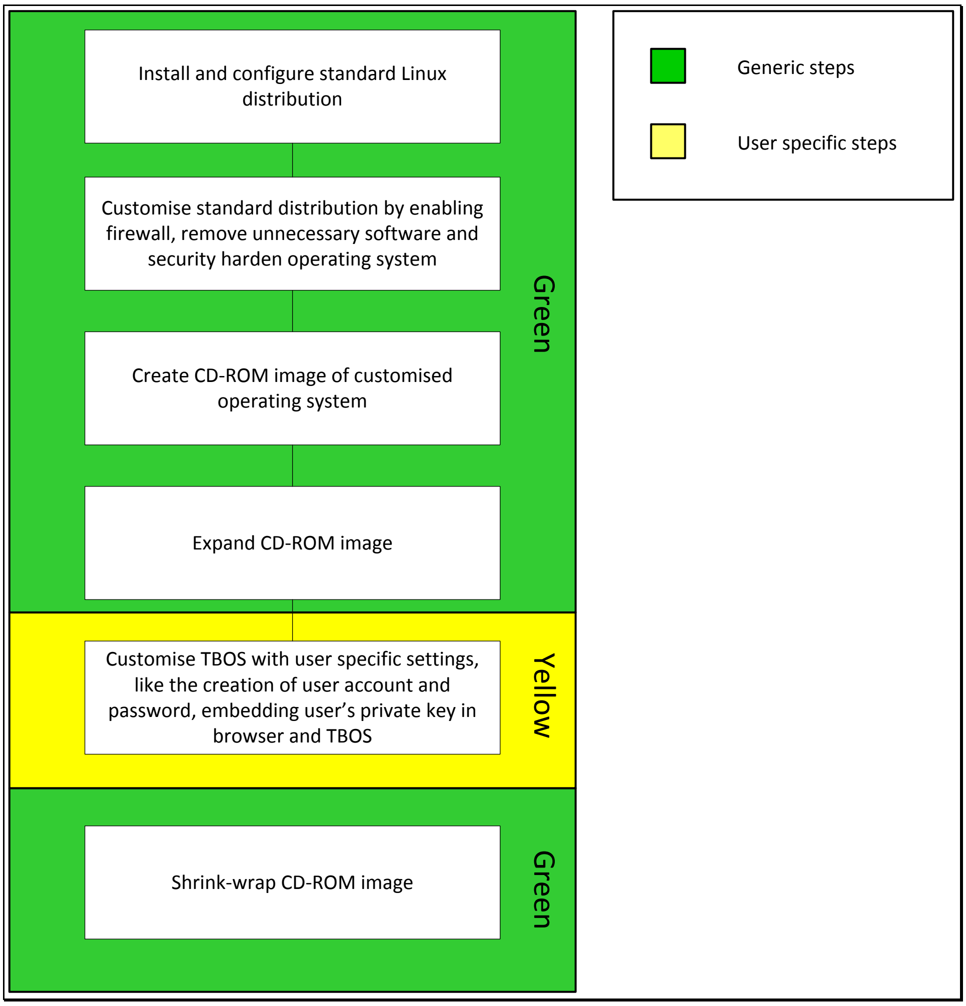

Install and configure standard Linux distribution
Customise standard distribution by enabling firewall, remove unnecessary software and security harden operating system
Create CD-ROM image of customised operating system
Expand CD-ROM image
Green
Customise TBOS with user specific settings, like the creation of user account and password, embedding user's private key in browser and TBOS
Yellow
Shrink-wrap CD-ROM image
Green
Generic steps
User specific steps

FIGURE 8:19 THE HIGH LEVEL STEPS REQUIRED TO CREATE THE TBOS

3.2.5 Less user friendly

The V-Bank system requires that the user has some knowledge in working with the virtual machine software. Users may feel negative towards the extra steps required to start the TBOS and the time it takes for the TBOS to start. These feelings can increase if the user is used to only starting his Internet browser and then directly going into the banking system.

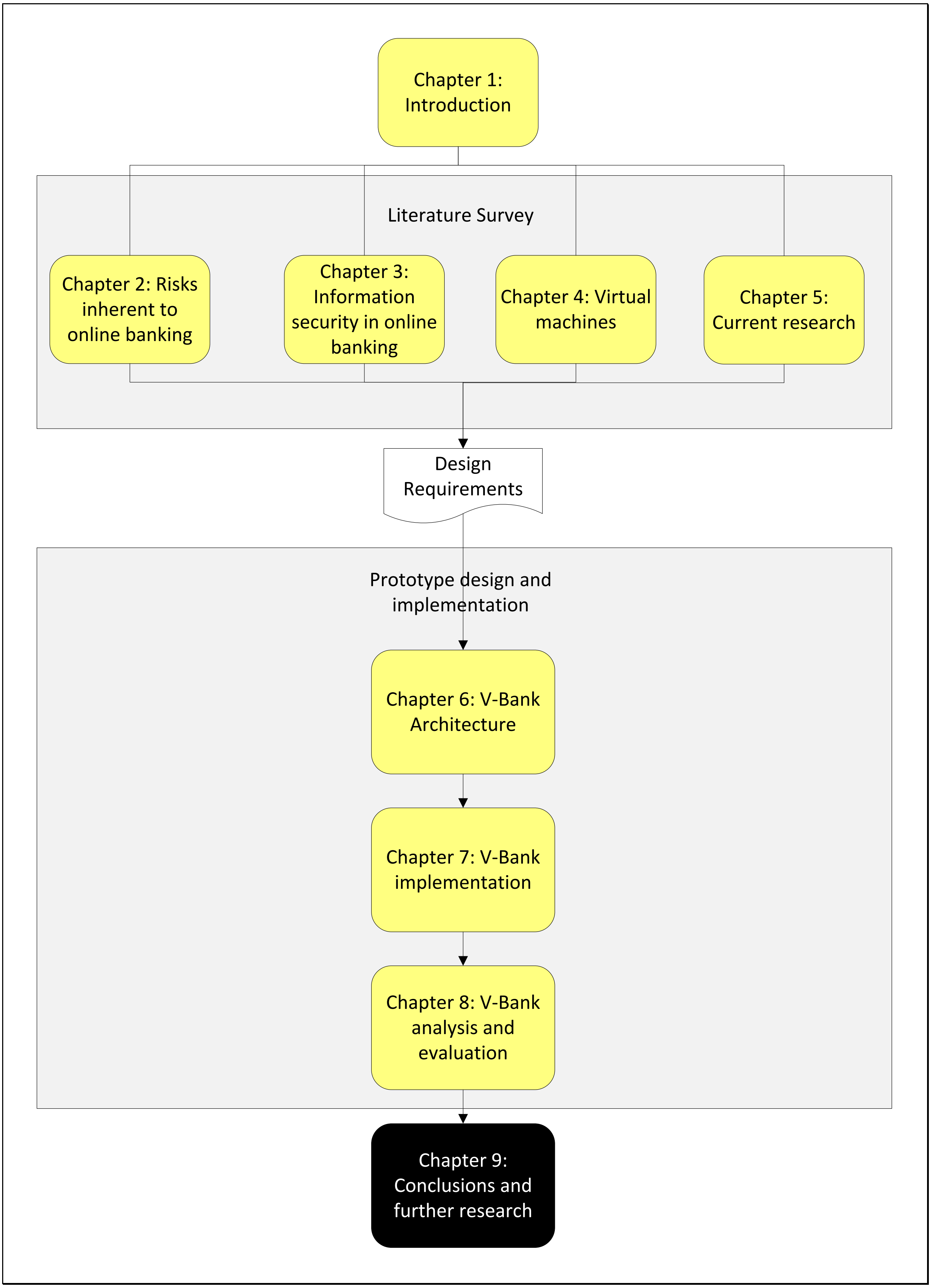
In a commercial environment where banks compete for customer numbers, this may also cause customers to rather decide to use banks with solutions that may be easier to use.

4 Conclusion

This chapter demonstrated how the V-Bank prototype works. The system was evaluated from the perspective of the bank, the banking user and from an attacker.

The V-Bank prototype can enhance the security of an online banking system. Users will be able to use the system with a high degree of assurance that their online banking sessions are protected from attackers.

A number of items make the V-Bank system difficult to implement in a commercial environment. If the system can be further developed to solve the weaknesses identified in the dissertation then the system has merit for commercial adoption.


Chapter 1: Introduction
Literature Survey
Chapter 2: Risks inherent to online banking
Chapter 3: Information security in online banking
Chapter 4: Virtual machines
Chapter 5: Current research
Design Requirements
Prototype design and implementation
Chapter 6: V-Bank Architecture
Chapter 7: V-Bank implementation
Chapter 8: V-Bank analysis and evaluation
Chapter 9: Conclusions and further research

Chapter 9: CONCLUSIONS AND FURTHER RESEARCH

1 Introduction

This study looked at the problems that users and banks face when conducting banking on the Internet. In this chapter we conclude the study that was defined in chapter 1.

This chapter reviews the problem statement and looks at the defined hypothesis and state whether the hypothesis was true. The chapter concludes with some areas that may require further research.

The next section reviews the problem statement and tests the hypothesis.

2 Problem statement and hypothesis

In Chapter 1:2 it was argued that the problem that this study attempts to resolve is that banks cannot control what users load on their computers, nor can they control how users' operating systems are configured. Many users are not security conscious and do not know what effect a specific piece of software has on the overall security of their computer. This all leads to an environment where the operating system environment of a normal desktop computer is in an unknown, potentially insecure state.

This study hypothesises that it is possible for banks to create a virtualized computer that can be configured to be in a known state. This known state can be evaluated and changed to conform to an accepted security configuration. This virtual computer, with its security configuration, can be locked so that it cannot change. The virtual computer can then be given to a banking user as an environment in which the user should do online banking, which should be more secure than their normal computer.

This study has demonstrated that the hypothesis is true. It is possible for a bank to create a virtualised computer that can be configured to be in a known state. This known state can be evaluated and changed according to an accepted security configuration. The virtual computer can be locked so that it cannot change. The virtual computer can be loaded on a various media that can be given to a user that gives them an environment in which the user can do online banking, which is more secure than their own computer.

3 Primary and Secondary Objectives

The primary objective of this study is:

Primary Objective:

Develop a proof of concept system using virtualisation to increase the security of online banking from an end-user's perspective

The first part of the objective is to develop a prototype that uses virtualisation. Chapters 7 and 8 showed how the prototype is designed and implemented.

The second part of the objective states that the prototype must increase the security of online banking. It was already argued in the problem statement that the typical computer of user can be an unsafe environment from which online banking is conducted. Chapter 8 shows the design requirements and that the V-Bank system fulfils the design requirements. This makes the V-Bank system a safer environment from which online banking can be conducted, thus fulfilling the primary objective.

The secondary objectives and the fulfilling of the objectives of the study are:

- Describe the security risks faced by online banking. Chapter 2 described the risks inherent to online banking. The risks were identified and from the risks an initial set of design requirements is formulated.
- Describe information security mechanisms used in online banking environments. Chapter 3 evaluates the security mechanisms used in online banking. The security mechanisms are evaluated using a case study and the design requirements for a safe online banking system are extended.
- Describe the security controls included in an online banking system. The existing security controls used in online banking are described in Chapter 3.
- Identify the security advantages provided by virtual machines. Chapter 4 evaluates virtual machines and the security advantages provide by virtual machines. Chapter 5 expanded on this by investigating existing research where virtual machines are used to enhance the web experience.
- Describe the architecture of an online banking system. The architecture of the V-Bank prototype is described in Chapter 6.
- Describe the isolation features that enhance the security of the online banking session. Chapter 4 described the isolation features of virtual machines. The isolation features are used as design requirements and forms part of the V-Bank prototype architecture and design described in chapters 6 and 7.

The dissertation fulfilled the primary objective and secondary objectives defined in Chapter 1.

The next section describes the areas that may require further study.

4 Areas for further study

This section covers some of the areas that may require further study. These areas were identified while doing research. Some areas also need to be expanded to show potential solutions.

The areas identified that may require further study are the following and discussed in the following sections:

- Interdependencies on the host system
- Software based private key
- Update unfriendly
- Static password

- Administrative overhead
- Less user friendly

4.1 Interdependencies of the host system

Because the virtual machine monitor is installed inside the user's operating system an attacker may be able to connect to the virtual machine through an unknown vulnerability in the host operating system or virtual machine monitor software. To minimise this from happening is it recommended that virtual machine monitor software not execute as a hosted virtual machine and that the whole virtual machine should be isolated from the host operating system running on the same hardware.

Figure 9:1 shows a potential solution where the virtual machine monitor software may have a secure mode. This secure mode allows the virtual machine to have direct access to the system's hyper visor. Further to this the secure mode of the virtual machine software may have the ability to freeze the host operating system. This stops any vulnerability that may exist in the host operating system to affect the guest operating system.

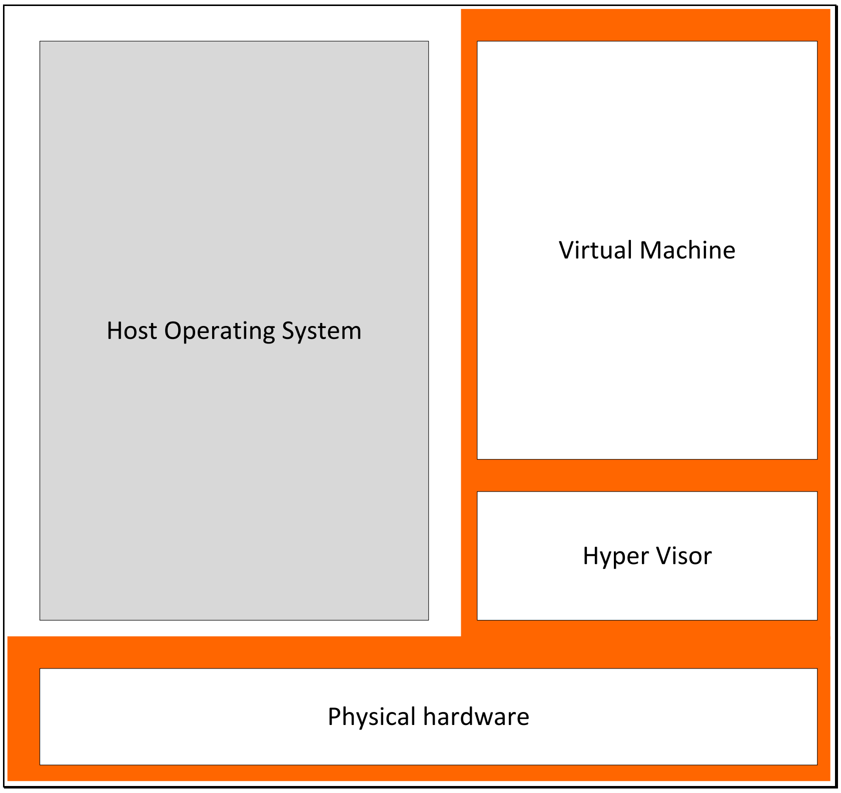


FIGURE 9:1 EXCLUSE HARDWARE USE BY A VIRTUAL MACHINE

4.2 Software based private key

The private key for the user in the V-Bank TBOS is stored within the Internet, as well as on a password protected file on the TBOS. An attacker that has physical access to the TBOS and has the password for the user may gain access to the user's private key and thus copying it, or stealing it.

A solution should be found where the private key is not part of the browser, nor is it stored on a file. The solution can even be a Federal Information Processing Standard (FIPS) compliant solution. These solutions already exist in the form of smartcards.

If the system can be implemented where the image of the TBOS is stored on a token that has a smartcard as part of the hardware, then the TBOS can read the user's private key from the smartcard. The security of the private key can then be further increased if the smartcard requires biometric unlocking.

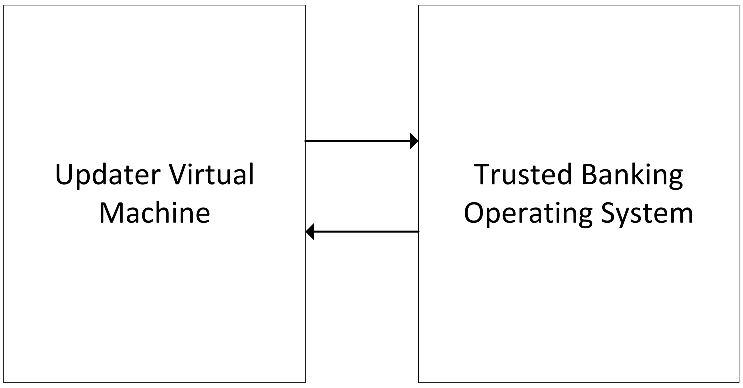
4.3 Update unfriendly

The persistent nature of the V-Bank TBOS makes it difficult for a bank to update deployed systems. The existing solution requires the bank to physically give the user a new image when the security configuration of the TBOS changes.

The TBOS should be redesigned so that when banks require a change in the security configuration of a TBOS that it can be easily deployed. The problem to try and overcome is that the image of the TBOS is persistent. This means that it is difficult for the image to update itself.

A potential solution for this requirement may be a second virtual machine whose role it is to update the TBOS. This updater virtual machine is in a locked state and can only be unlocked using the bank's private key.

Figure 9:2 shows how the process for updating the TBOS may work. It should be noted however that this solution may not be possible and requires study to determine if the process is secure and may not be exposed to other vulnerabilities.



1. User connects to the server-side banking system.
2. The server-side system determines through the configuration check that the TBOS is out of date.
3. A process executes on the TBOS that verifies the configuration of the Updater Virtual Machine (UVM) and updates it if necessary and unlocks the UVM.
4. The TBOS shuts down and starts the UVM
5. The UVM connects to the server-side system through secure channels and updates the TBOS
6. The UVM shuts down and starts the TBOS. Repeat from step 1

FIGURE 9:2 A MODEL FOR UPDATING THE TBOS

4.4 Static password

At the moment the password for the Linux user account cannot change because the image of the TBOS is persistent.

Users should have the option to change their password when required. A fully persistent system, cannot allow the user to change their password. There may be two solutions to this problem:

- The first solution is that the TBOS system has an area that is not persistent, that specifically hosts the user's password.
- The second solution may be that the TBOS does not use passwords at all, but protect the user's private key through biometric access control.

4.5 Administrative overhead

The V-Bank prototype requires significant administration to generate a TBOS for a client. This process should be automated and make it easy for a bank to generate the TBOS with the required hardware for a user.

The automation system mentioned here requires further study

4.6 Less user friendly

At the moment the user are required to first start the virtual machine monitor software. Once the virtual machine monitor software is running the user is required to start the TBOS. Only after the TBOS is running can the user log into the V-Bank system.

The amount of effort to start the environment may confuse users. A commercial solution requires that the user should go through very few steps to start the online banking session. The V-Bank system should be integrated into the virtual machine manager software so that a user does not even realise that the virtual machine manager software starts before the TBOS starts.

This may be possible by adapting open source virtual machine manager software, like Oracle VirtualBox to integrate seamlessly into the TBOS.

5 Conclusion

This chapter concludes the study into the design and implementation of a safe online banking solution that uses virtualisation. The chapter reviewed the problem that the solution addressed and determined whether the hypothesis that was stated in Chapter 1 was true.

The primary and secondary objectives were evaluated and found that the research objectives were met. The chapter also looked at areas for further study that resulted from the research.

The research described in this dissertation, showed that online banking faces many risks. Banks and consumers cannot always trust the state of their own computers, as it may be infected by malware.

The V-Bank prototype looks at the banking environment as a combination of both the client-side trusted banking operating system (TBOS) and the server-side back end. The V-Bank prototype does not try and solve the problem by just applying client-side controls, or just server-side controls. Instead, it approaches the online banking system as an enterprise system which incorporates the client, network and server.

Commercial banks can use the principles and architecture defined by the V-Bank prototype to increase the security of Internet banking. Further research and development can take the V-Bank system into a commercially viable solution that can be adopted by banks.

The V-Bank system has demonstrated that it is possible for banks to use virtualisation as an environment from where users can conduct online Banking.

Bibliography

ACM US Public Policy Council, 2012. *USACM - Privacy and Security.* [Online]
Available at: http://usacm.acm.org/images/documents/DNSDNSSEC-Senate.pdf
[Accessed 14 June 2012].

Agarwal, A. et al., 2012. *Reviewing the World of Virtualization.* Kota Kinabalu, IEEE Computer Society.

Akopyan, D. A. & Yelyakov, A. D., 2007. Cybercrimes in the Information Structure of Society: a Survey. *Scientific and Technical Information Processing ,* 36(6), pp. 338-350.

Akopyan, D. A. & Yelyakov, A. D., 2009. Cybercrimes in the Information Structure of Society: a Survey. *Scientific and Technical Information Processing,* 36(6), pp. 338-350.

AlZomai, M., AlFayyadh, B., Jøsang, A. & McCullagh, A., 2008. *An exprimental investigation of the usability of transaction authorization in online bank security systems.* Wollongong, Australian Computer Society, Inc., pp. 65--73.

an., 2013. *Main Page - KVM.* [Online]
Available at: http://www.linux-kvm.org/page/Main_Page
[Accessed 31 May 2013].

Anon., 2005. *IDesk.* [Online]
Available at: http://idesk.sourceforge.net/html/index.html
[Accessed 23 July 2012].

Anon., 2006. *Blackbox Wiki.* [Online]
Available at: http://blackboxwm.sourceforge.net/
[Accessed 23 July 2012].

Anon., 2013. *Bastille Linux.* [Online]
Available at: http://www.bastille-linux.org/
[Accessed 26 May 2013].

Beekmans, G., 2012. *Linux from scratch.* [Online]
Available at: http://www.linuxfromscratch.org/
[Accessed 19 July 2012].

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

Bhattacharjee, Y., 2011. Automated Theft Machines. *Time,* 177(2), pp. 53-54.

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

Business Software Alliance, 2011. *Definition of Keylogger - Business Software Alliance.* [Online]
Available at: http://www.bsacybersafety.com/threat/keylogger.cfm
[Accessed 13 June 2011].

Business Software Alliance, 2011. *Definition of Man-in-the-Middle Attack - Business Software Alliance.* [Online]
Available at: http://www.bsacybersafety.com/threat/man-in-the-middle.cfm
[Accessed 13 June 2011].

Carrow, E. L., 2007. *Puppetnets and botnets: information technology vulnerability exploits that threaten basic internet use.* s.l., Association for Computing Machinery.

Cheng, K., Gao, M. & Guo, R., 2010. *Analysis and Research on HTTPS Hijacking Attacks.* s.l., s.n., pp. 223-226.

Citrix Systems, Inc, 2012. *XenServer.* [Online]
Available at:
http://www.citrix.com/English/ps2/products/product.asp?contentID=683148&ntref=prod_cat
[Accessed 11 June 2012].

Claessens, J. et al., 2002. On the Security of Today's Online Electronic Banking Systems. *Computers and Security,* 21(3), pp. 253-265.

Clancy, C. T., Kiyavash, N. & Lin, D. J., 2003. *Secure Smartcard-Based Fingerprint Authentication.* Berkley, California, ACM, pp. 45-52.

Comella, R., Franham, G. & Jarocki, J., 2009. *Protecting Your Busienss from Online Banking Fraud.* [Online]
Available at: http://www.sans.edu/student-files/projects/200910_05.pdf
[Accessed 24 August 2011].

Cox, R., Hansen, J., Gribble, S. & Levy, H., 2006. *A safety-oriented platform for Web applications.* s.l., IEEE, pp. 349-364.

Creasy, R. J., 1981. The origin of the VM/370 time-sharing system. *IBM Journal of Research and Development,* 25(5), pp. 483-490.

Danchev, D., 2005. *The Complete Windows Trojans Paper.* [Online]
Available at:
http://www.windowsecurity.com/whitepapers/The_Complete_Windows_Trojans_Paper.html
[Accessed 15 March 2012].

Daw, D., 2012. The Growing Threat of ATM Skimmer Scams. *PC World*, 1 March, pp. 35-36.

Ellison, C. & Schneier, B., 2000. Ten Risks of PKI: What you're not being told about Public Key Infrastructure. *Computer Security Journal,* XVI(1), pp. 1-7.

England, P. & Manferdelli, J., 2006. Virtual machines for enterprise desktop security. *Inf. Secur. Tech. Rep.,* 11(January, 2006), pp. 193-202.

ESET, 2006. *Win32/Neshta.A.* [Online]
Available at: http://www.virusradar.com/en/Win32_Neshta.A/description
[Accessed 22 February 2013].

Federal Financial Institutions Examination Council, 2011. *Federal Financial Institutions Examination Council.* [Online]
Available at: http://www.ffiec.gov/pdf/Auth-ITS-Final%206-22-11%20(FFIEC%20Formated).pdf
[Accessed 1 January 2012].

Feily, M., Shahrestani, A. & Rmadass, S., 2009. *A Survey of Botnet and Botnet Detection.* Athens/Glyfada, Greece, IEEE Computer Society, pp. 268-273.

First National Bank, 2012. *FNB - Stay Secure.* [Online]
Available at: https://www.fnb.co.za/security-centre/stay-secure.html
[Accessed 17 December 2012].

Florêncio, D. & Herley, C., 2010. *Phishing and money mules.* Seatle, WA, s.n., pp. 1-5.

Florêncio, D. & Herley, C., 2010. *Phishing and money mules.* Seattle, IEEE, pp. 1-5.

Fraser, K., 2009. *Xen Hypervisor Project.* [Online]
Available at: http://xen.org/files/Marketing/HowDoesXenWork.pdf
[Accessed 4 May 2012].

Friedl, S. J., 2007. *SQL Injection Attacks by Example.* [Online]
Available at: http://www.unixwiz.net/techtips/sql-injection.html
[Accessed 27 August 2012].

Gibbs, M., 2011. Cracking MD5 ... with Google?!. *Network World*, 12 May, pp. 18-18.

Goldberg, I., Wagner, D. & Brewer, E., 1996. *A Secure Environment for Untrusted Helper Applications (Confining the Wily Hacker).* San Jose, s.n.

Goldstuck, A., 2004. *Online Banking in South Africa 2004 Report,* Johannesburg: World Wide Worx.

Goring, S. P., Rabaiotti, J. R. & Jones, A. J., 2007. Anti-keylogging measures for secure Internet login: An example of the law of unintended consequences. *Computers and Security,* Issue 21, pp. 421-426.

Grebennikov, N., 2011. *Keyloggers: Implementing keyloggers in Windows. Part Two.* [Online]
Available at:
http://www.securelist.com/en/analysis/204792178/Keyloggers_Implementing_keyloggers_in_Windows_Part_Two
[Accessed 20 February 2012].

Green, M., 2013. The Threat in the Cloud. *Security Privacy, IEEE,* 11(1), pp. 86-89.

Guan, B., Wu, Y. & Wang, Y., 2012. *A Novel Security Scheme for Online Banking Based on Virtual Machine.* s.l., IEEE Computer Society, pp. 12-17.

Gühring, P., 2006. *Welcome to CACert.org.* [Online]
Available at: http://www.cacert.at/svn/sourcerer/CAcert/SecureClient.pdf
[Accessed 14 06 2012].

Guimaraes, M., 2006. *New challenges in teaching datbase security.* Kennesaw, Georgia, ACM.

Halliday, S. G., 1997. *High Tech Aid.* [Online]
Available at: http://www.hightechaid.com/tech/card/intro_ms.htm
[Accessed 31 August 2012].

Hansman, S. & Hunt, R., 2005. A taxonomy of network and computer attacks. *Computers & Security,* Issue 24, pp. 31-43.

Horschman, E., 2008. *A Look at Some VMware Infrastructure Architectural Advantages.* [Online]
Available at: http://blogs.vmware.com/virtualreality/2008/06
[Accessed 28 August 2012].

Hussain, A., Heidemann, J. & Papadopoulos, C., 2003. *A framework for classifying denial of service attacks.* Karlsruhe, Germany, ACM, pp. 99-110.

Hutchinson, D. & Warren, M., 2003. Security for Internet banking: a framework. *Logistics Information Management,* 16(1), pp. 64-73.

Ianelli, N. & Hackworth, A., 2005. *Botnets as a vehicle for online crime.* s.l.:Carnegie Mellon University.

ISO, 1989. *ISO 7498-2:1989.* s.l.:ISO.

Johnson, C. & Kaufman, G. G., 2007. A bank by any other name .... *Economic Perspectives,* 3(4), pp. 37-49.

Jordaan, L. & von Solms, B., 2011. A Biometrics-Based Solution to Combat SIM Swap Fraud. *Open Research Problems in Network Security,* Volume 6555/2011, pp. 70-87.

Jovanovic, N., Kirda, E. & Kruegel, C., 2006. Preventing Cross Site Request Forgery Attacks. *Securecomm and Workshops,* Issue Aug. 28 2006-Sept. 1 2006, pp. 1-10.

Karim, Z., Rezaul, K. & Hossain, A., 2009. *Towards secure information systems in online banking.* London, IEEE, pp. 1-6.

Kolyshkin, K., 2012. *OpenVZ Wiki.* [Online]
Available at: http://wiki.openvz.org/Main_Page
[Accessed 4 May 2012].

Lindholm, T. & Yellin, F., 1999. *The JavaTM Virtual Machine Specification.* [Online]
Available at:
http://docs.oracle.com/javase/specs/jvms/se5.0/html/Introduction.doc.html#3057
[Accessed 28 March 2012].

Li, S. & Schmitz, R., 2009. *A Novel Anti-Phishing Framework Based on Honeypots.* Tacoma, WA, s.n., pp. 1-13.

Litchfield, D., Anley, C., Heasman , J. & Grindlay, B., 2005. *System level attacks.* 1 ed. s.l.:Wiley Publishing.

Macmillan Publishers Limited, 2009. *Online banking - defenition.* [Online]
Available at: http://www.macmillandictionary.com/dictionary/british/online-banking
[Accessed 12 December 2012].

Mahjoub, M., MDHAFFAR, A., BEN HALIMA, R. & JMAIEL, M., 2011. *A Comparative Study of the Current Cloud Computing Technologies and Offers.* Toulouse, IEEE , pp. 131-134.

Markovic, M., 2007. *Data Protection Techniques, Cryptographic Protocols and PKI Systems in Modern Computer Networks.* Maribor, Slovenia, IEEE Conference Publications.

Marty, M. R. & Hill, M. D., 2007. Virtual hierarchies to support server consolidation. *SIGARCH Comput. Archit. News,* 2(35), pp. 46-56.

McDowell, M., 2004. *US-CERT Tip ST04-014 - Avoiding Social Engineering and Phishing Attacks.* [Online]
Available at: http://www.us-cert.gov/cas/tips/ST04-014.html
[Accessed 14 June 2012].

Merriam Webster, 2012. *Merriam-Webster fraud.* [Online]
Available at: http://www.merriam-webster.com/dictionary/fraud
[Accessed 12 December 2012].

Microsoft Corporation, 2012. *Microsoft Application Virtualization (App-V).* [Online]
Available at: http://www.microsoft.com/en-us/windows/enterprise/products-and-technologies/virtualization/app-v.aspx
[Accessed 28 August 2012].

Microsoft Corporation, 2012. *Windows Virtual PC: Get Started.* [Online]
Available at: http://www.microsoft.com/windows/virtual-pc/get-started.aspx
[Accessed 18 June 2012].

Microsoft Corporation, 2013. *Windows 7.* [Online]
Available at: http://windows.microsoft.com/en-za/windows7/products/home
[Accessed 24 May 2013].

Microsoft, 2007. *Supported Guest Operating Systems in Virtual PC 2007: KB831461.* [Online]
Available at: http://support.microsoft.com/kb/831461
[Accessed 7 May 2012].

Microsoft, 2011. *Microsoft Security Intellgence Report,* s.l.: Microsoft.

Microsoft, 2011. *Windows Virtual PC: Requirements.* [Online]
Available at: http://www.microsoft.com/windows/virtual-pc/support/requirements.aspx
[Accessed 7 May 2012].

Microsoft, 2011. *Worm:Win32/Conficker.E.dll.* [Online]
Available at:

http://www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=Worm%3aWin32%2fConficker.E.dll
[Accessed 27 August 2012].

Microsoft, 2012. *Technical Documentation Download for System Center 2012 – Virtual Machine Manager.* [Online]
Available at: http://www.microsoft.com/en-us/download/details.aspx?id=6346
[Accessed 28 August 2012].

Microsoft, 2013. *Windows - Microsoft Windows Help.* [Online]
Available at: http://windows.microsoft.com/en-us/windows/windows-help?os=winxp#windows=windows-xp
[Accessed 22 February 2013].

Mills, E., 2010. *Zeus Trojan steals $1 million from U.K. bank accounts.* [Online]
Available at: http://news.cnet.com/8301-27080_3-20013246-245.html
[Accessed 12 November 2012].

Mirkovic, J. & Reiher, P., 2004. A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Comput. Commun. Rev.,* April, 34(2), pp. 39-53.

Nash, T., 2005. An Undirected Attack Against Critical Infrastructure. *Case Study Series: Vol 1.2*, September, p. 10.

Nedbank, 2012. *Nedbank - Security Tips.* [Online]
Available at:
https://netbank.nedsecure.co.za/Browser/Brands/Nedbank/Logon/html/SecurityTips.htm
[Accessed 17 December 2012].

Nilsson, M., Adams, A. & Herd, S., 2005. *Building security and trust in online banking.* Portland, OR, USA, ACM, pp. 1701-1704.

Olivier, M. S., 2009. *Information Technology Research.* Pretoria: van Schaik Publishers.

Opera Corporation, 2013. *Opera Browser.* [Online]
Available at: http://www.opera.com/
[Accessed 26 May 2013].

Oracle Corporation, 2011. *Oracle VM VirtualBox.* [Online]
Available at: https://www.virtualbox.org/UserManual.pdf
[Accessed 28 March 2012].

Oracle Corporation, 2012. *Downloads - Oracle VM VirtualBox.* [Online]
Available at: http://download.virtualbox.org/virtualbox/4.1.14/UserManual.pdf
[Accessed 7 May 2012].

Oracle Corporation, 2012. *Oracle VM Templates.* [Online]
Available at: http://www.oracle.com/technetwork/server-storage/vm/overview/templates-101937.html
[Accessed 28 August 2012].

Oracle, n.d.. *Java+ You.* [Online]
Available at: http://java.com/en/
[Accessed 23 July 2012].

Overseas Security Advisory Council, 2012. *South Africa 2012 OSAC Crime and Safety Report.* [Online]
Available at: https://www.osac.gov/pages/ContentReportDetails.aspx?cid=12014
[Accessed 27 August 2012].

Parallels IP Holdings GmbH, 2013. *OpenVZ Linux Containers.* [Online]
Available at: http://openvz.org/Main_Page
[Accessed 31 May 2013].

PC Magazine, 2012. *LiveCD Definition from PC Magazine Encyclopedia.* [Online]
Available at:
http://www.pcmag.com/encyclopedia_term/0,1237,t=LiveCD&i=58080,00.asp
[Accessed 11 June 2012].

Perez, R., van Doorn, L. & Sailer, R., 2008. Virtualization and Hardware-Based Security. *Security Privacy, IEEE,* 6(5), pp. 24-31.

Pieters, W., van Cleef, A. & Wieringa, R., 2009. *Security Implications of Virtualization: A Literature Study.* Loa Alamitos, CA, USA, IEEE Computer Society, pp. 353-358.

Piromsopa, K. & Enbody, R. J., 2006. *Buffer-Overflow Protection: The Theory.* s.l., s.n., pp. 454-458.

Rabkin, A., 2008. *Personal knowledge questions for fallback authentication: security questions in the era of Facebook.* Pittsburgh, ACM, pp. 13--23.

Riley, S., 2006. *Mandatory integrity control in Windows Vista.* [Online]
Available at: http://blogs.technet.com/b/steriley/archive/2006/07/21/442870.aspx
[Accessed 3 January 2013].

Rosenblum, M., 2004. The Reincarnation of Virtual Machines. *Queue,* 2(5), pp. 34-40.

Rosenblum, M. & Garfinkel, T., 2005. Virtual Machine Monitors: Current Technology And Futur Trends. *Computer,* 38(5), pp. 39-47.

Rowan, T., 2008. Virtual Desktop: The secure virtual computer on your keychain. *Network Security,* Volume 2008, pp. 11-14.

Rowan, T., July, 2008. Virtual Desktop: The secure virtual computer on your keychain. *Network Security,* 2008(7), pp. 11-14.

Russinovich, M. E. & Solomon, D., 2005. *Microsoft Windows Internals Fourth Edition.* 4th ed. Redmond: Microsoft Press.

Sagiroglu, S. & Canbek, G., 2009. Keylogger. *IEEE Technology and Society,* 28(3), pp. 10-17.

Samarati, P. & De Capitani di Vimercati, S., 2001. Access Control: Policies, Models, and Mechanisms. In: R. Focardi & R. Gorrieri, eds. *Foundations of Security Analysis and Design.* s.l.:Springer-Verlag.

Scarfone, K., Souppaya, M. & Hoffman, P., 2011. *Guide to security for full virtualization technologies,* Gaithersburg, MD: National Institute of Standards and Technology.

Shamir, A. & van Someren, N., 1999. Playing "Hide and Seek" with Stored Keys. *Financial Cryptography,* Volume 1648/1999, pp. 118-124.

Sherstobitoff, R., 2011. The new frontier for Zeus & SpyEye. *Information Systems Security Association Journal,* Volume September 2011, pp. 12-18.

Shirey, R., 2000. *Internet Security Glossary.* [Online]
Available at: http://www.ietf.org/rfc/rfc2828.txt
[Accessed 15 March 2012].

Shuler, R., 1998. *How does the Internet work?.* [Online]
Available at: http://www.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm
[Accessed 13 December 2012].

Smith, J. E. & Nair, R., 2005. The architecture of virtual machines. *Computer,* 38(5), pp. 32-38.

Standard Bank, 2005. *Spoof website.* [Online]
Available at:
https://www.standardbank.co.za/secure/securitycentre/spoof_overview.html
[Accessed 27 August 2012].

Standard Bank, 2012. *Internet Banking - General Information - Security.* [Online]
Available at:
http://www.standardbank.co.za/portal/site/standardbank/menuitem.de435aa54d374eb6fcb695665c9006a0/?vgnextoid=a6e1f8bc8f35b210VgnVCM100000c509600aRCRD
[Accessed 17 December 2012].

Symantec Corporation, 2012. *Malware - Malicious Virus Code Detection.* [Online]
Available at: http://us.norton.com/security_response/malware.jsp
[Accessed 14 June 2012].

Symantec Corporation, 2013. *W32.SQLExp.Worm Technial Details.* [Online]
Available at: http://www.symantec.com/security_response/writeup.jsp?docid=2003-012502-3306-99&tabid=2
[Accessed 26 February 2013].

Symantec, 2003. *Analysis-SQLExp.* [Online]
Available at: http://securityresponse.symantec.com/avcenter/Analysis-SQLExp.pdf
[Accessed 26 February 2013].

Symantec, 2009. *What is the difference between virusses, worms and Trojans?.* [Online]
Available at:
http://www.symantec.com/business/support/index?page=content&id=TECH98539
[Accessed 27 August 2012].

Symantec, 2010. *Trojan.Zbot.* [Online]
Available at: http://www.symantec.com/security_response/writeup.jsp?docid=2010-

011016-3514-99
[Accessed 22 February 2013].

Symantic Enterprise Security, 2011. *Symantec Internet Security Threat Report.* [Online]
Available at: http://go.symantec.com/istr
[Accessed 20 February 2012].

TechTarget, 2012. *Defnition: Attack vector.* [Online]
Available at: http://searchsecurity.techtarget.com/definition/attack-vector
[Accessed 14 June 2012].

The Gnome Project, 2005. *GDM - The GNOME Display Manager.* [Online]
Available at: http://projects.gnome.org/gdm/
[Accessed 23 July 2012].

VMWare Inc., 2005. *VMware vCloud Director 5.1 Documentation CenterCommunities Support Blogs Downloads.* [Online]
Available at: http://pubs.vmware.com/vcd-51/index.jsp?topic=%2Fcom.vmware.glossary.doc%2FGUID-045A0511-5BE7-41A4-9350-A9BED8C166B3.html
[Accessed 3 March 2013].

VMWare Inc., 2012. *ACE 2.5 Product FAQs.* [Online]
Available at: http://www.vmware.com/de/products/desktop_virtualization/ace/faqs
[Accessed 3 March 2013].

VMWare Inc, 2012. *VMware | Solution Exchange.* [Online]
Available at: https://solutionexchange.vmware.com/store
[Accessed 18 June 2012].

VMware, 2012. *VMware Player 4.0 EULA.* [Online]
Available at: http://www.vmware.com/download/eula/player40.html
[Accessed 7 May 2012].

Vogels, W., 2008. Beyond Server Consolidation. *Queue,* 6(January/February), pp. 20-26.

von Solms, S. H. & Eloff, J. H., 2004. *Information Security.* Johannesburg: N/A.

Wang, J., Huang, Y. & Ghosh, A., 2010. *SafeFox: A Safe Lightweight Virtual Browsing Environment.* s.l., s.n.

Wang, Z. & Jiang, X., 2010. *HyperSafe: A Lightweight Approach to Provide Lifetime Hypervisor Control-Flow Integrity.* s.l., IEEE, pp. 380-395.

Wert, R., 2012. *Jalopnik.* [Online]
Available at: http://jalopnik.com/5903732/breaking-anonymous-takes-down-formula-1-website-with-ddos-attack
[Accessed 28 August 2012].

Williams, M., 2012. *PC World Business Center.* [Online]
Available at:

http://www.pcworld.com/businesscenter/article/260940/atandt_hit_by_ddos_attack_suffers_dns_outage.html
[Accessed 28 August 2012].

Wu, L. & Yu, Z., 2011. *Automatic Detection Model of Malware Signature for Anti-virus Cloud Computing.* Sanya, Hainan Island, China, IEEE Computer Society.

Xen Project, 2013. *Xen Hypervisor.* [Online]
Available at: http://www.xen.org/products/xenhyp.html
[Accessed 31 May 2013].

Zoller, T., 2011. *TLS/SSL Hardening and compatibility report.* [Online]
[Accessed 5 May 2012].

Appendix A: Configuration settings for the V-Bank system

This appendix lists some of the commands and steps that were used when creating the V-Bank TBOS configuration.

1 High level steps to generate a TBOS client

The high level steps required to create a TBOS client are as follow:

1.) Install an Ubuntu desktop client as a virtual machine
2.) Configure, install and remove software required for the TBOS
3.) Use xxx to create a backup *.iso file from the existing installation
4.) Use the steps in 12 to extract the iso and modify some of the settings in the ISO and repackage it.

The rest of the appendix explains how to create the user accounts, install the JAVA runtime, configure the Opera browser, IPTables and package the TBOS client as in ISO. It also shows some of the settings required to modify IIS 7.5 to accept TLS 1.2 connections.

1 Create limited User Account

From the original Ubuntu window manager the following steps create an account with limited rights in the operating system.

System > Administration > Users and Groups

From the User settings window select, **Add User**

From the New User Account-window fill in the Basic Settings for your user

Name: bankinguser

Advanced Settings:

User Privileges: Ensure nothing is selected.

This will give the user only user permissions on the system without any special privileges.

2 Install Oracle Java Run-time

Login as an account with root permissions and ensure that you have downloaded the Oracle Java JRE for Linux. Go to the directory where the java runtime has been downloaded.

Copy and extract the Java source files

```
sudo -s cp –r jre-6u33-linux-i586.bin /opt/java
cd /opt/java/
sudo -s chmod a+x jre-6u33-linux-i586.bin
sudo ./jre-6u33-linux-i586.bin
sudo gedit /etc/profile
```

Add the following lines to the profile file

```
JAVA_HOME /opt/java/jre1.6.0_33
PATH=$PATH:$HOME/bin:$JAVA_HOME/bin
JRE_HOME= /opt/java/jre1.6.0_33
PATH=$PATH:$HOME/bin:$JRE_HOME/bin
export JAVA_HOME
export JRE_HOME
export PATH
```

Save the profile file.

```
sudo     update-alternatives     --install     "/usr/bin/java"     "java"
"/opt/java/jre1.6.0_33/bin/java" 1
sudo     update-alternatives     --install     "/usr/bin/javaws"     "javaws"
"/opt/java/jre1.6.0_33/bin/javaws" 1
sudo update-alternatives --set java /opt/java/jre1.6.0_33/bin/java
sudo update-alternatives --set javaws /opt/java/jre1.6.0_33/bin/javaws
```

Update the environment

```
. /etc/profile
```

Update the Opera browser Java plugin.

```
cd /usr/lib/opera/plugins
sudo ln –s /opt/java/jre1.6.0_33/lib/i386/libnpjp2.so
```

3 Install and configure Opera browser

3.1 Creat Opera URL Approve List

```
   cat > ~/urlfilter.ini << "EOF"

   [prefs]

   prioritize excludelist=0



   [include]

   file://localhost/usr/share/bankinghome/default.htm

   file://localhost/usr/share/bankinghome/*

   https://v-bank/banking/*

   *.class



   [exclude]

   *
```

```
http://help.opera.com/img/*=UUID:08FD5E80E9E511E084B8F1090A5DB689

EOF


sudo mv ~/urlfilter.ini /usr/share/bankinghome

sudo chgrp root /usr/share/bankinghome/urlfilter.ini

sudo chown root /usr/share/bankinghome/urlfilter.ini
```

3.2 Install Opera Web Browser – Required for TLS 1.2

Open existing Web Browser and go to: http://www.opera.com/browser/download/

Save file.

Double click the *.deb file.

Click on the Install button.

Copy Required Certificates

Copy the certificates folder to home directory

```
sudo mv ~/certificates /home/bankinguser

sudo chgrp bankinguser /home/bankinguser/certificates -R

sudo chown bankinguser /home/bankinguser/certificates -R
```

3.3 Configure Opera settings

As the banking user:

Open Opera browser

In the address bar, type opera:config

Bittorent client. (Remove the check box next to Enable)

Network \ URL Filter file, Select

Select Opera menu, Preferences, Advanced, Security.

Click the Security Protocols button, unselect SSL 3 and TLS 1. Select only TLS 1.2

Click on Manage Certificate button.

On the Authorities tab, click Import

Import the Naledi root certificate, as well as the user certificate.

4 Install Blackbox Window Manager

```
sudo apt-get install blackbox blackbox-themes

cat > .blackboxrc << "EOF"
session.styleFile: /usr/share/blackbox/styles/Gray
session.menuFile: /home/bankinguser/.blackbox/menu
session.screen0.workspaces: 1
session.screen0.workspaceNames: My Banking Screen
session.fullMaximization: True
EOF
sudo mv .blackboxrc /home/bankinguser/.blackboxrc


sudo mkdir /home/bankinguser/.blackbox
sudo chown bankinguser /home/bankinguser/.blackbox
sudo chown bankinguser /home/bankinguser/.blackboxrc
sudo chgrp bankinguser /home/bankinguser/.blackbox
sudo chgrp bankinguser /home/bankinguser/.blackboxrc
```

4.1 Create Blackbox Menu

```
cat > menu << "EOF"
[begin] (Blackbox)
[exec] (Opera) { opera -kioskmode -kioskbuttons -nocontextmenu -nodownload
-nokeys -nomail -nomaillinks -nomenu -nosave -resetonexit }
[end]
EOF
sudo mv menu /home/bankinguser/.blackbox/menu

sudo chgrp bankinguser /home/bankinguser/.blackbox/menu
sudo chown bankinguser /home/bankinguser/.blackbox/menu


{opera -nocontextmenu -nodownload -nokeys -nomail -nomaillinks -nomenu -
nosave -resetonexit }
```

5 Install iDesk

```
sudo apt-get install idesk

cat > .ideskrc << "EOF"
table Config
Background.Color: #C2CCFF
end
table Actions
Execute[0]: left singleClk
end
EOF
sudo mv .ideskrc /home/bankinguser/.ideskrc

sudo mkdir /home/bankinguser/.idesktop
sudo cp /usr/share/pixmaps/firefox.png
/home/bankinguser/.idesktop/touchicon.png

cat > touchicon.lnk << "EOF"
```

```
table Icon
Caption: Opera
ToolTip.Caption: Double click me to launch
Command: opera -kioskmode -nocontextmenu -nodownload -nokeys -nomail -
nomaillinks -nomenu -nosave -resetonexit
Icon: /home/bankinguser/.idesktop/touchicon.png
Width: 100
Height: 100
X: 100
Y: 100
end
EOF
sudo mv touchicon.lnk /home/bankinguser/.idesktop/touchicon.lnk

sudo chgrp bankinguser /home/bankinguser/.ideskrc
sudo chgrp bankinguser /home/bankinguser/.idesktop
sudo chown bankinguser /home/bankinguser/.ideskrc
sudo chown bankinguser /home/bankinguser/.idesktop
sudo chown bankinguser /home/bankinguser/.idesktop/touchicon.lnk
sudo chgrp bankinguser /home/bankinguser/.idesktop/touchicon.lnk
```

5.1 Create Startup script

```
cat > .bbstartup.sh << "EOF"
#!/bin/sh
idesk &
exec blackbox
EOF
sudo mv .bbstartup.sh /home/bankinguser/.bbstartup.sh


sudo chgrp bankinguser /home/bankinguser/.bbstartup.sh
sudo chown bankinguser /home/bankinguser/.bbstartup.sh
sudo chmod +x /home/bankinguser/.bbstartup.sh

sudo cp /usr/share/xsessions/blackbox.desktop
/usr/share/xsessions/blackbox.desktop_backup

cat > blackbox.desktop << "EOF"
[Desktop Entry]
Encoding=UTF-8
Name=BlackBox
Comment=Highly configurable and low resource X11 Window manager
Exec=/home/bankinguser/.bbstartup.sh
Terminal=False
TryExec=/home/bankinguser/.bbstartup.sh
Type=Application
EOF
sudo mv blackbox.desktop /usr/share/xsessions/blackbox.desktop
```

Change user profile desktop

Open the Login Window Preferences by selecting:

System > Administration > Login Window

Under the General tab select 'Blackbox' for default session.

```
sudo /usr/lib/lightdm/lightdm-set-defaults -s blackbox
```

6 Use GDM Login Manager

```
sudo dpkg-reconfigure gdm
```

7 Remove sessions apart from blackbox

Rename all *.desktop file under /usr/share/xsession to *.desktop_backup

8 Remove a list of user accounts from login screen

To "hide" the first configuration user account, do the following:

```
gdm gconftool-2 --type bool --set /apps/gdm/simple-greeter/disable_user_list 'true'
```

9 Give banking user permissions to shutdown computer

```
sudo
The program sudo allows normal users to execute certain root-only commands.
Which users are authorized to run which commands is specified in the
/etc/sudoers file.
This should only be edited with the command visudo.

For example, suppose I wanted to add a group of users who are allowed to
shut down the machine.
So I first want to add a group called "shutdown" (run these commands while
root)

groupadd shutdown

Then I need to edit the /etc/group file to add users to the "shutdown"
group.
I just tack the usernames at the end of the shutdown line, separated by
commas, e.g.

shutdown:x:407:user1,user2,...

Whatever users I put there will be able to shut down the computer (so
choose wisely).
Now I need to configure sudo to allow members of the "shutdown" group to
actually
invoke the assorted shutdown commands provided in linux. Run visudo and add
the following lines

%shutdown ALL=(root) NOPASSWD: /sbin/reboot
%shutdown ALL=(root) NOPASSWD: /sbin/halt
%shutdown ALL=(root) NOPASSWD: /sbin/shutdown

This allows the "shutdown" group to run /sbin/reboot, /sbin/halt, and
/sbin/shutdown
AS IF THEY WERE ROOT
```

```
Add the following to the blackbox/menu
sudo /sbin/shutdown -h now
```

10 Create a banking application web page.

Create a default.htm file, with hyperlinks to the various banks to where the user needs to connect to.

```
Sudo mkdir /usr/share/bankinghome
Sudo cp *.htm /usr/share/bankinghome
```

Change the Opera home page to file:///usr/share/bankinghome/default.htm

11 Configure iptables in Ubuntu

The IP address 192.168.209.1 in the rule set below is the IP address of the V-Bank server. Change this IP address to the actual IP address of the server

```
sudo cat > ~/iptables.rules
# Generated by iptables-save v1.4.10 on Tue Jun 28 21:46:16 2011
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -d 255.255.255.255/32 -i eth0 -j DROP
-A INPUT -d 192.168.255.255/32 -i eth0 -j DROP
-A INPUT -d 192.168.1.255/32 -i eth0 -j DROP
-A INPUT -d 10.0.0.0/8 -j DROP
-A INPUT -d 169.254.0.0/16 -j DROP
-A INPUT -m recent --rcheck --seconds 60 --name DEFAULT --rsource -m limit
--limit 10/sec -j LOG --log-prefix "BG "
-A INPUT -m recent --update --seconds 60 --name DEFAULT --rsource -j DROP
-A INPUT -s 192.168.32.128/32 -i eth0 -m recent --set --name DEFAULT --
rsource -j DROP
-A INPUT -i eth0 -p tcp -m multiport --dports 53,113,135,137,139,445 -j
DROP
-A INPUT -i eth0 -p udp -m multiport --dports 53,113,135,137,139,445 -j
DROP
-A INPUT -i eth0 -p udp -m udp --dport 1026 -j DROP
-A INPUT -i eth0 -p tcp -m multiport --dports 1433,4899 -j DROP
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -m limit --limit 10/sec -j ACCEPT
-A INPUT -p icmp -j DROP
-A INPUT -p udp -m udp --sport 53 -m state --state ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -j LOG --log-prefix "IN "
-A INPUT -j REJECT --reject-with icmp-port-unreachable
-A FORWARD -j LOG --log-prefix "FW "
-A FORWARD -j REJECT --reject-with icmp-port-unreachable
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -d 192.168.209.1/32 -p tcp -m state --state
NEW,RELATED,ESTABLISHED -m tcp --sport 1024:65535 --dport 80 -j ACCEPT
-A OUTPUT -d 192.168.209.1/32 -o eth0 -p tcp -m state --state
NEW,RELATED,ESTABLISHED -m tcp --sport 1024:65535 --dport 443 -j ACCEPT
-A OUTPUT -j LOG --log-prefix "OU "
-A OUTPUT -j REJECT --reject-with icmp-port-unreachable
```

```
COMMIT
# Completed on Tue Jun 28 21:46:16 2011
EOF
```

Create a restore file as well.

```
sudo cat > ~/iptables.restore
# These lines are here in case rules are already in place and the script is
ever rerun on the fly.
# We want to remove all rules and pre-exisiting user defined chains and
zero the counters
# before we implement new rules.
iptables -F
iptables -X
iptables -Z

# Set up a default DROP policy for the built-in chains.
# If we modify and re-run the script mid-session then (because we have a
default DROP
# policy), what happens is that there is a small time period when packets
are denied until
# the new rules are back in place. There is no period, however small, when
packets we
# don't want are allowed.
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
EOF

chmod +x ~/iptables.restore
```

Apply the iptables rules by doing the following:

```
sudo iptables-restore < ~/iptables.rules
```

Reset the iptables rules to defaults by doing the following:

```
sudo ./iptables.restore
```

12 Create the TBOS Ubuntu client

12.1 Obtain the base system

Download an official Desktop CD from http://releases.ubuntu.com/

Note: the example shown here uses the ubuntu-9.04-desktop-i386.iso Desktop CD

12.2 Move or copy it into an empty directory

```
   mkdir ~/livecdtmp

   mv image.iso ~/livecdtmp

   cd ~/livecdtmp
```

12.3 Extract the CD .iso contents

12.3.1 Mount the Desktop .iso

```
mkdir mnt

sudo mount -o loop image.iso mnt
```

12.3.2 Extract .iso contents into dir 'extract-cd'

```
mkdir extract-cd

rsync --exclude=/casper/filesystem.squashfs -a mnt/ extract-cd
```

12.4 Extract the Desktop system

12.4.1 Extract the SquashFS filesystem

```
sudo unsquashfs mnt/casper/filesystem.squashfs

sudo mv squashfs-root edit
```

12.4.2 Prepare and chroot

If you need the network connection within chroot

```
sudo cp /etc/resolv.conf edit/etc/
```

Depending on your configuration, you may also need to copy the hosts file

```
sudo cp /etc/hosts edit/etc/
```

```
sudo mount --bind /dev/ edit/dev

sudo chroot edit

mount -t proc none /proc

mount -t sysfs none /sys

mount -t devpts none /dev/pts
```

(these mount important directories of your host system - if you later decide to delete the edit/ directory, then make sure to unmount before doing so, otherwise your host system will become unusable at least temporarily until reboot)

To avoid locale issues and in order to import GPG keys

```
export HOME=/root

export LC_ALL=C
```

12.5 Customizations

12.5.1 Apt-get

```
sudo apt-get install sun-java6-jre sun-java6-plugin sun-java6-fonts
```

12.5.2 Prerequisites

In 9.10, before installing or upgrading packages you need to run

```
dbus-uuidgen > /var/lib/dbus/machine-id
```

and

```
dpkg-divert --local --rename --add /sbin/initctl

ln -s /bin/true /sbin/initctl
```

12.5.3 Install Trusted Root Certificates

Get the libnss3-tools

```
sudo apt-get install libnss3-tools

wget -O cacert-root.crt "http://www.cacert.org/certs/root.crt"

wget -O cacert-class3.crt "http://www.cacert.org/certs/class3.crt"

certutil  -d  .mozilla/firefox/zpigy3el.default/  -A  -t  "TC,C,C"  -n
"CAcert.org Class 3" -i cacert-class3.crt

certutil  -d  .mozilla/firefox/zpigy3el.default/  -A  -t  "TC,C,C"  -n
"CAcert.org" -i cacert-root.crt
```

You may also need to import the user's certificate that contains the private key into the firefox database.

Copy the "Private_Key_Cert.pfx" to the banking user's home directory. Note: This certificate would have to have been generated by a certificate authority like CACert.

Ubuntu openssl doesn't seem to like the "_" character in the filename. So for rename the filename using the following command"

```
Mv Private_Key_Cert.pfx cert.pfx
```

Now you can use the pk12util to import the pfx file

```
pk12util -i cert.pfx -d .mozilla/firefox/zpigy3el.default/ -n "Banking
User"
```

12.6 Advanced Customizations

12.6.1 Live CD Kernel

If you want to customize further the boot process, you can change the livecd kernel, by copying the vmlinuz and initrd you want in place of the ones you find in extract-cd/casper.

i.e.

```
sudo cp edit/boot/vmlinuz-2.6.15-26-k7 extract-cd/casper/vmlinuz

sudo cp edit/boot/initrd.img-2.6.15-26-k7 extract-cd/casper/initrd.gz
```

Note that the initial ramdisk filename for newer releases (since 9.10) is casper/initrd.lz (not .gz).

12.6.2 Removing the (Casper) Autologin

The autologin feature of the Jaunty/9.04 live CD is a bit of an on-the-fly boot-hack. After extracting the initrd.gz, you need to edit the casper-bottom/25configure_init script and then recreate the initrd.gz file, replacing the original in extract-cd/casper. The process to do so goes like this:

```
cd extract-cd/casper

mkdir tempdir

cd tempdir

gunzip -dc ../initrd.gz | cpio -imvd --no-absolute-filenames

cp scripts/casper-bottom/25configure_init scripts/casper-bottom/25configure_init.orig

vi scripts/casper-bottom/25configure_init
```

Now look for line 25 which has the conditional statement to test $USERNAME.

Line 25 performs a conditional evaluation and if it evaluates to true, it will execute the code within the if block. The if block contains code to modify files used in the boot process to create the live cd autologin.

To disable the autologin feature, Remove $USERNAME, but just leave the quotes. The -n modifier tests the $USERNAME string to see if it's length is non-zero. By removing the variable, and leaving two double quotes, this statement evaluates to false because the two double quotes effectively make a zero-byte string. Be sure to leave no whitespace between the quotes because whitespace will make the evaluation true and execution wil fall into the if block.

```
21:log_begin_msg "$DESCRIPTION"

22:

23:# Arrange for shells on virtual consoles, rather than login prompts

24:
```

```
25:if [ -n "$USERNAME" ]; then
```

After making the change, line 25 will look like this:

```
25:if [ -n "" ]; then
```

Save the file and quit the editor.

There are also some other settings which may be changed at this stage. Some of the startup scripts set the hostname. For the v-Bank system the hostname was changed to "TBOS".

Another setting that was changed in here was the script that generates the hosts file, which will normally be stored under /etc/hosts. The hosts file has been edited to include the hostname for the v-bank server-side system, with the IP address of the server.

Some of the scripts in here also needs to be updated so that the v-Bank TBOS does not ask for the "installation media" to be removed from the CD-ROM.

12.6.3 Boot init

You have to edit the files in edit/usr/share/initramfs-tools/scripts/casper-bottom/* For example you can change the hostname or the livecd user.

You can use this method to edit the default hosts file that recides under /etc/hosts after the system has started.

i.e.

```
sudo nano edit/usr/share/initramfs-tools/scripts/casper
```

and edit the username or hostname

```
sudo nano edit/usr/share/initramfs-tools/scripts/casper-bottom/10adduser
```

to edit even the livecd user's password.

If you're customizing 10.04, you need to edit variables in /etc/casper.conf for the user and host names instead of modifying the scripts

P.S. in order to obtain an encrypted password, you have to use the mkpasswd program that's shipped with whois package!

Then, from extract-cd/casper/tempdir run the following command to re-create the initrd.gz file. There are other methods for re-creating the initrd.gz file on this page which may work also:

```
cp ../initrd.gz ../initrd.gz.orig

find . | cpio -o -H newc | gzip -9 > ../initrd.gz
```

This will create a new initrd.gz file with no auto login. You can then continue to remaster the CD as described on this page. Be sure to create a user and password to login with before you remaster the cd. If you do not, you will not be able to login after booting!

12.7 Cleanup

Be sure to remove any temporary files which are no longer needed, as space on a CD is limited. A classic example is downloaded package files, which can be cleaned out using:

```
aptitude clean
```

Or delete temporary files

```
rm -rf /tmp/* ~/.bash_history
```

Or delete hosts file

```
rm /etc/hosts
```

Or nameserver settings

```
rm /etc/resolv.conf
```

If you installed software, be sure to run

```
rm /var/lib/dbus/machine-id
```

and

```
rm /sbin/initctl

dpkg-divert --rename --remove /sbin/initctl
```

from within the chroot environment.

now umount (unmount) special filesystems and exit chroot

```
umount /proc || umount -lf /proc

umount /sys

umount /dev/pts

exit

sudo umount edit/dev
```

Note: if "umount /proc" command fails, "umount -lf /proc" will be used to retry automatically.

12.8 Producing the CD image

The following steps show how to recreate the CD image from the temporary extracted CD files.

12.8.1 Regenerate manifest

```
sudo chmod +w extract-cd/casper/filesystem.manifest

sudo chroot edit dpkg-query -W --showformat='${Package} ${Version}\n' > extract-cd/casper/filesystem.manifest

sudo cp extract-cd/casper/filesystem.manifest extract-cd/casper/filesystem.manifest-desktop
sudo sed -i '/ubiquity/d' extract-cd/casper/filesystem.manifest-desktop

sudo sed -i '/casper/d' extract-cd/casper/filesystem.manifest-desktop
```

12.8.2 Compress filesystem

For slightly higher compression at the cost of compression time, you can increase the block size:

```
sudo rm extract-cd/casper/filesystem.squashfs

sudo mksquashfs edit extract-cd/casper/filesystem.squashfs -b 1048576
```

Update the filesystem.size file, which is needed by the installer:

```
printf $(sudo du -sx --block-size=1 edit | cut -f1) > extract-cd/casper/filesystem.size
```

12.8.3 Remove old md5sum.txt and calculate new md5 sums

```
cd extract-cd

sudo rm md5sum.txt

find -type f -print0 | sudo xargs -0 md5sum | grep -v isolinux/boot.cat | sudo tee md5sum.txt
```

12.8.4 Create the ISO image

```
sudo mkisofs -D -r -V "v-bank live cd" -cache-inodes -J -l -b isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table -o ../image1.iso .
```

12.9 Enable TLS 1.2 in IIS

```
# Enables TLS 1.2 on Windows Server 2008 R2 and Windows 7
# June 27, 2010
# Version 1.2
# These keys do not exist so they need to be created prior to setting values.

md
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2"
md
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Server"
md
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Client"

# Enable TLS 1.2 for client and server SCHANNEL communications

new-itemproperty -path
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Server" -name "Enabled" -value 1 -PropertyType "DWord"
new-itemproperty -path
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Server" -name "DisabledByDefault" -value 0 -PropertyType "DWord"
```

```
new-itemproperty -path
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocol
s\TLS 1.2\Client" -name "Enabled" -value 1 -PropertyType "DWord"
new-itemproperty -path
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocol
s\TLS 1.2\Client" -name "DisabledByDefault" -value 0 -PropertyType "DWord"

# Disable SSL 2.0 (PCI Compliance)
md
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocol
s\SSL 2.0\Server"
new-itemproperty -path
"HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocol
s\SSL 2.0\Server" -name Enabled -value 0 -PropertyType "DWord"
```