Ontological Lockdown Assessment

A thesis presented in partial fulfilment of the requirements for the degree of

Master of Science

in

Information Technology

at Massey University, Palmerston North, New Zealand

Aaron Steele

2008

Abstract

In order to keep shared access computers secure and stable system administrators resort to locking down the computing environment in order to prevent intentional and unintentional damage by users. Skilled attackers are often able to break out of locked down computing environments and intentionally misuse shared access computers. This misuse has resulted in cases of mass identity theft and fraud, some of which have had an estimated cost ranging in millions.

In order to determine if it is possible to break out of locked down computing environments an assessment method is required. Although a number of vulnerability assessment techniques exist, none of the existing techniques are sufficient for assessing locked down shared access computers. This is due to the existing techniques focusing on traditional, application specific, software vulnerabilities. Break out path vulnerabilities (which are exploited by attackers in order to break out of locked down environments) differ substantially from traditional vulnerabilities, and as a consequence are not easily discovered using existing techniques.

Ontologies can be thought of as a modelling technique that can be used to capture expert knowledge about a domain of interest. The method for discovering break out paths in locked down computers can be considered expert knowledge in the domain of shared access computer security. This research proposes an ontology based assessment process for discovering break out path vulnerabilities in locked down shared access computers. The proposed approach is called the ontological lockdown assessment process. The ontological lockdown assessment process is implemented against a real world system and successfully identifies numerous break out path vulnerabilities.

Acknowledgements

Firstly, I thank Jesus. Also, my lovely wife Sina, my supervisors: Sven Hartmann and Sebastian Link. Thanks also go to Stephen Marsland, Elizabeth Kemp, and Patrick Rhyhart, my Church and all the people therein.

Last of all I thank everyone else who helped me during the course of this project.

This thesis is dedicated to one of the finest graduates of Ngaumu University, my granddad, Bob Coulson.

Publications

A publication related to this research is:

Steele, A. (2008). Ontological Vulnerability Assessment. *Proceedings of the International Workshop on Web Information Systems Engineering for Electronic Businesses and Governments (E-BAG 2008).* S. Hartmann et al. (Eds): WISE 2008, LNCS 5176, pp. 24-35, 2008. © Springer-Verlag Berlin Heidelberg 2008

Table of Contents

List of Figures

List of Tables

1 Introduction

From libraries to hotels shared access computers can be found almost everywhere. A shared access computer is exactly that, a computer that has its access shared between users. In order to keep shared access computers stable and secure system administrators resort to locking down the computing environment in order to prevent the user from intentionally or unintentionally damaging the system.

Often attackers are able to break out of locked down computing environments. This occurs when an attacker is able to discover a break out path (or break out path vulnerability). A break out path consists of a series of actions performed by an attacker that results in the attacker gaining access to unintended functions of the system. Once an attacker has broken out of a locked down computing environment the attacker is free to misuse and abuse the shared access computer. This has led to cases of identify theft and fraud involving millions of dollars and hundreds of different people.

Although numerous methods exist for locking down shared access computers, there does not currently exist an effective method for assessing shared access computers for break out path vulnerabilities. It is this problem of assessing shared access computers for break out path vulnerabilities that forms the focus of this research.

1.1 Research Objectives

This research has four main objectives. They are:

1. Highlight the unique security issues that are encountered when dealing with shared access computers, in particular the lockdown problem.
2. Reveal that existing vulnerability assessment tools are insufficient to assess whether shared access computers are adequately locked down.
3. Show that by developing an ontology for lockdown assessment a systematic approach for assessing shared access computers can be achieved.
4. Prove the usefulness of the developed approach with a case study.

1.2 Thesis Structure

This thesis is divided into seven distinct chapters. Each chapter is summarised below and tied back to a corresponding research objective where applicable.

Chapter 1 has introduced the context of the research and has also clearly stated the research objectives of this project.

Chapter 2 aims to meet the first research objective. This is achieved by first looking at what shared access computers are and why they exist. Secondly given the unique characteristics of shared access computers, the unique security issues are then detailed. Thirdly the potential attacks that inadequately locked down shared access computers are susceptible to are then explained. Finally, the unique security issues and potential attacks are tied back to the lockdown problem, and a summary of the problem is given.

Chapter 3 includes a review of current vulnerability assessment tools and techniques. This review aims to show that the existing vulnerability assessment tools and techniques were not designed for assessing shared access computers for breakout path vulnerabilities. As a consequence, it is shown that these existing tools and techniques are not suitable for assessing whether shared access computers are adequately locked down. The content of this chapter works toward satisfying the second research objective.

Chapter 4 proposes that the development of an ontology for lockdown assessment would provide a systematic approach for assessing shared access computers for breakout path vulnerabilities. The chapter examines information systems ontologies and highlights aspects that could be useful for lockdown assessment. The chapter also reviews the use of ontologies in information systems security. This is to add support for their use in solving the lockdown assessment problem. The content of this chapter works toward satisfying the third research objective.

Chapter 5 details the development of the lockdown assessment ontology. This is achieved by examining the breakout process and identifying the underlying principles.

The underlying principles are then converted into a lockdown assessment ontology. Finally details are given as to how the lockdown assessment ontology can be used to discover breakout path vulnerabilities on shared access computers. This chapter, together with the previous chapter, aims to meet the third research objective.

Chapter 6 consists of a case study where the lockdown assessment ontology is applied to a supposedly locked down shared access library catalogue computer. The case study shows the systematic application of the ontology to the target system and includes a discussion on the results of the assessment. This segment aims to meet the fourth research objective. Following from this, comparative results of other vulnerability sources, as applied to the same system as the case study are given in order to further emphasise their unsuitableness. This segment, together with the third chapter, aims to meet the second research objective.

Chapter 7 concludes the research by providing a summary of the positive and negative aspects of the developed solution to the lockdown assessment problem, as well as the limitations of the research. Following from this, suggestions are made concerning the practical application of the ontological lockdown assessment process and the future direction of this research.

2 The Lockdown Problem

In order to fully appreciate the need for locking down shared access computers and the problems that this process entails, this chapter will start by looking at what shared access computer are and why they exist. Next the unique security challenges of shared access computers will be explored, followed by an examination of some potential attacks. Building on this foundation, the chapter will conclude by summarising the lockdown problem.

2.1 Shared Access Computers

A shared access computer can be defined as: A computer which intentionally exists to be physically accessed and used by multiple, independent users. As this is a rather broad category, it is worth pointing out that this chapter speaks in generalities about shared access computers, in order to highlight the underlying themes and issues that are encountered.

Shared access computers exist in various forms, including: library computers, university lab computers, school lab computers, hotel guest computers, business centre computers, and internet café computers. Although these computers exist in different organisations for different reasons, they all have at least one commonality; their primary function is to provide a service for multiple independent users.

As stated these computers exist in different forms for differing purposes. Some of the common services provided by these computers include: internet access, library catalogue access, printing services, word processing tools, and even software development environments:

Depending on the organisation and purpose of the computer, the level of anonymity with relation to user access can also vary. In some situations, user access is not only shared but is also anonymous. An example of this situation can be found in library catalogue computers that exist in a permanently logged on state in order to provide immediate access to the library's catalogue. Other situations may provide semi-anonymous access. An example of this would be an internet café that issues temporary login credentials to paying users. Finally, other situations may require the use of predefined login credentials that eliminate anonymity, for example university lab computers, that require a username and password. This research focuses on those shared access computers that provide anonymous or semi-anonymous access.

Regardless of the computers actual purpose, maintenance of shared access computers usually revolves around a common goal, which is to provide a stable and consistent computing environment from user to user. This goal is usually approached by the use of lockdown and disk protection software.

Shared access computers are also usually connected to a larger local area network (LAN). This connection typically exists for remote maintenance, shared internet access, and shared access to other resources (e.g. printers, file servers, etc).

Given these unique characteristics, shared access computers also provide some equally unique security issues which will be explored in the next section.

2.2 Unique Security Issues

Computer security is usually portrayed within two popular areas. The first revolves around the threat of hackers breaking into computer systems from remote locations. The second revolves around the spread of computer viruses and worms. However, the main security issues that affect shared access computers tend not to belong to either of these categories. As a result, these issues are not often given adequate attention.

2.2.1 Perceived Value

The first issue worth highlighting concerns the perceived value of shared access computers within organisations. When securing the information infrastructure of an organisation, a common process that is employed is called risk analysis (this process is also referred to as risk management, or risk assessment, however to some these three terms can hold very different definitions). The risk analysis process can be presented a number of different ways [55, 57], but essentially it consists of five main steps [55].

- Asset Assessment
- Vulnerability & Threat Assessment
- Risk Assessment
- Implementation of Countermeasures
- Monitor Effectiveness

To provide a quick summary, in the first step the organisation's assets are listed and valued. Second, vulnerabilities of those assets and threats to those assets are identified. Third, risk is calculated by evaluating the value of each asset against the identified vulnerabilities and threats. Fourth, countermeasures are implemented in order to remove or reduce risk in order of priority. In the last step, the countermeasures are monitored to assess their effectiveness, and then revisions can then be made if necessary. Although much could be said about this process, at this stage only a brief examination of the first step is needed to highlight the first security issue concerning shared access computers. When compared to other components of an organisation's information systems infrastructure, shared access computers can appear to be of relatively little value. For example, a catalogue computer in a university library that only needs to provide users access to a single application (i.e. the catalogue) would typically have very basic system requirements, so far as hardware and software are concerned. Secondly, the machine itself would not need to store any sensitive information, as its function is simply as a terminal into the catalogue. Given these specifications, this shared access catalogue computer would generally be considered of a lesser value than other more mission critical components of the university network, such as: web servers, file servers, print servers, routers, or even simply, staff computers. This perceived low value is the first security issue for shared access computers because of its consequent effect on the risk analysis process. In effect, low value results in a low risk classification and a low risk classification results in a low level of security concern. This ultimately leads to a low level of security countermeasures.

2.2.2 Usability, Security and Cost Trade Off

The second issue encountered with shared access computers revolves around the usability, security and cost trade off, and also ties in with the previous issue of perceived value. Security practitioners have long been aware of this trade off between usability, security and cost, with entire books being devoted to the subject [26]. This trade off can be seen presented in a trade off triangle where only two of the three sides can exist simultaneously (see Figure 1) [37]. The trade off generally gives rise to three distinct outcomes. The first results in cheap, usable systems with poor security; the second results in cheap, secure systems that are difficult to use; and the third results in, expensive systems that are both usable and secure.

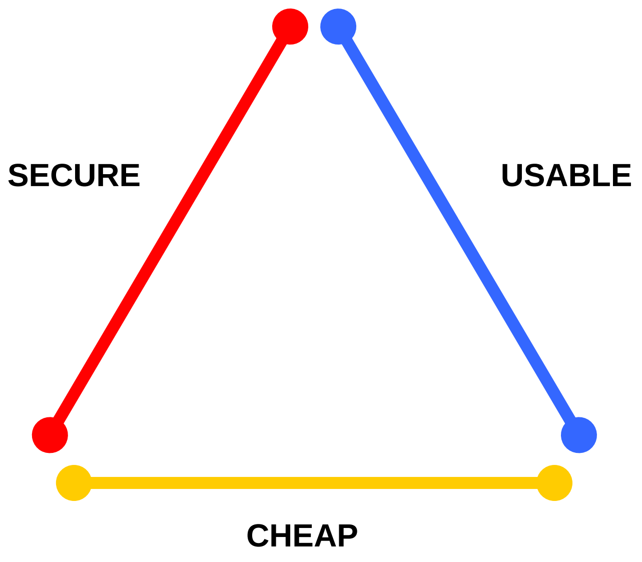


Figure 1. Usability, security and cost trade off triangle for computer security

Unfortunately, due to the characteristics of shared access computers, the outcome of this trade off usually leans toward cheap, usable computers with poor security. Building on the previous issue of a low perceived value, the trade off is reduced to one of the two cheaper options (i.e. shared access computers are not very valuable, therefore invest in other areas that are more valuable). Again, as a consequence of the low perceived value which ultimately results in a low risk classification, it makes sense that a cheap, usable system with minimal security countermeasures would usually be selected.

2.2.3 Insider Threat

The third issue facing shared access computers centres on a threat to network security which has been gaining more attention recently, the insider threat. Traditionally, threats to network security have originated from outside of the network by unknown attackers in remote locations. The insider threat is a threat to network security which originates from within the network by authorised users [29]. Some estimates indicate that as much as 60% of security breaches originate from insiders [58].

Research has indicated that attacks originating from insiders have a much higher success rate than those originating from external sources, and that these attacks can also go undetected [21]. Insiders benefit from many advantages that their external counterparts are forced to work without [21]. For example the often difficult task of overcoming perimeter defences such as firewalls do not need to be performed [39]. Furthermore, due to the fact that access has been granted to the insider, this conveys some level of trust [24], and with trust comes privilege to perform some level of action on the network which will not raise suspicions.

The insider threat problem is a significant issue with regards to shared access computers. This is due to the fact that freely available shared access computers (library catalogue computers for example) potentially allow any user to become an internal threat to network security. Furthermore, if the shared access computers in question share the same network as other, more significant network components (e.g. staff computers or servers), the level of concern should increase even more. This is due to the fact that with the advent of protocol analysis, or sniffer software, every station on a local area network can potentially be used as a network analysis tool or even as a surreptitious eavesdropper [40]. In addition, depending on the degree of anonymity provided by the shared access computers, this threat can become even more alarming.

In a worse case scenario, organisations with shared access computers can potentially be providing attackers with anonymous access to a workstation inside their network. From this workstation attackers could launch attacks on the rest of the network, with all the advantages of a network insider and without any of the disadvantages (i.e. due to the anonymous access, there is no way to trace the attacks back to a particular user).

2.2.4 User Information

The fourth issue to explore, regarding shared access computer security involves user information. Shared access computers can be seen as having very little value from an organisations point of view. Interestingly, the most valuable aspect of a shared access computer is not even the computer itself, but is often actually the information assets provided by the users of the machine. A good example is that of internet banking login credentials. Although a shared access computer may do nothing more than provide internet access to users, they can in effect, be used as a channel by which users access their bank accounts online. Aside from internet banking credentials, shared access computers often function as a channel for considerable amounts of sensitive information. Coupled with the number of different users passing this sensitive information through these shared access computers, the machines become a virtual thoroughfare of valuable user information.

Unfortunately for users of shared access computers, the risk analysis process comes back into play. The risk analysis process as referred to in the first issue of perceived value, begins by listing assets of value to the organisation. To start with, user information is not usually an asset of an organisation that provides shared access computers. Also, a compromise of the confidentiality of this valuable user information has little, if any, impact of the organisation itself. Sadly, although valuable in the sight of users, this valuable information still does not increase the value of shared access computers from an organisation’s point of view, according to the risk analysis process.

2.2.5 Dormant Technology

The final security issue facing shared access computer is concerned with dormant technology. Probably due to number of factors including Microsoft’s market share, compatibility, and familiarity, most shared access computers are locked down windows based machines. However, underneath the surface of these windows based shared access computers is a fully functioning Windows operating system. The standard installation of the Windows operating system (whether 2000, XP, or Vista) includes a number of powerful tools that have very useful and legitimate uses. At the same time, these built in tools can be misused by attackers to gather information, and launch attacks on other network components. Accordingly, these tools usually would not need to exist on shared access computers. For example, a shared access library catalogue computer would typically not need to have command line tools like: ipconfig, netstat, nslookup, tracert, ftp, arp or telnet tools installed on the system. Beyond this, even a simple text editor like notepad or the command line edit tool is all that is required to write malicious scripts or batch files.

This issue of dormant technology forms the final unique security issue for shared access computers. In the case of an attacker getting access to this dormant technology, substantial damage can occur, even without the introduction of external software.

The next section will examine some potential attacks on shared access computers. These attacks take advantage of one or more of the aforementioned unique security issues.

2.3 Potential Attacks

Shared access computers, due to their unique characteristics, are susceptible to a number of different attacks. These attacks can be divided into two distinct categories, according to the targets of the attacks. The first group of attacks target the users of the shared access computers and their valuable user information. The second group of attacks target the host network, resources and the information therein. This section will examine some of the potential attacks that can occur when an attacker can breakout of a locked down shared access computer. Each attack is explained with an example scenario that illustrates the impact of the attack. Additional information supporting the attack is also provided.

2.3.1 History

The first attack that will be examined is concerned with the actions of previous users of a shared access computer. This attack targets shared access computer users and their information. Consider a shared access computer that is set up only to allow users to surf the internet via a locked down web browser. The rest of the operating system has also been locked down so only the web browser window is available. An attacker who is able to breakout of the locked down environment could browse the file system to the web browser cache, and violate previous user privacy by viewing browser history and potentially even extract sensitive information.

The need to delete browser history when using shared access computers has been documented and is often found as a safety tip for using shared access computers [45, 51]. Unfortunately, in some situations particular shared access computers may be locked down to such an extent that deleting browser history is restricted [61].

2.3.2 Key Logging

Again, this attack also targets users of shared access computers and their information. Consider a shared access computer that allows users to surf the internet, use the MSN messenger chat program, and perform desktop publishing tasks with Microsoft Office. The rest of the operating system is locked down. An attacker who is able to break out of the locked down environment could install a key logger that would record the key strokes of subsequent users. This information would include login credentials for visited website, MSN messenger login credentials, personal chat transcripts, and all other text entered via the keyboard. The key logged information is then retrieved at a later date by the attacker, the log file cleared, and the key logger restarted. Due to the low value and low risk characteristics of the shared access computer it is months before the key logger is discovered, by which time the privacy of hundreds of users has been compromised.

Key loggers can exist in both hardware and software forms. Hardware key loggers can be installed in a matter of seconds, are impossible to detect with virus scanning software and can only be discovered by examining the connection between the keyboard and the machine [25]. Software key loggers can be freely downloaded, with installation being as simple as a double click on an exe.

Although the above scenario was hypothetical, there have been a number of documented cases of this exact scenario occurring. To add weight to the significance of this particular attack, a number of these documented cases will be given.

In 2003, Douglas Bodreau was charged with installing key logger software on over 100 computers at Boston College [9]. From this software, Bodreau was able to create a database of sensitive user information of over 4,800 people, that included 685 credit card and social security numbers, passwords and access codes to college buildings [9, 44]. With the access codes, Bodreau was also able to gain access to restricted areas of the university, where he further installed the key logger software [9].

In 2005, Juju Jiang was sentenced to 27 months in prison and ordered to pay $201,620 in restitution [34]. Jiang had installed key logger software on computer terminals located at Kinko’s stores throughout Manhattan [34] (Kinko’s is a chain of copy centre stores that also provide shared access computers to paying users). Jiang admitted to using the key logged information to access online bank accounts, and also to open new online bank accounts with stolen identities [34]. Jiang was able to collect over 450 internet banking login credentials from the Kinko’s terminals [59]. Interestingly, Jiangs crime succeeded, despite Kinko’s policy of re-imaging the computer terminals every week [59].

In January of 2006, key logger software was found installed on lab computers at Virginia Commonwealth University [8]. Upon investigation, it was determined the software was installed by an unknown user who had access to the lab computers [8]. The software had been installed several months earlier [8]. A month later, the same software was also found installed on lab computers in another part of the university [8].

The last case that will be documented is the most recent and also one of the most significant examples of shared access computer abuse that has occurred. In April of 2008, Mario Simbaqueba Bonilla was sentenced to nine years in prison [4]. Bonilla had illegally installed key logger software on shared access computers located in hotel business centers around the world [4]. Bonilla used this logged information to commit identity theft and fraud, victimising over 600 different people [4]. The estimated impact of the scheme was gauged at $1.4 million [4].

Each of these cases involved, to a greater or lesser extent, attackers taking advantage of shared access computers by installing key logger software, in order to compromise other users sensitive information.

2.3.3 Shares

Again, this attack also targets the host network of the shared access computer. Consider a shared access library catalogue computer that shares a network with a number of library staff computers. The catalogue computer is locked down to only allow catalogue access via a web browser. The network also provides staff with access to a shared folder that contains information relative to the organisation (e.g. staff phonebook, financial reports, etc). An attacker who is able break out of the locked down catalogue computer environment could gain access to, and potential even modify, the shared folder and its contents.

This attack again highlights the unique security issue of the insider threat. Without the presence of the shared access computer, having a globally accessible shared folder for staff is generally not an issue. However, introducing a shared access computer onto the same network brings that information within the reach of skilled attackers.

2.3.4 Sniffing

This attack targets the host network of the shared access computer. Consider a shared access lab computer at a high school that is connected to the same local area network as the high school’s staff computers. An attacker breaks out of the locked down lab environment and then installs and runs a network sniffer. The network sniffer then intercepts staff POP3 email credentials in plain text. The attacker then uses these credentials to access and send malicious emails from staff email accounts.

Network of packet sniffing software is the electronic equivalent to eavesdropping [57]. Information sent from one node to another node on a network is sent in the form of packets [57]. Sniffer software simply examines or intercepts these packets as they travel along the network medium [57, 64]. There currently exist a number of free and extremely powerful network sniffing software packages [48].

The network packet sniffing attack is a particular problem for an organisation with shared access computers, as it can potentially allow anonymous attackers access to sensitive, internal organisation information.

2.3.5 Scanning

This attack also targets the host network of the shared access computer. Consider a shared access computer in a university lab. An attacker breaks out of the locked down computing environment and then installs a port scanner from a USB drive and then runs a series of port scans on the host network. With this information, the attacker then researches the identified services for known vulnerabilities. Having discovered an old version of MySQL server running on the network, the attacker exploits the vulnerability and hijacks the server [22].

While network sniffing tools aim to exploit information in transit on a target network, network port scanners aim to identify open ports and vulnerable services running on various network nodes [57]. This information can be obtained from the network quietly and anonymously, and without identification or authentication [57]. Like network packet sniffing tools, network port scanners are also freely available for anyone to download and use, the most well known being Nmap [46]. Although traditional port scanners like Nmap are only designed for network reconnaissance (i.e. identifying names and versions of services and operating systems), a closely related category of tool, referred to as vulnerability scanners, go one step further and automatically identify known vulnerabilities [47].

A shared access computer that is not correctly locked down can potentially provide an attacker a doorway into the host network, where network scanning can be performed. The resulting information can then be used to internally attack certain network components.

2.3.6 Denial of Service and Vandalism

The final potential attack that will be examined, of which shared access computers are uniquely susceptible, is that of computer vandalism and denial of service. This attack, in effect targets the users of shared access computers. Consider a shared access computer in a library that has been set up for catalogue browsing, and some other limited web browsing. An attacker breaks out of the locked down browsing environment, opens notepad and creates two batch files that echo ‘hello’, then call each other. The attacker then resizes the screen from the monitor controls, hits left + shift, left + alt, PrintScrn to switch to high contrast mode (inverting every colour on the screen), then executes the batch files, creating a pointless, never ending cycle of flashing ‘hello’ messages. Finally, the attacker slightly unplugs the keyboard and mouse from the back of the machine and walks away.

This type of attack inhibits the shared access computer from performing its intended function and consumes the time of maintenance staff. Interestingly, this attack includes the simple abuse and misuse of the dormant technology that exists within most shared access computers.

2.4 Problem Summary

This section will summarise the lockdown problem, by first describing the locking down of computers and how, in an ideal situation, this can secure shared access computers. A brief summary of the tools and techniques used to lockdown computers will also be given. Finally, the section will end with a clear problem statement.

2.4.1 Locking down

All of the potential attacks that have been listed in this chapter can be individually combated from a technical point of view. However, from a practical point of view, this is not the reality with most shared access computers. Due to the low security priority commonly associated with shared access computers, combating these potential threats is usually reduced to cheap and non time consuming methods.

The de facto standard for securing shared access computers can be seen to consist of a combination of locking down the computing environment and implementing disk protection [36]. Lockdown can be seen as the first line of defence, with disk protection existing as the second line of defence. The idea of locking down a computer is to prevent the user from being able to intentionally or unintentionally damage or misuse the computer, so it remains stable from user to user. The idea of disk protection is to take an image of a clear system and constantly revert back to this image on each restart. This is to insure that if a user was somehow able to damage or alter the system, these alterations will be removed by the reversion back to the original clean image. This process thus aims to provide a consistent and stable computing environment from user to user. However, it is worth pointing out that the second line of defence, that is, disk protection, is only effective in situations where the computer is restarted from user to user. A situation where this does not occur (e.g. library catalogue computers that remain logged in all day) will reduce the effectiveness of this protection. For example, a key logger could be installed at the start of the day and the log file recovered at the end of the day before the computer is restored to the clean image. Secondly, disk protection does aim to restrict what the user is able to do while they are using the computer (e.g. sniffing, scanning, browsing shares, misuse of dormant technology, and some denial of service could all still easily occur on a disk protected shared access computer). Therefore, it can be seen that it is the first line of defence, that is, the locking down of the shared access computers, which aims to combat each of the potential attacks.

Hypothetically, a perfectly locked down shared access computer would prevent users from being able to perform anything other than the intended function of the computer.

However, ensuring that a locked down shared access computer only allows users the ability to perform the intended functions of the system is a difficult task.

To understand the problem more precisely, it is helpful to consider the situation in terms of functionality. When locking down a shared access computer, three sets of functions are present: intended functions (I), possible functions (P), and restricted functions (R). Collectively, these three sets combine as the set of all possible functions (A).

Figure 2 provides a graphical representation of a perfectly locked down computer (note: no functions exist in A other than those found in I, P and R. Figure 3 provides a graphical representation of a shared access computer that is not perfectly locked down (again, no functions exist in A other those found in I, P and R).

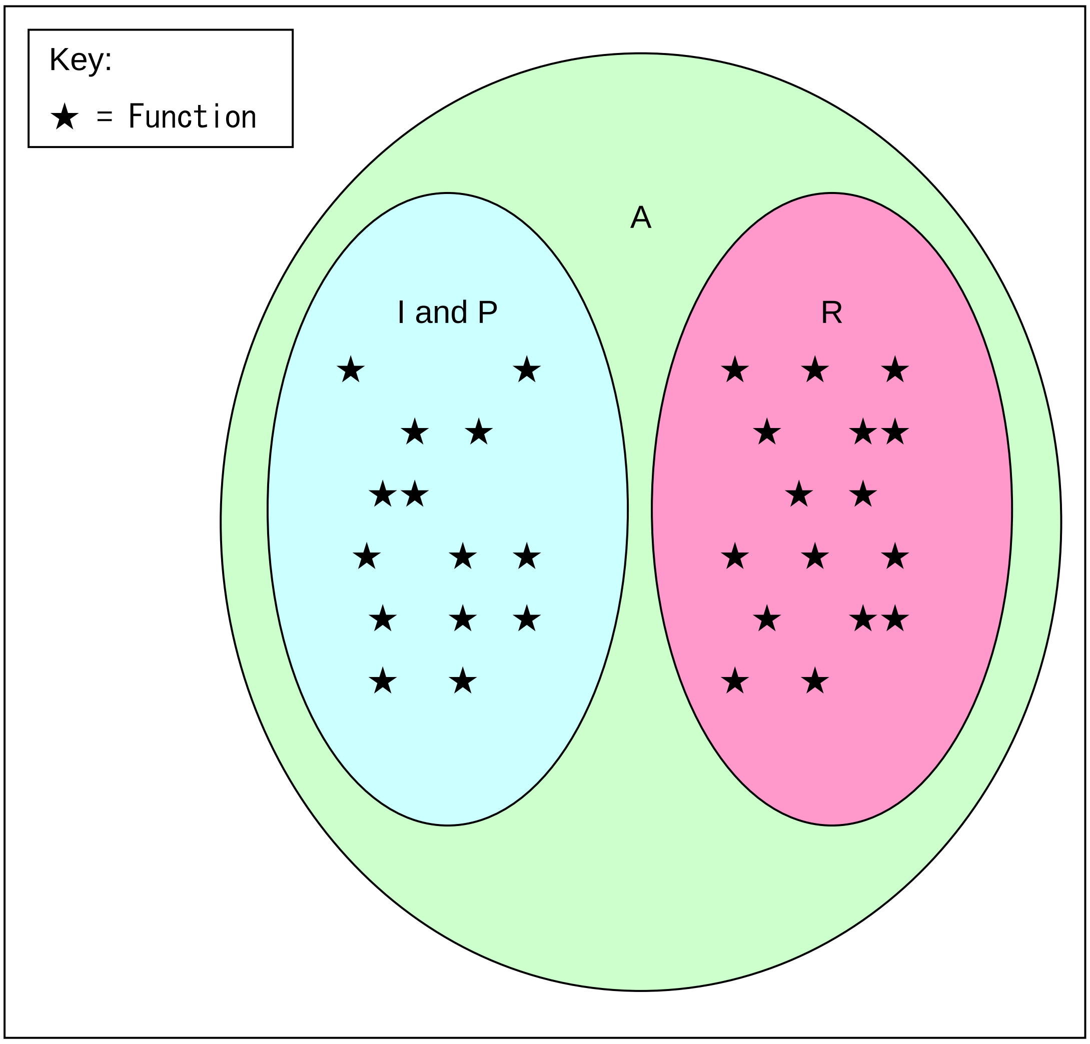


Figure 2. Function sets of a perfectly locked down shared access computer

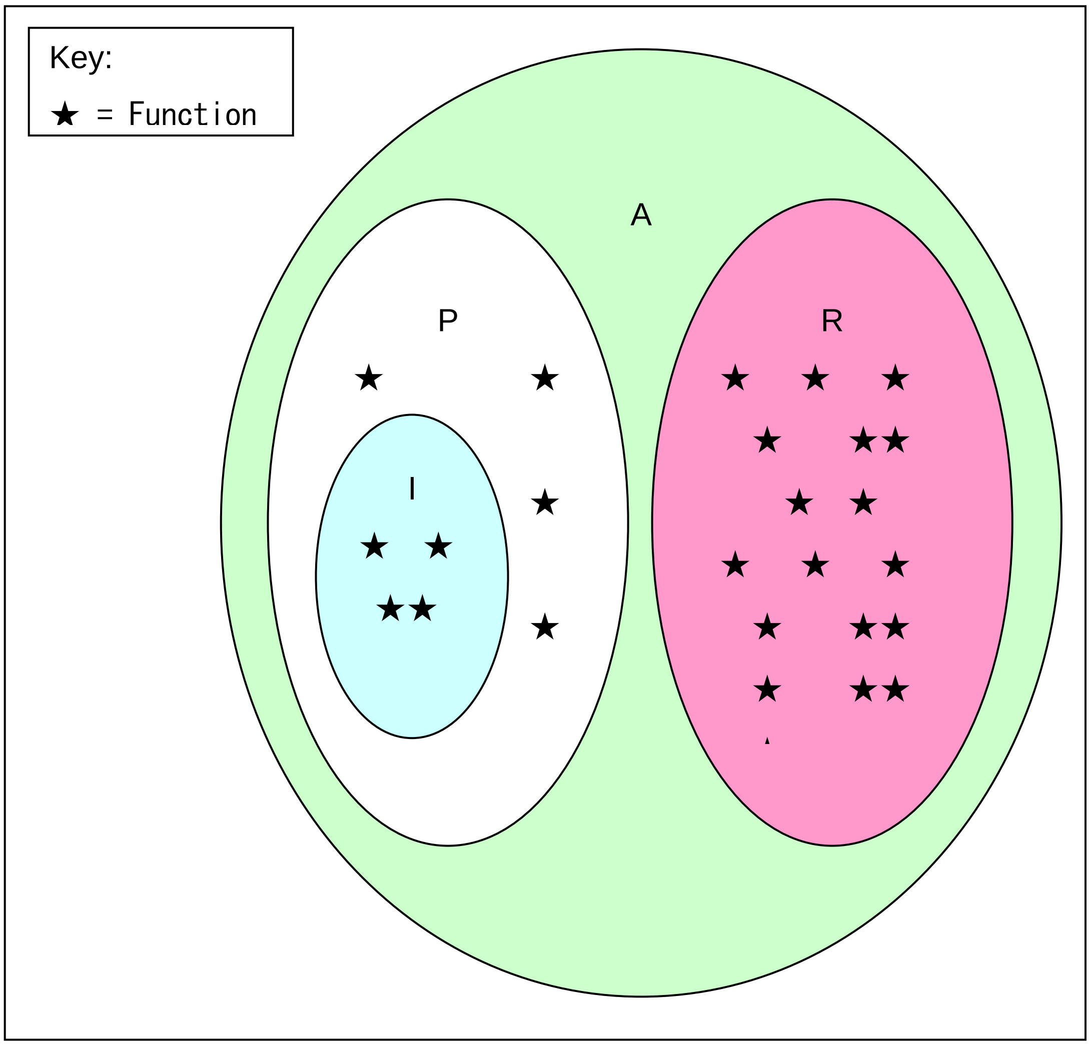


Figure 3. Function sets of an imperfectly locked down shared access computer

The set of intended functions represents those functions that the shared access computer should provide for the user. The set of possible functions encompasses the set of intended functions, but also includes all other possible functions that have not been restricted. These other possible functions can potentially be used by attackers to break out of locked down environments. The set of restricted functions are those functions that the shared access computer has been restricted from allowing. The union of the possible functions and restricted functions sets represents all possible functions (A) (this is due to the fact that the set of intended functions is a subset of the possible functions set). Also note that the intersection of the set of possible functions with the set of restricted functions is the empty set. If a shared access computer is perfectly locked down, that is, it only allows users to perform the intended functions, then the set of intended function equals the set of possible functions.

It is worth noting that in reality, concession can likely be made to allow some possible functions to exist that do not belong to the set of intended functions, so long as these functions do not enable the user to adversely affect the system.

2.4.2 Tools and Techniques

There are a number of different ways to lockdown shared access computers. Due to the market dominance of the Microsoft Windows operating system, the majority of lockdown software tools and techniques are developed for Windows based machines. Microsoft has itself produced a series of tools for locking down shared access computers. Initially, Microsoft produced the Public Access Computer Security Toolkit (although technically speaking this was created by the Bill & Melinda Gates Foundation [56]). Building on the inspiration of the Public Access Computer Security Toolkit, Microsoft then produced the Shared Computer Toolkit [56] (which also included a disk protection component). More recently Microsoft released Windows SteadyState, which is, in effect, version two of the Shared Access Toolkit [56]. A number of commercially available user restriction systems are also available including: Fortres 101 [5], WINSelect [3], WinLock and WinLock Professional [18, 19], Public PC Desktop [13], Executable Lockdown [1], Desktop Security Rx [1], and SiteKiosk [17]. Locking down shared access computers can also be achieved manually by editing the registry, using the group policy editor [61], and setting user permissions.

2.4.3 Problem Statement

Regardless of the tool or technique used to lock down shared access computers the goal remains the same, that is, to restrict the user functionality in order to provide a secure computing environment. Nevertheless, this research is not focused on the different ways shared access computers can be locked down. This research is focused on the following question: How can a locked down shared access computer be assessed to determine if the set of possible functions equal the set of intended functions, or that that the set of possible functions does not include any functions that could be abused by an attacker to break out of the locked down environment? Or more simply put:

> How can a locked down shared access computer be assessed to determine that no vulnerable break out paths exist?

3 Existing Vulnerability Assessment Techniques

In the previous chapter, the need for assessing locked down shared access computers for vulnerable break out paths was identified. This chapter examines the intentions and capabilities of existing vulnerability assessment methods, beginning with popular vulnerability assessment software tools and ending with the more general vulnerability assessment procedures. Ultimately, this chapter will reveal that these existing methods for vulnerability assessment are not suited for identifying break out path vulnerabilities in shared access computers, and that a more tailored solution is required.

3.1 Vulnerability Scanners

Vulnerability scanners are automated software tools that are designed to scan either a specific computer or network of computers for vulnerabilities. The following will provide an examination of these tools, with regards to break out path vulnerabilities.

3.1.1 Nessus

The Nessus vulnerability scanner claims to be the world-leader in active scanning [12, 47]. Nessus effectively works by executing NASL (Nessus Attack Scripting Language) scripts. The scripts are referred to as plugins. Each plugin is written to check for a specific known vulnerability or security flaw [11]. When a vulnerability is discovered in a particular software version (e.g. a programming error in MySQL 3.22 allowed remote attackers to bypass password authentication and access a database via short check string [22]), a script or plugin is written that checks for that version of the software on the host network. If the corresponding version is found, Nessus reports the finding as a vulnerability on the network. As of August 2008, Nessus boasted 23,267 different plugins, divided into 45 different categories [11]. The system is designed to be run by network administrators, who use Nessus to remotely check for each of the known vulnerabilities across their host networks. The system can also be run locally to check for the same vulnerabilities on the local machine.

Due to the intended function, and method of vulnerability discovery, Nessus is not suited for discovering break out path vulnerabilities in shared access computers.

The first reason to support this claim is the fact that Nessus is aimed at discovering vulnerabilities that are exploitable from across the network (i.e. what ports are open, what services are running on the open ports, and are there any known vulnerabilities). Break out path vulnerabilities are not exploited by an attacker from across the network, they are exploited by an attacker standing in front of the shared access computer who is interacting with the desktop environment.

The second reason that supports the unsuitableness of Nessus for discovering break out path vulnerabilities, is the way that Nessus plugins typically detect vulnerabilities. As already mentioned, Nessus typically discovers vulnerabilities by identifying and comparing version numbers of specific software packages. Break out path vulnerabilities are not caused by programming errors in specific software packages, nor are they restrained to individual software packages or versions. Therefore, break out point vulnerabilities are not able to be detected by the typical Nessus method. Break out path vulnerabilities involve a series of state changing actions, often utilising legitimate functions of numerous applications, in order to achieve a broken state.

The third reason supporting the unsuitableness of Nessus for break out path vulnerability assessment is the complexity and varying nature of break out path vulnerabilities. A break out path vulnerability on one shared access computer may be a completely acceptable action on another shared access computer. For example, being able to open documents from a USB drive and print these documents from a shared access university lab computer that students pay for, may be completely acceptable. At the same time being able to open files from a USB drive and print documents from a free anonymous access library catalogue computer may not be acceptable at all. If a script or plugin could be written to check for break out path vulnerabilities, it would need to be tailored depending of the intended function of the shared access computer on which it was being executed. The Nessus plugin scripts are centrally maintained by Tenable networks security, so the tailoring of scripts for low value shared access computers seems infeasible.

The final reason to support the unsuitableness of Nessus for break out path vulnerability assessment involves the order of creation of Nessus plugins. A Nessus plugin is created after a vulnerability has been identified and documented in a particular software package. Due to their uniqueness to any given shared access computer and its intended function, break out path vulnerabilities have not been identified or documented in this manner. Therefore, there is no explicitly known vulnerability that could be checked for by a Nessus plugin.

The above reasons highlight a paradigm mismatch. Attempting to use Nessus to discover break out path vulnerabilities in shared access computers is contrary to its intended use and capabilities.

3.1.2 GFI LANguard

GFI LANguard Network Security Scanner (N.S.S.) claims to be the number one Windows commercial security scanner [6, 47]. GFI LANguard N.S.S. promotes the following three features: vulnerability scanning, patch management, and network auditing [7]. The vulnerability scanning feature scans a host network by IP address and performs over 15,000 checks for known vulnerabilities. The vulnerability checks are based on a database that ships with GFI LANguard N.S.S., which is regularly automatically updated, and is based on information from online vulnerability databases [7]. Based on the results of the vulnerability scan, the tool then provides recommendations for the download and installation of necessary patches [7]. Finally, the tool is able to provide the user with a current view of the network status (i.e. what’s connected, installed, open shares, etc).

Being commercially in direct competition with Nessus, GFI LANguard is also unsuitable for identifying break out path vulnerabilities in shared access computers, for many of the same reasons as Nessus.

Again, the first reason to support this claim is the fact that GFI LANguard, like Nessus, is aimed at discovering vulnerabilities that are exploitable by attackers from across the network. Break out path vulnerabilities, as described in the Nessus section, are not exploited by attackers from across the network, but by attackers standing in front of shared access computers.

The second reason that supports the unsuitableness of GFI LANguard is similar to the issues found with Nessus, concerning the method used for discovering vulnerabilities. In a similar fashion to Nessus, GFI LANguard scans a host network looking for vulnerabilities. The vulnerabilities that are checked for, are based on previously identified and documented vulnerabilities that have been reported to online vulnerability databases such as CVE, OVAL, and BugTraq [7]. Again, online databases document vulnerabilities that are specific to software packages and versions. As stated in the Nessus section, break out path vulnerabilities do not fit with this paradigm.

The third reason involves the GFI LANguard solution to identified vulnerabilities. The patch management feature of the tool aims to remedy any identified vulnerabilities by recommending the download and installation of patches. Break out path vulnerabilities, due to their complex nature, cannot be remedied by patch installation, but usually require the refinement of system settings and user restrictions.

As with Nessus, the above reasons highlight a paradigm mismatch. GFI LANguard has been designed for network vulnerability assessment and using it to identify break out path vulnerabilities would be diverging from its intended purpose.

3.1.3 Other Vulnerability Scanners

Although there are a number of other free and commercially available vulnerability scanning software packages, they all suffer from the same paradigm mismatch as Nessus and GFI LANguard. These tools include but are not limited to: Retina [14], Core Impact Pro [2], Sara [16], SAINT [15], and MBSA [10].

The essential problem can be traced back to the location of the attacker. Each of these vulnerability assessment tools work on the premise that an attacker will be looking to exploit vulnerabilities from a remote location across a network medium. The fundamental difference between break out path vulnerabilities and the vulnerabilities identified by these tools, is that break out path vulnerabilities are only exploitable by attackers physically accessing the shared access computer.

3.2 Vulnerability Assessment Procedures

Vulnerability assessment is a procedure or process which is commonly found within the larger process of risk analysis [42, 55, 57]. While vulnerability scanners can be thought of as low level technical tools, the vulnerability assessment process can be thought of as a high level non-technical procedure, performed and managed by security professionals. This process involves multiple sources including: vulnerability databases, organisational policy, user interviews, and system requirements. What follows is an examination of existing vulnerability assessment procedures, with relation to break out path vulnerabilities of shared access computers. The name of the risk analysis process that each vulnerability assessment process is a part of will be used to distinguish between the different vulnerability assessment processes.

3.2.1 NIST Risk Management Guide

The Nation Institute of Standards and Technology in Special Publication 800-30 documents a Risk Management Guide for Information Technology Systems [63]. The guide defines risk assessment as the first process in the risk management methodology [63]. The risk assessment process is outlined as having the following nine steps [63]:

- Step 1 – System Characterization
- Step 2 – Threat Identification
- Step 3 – Vulnerability Identification
- Step 4 – Control Analysis
- Step 5 – Likelihood Determination
- Step 6 – Impact Analysis
- Step 7 – Risk Determination
- Step 8 – Control Recommendations
- Step 9 – Results Documentation

Step 3, Vulnerability Identification will form the focus of this examination, although the guide indicates that after the first step has been completed steps, 2, 3, 4 and 6 can be conducted in parallel [63].

The stated goal of the Vulnerability Identification step is to develop a list of system vulnerabilities that could be exploited by potential threat-sources (e.g. attackers). The guide also defines vulnerability as “A flaw or weakness in system security procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited) and result in a security breach or a violation of the system’s security policy” [63] (p. 14). The guide recommends three methods for identifying system vulnerabilities: examining vulnerability sources, performing system security testing, and the development and use of a security requirements checklist [63]. Each of these methods will now be examined individually.

The vulnerability sources that are recommended include: questionnaire results, on-site interview feedback, policy documents, system documentation, security related documentation, and results from automated scanning tools [63]. The questionnaire and on-site interviews are performed during the first step of system characterization and are aimed at obtaining security related information from the users of the system[63]. The questionnaire and interview results could potentially reveal break out path vulnerabilities. This would occur only if a questionnaire or interview respondent had become aware of such a break out path during the course of their use of a shared access computer. Policy, system, and security related documentation covers what is already known from a security standpoint, concerning the system components. Due to the uniqueness, complexity and varying nature of break out path vulnerabilities from system to system, these vulnerabilities are not easily documented and would be unlikely found in the given documentation. The automated scanning tools refer to those tools examined in the previous section, and are thus not suitable for discovering break out path vulnerabilities.

The system security testing methods that are suggested include: security testing and evaluation, penetration testing, and in an overlap between methods, automated vulnerability scanning tools are again also included [63]. Security testing and evaluation aims to test the security controls or countermeasures, to determine if they are performing as required by the organisation. It is suggested to employ test scripts and test procedures, with a list of expected results. A test procedure for assessing the shared access computers of an organisation to see if they are locked down, could fit into this category. However, the guide provides no further details on how to create these test procedures, possibly for the unstated reason that each test procedure would need to be tailored for each given system. Penetration testing aims to complement the security testing and evaluation method by testing a systems ability to withstand intentional attempts to circumvent the system security [63]. Penetration testing takes the view-point of the attacker, in an attempt to identify potential failures in the system. This method could possibly also identify break out path vulnerabilities in shared access computers. However, the successfulness of this approach would depend entirely on the tester's understanding of break out path vulnerabilities and their ability to identify them in a given shared access system.

The purpose of the security requirements checklist is to determine if the overall system is doing what it should be doing, in regards to security requirements. The checklist covers three categories: management, (this involves: assignment of responsibilities, background checks, separation of duties, etc [63]), operational (this includes: control of air-borne contaminants, humidity control, temperature control, etc [63]), and technical (this includes: cryptography, object reuse, system audit, etc [63]). The guide refers to the check list as containing basic security standards that can be presented in a table format, listing each security requirement and an explanation of how the system does or does not satisfy the requirement [63]. It is suggested that the checklist be built from sources like government regulatory and security directives [63]. This checklist appears to be a tool for gauging the overall compliance of an organisation information infrastructure to a given set of standards. Consequently, this checklist is not a tool that could be used for discovering break out path vulnerabilities in shared access computers.

To summarise, the approach to vulnerability assessment found in the NIST Risk Management Guide for Information Technology Systems seems to cover a very broad spectrum of vulnerability issues from high-level non-technical compliance problems through to application specific vulnerability scanning. Although most of the techniques are unsuitable for the discovery of break out path vulnerabilities in shared access computers, there did appear to be a couple of techniques that were focused in a useful direction. These techniques were: the employment of user questionnaires and interviews, system test procedures, and penetration testing. The common thread with each of these techniques is the presence of a tester or user, interacting with a given system in order to identify vulnerabilities. In the case of questionnaires and interviews, this interaction is retrospective, and the value is found in what a user may have already discovered during their use of the system with regards to break out path vulnerabilities. System test procedures and penetration testing appear useful, as they imply a proactive approach by a tester to perform system specific testing. This testing however, would need to be tailored to the given requirements of a given shared access computer in order to be useful. Secondly, the tester would also need a reasonably good understanding of how to discover break out path vulnerabilities in order to produce satisfactory results. Although each technique is focused in a useful direction, they each require a certain degree of improvement and refinement in order to become useful for the discovering of break out path vulnerabilities.

3.2.2 FRAP

Facilitated Risk Analysis Process (FRAP) was created by Thomas Peltier and is detailed in his book 'Information Security Risk Analysis' [55]. The initial motivation behind the FRAP was to create a risk assessment process that could be conducted by businesses themselves, a process that took weeks, as opposed to months, and utilised in-house expertise [55]. The idea of the FRAP is to assemble a FRAP team that consists of representatives from both the business side, and the technical systems side of the organisation [55]. This team is then led through the risk assessment process by the facilitator. The process itself has four phases: a pre-FRAP meeting, the FRAP session, FRAP analysis and report generation, and a post-FRAP meeting. The second phase, the FRAP session, is the phase where vulnerability assessment occurs, being performed by the FRAP team. The other three phases do not involve the FRAP team, nor the identification of vulnerabilities.

As mentioned, vulnerability assessment occurs in the second stage of the FRAP session. The way in which this occurs is through an ordered brainstorming session. The brainstorming session takes each security attribute (confidentiality, integrity, and availability) and aims to identify ‘risks, threats, concerns, and issues for each’ [55]. As a guide, each member of the FRAP team is given a definition of each security attribute (confidentiality, integrity, and availability), along with some samples threats [55]. Figure 4 provides an example of a FRAP brainstorming guide [55]. These concerns are then given a vulnerability rating (the suggested values are high, medium or low) [55].

Brainstorming Definition and Sample Risks

Examples of Risks (Not a complete list)

Threats to Confidentiality

- Access without authorization
- Disclose without authorization
- Observe or monitor transactions
- Copy without authorization
- Packet sniffing on network
- Contractor accessing confidential information

Definition:

Confidentiality: information has not undergone unauthorized or undesirable disclosure.

Figure 4. FRAP Brainstorming Guide [55] (p. 78)

In the overview of the FRAP, the author states that the “(FRAP) team relies on its general knowledge of threats and vulnerabilities obtained from national incident response centres, professional associations, and literature, and their own experience” [55] (p. 70). The aim is to utilise the background knowledge of those involved with the system being assessed, in order to identify vulnerabilities. The obvious down side to this approach, with regards to break out path vulnerability identification, is the total dependence the assessment technique has on the knowledge and expertise of the FRAP team. Further to this point, the FRAP requires that this background knowledge of vulnerabilities be exercised retrospectively during the brainstorming session, in which the FRAP team is given only three minutes per security attribute (confidentiality, integrity, and availability) to list relevant concerns [55]. Although this approach utilises numerous users of the system, it appears that a degree of thoroughness has been sacrificed in order to save time. Furthermore, the general nature of the example risks (see Figure 4) may not stimulate the identification of break out bath vulnerabilities beyond statements like ‘unauthorised printing from catalogue computers’), although this is simply conjecture, and each case would be unique, depending on the system being assessed and the characteristics of the FRAP team.

The Facilitated Risk Assessment Process communicates a quick solution to vulnerability assessment that aims to utilise the collective knowledge of the system users. It is interesting to note, that with typical shared access computers, the primary users of the systems are usually not employees of the host organisation, but are more commonly public or anonymous users with perhaps little vested interest in the host organisation. Involving adequate users from this group may prove difficult. Secondly, as a consequence of the FRAP team environment, users who do not belong to the organisation (if included), could potentially learn of ways to exploit the system.

For the FRAP to be more useful for identifying break out path vulnerabilities, the brainstorming guidelines could be tailored to the organisations shared access computers and their unique environment. However, again this tailoring would require someone with adequate understanding of break out path vulnerabilities and how to identify them on shared access computers.

3.2.3 VAM

The vulnerability assessment & mitigation methodology, or VAM was the result of research conducted by the RAND National Defense Research Institute [20]. Although the name may suggest a methodology focused solely on vulnerabilities, a closer inspection shows that it is actually a complete risk assessment methodology. The authors explain that the motivation for the research that resulted in the VAM methodology, was that most existing approaches took a bottom-up historical approach in identifying vulnerabilities that were already known, and did not offer the ability to discover new vulnerabilities [20]. By contrast, the authors state that the VAM methodology takes a top-down approach, that not only seeks to uncover known vulnerabilities, but also vulnerabilities that, as yet, have not been exploited or exercised in operation, by asking questions outside the range of known vulnerabilities [20]. The approach attempts to achieve this top-down approach by the use of a vulnerability matrix (Figure 5), which maps security attributes to system assets (or objects) [20].

		Object of Vulnerability			
		Physical	Cyber	Human/Social	Enabling Infrastructure
	Attributes:	Hardware (Data Storage, Input/Output, Clients, Servers), Network and Communications, Lottery	Software, Data, Information, Knowledge	Staff, Command, Management, Policies, Procedure, Training, Authentication.	Ship, Building, Power, Water, Air, Environment
Design/Architecture	Singularity				
	Uniqueness				
	Centrality				
	Homogeneity				
	Separability				
	Logic/ implementation errors; fallibility				
	Design sensitivity/ fragility/limits/ finiteness				
	Unrecoverability				
Behaviour	Behavioural sensitivity/fragility				
	Malevolence				
	Rigidity				
	Malleability				
	Gullibility/ deceivability/naiveté				
	Complacency				
	Corruptibility/ controllability				
General	Accessible/ detectable/ identifiable/ transparent/ interceptable				
	Hard to manage or control				
	Self unawareness and unpredictability				
	Predictability				

Figure 5. The VAM Vulnerability Matrix [20] (p. 27)

To perform the vulnerability assessment, an assessor is required to complete the vulnerability matrix by thoroughly considering each asset (or object) relative to each security attribute [20]. It is also recommended that each security attribute be considered at different levels of abstraction [20]. The authors point out very clearly that "successful vulnerability assessment requires the insights and experience of system users and developers" [20] (p. 12). This creates a similar predicament as that found in the FRAP, where the expertise and knowledge of the users becomes a primary information source for vulnerabilities. Again, the ability to discover break out path vulnerabilities in shared access computers will depend entirely on the ability of those people completing the vulnerability matrix.

The VAM methodology, like the FRAP, covers a very broad spectrum of vulnerabilities that it aims to identify. Due to the complexity and varying nature of break out path vulnerabilities from system to system, the use of the VAM vulnerability matrix (Figure 5) to discover break out path vulnerabilities would prove difficult. This is due to the VAM vulnerability matrix being tailored to discover a wide range of vulnerabilities, as opposed to those unique to shared access computers.

3.2.4 Pfleeger & Pfleeger

The last approach to vulnerability assessment to be considered is found in the book *Security in Computing* by Pfleeger and Pfleeger [57]. The approach outlined in the risk analysis section of the book is a summarised collection of popular and common vulnerability assessment techniques and is therefore a fitting place to end this review.

The authors explain that risk analysis can be performed in many different contexts, but that risk analysis for security places special emphasis on the kinds of problem that arise from security issues [57]. Although the authors acknowledge that different organisations take slightly different approaches to risk analysis, they claim the basic activities are the same, listing the following as the six basic steps for risk analysis [57]:

1. Identify Assets
2. Determine Vulnerabilities
3. Estimate likelihood of exploitation
4. Compute expected annual loss
5. Survey applicable controls and their costs.
6. Project annual savings of control.

The second step of determining vulnerabilities (vulnerability assessment) will form the focus of this review. As an introduction into vulnerability assessment, the authors begin by saying that imagination is required in order to determine the potential damage that might occur to system assets if attacked [57]. In order to aide in the determination of vulnerabilities, the authors suggest that developing a clear understanding of the nature of vulnerabilities is required [57]. It is suggested that this nature itself is derived from the need to ensure the three basic goals of computer security: confidentiality, integrity, and availability [57]. Thus, the authors conclude that a vulnerability is any situation where either of these three security goals is can be compromised [57].

The use of a matrix (Figure 6) for organisation and to stimulate thinking about vulnerabilities is recommended by the authors. When considering each matrix entry, the authors suggest considering a number of questions which include[57]:

- What are the effects of unintentional errors?
- What are the effects of wilfully malicious insiders?
- What are the effects of outsiders?
- What are the effects of natural and physical disasters?

Asset	**Confidentiality**	**Integrity**	**Availability**
Hardware			
Software			
Data			
People			
Documentation			
Supplies			

Figure 6. Assets and Security Properties [57] (p. 529)

The authors show that when filled in, the matrix can show certain general problems that can affect the assets of a computer system [57].

A number of other vulnerability assessment tools and techniques are also referenced by the authors, including hazard analysis techniques, integrated vulnerability assessments which is a process used by the U.S. Navy [57], CARVER (The Criticality, Accessibility, Recuperability, Vulnerability, Effect, and Recongnizability method), a method that assigns numerical ratings to each vulnerability, and even the VAM vulnerability matrix [20, 57]. However, the authors concede that there is no simple checklist or easy procedure for listing all possible vulnerabilities in a given system, and that tools like vulnerability matrices are only able to help by providing a structured way to think about a given problem [57]. The authors also note that by viewing examples of actual vulnerabilities, one can be trained to think of harm that can occur [57].

The view on vulnerability assessment provided by Pfleeger and Pfleeger appears to be focused at a more general system wide level, as opposed to vulnerabilities as specific as break out path vulnerabilities in shared access computers. It is interesting to note that the authors claim no value in using matrices other than for helping structure thoughts on vulnerabilities. As has been the case with the other vulnerability assessment procedures that have been reviewed, the Pfleeger and Pfleeger approach covers the areas where break out path vulnerabilities could be found, however does not provide a specific process for actively and systematically identifying break out path vulnerabilities in shared access computers.

3.3 Summary of Existing Techniques

The review of existing vulnerability assessment techniques given in this section has explored both vulnerability scanners, and more general vulnerability assessment procedures.

The vulnerability scanners, although automated and systematic in their approach to identifying vulnerabilities, were limited by their reliance on vulnerability repositories. This restricted the vulnerability scanners from identifying not only unique system dependant break out path vulnerabilities but also any other system vulnerabilities that were not documented in their repositories.

The vulnerability assessment procedures that were reviewed had a general advantage over the vulnerability scanners in that they were not directly dependant on any vulnerability repositories. Further too their advantage, the common intention was to apply the given procedures to unique systems with the aim of finding vulnerabilities specific to those systems. This application typically involved the use of a vulnerability matrix or similar tool, to stimulate the identification of vulnerabilities. These tools generally aimed to map system assets against the security attributes: confidentiality, integrity, availability (in the VAM matrix more specific attributes were used). The common downside to these more general procedures was the fact that the success of the assessment depended almost entirely on the ability of the assessor to identify vulnerabilities. This need for experience and expertise in finding vulnerabilities was also noted within the documentation for some of the vulnerability assessment procedures themselves.

From this review, it can be seen that the existing methods for vulnerability assessment are not optimally suited for identifying break out path vulnerabilities. In order to ensure that shared access computers are correctly locked down, a vulnerability assessment technique that is tailored to break out path vulnerabilities of these systems is required. Based on the review, a number of probable characteristics of this tailored technique can be identified, they include:

- Systematic: the technique should systematically assess the system for break out path vulnerabilities.
- System specific: the technique should be able to take into consideration a given systems unique requirements.
- Standalone: the technique should not be dependant on a vulnerability repository.
- Minimal user dependence: the technique should not be entirely dependant on the knowledge or ability of the user.

Therefore, it is suggested that a vulnerability assessment technique with the above characteristics would be more suitable for the identification of break out path vulnerabilities in shared access computers.

4 An Ontological Solution

The previous chapter identified the need for an improved vulnerability assessment technique that would be: systematic, system specific, standalone, and have minimal user dependence. This chapter proposes that an ontology for break out path vulnerability assessment will adequately fulfil these requirements. The chapter will begin by briefly reviewing ontologies in information systems, giving emphasis to the features which are potentially advantageous for vulnerability assessment. This will be followed by an examination of how ontologies have already been used in the security field. Finally, based on the supporting references, an ontology for lockdown assessment will be outlined. The aim of this chapter is to justify the selection of ontologies as a potential solution to the lockdown problem.

4.1 Ontologies

Originally a philosophical discipline concerned with nature and organization of what exists, ontology has in recent years been adopted by the information systems community to deal with the nature and organization of what exists in the information systems domain. [69]. Perhaps due to its philosophical origins, some debate exists concerning the precise definition of the term ontology. Therefore, the first order of business will be to consider some popular definitions and settle on a definition that will be sufficient for this thesis.

4.1.1 Definition

The most often quoted definition for ‘ontology’ is Gruber’s, which states that an ontology is “an explicit specification of a conceptualization” [32]. This is possibly the most often quoted definition for two main reasons. Firstly, it is concise. Secondly, it was published in 1995 and was one of the first of such definitions. In order to aide in understanding this definition, it is useful to also include Gruber’s explanation of the term conceptualization. Gruber defines a conceptualization as the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them [32]. Gruber also states that a conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose [32]. He also adds that every knowledge base, knowledge based system, or knowledge level agent is committed to some conceptualization, either explicitly or implicitly [32]. Returning to the original definition, an ontology is therefore the explicit specification of the objects, concepts, and entities of an area of interest and the relationships that hold among them.

A longer definition for the term is offered by Guarino, who attempts to refine the Gruber definition [33]. Guarino states that an ontology is a logical theory accounting for the intended meaning of a formal vocabulary [33], i.e. it’s ontological commitment to a particular conceptualization of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization), by approximating these intended models [33]. Guarino’s refinement is motivated by the argument that an ontology can only weakly represent a conceptualization, and that an ontology could allow models other than those intended [33].

Zuniga builds on the previous two definitions in the hope of providing a unified definition that is interdisciplinary understandable [69]. Zuniga states that an IS (information systems) ontology is an axiomatic theory, made explicit by means of a specific formal language. The IS ontology is designed for at least one specific and practical application. Consequently, it depicts the structure of a specific domain of objects, and it accounts for the intended meaning of a formal vocabulary or protocols, that are employed by the agents of the domain under investigation [69].

In 2004 the World Wide Web Consortium (W3C) made a recommendation for a Web Ontology Language (OWL) [50]. The recommendation describes an ontology as a definition of the terms used to describe and represent an area of knowledge [35]. The recommendation also states that ontologies include computer-usable definitions of basic concepts in a domain and the relationships among them [35]. The W3C recommendation also lists three concepts that ontologies need to be able to specify. These are [35]:

- Classes (general things) in the many domains of interest

- The relationships that can exist among things
- The properties (or attributes) those things may have.

Taking into consideration the above definitions and descriptions of ontology, the following definition, which is based on the Gruber definition [32], will be used to define an ontology.

Ontology: An explicit definition of the classes, relationships, and properties that collectively represent the concepts of a domain.

This definition has simply extended the Gruber definition by including the concepts listed by the W3C. Next, the features of ontologies that are potentially useful for break out path vulnerability assessment will be explored.

4.1.2 Features

Depending on the area of application, ontologies can have various beneficial features. The following will highlight those features of ontologies which could be useful for solving the lockdown assessment problem.

- Expert knowledge capture: Domain experts are able to perform complex tasks in their respective fields. This is commonly achieved by experts exercising substantial explicit and tacit knowledge about their domain of interest [43]. Ontologies aim to formally represent a domain on interest [32]. This representation can formally capture expert knowledge about the domain [43]. This is beneficial for break out path vulnerability assessment. Potentially, the expert knowledge that attackers exercise to break out of locked down computing environments could be captured with an ontology.
- Knowledge sharing: Ontologies can be used to facilitate knowledge sharing about specific domains of interest [43]. An ontology that represents attackers expert knowledge of breaking out of locked down computing environments could be shared amongst non-experts. The sharing of this expert knowledge amongst non-experts could help non-experts assess shared access computers for break out path vulnerabilities.

- Training: Ontologies can be used to aide in the training of challenging topics [62]. Not only could an ontology for break out path vulnerability assessment be used by non-experts to assess shared access computer systems, but it could also be used to train non-experts with the underlying expert knowledge.
- Minimal encoding bias: Ontologies should be specified at the knowledge level without depending on a particular symbol-level encoding [32]. If an ontology for break out path vulnerability assessment is developed with minimal encoding bias, it could potentially be universally useful for all types of shared access computers. This could result in a break out path vulnerability assessment technique that would be specific to shared access computers, independent of operating system or intended function.
- Extendibility: Ontologies should offer a conceptual foundation for a range of anticipated tasks, and they should be designed in such a way that they can be easily extended and specialised [32]. Having an ontology that is extendible would be beneficial for break out path vulnerability assessment, due to the varying nature of break out path vulnerabilities from system to system.

Interestingly, ontologies are perhaps better known for their connection with the semantic web and their potential to present knowledge in a machine readable format, which can be automatically reasoned about [50]. Although these are prominent features of ontologies, they are not the features that have led to the selection of ontologies for break out path vulnerability assessment. Next, is a review of how ontologies have already applied to the domain of security.

4.2 Ontologies in Security

Although the use of ontologies in information systems is on the rise, it is still a relatively new discipline. Accordingly, the amount of research that has been focused on the application of ontological concepts to vulnerability assessment is limited. However, the amount of research focused on the application of ontological concepts into the more general domain of information systems security is somewhat more substantial. This more general literature provides useful insights into the application of ontologies in security and is therefore worth examining. Following is a brief review of existing security related ontologies. Some aspects of these existing ontologies are tied back to break out path vulnerability assessment where relevant.

4.2.1 NRL Security Ontology

One interesting piece of research was conducted by Kim, Lou, and Kang from the United States Centre for High Assurance Computer Systems at the Naval Research Laboratory [41]. What resulted was a broad set of security ontologies for annotating resources, which the authors collectively refer to as the NRL Security Ontology [41]. Although the ontology does not cover vulnerability assessment or even risk analysis, the authors do point out an interesting benefit of using ontologies for security. The authors claim that the ontology is able to represent numerous security statements and can be applied to any electronic resource [41]. The authors also indicate that the developed ontology provides the ability for precisely describing security concepts at various levels of detail [41]. This example of using the same ontology to describe concepts of a varying nature is interesting, with regards to break out path vulnerability assessment, due to the varying nature of break out path vulnerabilities from system to system.

4.2.2 An Ontology for Network Security Attacks

An ontology for network security attacks is also proposed in the literature [49]. This ontology is built with the attacker in mind being represented by the Actor class. The ontology focuses on the relationships the attacker has with other parts of the domain, such as: Threat, Motive, Attack, Information, Outcome, and Intangible [49]. Although the authors present a table of vulnerability categories, the vulnerabilities or a class representing them is not found in the ontology itself. Even so, the vulnerability categories are of interest, as they give an overview of some of the different types of vulnerabilities that exist in systems. The categories are divided into two parts. The first is Security Policy vulnerabilities, which include social engineering (e.g. Information fishing, and Trojans) and policy oversight (e.g. poor planning, and poor control). The Second is Technology vulnerabilities which include Logic Errors (e.g. Bugs, OS vulnerabilities, etc) and Weakness (e.g. weak password system, old encryption standards). This is interesting, in regards to break out path vulnerabilities, as it provides a documented example of the division that exists between vulnerabilities that can and cannot be automatically discovered (recall that one unique characteristic of break out path vulnerabilities was that they were unable to be automatically detected by vulnerability scanners).

4.2.3 Security Ontology as a Methodical Tool

Raskin et al, in one of the earlier papers involving ontology and security, propose the use of ontology in information security as a useful theoretical foundation and methodical tool [60]. The authors indicate that the ultimate goal of their research is two fold. Firstly, to have natural language data sources as an integral part of the overall data sources used in information security. Secondly, to provide a formal specification of the know-how that exists in the information security community (i.e. capture, share, and reuse expert knowledge). Given these goals, the paper starts by trying to organize and unify the terminology used in the information security domain. At the time of printing, the research had only advanced as far as collecting terms to use in the theorised ontology, and in its current state only provided a taxonomy of security related terms. The authors do, however, point out some of the advantages of using ontologies for security. These include: organising and systematising domain concepts at different levels of detail, induced modularity (e.g. automatic relationships between an attack and a control that would prevent it), full combinational possibilities (e.g. possible attacks that have not occurred yet) [60].

The suggested use of ontologies for knowledge capture of security expertise, combined with the organisation and systematisation of the coinciding domain concepts, is again interesting with regards to break out path vulnerability assessment. The use of ontologies as a systematic, methodical tool relates directly to the first characteristic of the envisioned tailored vulnerability assessment technique for break out path vulnerability assessment, as detailed at the end of chapter 3.

4.2.4 Ontologies for Security Planning

In three related papers Ekelhart et al [23, 27] and Fenz and Weippl [28] employ the use of ontologies into security planning and quantitative risk analysis. The Fenz and Weippl paper proposes a security ontology that models the following concepts: attribute, threat, infrastructure, role, and person [28]. Although the concepts are similar to those used in vulnerability assessment, the scope of the paper appears to focus more on risk analysis of physical assets. The actual ontology itself is not provided in the paper, however the authors claim a proof of concept by providing a summary of how the ontology was applied to a simulated spreading fire situation (i.e. fire as a threat, asset damage by room, sprinkler control, etc) [28]. The two papers by Ekelhart et al [23, 27] are extremely similar in content and appear to build on the Fenz Weippl paper [28]. The authors propose a way of simulating threats to corporate assets and quickly calculating the cost of countermeasures. Again, the focus of these papers is on physical assets, and again the example of a simulated fire is given as a proof of concept. The authors provide a borrowed security and dependability taxonomy that defines six security attributes (an expansion of the CIA triad), six potential threats belonging to three different categories, and a number of controls or countermeasures referred to as ‘means’ [23, 27]. The ontology is provided graphically, divided across three diagrams, each providing a view of a different sub-tree of the ontology in what appears to be an instantiated form [23, 27]. Again, although these papers are focused more on calculating the cost of potential damage versus the cost of countermeasures, some interesting points emerge. In particular, is the use of the ontology to methodically simulate the progress of a fire from one room to the next, while also being used to keep track of the resulting damage. This is interesting with regards to break out path vulnerability assessment, as it shows the use of an ontology as a methodical tool for assessing the potential impact a certain threat has on assets.

4.2.5 Ontologies for Security Critical Software Development

Two papers by Karyda et al propose the use of ontologies to aide in the development of security critical applications [30, 38]. The authors state that the advantages of developing an ontology for this purposes is: to express the most important security concepts, realise the relationships among these security concepts, provide a common understanding and vocabulary of security issues among application developers, and facilitate the development of secure applications [30]. The ontology hierarchy is given, and consists of the following main classes: Assets, Countermeasures, Objective, Person, and Threat [38]. Each of these main classes, except Objective, each has a number of subclasses. The relationships between the significant classes are also given diagrammatically across four subsequent figures [30]. Although vulnerabilities are not explicitly included, it is feasible to assume that they would be included within the Threat subclasses (i.e. Errors and Technical Failure). The authors claim that the ontology itself is validated by querying the ontology [30]. The queries are meant to represent typical questions that a developer of a security critical application would have (e.g. what are the typical security objectives?). It appears however that the ontology first needs to be either tailored to a specific type of application (i.e. e-Tax, or e-Voting are examples used) either by extending the ontology, or by instantiating the ontology (it is not clear from the provided information which of these two methods were used).

Although not explicitly focused on vulnerability assessment, these papers offer some interesting insights into how ontologies can practically aide in information security. In particular, the example of being able to query the ontology, with questions like: which threats might compromise the anonymity of the voter?[38] This is a similar type of question that could be asked in vulnerability assessment (e.g. what vulnerabilities might compromise the confidentiality of a particular asset?). However from the papers it can be seen that the query will only return the correct answer if the necessary information and relationships are first hard coded into the ontology. This becomes a hindrance when considering break out path vulnerability assessment. This is because one of the major challenges with break out path vulnerabilities is that they are not yet known and therefore cannot be hard coded into an ontology. Nevertheless, the research gives an interesting example of how security information can be queried from an ontology.

4.2.6 Ontologies for Security Management

Finally, three related papers by Tsoumas and Gritzalis [65], Tsoumas, Dristas, and Gritzalis [31], and Tsoumas et al [66] make an attempt at using ontologies for security management. The authors claim that they have fully implemented a security ontology that relates to risk assessment [65]. They also state that their security ontology is a standards-based knowledge container, that has extended the CIM model with ontological semantics, and finally that it can also be used for reusable security knowledge interoperability, aggregation and reasoning, by using security knowledge from diverse sources [65, 66]. However, a closer examination of the security ontology offered by the authors reveals some obvious mistakes regarding ontology development.

Firstly, the Asset class and Countermeasure class are both classified as subclasses of the Risk Assessment class [65]. This seems counterintuitive, as it is difficult to see how an asset such as software, hardware, or information (all actual subclasses of the Asset class given in the ontology) could be considered a type of risk assessment. This is likewise the case with the Countermeasure class, for example, it is again difficult to see how a firewall could be considered a type of risk assessment.

The second problem involves how assets are related to given security attributes. Table 1 shows the six relationships that the authors claim exist between assets and different security attributes:

Asset Class	**Relationship**	**Security Attribute Class**
Asset	HasAuthenticity	Authenticity
Asset	HasAvailability	Availability
Asset	HasCompliance	Compliance
Asset	HasConfidentiality	Confidentiality
Asset	HasIntegrity	Integrity
Asset	HasNonRepudiation	NonRepudiation

Table 1. Asset to security attribute relationship table [65]

It seems redundant to have a unique relationship for each security attribute, when the security attributes are unique themselves. A more efficient way to model this would be to have a single relationship called HasSecurityAttribute that is used for every security attribute. X shows the resulting relationships if this approach were adopted.

Asset Class	**Relationship**	**Security Attribute Class**
Asset	HasSecurityAttribute	Authenticity
Asset	HasSecurityAttribute	Availability
Asset	HasSecurityAttribute	Compliance
Asset	HasSecurityAttribute	Confidentiality
Asset	HasSecurityAttribute	Integrity
Asset	HasSecurityAttribute	NonRepudiation

Table 2. Improved asset to security attribute relationship table

This reduces the number of relationships needed to one. Furthermore, it makes extending the ontology far easier. For example, if a new security attribute is required it can simply be added once into the security attribute class, without having to create a corresponding relationship.

The third and perhaps most significant error begins with the author's statement that the ontology development language OWL (Web Ontology Language) has limitations in expressing arrays [65]. The authors point out that a single asset can have many threats, and a single threat can itself have many countermeasures. This is true, however the authors go on to imply that this cannot be modelled in an ontology, specifically in using OWL and therefore implementing a multidimensional array is required [65, 66]. This claim is false and reveals a misunderstanding on the authors' part concerning ontology relationships. It appears the authors believe a unique relationship is required for each instance of a relationship type (this is also seen in the above noted redundant implementation of multiple security attribute relationships). This is incorrect. The ontology simply describes the types of relationships that exist between different classes. X shows a way of ontologically modelling what the authors claim cannot be modelled with OWL (bubbles represent classes, arrows represent relationships).

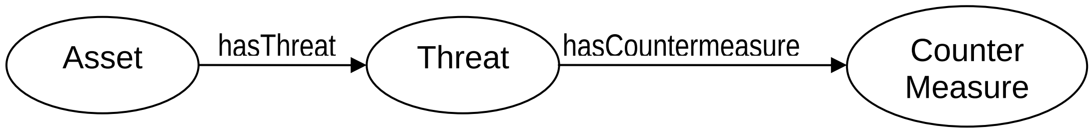


Figure 7. Asset, threat, countermeasure ontology model

This problem has adverse effects on the rest of the ontology. Having incorrectly modelled these relationships, the extendibility, reasoning, information sharing and reuse capabilities intended by ontologies and OWL are reduced, or even no longer possible, due to the suggested arrays not belonging to the OWL language. This is unfortunate for the research as it appears this mistake was made during the foundational stages and has eventually become a very large problem.

On the plus side, the authors do provide an interesting four phase process for using the ontology [31, 65, 66]. The first is to build the ontology (i.e. instantiate the ontology with a given systems information), the second to collect the security requirement of the system, the third phase involves matching the security requirement with specific controls, and the final phase involves deployment and monitoring of the secure system. This is interesting for break out path vulnerability assessment as it shows an ontology being used to facilitate the assessment of a given system. This is similar to the envisioned intention of an ontology for break out path vulnerability assessment.

4.2.7 Summary of Ontologies in Security

The literature involving the use of ontologies for information systems security, although not specifically focused on vulnerability assessment, has been useful via a number of semi-related illustrations. The literature has shown that there is potential for concepts such as vulnerabilities, to be modelled at different levels of abstraction using ontologies. It has also shown that vulnerabilities can be categorised in such a way that a division can be seen between those due to technology and those due to higher level concepts, such as social engineering or policy oversight. It has shown that concepts involved in the security domain can be organized by hierarchy, and that the relationships between those concepts can also be modelled. The literature has given an example of how an ontology can be used as a methodical tool. It has also given an example of the phases involved when instantiating a security ontology from an arbitrary system. It has provided examples of how ontologies can be queried to answer certain questions. Finally, the literature has also shown how making mistakes when developing ontologies can have a detrimental impact on the usefulness of the ontology.

4.3 Proposed Ontological Solution

The creation of an ontology for break out path vulnerability assessment seems fittings when the relevant information is considered collectively. Firstly, the theorised requirements of a useful break out path vulnerability assessment technique (i.e. systematic, system specific, standalone, and minimal user dependence). Secondly, the potentially useful features of ontologies (i.e. expert knowledge capture, knowledge sharing, training, minimal encoding bias, and extendibility). Finally, the evidence found in the literature of the application of ontologies to security problems.

Therefore the ontology for break out path vulnerability (or lockdown) assessment was developed with the following intended purpose:

- To capture the expert knowledge of attackers, that is exercised when breaking out of locked down computing environments
- Hold this captured knowledge in such a way that it can be shared amongst non-experts (knowledge sharing).
- Be used by non-experts as a tool for break out path vulnerability assessment
- Through its use, increase the expertise of the user with regards to break out path vulnerability assessment (training).
- Be able to be applied to different systems and tailored to different intended functions (standalone and system specific).
- Provide a systematic approach to break out path vulnerability assessment (systematic).

The next chapter is focused on the development of the break out path vulnerability (lockdown) assessment ontology.

5 Lockdown Assessment Ontology

In order to develop an ontology that captures the expert knowledge that attackers use to break out of locked down computing environments, it is worthwhile to consider the break out process. This chapter begins by looking at some examples of break out path vulnerabilities being exercised to achieve a broken state in locked down shared access computers. The chapter continues by extracting the underlying principles of the break out path examples. Next, based on these underlying principles, an ontology is developed and explained. Finally, the process of using the ontology for break out path vulnerability assessment is described.

5.1 Breaking Out

The process of breaking out of a locked down computing environment involves executing a series of state changing actions until a broken state is achieved. The broken state is dependant on the intended function and system requirements of each shared access computer (i.e. different shared access computers may vary in what functions they allow users to perform). Each example will begin with a basic description of the shared access computer and its intended use. This will be followed by a description of state changing actions that lead to a broken state. Each example is based on actual break out path vulnerabilities found in real systems, but have been anonymised for obvious ethical, privacy and security reasons.

5.1.1 Example One

The first break out path vulnerability to examine was found on a Windows based shared access computer that was set up with an automatic login. The desktop provided the user with four icons which each opened various web pages in a locked down version of Internet Explorer. The start menu was empty, apart from the same four icons and the shut down icon. The standard right click context menu was disabled for the desktop, start menu, taskbar, and their corresponding icons. The intended function of the computer was to provide instant anonymous user access to the four web resources accessible via the desktop or start menu icons.

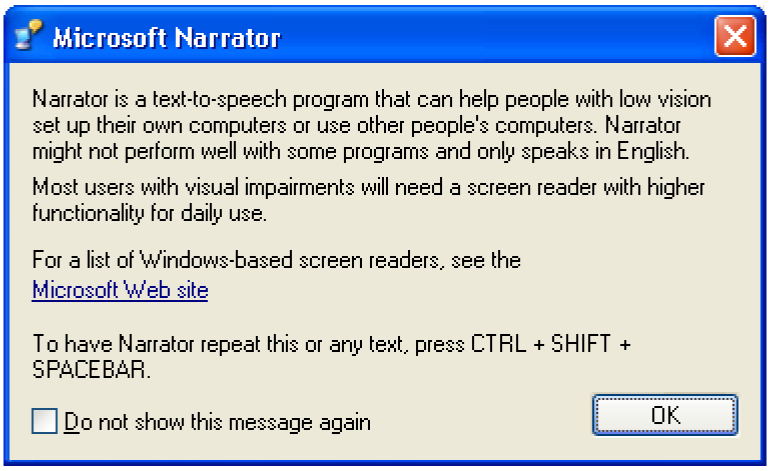
The broken state was achieved according to the following process. Firstly, one of the desktop icons was double clicked opening a web page in an internet explorer window. Next the web page was navigated to the ‘contact’ page of the website. This page included a number of mailto hyperlinks for staff email addresses. Next, a mailto hyperlink was clicked. This launched an outlook express email compose window. From this window, the attachment paperclip button was clicked. This opened a file browser window. From within this file browser window, the file system could be browsed. The right click context menu was also available on every icon in the file system from this view, which included options such as: explore, open, search, cut, copy, paste, and delete. Write access to parts of the file system was also available.

In this instance, the broken state was achieved by spawning the default mail client (outlook express). This in turn was used to obtain access to the file system via the email attachment feature. This provides an example of leveraging a legitimate intended function (i.e. the intended use of internet explorer), to perform an unintended function (i.e. open outlook express) which resulted in a broken state (i.e. access to the file system).

5.1.2 Example Two

The second break out path vulnerability to examine was also found on a Windows based shared access computer that was set up with an automatic login. In this instance, the start menu was empty apart from a shut down button and the start up folder which had a single Public Web Browser (PWB) icon. The PWB was automatically launched at start up and was set to be permanently maximised, without the option of closing the window. The web navigation was also restricted to a single web resource. The intended function of the computer was to provide anonymous user access to a single web resource via the PWB window.

The broken state was reached according to the following process. Firstly, the keyboard combination of ‘windows’ key and the ‘U’ key was pressed. This opened the Microsoft Utility Manager which had a list of three utilities: Magnifier, Narrator, and On-Screen Keyboard. Next, the Narrator utility was selected and the Start button was pressed. This opened the Narrator dialog box which gives an overview of the program and also a blue hyperlink to the Microsoft website (Figure 8 provides a screen capture of the dialog box).


Microsoft Narrator

Narrator is a text-to-speech program that can help people with low vision set up their own computers or use other people's computers. Narrator might not perform well with some programs and only speaks in English.

Most users with visual impairments will need a screen reader with higher functionality for daily use.

For a list of Windows-based screen readers, see the Microsoft Web site

To have Narrator repeat this or any text, press CTRL + SHIFT + SPACEBAR.

☐ Do not show this message again

Figure 8. Generic Microsoft Narrator dialog box

Next, the hyperlink was clicked, which opened the default Internet Explorer web browser. This web browser was not locked down and allowed unrestricted user access to the internet. This included the download, installation, and execution of applications.

The break out path provides two examples of leveraging dormant technology. Firstly, the keyboard shortcut opened the dormant Microsoft Utility Manager and the Microsoft Narrator. Secondly, the hyperlink opened the dormant Internet Explorer browser window. This resulted in a broken state, as the Internet Explorer browser was unrestricted in its capabilities.

5.1.3 Example Three

The third example of a break out path vulnerability was found on the same shared access computer as the second example. That is, the Windows based machine with the empty start menu (bar PWB icon and shut down button) and PWB window. Again, the intended function of the computer was to provide anonymous user access to a single web resource via the PWB window.

The broken state in this instance was reached in the same machine via a different process. Firstly, the CD drive open button was pressed, which unsurprisingly opened the CD drive. Next, a music CD was inserted and the CD drive was then closed. This caused an autorun window to appear on the screen with two possible options: Open folder to view files using Windows Explorer, or Take no action. The Open folder to view files using Windows Explorer option was then selected and the OK button clicked. This caused a Windows Explorer window to open which provided access to file system and network neighbourhood. From here, other applications including the command prompt, registry editor and internet explorer were able to be executed.

The broken state in this example was again reached by using an unintended, yet possible system function, associated with a hardware device. Specifically, the default autorun function associated with the CD drive was exercised. Interestingly, the computer did not even have speakers. Through the Windows Explorer option in the autorun menu, the broken state was achieved.

5.1.4 Example Four

The fourth break out path vulnerability to examine was found on a pay for use shared access computer. The computer was set up to allow users to perform basic desktop publishing tasks (Microsoft Office) and provide users internet access. The computer required payment via an attached eftpos device in order to use the computer. The screen consisted of a full screen display (i.e. no start button or taskbar, the operating system could not even be discerned from this state). This display gave directions on using the eftpos device. On paying, via the eftpos device, the full screen display would close and the user could use the computer.

The broken state was achieved starting from an unpaid full screen display according to the following process. Firstly, the keyboard combination of the ‘windows’ key and the

'D' key was pressed. This caused the full screen display to minimise into the taskbar. From here the desktop publishing tools could be opened and used, and the internet browser could be open and the internet accessed.

In this example of very simple break out path vulnerability, the payment system for a shared access computer was bypassed. This was achieved by leveraging a built in operating system command against the payment software. In this instance, an unintended possible function was used to bypass a security control to achieve a broken state.

5.2 *The Underlying Principles*

The above examples begin in various initial states and exercise different break out paths in order to reach various broken states. A break out path, in general, has been referred to as a series of state changing actions performed by the user. In order to help identify the underlying principles that can be extracted from the above four examples, it is useful to begin by looking at an information system from a very basic level. Figure 9 provides a graphical representation of the input processes output (IPO) model, which can be used to describe a very basic system.

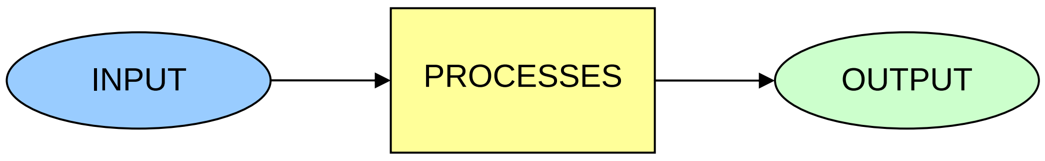


Figure 9. The basic IPO model

With regards to break out paths, it can be seen that the attacker is responsible for the inputs. These inputs cause the shared access computer to perform some internal processes which result in a state change, which is then output to the attacker via the screen. Based on this output, the attacker repeats the process (beginning with the next input) until the output produced by the system is a desired broken state. Therefore the first underlying principle that will be examined will involve the different possible inputs of a given system.

5.2.1 Inputs

In order to break out of a locked down computing environment, an attacker works on the computer via its inputs. The inputs that can be extracted from the break out path examples, can be seen to fit into three distinct categories: mouse inputs, keyboard inputs, and hardware inputs. The next step is to explore each of these categories to identify the potential inputs.

Mouse inputs can vary depending on the type of pointing device connected to the machine. A generic apple mouse for example only has one button and is therefore limited to single click combinations. A generic PC mouse usually has at least a left and a right mouse button which increases the number of possible click inputs. A common feature of a newer PC mouse is the presence of a scroll wheel. This scroll wheel also functions as a third button which can also be clicked and double clicked. A standard set of generic mouse inputs can be listed as:

- Primary Button Single Click
- Primary Button Double Click
- Secondary Button Single Click
- Secondary Button Double Click
- Scroll Wheel Single Click
- Scroll Wheel Double Click

Keyboard inputs are restricted to the keys available on a given keyboard and the combinations therein. For a standard QWERTY keyboard this includes the twenty six English alphabet letter keys, numeric keys, function keys (F1-F12), punctuation and bracket keys, formatting keys (enter, space, tab, etc), modifier keys (alt, ctrl, shift, etc), arrow keys, number pad keys, and the other miscellaneous keys (Print Screen, Scroll Lock, Pause/Break, Insert, Delete, Home, Page Up, etc). This set of potential inputs allows a significant number of combinations via the modifier keys. The key combinations that involve modifier keys usually represent keyboard shortcuts of application or operating system standard functions.

Hardware inputs are restricted by the variety of input devices and input ports that are present on a given computer. Common hardware inputs include CD/DVD drives, floppy disk drives, USB ports, and IEEE 394 Firewire ports. The ports in particular can potentially allow additional input devices to be connected (e.g. microphones, or even different pointing devices).

Interestingly, combinations of inputs from different categories can also cause different processes to occur on some applications. For example using the keyboard modifier Shift key and performing a primary button single click on a hyperlink in Internet Explorer will cause the link to open in a new window.

From an attacker's perspective, the members and combinations of these input categories represent the means by which they are able to attack the system. Combined with objects of focus (i.e. what they click on, or what is in focus when a key press is performed), they become the access points of the system.

Having explored the realm of possible system inputs, the next step is to consider the processes and outputs of the system. These two will be considered together.

5.2.2 Processes and Outputs

Generally speaking, the processes that occur within a given system are invisible to the user. The end result of an input made by a user, from the user's perspective, is the system output. However, depending on the state of a given system the same input by the user may cause a different process and consequently a different output. Therefore, an appreciation of the varying processes of a system depending on its state is useful. A simple way to illustrate this is through an example. Pressing Ctrl + S while an unsaved Microsoft Word document is open and in focus, will result in a process that outputs a save dialogue box to the screen. Pressing the same Ctrl + S key combination while a previously saved Microsoft Word document is open and in focus will result in a process that saves changes made to the document since the save and will produce an output of a progress bar at the bottom of the window for a short moment. Pressing yet again the same Ctrl + S key combination on a blank Windows desktop will result in a process that produces an unchanged output (i.e. it does nothing).

As mentioned the outputs of a given system are the result of inputs and internal processes. Based on the output of a given input, an attacker then proceeds with the next input and continues until a broken state is achieved. The impact of a given input has been shown to be dependant on the current state of the system. More accurately, for a given input its impact is dependant on object of focus of that input. These objects of focus can be generally referred to as the assets of the system. From a hardware perspective hardware assets also exist and are the objects of focus for hardware inputs (e.g. CD drive or USB port). From a non hardware perspective, the objects of focus fit into two main asset groups, software (i.e. applications) and information (i.e. data used and displayed by applications). The next section will show an ontology that represents the extracted underlying principles of the basic break out process.

5.3 The Ontology

Based on the above identified underlying principles of the break out process, the following ontology was developed. X gives a graphical representation of the ontology. The bubbles represent classes of the ontology, and the arrows represent relationships.

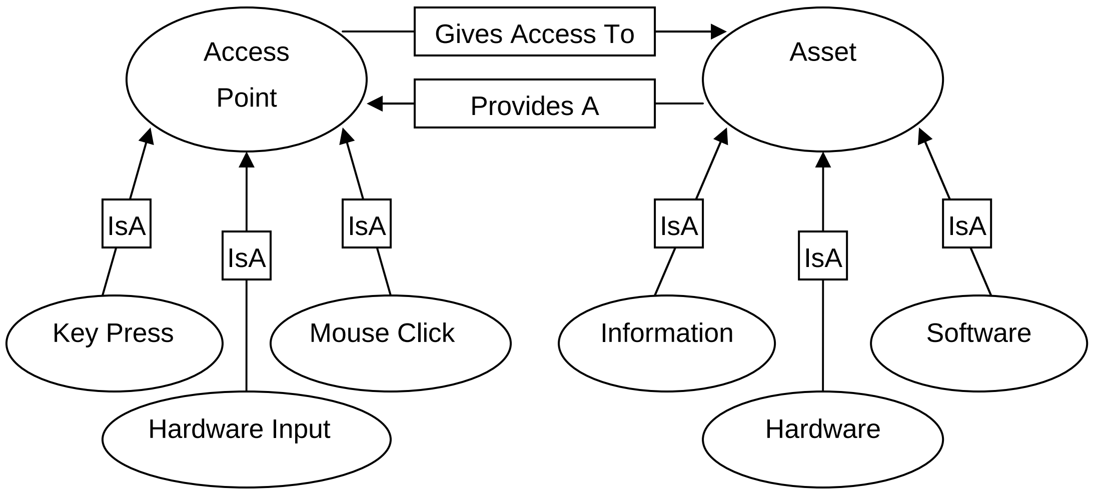


Figure 10. Ontology of the break out process

The ontology of the break out process has two main parts. The first is the *Access Point* class. This class represents the different system inputs and has three subclasses: Key Press, Mouse Click, and Hardware Input. These correspond to the three previously outlined categories of possible inputs. If additional inputs are encountered with a particular shared access computer system, the ontology can easily be extended by adding additional Access Point subclasses (e.g. voice input).

The second part is the *Asset* class. This class represents the different objects of focus that can be presented by the system. The Asset class also has three subclasses, which again correspond to the three previously identified asset categories: Information, Software, and Hardware.

The final aspect of the ontology that needs to be explained is the relationships that exist between assets and access points. The basic relationship is as follows: A given asset can potentially contain a number of access points (Provides A relationship). Conversely, a given access point will give access to usually one specific asset (Gives Access To relationship). An example of the *Asset Provides A Access Point* relationship can be seen in a basic Windows desktop. The Windows desktop is an Asset (Software subclass) and this Asset provides a number of access points. The Access Points could be Double Primary Click on the My Documents Icon, Single Secondary Click on the My Documents Icon, Enter Key Press on the My Documents Icon, or even Delete Key Press on ArbitraryFile.txt. Building from this example, corresponding examples of the *Access Point Gives Access To Asset* relationship can be shown. The Double Primary Click on the My Documents Icon would give access to the My Documents Folder in Windows Explorer Asset. The Single Secondary Click on the My Documents Icon would give access to the My Documents Folder Context Menu Asset. The Enter Key Press on the My Documents Icon would give access to the My Documents Folder in Windows Explorer. Finally the Delete Key Press of ArbitraryFile.txt would give access to ArbitraryFile.txt Confirm Delete Dialogue Box Asset. As the building up of relationships from Assets to Access Points and then back to Assets continues, a path is created. Figure 11 shows a graphical representation of an example path (note: white bubbles are Software Assets, grey bubbles are Mouse Click Access Points, black arrows are Provides A relationships, grey arrows are Gives Access To relationships, and PBSC stands for Primary Button Single Click).

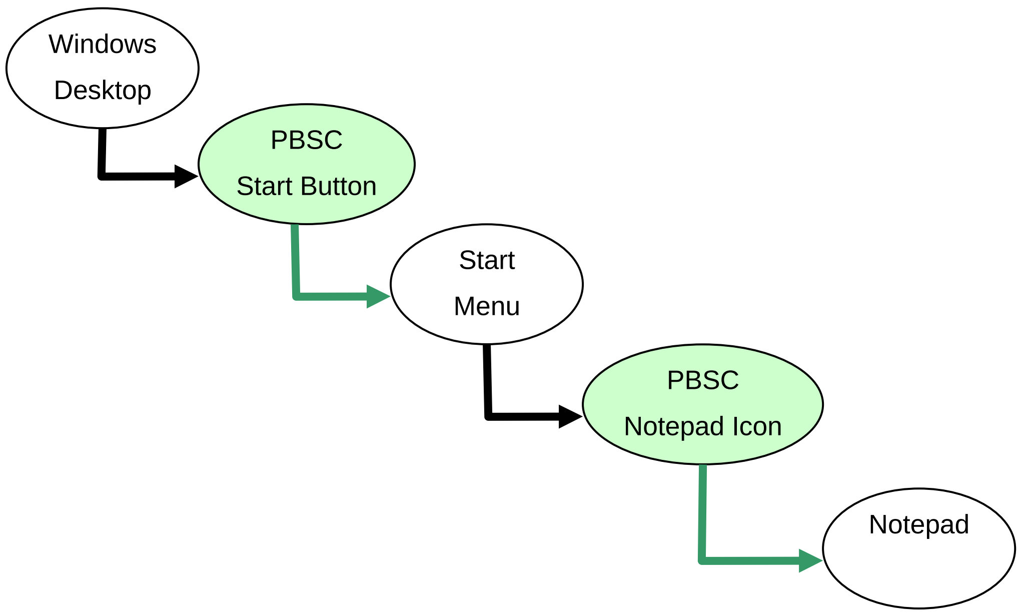


Figure 11. Ontological Asset to Access Point path

If the created path reaches an asset that corresponds to a broken state, the path becomes a break out path. What defines the broken state will depend on the intended function of the system being assessed. For example, the broken state may be defined by reaching an asset that can be used to interact with the file system (e.g. Windows Explorer, or the Command Prompt). A broken state could also be defined as reaching a particular asset via a path that does not involve another particular asset (e.g. reaching the desktop via a path that bypasses using the eftpos payment asset).

The next section will explain in more detail the process of using the ontology for break out path vulnerability assessment (i.e. ontological lockdown assessment).

5.4 Ontological Lockdown Assessment

The ontological lockdown assessment process can be separated into four phases. These phases provide a structured approach for tailoring the assessment process to a specific shared access computer system. The first phase is concerned with defining the broken state. The second phase involves identifying the initial assets of the system. The third phase requires the building Asset Access Point paths. The final phase involves the compilation and analysis of the discovered paths. The following section provides a more detailed description of each of these phases.

5.4.1 Phase 1: Define the Broken State

The purpose of this phase is to define the goal, or broken state of the assessment (i.e. if the broken state is reached the system is not correctly locked down).

This phase is only dependant on knowing the intended function of the shared access computer that is being assessed.

The broken state of a shared access computer will vary from system to system, depending on the intended function of the system. Given the intended function of the system, a perfectly locked down shared access computer would be one that has all other functions restricted. Therefore, a reasonable place to begin is by defining the broken state as any function that is not an intended function of the system. Or with regards to assets, the broken state can be defined as access to an asset which is not an asset that was intended to be accessed by the system.

This is a conservative definition, as some unintended yet possible functions may not be detrimental to the shared access computer. However, this definition of the broken state is quick and simple and has the advantage of covering all unintended functions, including all those that could be detrimental to the shared access computer. Decisions concerning unintended yet possible functions that are not detrimental to the system are made in phase four.

The output of this phase is a definition of the broken state which is unique to the shared access computer that is being assessed.

5.4.2 Phase 2: Identify the Initial Assets

Having defined the goal of the assessment in the first phase, the purpose of this second phase is to define that starting points of the assessment (i.e. in order to reach the broken state the assessment will begin from these starting points or initial assets).

This phase is only dependant on having physical access to the shared access computer that is being assessed.

The initial assets of shared access computers will also vary from system to system, depending on what they consist of, with regards to hardware, and how they are set up with regards to software.

A good practice is to begin by identifying the hardware assets through physical inspection of the system (this includes computer case, monitor, keyboard and other components). This will result in a list of devices, drives, and ports present on the physical system (USB ports for example can be found on monitors and keyboards as well as the traditional locations). The next step is to identify the software assets. A good approach is to start by turning on the shared access computer and wait for it finish loading. Once the system is loaded, begin listing what is on the screen. This will result in a list of software assets (i.e. start button, network connections system tray icon, my documents desktop icon, internet explorer web browser window). The operating system of the machine should also be included in this list, in order to cover operating system specific keyboard short cuts for example (e.g. ‘Windows’ key and ‘U’ key will open the Microsoft Utility Manager application of windows based machines).

The combined list of initial hardware and software Assets forms the starting points for the ontological lockdown assessment. It is from these initial assets that the broken state needs to be reached.

The output from this phase is a list of initial assets.

5.4.3 Phase 3: Build Access Paths

The purpose of this phase is to systematically step through the system building Asset Access Point paths, making note of paths that result in a broken state (i.e. break out paths).

This phase is dependant on having physical access to the shared access computer that is being assessed and also requires the outputs from the first two phases, specifically the definition of the broken state and the list of initial assets (i.e. the goal and the starting points).

This phase is where the main discovery of break out paths occurs and is performed according to the follow procedure:

- Beginning with the first initial Asset, start by building Asset Access Point paths according to the ontology. It is likely that starting from a single initial Asset, it will be possible to build a number of different paths (e.g. when a single Asset provides multiple Access Points).
- When a path reaches a broken state, it becomes a break out path and should be recorded. To record the break out path, write down the chain of Assets and Access Points used to reach the broken state.
- If a particular path does not reach a broken state, then that path does not need to be recorded.
- If, during the creation of a path an asset is reached that appears in a previously recorded break out path, the current path can be immediately recorded and joined to the existing break out path at that particular asset (this removes the redundancy of exploring the same path segments multiple times).
- Continue this process until all possible paths have been explored for every initial asset.

The output from this phase is either an empty set (for a suitably locked down system), or a set of break out paths.

5.4.4 Phase 4: Compilation and Analysis

The purpose of this phase is to organise and analyse the break out paths that have been identified in phase three.

This phase is dependant on successful completion of phase three and requires the set of break out paths and knowing the intended function of the shared access computer. It does not necessarily require physical access to the shared access computer being assessed.

The first step in this phase is to organise the identified break out paths. It can be useful to organise the interconnected break out paths in graph format. The next step is to examine each break out path, paying particular attention to the assets that have caused the broken state. Each break out path represents an unintended yet possible system function. At this point, these unintended yet possible functions should be considered with regards to the intended function of the shared access computer. If these unintended functions are deemed detrimental or unwanted, they can be marked as break out path vulnerabilities. There also remains the option of returning to the shared access computer to review particular break out paths, to determine if they are in fact detrimental or unwanted. Optionally, once the vulnerabilities have been identified, they can then be given an impact rating. This rating should correspond to a predefined scale (e.g. high, medium, low or 1-10, etc) and is selected at the discretion of the assessor. Having identified the break out path vulnerabilities, technically speaking, at this stage the vulnerability assessment process has been completed.

The output from this phase is a list of break out path vulnerabilities.

As a side note, once the break out path vulnerabilities have been marked the graph format of the recorded break out paths can be used as an aid when mitigating the vulnerabilities. This graph format can used to identify commonalities between break out paths. However, mitigation of vulnerabilities is a different step in the risk analysis process and is outside the scope of this research.

The next chapter involves a case study that provides a real world application of the aforementioned ontological locked assessment process.

6 Case Study

As a proof of concept, a case study has been performed in order to showcase the usefulness of the ontological lockdown assessment process. Due to privacy, security, and ethical reasons, the shared access computer that forms the focus of the case study has been anonymised as much as possible. The case study begins by introducing the characteristics of the shared access computer, its intended function and environment. Next, the case study will be presented according to the four phases of the assessment process. Following the ontological lockdown assessment process of the case study, some comparative results from other vulnerability sources will be proved. Finally, a conclusion of the case study will be given.

6.1 System Characteristics

The shared access computer that forms the focus of this study is a library catalogue computer. The catalogue computer is a Windows XP based machine and is set up with an automatic logon. In its default state, the computer presents the user with a locked down Public Web Browser window which displays the library catalogue home page. The Start menu only includes the programs, start up folder with the PWB icon, and the Turn off computer button. The system tray consists of the clock and three icons: audio, virus scanner, and security centre. The desktop is inaccessible due to the PWB window being permanently maximised.

Physically, the computer consists of screen, keyboard, mouse, and main case. The system sits on a standard desk with all parts of the computer physically accessible.

The intended function of the shared access computer is to provide anonymous user access to the library catalogue. As a second function, the web based catalogue system also allows known users to log in with a usernames and password, in order to view their lending status, renew and reserve books.

6.2 Ontological Lockdown Assessment

This section presents the four phases of ontological lockdown assessment process, as applied to the shared access computer with the aforementioned characteristics, intended function, and physical environment.

6.2.1 Phase 1: Define the Broken State

The broken state will be defined by taking the compliment of the intended function of the system. Firstly, the intended function of the system is to provide anonymous user access to the web based library catalogue via a Public Web Browser window and secondly to provide known users access to their lending status via the same Public Web Browser window. Therefore, the broken state is defined as any function that is not either browsing the library catalogue or viewing lending status via the Public Web Browser Window.

6.2.2 Phase 2: Identify the Initial Assets

The initial assets form the starting points of the access paths in phase three. Firstly, the initial hardware assets will be identified, and secondly the initial software assets.

The initial hardware assets that were identified by physical inspection were as follows:

- Screen control panel
- Keyboard port
- Mouse port
- Two USB ports
- CD Drive
- Floppy Disk Drive

The initial software assets that were identified by visual inspection were as follows:

- Start Button
- PWB window

- Symantic Antivirus system tray icon
- C-Media Mixer system tray icon
- Security Centre system tray icon
- System tray clock
- Taskbar
- Windows XP operating system

6.2.3 Phase 3: Build Access Paths

The access paths that were derived from the exploration of each of the initial assets, and that resulted in a broken state, will be presented next. An explanation of each broken state will also be given. The access paths consist of numbered steps of alternate assets and access points, where assets are odd numbered plain text and access points are even numbered underlined text. The following acronyms for mouse clicks are also used:

- PBSC: Primary Button Single Click
- PBDC: Primary Button Double Click
- SBSC: Secondary Button Single Click

Screen control panel access path 1:

1. Screen control panel
2. <u>Contrast down button press and hold</u>
3. Contrast level = 0 information box, black screen

Broken state: Results in a black, blank screen, rendering the computer unusable.

Screen control panel access path 2:

1. Screen control panel
2. <u>OK button press</u>
3. Main controls menu
4. <u>Down button press</u>
5. Zoom option highlighted
6. <u>OK button press</u>
7. Zoom menu

8. Left button press and hold
9. Zoom level = 0 information box, tiny image

Broken state: Results in a tiny image on the screen, reducing the usability of the computer.

Screen control panel access path 3:

(Path begins at step 7 of screen control panel access path 2)

7. Zoom menu
8. Right button press and hold
9. Zoom level = 100 information box, maximum zoomed in image

Broken state: Results in a maximum zoomed in image, reducing the usability of the computer by pushing much of the user interface out of view.

Screen control panel access path 4:

(Path begins at step 3 of screen control panel access path 2)

3. Main controls menu
4. Down button press x2
5. Adjust horizontal option highlighted
6. OK button press
7. Adjust horizontal menu
8. Left button press and hold
9. Horizontal position = 0 information box, image off centre to the left.

Broken state: Results in the screen image being pushed off the left hand side of the screen, reducing the usability of the system.

Screen control panel access path 5:

(Path begins at step 7 of screen control panel access path 4)

7. Adjust horizontal menu
8. Right button press and hold
9. Horizontal position = 100 information box, image off centre to the right

Broken state: Results in the screen image being pushed off the right hand side of the screen reducing the usability of the system.

Screen control panel access path 6:

(Path begins at step 7 of screen control panel access path 4)

7. Adjust horizontal menu
8. Down button press
9. Horizontal size option highlighted
10. Right button press and hold
11. Horizontal size = 100 information box, image expanded left and right

Broken state: Results in the screen being expanded off either side of the screen, reducing the usability of the system.

Screen control panel access path 7:

(Path begins at step 3 of screen control panel access path 2)

3. Main controls menu
4. Down button press x3
5. Adjust vertical option highlighted
6. OK button press
7. Adjust vertical menu
8. Left button press and hold
9. Vertical position = 0 information box, image moved off screen upwards

Broken state: Results in the screen image being pushed upwards out of the viewable screen area, reducing the usability of the system.

Screen control panel access path 8:

(Path begins at step 7 of screen control panel access path 7)

7. Adjust vertical menu
8. Right button press and hold
9. Vertical position = 100 information box, image moved off screen downwards

Broken state: Results in the screen image being pushed downwards out of the viewable screen area, reducing the usability of the system.

Screen control panel access path 9:

(Path begins at step 7 of screen control panel access path 7)

7. Adjust vertical menu
8. Down button press
9. Vertical size option highlighted
10. Right button press and hold
11. Vertical size = 100 information box, image expanded off screen vertically

Broken state: Results in the screen image being expanded vertically out of the viewable screen area, reducing the usability of the system.

Keyboard port access path 1:

1. Keyboard port
2. Unplug keyboard
3. Disconnected keyboard

Broken state: The keyboard can be disconnected, resulting in reduced usability availability of the system. This also allows the possibility of hardware key logger attacks.

Mouse port access path 2:

1. Mouse port
2. Unplug mouse
3. Disconnected mouse

Broken state: The mouse can be disconnected, resulting in reduced usability and availability of the system.

USB port access path 1:

1. USB port
2. Insert USB flash drive

3. Auto Run Window
4. Open folder to view files PBDC
5. Windows Explorer window
6. Local Disk (C:) PBSC
7. Windows Explorer view of C:\
8. Windows folder PBSC
9. Windows Explorer view of C:\windows\
10. System32 PBSC
11. Windows Explorer view of C:\windows\system32\
12. cmd.exe PBDC
13. Command Prompt window

Broken State: Results in the system opening a Windows Explorer window that can be used to navigate the file system. Furthermore, the command prompt can be opened.

Note: Steps 6 to 13 were included to test system permissions

USB port access path 2:

(Path begins at step 11 of USB port access path 1)

11. Windows Explorer view of C:\windows\system32\
12. regedt32.exe PBDC
13. Registry Editor window

Broken state: Results in the system opening a Windows Explorer window that can be used to navigate the file system. Furthermore the registry editor can be opened.

Note: Again, this path was included to text system permissions

USB port access path 3:

(Path begins at step 3 of USB port access path 1)

3. Auto Run Window
4. Print the pictures PBDC
5. Photo Printing Wizard window

Broken state: Results in the system opening the Photo Printing Wizard, an unintended function of the system.

USB port access path 4:

(Path begins at step 3 of USB port access path 1)

3. Auto Run Window
4. View a slideshow of the images PBDC
5. Windows Picture and Fax Viewer

Broken state: Results in the system opening the Windows Picture and Fax Viewer, an unintended function of the system.

USB port access path 5:

(Path begins at step 3 of USB port access path 1)

3. Auto Run Window
4. Copy pictures to a folder on my computer PBDC
5. Microsoft Scanner and Camera Wizard window

Broken state: Results in the system opening the Microsoft Scanner and Camera Wizard window, an unintended function of the system.

CD drive access path 1:

1. CD drive
2. Open Button press
3. Opened CD drive
4. Insert data CD, close button press
5. Auto Run Window

   (Continue from step 3 of USB port access paths 1, 2, 3, 4 and 5)

Broken State: Results in an Auto Run window that can be used to launch four unintended system applications, including Windows Explorer, which can be used to navigate and manipulate the file system and execute system applications. This is not an intended function of the system.

PWB window access path 1:

1. PWB window
2. Hyperlink SBSC
3. Hyperlink context menu
4. Save Target As option PBSC
5. Directory Browser window
6. Folder SBSC
7. Folder context menu
8. Explore option PBSC
9. Windows Explorer window

   (Continue from step 5 of USB port access path 1)

Broken state: Results in the system opening a Windows Explorer window, an unintended function of the system.

PWB window access path 2:

1. PWB window
2. Ctrl + S combination key press
3. Directory Browser window

   (Continue from step 5 of PWB window access path)

Broken state: Results in the system opening a Windows Explorer window, an unintended function of the system.

PWB window access path 3:

(Path begins at step 3 of PWB window access path 1)

3. Hyperlink context menu
4. Print option PBSC
5. Print window
6. Preferences button PBSC
7. Preferences window
8. Go online hyperlink PBSC
9. Internet Explorer window

Broken state: Results in the system opening an unrestricted Internet Explorer window which can be used to navigate the entire internet and also the local file system.

PWB window access path 4:

1. PWB window
2. Ctrl + P combination key press
3. Print window

   (Continue from step 5 of PWB window access path 3)

Broken state: Results in the system opening an unrestricted Internet Explorer window which can be used to navigate the entire internet and also the local file system.

PWB window access path 5:

1. PWB window
2. F9 key press
3. Print window

   (Continue from step 5 of PWB window access path 3)

Broken state: Results in the system opening an unrestricted Internet Explorer window which can be used to navigate the entire internet and also the local file system.

Symantic Antivirus system tray icon access path 1:

1. Symantic Antivirus system tray icon
2. Symantic Antivirus system tray icon SBSC
3. Symantic Antivirus context menu
4. Open Symantic Antivirus option PBSC
5. Symantic Antivirus window
6. Look for help button PBSC
7. Look for help contents page
8. Help System Contents link PBSC
9. Symantic Antivirus Help window
10. Print button PBSC
11. Print window

    (Continue from step 5 of PWB window access path 3)

Broken state: Results in the system opening an unrestricted Internet Explorer window which can be used to navigate the entire internet and also the local file system.

Security Centre system tray icon access path 1:

1. Security Centre system tray icon
2. Security Centre system tray icon SBSC
3. Security Centre context menu
4. Go to Microsoft Security Web Site option PBSC
5. Internet Explorer window

Broken state: Results in the system opening an unrestricted Internet Explorer window which can be used to navigate the entire internet and also the local file system.

Windows XP operating system access path 1:

1. Windows XP operating system
2. Windows key + U combination key press
3. Utility Manager window, Microsoft Narrator window
4. Microsoft Web Site hyperlink PBSC
5. Internet Explorer window

Broken state: Results in the system opening the Windows Utility Manager, an unintended function of the system. This can then be used to open an unrestricted Internet Explorer window which can be used to navigate the entire internet and also the local file system.

6.2.4 Phase 4: Compilation and Analysis

Having discovered numerous access paths that lead to broken states in phase three, the fourth phase aims to compile and analyse these access paths to determine if they are potentially detrimental to the system (i.e. break out path vulnerabilities). In order to aid in the analysis, the break out paths will be compiled into graph format. This provides a visual representation of the break out paths and their relationships to one another. Each node in the graph uses the following notation: Letter(s) Number decimal point Number (e.g. S1.2). Where the letters stand for the initial asset, the first number indicates which access path, and the final number identifies the step within the specified path.X shows the screen control panel access paths in graphical format (e.g. S1.2 translates as Screen Control Panel access path 1 step 2). White nodes represent assets and grey nodes represent access points. Figure 12 provides the graphical representation of the Screen Control Panel access paths.

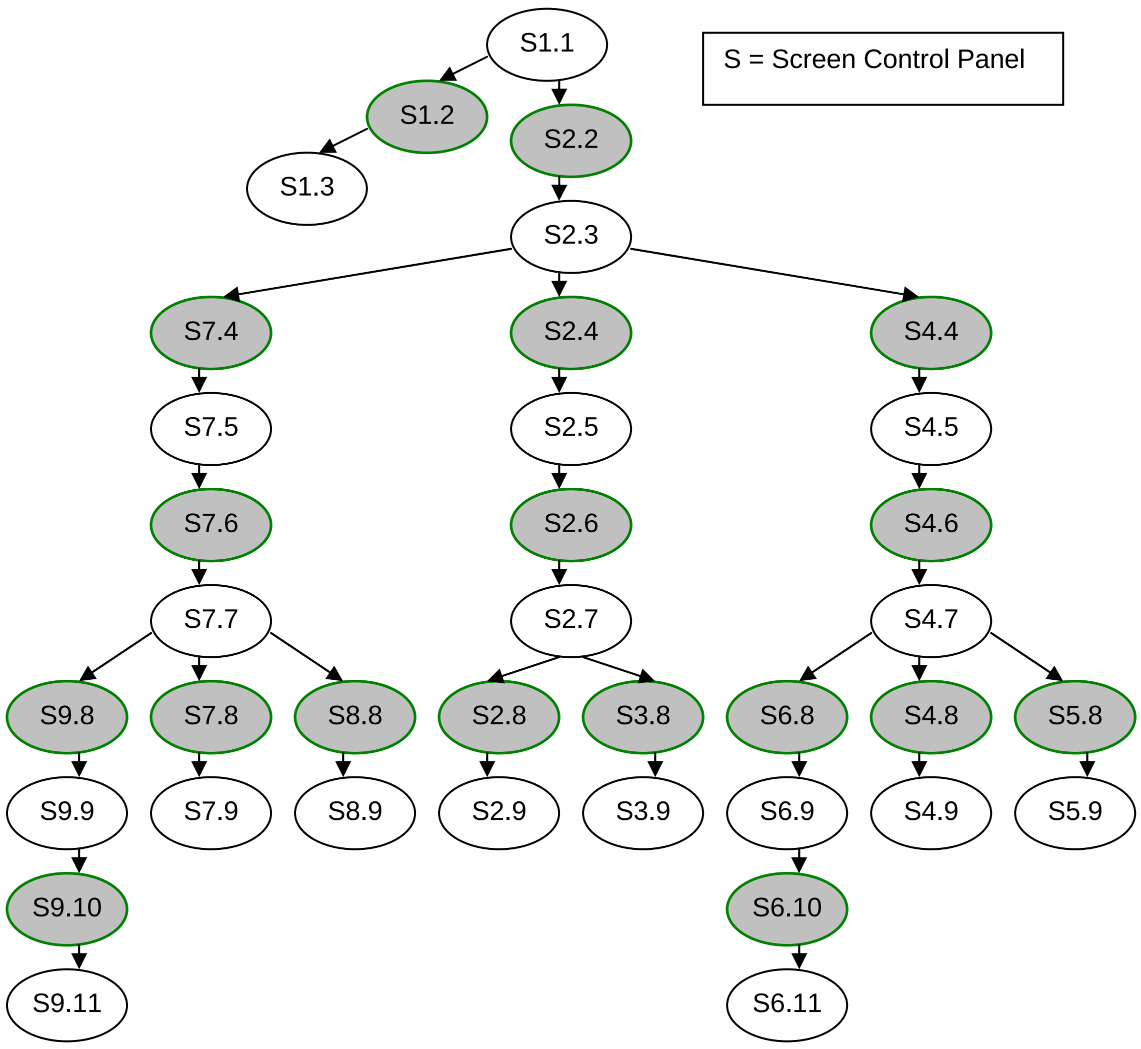


Figure 12. Screen Control Panel access paths in graph format

Figure 13 provides the interconnected graphical representation of the USB port, CD drive, PWB window, Symantic Antivirus, Security Centre, and Windows XP operating system access paths. Figure 14 provides the graphical representation of the Keyboard and Mouse access paths

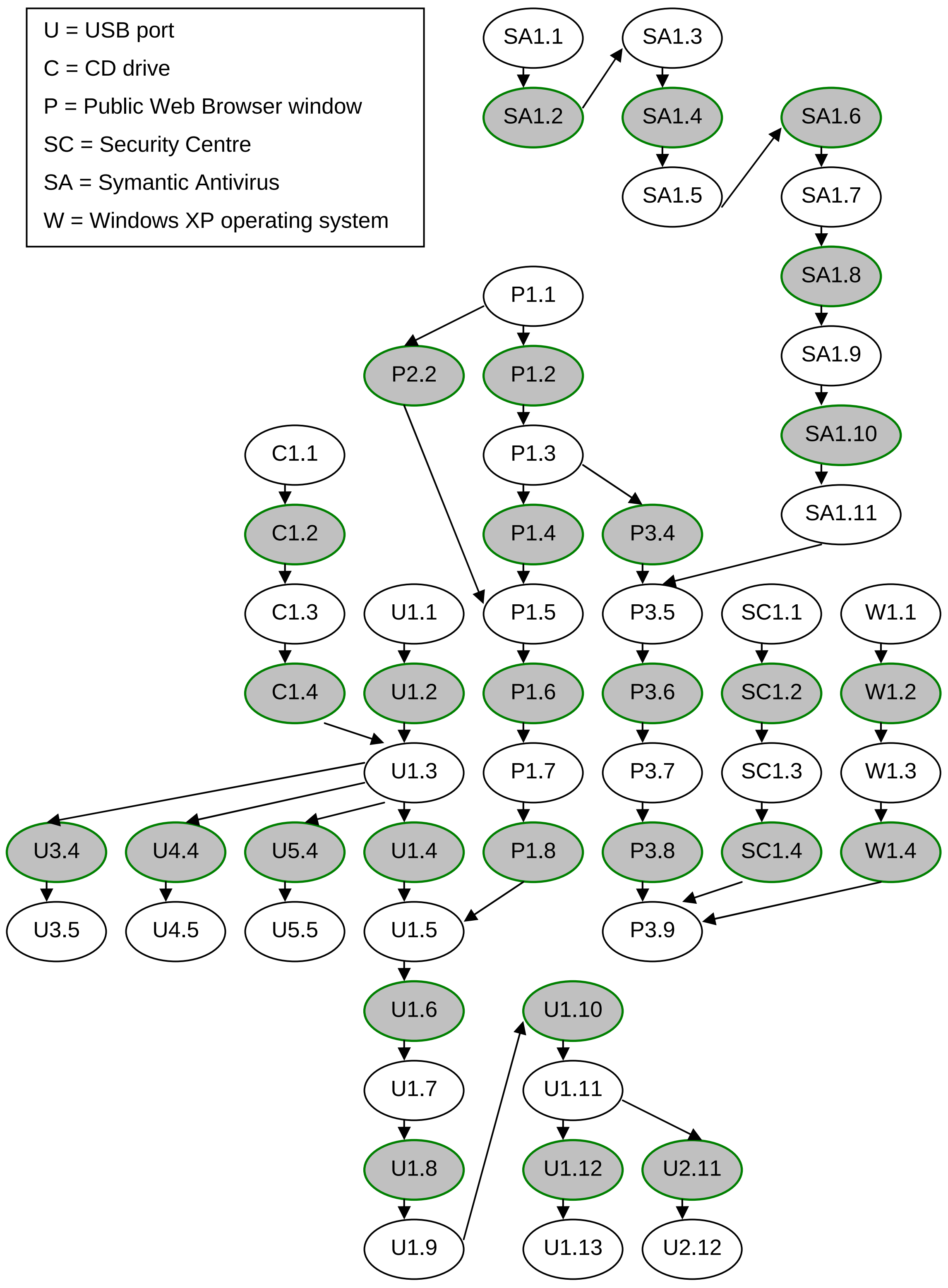


Figure 13. Interconnect access path graph

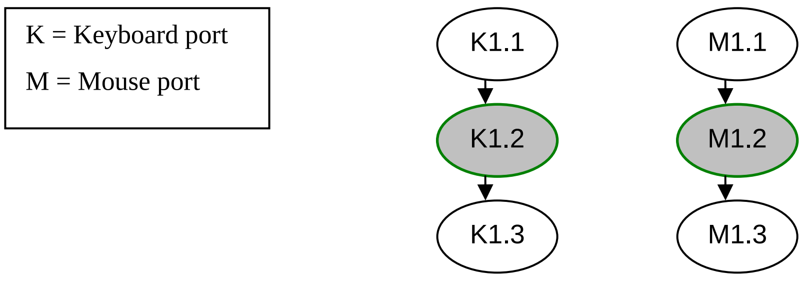


Figure 14. Keyboard port and Mouse port access paths in graph format

The determination as to whether a particular break out path is detrimental (i.e. a vulnerability) is at the discretion of the assessor. For each break out path in this case study, an analysis will be given to provide an explanation as to whether the break out paths have been deemed a vulnerability or not. An impact rating of between 1 and 10 has also been assigned to each break out path vulnerability, where 1 is the least impact and 10 is the most severe impact.

Screen Control Panel access paths

The first group access paths that resulted in a broken state all stemmed from the same initial asset, that is, the screen control panel. All of the eight broken states reached were concerned with usability, specifically altering the screen configuration to reduce the usability of the system.

The first access path that sets the contrast to zero, which results in a blank screen appears the most damaging, as it is not immediately obvious from a maintenance perspective what has occurred. The remaining seven access paths are less significant because, from a maintenance perspective, it is obvious the screen configurations have been altered.

The screen control panel break out paths fall into the category of computer vandalism vulnerabilities. Table 3 shows the impact ratings that were assigned to each of the eight vulnerabilities:

Access Path	**Impact Rating**
Screen Control Panel access path 1	5
Screen Control Panel access path 2	1
Screen Control Panel access path 3	1
Screen Control Panel access path 4	1
Screen Control Panel access path 5	1
Screen Control Panel access path 6	1
Screen Control Panel access path 7	1
Screen Control Panel access path 8	1

Table 3. Screen Control Panel vulnerability impact ratings

Keyboard and Mouse port access paths

The next two access paths that resulted in a broken state relate to the keyboard and mouse ports, respectively. Both vulnerabilities simply involved the removal of the keyboard and mouse plugs from their respective ports. The keyboard port vulnerability is of a more serious nature, as a hardware key logger can easily be installed between the keyboard plug and the keyboard port. This is a particular issue, due to the intended function of the computer, which includes the use of the system to log in and checking lending status. The mouse port vulnerability is less severe and, at most, reduces the usability of the system. Table 4 shows the impact ratings that were assigned to each of these vulnerabilities.

Access Path	**Impact Rating**
Keyboard port access path 1	7
Mouse port access path 1	1

Table 4. Keyboard and Mouse port vulnerability impact ratings

USB, CD, PWB, Windows, Security Centre and Symantic Antivirus access paths

The finally group of access paths to be analysed consists of twelve interconnecting paths, originating from six different initial assets.

The Windows XP operating system, Security Centre, Public Web Browser path 3, and the Symantic Antivirus access paths all reached the same broken state, that is, the opening of an unrestricted Internet Explorer window. Although the computer had a restricted version of the Public Web Browser installed, the default system web browser appeared to still be Internet Explorer, which was not restricted and allowed full access to the internet. This was not an intended function of the system and shows the danger of dormant technology. A secondary examination of the broken state revealed that the Internet Explorer window could be also be used to reach the local file system. Malicious software, such as software key loggers could easily be downloaded from the internet and installed on the local machine. The four break out path vulnerabilities were given high ratings, with the Windows XP operating system and Security Centre vulnerabilities receiving slightly higher ratings due to the shortness of their paths (i.e. the Internet Explorer window was launched in fewer steps).

The USB port, CD drive, and Public Web Browser access paths were all able to be used to reach a Windows Explorer window, thus creating a broken state. A deeper exploration into the file system revealed that powerful system tools, such as the command prompt and registry editor, were unrestricted. It was also found that write access was allowed in many parts of the file system including the Start up start menu folder. Providing this level of system access was not part of the intended function of the shared access computer.

Finally, the third, fourth, and fifth access paths of the USB port allowed the opening of three different built in system applications. This again was not part of the intended function of the system. A second examination of the fourth access path revealed that the Windows Picture and Fax Viewer could be used to reach the file system via the file save and open menus. The graph format also shows that the CD drive initial asset can also be used to reach these same broken states.

Through the analysis ten of the twelve access paths were found to provide access to the local file system.

Table 5 shows the impact ratings that were assigned to each of these twelve vulnerabilities.

Access Path	**Impact Rating**
USB port access path 1	8
USB port access path 2	8
USB port access path 3	5
USB port access path 4	7
USB port access path 5	5
CD drive access path 1	8
Public Web Browser access path 1	8
Public Web Browser access path 2	8
Public Web Browser access path 3	8
Symantic Antivirus access path 1	8
Security Centre access path 1	9
Windows XP operating system access path 1	9

Table 5. Interconnected break out path vulnerability impact ratings

The ontological lockdown assessment process has successfully been able to identify 22 break out path vulnerabilities, of varying degrees of impact and nature in a supposedly locked down shared access computers. The following section will show the comparative results of other vulnerability identification sources, with regards to the same system.

6.3 Comparative Results

This section will briefly examine the vulnerability assessment results for the same system, based on other vulnerability sources. The section aims to further emphasise the uniqueness of break out path vulnerabilities in shared access computers and to show the unsuitableness of traditional vulnerability assessment techniques.

6.3.1 Vulnerability Scanner

Despite the paradigm mismatch that has been identified between network vulnerability scanning and break out path vulnerabilities, the results of an automated vulnerability scanner have been included. Due to limitations of this research, not every commercially available vulnerability scanner could be tested against the target system. Fortunately though, the Nessus software package, which is regarded as the leader in automated vulnerability assessment, is free to use in non-commercial situations. Nessus scan results list open ports and categorise vulnerabilities according to three levels of severity: low, medium, and high. Table 6 shows the results of a full Nessus local scan (note: some fields have been removed for privacy, security, and ethical reasons, these field are indicated by the italic text *Removed*)..

127.0.0.1 (Localhost)		
Number of vulnerabilities	Open ports:	3
	Low:	11
	Medium:	0
	High:	0
Information:	Operating System:	Microsoft Windows XP
	NetBIOS name:	*Removed*
	DNS name:	localhost
Port general/tcp		
Host FQDN		

127.0.0.1 resolves as Localhost

OS Identification

```
Remote operating system : Microsoft Windows XP
Confidence Level : 99
Method : MSRPC

The remote host is running Microsoft Windows XP

Nessus ID : 11936
```

Information about the scan

```
Information about this scan :

Nessus version : 3.2.1.1
Plugin feed version : $Date: 2005/11/08 13:18:41 $
Type of plugin feed : CVS
Scanner IP : 127.0.0.1
Port scanner(s) : synscan
Port range : default
Thorough tests : no
Experimental tests : no
Paranoia level : 1
Report Verbosity : 1
Safe checks : yes
Optimize the test : yes
Max hosts : 20
Max checks : 5
Recv timeout : 5
Scan Start Date : 2008/8/20 9:10
Scan duration : 199 sec


Nessus ID : 19506
```

Port microsoft-ds (445/tcp)

SMB Detection

```
A CIFS server is running on this port

Nessus ID : 11011
```

Using NetBIOS to retrieve information from a Windows host

Synopsis :

It is possible to obtain the network name of the remote host.

Description :

The remote host listens on tcp port 445 and replies to SMB requests.
By sending an NTLMSSP authentication request it is possible to obtain
the name of the remote system and the name of its domain.

Risk factor :

None

Plugin output :

The following 2 NetBIOS names have been gathered :

Removed = Computer name
Removed = Workgroup / Domain name

CVE : CVE-1999-0621
Other references : OSVDB:13577

Nessus ID : 10150

SMB NativeLanMan

Synopsis :

It is possible to obtain information about the remote operating
system.

Description :

It is possible to get the remote operating system name and
version (Windows and/or Samba) by sending an authentication
request to port 139 or 445.

Risk factor :

None

Plugin output :

The remote Operating System is : Windows 5.1
The remote native lan manager is : Windows 2000 LAN Manager
The remote SMB Domain Name is : *Removed*

Nessus ID : 10785

SMB log in

Synopsis :

It is possible to log into the remote host.

Description :

The remote host is running one of the Microsoft Windows operating systems. It was possible to log into it using one of the following account :

- NULL session
- Guest account
- Given Credentials

See also :

http://support.microsoft.com/support/kb/articles/Q143/4/74.ASP
http://support.microsoft.com/support/kb/articles/Q246/2/61.ASP

Risk factor :

none

Plugin output :

- NULL sessions are enabled on the remote host

CVE : CVE-1999-0504, CVE-1999-0505, CVE-1999-0506, CVE-2000-0222, CVE-2002-1117, CVE-2005-3595
BID : 494, 990, 11199

Nessus ID : 10394

SMB registry can not be accessed by the scanner

Synopsis :

Nessus is not able to access the remote Windows Registry.

Description :

It was not possible to connect to PIPE\winreg on the remote host.

If you intend to use Nessus to perform registry-based checks, the
registry checks will not work because the 'Remote Registry Access'
service (winreg) has been disabled on the remote host or can not be
connected to with the supplied credentials.

Risk factor :

None

Nessus ID : 26917

SMB NULL session

Synopsis :

It is possible to log into the remote host.

Description :

The remote host is running one of the Microsoft Windows operating
systems. It was possible to log into it using a NULL session.

A NULL session (no login/password) allows to get information about
the remote host.

See also :

http://support.microsoft.com/support/kb/articles/Q143/4/74.ASP
http://support.microsoft.com/support/kb/articles/Q246/2/61.ASP

Risk factor :

None
CVE : CVE-2002-1117
BID : 494

Nessus ID : 26920

Port epmap (135/tcp)

MSRPC Service Detection

Synopsis :

A DCE/RPC server is listening on the remote host.

Description :

The remote host is running a Windows RPC service. This service
replies to the RPC Bind Request with a Bind Ack response.

However it is not possible to determine the uuid of this service.

Risk factor :

None

Nessus ID : 22319

Port ntp (123/udp)

NTP read variables

Synopsis :

An NTP server is listening on the remote host.

Description :

An NTP (Network Time Protocol) server is listening on this port.
It provides information about the current date and time of the
remote system and may provide system information.

Risk factor :
None
Nessus ID : 10884

Table 6. Comparative Nessus vulnerability scan results

These results clearly show the unsuitableness for using Nessus or any other automated network vulnerability scanner for the identification break out path vulnerabilities in shared access computers. Firstly, the vulnerabilities identified by the scanner all had a risk factor of ‘None’. Secondly, none of the identified vulnerabilities were break out path vulnerabilities. These results further emphasise the uniqueness of break out path vulnerabilities and particularly their difference from network vulnerabilities.

6.3.2 Online Vulnerability Databases

Online vulnerability databases are designed to be used by security experts to keep up to date with the latest vulnerabilities. These databases are also a common reference point used by automated scanning tools. Interestingly, they are a vulnerability source recommended for review by the more general vulnerability assessment procedures, and are indicated as the basis of vulnerability expert knowledge [55]. This is significant, as it was found that the more general vulnerability assessment procedures relied heavily on the expertise of the assessor. For these reasons, it is worthwhile using the contents of these extremely comprehensive online vulnerability databases, in an attempt to identify break out path vulnerabilities in the case study shared access computer.

The online vulnerability databases that were consulted were: National Vulnerability Database (NVD) [54] which includes the CVE database, US-CERT databases, and OVAL database.

The National Vulnerability Database links to five distinct resources that can be utilised for vulnerability assessment. As of August 2008, this included 32358 CVE (Common

Vulnerabilities and Exposures) Vulnerabilities [52], 161 Checklists [54], 147 US-CERT (United States Computer Emergency Readiness Team) Alerts [67], 2238 US-CERT Vulnerability Notes [68], and 3258 OVAL (Open Vulnerability and Assessment Language) queries [53]. Interestingly, the NVD database is maintained by the Nation Institute of Standards and Technology (NIST), which were also responsible for the creation of the NIST Guide to Risk Management, as reviewed in section 3.2.1.

The 32358 CVE vulnerabilities are divided into two categories; entries and candidates. The entries are acknowledged vulnerabilities and the candidates are potential vulnerabilities that are yet to be reviewed (i.e. check they exist and have not already been reported etc). The vulnerabilities listed are all application or component specific. Due to this fact, and the unique nature of break out path vulnerabilities, these vulnerabilities do not address break out path vulnerabilities as found in shared access computers.

The 161 checklists were all focused on the configuration aspects of specific software (65 checklists) packages and operating systems (96 checklists). Not one of the checklists took into consideration the unique security requirements of shared access computers, or break out path vulnerabilities.

The 147 US-CERT alerts are announcements concerning updates released by specific vendors, where detail is provided concerning the updates and the addressed vulnerabilities. Each alert details the software packages involved and specific vulnerabilities that have been addressed. Being application specific, none of these alerts were concerned with break out path vulnerabilities in shared access computers.

The 2238 US-CERT vulnerability notes were all application specific vulnerabilities. Each vulnerability note detailed an error in a particular application, for example, Adobe Flash Player long string buffer overflow. Again, being application specific, none of the vulnerability notes were concerned with break out path vulnerabilities in shared access computers.

The 3258 OVAL queries are similar to Nessus plugins and are designed to be used to automatically test for a specific vulnerability. As with the Nessus plugins the OVAL queries are tailored to identify already documented application specific vulnerabilities. Yet again, being application specific, none of the OVAL queries address break out path vulnerabilities in shared access computers.

The underlying problem with using these online vulnerability databases for break out path vulnerability assessment is essentially the same problem that was encountered with the automated scanner. That is, that the vulnerabilities are application or component specific. This has resulted in a null result in identifying break out path vulnerabilities in the target case study system.

6.4 Case Study Summary

The ontological lockdown assessment process managed to identify 22 break out path vulnerabilities in the target shared access computer system. Nessus, the industry leading vulnerability scanner reported 11 zero risk factor vulnerabilities which, upon examination, were not detrimental vulnerabilities. Furthermore, none of the vulnerabilities identified by Nessus were break out path vulnerabilities, nor were they related to break out path vulnerabilities. Finally, the online vulnerability databases due to the application and component specific nature also failed to identify any break out path vulnerabilities in the case study system.

7 Conclusion

This chapter begins by reviewing the research and showing that each research objective was successfully fulfilled. Following from this, potential future work associated with this research is given. Next, the limitations experienced in this research are outlined. Finally, the thesis ends with a brief discussion, concerning the path that led to the developed process

7.1 Review

In the introduction, four research objectives were identified They were as follows:

1. Highlight the unique security issues that are encountered when dealing with shared access computers, in particular the lockdown problem.
2. Reveal that existing vulnerability assessment tools are insufficient to assess whether shared access computers are adequately locked down.
3. Show that by developing an ontology for lockdown assessment, a systematic approach for assessing shared access computers can be achieved.
4. Prove the usefulness of the developed approach with a case study

The first research objective of highlighting the unique security issues of shared access computers was fulfilled in the second chapter. The unique issues included the under appreciation of shared access computers, the low security emphasis, the significant threat to the internal network, the threat to user information, and issues that can arise from dormant technology. To further emphasise these issues, potential attacks on shared access computers were explored. Finally, it was shown that locking down shared access computers was the first and most significant line of defence. However, the problem of assessing a locked down computer for break out path vulnerabilities was also apparent.

In the third chapter, the existing tools and techniques for vulnerability assessment were shown to be insufficient for identifying break out path vulnerabilities. This fulfilled the second research objective. The existing methods for vulnerability assessment were all shown to suffer from essentially the same problem of being focused on vulnerabilities that were application specific. Also, another significant issue identified with the more general vulnerability assessment procedures, was their dependence on the expertise of the assessor. With break out path vulnerabilities consisting of a series of actions performed by an attacker, it became apparent that these existing techniques were unsuitable for break out path vulnerability assessment.

The fourth and fifth chapters presented an ontological solution to the lockdown assessment problem. Chapter four justified the selection of ontologies based on their unique characteristics and existing applications in the field of information system security. The chapter finished with specifications for an ontology for break out path vulnerability assessment. Chapter five presented the development and documentation of the ontological lockdown assessment process. The chapter ended with a four phase description on how to use the process to identify break out path vulnerabilities. The combination of these two chapters fulfilled the third research objective, of showing that an ontology could be developed to provide a systematic approach to break out path vulnerability assessment.

The fourth research objective of proving the usefulness of the developed approach with a case study was fulfilled in the sixth chapter. The case study chapter presented the ontological lockdown assessment process being applied phase by phase to a real world shared access computer. The ontological lockdown assessment process was used to systematically find a significant number of break out path vulnerabilities in the case study system. The same system was assessed for break out path vulnerabilities using an automated scanner and by examining online vulnerability databases. Both of these methods produced null results with regards to break out path vulnerabilities.

7.2 Future

The ontological lockdown assessment process is essentially the first of its kind. For this reason improvement, additions and modifications are expected.

One foreseeable development would incorporate the association of security attributes such as confidentiality, integrity, availability to particular break out paths or perhaps even particular assets or access points.

Development of a metric for the impact rating of each break out path vulnerability that takes into consideration the length of the break out path could also be explored.

Methods for the remediation of the identified break out path vulnerabilities could also be explored.

7.3 Limitations

Two main limitations to this research were encountered during the comparative results section. The first was due to the funding limitations which restricted the ability for the research to comprehensively test numerous commercial vulnerability scanners. With scanners costing hundreds, and even thousands of dollars, the purchase of each of these tools was infeasible. However, as noted earlier, fortunately the industry leading vulnerability scanner Nessus is free for use in non-commercial situations. Nessus therefore provided an adequate example of the type of results that could be expected from an automated scanner. Due to the identified paradigm mismatch between network vulnerabilities and break out path vulnerabilities, this inability to test every automated scanner is not seen as significant with regards to the outcome of this research.

The second limitation was concerned with the more general vulnerability assessment procedures, which were shown to rely heavily on the ability of the assessors. The inability to assess the results of these more general procedures, independent of the ability of the assessor, limited the degree to which the comparative results supported the usefulness of the ontological lockdown assessment process. In order to compensate, a review of online vulnerability databases was conducted, in order to ascertain if any break out path vulnerabilities could be identified by using these resources. The motivation in doing this was based on the premise that vulnerability assessment expert knowledge comes from having an extensive knowledge of known vulnerabilities. If the entire online vulnerability databases were unable to identify any break out path vulnerabilities, then it could be reasonably concluded that the average assessor would have at best the same result. However, it is appreciated that some assessors, due to experience and intuition, may produce better results than those obtained by referencing online vulnerability databases. This factor again limited the degree to which the comparative results were able to support the usefulness of the ontological lockdown assessment process.

The third and final limitation that was noted, was the lack of existing research in this field. With break out path vulnerabilities in shared access computers being a very specific area of focus, the amount of corresponding literature was accordingly quite limited. Also, due to the commercial nature of vulnerability assessment, a number of industry sources were referenced and examined throughout the course of the research. Due to the lack of peer-review and sometimes biased nature of industry produced literature, its quality and relevance can be debated. Due to these two factors, the research was limited in its ability to reference perfectly sound literature. However, in order to compensate for this, every effort was employed in order to ensure that the industry documentation that was being referenced was well regarded and accepted within the security community.

7.4 Discussion

As far as is known, the developed approach of ontological lockdown assessment appears to be the first and only vulnerability assessment technique focused specifically on identifying break out path vulnerabilities in shared access computers.

By the final stages of this research, it became apparent that the third phase in the ontological lockdown assessment process was very clearly recursive in nature. As the core phase of break out path vulnerability identification, it can be seen as the most important phase of the process. In hindsight, it would have been interesting to have explored process modelling techniques for this aspect of the developed approach.

Initially, due to the ontological backbone of developed approach, formally developing the ontology in the Protégé ontology development environment according to the Web

Ontology Language (OWL) was performed. In the beginning stages of the research, it was envisioned that the ontology would exist as a static resource, and that the assessment process would require each break out path found on a given system to be formally instantiated in Protégé (or other ontology development environment) according to the OWL ontology. This would have allowed automated reasoning about the break out path vulnerabilities. However, it was found very quickly that this formal instantiation incurred a very large and time consuming overhead. Also, the benefits of formally instantiating each break out path in Protégé were minimal and in some cases non-existent. Furthermore, due to limitations in the description logics field regarding the ability to perform transitive queries, the usefulness of having formally instantiated interconnected break out paths was reduced even more. It was noted during experimentation that the main value and most useful part of the process occurred during the discovery phase, where the literal exploration of the system was performed according to the underlying principles documented in the ontology. For this reason, the formal instantiation of the break out paths was abandoned in favour of a more practical approach, that is, the developed four phase ontological lockdown assessment process.

References

1. *Application Access Control*. HorizonDataSys.com (2008) Retrieved on: August 5, 2008. Available from: http://www.horizondatasys.com/304850.ihtml.

2. *CORE IMPACT Pro Overview*. CORE Security Technologies, coresecurity.com (2008) Retrieved on: August 7, 2008. Available from: http://www.coresecurity.com/content/core-impact-overview.

3. *Faronics WINSelect: DYNAMIC Preference Control*. Faronics.com (2008) Retrieved on: August 5, 2008. Available from: http://www.faronics.com/html/Winselect.asp.

4. *Foreign National Sentenced to Nince Years in Prison for Hotel Business Center Computer Fraud Scheme*. Department of Justice (2008) Retrieved on: August 5, 2008. Available from: http://www.justice.gov/criminal/cybercrime/bonillaSent.pdf.

5. *Fortres 101: Proven Desktop Security Software*. Fortresgrand.com (2008) Retrieved on: August 6, 2008. Available from: http://www.fortresgrand.com/products/f101/f101.htm.

6. *GFI LANguard Netowrk Security Scanner: Overview*. gfi.com (2008) Retrieved on: August 7, 2008. Available from: http://www.gfi.com/lannetscan/.

7. *GFI LANguard Network Security Scanner: Features*. gfi.com (2008) Retrieved on: August 7, 2008. Available from: http://www.gfi.com/lannetscan/lanscanfeatures.htm.

8. *Information on Security Compromise in Computer Labs* VCU Technology Services (2006) Retrieved on: August 5, 2008. Available from: http://www.ts.vcu.edu/security/lab_compromise.html.

9. *Keystroke-logging software -- secret threat.* Computer Fraud & Security, 2003. 2003(3): p. 2-2.

10. *MBSA: Microsoft Baseline Security Analyzer*. MicrosoftTechNet (2008) Retrieved on: August 7, 2008. Available from: http://technet.microsoft.com/en-nz/security/cc184924(en-us).aspx.

11. *Nessus: Plugins*. Tenable Network Security (2008) Retrieved on: August 7, 2008. Available from: http://www.nessus.org/plugins/.

12. *Nessus: the Network Vulnerability Scanner*. Tenable Network Security (2008) Retrieved on: August 7, 2008. Available from: http://www.nessus.org/nessus/.

13. *Public PC Desktop: Turn your PC into a public access workstation.* (2008) Retrieved on: August 5, 2008. Available from: http://www.softheap.com/pubpcd.html.

14. *Retina Network Security Scanner*. eEye Digital Security, eyee.com (2008) Retrieved on: August 7, 2008. Available from: http://www.eeye.com/html/Products/Retina/index.html.

15. *SAINT: Security Administrator's Integrated Network Tool*. SAINT Vulnerability Scanner (2008) Retrieved on: August 7, 2008. Available from: http://www.saintcorporation.com/products/vulnerability_scan/saint/saint_scanner.html.

16. *SARA: Security Auditor's Research Assistant*. The Advanced Research Corporation, www-arc.com (2008) Retrieved on: August 7, 2008. Available from: http://www-arc.com/sara/.

17. *SiteKiosk*. SiteKiosk.com (2008) Retrieved on: August 5, 2008. Available from: http://www.sitekiosk.com/en-US/SiteKiosk/Default.aspx.

18. *WinLock*. CrystalOffice.com (2008) Retrieved on: August 6, 2008. Available from: http://crystaloffice.com/winlock/.

19. *WinLock Professional*. CrystalOffice.com (2008) Retrieved on: August 6, 2008. Available from: http://crystaloffice.com/winlockpro/.

20. Anton, P.S., et al. *Finding and Fixing Vulnerabilities in Information Systems*. The Vulnerability Assessment and Mitigation Methodology. RAND, (2003) Retrieved on: August 8, 2008. Available from: http://www.rand.org/pubs/monograph_reports/2005/MR1601.pdf.

21. Anusha, I. and Hung, Q.N., *Towards a Theory of Insider Threat Assessment*, in Proceedings of the 2005 International Conference on Dependable Systems and Networks. 2005, IEEE Computer Society.

22. Arno, K. *How MySQL Treats Security Vulnerabilities*. MySQL, (2007) Retrieved on: August 7, 2008. Available from: http://dev.mysql.com/tech-resources/articles/security_vulnerabilities.html.

23. Bagchi, A. and Atluri, V., eds. *Security Ontology: Simulating Threats to Corporate Assets*. ICISS 2006, LNCS 4332. 2006, Springer Verlag Berlin Heidelberg. 249-259.

24. Brackney, D., *The Cyber Enemy Within ... Countering the Threat from Malicious Insiders*, in Proceedings of the 20th Annual Computer Security Applications Conference. 2004, IEEE Computer Society.

25. Chahrvin, S., *Keyloggers - your security nightmare?* Computer Fraud & Security, 2007. 2007(7): p. 10-11.

26. Cranor, L.F. and Garkinkel, S., eds. *Security and Usability: Designing Secure Systems That People Can Use*. 2005, O'Reilly.

27. Ekelhart, A., et al., *Security Ontologies: Improving Quantitative Risk Analysis.* System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on, 2007: p. 156a-156a.

28. Fenz, S. and Weippl, E., *Ontology based IT-security planning.* Dependable Computing, 2006. PRDC '06. 12th Pacific Rim International Symposium on, 2006: p. 389-390.

29. Frank, L.G., et al., *Combating the Insider Cyber Threat.* IEEE Security and Privacy, 2008. 6(1): p. 61-64.

30. Funabashi, M. and Grzech, A., eds. *Employing Ontologies for the Development of Security Critical Applications* IFIP 2005, I3E 2005. Vol. 189. 2005, Springer-Verlag Berlin Heidelberg.

31. Gorodetsky, V., Kotenko, I., and Skormin, V., eds. *An Ontology-Based Approach to Information Systems Security Management*. MMM-ACNS 2005, LNCS 3685. 2005, Springer-Verlag Berlin Heidelberg.

32. Gruber, T.R., *Toward principles for the design of ontologies used for knowledge sharing.* International Journal of Human-Computer Studies, 1995. 43(5-6): p. 907-928.

33. Guarino, N., *Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy*. 1998: IOS Press. 337.

34. Hadad, H. and Gaffney, M. *Queens man sentenced to 27 months' imprisonment on federal charges of computer damage, access device fraud and software piracy*. (2005) Retrieved on: August 7, 2008. Available from: http://www.usdoj.gov/usao/nys/pressreleases/February05/jiangsentencingpr.pdf.

35. Heflin, J. *OWL Web Ontology Language Use Cases and Requirements*. W3C (2004) Retrieved on: August 11, 2008. Available from: http://www.w3.org/TR/webont-req/.

36. Hough, B. *The Joy of Computing: A Cookbook for Small and Rural Libraries*. (2008) Retrieved on: August 7, 2008. Available from: http://webjunction.org/maintainit-cookbooks/articles/content/456270.

37. Johansson, J.M. *Security Management - The Fundamental Tradeoffs*. (2004) Retrieved on: August 7, 2008. Available from: http://www.microsoft.com/technet/archive/community/columns/security/essays/tradeoff.mspx?mfr=true.

38. Karyda, M., et al., *An ontology for secure e-government applications*, in First International Conference on Availability, Reliability and Security (ARES'06). 2006, IEEE Computer Society.

39. Kemp, M., *Barbarians inside the gates: addressing internal security threats.* Network Security, 2005. 2005(6): p. 11-13.

40. Kessler, G.C. and Pritsky, N.T., *Local Area Networks*, in *Computer Security Handbook*, S. Bosworth and M.E. Kabay, Editors. 2002, John Wiley & Sons, Inc. p. 1224.

41. Kim, A., Luou, J., and Kang, M., *Security Ontology for Annotating Resources*, in 4th International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE'05). 2005: Agia Napa, Cyprus.

42. Kizza, J.M. and Kizza, F.M., *Securing the Information Infrastructure*. 2008: CyberTech Publishing.

43. Koenderink, N.J.J.R., Top, J.L., and van Vliet, L.J., *Expert-based ontology construction: a case-study in horticulture.* Database and Expert Systems Applications, 2005. Proceedings. Sixteenth International Workshop on, 2005: p. 383-387.

44. Leyden, J. *Student charged with massive ID fraud*. The Register, (2003) Retrieved on: August 5, 2008. Available from: http://www.theregister.co.uk/2003/02/07/student_charged_with_massive_id/.

45. Littlejohn, K. *10 things you should do to protect yourself on a public computer*. TechRepublic, (2008) Retrieved on: August 5, 2008. Available from: http://blogs.techrepublic.com.com/10things/?p=322.

46. Lyon, G. *Nmap - Free Security Scanner For Network Exploration & Security Audits*. Nmap.org, (2008) Retrieved on: August 5, 2008. Available from: http://nmap.org/.

47. Lyon, G. *Top 10 Vulnerability Scanners*. insecure.org, (2006) Retrieved on: August 5, 2008. Available from: http://sectools.org/vuln-scanners.html.

48. Lyon, G. *Top 11 Packet Sniffers*. insecure.org, (2006) Retrieved on: August 5, 2008. Available from: http://sectools.org/sniffers.html.

49. Manandhar S. et al, ed. *An Ontology for Network Security Attacks. AACC.* LNCS 3285. 2004, Springer-Verlag Berlin Heidelberg. 317-323.

50. McGuinness, D.L. and van Harmelen, F. *OWL Web Ontology Language Overview*. W3C (2004) Retrieved on: August 11, 2008. Available from: http://www.w3.org/TR/owl-features/.

51. Microsoft *5 Safety tips for using a public computer*. Retrieved on: August 5, 2008. Available from: http://www.microsoft.com/protect/yourself/mobile/publicpc.mspx.

52. MITRE. *Common Vulnerabilities and Exposures (CVE)*. cve.mitre.org (2008) Retrieved on: August 20, 2008. Available from: http://cve.mitre.org/.

53. MITRE. *Open Vulnerability and Assessment Language Repository*. oval.mitre.org (2008) Retrieved on: August 20, 2008. Available from: http://oval.mitre.org/.

54. NIST. *National Vulnerability Database Version 2.1*. nvd.nist.gov (2008) Retrieved on: August 20, 2008. Available from: http://nvd.nist.gov/.

55. Peltier, T.R., *Information Security Risk Analysis*. 2001: Auerbach.

56. Peters, C. *Introduction to Windows SteadyState*. WebJunction.org, (2007) Retrieved on: August 5, 2008. Available from: http://webjunction.org/25/articles/content/448713.

57. Pfleeger, C.P. and Pfleeger, S.L., *Security in Computing*. 4th ed. 2007: Prentice Hall.

58. Poole, O., *Network Security a practical guide*. 2003: Butterworth Heinemann.

59. Poulsen, K. *Guilty plea in Kinko's keystroke caper*. The Register, (2003) Retrieved on: August 5, 2008. Available from: http://www.theregister.co.uk/2003/07/19/guilty_plea_in_kinkos_keystroke/.

60. Raskin, V., et al., *Ontology in information security: a useful theoretical foundation and methodological tool*. Proceedings of the 2001 workshop on New security paradigms, ACM, 2001: p. 53-59.

61. Schauland, D. *Control users' temporary Internet files and browser history using Windows Server 2003 Group Policy*. TechRepublic, (2007) Retrieved on: August 5, 2008. Available from: http://blogs.techrepublic.com.com/datacenter/?p=204.

62. Slotta, J.D. and Chi, M.T.H., *Helping Students Understand Challenging Topics in Science Through Ontology Training*. Cognition and Instruction, 2006. Science Through Ontology Training(24:2): p. 261-289.

63. Stoneburner, G., Goguen, A., and Feringa, A. *Risk Management Guide for Information Technology Systems*. NIST, (2002) Retrieved on: August 7, 2008. Available from: http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf.

64. Tanase, M. *Sniffers: What They Are and How to Protect Yourself*. SecurityFocus, (2002) Retrieved on: August 5, 2008. Available from: http://www.securityfocus.com/infocus/1549.

65. Tsoumas, B. and Gritzalis, D. *Towards an Ontology-based Security Management*. in Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06). 2006: IEEE Computer Society.

66. Tsoumas, B., et al., *Security-by-Ontology: A Knowledge-Centric Approach*, in *Security and Privacy in Dynamic Environments. IFIP International Federation for Information Processing*, S. Fischer-Hubner, Rannenberg, K., Yngstrom, L., Lindskog, L., Editor. 2006, Springer Boston. p. 99-110.

67. US-CERT. *US-CERT Technical Cyber Security Alerts*. us-cert.gov (2008) Retrieved on: August 20, 2008. Available from: http://www.us-cert.gov/cas/techalerts/.

68. US-CERT. *US-CERT Vulnerability Notes*. kb.cert.org (2008) Retrieved on: August 20, 2008. Available from: http://www.kb.cert.org/vuls/byupdate?open&start=1&count=10.

69. Zuniga, G.L., *Ontology: its transformation from philosophy to information systems*. Proceedings of the international conference on Formal Ontology in Information Systems - Volume 2001, 2001: p. 187-197.