SECURITY BY DESIGN

by

M. James Tanner

A Thesis Presented in Partial Fulfillment

of the Requirements for the Degree

Master of Science

University of Advancing Technology

August 10, 2009

SECURITY BY DESIGN

by

M. James Tanner

has been approved

August 10, 2009

APPROVED:

VICTORIA SCHAUFUSS, PhD (ABD), Chair

GREG MILES, PhD, CISSP, CISA, CISM, IAM, IEM, Independent Reviewer

SHELLEY KEATING, MSIT, IEM, IAM, CCNA, MCSE+l, Network+, A+, CTT+, Independent Reviewer

AL KELLY, MSIS, MCSA, MCSE 2000m /2003, IEM, IAM, Network+, A+, MCT, Independent Reviewer

ACCEPTED AND SIGNED:

VICTORIA SCHAUFUSS, PhD (ABD)

Abstract

Securing a computer from unwanted intrusion requires astute planning and effort to effectively minimize the security invasions computers are plagued with today. While all of the efforts to secure a computer are needed, it seems that the underlying issue of what is being secured has been overlooked. The operating system is at the core of the security issue. Many applications and devices have been put into place to add layers of protection to an already weak operating system. Security did not used to be such a prominent issue because computers were not connected 24/7, they used dialup and did not experience the effects from connecting to multiple computers. Today computers connect to high speed Internet and seem useless without access to email, chat, Internet, and videos. This interconnectedness of computers has allowed the security of many computers to be compromised because they have not been programmatically secured. The core component of computer security might best be done through security layers protecting the operating system. For this research, those who work in the computer field were asked to complete a survey. The survey was used to gather information such as the security layers and enhancements implemented on Linux computers and networks their surrounding network. This research is a stepping stone for further research as to what can be done to further improve upon security and its current implementations.

Acknowledgements

Thanks goes out to all of the professors at the University of Advancing Technology who have brought so much information to the table and been available throughout this process. Professor Greg Miles thanks for all the information that has been shared in the information security classes. Professor Shelley Keating and Professor AL Kelly, thanks for the guidance and assistance with refining this thesis to make it what is today. Thanks also goes to those who assisted in the revision and editing processes namely Dad, Jose-Miguel Maldonado, and UAT Assistant Professor, Nathan Hamiel. Professor Victoria Schaufuss thanks for the assistance with each step of the thesis process. Most of all Onalisa is greatly appreciated for her unending support of both this thesis and the graduate classes.

Table of Contents

List of Tables

List of Figures

CHAPTER 1. INTRODUCTION

Introduction to the Problem

Operating systems are the core component of computer security. Computers have been designed for optimal use which is great for speed and ease of use but at the same time it could be the reason why computer security has become such a perplexing issue. Security by Design changes the mentality of making a computer secure because security starts with the operating system and continues on to each piece of software. Well built operating systems will have fine grained permissions which control access rights to specific files, folders, and applications on a user by user basis. Fine grained permissions are time consuming and slow down everything until all the rules and account permissions are defined; the user will be locked out of security aspects of the root program that in the past were easily accessible. However, correctly designed programs will not require the user to have full rights to the program and all related directories.

Secure systems consist of protections at all points of connections. Each device that is connected to a computer could introduce malicious code, so an antivirus program must be installed to compensate for the vulnerability. Another point of connection is the Internet; layers can be implemented to compensate for the vulnerabilities that occur. The layers consist of antivirus software, firewalls, intrusion detection, proxy server, and demilitarized zone. Even with all of these security layers deployed, intruders are still able to get past the most advanced systems. Many firewalls are configured with settings to deny inbound traffic unless requested or defined, and allow any outbound traffic. The default settings to allow any outbound traffic are a significant security risk. To have maximum security, a firewall should have the default set to deny unless permission is granted to enter or exit the network. Maximum security is not always possible and should be setup in a manner that maintains confidentiality, integrity and availability of the information.

Statement of the Problem

Security layers are critical because they protect the computer and network from intrusions. Security needs to be designed around the entire operating system, because it will provide another essential layer that will help minimize the exploitation of computers. Applications that require users to have full access to programs, and associated files may leave the computer open to exploitation. As a result, system administrators must limit users rights to allow each person to perform their functions while still giving them flexibility to use the computer. The implementation of security layers has yet to fully protect computers from all vulnerabilities.

Purpose of the Study

The overall goal of this analysis provides academic thought by exploring what can be done to improve the overall security of computers systems. The answer to this question would give a good perspective to the Linux community as to what would best improve security. This study will also provide businesses who are considering switching to Linux with a reason to do so. This research will be a benefit to those companies because it will show what is being done to protect Linux computers and networks. The summary results will be offered to the respondents, and the thesis as a whole will be submitted to the networked digital library of theses and dissertations.

Research Questions

The questions this thesis attempts to answer are listed here. Has AppArmor enhanced fine grained application controls and as a result helped secure Unix based computers? Has the incorporation of SELinux into the Linux Kernel been a security enhancement? Does a hardened version of Linux lower compromises of the operating system? What other security layers have been implemented to enhance the level of security?

Significance of the Study

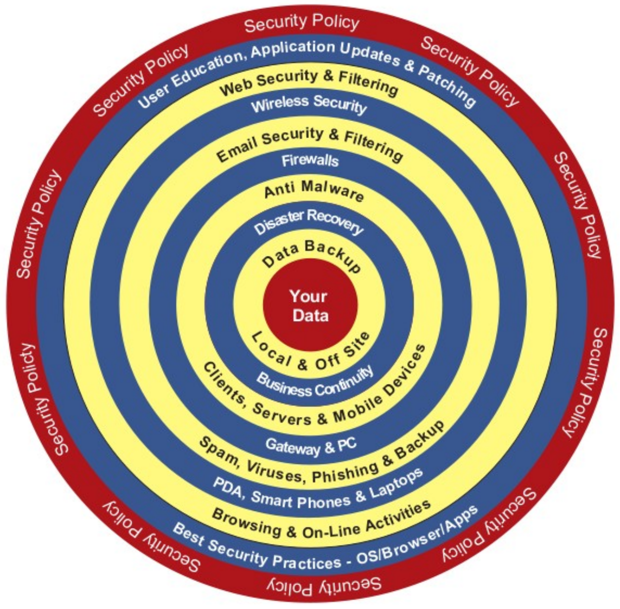
If Linux is found to be more secure because of modifications to the basic Linux install, and the implementation of security layers such as AppArmor, SELinux, PaX, GRSecurity and User Account Controls (UAC), then this study will point out the correlations. These results will help the system administrators know where to focus their efforts to continue to improve Linux security as the digital age moves forward. The businesses who will be a part of this study use both terminal server and terminal client so overall Linux security will be included to help the administrator improve security in a network environment. The results also may be of worth to the programmer so they can see the effects of their work and more effectively improve the security after seeing the results through the eyes of the Linux administrator.

CHAPTER 2. LITERATURE REVIEW

Introduction

Computers that do not have proper security measures in place will most likely be affected by security threats. Why does it matter that a computer is infected? Computer infections can affect the end user by causing the computer catastrophic hardware malfunction or the invasion of personal data which can ultimately allow criminal behavior such as identity theft. Some computers can become infected which enables malicious users to use a computer without the owner's knowledge. What can be done to protect computers? Currently, the most effective solution to computer security is to implement security layers. Computer security layers can be either proactive or reactive. Proactive computer security examples are policies, procedures, training, firewalls and regular monitoring of the network. Reactive computer security examples are up-to-date Anti Malware protection, and Intrusion Detection Systems. But these security measures only protect from a distance, they do not solve the vulnerabilities that exist. Software and operating systems should be redesigned so that they have built in security layers thus providing a “belt and suspenders” approach to fortify existing measures. The belt and suspenders approach will not allow software to interact directly with each other but rather with a separate program that deals with the resource coordination (Bunnell & Weinberg, 1996)? Before a software program comes to market it must be created. Security should be a part of the software creation process but is often skipped or the software comes insecure by default thus leaving users and administrators with security vulnerabilities from the start.

Figure 1. Security Solutions


Security Policy
User Education, Application Updates & Patching
Web Security & Filtering
Wireless Security
Email Security & Filtering
Firewalls
Anti Malware
Disaster Recovery
Data Backup
Your Data
Local & Off Site
Business Continuity
Clients, Servers & Mobile Devices
Gateway & PC
Spam, Viruses, Phishing & Backup
PDA, Smart Phones & Laptops
Browsing & On-Line Activities
Best Security Practices - OS/Browser/Apps

Note. From Security Solutions. (n.d.). *Layered Model.* Retrieved April 2, 2009, from softwaresecuritysolutions.com/PDF/layeredModel062308.pdf

Both Linux and Unix are built upon the kernel architecture;

> the kernel accomplishes all these tasks by providing an interface between the other programs running under its control and the physical hardware of the computer; this interface, the system call interface, effectively insulates the other programs on the UNIX system from the complexities of the computer (Indiana University, 2003, ¶ 1).

In addition, Unix runs as a low level user except for when access is needed to root. Root is similar to administrator on a Windows machine. What can be done to improve the security of Linux and make it secure by design?

Current Security Solutions

Security is a process by which computers something become less vulnerable (Granneman, 2003). It can be a process of eliminating the holes in a computer. The Internet allows all computers to become so interconnected that it becomes almost impossible to know where to start to implement security. The more security that is implemented, the more complex they become which makes them even more difficult to maintain. Each user, resource and trust relationship must be defined to correctly allow or place limitations on what can be done (Lampson, 2004). Keeping these relationships limited while still allowing the connection to exist between organizations really makes security hard to establish and coordinate. Security really provides nothing, other than protecting data or computers (Lampson). Because of the complexity with security implementation, it has been implemented in a limited fashion so computers can continue to be useful. Network security is currently implemented through intrusion tools, which includes Anti-virus programs, Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), and penetration testing. OS security is implemented through access control such as Mandatory Access Control (MAC) or Role Based Access Control (RBAC). Patching has also become a necessary part of security because as software flaws are detected they are repaired; the most common option available is to patch the vulnerabilities. Patching often times does not remove the need to identify and fix security flaws because the vulnerabilities may just be getting covered up. Think of patching like a band-aid; it is not better yet it may take more patches or the software may need to be completely rewritten in the next version.

Computer security affects every user on a daily basis; passwords are one method used to authenticate users. Bardram (2005) talks about the problems logging in either with 1-factor authentication such as passwords or 2-factor authentication such as smart cards and encourages the use of biometric devices. It is easier to place a fingerprint on a device than type a user name and password, and it can be secure. George Waller is also convinced that security on computers would increase if consumers got rid of username and PIN or password combinations (Germain, 2005). The method Waller refers to is using one biometric device and a non-network device such as a phone. This method is supposed to verify a user in a much more secure way than a username and password that can be cracked, or be susceptible to man-in-the-middle attacks. Biometric devices will not solve phishing and keylogging completely because computers still have keyboards which are used to enter information other than user names and passwords. Acceptance of biometric devices is also slow because of the cost, and complexity. It is not quite ready for full blown usage, biometrics may only work on the local computer and not over the Internet.

It is very interesting to note that just because malware protection is implemented it feels like there is a safety net even without proof that it is working. Antivirus programs, the very program that users feel protects them actually does the opposite of what is expected. Antivirus coders have a quick reaction time to find solutions to malware. This could be the cause of a decrease in the quality of the software. Nevertheless the more the program parses files the more susceptible the computer can become because the antivirus solution becomes a target as it opens these files (Danchev, 2008). The antivirus program actually has to understand many file formats, parse them, and by doing so, the files get split up into recognizable formats which increases the infiltration from the viruses and makes the antivirus program itself a target for attacks.

Why does it matter that a computer is infected or hacked?

It takes less than five minutes to get infected, according to Richie Lai who was cited in a post about zombie PCs (Taylor & Nusca, 2008). Zombie computers are computers that get infected and run programs in the back-ground to gather financial information, send Spam, and spread the infection to other computers. These computers are joined to other computers in the same purpose “into systems called 'botnets' and forced to do a shadowy figure’s bidding, namely in the form of automated programs” (Taylor & Nusca, ¶ 1). The information security field was created because there is a need to protect the information that is stored on computers. Infected computers lead to data loss and data loss results in businesses leaking out information. Data loss usually represents customer data, or information that is specific to their organization which could lose them money and customers. More recently the FBI has warned that there are “'a couple dozen' countries ... eager to hack U.S. government, corporate and military networks” (Noyes, 2008, ¶ 1). Hackers can be either good or bad; some hackers hack for the challenge and then report the information so the bugs can get fixed while other hackers can be malicious, like the hackers that would like to get into the U.S. Government networks.

What can be done to protect computers?

When best practices are the ideal, it is more important to actually see results that a network is more secure than to state a business is following best practices. Use best practices to the company's advantage but do not expect a network to be secure because every step has been followed. Security takes constant monitoring, patching, and consistent effort by securing the vulnerabilities. Keeping software up to date is very important; each new version of the operating system attempts to increase the overall security level. Security is often an afterthought; it is commonly not planned into software. As a result it is important for system administrators to implement security on each network as a layer of protection. Security is improved or at the very least maintained as patches are applied to the operating system, and its accompanying software. Patches fix exploits and vulnerabilities of software (Williams, 2007). Software is constantly under scrutiny for an acceptable level of security; however, a majority of the time there is always a vulnerability that needs to be fixed. By the time it gets deployed to the public many computers may have been compromised thus it is important to apply the updates as quickly as possible.

Software designers should have a quality assurance process where they find bugs and vulnerabilities in their code. This whole process should be a regular part of all software development life cycles. Obscuring or protecting code will not work because someone will always want to hack the code and find vulnerabilities.

> The publication of source code actually improves security because the program or operating system can be peer-reviewed by anyone who cares to read it. Many security bugs that are overlooked in other operating systems have been caught and repaired in Linux, because of its extensive peer-review process (Perens, 1998, ¶ 6).

A secure network includes hardened clients, servers and firewalls. The use of firewalls and Intrusion Prevention Systems (IPS) enhances a networks protection and takes it to the next level especially when they are tested and monitored regularly. Well written enforceable security policies can also aid in the maintenance of a secure network. However, an experienced hacker can get past what was thought to be a secured network . Big mistakes companies make is to not hire skilled professionals or at least have them trained in the use of IPS. Security professionals should know what regular network traffic looks like and help detect network intrusions (Parker, 2008).

Another important layer of protection is the hardening of computers which consists of the removal of unused programs, and disabling any unnecessary services. Many programs open the computer by listening on well known ports, which gives more attack surface to that computer. Some distributions come with local root login disabled, this makes root more of a group. The security comes from not having a root user that will be targeted. If the administrative account is not named root or administrator it may be more difficult to decipher who has administrative privileges especially when these privileges can be revoked easily. Another security aspect of not sharing a central administrative account it becomes more difficult to discover an unknown password. Entering the root password allows the user to temporarily escalate privileges to perform administrative functions. The use of the “sudo” command is one example of temporarily privilege escalation to perform administrative tasks (Hamid, 2008).

Williams (2007) recommends the use of Bastille, an open source program for automating the hardening process, because it analyzes the system and asks a series of questions with explanations of what will occur if each task is done. As a result Bastille “will disable unnecessary services and install operating system updates as well as configure a firewall, enforce password policies, create a second root-level account and more” (Williams, p. 2). A hardened system regardless, of the operating system is an important part of a secure computer. The best solution to security, according to Ranum (2005), is default deny. Default deny, also known as least privilege, blocks access until given explicit permissions to each resource on the network

(Woods Hole Oceanographic Institution, 2007). Implementing this kind of policy is much more challenging but in the long run it will save a lot of time and effort. The application of least privilege would lock every user out until each user is given rights to each piece of the network or system. Other items mentioned by Chuvakin (2002) to harden a Linux system are to patch software, secure file permissions and S*ID binaries, improve login and user security, set controls on physical devices and boot settings, secure daemons, increase logging and audits, and configure security software.

A hardened system would also include the use of secure FTP, HTTPS, encryption and other security methods to limit the ease of access to information that is transmitted or stored on a server or client computer (Perrin, 2007). Encryption takes readable files and makes them unreadable until they are unencrypted. Encryption can be done by different mathematical algorithms, and through the use of many different programs. Once the user enters the key or gives their certificate then the information can be used (Cherry & Imwinkelried, 2008). Linux Unified Key Setup (LUKS) can encrypt hard drives. LUKS is a standardized way to encrypt a disk because it is not vendor specific and can encrypt any operating system (Fruhwirth, 2006). One of the main reasons to encrypt hard drives is to make sensitive information unreadable. However, while the hard drive or files are unencrypted then they are completely readable by the user, and anyone who can gain access to the system (Absolute Software, n.d.). An encrypted volume has its limitations for example while the data is being accessed the key and other data are stored in memory. It is possible that an attacker could get the keys while the computer is on or in standby mode. MacIver (2008) suggests hibernation while the computer is not in use. Hibernation stores the information to the hard drive and powers the computer off completely where placing the computer in standby would leave the information in memory so an attacker could potentially get to the key.

Encryption can also help secure the network traffic. It takes coordination but is worth the effort because it adds one layer of protection to the data being transmitted. The Linux kernel 2.6 includes IPsec support:

> Without kernel support, Linux would have continued to be perhaps the only enterprise-wide platform that could not do IPsec out of the box -- a definite shortfall. Not only will it increase the overall reliability and performance of the product, but it also adds to the overall perception that Linux is truly an enterprise-ready platform for the future (Milberg, 2004, ¶ 10).

Encryption protects the Confidentiality and Integrity of the information transmitted on the network. Perrin (2007) gives an example that something as simple as a user name and password could be seen if the traffic is not encrypted which would add a vulnerability to the FTP server and the FTP server could become another porn site.

A hardened system also includes a comprehensive security policy. Without a policy users can have blank passwords, leave the computer unlocked, and create a security nightmare. Cisco did a study to find out the perceptions of 2,000 security professionals. Cisco found that 42% believed that security did not fit with how they did their job. If users choose not to follow security policies they are leaving the network open to intrusions (Kerner, 2008). Many IT people blame the users of the network for security when the policy could have been created without thinking about the user and the company culture. Kerner goes on to state that when users understand the business consequences of their actions they are more likely to comply with the security policy.

Security breach disclosure may not sound like a good idea but “full disclosure helps to ensure that when bugs are found the 'window of vulnerability' is kept as short as possible” (De Groot, 2006, ¶ 9). Disclosure helps people become informed that there is a security problem; it makes it so other software creators can fix similar issues and in general it is a good idea because it actually decreases security risks. For example, airliners must disclose errors. Similar to open source, the industries that are under public scrutiny such as pharmaceutical companies and the airline industry have improved. Doing a similar thing with security breaches will at first be a big deal, but as time goes on, disclosing security breaches will become common practice and as a result the information that can be gathered from these breaches will provide a learning opportunity for other organizations to make improvements on their security (Shostack & Stewart, 2008). In addition to disclosing security breaches, knowing what dependencies operating systems are going to have could make a computer more secure if the information is used by the software vendors to tie their software into the security system of the operating system. Additionally, as more demand security built into the operating system more research and development will occur (Loscocco, Smalley, Muckelbauer, Taylor, Turner & Farrell, 1998).

Frameworks

Frameworks can be very difficult to follow because these guidelines are very general and can be applied in many ways. Keeping things general makes it harder to implement security because security is more lax in these conditions. Checklists are great to make sure that the bare minimum is done, but in reality security needs to go way beyond generalities and checklists (Baskerville, 1993). Security frameworks if used as a guideline will assist in creating security from the Internet that comes into the organization to the databases (Frazier, n.d.). Which frameworks provide the guidance needed to secure a network? Rothman (2007) believes Control Objectives for Information and related Technology (CobiT) is one of two leading frameworks. CobiT is a framework for IT governance to help effectively control each IT process. “Effective controls reduce risk, increase the likelihood of value delivery and improve efficiency because there will be fewer errors and a more consistent management approach” (IT Governance Institute, 2007, p. 15). Frameworks are just guidelines and do not actually secure software unless proof can be shown that the systems actually had improvement. A before and after snapshot or a vulnerability scan could prove that there was an actual difference.

Architecture Design of the Operating System

Architecture of an operating system has much to do with why there are security flaws. Software needs to be designed and tested for security throughout the creation process (Yodaiken, 2004). Software including the operating system should be designed with security in mind as each user will be affected by the design of the operating system. Halkidis, Tsantalis, Chatzigeorgiou, and Stephanides (2008) did a study on secure and non-secure e-commerce systems. They found out that all else being equal the non-secure application was at a much higher risk than the secure application of being affected by STRIDE attacks (Halkidis et al.). In addition to lower risks, the expenses were lower when security was a part of the design phase rather than after the software was designed (Halkidid et al.). Lower costs could be one incentive to implement security early in the development stage.

Linux keeps a separation between the operating system and the applications that are run on Linux. Integration of applications with the operating system appears to make things more functional and user friendly, but at the same time security is greatly reduced because all programs can affect any other program. For example, while browsing the Internet a script could be ran which states the anti-virus program is out of date, a user would then click to update the program. The program runs as administrator and infects the computer. Although this is a devious way of infecting a computer it happens. What are the problems with this scenario? The default user should not be administrator, and programs should be given limited access to the rest of the computer. One method of limiting programs access to a computer is through user access controls (UAC):

> Rather than merely acting as containers and dispensers of data and functionality, software objects may become actively responsible for the protection of those resources. By basing the design of SAAs on the SDO concept, the provision of application-specific usercentric access control is simplified by localizing the access control mechanisms (Holford, 2006, p. 9).

Security is a software design choice and many security precautions are not taken because implementing them takes more effort from the software designers and the network administrators.

The principle of least authority (POLA) as another access control but it only worked at the level of the user. It would be better implemented this at the level of the processes or objects within a process. Using POLA would limit the damage that a virus could have on a computer because it would stop it at the process level instead of the user level (Karp, 2003). Users are given more access to a computer than a process. POLA greatly restricts what processes can and cannot do. Mandatory access control (MAC) could also limit the extent malicious software could have on a computer. MAC forces separation, and adds security to a computer because of this separation. It is missing on the popular OS and as a result application security suffers because it can be tampered with until patched (National Security Agency, 2007). Mandatory access controls restrain programs from getting administrative access to the operating system. In order to increase overall security, it is important to implement the principle of least privilege which means that course grained policies should be replaced with fine grained permissions (Loscocco et al., 1998). The principle of least privilege is similar to not trusting anyone until trust is deserved. Trusted applications should be limited by POLA because it controls what each application gets to do regardless of if it is trustworthy. Administrative rights to some features such as the installation of printers, approved patches, setting up email, the modification of power settings and other little tasks can be set by groups thus relieving network administrators to their important tasks. However, this relief needs to have limitations, otherwise root access in the wrong hands could allow the following issues:

> Machine Misconfiguration — Users with root access can misconfigure their machines and require assistance or worse, open up security holes without knowing it. Run Insecure Services — Users with root access may run insecure servers on their machine, such as

> FTP or Telnet, potentially putting usernames and passwords at risk as they pass over the network in the clear. Running Email Attachments As Root — Although rare, email viruses that effect Linux do exist. The only time they are a threat, however, is when they are run by the root user (Administrative Controls, n.d., ¶ 4).

Role based access control (RBAC) is another method to improve security. PaX uses RBAC to prevent and contain exploitation; it does so through a patch to Linux. Attacks can come from three main levels: (1) introduce/execute arbitrary code, (2) execute existing code out of original program order and (3) execute existing code in original program order with arbitrary data. Stack smash protection can come from PaX. PaX is implemented by either the hardened Gentoo project or GRSecurity (Documentation for the PaX project, 2006).

Kernels

Kernels can be either monolithic, macro or micro (Linux-friendly microkernel OS tightens mobile security, 2008). For example some phones run OKL4 which is a microkernel OS whereas Linux is a macrokernel OS. Microkernels run with a minimalistic kernel mode and let the rest run at the user level. The dilemma with microkernels is that they can be slower (LinuxDevices.com, 2007). The advantage to microkernels are that they tend to be more secure because of the complete separation between address spaces and user-mode (Device management | Channel 9, n.d.).

Although Linux uses a macrokernel it tends to adopt the best design from multiple Unix Kernels (Bovet & Cesati, 2005). Some design features of Linux include the default user is not root (administrator), but root access could be gained through privileged escalation. Each time a user needs to do administrative work the root password must be entered. What makes this an improved security feature is the fine grained permissions and a kernel controlling what processes get access to each piece of the system. Linux splits permissions into three levels user, group and everyone else (Jones, 2008).

Linux is divided into what is called a user-space and a kernel-space. The kernel is the core of the operating system; all applications are able to talk to the hardware and programs after going through the kernel. “The kernel actually runs several processes concurrently, and is responsible for mediating access to hardware resources so that each process has fair access while inter-process security is maintained” (Bowman, 1998, ¶ 10). User space, also known as user land, is where programs are run. Dividing the processes from the user applications help to secure the operating system. Linux also has five main subsystems: The process scheduler, memory manager, virtual file system, network Interface, and inter-process communication. All processes including the subsystems depend upon the process scheduler. Bovet and Cesati (2005) state that running an operating system in user mode and placing a level between the user and the operating system makes programming easier. Programming for this type of operating system makes it easier because the programmer does not have to learn low level code. Security is also enhanced because there is a check and balance system. The Linux kernel can verify the validity of the request at the interface before allowing it to proceed (Bovet & Cesati). Some examples of kernel security issues are denial of service vulnerabilities such as the '/ipc/shm.c' “There is a race between sctp_rcv() and sctp_accept() where we have moved the association from the listening socket to the accepted socket, but sctp_rcv() processing cached the old socket and continues to use it" (Kroah-Hartman, 2009, ¶ 1).

AppArmor is another option which restrains programs from having unlimited access to the operating system through the use of policies or profiles. It does not stop vulnerabilities but it does mitigate the affects of intrusions by implementing access controls on applications. It uses a learning based system similar to IPS which can learn the network traffic, yet AppArmor restricts the access each program has to the system. Although AppArmor has its benefits, it is not a silver bullet.

Another option or implementation of the separation of programs from privileges is PolicyKit. PolicyKit is a framework for allowing privilege processes to interact with non privileged processes. PolicyKit assumes that a program is split in two parts one part called the mechanism that can run with elevated privileges while the policy part can run in the user session with limited privileges. The hardware abstraction layer (HAL) and NetworkManager work under this model. The important part of this separation is that the mechanism should verify all requests it receives from the application to prevent misuse (Policy Kit Library Reference Manual, 2008).

Sandboxing has been introduced by Apple in Mac OS X Leopard. Sandboxing is similar to the functionality of AppArmor, any process or program running in a sandbox has limitations on what effects it can have on the other processes. For example, a process called soso is run in a sandbox, any subprocesses of that game can only see processes within the same container. Yet the main process, called soso in this instance, can see all processes running on the computer. The security concept behind sandboxes is to contain or place limitations on what processes can do. So if a process is malicious in nature it will be contained to the sandbox and not affect the whole computer (Singh, 2004). Although Sandboxing has been included in Leopard it is not used as it should be used:

> By default, the provided sandbox policies aren't even used by OS X to protect many of the daemons which run by default: such as syslog or ntpd. To enable sandbox protection on these services, the files /System/Library/LaunchDaemons/com.apple.syslogd.plist and /usr/libexec/ntpd-wrapper have to be edited (Ruoho, 2007, ¶ 7).

Sandboxing is yet another example of applications that come insecure by default. Using either AppArmor, PolicyKit or Sandboxing together with MAC will help contain the effects of an attack. AppArmor will aid in the separation process and mandating the policies of the computer together make for a more secure computer. MAC only protects resources not what application rights exist on a computer. MAC can also be applied to databases by restricting tables and other contents of the database. However, anyone who gains access to the system will most likely have control of the database (Runesson, 2006). MAC is just one portion of the security layers needed to keep computers secure. Another security layer is role based access control (RBAC) which has been studied by the National Security Agency (NSA). They have also studied MAC and the feasibility of their implementation in commercial operating systems (Holford, 2006) The feasibility of RBAC being truly effective depends upon its implementation and constant updates to keep the access control in compliance with the current state of the file and operating system structure.

> Support for RBAC will significantly increase the security provided by the system, however the ability to take this option is reliant upon the provision of an appropriate budget and availability of highly skilled personnel, to support the design the role hierarchy, appropriately configure the operating system and continued maintenance of a secure configuration despite changing organizational requirements (Holford, p. 226).

Address space layout permutation (ASLP) could secure a computer from malicious code. Kil, Jun, Bookholt, and Xu (n.d.) analyzed ASLP; ASLP randomizes the address location for programs that are run on a computer. The randomization can be done from the user or kernel level. The benefit of ASLP is that it makes it difficult for an attacker because the location where the program is being run is random and because of the randomness it makes it difficult to do a callback to the program. (Shacham, Page, Pfaff, Goh, Modadugu, and Boneh, 2004)

Randomization helps keep an attacker from knowing where the program is running. Three programs that use the ASLR features are Position Independent Executables (PIE), PaX and NX. It is much harder to exploit if the memory address is unpredictable. PIE makes it possible for applications to load to a different memory address at startup (Sundaram, n.d.). PaX also randomizes the addresses a program uses each time a task is created which causes the attacker to have to guess where the program is running. If the attacker chooses to use brute force the program will most likely crash the attacked application. Crash detection and reaction mechanism is not a part of PaX however, it would be a good thing to because it could stop attacks completely (Documentation for the PaX project, 2006). NX takes protection to the next level by allowing the system to flag data and application memory as non-executable (Sundaram). Without the ability to write to memory exploitations will not function properly.

Another enhancement to the security design of Linux came from the NSA. SELinux is in an enhancement that has become a part of the Linux kernel. Originally SELinux was implemented as single-access control architecture but it was then adapted to the Linux Security Module which allowed hooks to be used to enforce a specific security policy. SELinux allows policies to be defined to allow or disallow items from taking place. The protection behind SELinux comes from the ability to make limitations on what can or cannot be done especially with core operating system components (Jones, 2008). Each process that asks for permission to run has to be checked against known programs. SELinux policy server makes it possible for user-space applications and daemons to integrate into SELinux (Tresys Technology, 2007) . The policy management server allows object classes to be dynamically registered (Jones).

Linux is open source which is the opposite of security through obscurity. Opening the source code makes it easier to find and make corrections to flaws in applications. This is because there is a broad community of individuals willing to test and make suggestions to improve the code (Lettice, 2004). Linux is not secure just because it shares its source code. If Linux source code were to undergo an audit of the FAA it would not last five minutes (O'Dowd, 2004). Security for Green Hills software occurs because it undergoes testing by authorities and then improvement is made for those security problems. Green Hills software publishes their source code similar to Linux. Security through obscurity is a label tagged on to vendors who do not open their code up to the community. Typically this label is for those who hide their source code to prevent attackers from finding vulnerabilities in their applications. However, it is a mode of survival for many commercial vendors because without keeping their code secret many others could take the code and improve upon it similar to what has been done with Linux (O'Dowd).

Linux is not secure because the source code is open; it takes rigorous testing by hackers, coders and users to take it to the level it has achieved. Open source coders give their code to a community of people who are willing to make adjustments, corrections or suggestions because the open source community thrives on sharing with others. This follows the thinking by O'Reilly and Raymond (2001) where it is suggested to release early and often so the open source community can reply and help fix bugs before the official software is released. This type of releasing of source code works well with the open source community. It could be one of the best ways to correct errors because the bugs can be fixed before the public start using the same version of the software. This type of testing is called beta testing.

Guardian Digital has created EnGarde Secure Linux using open source software with a top priority to make security ingrained in all every part of the operating system. One of the methods used is the principle of least privilege which by default only gives users, programs, and services the least access necessary to the system (Secure By Design, 2006). This fine grained security model of least privilege is very time consuming but it is needed to help contain the affects of intrusions. If an operating system never gives an application full rights to the computer then security is greatly enhanced because without the resources malicious code might not have the ability to control the computer or the network.

EnGarde has revamped the security policies for their version of Linux. This is a very hardened operating system because not even the root user has unlimited power. Policies define the scope of what can or cannot be done with each user, process or program. Access must be granted deliberately, in other words this operating system is secure by default and has fine grained permissions (Secure By Design, 2006). EnGarde Linux has created fine grained UAC which could be part of the solution to repair security problems. Dividing the operating system so that multiple subsystems need to be compromised before the attacker can have any effect on the system or the information it holds. In addition each system and subsystem should default to secure instead of insecure because less systems would be left open to undesired intrusions (De Groot, 2006). Fine grained UAC also makes it difficult to do much to the computer because gaining root access to one part of the computer does not give full access to the whole computer. Each function would be separated into user functions, and making a computer insecure would need to be deliberate rather than the default. “The specific policy that is enforced by the kernel is dictated by security policy configuration files which include type enforcement and role-based access control components” (National Security Agency, 2009, ¶ 1). Role based access control is addressed by Loscocco and Smalley (2004).

Operating system security is only part of the security model. Having fine grained security that locks out programs and users from malicious intent is another portion of the solution to computer security. Additionally it takes application-specific security that builds upon the OS

> A highly secure operating system would be insufficient without application-specific security built upon it. Certain problems are actually better addressed by security implemented above the operating system. One such example is an electronic commerce system that requires a digital signature on each transaction. An application-space cryptographic mechanism in the transaction system protected by secure operating system features might offer the best system security solution (Loscocco et al., 1998, ¶ 78).

Security needs to be well rounded for security to be effective on the system as a whole.

Conclusion

Computer security is a fairly recent field of research and is being developed as new ideas are introduced. Security issues can be greatly minimized by implementing access controls. Currently the solution to security issues is to patch and cover issues when in reality the software needs to be designed with more security. Updates are released on a regular basis to secure one more security issue with the Linux kernel, or one of the installed software applications and remote executable code. No operating system, computer or network is completely secure.

Operating systems architectures are designed differently. The analysis in this literature review shows that some flaws have not been properly addressed. These flaws need to be improved to properly secure computers. Some options for further research include investigating ASLP. “Although ASLP can mitigate many types of memory corruption attacks, current implementation of ASLP does not support stack frame randomization. This can be achieved by adding pads among elements [1] in the stack frame” (Kil, Jun, Bookholt, & Xu, n.d., p. 9). In addition operating systems need to have proper auditing as well as proper controls on what software restrictions (National Security Agency, 2007). Auditing and control could be done through more fine grain permissions, but optimally should be done in combination with other security layers like MAC, NAC, AppArmor or sandbox, ASLP.

Statement of Thesis

Security layers must be added to an already weak operating system to make the system secure by design. Operating systems will never be completely secure and as such they need layers of protection. Security layers such as Anti-malware programs, firewalls, and intrusion prevention systems have been implemented to decrease the attack front of computers; however, these layers are not enough to protect an already weak operating system. Incorporating security layers into the operating system involves implementing access controls. Computers were built to have maximum usability such that applications come insecure by default and must be setup for the environment. Networks also provide insecurities because they connect multiple computers together. Network devices are set insecure by default which permits anything that is not defined to be allowed. Clearly there has been a choice to set the default settings to minimal security which gives maximum usability of the operating system and its accompanying applications to the users.

CHAPTER 3. METHODOLOGY

Security layers are critical because they protect the network and computer, but what about the operating system? Security needs to be designed into the entire operating system, because it will provide another essential layer that will help minimize the exploitation of computers. Zhang (2006) did research on the subject of software security. Zhang did case analyses on Microsoft IIS 6.0, and Apache from the perspective of the software vendor/programmers. Research should also be done from the perspective of those who install and maintain software. For this very reason a survey was conducted on businesses that install and maintain operating system software. Vulnerabilities exist in each operating system. This research has attempted to answer the following question: What do system administrators do to improve overall security in a Linux operating system environment? If the research showed that Linux security has improved then this research has potentially discovered what improved the security. The analysis was done using a quantitative analysis in an Ex post facto design format.

Research Expectations

It was expected that of the 474 possible respondents listed on http://hitachi-id.com/linux-biz/ and the LinkedIn group of “Linux System admins” that maybe one third would have responded. Of those who responded some may not have the experience desired and the results would vary greatly from those who have both experience and knowledge in the information security field. As further motivation and to keep the sample size reasonable the summary results were offered to those who take the survey.

Research Design

The survey was hosted by http://www.kwiksurveys.com due to the ability to restrict surveys by IP address, and automatic survey result statistics are calculated. Research was gathered through the survey that was sent to IT staff members who have experience installing and maintaining operating system security on the Linux OS. A list of businesses that use Linux is maintained at http://hitachi-id.com/linux-biz/. This list was used to invite those businesses to participate in the survey. The survey consisting of 26 questions was available online for respondents to take once. The first questions asked were to understand the survey participant and what level of experience they had with security and the information technology field. The next series of questions asked what operating systems and additional security enhancements were run on their computers, such as psad, AppArmor, or SELinux. Each security enhancement was evaluated for its effectiveness in securing the Linux computers and reported in the results section. Lastly, an analysis of UAC, fine grained permissions and overall computer security was included in the survey. The changes analyzed were the inclusion of SELinux into the Linux kernel, and if AppArmor had been implemented. GRSecurity & PaX provide restrictions on applications in a UNIX environment and it was desired to know if it had been implemented. Finally, it was desired to know if any of these layers could be correlated to a decrease in malware, kernel crashes or memory exceptions.

Participants

The participants of this survey were IT staff members who had experience installing and maintaining operating systems and security mechanisms in a Linux or mixed operating system environments. Businesses who use Linux may run in a mixed operating system environments with both Linux and a variety of other operating systems. The operating system that was included in this survey was any Linux distribution using a kernel before or after 2.6.

Sampling Strategy

Businesses were chosen as respondents instead of home users because security rules and regulations are on the rise and they are legally required to comply. Therefore, they have a more vested interest to attend to security enhancements. Convenience sampling was used because it is virtually impossible to acquire a list of all businesses that use Linux for daily operations. Without an all encompassing list of businesses it was impossible to pick a random sample. Instead a list of sufficient size has been provided by Hitachi ID systems INC who maintains a list of businesses who use Linux to support their daily operations. This list contains 425 businesses from worldwide locations who operate in the following fields; Aerospace, agriculture, automotive, business services, computer vendors, other consultants, educational and research, engineering, environmental, financial, government, health care, instrumentation, Internet, network services, Internet service providers, law firms, manufacturing, media companies, military, non-profit/charity, oil and gas, publishers, real-estate and construction, software developers, sports and recreation, technical support, telecoms, tourism and travel, transportation, wholesale and retailers. IT departments from each industry were invited to take the online survey. A vast majority of the listing was out of date and as a result there were only 425 of the original 717 on the list. LinkedIn was also used to contact Linux system administrators as a result of the decreased sample size. The group “Linux System admins” found on LinkedIn were contacted individually with a request to take the exact same survey as the previous group. Some of the group members did not currently work in IT as a result they were not invited to take the survey. The group had 61 members not including myself, however, subtracting those who were in other fields such as marketing or recruitment the total survey invitations sent out were 49. Gay and Airaisan (2003) suggest that for any population of 5,000 or more 400 should be adequate. The Local Program Evaluation in Tobacco Control (2003) confirms this information because for a population of 100,000 the sample size should be 398 for a 5% precision level. 474 is a reasonable sampling size of IT staff members.

Process of Data Collection

The survey was available on the Internet where each business was able to take the survey once. Email messages and telephone calls were the method used to asked businesses to take the survey. Due to the fact that most of the list was outdated it was much faster to go to each business website and find an updated contact such as an email address. Then an an email was sent to them with the invitation link to the survey. Any IT staff member, preferably managers or department heads of information technology were invited to a secure web based survey. These web surveys were used to find out what security layers are used to enhance the Linux operating system. Some questions were yes or no, multiple choice, fill in the blank, or based on a five point scale from daily to yearly. Before beginning the survey the following introduction was posted.

> Just to give you a brief run down on how the questions go: The first questions are to understand what angle of the IT field with which you have experience. The questions will then be directed to see what Linux Distributions are used on servers and workstation

computers. Following those questions the security layers will be addressed because as IT professionals we are constantly addressing the security flaws of the operating system. Please be aware that the survey is anonymous and any information directly related to your company cannot be tied back to you nor will it be be disclosed if it isn't anonymous.

After extending the initial invitation to take the survey each respondent received a follow up email, unless they sent an email as notification that the survey had been taken. The follow up email was to remind and re-invite each person in the sample to take the survey.

Research Questions

Has AppArmor enhanced fine grained application controls and as a result helped secure Unix based computers? Does a hardened version of Linux lower compromises of the operating system? What other security layers have been implemented to enhance the level of security? Has the incorporation of SELinux into the Linux Kernel been a security enhancement?

Data Processing and Analysis

Data analysis consisted of preparation, exploration, analysis, representing, and validating the data (Clark, & Creswell, 2006). The data was the basis for proving if the hypothesis is correct. Preparing the collected data was done by exporting the survey data into excel Exploration consisted of reading through all of the surveys to get an idea what the general idea of the survey results. The results were automatically tabulated after the surveys are taken. This allowed each question total to be seen, and the overall results were viewed at a glance. Abbreviations or code names were then created; codes made it easier to analyze the data. In addition to coding the data, labels were assigned to the codes. The information was grouped into themes. Excel and StatPlus was used to further analyze the data, which transfered over into representing the data. The data was represented by creating graphs, tables and figures. Validating the data is described below in the results section. The survey results were evaluated for correlations between the changes made by the IT staff members with the implementation and analysis provided by each respondent.

Measurements/Instrumentation

Correlation between two differing variables was a desired result from this survey. As a result, the different security questions asked were tested for correlation to the overall operating system security level. If the security of the operating system according to the respondents had increased the variables that influenced this result were tested for correlations. As an example if businesses in general felt that the overall computer security had improved a correlation may be drawn to the fact that the operating system was switched from one distribution of Linux to another or the computers were upgraded to the latest Linux distribution with an updated kernel and security updates. The aforementioned example applied to statistics was tested for a correlation coefficient in both direction and strength, as well as all other security enhancements that could be correlated to improved overall security (Leedy & Ormrod, 2004). Multiple correlation tests were performed to see if two or more variables are correlated. Partial correlation tests were also used to verify the consistency of variables in their correlations. What partial correlation tests do is factor out the correlation with a third variable which allows one variable to be held consistent (Leedy & Ormrod). Controlling variability was completed through partial correlation tests and an analysis of covariance.

Methodological Assumptions

It was assumed that all participants were capable of evaluating the security on their computers or the network they oversee. This assumption was made if they were in a position within the IT field and they had been in IT sufficiently long enough to be competent. The length of time that was long enough varies greatly for each person as some people gather and retain information at a faster pace than others. The fill in the blank answers also aided in the evaluation process because an incompetent person will be unable answer the questions.

Since computers are hard to predict and they have random acts take place; it was assumed that the events recorded have a cause and effect relationship. For example, if a network administrator had many issues with a systems getting DoS'ed and it stopped it could be correlated to actions that have been taken such as the installation of better endpoint protection, the application of patches, and blocking of the attackers ip address. However, each relationship was tested for correlation to verify that the two items really have a cause and effect relationship.

Also it was assumed the participants in the survey represent a good portion of the businesses who use Linux. Thus if the participants in the survey were improving security by keeping up to date with patches, installing endpoint protection such as firewalls, anti-virus, and using a sandboxing technique to protect files and program access then it could be concluded many businesses are most likely doing likewise.

Study Limitations

A survey is a method to gather information quickly but is completely dependent upon the participants' memory. The survey participants may have been unable to recall specific events while taking the survey. Some participants may have been unable to improve security or the results may be so minuscule that it is hard to record the results in the survey. The information acquired from surveys may not be enough evidence to prove that security had improved. Lack of rock solid evidence occurred because the hard core evidence is seen over a length of time and can go unnoticed until a realization hits that past security problems have gone away.

Validity and Reliability

Valid information stems from setting proper controls. One of the methods to ensure validity is to limit one survey per public IP address. This control has verified that one organization has not filled out the survey more than once. After gathering the information the data needs to be validated. One method employed was the use of multiple methods to confirm that the variables are highly correlated. Also since the survey results came back from many experts in the field of information technology then the information is more likely reliable data. An expert in the field gains their expertise through experience (Expert, 2006). For the purposes of this thesis an expert had a good mixture of time in the field, as well as a degree or certificate to go along with their experience. However, it is not necessary to have a degree or certificate if they have been in the field of information technology at least 5 years. A high correlation between time in the field of information technology, and understanding of the topics in the survey confirmed if the survey takers were experts. It was also possible that there were different levels of experience and specialists for each of the areas within the IT field who took the survey.

Reliability was also important; as a result the survey has been administered in a standardized fashion with no changes made to the survey between respondents. All Linux distributions were included in the survey results because choosing one distribution of Linux may skew the data. Test-retest and internal consistency reliability was used in testing data. Test-retest is where the same test gets the same results at different times. Internal consistency reliability is where instruments yield similar results. Bias was most likely be a part of the results because it is virtually impossible to remove all bias, but bias has been reduced as much as possible. For example all participants were unknown to the researcher which greatly aided in the removal of bias from the survey results (LearnHigher Centre for Excellence in Teaching and Learning, 2008). Bias is discussed further in the analysis section of this thesis paper. Finally any other discrepancies or negative information that is found was discussed in the analysis section.

Ethical Assurances

Confidentiality, data access and ownership, and risk assessment all need to be addressed as a part of ethics. The participants of the survey have been informed of the purpose of the study which will include the risks and benefits for being a participant in the survey (Royal District Nursing Service, 2007). Personally identifiable information has not been a part of the survey. If any of the questions result in more specification than is needed it has not been included in the survey results, especially if it would in any way identify or compromise security of any respondent. Confidentiality of trade secrets, open ports, rules, group policy or any other information disclosed in communications related to the survey has not been included in any documentation without prior consent. Any information that is included in the results has been summarized which will remove the details that could compromise the security of any business who has chosen to participate. Although information may be withheld for confidentiality purposes, the data has been analyzed and reported as accurately as possible. It is expected that the results may be used for further research and as a result the information must be as accurate as possible.

As an incentive for participation the survey results were offered to survey takers. In order to send these results to the participants their contact information was gathered in a format such as an email or a physical address. The contact information has not been used to identify the participant or company. The information was gathered apart from the survey which has aided in the separation of personal information from the survey itself. The information contained in this thesis is credited to the author in the references section below. Credit for survey participants was given, but without names of individuals or businesses.

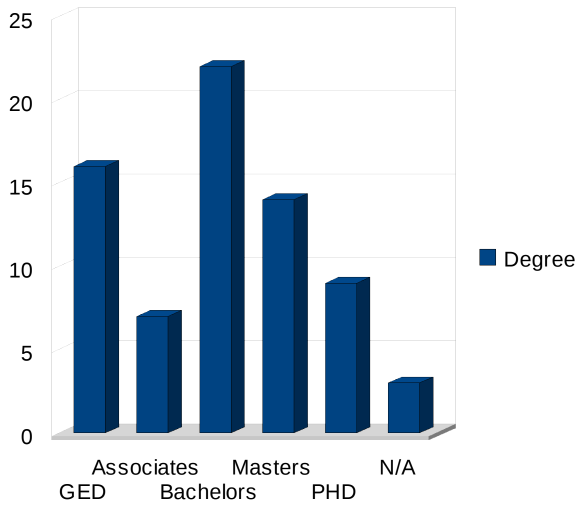
CHAPTER 4. RESULTS

71 completed survey responses were received. 70 of the respondents use Linux. This gives a response rate of 15% which is lower than average. Average online surveys typically are around 30% (IAR: Assess teaching > Response rates, 2007). Surveys with low response rates have shown insignificant differences when compared to surveys with a higher response rate. In addition bias tends to be reduced when there is a lower response rate (AAPOR | Response rates - An overview, 2007). The results are broken down into three main sections.

Demographics

Figure 2 lists the Education of the respondents. 94% have spent a total of 5 years or more in IT. 4% have spent 1-5 years in IT, 1% has never worked in IT. 58% of the respondents have spent greater than 5 years at their current employer. 28% have spent 1-5 years with their current employer, 11% less than 1 year and 3% are not employed. 97% install, upgrade or maintain

Figure 2. Level of Education

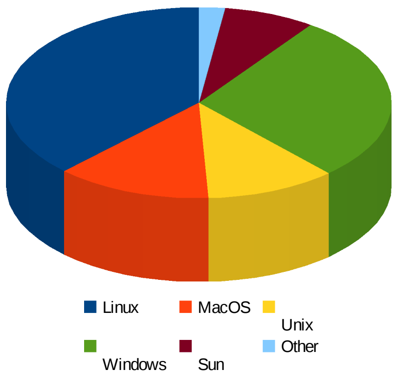


software for their organization. When asked about certificates 49% responded with a yes, 51% answered no. For those who have certificates: 17% have a Linux+, 10% have an A+ and 73% have other types of certificates.

Operating System Information

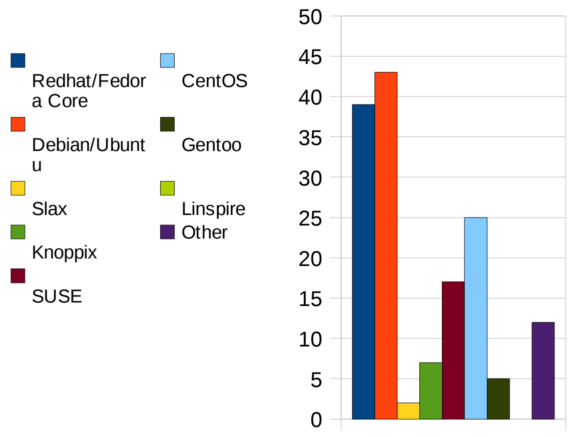
Figure 3 shows the operating systems used by the businesses surveyed. The questions about the kernel versions of Linux before and after upgrading have been thrown out mostly because 37% did not even know what version of the kernel they used before upgrading. Using

Figure 3. Operating Systems



this question would introduce inconsistencies, thus it was thrown out. Please see the discussion in the next chapter for more information about Linux kernels. In addition the version of each distribution currently in use is not listed since it was only used to gather information to better understand the background of each respondent. The distributions of Linux used by the respondents are displayed in Figure 4.

Figure 4. Linux Flavors



The statistics for those who upgraded their Linux operating systems within the past year is 79%. Patches are applied daily 13%, weekly 32%, monthly 18%, quarterly 23%, yearly 4%, and never 10% of the time.

Security layers and issues

70% of the respondents, have installed software that was not signed. 44% would continue installing software if they were in the process of installing Linux updates and received an error that the software was not signed. 56% would not install the software if they were told it was not signed. The changes in the effects of malware before and after upgrading is shown in Figure 5. Viruses affect the uptime of the Linux systems 4% quarterly, 13% yearly and 82% answered never. The frequency of buffer overflow errors is shown in Figure 6.

Figure 5. Visible Effects of Malware

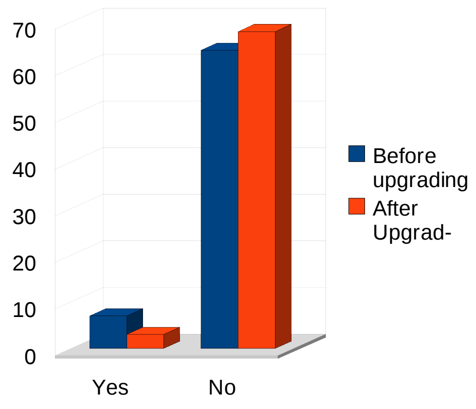
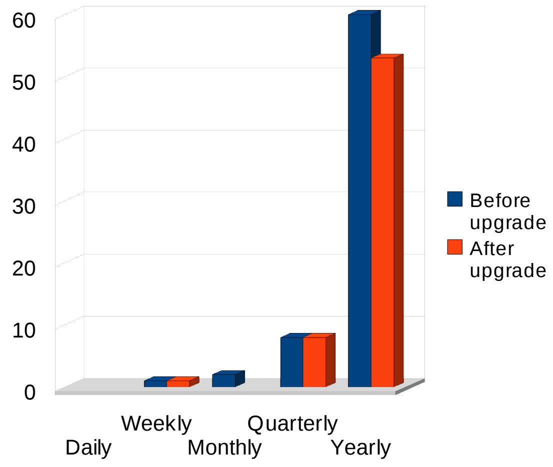
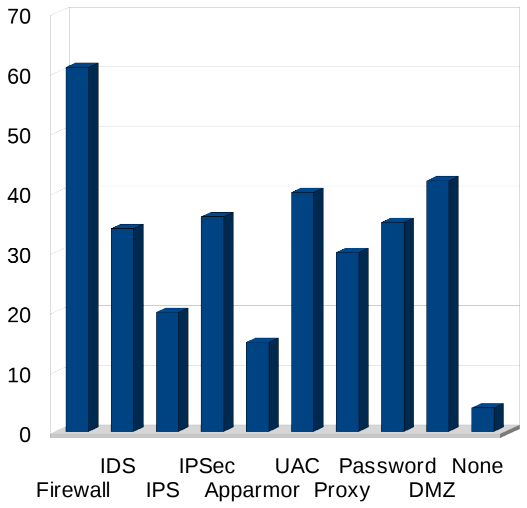


Figure 6. Frequency of Buffer Overflow Errors



The other security layers implemented on Linux systems were as follows: 72% have not turned on NSA Security-Enhanced Linux (SELinux), whereas 27% have turned SELinux on and configured it. The question about the use of the SELinux policy must not have been interpreted incorrectly. The question was vague as to what “it” was referring to because 30 people responded that they used the example policy, and 19 started from scratch to configure the SELinux policy, yet only 19 configured and turned SELinux on. As a result the question about SELinux configuration has been removed from any analysis. Role based access control (RBAC) was implemented by 8% of the respondents. The kernel crashes quarterly for 1% of the respondents, yearly for 28%, and never for 71%. The question about all other security layers was answered as displayed in Figure 7.

Figure 7. Security Layers



Variable Selection

The variables were chosen from these research questions: Has AppArmor enhanced fine grained application controls and as a result helped secure Unix based computers? Has the incorporation of SELinux into the Linux Kernel been a security enhancement? Does a hardened version of Linux lower compromises of the operating system? What other security layers have been implemented to enhance the level of security? In order for any of the analysis to be statistically significant a p-value approximating 0.05 or higher was determined to be significant for the calculations of the results.

The variables chosen for analysis are coded by the question number. The following questions were analyzed in various ways to aid in answering the research questions mentioned above. Q12 refers to upgrades of the Linux OS in the past year. Q14 refers to the frequency of patching. Q18 since upgrading have the computers been affected by viruses, worms, trojans, malware or other malicious software? Q20 refers to buffer overflow errors since upgrading. Q22 has SELinux been turned on and configured? Q24 Role Based Access Control, was it used? Q25 How often has the kernel crashed? Q26 looks to see if AppArmor was used as a application restriction. The variable held constant was (Q12) upgrades to the Linux OS in the past year.

The first test applied to these variables was the Pearson correlation coefficient matrix as seen in Table 1. This table shows the correlations that exist between Q12, Q14, Q18, Q20, Q22, Q24, Q25, and Q26. The closer the Pearson Correlation Coefficient is to 1 the more probable that there was a correlation between the two variables. Additionally at the 5% significance interval each test is written in red if the variables were accepted. Further discussion will take place in the next chapter.

Table 1

Pearson Correlation Coefficients Matrix

Sample size 71 *Critical Value(5%)* 1.9949

		Q12	Q14	Q18	Q20	Q22	Q24	Q25	Q26 Apparmor
Q12	**Pearson Correlation Coefficient**	**1**							
	R Standard Error								
	t								
	Significance Level								
	Ho (5%)								
Q14	**Pearson Correlation Coefficient**	**0.4411**	**1**						
	R Standard Error	0.0117							
	t	4.0824							
	Significance Level	0.0001							
	Ho (5%)	*rejected*							
Q18	**Pearson Correlation Coefficient**	**0.1087**	**0.0522**	**1**					
	R Standard Error	0.0143	0.0145						
	t	0.9084	0.4346						
	Significance Level	0.3668	0.6652						
	Ho (5%)	*accepted*	*accepted*						
Q20	**Pearson Correlation Coefficient**	**0.0979**	**0.0037**	**-0.0809**	**1**				
	R Standard Error	0.0165	0.0167	0.0166					
	t	0.8172	0.031	-0.6746					
	Significance Level	0.4166	0.9753	0.5022					
	Ho (5%)	*accepted*	*accepted*	*accepted*					
Q22	**Pearson Correlation Coefficient**	**0.0837**	**0.1555**	**0.188**	**0.2558**	**1**			
	R Standard Error	0.0146	0.0144	0.0142	0.0137				
	t	0.6979	1.3078	1.5901	2.1977				
	Significance Level	0.4876	0.1953	0.1164	0.0313				
	Ho (5%)	*accepted*	*accepted*	*accepted*	*rejected*				
Q24	**Pearson Correlation Coefficient**	**0.0402**	**0.0565**	**-0.0668**	**0.0962**	**0.1724**	**1**		
	R Standard Error	0.0151	0.0151	0.0151	0.015	0.0147			
	t	0.3343	0.4699	-0.5558	0.803	1.4536			
	Significance Level	0.7392	0.6399	0.5801	0.4248	0.1506			
	Ho (5%)	*accepted*	*accepted*	*accepted*	*accepted*	*accepted*			
Q25	**Pearson Correlation Coefficient**	**0.0452**	**0.0233**	**0.0107**	**0.0148**	**0.1157**	**0.2259**	**1**	
	R Standard Error	0.0151	0.0151	0.0151	0.0151	0.0149	0.0144		
	t	0.3762	0.194	0.0887	0.1227	0.9679	1.9261		
	Significance Level	0.7079	0.8468	0.9296	0.9027	0.3365	0.0582		
	Ho (5%)	*accepted*	*accepted*	*accepted*	*accepted*	*accepted*	*accepted*		
Q26 Apparmor	**Pearson Correlation Coefficient**	**0.1628**	**0.0722**	**0.3604**	**-0.1106**	**-0.0744**	**-0.0357**	**-0.1672**	**1**
	R Standard Error	0.0749	0.0765	0.0669	0.076	0.0765	0.0768	0.0748	
	t	1.3708	0.601	3.2097	-0.9242	-0.6193	-0.2969	-1.4091	
	Significance Level	0.1749	0.5498	0.002	0.3586	0.5377	0.7675	0.1633	
	Ho (5%)	*accepted*	*accepted*	*rejected*	*accepted*	*accepted*	*accepted*	*accepted*	

Some other correlation tests have been performed besides the Pearson Correlation test to see if two or more variables are correlated as described in the methodology section. Testing for correlation using different methods helps verify that the correlations truly exist. As a result the Fechner correlation coefficients matrix has also been included. The calculations were done with StatPlus using the formula r = (C-H) / (C+H). For the purpose of the analysis of the Fechner correlation matrix analysis anything over .50 will be considered correlated. Although the closer to 1 that the correlations are the higher the probability that they were correlated.

Table 2

Fechner Correlation Matrix

	Q12	*Q14*	*Q18*	*Q20*	*Q22*	*Q24*	*Q25*	*Q26 Apparmor*
Q12	1							
Q14	0.2222	1						
Q18	-0.4722	0.0833	1					
Q20	-0.2222	0.1111	0.4167	1				
Q22	-0.1667	0.1667	0.4722	0.3889	1			
Q24	-0.3611	0.1389	0.6667	0.5278	0.5278	1		
Q25	-0.1389	0.1389	0.3333	0.25	0.3056	0.4444	1	
Q26 Apparmor	0.0278	0.0833	0.0556	0.3056	0.1944	0.2778	0.2778	1

An analysis of covariance (ANCOVA) has also been performed to see if the variables move together. The results of ANCOVA removes variability and gives another angle to look at the data. If the variables do move together then then the numbers will be either positive or negative but not close to zero. A positive number is because a large number of one item moves with a large number of another time. Negative numbers are from variables moving together where one variable is a small group and the other variable is a large group.

Table 3

Analysis of Covariance

	Q12	*Q14*	*Q18*	*Q20*	*Q22*	*Q24*	*Q25*	*Q26 Apparmor*
Q12	0.1844							
Q14	0.3083	2.2081						
Q18	0.0413	0.0963	0.0924					
Q20	-0.0388	-0.1466	-0.0121	0.6022				
Q22	0.0202	0.1492	0.0157	0.206	0.2365			
Q24	0.0107	0.0809	-0.005	0.1044	0.0422	0.1315		
Q25	0.0271	0.1878	-0.0025	0.3278	0.0863	0.1865	0.702	
Q26 Apparmor	0.0195	-0.3516	-0.0586	-0.0195	0.2344	-0.0195	0.2539	1.4648

Conclusion

There was a 15% response rate which is not bad for an online survey. There was a 99% usage of Linux. This was because the respondents that were sought out needed to use Linux in a production environment. The demographics, operating system information and security layers all were reported in this section. In addition an analysis of Q12, Q14, Q18, Q20, Q22, Q24, Q25, and Q26 was performed. The analysis consisted of a Pearson correlation, and a Fechner correlation. The correlations between variables were really low but not low enough for a majority of the correlations to be rejected. Additionally to remove variability an analysis of covariance was performed which checked to see which variables moved together.

CHAPTER 5. DISCUSSION, IMPLICATIONS, AND RECOMMENDATIONS

The demographics for the survey respondents showed that the respondents are experts because they have 5 or more years in the field of Information Technology (the methodology section defined this level as an expert). Thus the respondents were able to provide good feedback because of their status. Overall it looks as if the computer systems that run Linux at these locations are very secure. It may be that bias was introduced since the survey was basically a self analysis and the respondents believed their Linux environment to be secure. It could also have been lack of knowledge about security incidents that may have occurred on their Linux computers.

Noticeable trends from the results were that minimal changes occurred after upgrading with relationship with viruses, kernel crashes and buffer overflow errors. There could be many reasons for there to have been little improvements from upgrading. Some of the unknown variables are the firewall rules, and the location of the computers in relation to the Internet. Not knowing where the computers were in relation to the Internet as well as what firewall rules they have make it hard to make an educated guess on why the system administrators believe their Linux computers to be so secure. One factor of increased security could be that a majority (45%) of the respondents apply patches on a daily or weekly basis.

Some other important information received from experts in the field helped clarify why upgrading the kernel is irrelevant. This is because the fixes are backported to old versions of the kernel. Meaning that the long term Linux distribution releases get security patches backported to them. This means that the long term releases typically have less bugs because they run on older kernels, yet they also get the security patches to keep them up to date. As a result of this information the kernel questions have not been included in the analysis.

The correlations between the variables were very low. For the purposes of this analysis a Pearson correlation coefficient of .50 or greater was highly correlated. The closer that the Pearson correlation was to one the stronger the correlation between the two variables. However, none of the correlations came close to one. This is because they may be correlated but they do not have a cause and effect relationship. Stockburger, gives clarity on this subject by stating that “Calculators and computers will produce a correlation coefficient regardless of whether or not the numbers are 'meaningful' in a measurement sense” (2006, ¶ 34). Looking at the survey results and the analysis data gives a better idea of what may be going on. The questions about buffer overflow errors had a minuscule change from before upgrading and after upgrading. This could be because outside of a controlled environment this is not monitored. Actually remembering and linking buffer overflow errors to the events that occurred on their Linux computers could be more difficult than should be required for a survey. Especially since the administrators may not have seen these events occur on the computer. Malware before (Q17) and after (Q18) upgrading also had little change, the little change could have been for many reasons. An unknown variable is what anti-virus solution was currently implemented and how up to date the definitions are kept. It could be that the system administrators did not know the system had a virus. Resident virus protection does not exist in Linux; meaning that a scan does not run automatically unless scheduled as a cron job. The system administrator may just have seen quirky things occur on the system which may have gone away or the computer was re-imaged.

The Fechner correlation matrix showed that RBAC (Q24) is highly correlated to a decrease in computers that are affected by viruses, worms, trojans, malware or other malicious software (Q18). RBAC can limit the effects malware on a computer because malicious software can have free reign on a computer that has root or administrative privileges. Linux prevents this by nature because many distributions require even a user with root privileges to enter the credentials before privileges can be elevated. RBAC can place further limitations on what effects malware can have through the separation of duties. Similarly RBAC (Q24) is also highly correlated to buffer overflow errors since upgrading (Q20). This correlation could mean that the use of RBAC can reduce the amount of buffer overflow errors that occur on a Linux machine. The Fechner correlation also shows that RBAC (Q24) and SELinux (Q22) were highly correlated. The correlation between RBAC and SELinux could mean that if the two layers were both used and configured that there would be an increased level of security on the computers.

The ANCOVA test showed that upgrades (Q12) and frequency of patching (Q14) tend to move together. The fact that upgrades and patching move together could mean that the more upgrades and patching that takes place tends to improve overall security. AppArmor (Q26) and frequency of patching (Q14) also move together in a negative way meaning that small values of Q26 move with large values of Q14. The results show that only 21% of the respondents actually use AppArmor which is the reason why there is a negative relationship between AppArmor and Q14.

Implications

Overall it appears as if patching the Linux OS and the implementation of security layers helps increase security. The null hypothesis is not rejected for most of the variables. The factors in increased security include the frequency of patching, the implementation of SELinux, Role Based Access Control such as PaX and GRsecurity, and the use of AppArmor. The implementation of these controls, with the exception of AppArmor, were correlated with the decrease in the effects viruses, worms, trojans, malware or other malicious software could have on a Linux computer.

The information that was seen as correlated is not enough proof to solidly state that security has improved because of one effect or another. The answers to the research questions definitely have been addressed and can be further addressed in by performing more focused research. AppArmor was only implemented on few computers (21%) which made it hard to tell if it had much of any effects on the Linux computers. AppArmor may have a relationship with patching which means that patched systems which use AppArmor could be more secure. Yet AppArmor was specifically rejected from any correlation to malware after upgrading. SELinux and RBAC may enhance security on the overall Linux system especially when used together because they were shown to be correlated. Buffer overflow errors and the frequency of kernel crashes can be reduced through the implementation of RBAC, and patching. SELinux did not show that it would reduce the number of buffer overflow errors after upgrading.

Recommendations

A more refined research in a controlled environment would make it easier to view the effects of certain variables. Another idea is to pick a small focus group of system administrators and monitor the way their Linux computers function with patching, the implementation of SELinux, Role Based Access Control such as PaX and GRsecurity, and the use of AppArmor. This would allow the researcher to actively see what effects each layer has on the overall system. This type of research would make it possible to keep some variables constant while making changes to the other variables. This would be an improvement upon the research done for this thesis, especially since the respondents memory was put to the test while taking the survey. Actually remembering events that occurred for up to a year can be difficult to do especially without proper documentation to aid in the process of remembering exactly what happened.

Further research could look deeper into the effects on security when access controls are implemented. Such questions for exploration could be why AppArmor, SELinux or RBAC are used so little. An even deeper question could be used to focused not on the controls themselves but rather the item that security helps maintain. That is to say data, and how often information gets lost. Doing research on data leakage will be a challenge because it is not desirable for businesses to give out that type of information unless it is absolutely necessary. This also may be the reason that so many results came back looking as if the companies networks are perfectly secure. A highly secure computer is something very desirable but it is more likely that some information has been lost due to a virus, improper firewall rules, or another vulnerability either in the network or computer.

Conclusions

It takes many layers to protect a network, its devices and all of the computers from unwanted invasions. This research focused solely on Linux operating systems and looked at these systems from the system administrators perspective. Although no rock solid evidence was found to state that one device or application keeps the computer secure it can be said that those who were surveyed did believe their networks to be secure. The security applications and techniques used to secure these Linux computers and their surrounding networks were: patching, firewalls, IDS, IPS, AppArmor, SELinux, RBAC, IPsec, UAC, proxy server, complex password policies, and DMZ.

The stronger correlations found between layers and the minimized effect of security intrusions was AppArmor although it has been implemented on few computers it may have a relationship with patching which means that patched systems that use AppArmor could be more secure. It could be said that AppArmor enhances fine grained permissions and help secure computers that run Linux. SELinux and RBAC also enhance security on a Linux system. Buffer overflow errors and the frequency of kernel crashes can be reduced through the implementation of RBAC, and patching. The use of RBAC, AppArmor, or SELinux are a part of hardening a Linux system which in turn lowers the compromises of the operating system.

The information gathered in this survey provides a good basis of information that has not been documented which will provide a good basis for further research. As such this research is a stepping stone for a further in an depth analysis of access controls. In fact it might be possible to use the same control group using other testing methods which would further improve this thesis and give a much clearer picture of these Linux security layers and their implementations.

REFERENCES

AAPOR. (2007). *Response rates: An overview*. Retrieved July 25, 2009, from http://www.aapor.org/responseratesanoverview

Absolute Software. (n.d.). *Protecting data on laptops: Why encryption isn't enough*. Retrieved March 12, 2009, from http://www.webbuyersguide.com/Resource/ResourceDetails.aspx?id=11431

Aitoro, J. (2008, August 11). *Top IT cops say lack of authority, resources undermine security.* Retrieved September 27, 2008, from http://www.govexec.com/story_page.cfm?filepath=/dailyfed/0808/081108j2.htm

Anonymous. (2005). FTC: 'Reasonable & appropriate' measures to protect digital assets. *International Journal of Micrographics & Optical Technology, 23*(4), 7. Retrieved September 29, 2008, from ProQuest Computing database.

AppArmor - openSUSE. (n.d.). *AppArmor.* Retrieved November 6, 2008, from http://en.opensuse.org/Apparmor

Bardram, J. (2005). The trouble with login: On usability and computer security in ubiquitous computing. *Personal and Ubiquitous Computing, 9*(6), 357-367. Retrieved October 2, 2008, from ProQuest Computing database.

Baskerville, R. (1993). Information systems security design methods: Implications for information systems development. *ACM Computing Surveys*, *25*(4), 375. Retrieved September 30, 2008, from Research Library Core database.

Bowman, I. (1998, January). *Conceptual architecture of the Linux kernel*. Retrieved October 24, 2008, from http://docs.huihoo.com/linux/kernel/a1/index.html

Bovet, D., & Cesati, M. (2005). *Understanding the Linux kernel*. Sebastopol, CA: O'Reilly Media, Inc.

Bunnell, M., & Weinberg, W. (1996, December). Kernel modules tailor small-OS solution. *Electronic Engineering Times 932*, 76-78. Retrieved September 29, 2008, from ProQuest Computing database.

Center for Internet Security, The: (2008, September). The center for internet security announces industry's first consensus based metrics for information security. *Internet Business Newsweekly*, 14. Retrieved September 24, 2008, from Sciences Module database.

Cherry, M. & Imwinkelried, E. (2008). Internet theft is avoidable. *Judicature, 92*(1), 7. Retrieved September 30, 2008, from Law Module database.

Chuvakin, A. (2002, January 23). *Linux Kernel Hardening*. Retrieved March 4, 2009, from http://www.securityfocus.com/infocus/1539

Clark, V., & Creswell, J. (2006). *Designing and conducting mixed methods research*. Thousand Oaks, CA: Sage Publications, Inc.

Danchev, D. (2008, July 7). Approximately 800 vulnerabilities discovered in antivirus products. Posted to http://blogs.zdnet.com/security/?p=1445

De Groot, D. (2006). *Computer security: Article on computer security*. Retrieved October 2, 2008, from http://bluefive.pair.com/articles_computer_security.htm

Device management | Channel 9. (n.d.). *Microkernel vs Macrokernel (aka Monolithic kernel).* Retrieved June 17, 2009, from http://channel9.msdn.com/wiki/devicemanagement/

Documentation for the PaX project. (2006, April 26). *Aslr.* Retrieved June 16, 2009, from http://pax.grsecurity.net/docs/aslr.txt

Documentation for the PaX project. (2006, April 26). *Pax.* Retrieved June 16, 2009, from http://pax.grsecurity.net/docs/pax.txt

Expert. (2006, December 14). *Online Etymology Dictionary*. Retrieved August 11, 2009, from Dictionary.com website: http://dictionary.reference.com/browse/expert

Frazier, R. (n.d.). *Security frameworks*. Retrieved November 3, 2008, from www.hackerz.ir/e-books/127%20Security%20Frameworks.pdf

Fruhwirth, C. (2006, February 18). *LUKS on disk format specification*. Retrieved November 8, 2008, from luks.endorphin.org/LUKS-on-disk-format.pdf

Garfinkel, S. (2005). *DSpace@MIT : Design principles and patterns for computer systems that are simultaneously secure and usable*. Retrieved February 7, 2009, from http://dspace.mit.edu/handle/1721.1/33204

Gay, L. R., & Airasin, P. (2003) *Educational research: Competencies for analysis and application* (7th ed.). Upper Saddle River, NJ: Merrill/Prentice Hall.

Germain, J. (2005, February 19). *Computer security comes of age*. Retrieved September 23, 2008, from http://www.technewsworld.com/story/40686.html?wlc=1222693994

Granneman, S. (2003, October 2). Linux vs. Windows viruses. Posted to http://www.securityfocus.com/columnists/188

Halkidis, S., Tsantalis, N., Chatzigeorgiou, A., & Stephanides, G. (2008, July - September). Architectural risk analysis of software systems based on security patterns. *IEEE Transactions on Dependable and Secure Computing, 5*(3), 129. Retrieved October 1, 2008, from ProQuest Computing database.

Hamid, S. (2008, April 16). *How to login to Ubuntu as root user?* Retrieved April 2, 2009, from http://www.sizlopedia.com/2008/04/16/how-to-login-to-ubuntu-as-root-user/

Harrington, J. (2005). *Network security: A practical approach.* San Francisco: Morgan Kaufmann.

Holford, J. W. (2006). *The concept of self-defending objects and the development of security aware applications*. Retrieved March 12, 2009, from eprints.qut.edu.au/16227/2/02whole.pdf

Hsu, C. & Backhouse, J. (2002). Information systems security education: Redressing the balance of theory and practice. *Journal of Information Systems Education, 13*(3), 211-218. Retrieved October 2, 2008, from Education Module database.

IAR. (2007, July 16). *Response rates.* Retrieved July 25, 2009, from http://www.utexas.edu/academic/diia/assessment/iar/teaching/gather/method/survey-Response.php

Indiana University. (2003, February 26). The *UNIX system kernel.* Retrieved October 11, 2008, from http://www.uwsg.iu.edu/usail/concepts/

IT Governance Institute. (2007, October 14). *COBIT 4.1 executive summary and framework.* Retrieved February 11, 2009, from http://www.isaca.org/AMTemplate.cfm?Section=Downloads&Template=/ContentManagement/ContentDisplay.cfm&ContentID=34172

Jones, M. (2008, April 29). *Anatomy of security-enhanced Linux (SELinux)*. Retrieved November 6, 2008, from http://www.ibm.com/developerworks/linux/library/l-selinux/index.html

Karp, A. (2003). Enforce POLA on processes to control viruses. *Association for Computing Machinery. Communications of the ACM*, *46*(12), 27-29. Retrieved September 29, 2008, from Research Library Core database.

Kerner, S. (2008, October 29). *InternetNews realtime IT news - security problem? Blame the human element*. Retrieved October 31, 2008, from http://www.internetnews.com/security/article.php/3781426/Security+Problem+Blame+the+Human+Element.htm

Kil, C., Jun, J., Bookholt, C., & Xu, J. (n.d.). *Address space layout permutation*. Retrieved October 31, 2008, from http://discovery.csc.ncsu.edu/pubs/acsac06a.pdf

Krebs, B. (2006, July 19). *Hacked ad seen on MySpace served spyware to a million - security fix*. Retrieved October 4, 2008, from http://blog.washingtonpost.com/securityfix/2006/07/myspace_ad_served_adware_to_mo.html

Kroah-Hartman, G. (2009, February 12). *ChangeLog-2.6.28.5*. Retrieved June 17, 2009, from kernel.org/pub/linux/kernel/v2.6/ChangeLog-2.6.28.5

Lampson, B. (2004, February 25). *Computer security in the real world*. Retrieved October 2, 2008, from research.microsoft.com/lampson/64-SecurityInRealWorld/Acrobat.pdf

LearnHigher Centre for Excellence in Teaching & Learning. (2008) Learning to analyze quantitative data. Retrieved April 2, 2009, from http://www.learnhigher.ac.uk/analysethis/main/quantitative1.html

Leedy, P. D., & Ormrod, J. E. (2004). *Practical research: Planning and design (8th Edition)*. Alexandria, VA: Prentice Hall.

Lettice, J. (2004, October 22). *Windows v Linux security: The real facts.* Retrieved September 28, 2008, from http://www.theregister.co.uk/2004/10/22/linux_v_windows_security/

LinuxDevices.com (2008, May 2). *Linux-friendly microkernel OS tightens mobile security*. Retrieved November 5, 2008, from http://www.linuxdevices.com/news/NS3442674062.html

LinuxDevices.com. (2007, November 30). *Virtualization microkernel supports ARMv6*. Retrieved November 5, 2008, from http://www.linuxdevices.com/news/NS8552513536.html

Local Program Evaluation in Tobacco Control. (2003, September 25). *Sample Size Selection Chart.* Retrieved July 23, 2009, from http://www.uwex.edu/ces/tobaccoeval/resources/surveychart.html

Loscocco, P., & Smalley, S. (2004, July 19). *Meeting critical security objectives with security-enhanced Linux*. Retrieved November 3, 2008, from www.nsa.gov/selinux/papers/ottawa01.pdf

Loscocco, P., Smalley, S., Muckelbauer, P., Taylor, R., Turner, S., & Farrell, J. (1998, October). *The inevitability of failure: The flawed assumption of security in modern computing environments*. Retrieved November 3, 2008, from http://www.cs.utah.edu/flux/fluke/html/inevitability.htm

MacIver, D. (2008, February 25). System integrity team blog: Protecting BitLocker from cold attacks (and other threats). Posted to http://blogs.msdn.com/si_team/archive/2008/02/25/protecting-bitLocker-from-cold-attacks-and-other-threats.aspx

Milberg, K. (2004, January 19). *Commentary: Addition of IPsec locks down 2.6 kernel*. Retrieved May 14, 2009, from http://searchenterpriselinux.techtarget.com/news/article/0,289142,sid39_gci944836,00.html

National Security Agency. (2007, August 27). *Security-enhanced Linux*. Retrieved November 3, 2008, from http://www.nsa.gov/research/selinux/index.shtml

National Security Agency. (2009, January 15). *Security policy abstractions*. Retrieved January 15, 2009, from http://www.nsa.gov/research/selinux/policy.shtml

Noyes, A. (2008, October 15). *NextGov - FBI warns of sweeping global threat to U.S. cybersecurity*. Retrieved October 16, 2008, from http://www.nextgov.com/nextgov/ng_20081015_7578.php

O'dowd, D. (2004, May 24). *Linux security: Unfit for retrofit - Green Hills software*. Retrieved October 24, 2008, from http://www.ghs.com/linux/unfit.html

O'Reilly, T., & Raymond, E. (2001). *The cathedral & the bazaar: Musings on Linux and open source by an accidental revolutionary.* Sebastopol, CA: O'Reilly.

Parker, D. (2008, March 12). *Catch them if you can.* Retrieved June 5, 2008, from http://www.securityfocus.com/columnists/468

Perens, B. (1998, July 20). Slashdot Feature: Security through obscurity. Retrieved January 22, 2009, from http://slashdot.org/features/980720/0819202.shtml

Perrin, C. (2007, July 22). *Myth: I'm not really at risk. IT Security*. Retrieved October 4, 2008, from http://blogs.techrepublic.com.com/security/?p=259

Policy Kit Library Reference Manual. (2008, January 16). *PolicyKit model*. Retrieved March 12, 2009, from http://hal.freedesktop.org/docs/PolicyKit/model.html

Redhat (2003, April 8). *Administrative controls*. Retrieved March 28, 2009, from www.redhat.com/docs/manuals/linux/RHL-9-Manual/security-guide/s1-wstation-privileges.html

Rothman, M. (2007, August 27). *Frameworks just part of security plan*. Retrieved November 3, 2008, from http://searchcio-midmarket.techtarget.com/news/column/0,294698,sid183_gci1269420,00.htm

Royal District Nursing Service. (2007). *Guidelines for researchers seeking ethics approval for research projects*. Retrieved March 27, 2009, from www.rdns.com.au/research_and_innovation/Assets/REC%20Guidelines%20for%20researchers.pdf

Ranum, M. (2005, September 1). *The six dumbest ideas in computer security.* Retrieved October 2, 2008, from http://www.ranum.com/security/computer_security/editorials/dumb/index.html

Raymond, E. (2003). *The art of UNIX programming*. New York: Addison-Wesley Professional.

Runesson, M. (2006, May 21). *AppArmor*. Retrieved November 6, 2008, from https://wiki.ubuntu.com/AppArmor

Ruoho, C. (2007, October 30). Leopard's sandbox feature. Posted to http://www.laconicsecurity.com/leopards-sandbox-feature-just-for-play.html

Secure by design (2006, October 15). *How Guardian Digital secures EnGarde secure Linux.* Retrieved October 21, 2008, from http://www.engardelinux.org/doc/other/wmes/wmes.html

Security Solutions. (n.d.). *Layered Model.* Retrieved April 2, 2009, from softwaresecuritysolutions.com/PDF/layeredModel062308.pdf

Shacham, H., Page, M., Pfaff, B., Goh, E., Modadugu, N., & Boneh, D. (2004, October 25). *On the effectiveness of address-space randomization*. Retrieved March 12, 2009, from www.stanford.edu/~blp/papers/asrandom.pdf

Shostack, A., & Stewart, A. (2008). *The new school of information security*. New York: Addison-Wesley Professional.

Singel, R. (2008, April 9). Zombie computers decried as imminent national threat. Posted to http://blog.wired.com/27bstroke6/2008/04/zombie-computer.html

Singh, A. (2004, June). *Sandboxing*. Retrieved November 8, 2008, from http://www.kernelthread.com/publications/security/sandboxing.html

Stockburger, D. W. (2006, July 6). *Correlation*. Retrieved July 27, 2009, from http://www.psychstat.missouristate.edu/introbook/sbk17m.htm

Sundaram, R. (n.d.). *Security/Features – FedoraProject.* Retrieved March 12, 2009, from http://fedoraproject.org/wiki/Security/Features

Taylor, J., & Nusca, A. (2008, October 21). Zombie PCs: Time to infection is less than five minutes. Posted to http://blogs.zdnet.com/gadgetreviews/?p=441

Tresys Technology. (2007, March 17). *SELinux policy server*. Retrieved November 6, 2008, from oss.tresys.com/projects/policy-server

Venema, W. (1998, December 5). *Murphy's law and computer security*. Retrieved October 3, 2008, from http://insecure.org/stf/wietse_murphy.html

Warren, S. (2002, March 5). Would you hire a hacker to help protect your network? Posted to http://articles.techrepublic.com.com/5100-10878_11-1039747.html?tag=sc

Whoriskey, P. (2008, April 8). *Every click you make - washingtonpost.com*. Retrieved Oct. 16, 2008, from http://www.washingtonpost.com/wp-dyn/content/article/2008/04/03/AR2008040304052.html

Williams, D. (2007, August 12). iTWire - Hardening Linux. Posted to http://www.itwire.com/content/view/13976/53/

Woods Hole Oceanographic Institution. (2007, November 21). *CIS.* Retrieved November 25, 2008, from http://www.whoi.edu/cis/security/news/firewall-annc.html

Xu, D. & Nygard, K. (2006). Threat-driven modeling and verification of secure software using aspect-oriented petri nets. *IEEE Transactions on Software Engineering, 32*(4), 265-278. Retrieved September 30, 2008, from Sciences Module database.

Yodaiken, V. (2004, July 19). *Guest editorial: Thoughts on secure operating systems*. Retrieved October. 24, 2008, from http://www.linuxdevices.com/articles/AT6311679886.html

Zhang, C. (2006). *DSpace@MIT : Designing security into software*. Retrieved January 31, 2009, from http://dspace.mit.edu/handle/1721.1/35098

APPENDIX A. DATA COLLECTION INSTRUMENT

The questions asked in a survey to businesses who use Linux are:

Questions:	Answer choices:
How many years have you worked in IT?	• Never • Less than 1 year • 1-5 years • Greater than 5 years
How long have you been at your current place of employment?	• Not Currently Employed • Less than 1 year • 1-5 years • Greater than 5 years
Do you install, maintain or upgrade software for your organization?	• Yes • No
What is the highest level of education have you achieved?	• High School Diploma or GED • Associates • Bachelors • Masters • PHD • N/A
Are any of your degrees in IT?	• Yes • No
Do you have any certifications?	• Yes • No
Please select which certifications you have attained.	• Linux+ • A+

	• Security+ • CISSP • SSCP • Other
Which operating systems are currently in use on the network?	Select all that apply: • Linux • MacOS • Unix • Windows • Sun • Other
What kernel version(s) of the Linux operating system is/are currently deployed?	• 2.2.26 • 2.4.37 • 2.6.27 • 2.6.28 • 2.6.29 • Other
What distributions of Linux do you use?	Please select all that apply: • Redhat/Fedora Core • Debian/Ubuntu • Slax • Knoppix • SUSE • CentOS • Gentoo • Linspire • Other

Which version of each distribution do you use?	Fill in the blank
Have any of the Linux operating systems at your organization been upgraded in the past year?	• Yes • No
What was the Kernel version before the upgrade?	Please choose the closest match: • 2.2.26 • 2.4.37 • 2.6.27 • 2.6.28 • 2.6.29 • Other
How often do you patch the Linux software?	• Daily • Weekly • Monthly • Quarterly • Yearly • Never
Linux updates are signed which protects a computer from installing malicious software. Have you ever installed software that wasn't signed?	• Yes • No
If you were in the process of installing Linux updates and you got an error that the software wasn't signed, would you continue with the installation?	• Yes • No
Before upgrading have any Linux computers on your network been affected by viruses, worms, trojans, malware or other malicious software within the past year?	• Yes • No
After upgrading have any Linux computers on your network been affected by viruses, worms, trojans, malware or other malicious software	• Yes • No

within the past year?	
How often do viruses affect the uptime of your systems?	• Daily • Weekly • Monthly • Quarterly • Yearly • Never
How often since an upgrade have there been buffer overflow errors such as a memory exception within the past year?	• Daily • Weekly • Monthly • Quarterly • Yearly
How often did buffer overflow errors occur before upgrading the Kernel within the past year?	• Daily • Weekly • Monthly • Quarterly • Yearly
NSA Security-Enhanced Linux (SELinux) was merged in Kernel 2.6 but the default setting is off. Has it been turned on and configured?	• Yes • No
Did you use the example policy or start from scratch?	• Example Policy • Started from scratch
PaX provides Role based Access Control (RBAC) one of the controls provided is prevention against stack smashing. Do you use RBAC either through PaX, GRSecurity, or a hardened version of Gentoo?	• Yes • No
How often does the kernel crash?	• Daily • Weekly

	• Monthly • Quarterly • Yearly • Never
What other security layers have been implemented either on your network or on the Linux computers specifically?	• AppArmor or application restrictions • Demilitarized Zone (DMZ) • Firewall • IPsec or other network encryption • Intrusion Detection System • Intrusion Prevention System • None • Password policy that requires complex passwords • Proxy Server • User Account Controls