Global Information Assurance Certification Paper

SECURE LINUX FIREWALL DEPLOYMENT
A STEP-BY-STEP GUIDE FOR THE SUSE LINUX 9

GCUX Practical Assignment
Version 2.0

Option 1

MARK E. DONALDSON
APRIL 11, 2004

TABLE OF CONTENTS

ABSTRACT

This paper is intended to be a technically detailed, step-by-step guide for the ground up installation, hardening, maintenance, and monitoring of a SUSE[1] 9.0 Linux Firewall running the Linux 2.4.25 kernel for a small to medium sized business. In other words, it is about securing the computer that secures and protects the systems and hosts on the network behind it. Although it is imperative that no host on a business network should ever be compromised, it is even more critical that the "guardian" of the network not be subject to tampering or penetration. It is from this prospective that I proceed.

This project is presented to fulfill the requirements of Version 2 Option 1 of the GIAC Certified UNIX Security Administrator (GCUX) practical assignment. It is my sincere hope that this work serves as both a valuable and useful contribution to the security community. It should also be noted that several Linux distributions available today could have been selected for use on this project. However, SUSE 9.0[2] has been chosen for a number of reasons that will be alluded to and discussed throughout this writing.

It should also be noted that all the software used in this project is "open source" and "freely available". With that in mind, this project intends to construct a "world-class" and "highly secure" firewall gateway at a cost not to exceed $1700.00.

[1] As a note of interest, when SUSE Linux AG announced the release of SUSE Linux 9.0 in September of 2003, they also announced a "name change" for the open source Linux distribution. Known as SuSE Linux through Version 8.2, Joseph Eckert, SUSE VP of Corporate Communications released a press statement on September 30, 2003, announcing that SuSE Linux was now SUSE LINUX. The change was deemed as a necessary step in developing "corporate identity" and to "broaden its appeal to a business audience" in their endeavor to "penetrate the North American and Australian markets". Eckert further indicated the upper casing of SUSE LINUX brought more attention to the name".

[2] SUSE 9.0 should actually be considered a "minor" revision over SuSE 8.2. Any changes are largely under-the-hood, and are not directly apparent or visible to either the end-user or the administrator. Hence, this paper can be appropriately be applied to SuSE 8.2 without reservation.

INTRODUCTION

Since 1993, the Linux Operating System has grown from a mere concept in the mind of a young and inquisitive Finnish computer science student to a full-featured and highly extensible network operating system. The SCO group's legal assault against IBM[3], launched in March of 2003, and Novell's purchase of SUSE LINUX AG in November of 2003, certify and demonstrate that the time for Linux has finally arrived. Linux, indeed, must now be taken seriously and regarded as a true force today in the world of enterprise computing[4]. An obvious question might be "Why has this happened now?" An obvious answer might well then be "because if properly installed and configured, Linux can easily withstand any and all of the punishment the Internet can deliver". Such is the fundamental premise of this project, and the premise upon which this written guide is based.

This step-by-step guide is about installing, configuring, and hardening an Internet facing SUSE 9 Linux firewall using netfilter/iptables. For organizational purposes and readability reasons, the paper has been divided into four general sections:

- Server Specification and Risk Mitigation Plan
- Installation and Hardening
- Server Maintenance and Monitoring
- Testing and Verification

Additionally, some of the sections have been divided into subsections. Although each section or subsection in itself may serve as a stand-alone logical and procedural step-by-step instruction for a specific element, this paper will attempt to integrate all elements into a complete and definitive guide for the securing, hardening, monitoring, and maintaining of a SUSE 9 Linux firewall.

Finally, 18 appendices are included at the end of the paper. Each appendix provides additional detail related to the system configuration or hardening process, or scripts that help enable the security process.

We begin by developing a specification for the Linux firewall.

---

[3] Although not surprising, evidence is now surfacing that suggests Microsoft Corporation may be secretly assisting the SCO Group in their attempt to squelch the Linux open source operating system. Until now, Microsoft has not publicly acknowledged that Linux is a threat. This may be about to change.

[4] An interesting read is the "The Story of the Linux Kernel", by Linus Torvalds himself. The article, reproduced by LinuxWorld.com, with the permission of O'Reilly & Associates, is excerpted from the O'Reilly Book Open Sources: Voices from the Open Source Revolution printed in 1999.

PART 1: SPECIFICATION & RISK MITIGATION

SERVER SPECIFICATION

I. Server Role

The Linux Firewall used in this guide provides gateway functionality and first-line security defense for a small to mid-sized business that requires secure and reliable Internet access, both for its customers and its employees. The network behind the firewall is divided into three segments consisting of a highly protected internal Windows Server 2003 AD network segment on which the company employees reside, a service network segment that provides secure Internet services, and a highly secure administrative network segment. A generalized view of the network layout and architecture is shown in **Figure 1 "Network Architecture Overview"**.

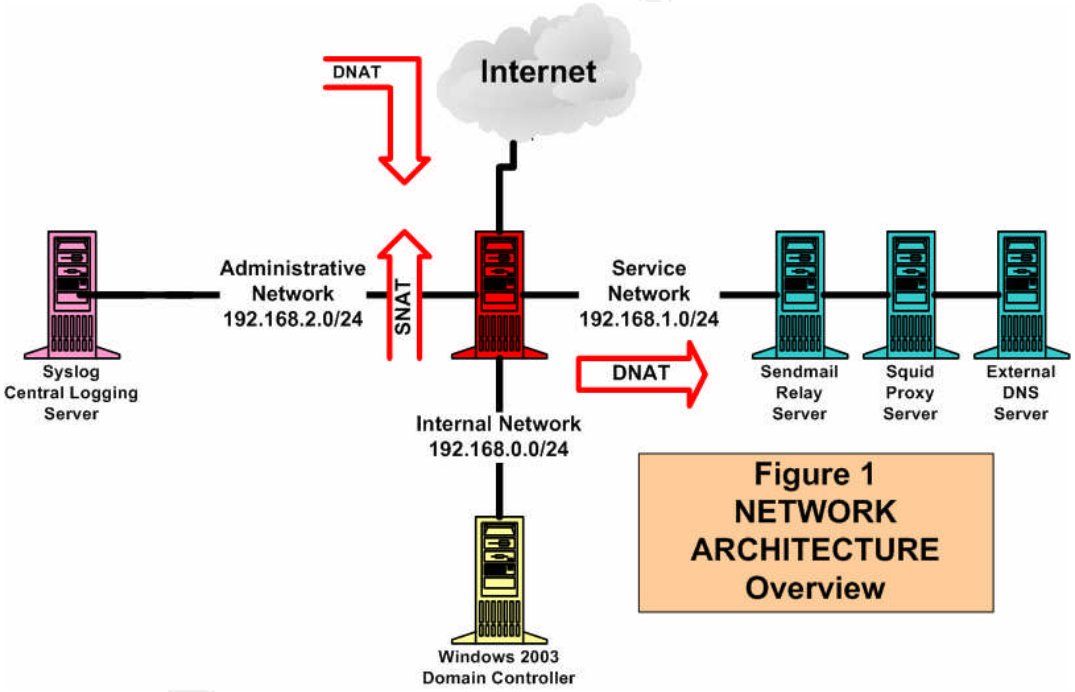


Figure 1 NETWORK ARCHITECTURE Overview

Since the company of concern has a limited number of routable IP addresses, the Linux Firewall will be required to perform the following tasks and services:

- Provide "stateful inspection" of all packets either entering from or leaving to the internet, and all packets traversing between the various network segments.

- Provide DNAT[5] services for inbound Internet connections on the service network segment.

- Provide SNAT[6] services for outbound Internet connections from either the internal or service network segments.

- Permit SMTP (port 25), HTTP and SHTTP (ports 80 and 443), and DNS and RNDC (ports 53 and 953) transactions, both inbound and outbound.

- Permit both Squid web proxy cache (port 3128) and reverse web proxy transactions.

- Permit secure network administration and monitoring.

- Block all remaining traffic.

II. Hardware Requirements

The following components and component specifications are required for the Linux Firewall:

- **Mainboard:** Intel D845PESVL Socket 478 **ATX**
- **Processor:** Intel Pentium 4 2.5 GHz 400 FSB
- **Memory:** Crucial DDR RAM 1024 MB
- **Network Interfaces:** Four (4) Intel 10/1000 Mbps PCI Ethernet Server Adapters
- **Hard Disks:** One (1) Maxtor 160 GB IDE
- **Media:** Sony CDR/DVD ROM Drive
- **UPS:** APC Smart UPS 1000
- **Floppy Disk Drive:** None (**RMM:F:1**)

III. Software Requirements

SUSE 9.0 on removable has been selected for the firewall operating system. The following software programs and packages will be installed from the original SUSE 9 installation media:

- Netfilter: POM patched to Version 20031219
- Iptables: Upgraded to Version 1.2.9
- Sendmail: Version 8.12.10
- OpenSSH: Version 3.7
- OpenSSL: Version 0.9.7

---

[5] DNAT stands for destination network address translation. Although there are several DNAT variations, it is most typical used wherein the destination IP address is translated from a Internet "routable" IP address to a "non-routable" RFC 1918 private IP address on a private network.

[6] SNAT is source network address translation. SNAT is typically used to translate a "non-routable" RFC 1918 private IP address on a private network to "routable" IP address on packets bound for the Internet.

- NTP: Version 4.2.0
- Tripwire ASR: Version 1.3.1
- SUDO: Version 1.6.7
- John the Ripper: Version 1.6
- TCP Wrappers: Version 7.6
- TCPDump: Version 3.8.1
- LIBPCAP: Version 0.8.1
- NTOP: Version 3.0
- IPTraf: Version 2.7.0
- GCC: Version 3.3.2[7]

Additionally, the following third-party software programs, not available on the installation media, will be installed from compressed archives:

- SmartMonTools: Version 5.30.0
- APC PowerChute Business Edition PBEAgent: Version 6.2.2
- mkCDrec: Version 0.7.8
- mkCDrec Utilities: Version 0.7.5
- PortSentry: Version 1.1
- LogSentry: Version 1.1.1
- HostSentry: Version 0.02
- Snort: Version 2.1.1
- Root Kit Hunter: Version 1.05
- Proprietary Hardening and Forensic Scripts

IV. Services and Daemons

The following services and daemons will be available on the SUSE Linux firewall, and thus their respective ports must be left open on the firewall itself:

- XNTPD: UDP/123
- SSHD: TCP/22
- NTOP: TCP/3000:3001
- Syslogd: UDP/514
- APCPCNS TCP/3052

V. User Access and Access Control

Access to the available services on the SUSE Linux firewall will be restricted to two authorized firewall administrators. Each firewall administrator will have a personal account on the system, and only these administrators will be granted system logon

[7] This is a temporary installation only. It is fundamentally a bad idea to have a compiler installed on a security sensitive computer. However, the GNU compiler will be needed to configure and install several of the needed software packages, including the Linux kernel itself. Once the installation process is complete, the GCC package must be removed from the machine (**RMM:G:17**).

privileges. Access to system commands and system services will be controlled by the following means:

- Written security policy.
- The sudoers configuration file.
- Explicit firewall rulesets.

RISK MITIGATION PLAN

In accessing the risks of this SUSE 9 Linux server, two primary conditions must be given foremost consideration:

- The firewall box is directly connected to and exposed to the Internet.

- The health and viability of this business is directly dependent upon secure and reliable Internet connectivity.

As the firewall stands between the business itself and the Internet, one must conclude that the business is directly dependent on this security and reliability of this machine. Consequently, both the risks and the stakes are equally and significantly high. Thus, the security of this Linux Firewall must be taken very seriously, and any element that may limit, restrict, or deny functionality must be considered and addressed.

This risk mitigation plan will first categorize the major elements of risk, and then outline the measures or remedies that must be put into place to minimize, or at least reasonably reduce, the risk for each element (see **Table I Risk Mitigation Matrix**). The resultant plan will thus provide a set of specific working guidelines that must be applied as the system is initially constructed and configured, and subsequently operated, monitored, and maintained.

Categories of Concern

We categorize the concerns as follows:

- Physical Security
- Natural Disaster
- Hardware and Power Failure
- Software Integrity Attacks
- Software Vulnerabilities
- Password and Account Attacks
- File System and System Attacks (Environment and Privilege Elevation)
- Memory Attacks
- Malware Attacks (Trojans, Backdoors, Keyloggers, Rootkits, and Worms)
- Network Service Attacks
- Network Sniffing Attacks
- Denial of Service Attacks

Mitigation Matrix

TABLE I
RISK MITIGATION MATRIX

CONCERN	PREVENTATIVE OR MITIGATIVE MEASURES
A - Physical Security	1. Place firewall box in locked server root. 2. Restrict access to server room to firewall administrators. 3. Control server room keys issue through company security services. 4. Remove any false ceilings or floors from server room. 5. Install security camera in server room.
B - Natural Disaster	1. Bolt firewall box into position. 2. Locate server room at or above ground floor. 3. Locate server room within buildings interior (no windows please).
C - Hardware & Power Failure	1. Control server room temperature and adequately ventilate. 2. Attach firewall box to substantial UPS system. 3. Install UPS control and warning software (APC PBEAgent). 4. Install hard disk SMART analytical software (SmartMonTools). 5. Install disaster recovery software & schedule system imaging (mkCDrec). 6. Schedule regular system backups. 7. Utilize "heartbeat" remote monitoring and notification software (Mon).
D - Software Integrity Attacks	1. Install operating system with server isolated from network (unplugged). 2. Utilize dedicated "proxy patch server" for installation of system updates, patches, and bug-fixes. 3. Verify digital signatures on all third-party software packages & programs prior to installation (follow procedure). 4. Review source code on all third-party software packages & programs prior to configuration & compilation.
E - Software Vulnerabilities	1. Apply upgrades, patches, and bug-fixes promptly as available. 2. Only install software on system that is necessary for the system to perform as designed and intended. 3. Identify and remove any unnecessary software from system.
F - Password & Account Attacks	1. Set policy of NO floppy disk drives present or installed on server box. 2. Permit and create a minimal number of user accounts on system. 3. Identify and remove any unnecessary user or system accounts. 4. Set password complexity standard. 5. Set password aging standard (/etc/login.defs). 6. Password Protect boot loader. 7. Remove shell access for accounts not requiring shell access (/dev/null). 8. Schedule and run frequent user and system account audits. 9. Periodically audit password complexity (John). 10. Regularly audit /etc/passwd and /etc/shadow files for anomalies. 11. Regularly audit login and account access (wtmp, utmp, btmp). 12. Control command execution & service access with SUDO configuration. 13. Install HostSentry. 14. Appropriately utilize logon banners (/etc/issue & sshd_config). 15. Log Analysis (LogSentry & proprietary scripts).
G - File System & System Attacks	1. Implement "least privilege" policy. 2. Mount file systems "ro", "nodev", "nosuid", and "noexec" when feasible. 3. Restrict system startup capabilities (inittab and xdm). 4. Restrict system shutdown capabilities (inittab and xdm). 5. Restrict system reboot capabilities (inittab and xdm). 6. Permit no direct root logins except from secure console (/etc/securetty). 7. Permit no direct remote root logins (sshd).

	8. Regularly audit system for SUID & SGID enabled scripts or executables. 9. Restrict cron access to root account (cron.allow and cron.deny). 10. Establish intelligent environmental variable settings (profile and umask). 11. Protect security sensitive scripts by sourcing protective function library. 12. Use full path names of programs and executables in scripts. 13. Regularly audit for "privilege mismatches" and world writable directories. 14. Regularly audit file system for permissions and ownerships. 15. Permit no local mail delivery on system. 16. Regularly flush /tmp file system. 17. Limit available shells, interpreters, and compilers on system. 18. Apply appropriate IDS rules. 19. System Log Analysis (LogSentry & proprietary scripts).
H - Memory Attacks	1. Enable "core dump" prevention (ulimit –c 0). 2. Establish software integrity checking policy. 3. Apply upgrades, patches, and bug-fixes promptly as available. 4. Apply appropriate IDS rules.
I - Malware Attacks	1. Establish software integrity checking policy. 2. Enable capability of logging to remote syslog server (-h). 3. Install and regularly run rootkit hunter (rkhunter). 4. Install and appropriately configure and use tripwire.
J - Network Service Attacks	1. Use default "drop" policy on firewall input and forward chains. 2. Tightly control remote access to services on firewall machine itself with a dedicated "lockdown" rule set. 3. Install PortSentry. 4. Install LogSentry (LogCheck). 5. Install Tcpwrappers. 6. Identify and remove unnecessary services. 7. Limit, restrict, or eliminate inetd service control. 8. Remove unnecessary inetd.conf entries as opposed to "commenting". 9. Jail (chroot()) services when possible and feasible. 10. Apply appropriate IDS rules. 11. Appropriately utilize warning banners (/etc/issue, SSH, & tcpwrappers). 12. Strictly control X11 access. 13. Firewall Log Analysis (proprietary scripts). 14. Block potentially malicious ICMP packets. 15. Synchronize network clocking for accurate time stamping & correlation. 16. Retain record of packets entering/leaving network for forensic purposes. 17. Permit remote access from trusted administrative workstations only.
K - Network Sniffing Attacks	1. Encrypt the transfer of all security sensitive data and information. 2. Require SSH for remote console logins and sessions. 3. Require SSH for remote file copy and transfer. 4. Use SSL/TSL security & encryption for administrative web accesses. 5. Schedule network scans for promiscuous network interfaces (rkhunter). 6. Implement IP spoofing detection and prevention measures. 7. Deny use of insecure protocols.
L - Denial of Service Attacks	1. Activate kernel syn-cookie support. 2. Utilize "limit match" firewall rule sets as appropriate. 3. Configure PortSentry with tcpwrappers and iptables blocking modes. 4. Place all file systems on a separate disk partition as the system allows. 5. Place the /tmp file system on a separate partition. 6. Disable capability to accept logging from remote device (-r). 7. Implement an intelligent log file rotation and archiving system. 8. Block ICMP broadcast (amplification) packets.

The "Risk Mitigation Matrix" will be referenced throughout all stages of this guide. When a particularly area of security concern is encountered and a preventative or mitigative measure must be applied, the matrix will be referenced in the following format:

An example would be **RMM:D:3**, which means **Measure 1** of **Software Integrity Attack Concern** of the **Risk Mitigation Matrix** is being addressed at this point in the document. All such references will be highlighted in ***green***.

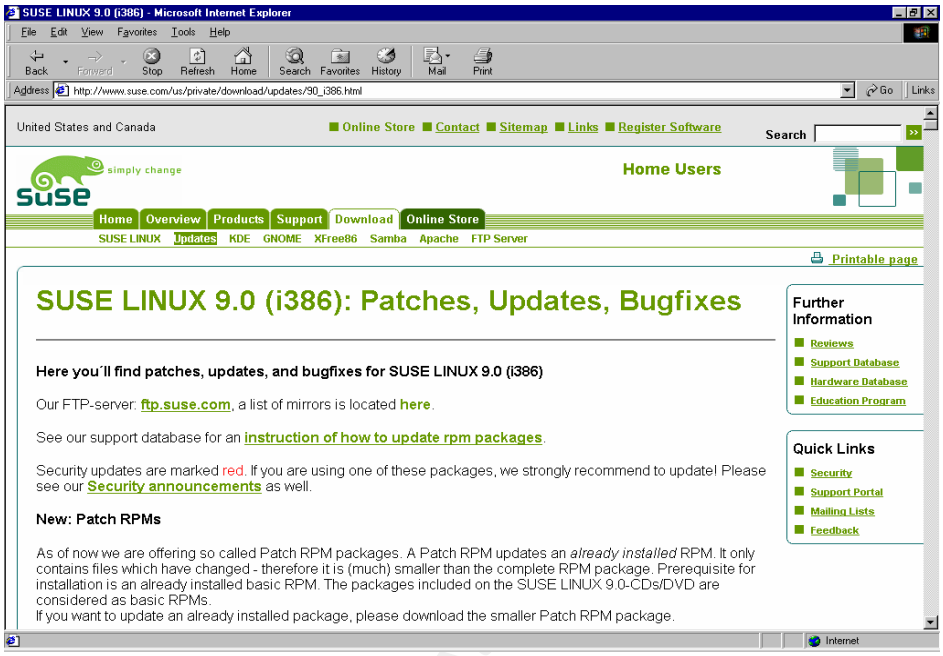
Notable Linux Kernel and Netfilter Vulnerabilities

Several vulnerabilities have been found or are known to exist in the Linux Kernel prior to kernel version 2.4.25. These vulnerabilities are detailed in **Table II Known Linux Kernel Vulnerabilities**. Vulnerabilities must be mitigated either by patch application or kernel upgrade.

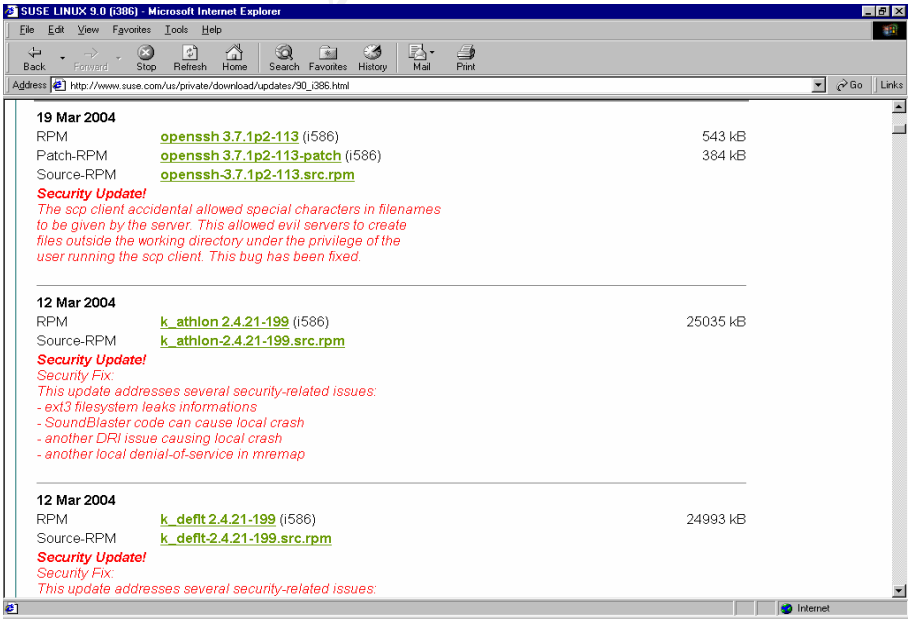
TABLE II
KNOWN LINUX KERNEL VULNERABILITIES

VULNERABILITY	DESCRIPTION
CAN-2003-0187	The connection tracking core of Netfilter for Linux 2.4.20, with CONFIG_IP_NF_CONNTRACK enabled or the ip_conntrack module loaded, allows remote attackers to cause a denial of service (resource consumption) due to an inconsistency with Linux 2.4.20's support of linked lists, which causes Netfilter to fail to identify connections with an UNCONFIRMED status and use large timeouts.
CAN-2003-0244	The route cache implementation in Linux 2.4, and the Netfilter IP conntrack module, allows remote attackers to cause a denial of service (CPU consumption) via packets with forged source addresses that cause a large number of hash table collisions.
CAN-2003-0246	The ioperm systemcall in Linux kernel 2.4.20 and earlier does not properly restrict privileges, which allows local users to gain read or write access to certain I/O ports.
CAN-2003-0619	Integer signedness error in the decode_fh function of nfs3xdr.c in Linux kernel before 2.4.21 allows remote attackers to cause a denial of service (kernel panic) via a negative size value within XDR data of an NFSv3 procedure call.
CAN-2003-0985	Critical security vulnerability in the Linux kernel memory management code in mremap system call due to incorrect bound checks. An incorrect bound check discovered inside the do_mremap() kernel code performing remapping of a virtual memory area may lead to creation of a virtual memory area of 0 bytes in length.
CAN-2003-0984	Real time clock (RTC) routines in Linux, Linux kernel 2.4.23 and earlier do not properly initialize their structures, which could leak kernel data to user space.
CAN-2004-0079	The OpenSSL group using the Codenomicon TLS Test Tool uncovered a null-pointer assignment in the do_change_cipher_spec() function. A remote attacker could perform a carefully crafted SSL/TLS handshake against a server that used the OpenSSL library in such a way as to cause OpenSSL to crash. This could lead to a denial of service. All versions of OpenSSL from 0.9.6c to 0.9.6l inclusive and from 0.9.7a to 0.9.7c inclusive are affected.
CAN-2004-0112	Flaw in SSL/TLS handshaking code when using Kerberos ciphersuites. A remote attacker could perform a carefully crafted SSL/TLS handshake against a server configured to use Kerberos ciphersuites in such a way as to cause OpenSSL to crash. Versions 0.9.7a, 0.9.7b, and 0.9.7c of OpenSSL are affected by this issue. Any application that makes use of OpenSSL's SSL/TLS library may be affected.

SUSE Linux AG maintains a dedicated site at **http://www.suse.com/us/private/download/updates/90_i386.html** where patches, updates, and bugfixes can readily be obtained and downloaded (see figure below).



Each security update is clearly explained, to include the underlying flaw or vulnerability mechanism that the update addresses or fixes. SUSE Linux AG makes every attempt possible to make “patch management” both digestible and simple.



PART 2: SYSTEM INSTALLATION & HARDENING

PHASE I: OPERATING SYSTEM INSTALLATION

STEP 1: INSTALL CORE OPERATING SYSTEM

1. **Power On**
   Disconnect Linux firewall box from the network (**RMM:D:1**), turn on the power, and the insert CD-ROM media. The system will boot to the SUSE 9 installation program automatically[8].

2. **Language Selection**
   The first screen to appear is the "Language Selection" screen. Select the desired language. For this install "**English (US)**" is selected.

3. **Select Installation Type**
   Next, select the type of installation. The choices offered are:

   - New installation
   - Update existing system
   - Repair installed system
   - Boot installed system
   - Abort installation

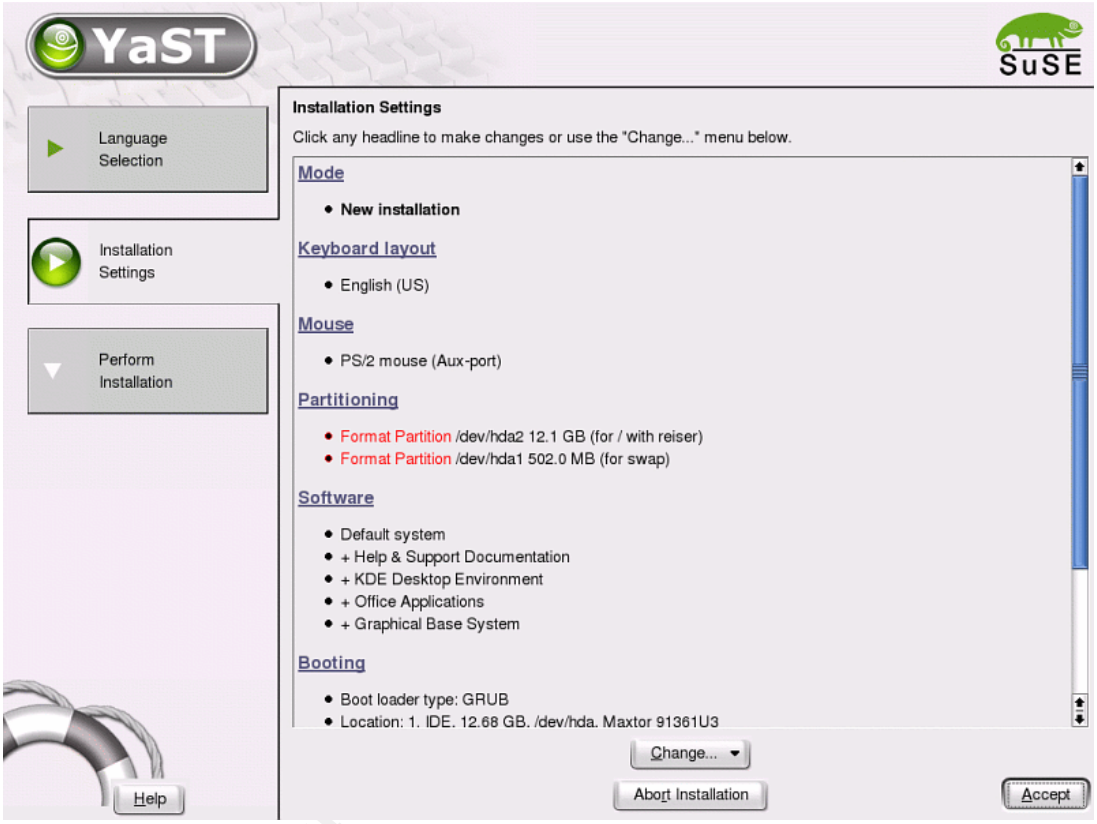
   Select "**New installation**".

4. **SUSE 9 Recommendations vs. What You Really Want**
   At this point, the SUSE 9 YaST2installation program recommends an installation configuration (see figure below), to include the following:

   - Keyboard type
   - Mouse type
   - Partitioning scheme
   - Software
   - Booting scheme to include the boot loader
   - Time Zone

[8] This assumes, of course, that the system BIOS supports booting from CD-ROM, and the BIOS boot order has been to boot from CD-ROM. If this fails, it may be necessary to enter the BIOS setup program and set the appropriate "boot order". However, once installation is complete, the CD-ROM boot option should be disabled in the BIOS (see the section on **System BIOS**.

Other than the keyboard, mouse type, and time zone, the installation program wishes to take us in the wrong direction. Clearly, the time has come to assume control of the situation. Fortunately, SUSE 9 provides the ability to change these selections upon request.



5. **Select Boot Scheme**
First, select "**Booting**" to set the boot schemata, and then select "**Edit**". SUSE 9 installs GRUB by default. This guide will utilize LILO, so select "**LILO**", and then select "**Start configuration from scratch**". This will install LILO in the MBR (see section **Hardening Boot Environment**).

6. **Select Time Zone**
Select "**Time zone**" and set as necessary.

7. **Select Partitioning**
Next, select "**Partitioning**", then "**Create custom partition setup**", and then "**Custom partitioning – for experts**". The /etc/fstab template format on the following page will be applied to the Linux firewall. This will determine the

partitioning scheme applied and access control to the file system itself (see section **Hardening File System Access Control**).

```
################################################################################
# /etc/fstab
################################################################################
# Device or     Mount          FS Type    Options                    Dump  Fsck
# Partition     Point                                                      Pass
################################################################################
/dev/hda3       /              ext2       defaults                   1     1
/dev/hda1       /boot          ext2       rw,nosuid,nodev            1     2
/dev/hda7       /home          ext2       rw,nosuid,nodev            1     2
/dev/hda9       /opt           ext2       rw,nosuid,nodev            1     2
/dev/hda2       /safe          ext2       rw,nosuid,nodev            1     2
/dev/hda10      /storage       ext2       rw,nosuid,nodev            1     2
/dev/hda6       /usr           ext2       ro                         1     2
/dev/hda8       /var           ext2       rw,nosuid,nodev            1     2
/dev/hda11      /tmp           ext2       nosuid,nodev,noexec        1     2
/dev/hda5       swap           swap       pri=42                     0     0
devpts          /dev/pts       devpts     mode=0620,gid=5            0     0
proc            /proc          proc       defaults                   0     0
usbdevfs        /proc/bus/usb  usbdevfs   noauto                     0     0
/dev/cdrecorder /media/cdrecorder auto     ro,noauto,user,nosuid,nodev 0    0
/dev/cdrom      /media/cdrom   auto       ro,noauto,user,nosuid,nodev 0    0
################################################################################
```

For the Linux firewall, the objectives for the partitioning and file system setup are as follows:

- **Keep It Simple**. SUSE 9 Linux allows partitions to be formatted as Ext2, Ext3, FAT, JFS, ReiserFS, or, XFS, with the ReiserFS selected by default. However, the ext2 file system is more in line with our objective, so each of the partitions on the Linux firewall will be formatted with ext2[9].

- **Keep It Segregated**. The segregation of data is important for many reasons and concerns. Primarily, however, the goal is to prevent or avoid a system denial of service attack by exhausting the free space on a partition. Therefore, each file system will be placed on a separate partition, to include /tmp[10] (**RMM:G:1**).

[9] There is a good deal of room for debate here. Journaling files systems were designed for enterprise computing, hard-disks with large partitions, quick recovery from crashes, high-performance I/O, and reduced fragmentation. Additionally, journaling file systems have eliminated the need for fsck.

[10] Through the use of "extended partitioning", Linux can support up to 15 partitions on an SCSI disk and 63 partitions on an IDE disk.

- **Keep It Flexible**. Separate partitions provide flexibility when applying security settings to the file system (**RMM:G:2**).

- **Keep It Safe**. File system corruption is always possible after a system crash or disruption. For this reason it is always advisable to place a separate copy of a bootable kernel on a separate partition of its own.

Based on the above criteria, the 160 GB hard drive will be partitioned as follows:

▪ /dev/hda1	/boot	100 MB
▪ /dev/hda2	/safe	100 MB
▪ /dev/hda3	/	10 GB
▪ /dev/hda5	swap	2 GB
▪ /dev/hda6	/usr	20 GB
▪ /dev/hda7	/home	5 GB
▪ /dev/hda8	/var	40 GB
▪ /dev/hda9	/opt	5 GB
▪ /dev/hda10	/storage	67.8 GB
▪ /dev/hda11	/tmp	10 MB

8. **Select Software**

Once the partitioning configuration is complete, select "**Software**". At this point, the SUSE 9 Linux YaST2installation program provides three selections:

- Minimum system
- Minimum graphical system
- Default system

Select "**Minimum graphical system**". It will be necessary to either manually "add" or "remove" software packages or programs from this point onward.

Add the following programs necessary for our Linux firewall installation and setup[11]:

- Documentation
- IPTraf
- Libpcap
- Ntop
- TCPWrappers
- TCPDump
- Sendmail
- XNTP
- Tripwire
- SUDO
- John
- Dump
- Strace
- KDE (refer to section on **Hardening XDMCP**)
- C/C++ Compiler (refer to **Software Requirements**)

[11] SUSE's netfilter/iptables is included by default with the "minimum" or core installation.

Deselect the following packages included by default with the minimum system (**RMM:E:3**):

- NFS (client and server)
- NIS (client and server)
- Portmap
- Java2-jre
- Postfix

9. **Install System**
Once all the software packages have been selected and any dependencies have been resolved, select "**Accept**". All the requirements for installation have now been met, so select "**Accept**" once again. The YaST2 installer will ask for permission to start the installation. Select "**Yes, install**". The installer then prepares the hard disk and begins the installation.

10. **Set Root Password**
Once the basic installation has completed, the YaST2 installer will ask you set the root password. Set according to policy (**RMM:F:4**).

11. **Configure Network**
Next, the YaST2 installer will ask you to configure the network devices (see figure below). Set the network interface IP addresses, the gateway IP address, and the DNS resolver configuration at this time.

12. **Update System**
The YaST2 installer will ask you if you wish to update the system. Select "**yes**", and jump to Step 2 below (**RMM:D:2**).

13. **Complete Installation**
Once the patches and updates have been successfully installed from the "proxy patch server", the YaST2 installer provides the opportunity to change any hardware configuration settings, and the opportunity to add new users to the system. Add two firewall administrator accounts (**fwadmin1 and fwadmin2**) at this time (**RMM:G:6**) (**RMM:F:3**). The core installation phase is now complete.

STEP 2: INSTALL PATCHES & BUG FIXES

Once the core operating system has been installed, all relevant and necessary updates, patches, and bugfixes must be installed or applied (**RMM:E:1**). The security policy established for this guide requires that updates, patches, and bugfixes never be directly download and applied to the system. Rather, they should be installed indirectly though what we shall refer to as a "proxy patch server[12]" (**RMM:D:2**). The reasons for this are numerous, but include:

[12] The proxy patch server is itself a specially hardened machine configured particularly for the purpose of retrieving software updates and bug fixes from the internet. The Internet connection for this server is through a bridged, and thus stealth, interface with no assigned IP address.

- The retrieval of patches and updates can be scheduled and acquired locally at a time when bandwidth requirements are inconsequential.

- The server to be patched (in this case a SUSE 9 Firewall) is not subject to the external forces or elements of compromise during the retrieval process.

- Software integrity can be checked and confirmed prior to application to the affected server.

- The software can be tested in advance to make sure it does not break the system or other software packages.

The procedure for system patching and updating follows below[13]. This procedure assumes the "proxy patch server" has already been selected, configured, and hardened for this role.

1. Select an appropriate remote SUSE mirror update site.

2. Schedule the "proxy patch server" to regularly retrieve SUSE 9.0 system updates from remote mirror server. This can easily be accomplished with cron and the following script "**sync_patch-server.sh**":

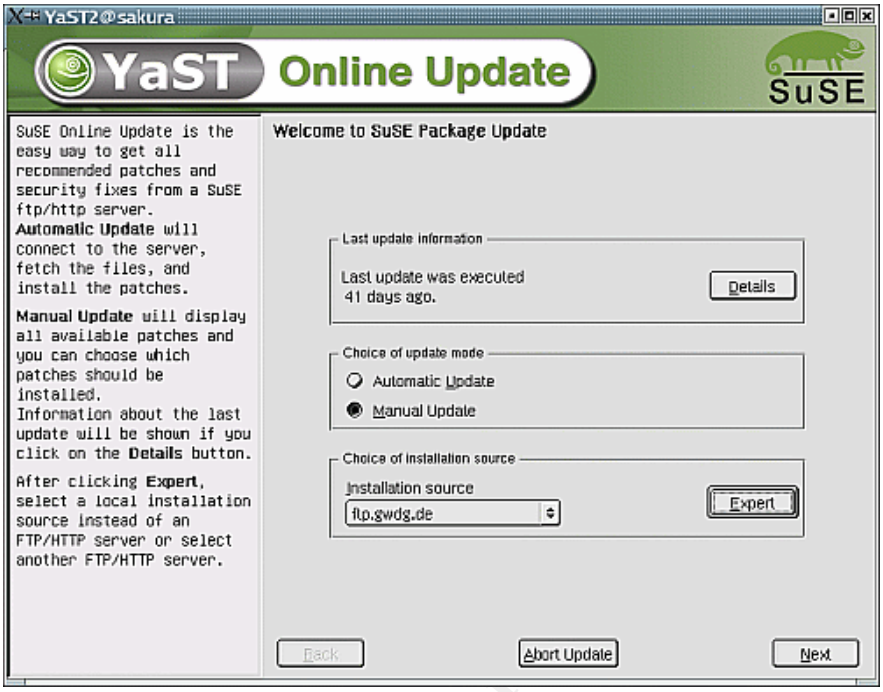
```
#!/usr/bin/sh
#######################################################
# Program:          sync_patch-server.sh
# Date:             March 12, 2004
# Description:      synchronizes suse 9.0 updats and patches
#                   on a local patch-server.
#######################################################
if [ ! –d /usr/local/update/9.0 ];
then
      mkdir usr/local/update/9.0
fi

logger -t update80.sh[$$] Updating from gwdg - 9.0
rsync -auqz  --stats --delete --exclude zq1 ftp.gwdg.de::SuSE/ftp.suse.com/suse/i386/update/9.0/. \
      /usr/local/update/update/9.0/.
logger -t update90.sh[$$] Finish Updating from gwdg - 9.0
#######################################################
```

3. Verify and test the updated packages.

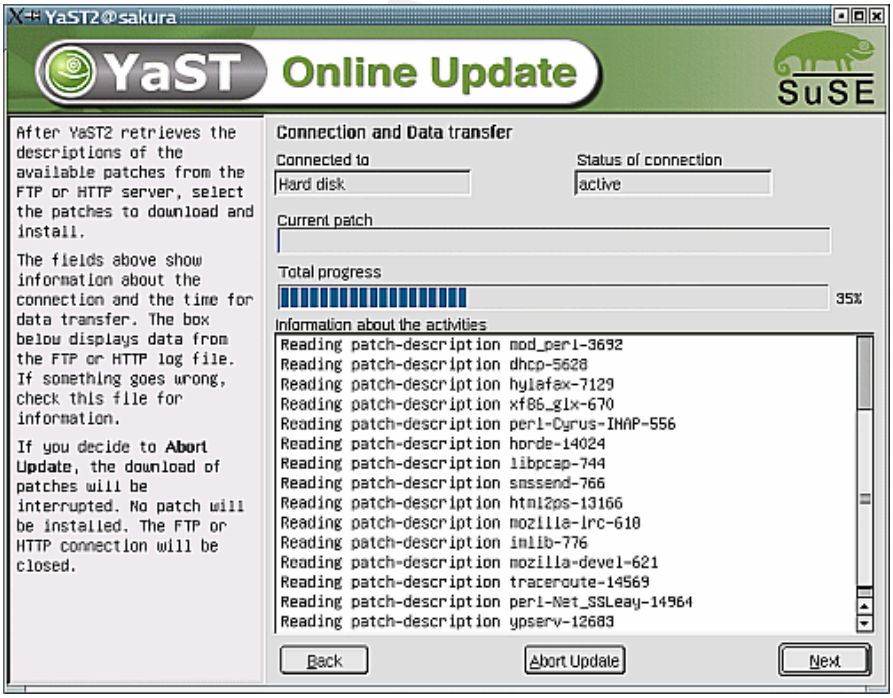
4. On the SUSE 9.0 firewall, start the YAST2 Online Update (YOU) module, and select "**User defined location**".

---

[13] The concept, and some of the script segments, for this procedure was gleaned from the "SUSE Linux Unofficial FAQ" at **http://susefaq.sourceforge.net**.



5. Enter the URI path to the “proxy patch server” in the “location” box and apply the patches. If successful, the update screen should appears as it does below:



6. All patches and updates have now successfully been applied.

STEP 3: UPGRADE LINUX KERNEL & IPTABLES

SUSE 9 installs the Linux kernel 2.4.21 and iptables 1.2.8 by default. This procedure describes and details a "from scratch" compilation and installation of the latest Linux kernel version with iptables (to include POM for the 2.4.x kernel or POM-ng for the 2.6.x kernel if desired). As of this date, the latest stable kernel version stands at 2.4.25, and the latest version of iptables stands at 1.2.9 (**RMM:E:1**).

1. Download the most recent versions of the following software into the /usr/src directory:

   - **Linux kernel version 2.4.25 from http://www.kernel.org.**
   - **iptables version 1.2.9 from http://www.netfilter.org.**
   - **iptables patch-o-matic from http://www.netfilter.org.**

2. Verify software integrity by applying the procedure "Software Integrity Verification Procedure" in **Appendix K** (**RMM:D:3**).

3. Delete symbolic link to the current kernel source tree, and optionally, remove the current kernel source tree:

   - **rm /usr/src/linux**
   - **rm –Rf /usr/src/linux-2.4.x (optional if needed)**

4. Unpack the software downloaded in Step 1 in the /usr/src directory:

   - **cd /usr/src**
   - **bunzip2 linux-2.4.25.tar.bz2**
   - **tar –xvf linux-2.4.25.tar**
   - **bunzip2 iptables-1.2.9.tar.bz2**
   - **tar –xvf iptables-1.2.9.tar**
   - **bunzip2 patch-o-matic.tar.bz2**
   - **tar –xvf patch-o-matic.tar**

5. Create symbolic links to the Linux kernel and iptables source trees:

   - **ln –s /usr/src/linux-2.4.25 /usr/src/linux**
   - **ln –s /usr/src/iptables-1.2.9 /usr/src/iptables**

6. (Optional) Apply the kernel patch (patch-o-matic). You have three choices: pending (modules slated to be added to the kernel), base (extensions to the kernel), or extra (more extensions).

   - **cd /usr/src/patch-o-matic**
   - **KERNEL_DIR=/usr/src/linux ./runme pending**
   - **(optional) KERNEL_DIR=/usr/src/linux ./runme base**

- **(optional) KERNEL_DIR=/usr/src/linux ./runme extra**

7. Prepare the kernel source code. Move to the kernel source directory and clean up any existing .o files and old dependencies:

   - **cd /usr/src/linux**
   - **make mrproper**

8. Configure the new 2.4.25 Kernel. The options available are “make config”, “make menuconfig”, “make xconfig”, and “make gconfig”. To use “xconfig”, the “qt-devel” package must be installed on the system. To use “gconfig”, the “GTK+ 2.0-devel”, “glib2”, and “libglade 2.0” packages must be installed on the system. We will use “make xconfig”:

   - **make xconfig**

   To set the configuration options for netfilter, enter the “network options” area of the configuration file. Select all desired options as “modules. Once complete, save the configuration and exit from the configuration tool.

9. Make (compile) the new 2.4.25 kernel:

   - **make dep**
   - **make bzImage**
   - **make modules**
   - **make modules_install**

10. Make (compile) and install iptables 1.2.9:

    - **cd /usr/src/iptables**
    - **make KERNEL_DIR=/usr/src/linux**
    - **make install KERNEL_DIR=/usr/src/linux**
    - **(optional for developers) make install-devel**

11. Copy the new 2.4.25 kernel image to the boot directory, and the new iptables 1.2.9 binaries to their proper directories. This guide suggests you use the following script to guard against frivolous error:

```
#!/bin/sh
#########################################################################
# copy_kernel.sh
# copies new kernel and support files to boot partition
#########################################################################

########################
# VARIABLES
########################
SCRIPT_NAME="copy_kernel.sh"
```

```
VERSION="-2.4.25"
OLD_VERSION="-2.4.20"
SOURCE_DIR="/usr/src/linux/"
BOOT_DIR="/boot/"
IMAGE="/usr/src/linux/arch/i386/boot/bzImage"
MAP="System.map"
DOT_CONFIG=".config"
CONFIG="vmlinuz.config"
BUILD_IPTABLES="1"
IPT_VERSION="1.2.9"

########################
# BEGIN
########################
echo "Running $SCRIPT_NAME"
echo "Renaming $BOOT_DIR/$MAP to $BOOT_DIR/$MAP.$OLD_VERSION"
mv $BOOT_DIR/$MAP $BOOT_DIR/$MAP.$OLD_VERSION

echo "Renaming $BOOT_DIR/$CONFIG to $BOOT_DIR/$CONFIG.$OLD_VERSION"
mv $BOOT_DIR/$CONFIG $BOOT_DIR/$CONFIG.$OLD_VERSION

echo "Copying $SOURCE_DIR/$MAP to $BOOT_DIR/$MAP.$VERSION"
cp $SOURCE_DIR/$MAP $BOOT_DIR/$MAP.$VERSION

echo "Copying $SOURCE_DIR/$CONFIG to $BOOT_DIR/$CONFIG.$VERSION"
cp $SOURCE_DIR/$CONFIG $BOOT_DIR/$CONFIG.$VERSION

echo "Linking $BOOT_DIR/$MAP.$VERSION to $BOOT_DIR/$MAP"
ln -fs $BOOT_DIR/$MAP.$VERSION $BOOT_DIR/$MAP

echo "Copying new kernel image $IMAGE to boot directory as $BOOT_DIR/vmlinuz$VERSION"
cp $IMAGE $BOOT_DIR/vmlinuz$VERSION

if [ "$BUILD_IPTABLES" = "1" ];
then
        echo "Copying new iptables version $IPT_VERSION executables to /usr/local/sbin directory"
        cp /usr/local/sbin/iptables /usr/sbin
        cp /usr/local/sbin/ip6tables /usr/sbin
        cp /usr/local/sbin/iptables-save /usr/sbin
        cp /usr/local/sbin/iptables-restore /usr/sbin
        cp /usr/local/sbin/ip6tables-save /usr/sbin
        cp /usr/local/sbin/ip6tables-restore /usr/sbin
fi

echo "Completed Running $SCRIPT_NAME"
```

12. Update boot loader configuration. Add a section to the lilo.conf (or grub.conf) file to boot the new kernel. We want to leave the old kernel in a bootable condition in case there are problems with the new kernel:

    image=/boot/vmlinuz-2.6.3
    label=/linux-2.6.3
    root=/dev/hda1
    read-only

13. Reinstall the boot loader: **/sbin/lilo**

14. Reboot the machine and test the configuration.

PHASE II: INSTALL & CONFIGURE THIRD PARTY SOFTWARE

When installing third-party software on a computer system, it is imperative that the proper checks are made to ensure the original source code or distribution files have not been compromised[14]. Generally, software packages will offer one of three means of integrity checking:

- MD5 checksums
- Cryptographic signatures using GnuPG, the GNU Privacy Guard
- The built-in RPM integrity verification mechanism for RPM packages

The procedure detailed in **Appendix K Software Integrity Verification Procedure** should be included as a critical step when installing all third party software programs and packages.

STEP 1: Install PortSentry[15]

There are several version of PortSentry available, each containing subtle variances in functionality. This guide has selected version 1.1 as it provides the functionality needed for the Linux firewall. To configure and install, follow these steps (**RMM:J:3**):

1. Acquire PortSentry version 1.1 from an appropriate source.

2. Verify software integrity by applying the procedure "Software Integrity Verification Procedure" in **Appendix K** (**RMM:D:3**).

3. Uncompress the tarball: **tar -zxvf portsentry-1.1.tar.gz**

4. Review the source code (**RMM:D:4**).

5. Change to the directory portsentry-1.1: **cd portsentry-1.1**

---

[14] A classic example of this is documented by CERT® Advisory CA-2002-30 "Trojan Horse tcpdump and libpcap Distributions" from November 13, 2002, where it was announced that the server hosting tcpdump and libpcap at www.tcpdump.org, was compromised. Reportedly, an intruder made modifications to the source code of tcpdump and libpcap to include trojan horse code. Downloads of the source code contained the trojan code. Once the code was compiled, the trojan executed and attempted to connect to host 212.146.0.34 on port 1963.

[15] PortSentry, LogSentry (LogCheck), and HostSentry, the creations and brainchildren of Craig Rowland, are affectionately known now as the TriSentry Suite or the Sentry Tools. This excellent suite of tools was once freely available from Psionic Software. However, in October of 2002, Cisco Systems bought Psionic Software, and that included Craig along with his intellectual property. Hence, the original tools are now somewhat hard to find. Many copies, such as the authors, existed in personal software archives. Fortunately for those without an archive, PortSentry 1.2 and LogCheck 1.1.1 have now been made available through SourceForge at **http://sourceforge.net/projects/sentrytools**. HostSentry, however, is conspicuously missing.

6. Edit the portsentry_config.h file to change the default path to the PortSentry configuration file, path and name of TCP wrapper hosts.deny file, the syslog facility (LOG_DAEMON), or syslog logging level (LOG_NOTICE).

7. Edit the **portsentry.conf** file, and make the following changes[16]:

   - Uncomment "anal" port list, and comment out the "aware" and "bare bones" ports lists.

   - Set advanced ports to monitor and exclude:

     ADVANCED_PORTS_TCP="1024"
     ADVANCED_PORTS_UDP="1024"
     ADVANCED_EXCLUDE_TCP="113,139"
     ADVANCED_EXCLUDE_UDP="520,138,137,67"

   - Set the correct paths to be used for related configuration files:

     IGNORE_FILE="/usr/local/psionic/portsentry/portsentry.ignore"
     HISTORY_FILE="/usr/local/psionic/portsentry/portsentry.history"
     BLOCKED_FILE="/usr/local/psionic/portsentry/portsentry.blocked"

   - Set "**blocking mode**" to TRUE:

     BLOCK_UDP="1"
     BLOCK_TCP="1"

   - Configure PortSentry to integrate with iptables and tcpwrappers for blocking. Add the following two entries:

     KILL_ROUTE="/usr/local/bin/iptables -I INPUT -s $TARGET$ -j DROP"
     KILL_HOSTS_DENY="ALL: $TARGET$"

   - Set the "port banner" directive.

8. Edit the portsentry.ignore files, and enter entries for network segments or hosts to be ignored by PortSentry. Be certain to include the firewall loopback address, local routers, and all network segments being protected by the firewall in this file:

[16] It is critical that the portsentry.conf file be precisely configured as PortSentry will be the primary defense for the SUSE Linux firewall during the time periods when the firewall rule sets are loaded or reloaded. A complete portsentry.conf file is available at **Appendix G: "PortSentry Configuration File"**.

127.0.0.0/8
0.0.0.0
192.168.0.0/24
192.168.1.0/24
192.168.2.0/24
192.168.3.0/24

9. Compile and Install PortSentry:

   make linux
   make install

10. Create a startup script for PortSentry. For the SUSE Linux firewall, we will use "**advanced stealth**" mode, and utilize the same script to start HostSentry. The following script is recommended:

```
#!/bin/sh
###############################################################################
# portsentry
# description: Portsentry & Hostsentry Startup Script
###############################################################################

case "$1" in
'start')
        # startup PortSentry
        echo "Starting PortSentry . . . . . "
        /usr/local/psionic/portsentry/portsentry -atcp
        /usr/local/psionic/portsentry/portsentry -audp
        ps -ef | grep portsentry | awk '{print $2}' > /var/run/portsentry.pid

        # startup HostSentry
        echo "Starting HostSentry . . . . . "
        python /usr/local/abacus/hostsentry/hostsentry.py
        ps -ef | grep hostsentry | awk '{print $2}' > /var/run/hostsentry.pid
;;

'stop')
        # stop HostSentry
        echo "Stopping HostSentry . . . . . "
        kill `ps -ef | awk '/hostsentry/ { print $2 }'`

        # stop PortSentry
        echo "Stopping PortSentry . . . . . "
        kill `ps -ef | awk '/portsentry/ { print $2 }'`
;;

'restart')
        # restart PortSentry and HostSentry
        echo "Restarting PortSentry & HostSentry . . . . . "
        kill -HUP `ps -ef | awk '/hostsentry/ { print $2 }'`
        kill -HUP `ps -ef | awk '/portsentry/ { print $2 }'`
;;

*)
        echo "Usage: $0 { start | stop | restart }"
;;
esac
exit 0
```

11. Start PortSentry and check for errors: **./etc/init.d/portsentry start**

STEP 2: Install LogSentry (LogCheck)
To configure and install LogSentry, follow these steps (**RMM:J:4**):

1. Acquire LogCheck version 1.1.1 from an appropriate source.

2. Verify software integrity by applying the procedure "Software Integrity Verification Procedure" in **Appendix K** (**RMM:D:3**).

3. Uncompress the tarball: **tar -zxvf logcheck-1.1.1.tar.gz**

4. Review the source code (**RMM:D:4**).

5. Change to the directory logcheck-1.1.1: **cd logcheck-1.1.1**

6. Edit the "**logcheck.sh**" file. Be sure to point the $LOGTAIL executable to the system logs it should parse and report on. A full "**logcheck.sh**" file is available at **Appendix H: LogSentry Configuration File**.

7. Edit the "**logcheck.hacking**", "**logcheck.ignore**", "**logcheck.violations**", and "**logcheck.violations.ignore**" files as needed.

8. Schedule frequent "log checks" with cron:

   00,15,30,45 * * * * /usr/local/etc/logcheck.sh

STEP 3: Install HostSentry
To configure and install HostSentry, follow these steps (**RMM:F:13**):

1. Acquire HostSentry version 0.02 from an appropriate source.

2. Verify software integrity by applying the procedure "Software Integrity Verification Procedure" in **Appendix K** (**RMM:D:3**).

3. Uncompress the tarball: **tar -zxvf hostsentry-0.02.tar.gz**

4. Review the source code (**RMM:D:4**).

5. Change to the directory hostsentry-0.02: **cd hostsentry-0.02**

6. Edit the "**hostsentry.conf**" file as necessary. A full "**hostsentry.conf**" file is available at **Appendix I: HostSentry Configuration File**.

7. Edit the "**hostsentry.ignore**" file as needed.

STEP 4: Install SmartMonTools

Hard disk failure is one of the most common causes of unexpected system failure. SmartMonTools is a set of utilities that continuously analyzes the data produced by SMART enabled ATA disks, such as internal temperature, and warns the administrator of the potential or imminent failure. Armed with this information, the administrator can schedule maintenance under controlled circumstances and avoid device or network failure. Thus, SmartMonTools provide an important element in improved network reliability and uptime (**RMM:C:4**).

1. Download SmartMonTools from **http://smartmontools.sourceforge.net**.

2. Verify software integrity by applying the procedure "Software Integrity Verification Procedure" in **Appendix K** (**RMM:D:3**).

3. Uncompress the tarball: **tar -zxvf smartmontools-5.20.tar.gz**

4. Review the source code (**RMM:D:4**).

5. Change to the directory smartmontools-5.30: **cd smartmontools-5.30**

6. Configure and install:

   ./configure
   make
   make install

7. Open and edit the configuration file **/etc/smartd.conf** and add the following line to monitor all attributes on the machine's hard disk:

   /dev/hda -S on -o on -a -I 194 -m administrator@network.com -M exec /root/bin/smart_report.sh

8. The administrator will be now be immediately notified by email of any significant changes to the hard disk.

STEP 5: Install APC PowerChute PBEAgent

The PBEAgent will provide graceful power downs and reboots in the event of power failure, providing another element of network reliability (**RMM:C:3**).

1. Download the Linux binary **pbe_agent_linux_jvm.bin** from **http://www.apc.com**.

2. Execute the binary file

3. Accept the license agreement.

4. Allow the install to automatically detect the attached UPS.

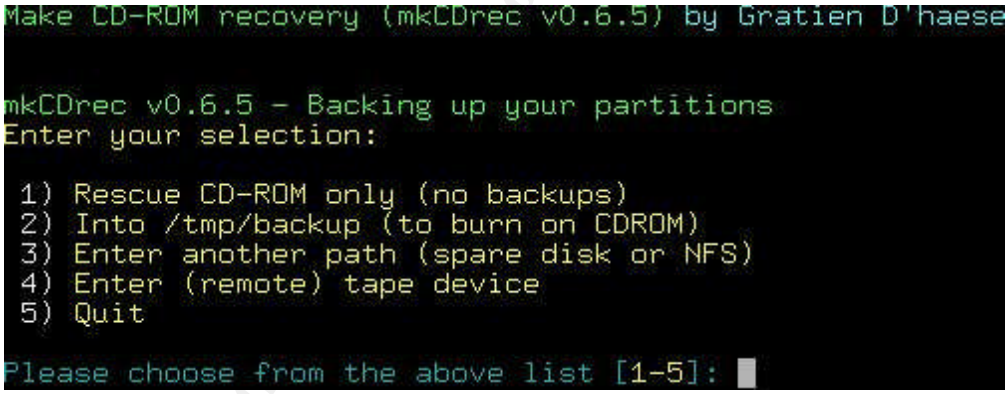
5. Enter a “user name” and “password” for agent access.

6. Allow the PowerChute server to detect and assume control of the agent.

STEP 6: Install mkCDrec and Utilities

mkCDrec provides a means for complete disaster recovery (**RMM:C:5**). The main web site provides the following description:

> *“mkCDrec makes a bootable El Torito disaster recovery image, including backups of the linux system to the same CD-ROM (or CD-RW), or to a multi-volume CD-ROM set. After a disaster (disk crash or system intrusion) the system can be booted from the CD-ROM and one can restore the complete system as it was (at the time mkCDrec was run) with the command /etc/recovery/start-restore.sh. Disk cloning (clone-dsk.sh script) allows one to restore a disk to another disk (the destination disk does not have to be of the same size as it calculates the partition layout itself). A third script, restore-fs.sh, will restore only one filesystem to a partition of your choice, and the user can choose with which filesystem the partition has to be formatted. To restore your system completely just boot from the first CD-ROM made by mkCDrec and type "/etc/recovery/start-restore.sh".*



To install and execute mkCDrec, follow this procedure:

1. Download mkCDrec and utilities from **http://mkcdrec.ota.be/project/download.html**.

2. Verify software integrity by applying the procedure “Software Integrity Verification Procedure” in **Appendix K** (**RMM:D:3**).

3. Uncompress the tarball: **tar -zxvf mkCDrec_v0.7.8.tar.gz**

4. Change to the directory mkcdrec: **cd mkCDrec**

5. Run **make test** to determine if all prerequisites are met. Correct any noted problems.

6. Edit the **Config.sh** file as necessary.

7. Run **make** for interactive mode and you will receive the menu on the preceding page. Select menu item **2)**, or run **make CD-ROM** for batch or script mode.

8. A complete system backup is now available on a bootable CD-ROM if needed for disaster recovery.

STEP 7: Install Snort (RMM:G:18)

1. Download snort version 2.1.1 from **hhttp://www.snort.org/dl**.

2. Verify software integrity by applying the procedure "Software Integrity Verification Procedure" in **Appendix K** (**RMM:D:3**).

3. Uncompress the tarball: **tar -zxvf snort-2.1.1.tar.gz**

4. Change to the directory snort-2.1.1: **cd snort-2.1.1**

5. Configure snort with support for the MySQL database and "flexible response":

   ./configure –with-mysql=/usr/local/src/mysql --enable-flexresp

6. Compile and install snort:

   make
   make install

7. Copy the contents of the etc/ directory and the entire rules/ directory to /etc/snort/.

8. Edit the **snort.conf** file and snort rules as necessary.

9. Add restrictive rules to firewall script permitting access to tcp port 3306 from Linux firewall (**RMM:J:2**) (**RMM:J:17**).

10. Start snort: **./etc/init.d/snort start**

STEP 8: Install Root Kit Hunter (RMM:I:3)

Rootkit Hunter checks for network interfaces in promiscuous mode, and deep scans for the following rootkits, trojans, and backdoors:

- 55808 Trojan - Variant A
- Apache Worm
- Ambient (ark) Rootkit
- BeastKit
- BOBKit
- CiNIK Worm (Slapper.B variant)
- Devil RootKit
- Dica

- Dreams RootKit
- FreeBSD Rootkit
- Fuck`it Rootkit
- Suck`it Rootkit
- GasKit
- Heroin LKM
- ImperalsS-FBRK
- Knark
- Li0n Worm
- MRK
- RootKit for SunOS / NSDAP
- Optic Kit (Tux)
- Oz Rootkit
- Portacelo
- Scalper Worm
- Shutdown
- SHV4
- Sin Rootkit
- Slapper
- Sneakin Rootkit
- SunOS Rootkit
- Superkit
- TBD (Telnet BackDoor)
- T0rn Rootkit
- Trojanit Kit
- Volc Rootkit
- X-Org SunOS Rootkit
- zaRwT.KiT Rootkit

Install Root Kit Hunter as follows:

1. Download Root Kit Hunter from **http://www.rootkit.nl/projects/rootkit_hunter.html**.

2. Verify software integrity by applying the procedure "Software Integrity Verification Procedure" in **Appendix K** (**RMM:D:3**).

3. Uncompress the tarball: **tar -zxvf rkhunter-1.0.5.tar.gz**

4. Review the source code (**RMM:D:4**).

5. Change to the directory rkhunter: **cd rkhunter**

Root Kit Hunter can be run in either "interactive" mode or in a "non-interactive" mode suitable for scripting.

PHASE III: SYSTEM HARDENING

STEP 1: Harden Linux Kernel and TCP/IP Stack

Numerous Linux kernel and TCP/IP stack parameters are modifiable and tunable, many of which are directly related to the performance, reliability, and security of a Linux Firewall. **Table III Tunable Kernel Parameters** lists and describes those parameters of importance to the firewall hardening process.

Variables within the Linux kernel can be set by several means. For instance, the "**sysctl**" utility can be used to either read or set variables through the command line, or through a configuration file. To read a specific variable, enter the following:

sysctl net.ipv4.ip_forward

To set a value with “sysctl”, use the –w option with the assignment as follows:

sysctl -w net.ipv4.ip_forward =0

This command sets the ip_forward value to 0, prints the variable with its new value, and exits.

To set variables from a configuration file, use the –p option with the name of the file to be loaded:

sysctl -p /etc/sysctl_params.conf

All commands within the configuration file starts with the path to the variable, including the variable name, an equal sign, and the value to be set. The path itself is relative to /proc/sys. The following is an example for setting ip_forward to 0:

net.ipv4.ip_forward = 0

The /proc/sys/net file system itself may be used to set values in ipsysctl, and this is particularly useful when configuring variables that should only be set when the firewall rule set is applied. For example, the variable ip_forward should be activated (set to 1) only after the complete firewall rule set has been loaded and the routing table has been established. Using this method, ip_forward would be set as follows:

```
# deactivate ip forwarding
echo "0" > /proc/sys/net/ipv4/ip_forward

# activate ip forwarding
echo "1" > /proc/sys/net/ipv4/ip_forward
```

In most cases and most of the time, the system default value will be satisfactory for the majority of the kernel parameters. In some cases, the setting may depend on the mission assigned to the Linux firewall, or the traffic load the firewall must handle. Consequently, this guide will not make a recommendation for each parameter. However, the parameters outlined below should be set for the maximum security and protection of the firewall box. Additionally, this guide also recommends all kernel parameters to be set in relation to the functioning of the firewall itself, be set within the firewall script, prior to the loading of the firewall rules (refer to the section “**Apply Firewall Rule Set**” and **Appendix L: Function set_kernel_params()**).

1. **Control State Table Entries** (/proc/sys/net/ip_conntrack_max). The default value assigned to this parameter is determined by the amount of system memory. For instance, with 128 MB of RAM, the system will allocate 8192 possible entries and

256 MB of RAM will result in 16376 entries. Since the system in this guide has 1024 MB of RAM, 32,768 connection tracking entries are possible.

2. **Enable Reverse Path Filtering** (/proc/sys/net/ipv4/conf/*/rp_filter) (**RMM:K:6**). This variable provides a built-in means of egress and ingress filtering. When set to TRUE, it will drop spoofed packets which cannot be replied to on the received interface based on the routing table:

```
# enable reverse path filtering and route verification
RP_FILTER="/proc/sys/net/ipv4/conf/*/rp_filter"

for f in $RP_FILTER;
do
    $ECHO "1" > $f
done
```

3. **Ignore ICMP ECHO Broadcasts** (/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts) (**RMM:L:8**). This variable provides a built-in "smurf" protection and therefore should be activated:

```
# set variable for ICMP broadcasts
ICMP_ECHO_BROADCASTS="/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts"
$ECHO "1" > $ICMP_ECHO_BROADCASTS
```

4. **Block Source Routed Packets** (/proc/sys/net/ipv4/conf/*/accept_source_route) (**RMM:K:6**). Quite obviously this is a good thing to do so we want to activate this:

```
# set variable for source routing
ACCEPT_SOURCE_ROUTE="/proc/sys/net/ipv4/conf/*/accept_source_route"

for f in $ACCEPT_SOURCE_ROUTE;
do
    $ECHO "0" > $f
done
```

5. **Block ICMP Redirects** (/proc/sys/net/ipv4/conf/*/accept_redirects) (**RMM:J:15**). Suggestions from the outside on how best to run our business are not needed. We want to turn this feature off:

```
# set variable for accepting ICMP redirects
REDIRECTS="/proc/sys/net/ipv4/conf/*/accept_redirects"

for f in $REDIRECTS;
do
    $ECHO "0" > $f
done
```

6. **Enable Sending ICMP Redirects** (/proc/sys/net/ipv4/conf/*/send_redirects) (**RMM:J:15**). This should be turned off as well:

```
# set variable for sending ICMP redirects
MAKE_REDIRECTS="/proc/sys/net/ipv4/conf/*/send_redirects"

for f in $MAKE_REDIRECTS;
do
    $ECHO "0" > $f
done
```

7. **Enables Martian Logging** (/proc/sys/net/ipv4/conf/*/log_martians) (**RMM:K:6**). This variable essentially allows the blocking of any IP address that essentially makes no sense. We want to activate this:

```
# set variable for bad or impossible addresses
LOG_MARTIANS="/proc/sys/net/ipv4/conf/*/log_martians"

for f in $LOG_MARTIANS;
do
    $ECHO "1" > $f
done
```

8. **Enable TCP SYN Cookie Protection** (/proc/sys/net/ipv4/tcp_syncookies) (**RMM:L:1**). The Syn-Cookie mechanism protects against SYN flooding. We want to activate this:

```
# set variable for TCP SYN cookie protection
SYN_COOKIES="/proc/sys/net/ipv4/tcp_syncookies"
$ECHO "1" > $SYN_COOKIES
```

9. **Ignore Bogus Broadcasts** (/proc/sys/net/ipv4/icmp_ignore_bogus_error_responses) (**RMM:L:8**). Some routers violate RFC 1122 by sending bogus responses to broadcast frames. Such violations are normally logged via a kernel warning. We want to activate this feature:

```
# set variable to ignore bogus error
IGNORE_BOGUS="/proc/sys/net/ipv4/icmp_ignore_bogus_error_responses"
$ECHO "1" > $IGNORE_BOGUS
```

10. **Block ECHO (ping) Requests** (/proc/sys/net/ipv4/icmp_echo_ignore_all). Set to true, this variable will cause the Linux firewall to block all echo requests. Although this sounds like it might be a great feature, it would disable network troubleshooting capabilities. This feature should be deactivated as responses to ECHO request should be controlled by the firewall rule set:

```
# set variable to deny all ping (echo requests)
```

```
DENY_PING="/proc/sys/net/ipv4/icmp_echo_ignore_all"
$ECHO "0" > $DENY_PING
```

11. **Support ARP Proxy** (/proc/sys/net/ipv4/conf/all/proxy_arp) (**RMM:J:2**). Deactivate ARP Proxy capabilities unless absolutely needed:

```
# support for arp proxy
ARP_PROXY="/proc/sys/net/ipv4/conf/all/proxy_arp"
$ECHO "0" > $ARP_PROXY
```

12. **Support Dynamic Addressing** (/proc/sys/net/ipv4/ip_dynaddr) (**RMM:K:6**). Deactivate dynamic addressing support unless absolutely needed:

```
# support for dynamic addressing
DYNAMIC_IP="/proc/sys/net/ipv4/ip_dynaddr"
$ECHO "0" > $DYNAMIC_IP
```

TABLE III
TUNABLE KERNEL PARAMETERS[17]
(/proc/sys/net)

Parameter	Value	Description
IPv4 /proc/sys/net/ipv4/*		
Ip_forward	BOOLEAN	Forwards packets between interfaces. Default = false.
Ip_default_ttl	INTEGER	Default = 64.
ip_local_port_range	INTEGERS	Defines the local port range that is used by TCP and UDP to choose the local port. The first number is the first, the second the last local port number. Default value depends on amount of memory available on the system: > 128Mb 32768-61000 < 28Mb 1024-4999 or even less. This number defines number of active connections, which this system can issue simultaneously to systems not supporting TCP extensions (timestamps). With tcp_tw_recycle enabled (i.e. by default) range 1024-4999 is enough to issue up to 2000 connections per second to systems supporting timestamps.
ip_nonlocal_bind	BOOLEAN	If set, allows processes to bind() to non-local IP addresses, which can be quite useful - but may break some applications. Default is 0.
ip_dynaddr	BOOLEAN	If set non-zero, enables support for dynamic addresses. If set to a non-zero value larger than 1, a kernel log message will be printed when dynamic address rewriting occurs. Default is 0.
ip_no_pmtu_disc	BOOLEAN	Disable Path MTU discovery. Default = false.
ipfrag_high_thresh	INTEGER	Maximum memory used to reassemble IP fragments. When ipfrag_high_thresh bytes of memory is allocated for this purpose, The fragment handler will toss packets until ipfrag_low_thresh is reached.
Ipfrag_low_thresh	INTEGER	See above
Ipfrag_time	INTEGER	Time in seconds to keep an IP fragment in memory.

[17] Information obtained from the Linux kernel version 2.4.20 documentation at **http://www.kernel.org** and the "Ipsysctl Tutorial v.1.0.4" by Oskar Andreasson at **http://ipsysctl-tutorial.frozentux.net/ipsysctl-tutorial.html**.

INET		
Inet_peer_threshold	INTEGER	The approximate size of the storage. Starting from these threshold entries will be thrown aggressively. This threshold also determines entries' time-to-live and time intervals between garbage collection pass. More entries, less time-to-live, less GC interval.
Inet_peer_minttl	INTEGER	Minimum time-to-live of entries. Should be enough to cover fragment time-to-live on the reassembling side. This minimum time-to-live is guaranteed if the pool size is less than inet_peer_threshold. Measured in jiffies.
Inet_peer_maxttl	INTEGER	Maximum time-to-live of entries. Unused entries will expire after this period of time if there is no memory pressure on the pool. Measured in jiffies.
inet_peer_gc_mintime	INTEGER	Minimum interval between garbage collection passes. This interval is in effect under high memory pressure on the pool. Measured in jiffies.
inet_peer_gc_maxtime	INTEGER	Minimum interval between garbage collection passes. This interval is in effect under low (or absent) memory pressure on the pool. Measured in jiffies.
TCP		
tcp_syn_retries	INTEGER	Number of times initial SYNs for an active TCP connection attempt will be retransmitted. Should not be higher than 255. Default value is 5, which corresponds to ~180seconds.
tcp_synack_retries	INTEGER	Number of times SYN/ACKs for a passive TCP connection attempt will be retransmitted. Should not be higher than 255. Default value is 5.
tcp_keepalive_time	INTEGER	How often TCP sends out keepalive messages when keepalive is enabled. Default is two hours.
tcp_keepalive_probes	INTEGER	How many keepalive probes TCP sends out, until it decides that the connection is broken. Default is 9.
tcp_keepalive_interval	INTEGER	How frequently the probes are send out. Multiplied by tcp_keepalive_probes it is time to kill not responding connection, after probes started. Default value is 75 seconds, meaning connection will be aborted after 11 minutes of retries.
tcp_retries1	INTEGER	How many times to retry before deciding that something is wrong and it is necessary to report this suspicion to network layer. Minimal RFC value is 3, it is default, which corresponds to three seconds to 8 minutes depending on RTO.
tcp_retries1	INTEGER	How may times to retry before killing alive TCP connection. RFC1122 says that the limit should be longer than 100 sec. It is too small number. Default value of 15 corresponds to 13 to 30 minutes depending on RTO.
tcp_orphan_retries	INTEGER	How may times to retry before killing TCP connection, closed by our side. Default value is 7, which corresponds to 50 seconds to 16 minutes depending on RTO. If you machine is loaded WEB server, you should think about lowering this value, such sockets may consume significant resources.
tcp_fin_timeout	INTEGER	Time to hold socket in state FIN-WAIT-2, if it was closed by our side. Peer can be broken and never close its side, or even died unexpectedly. Default value is 60 seconds. Usual value used in 2.2 kernel was 180 seconds. This may be restored but remember that if your machine is even underloaded WEB server, you risk to overflow memory with kilotons of dead sockets, FIN-WAIT-2 sockets are less dangerous than FIN-WAIT-1, because they eat maximum 1.5K of memory, but they tend to live longer.
tcp_max_tw_buckets	INTEGER	Maximal number of timewait sockets held by system simultaneously. If this number is exceeded time-wait socket is immediately destroyed and warning is printed. This limit exists only to prevent simple DoS attacks, you must not lower the limit artificially, but rather increase it (probably, after increasing installed memory), if network conditions require more than default value.
tcp_tw_recycle	BOOLEAN	Enable fast recycling TIME-WAIT sockets. Default value is 1. It should not be changed without advice/request of technical experts.
tcp_max_orphans	INTEGER	Maximal number of TCP sockets not attached to any user file handle, held by system. If this number is exceeded orphaned connections are reset immediately and warning is printed. This limit exists only to prevent simple DoS attacks, you must not rely on this or lower the limit artificially, but rather increase it (probably, after increasing installed memory), if network conditions require more than default value, and tune network services to linger and kill such states more aggressively. Let me to remind again: each orphan eats up to 64K of unswappable memory.
tcp_abort_on_overflow	BOOLEAN	If listening service is too slow to accept new connections, reset them. Default state is FALSE. It means that if overflow occurred due to a burst,

		connection will recover. Enable this option _only_ if you are really sure that listening daemon cannot be tuned to accept connections faster. Enabling this option can harm clients of your server.
tcp_syncookies	BOOLEAN	Only valid when the kernel was compiled with CONFIG_SYNCOOKIES. Send out syncookies when the syn backlog queue of a socket overflows. This is to prevent against the common syn flood attack Default is FALSE. syncookies is fallback facility. It MUST NOT be used to help highly loaded servers to stand against legal connection rate. If you see synflood warnings in your logs, but investigation shows that they occur because of overload with legal connections, you should tune another parameters until this warning disappear. Syncookies seriously violate TCP protocol, do not allow to use TCP extensions, can result in serious degradation of some services (SMTP relaying), visible not by you, but your clients and relays, contacting you. While you see synflood warnings in logs not being really flooded, your server is seriously misconfigured.
tcp_stdurg	BOOLEAN	Use the Host requirements interpretation of the TCP urg pointer field. Most hosts use the older BSD interpretation, so if you turn this on Linux might not communicate correctly with them. Default is FALSE
tcp_max_syn_backlog	INTEGER	Maximal number of remembered connection requests.
tcp_window_scaling	BOOLEAN	Enable window scaling as defined in RFC1323.
tcp_timestamps	BOOLEAN	Enable timestamps as defined in RFC1323.
tcp_sack	BOOLEAN	Enable select acknowledgments (SACKS).
tcp_dsack	BOOLEAN	Allows TCP to send "duplicate" SACKs.
tcp_fack	BOOLEAN	Enable FACK congestion avoidance and fast retransmission. The value is not used, if tcp_sack is not enabled.
tcp_ecn	BOOLEAN	Enable Explicit Congestion Notification in TCP.
tcp_reordering	INTEGER	Maximal reordering of packets in a TCP stream. Default is 3.
tcp_retrans_collapse	BOOLEAN	Bug-to-bug compatibility with some broken printers. On retransmit try to send bigger packets to work around bugs in certain TCP stacks.
tcp_wmem	VECTOR	Amount of memory reserved for send buffers for TCP socket. Each TCP socket has rights to use it due to fact of its birth. Default: is 128K.
tcp_rmem	VECTOR	Minimal sizes of receive buffer used by TCP sockets. It is guaranteed to each TCP socket, even under moderate memory pressure. Default: is 8K.
tcp_mem	VECTOR	Below this number of pages TCP is not bothered about its memory appetite.
tcp_app_win	INTEGER	Reserve max(window/2^tcp_app_win, mss) of window for application buffer. Value 0 is special, it means that nothing is reserved. Default is 31
tcp_adv_win_scale	INTEGER	Count buffering overhead as bytes/2^tcp_adv_win_scale (if tcp_adv_win_scale > 0) or bytes-bytes/2^(-tcp_adv_win_scale), if it is <= 0. Default is 2
tcp_rfc1337	BOOLEAN	If set, the TCP stack behaves conforming to RFC1337. If unset, we are not conforming to RFC, but prevent TCP TIME_WAIT assassination. Default is 0.
ICMP		
icmp_echo_ignore_all	BOOLEAN	
icmp_echo_ignore_broadcasts	BOOLEAN	If either is set to true, then the kernel will ignore either all ICMP ECHO requests sent to it or just those to broadcast/multicast addresses, respectively.
icmp_destunreach_rate	INTEGER	
icmp_paramprob_rate	INTEGER	
icmp_timeexceed_rate	INTEGER	
icmp_echoreply_rate	INTEGER	Limit the maximal rates for sending ICMP packets to specific targets. 0 to disable any limiting, otherwise the maximal rate in jiffies.
icmp_ignore_bogus_error_responses	BOOLEAN	Some routers violate RFC 1122 by sending bogus responses to broadcast frames. Such violations are normally logged via a kernel warning. If this is set to TRUE, the kernel will not give such warnings, which will avoid log file clutter. Default is FALSE.
IGMP		
igmp_max_memberships	INTEGER	Change the maximum number of multicast groups we can subscribe to. Default is 20.
/proc/sys/net/conf/interface/* /proc/sys/net/conf/all/*		

log_martians	BOOLEAN	Log packets with impossible addresses to kernel log.
accept_redirects	BOOLEAN	Accept ICMP redirect messages. Default is TRUE for host and FALSE for router.
forwarding	BOOLEAN	Enable IP forwarding on this interface.
mc_forwarding	BOOLEAN	Do multicast routing. The kernel needs to be compiled with CONFIG_MROUTE and a multicast routing daemon is required.
proxy_arp	BOOLEAN	Do proxy arp.
shared_media	BOOLEAN	Send(router) or accept(host) RFC1620 shared media redirects. Overrides ip_secure_redirects. Default is TRUE.
secure_redirects	BOOLEAN	Accept ICMP redirect messages only for gateways, listed in default gateway list. Default is TRUE.
send_redirects	BOOLEAN	Send redirects, if router. Default is TRUE.
bootp_relay	BOOLEAN	Accept packets with source address 0.b.c.d destined not to this host as local ones. It is supposed, that BOOTP relay daemon will catch and forward such packets. Default is FALSE.
accept_source_route	BOOLEAN	Accept packets with SRR option. Default TRUE for routers and FALSE for hosts.
rp_filter	BOOLEAN	1 = do source validation by reversed path, as specified in RFC1812 Recommended option for single homed hosts and stub network routers. Could cause troubles for complicated (not loop free) networks running a slow unreliable protocol (sort of RIP), or using static routes. 0 = No source validation. Default value is 0.
IPv6 /proc/sys/net/conf/interface/* /proc/sys/net/conf/all/*		
accept_ra	BOOLEAN	Accept Router Advertisements; autoconfigure using them. Functional default: enabled if local forwarding is disabled. Disabled if local forwarding is enabled.
accept_redirects	BOOLEAN	Accept Redirects. Functional default: enabled if local forwarding is disabled. Disabled if local forwarding is enabled.
autoconf	BOOLEAN	Configure link-local addresses using L2 hardware addresses. Default is TRUE.
dad_transmits	INTEGER	The amount of Duplicate Address Detection probes to send. Default is 1.
forwarding	BOOLEAN	By default, Host behavior is assumed.
hop_limit	INTEGER	Default Hop Limit to set. Default is 64.
mtu	INTEGER	Default Maximum Transfer Unit Default is 1280 (IPv6 required minimum).
router_solicitation_delay	INTEGER	Number of seconds to wait after interface is brought up before sending Router Solicitations. Default is 1.
router_solicitation_interval	INTEGER	Number of seconds to wait between Router Solicitations. Default is 4.
router_solicitations	INTEGER	Number of Router Solicitations to send until assuming no routers are present. Default is 3.

STEP 2: HARDEN BOOT ENVIRONMENT

1. **Harden Boot Loader**

   The steps for hardening the boot loader are quite different depending on whether LILO or GRUB has been installed on the system. Since this guide has elected to use LILO, follow these steps to harden the Linux Loader (LILO):

   - To enable boot time password protection, edit the **/etc/lilo.conf** file and add the following lines to the "global section" (**RMM:F:6**):

     restricted
     password=<secret-password-selected>

The restricted keyword will only require a password when entering non-standard boot commands, and is important for permitting non-interactive administrative system reboots, or for quick system outage recovery.

- To enable "safe" kernel booting from **/safe** primary partition, edit the **/etc/lilo.conf** file and add the following lines to the "image section"

```
image = /safe/vmlinuz
    label = safe
    append = "hdc=ide-scsi hdclun=0 splash=silent"
    initrd = /safe/initrd
    root = /dev/hda3
    vga = 0x314
```

- Once complete, the lilo.conf file for the SUSE Linux firewall should appear similar to the example below:

```
boot = /dev/hda
change-rules
   reset
default = linux
lba32
menu-scheme = Wg:kw:Wg:Wg
message = /boot/message
restricted
password = "secret-password"
prompt
read-only
timeout = 80

image = /boot/vmlinuz
   label = linux
   append = "hdc=ide-scsi hdclun=0"
   initrd = /boot/initrd
   root = /dev/hda3
   vga = 0x314

image = /boot/vmlinuz-2.4.25
   label = linux-2.4.25
   append = "hdc=ide-scsi hdclun=0"
   initrd = /boot/initrd
   root = /dev/hda3
   vga = 0x314

image = /safe/vmlinuz
   label = safe
   append = "hdc=ide-scsi hdclun=0"
   initrd = /safe/initrd
   root = /dev/hda3
   vga = 0x314
```

- Set proper ownerships and permissions on /etc/lilo.conf (**RMM:G:14**):

```
chown root:root /etc/lilo.conf
chmod 600 /etc/lilo.conf
```

- Set boot script permissions by editing the **/etc/permissions** file and adding the following line:

  /etc/boot root.root 755

- Reinstall the Bootloader in the MBR: **/sbin/lilo**

2. **Harden Initialization Table (/etc/inittab) and Login Daemons**
   Edit the system initialization table (**/etc/inittab file**) and complete the following:

   - Add the following line to require root password entry upon entering "single user" or "maintenance" mode:

     sum:S:wait:/sbin/sulogin

   - Add the following line to prevent anonymous [ctrl-alt-del] shutdowns and system reboots:

     ca::ctrlaltdel:/usr/sbin/logger –p authpriv:info 'ctl-alt-delete trapped'

   - Disable all "**mingetty**" processes not referencing the standard system console by commenting or removing:

     ```
     ###################################
     # SET GETTY LOGINS
     ###################################
     1:2345:respawn:/sbin/mingetty --noclear tty1
     # 2:2345:respawn:/sbin/mingetty tty2
     # 3:2345:respawn:/sbin/mingetty tty3
     # 4:2345:respawn:/sbin/mingetty tty4
     # 5:2345:respawn:/sbin/mingetty tty5
     # 6:2345:respawn:/sbin/mingetty tty6
     # S0:12345:respawn:/sbin/agetty -L 9600 ttyS0 vt102
     ###################################
     ```

   The complete **/etc/inittab** file for the SUSE Linux firewall is shown below:

```
#########################################################################
# /etc/inittab
#########################################################################
###################################
# DEFAULT RUNLEVEL
###################################
id:5:initdefault:

###################################
# FIRST SCRIPT TO BE EXECUTED
###################################
si::bootwait:/etc/init.d/boot
```

```
################################
# RUN LEVEL HANDLING
################################
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
# l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

################################
# SINGLE USER MODE
################################
ls:S:wait:/etc/init.d/rc S
sum:S:wait:/sbin/sulogin

################################
# SET CTRL-ALT-DEL
################################
ca::ctrlaltdel:/usr/sbin/logger –p authpriv:info 'ctl-alt-delete trapped'

################################
# SET SPECIAL KEYBOARD REQUESTS
################################
kb::kbrequest:/bin/echo "Keyboard Request -- edit /etc/inittab to let this work."

################################
# POWER FAILURE HANDLING
################################
pf::powerwait:/etc/init.d/powerfail start
pn::powerfailnow:/etc/init.d/powerfail now
po::powerokwait:/etc/init.d/powerfail stop
sh:12345:powerfail:/sbin/shutdown -h now THE POWER IS FAILING

################################
# SET GETTY PROGRAMS
################################
1:2345:respawn:/sbin/mingetty --noclear tty1
# 2:2345:respawn:/sbin/mingetty tty2
# 3:2345:respawn:/sbin/mingetty tty3
# 4:2345:respawn:/sbin/mingetty tty4
# 5:2345:respawn:/sbin/mingetty tty5
# 6:2345:respawn:/sbin/mingetty tty6
```

3. **Harden Boot Scripts**[18]
   SUSE 9 Linux places all boot scripts by default in the **/etc/init.d** directory and symbolically links them into the appropriate run-level (**rcx.d**) subdirectory. Harden the boot scripts by completing the following steps:

   - Identify and generate a list of all scripts scheduled to run at boot time by running the following command:

     chkconfig –list > /tmp/chkconfig_list.txt

[18] In addition to hardening boot scripts, we will be identifying, and either removing or disabling, any unnecessary services that are installed on the system (**RMM:J:6**).

- Review the chkconfig_list.txt list[19] and disable all unnecessary scripts by running the script below[20]:

```
#!/bin/sh
#########################################################################
# set_boot_off.sh
# disables boot time settings for scripts in /etc/init.d
#########################################################################

#########################################################################
# VARIABLES
#########################################################################
CHKCONFIG="sbin/chkconfig"
TURN_OFF_LIST=" esound joystick ncsd openct rpmconfigcheck gmm SuSEfirewall2*"

for script in $TURN_OFF_LIST
do
      $CHKCONFIG --level 12345 $script off
done
exit 0
```

  Alternatively, these scripts can either be removed or renamed.

- Set boot script permissions by editing the **/etc/permissions** file and adding the following line:

  /etc/init.d root.root 755

4. **System BIOS**
   Enter the system BIOS setup program and disable support for booting from CD-ROM. This assumes, of course, that system installation has been completed.

STEP 3: HARDEN FILE SYSTEM ACCESS CONTROL

The /etc/fstab file was briefly discussed earlier (see Partitioning step for **Operating System Installation**), but in terms of determining and setting the disk partitioning and mounting scheme, and the type of file system. Since the /etc/fstab file determines how partitions and the file system are mounted, it must now be viewed in greater detail, and as a determinate of file basic file system access control. The goals of this step are to:

- Protect system binaries located in **/usr**
- Prevent the introduction of SUID programs and rogue device files
- Permit software installation as necessary

The format of **/etc/fstab** is as follows:

[19] This may seem like a very short list, and indeed it is. The careful selection of software during the install process has significantly reduced what must be dealt with now. For instance, by intention, the super daemon (inetd or xinetd) were not installed (**RMM:J:7**). Additionally, NFS, NIS, and the Portmap daemon were not installed. Had they been installed, they would surely be on this list.

[20] The idea for this script was provided by the courtesy of Mr. Hal Pomeranz.

block-special file mount-loc type options dump-freq pass-number

Table IV "/etc/fstab/File Format" below describes these formatting elements.

TABLE IV
/etc/fstab File Format

Element	Description
block-special-file	The name of the special file or partition on which the file system resides. Must be a block name device.
mount-loc	The directory at which to mount the file system. If the partition will be used for swapping, use / for this field.
type	The type of file system on the partition.
options	This filed consists of one or more options, separated by commas. The type field, above, determines which options are allowed for any given kind of file system. For ignore type entries, this filed is ignored. For local file systems, the options field may include the following keywords, separated by commas: **rw** — **Read-write file system** **ro** — **Read-only file system (cannot be modified)** **suid** — **SUID access mode is permitted** **nosuid** — **SUID access mode is not permitted** **noauto** — **Do not automatically mount this file system** **nodev** — **Device files will not operate** **noexec** — **Disables executable files**
dump-freq	A decimal number indicating the frequency with which this file system should be backed up by the **dump** utility. A value of 1 means backup should occur every day, 2 means every other day, and so on. A value of 0 means that the device is not to be backed up (swap devices for example).
pass-number	A decimal number indicating the order in which **fsck** should check the file system. A pass-number 1 indicates that the file system should be checked first, 2 indicates it should be checked second, and so on. The root file system must have the number of 1. All file system should have higher pass numbers. For optimal performance, two file systems that are on the same disk drive should have different pass numbers. However, file systems on different drives may have the same pass number, letting **fsck** check the two file systems in parallel. This field should be 0 for swap devices.

The focus and concern here is the "options" element. The ideal outcome would be to mount each file system either as "nosuid" or "ro", and utilize the "nodev" option whenever possible (**RMM:G:2**). Unfortunately, due to the location of the /etc directory (which must be writable), the /dev file system, and the /bin and /root directories (which contains critical SUID binaries such as su) on the same partition as the root file system, very little access protection can be applied to the root file system on a SUSE system. However, since the remaining file systems are located on separate partitions, and no "chrooted" environments are used, the following access controls can be added by editing the /etc/fstab file:

1. Set the /tmp file system to "nosuid", "nodev", and "noexec" (**RMM:G:2**).

2. Set the /usr file system to "ro" (**RMM:G:2**).

3. Set the /var, /home, /opt, and /storage file systems to "rw", "nosuid", and "nodev" (**RMM:G:2**).

4. Set the /boot and /safe file systems to "rw", "nosuid", and "nodev" (**RMM:G:2**).

5. Set removable media mount options to “ro”, ”noauto”, ”user”, “nosuid”, and “nodev” (**RMM:G:2**).

STEP 4: HARDEN SYSTEM ACCESS CONTROL

To harden and secure system access to the SUSE 9 Linux firewall, complete the following steps:

1. **Restrict Root Logins** (**RMM:G:6**). To restrict root logins to “main system terminal” only, edit the **/etc/securetty** file and either comment or remove all console references except for **tty1** (/dev/tty1).

2. **Cleanup User Accounts** (**RMM:F:3**) (**RMM:F:7**). Identify, and either remove unnecessary system accounts, or sanitize shell assignment for system accounts. This guide recommends the following script be used for account deletion and modification:

```
#!/bin/sh
#####################################
# clean_accounts.sh
# deletes and cleans up system accounts
#####################################
#####################################
# VARIABLES
#####################################
USERDEL="/usr/sbin/userdel"
USEWRMOD="/usr/sbin/usermod"
OPTIONS=""
BIT_BUCKET="/dev/null"

# user accounts to remove
DEL_USER_LIST="lp mail news uucp games ftp"

# user accounts to set shell to /dev/null
CLEAN_USER_LIST="nobody"

# delete user accounts in $DEL_USER_LIST
for user in $DEL_USER_LIST
do
    $USERDEL -r $user
done

# set shell to /dev/null in system accounts above UID=2 and below UID=500
for user in `awk -F: '( $3 < 500 && $3 > 2 ) { print $1 }' /etc/passwd`
do
    $USERMOD -L -s $BIT_BUCKET $user
done

# set shell to /dev/null for accounts in $CLEAN_USER_LIST
for user in $CLEAN_USER_LIST
do
    $USERMOD -L -s $BIT_BUCKET $user
done
exit 0
```

3. **Set Password Complexity and Aging Policy** (**RMM:F:4**) (**RMM:F:5**). To set password complexity policy, set the following parameters in the **/etc/login.defs** file:

```
PASS_MIN_LEN                8
PASS_MAX_LEN            8
OBSCURE_CHECKS_ENAB     yes
```

To set password aging policy, set the following parameters in the **/etc/login.defs** file:

```
PASS_MAX_DAYS           60
PASS_MIN_DAYS           0
PASS_WARN_AGE           7
```

4. **Disable "rhosts" and ".rhosts" Functionality**. Sub-step 4 is a "Defense-in-Depth measure, as is sub-step 5. Despite the fact the notoriously insecure "R Commands" have not been installed on the Linux firewall, some additional defensive security should be in place to prevent any potential subversion arising from system compromise. The **/etc/hosts.equiv** file, along with any local **$HOME/.rhosts** files, defines the hosts and user accounts that can invoke remote commands on a local host without supplying a password. Therefore, we will disable this potential capability by symbolically linking these files to /dev/null. This guide recommends the following script be used:

```
#!/bin/sh
#########################################################################
# disable_rhosts.sh
# symlinks "r-command" files to /dev/null
#########################################################################

#########################################################################
# VARIABLES
#########################################################################
BIT_BUCKET="/dev/null"
HOME="/hone"
RFILE_LIST="/etc/hosts.equiv $HOME/*/.rhosts $HOME/*/.rhosts  "

for rfile in $RFILE_LIST
do
    ln -s $BIT_BUCKET $rfile
done
exit 0
```

5. **Prevent All FTP Logins**. This Linux firewall does not have the FTP service installed nor is it available. Thus, this sub-step is a defense-in-depth endeavor to set a "DENY ALL" policy for FTP logins. This guide recommends the following script be used to harden and lock-down any potential FTP logins by a secure configuration of the /etc/ftpusers file:

```
#!/bin/sh
####################################
```

```
# harden_ftpusers.sh
####################################
FTPUSERS_FILE="/etc/ftpusers"

if [ ! -e $FTPUSERS_FILE ];
then
                                        touch $FTPUSERS_FILE
fi

# add all system account to ftpusers file to disable all potential ftp logins
for user in `awk -F: '( $3 < 500 || $3 > 6000 ) { print $1 }' /etc/passwd`
do
    echo $user >> $FTPUSERS_FILE
done

# set file ownership and permissions
chown root:root $FTPUSERS_FILE
chmod 600 $FTPUSERS_FILE
exit 0
```

This script accomplishes the following:

- Add all system account users to /etc/ftpusers file, denying them the ability to login through FTP.
- Sets permissions and ownerships on the **/etc/ftpusers** file.

6. **Control cron Access** (**RMM:G:9**).
   On the SUSE Linux firewall, root is the only user that needs access to cron or should have access to cron. The cron.allow (at.allow) contains a list of users that are allowed are allowed to run the **crontab** command to add, remove, or modify cron jobs. When the cron.allow (at.allow) files do not exist, the Linux cron daemon parses the cron.deny (at.deny) files for a list of users who are not permitted to use the crontab command. By default, the cron.* and at.* files are not created and do not exist on a SUSE 9 Linux system. Consequently, several actions must be performed to tighten and secure this subsystem:

   - Create cron.allow and at.allow files and populate with the user root only.
   - Create cron.deny and at.deny files and populate with all system users (users in /etc/passwd) except root.
   - Set permissions and ownerships to root for all cron related files in /etc, and on the /var/spool/cron directory.

   This guide recommends the following script be used:

```
#!/bin/sh
####################################
```

```
# harden_cron.sh
# hardens cron facility
####################################
####################################
# VARIABLES
####################################
CRON_CONF_DIR="/etc"
CRON_CONF_BASE="/etc/cron*"
CRON_ALLOW="$CRON_CONF_DIR/cron.allow"
CRON_DENY="$CRON_CONF_DIR/cron.deny"
AT_ALLOW="$CRON_CONF_DIR/at.allow"
AT_DENY="$CRON_CONF_DIR/at.deny"
CRON_USER_DIR="/var/spool/cron"

####################################
# CREATE MISSING FILES
####################################
touch $CRON_ALLOW
touch $CRON_DENY
touch $AT_ALLOW
touch $AT_DENY

####################################
# POPULATE  FILES
####################################
# populate *.allow with root
echo "root" >> $CRON_ALLOW
echo "root" >> $AT_ALLOW

# populate *.deny files with accounts above UID=0 and below UID=500
for user in `awk -F: '( $3 < 6000 && $3 > 0 ) { print $1 }' /etc/passwd`
do
    echo $user >> $CRON_DENY
    echo $user >> $AT_DENY
done

####################################
# SECURE FILES
####################################
chown -R root:root $CRON_USER_DIR
chmod -R 700 $CRON_USER_DIR
chown -R root:root $CRON_CONF_BASE
chmod -R 700 $CRON_CONF_BASE
exit 0
```

7. **Set Default File and Directory Permissions** (**RMM:G:13**) (**RMM:G:14**).
   SUSE Linux 9 provides a unique and very reliable mechanism to not only set file and directory permissions and ownerships, but to "heal"[21] any changes that may occur outside of the default values that are set. This accomplished through an "**administratively expandable**" series of /etc/permissions.* files. The following files are provided for all installations:

   - **/etc/permissions**: Provides a minimal basic set of permissions and ownerships containing mostly entries for directories. For example, the core file system directories are listed in this manner:

[21] This feature is just one example of why SUSE Linux was selected and trusted for this firewall installation. The "healing" of permissions and ownerships is made possible by the proprietary "chkstat" perl executable provided with the system installation.

```
/                root.root    755
/root            root.root    700
/dev             root.root    755
/bin             root.root    755
/sbin            root.root    755
/lib             root.root    755
/etc             root.root    755
/home            root.root    755
/boot            root.root    755
/opt             root.root    755
/usr             root.root    755
```

- **/etc/permissions.easy**: Provides a set of permissions and ownerships for the remaining system files and directories for low security implementations.

- **/etc/permissions.secure**: Provides a set of permissions and ownerships for the remaining system files and directories for high security implementations.

- **/etc/permissions.paranoid**: Provides a set of permissions and ownerships for the remaining system files and directories extremely high security implementations (guaranteed to break most functionality).

- **/etc/permissions.local**: Local machine settings that are capable of overriding all the above.

All permissions and ownerships are checked again the "configured" databases when the "chkstat" executable is run. Any inconsistencies found are rectified according to the configured settings.

Configure the SUSE 9 Linux firewall as follows:

- Edit the **/etc/sysconfig/security** file and set the following parameters:

```
CHECK_PERMISSIONS="set"
PERMISSION_SECURITY="secure local"
```

- Run the command "**SuSEconfig**" to initially set all permissions and ownerships.

- Schedule the "**/usr/bin/chkstat**" utility to run periodically with cron.

8. **Regularly Purge /tmp File System** (**RMM:G:16**). To keep the /tmp files system free of needless data or possible "rouge programs", edit the **/etc/sysconfig/cron** file and set the following parameters:

```
MAX_DAYS_IN_TMP="10"
TMP_DIRS_TO_CLEAR="/tmp /var/tmp"
```

OWNER_TO_KEEP_IN_TMP="root"
CLEAR_TMP_DIRS_AT_BOOTUP="yes"

STEP 5: HARDEN ENVIRONMENT

The user environment on the Linux firewall will be accomplished by the following actions:

- Setting of default login and environmental parameters in the /etc/login.defs file (**RMM:G:10**).
- Setting safe default environmental variables in the /etc/profile file (**RMM:G:10**).
- Creation of a safe function library for the safeguarding of scripts and executables (**RMM:G:11**) (**RMM:H:1**).

1. Edit the **/etc/login.defs** file and set default PATH and UMASK values:

ENV_PATH	**/usr/local/bin:/usr/bin:/bin**
ENV_ROOTPATH	**/sbin:/bin:/usr/sbin:/usr/bin**
TTYPERM	**0620**
UMASK	**022**

2. Edit the /etc/profile file and set a default trusted path and shell:

```
PATH=/sbin:/usr/sbin:/usr/local/sbin:/bin:/usr/bin:/usr/local/bin
SHELL=/bin/sh
export PATH
export SHELL
```

3. Import the following functions into "root" executed shell scripts to set a safe operating environment, safeguard against potential "race conditions", and negate various elevation of privilege attacks:

   - Function **confirm_root()** runs two separate checks to confirm the script has been executed by user root, and exits with error if either check fails.

```
#####################################################
# Program:          protection.functions.sh
# Programmer:       Mark E. Donaldson
# Date:             March 22, 2004
# Description:      protective functions library for all scripts
#####################################################
#####################################################
# source this function library into all security sensitive scripts
#####################################################
#########################
# FUNCTION CONFIRM ROOT
#########################
confirm_root()
```

```
{
	if [ "$(whoami)" != "root" ] ;
	then
		$ECHO "ROOT CHECK 1 - ERROR: You must be root to run this script." >&2
		$LOGGER -p warning "ROOT CHECK 1 - ERROR: You must be root to run this script."
		exit 1
	else
		$ECHO "ROOT CHECK 1 - EXCELLENT: You are root." >&2
		$LOGGER -p warning "ROOT CHECK 1 - EXCELLENT: You are root."
	fi

	if [ $UID -eq 0 ];
	then
		$ECHO "ROOT CHECK 2 - EXCELLENT: You are root." >&2
		$LOGGER -p warning "ROOT CHECK 2 - EXCELLENT: You are root."
	else
		$ECHO "ROOT CHECK 2 - ERROR: You must be root to run this script." >&2
		$LOGGER -p warning "ROOT CHECK 2 - ERROR: You must be root to run this script."
		exit 1
	fi
}
```

- Function **version_info()** extracts and displays version information to verify that not only the correct script has been executed, but the correct version as well.

```
#####################################################
# FUNCTION VERSION_INFO()
#####################################################
version_info()
{
	VERSION="$1"
	VDATE="$2"
	NAME="Mark E. Donaldson"

	$ECHO "Bandwidthco Scripts -- Version (c) $VERSION ($VDATE) by $NAME"
	$LOGGER –p warning "Bandwidthco Scripts -- Version (c) $VERSION ($VDATE) by $NAME"
}
```

- Function **lock_file()** provides a file locking mechanism for scripts that must work either create a file, or work on an already existing file.

```
#####################################################
# FUNCTION LOCK_FILE()
#####################################################
lock_file()
{
	lockfile="/var/lock/$1"
	[ -f "$lockfile" ] &&
	{
		pid="$(cat "$lockfile")"
		if ps $pid >/dev/null ;
		then
			echo "ABORTING: Another instance is running ($pid)" >&2
			exit 1
		else
			rm -f "$lockfile"
		fi
	}
	echo "$$" > "$lockfile"
}
```

- Function **set_trusted_env()** set the environmental IFS (inter-field separator) variable to the expected trusted value, sets a trusted path and trusted shell, sets a secure UMASK value, and places control on process "core dump" capabilities.

```
#########################
# FUNCTION SET_TRUSTED_ENV
#########################
set_trusted_env()
{
      # set inter-field separator (IFS) to space, tab, newline
      IFS='
      '
      export IFS

      for var in `/usr/bin/env | /usr/bin/cut -f1 -d=`
      do
            [ "$var" != "IFS" ] && unset "$var"
      done

      # set trusted path and shell
      PATH=/sbin:/usr/sbin:/usr/local/sbin:/bin:/usr/bin:/usr/local/bin
      SHELL=/bin/sh
      export PATH
      export SHELL

      # set file mode creation mask
      umask 077

      # core dump control
      ulimit –c 0
}
```

STEP 6: HARDEN XDMCP

It is recommended the GUI logins be disabled in the /etc/inittab file (see **Step 2: Harden Boot Environment**). However, if GUI is permitted and activated, the following steps should be taken:

1. **Disable XDMCP protocol** (**RMM:J:12**). If using X11, edit the /etc/X11/xdm/xdm-config file and either uncomment or add the following line:

   DisplayManager.requestPort: 0

   Next, edit the **/etc/X11/xdm/Xservers** file and add the following line: **- nolisten tcp**

   If using KDM[22], edit the /etc/opt/kde3/share/config/kdm/kdmrc file and either uncomment or add the following lines:

   [Xdmcp]

[22] Although SUSE 9 Linux offers GNOME and many other GUI system options, KDE is the default and preferred installation. Thus, this guide places the focus on KDE. If GNOME is installed instead, the configuration files are located in /opt/gnome2/share/config/gdm/.

Enable=false

Edit the /etc/opt/kde3/share/config/kdm/Xservers file and add the following line:

- nolisten tcp

2. **Prevent Remote Access to Xserver** (**RMM:J:12**). Edit the Xaccess file (for both xdm and kdm), and make the following changes:

   - Negate any host from getting a login window by adding the line: **! ***
   - Permit logins from local terminal only: **terminal-local firewall hostname**
   - Negate any host from getting a chooser by adding the line:

     ! * CHOOSER BROADCAST

   - Permit chooser from local host only by adding the lines:

     %hostlist firewall hostname
     *** CHOOSER %hostlist**

   - When complete, the Xaccess file should look similar to the one below.

```
#########################################################
# HOST ACCESS
#########################################################
! *
terminal-local firewall hostname

#########################################################
# CHOOSER
#########################################################
! *          CHOOSER BROADCAST

%hostlistfirewall hostname
*           CHOOSER %hostlist
#########################################################
```

3. **Control Logins, Account Access, and Passwords** (**RMM:G:6**). KDM has several options that must be properly set to prevent major access security problems. The following option settings are recommended in the **/etc/opt/kde3/share/config/kdm/kdmrc** file:

```
[X-*-Core]
AutoReLogin=false
AllowRootLogin=false
AllowNullPasswd=false

[X-:*-Core]
AllowRootLogin=false
AllowNullPasswd=false
```

```
NoPassEnable=false
NoPassUsers=frick, frack

[X-:0-Core]
AutoLoginEnable=false
# AutoLoginUser= frick, frack
# AutoLoginPass=secret!
# AutoLoginSession=none
AutoLogin1st=false
```

4. **Prevent System Shutdown or Reboot from GUI Terminal** (**RMM:G:3-4-5**). To prevent anonymous or unauthorized system shutdowns or reboots from the KDM terminal, add the following line to the /etc/opt/kde3/share/config/kdm/kdmrc file under **[X-*-Core]**:

   AllowShutdown=None

   This will disable the X Widget shutdown button on the window terminal.

5. **Set KDM Warning Banner** (**RMM:J:11**). To set the warning (greeting) banner on the KDM terminal, add the following line to the /etc/opt/kde3/share/config/kdm/kdmrc file under **[X-*-Greeter]**:

   GreetString=warning banner SUSE Linux 9.0 (%h)

STEP 7: CONFIGURE WARNING BANNERS

The SUSE Linux firewall will rely upon **/etc/motd** and **/etc/issue** for all system warning and security banners (**RMM:F:14**)[23]. Insert the respective text files below into the appropriate banner file:

cat issue.txt > /etc/issue
cat motd.txt > /etc/motd

issue.txt

```
* * * * * * * * * * * * W A R N I N G * * * * * * * * * * * * *
THIS SYSTEM IS RESTRICTED TO AUTHORIZED USERS FOR AUTHORIZED USE
ONLY. UNAUTHORIZED ACCESS IS STRICTLY PROHIBITED AND MAY BE
PUNISHABLE UNDER THE COMPUTER FRAUD AND ABUSE ACT OF 1986 OR
OTHER APPLICABLE LAWS. IF NOT AUTHORIZED TO ACCESS THIS SYSTEM,
DISCONNECT NOW. BY CONTINUING, YOU CONSENT TO YOUR KEYSTROKES
AND DATA CONTENT BEING MONITORED. ALL PERSONS ARE HEREBY
NOTIFIED THAT THE USE OF THIS SYSTEM CONSTITUTES CONSENT TO
MONITORING AND AUDITING.
* * * * * * * * * * * * W A R N I N G * * * * * * * * * * * * *
```

[23] For specifics and X-Windowing banners, refer to section on **Hardening XDMCP**. For specifics and banners for SSH, see section on **Hardening SSH**.

motd.txt

```
##########################################################
**************************************************************************

This is a private system!!! All connection attempts are logged and
monitored. All unauthorized connection attempts will be investigated and
handed over to the proper authorities.

**************************************************************************
##########################################################
```

The following script can be used to both set and test warning banners:

```
#!/bin/sh
####################################
# set_banners.sh
# sets and tests system banners
####################################
####################################
# VARIABLES
####################################
ISSUE_TEXT="/root/bin/issue.txt"
MOTD_TEXT="/root/bin/motd.txt"
ISSUE_FILE="/etc/issue"
MOTD_FILE="/etc/motd"
POUND="############################"

####################################
# SET BANNERS
####################################
cat $ISSUE_TEXT > $ISSUE_FILE
cat $MOTD_TEXT > $MOTD_FILE

####################################
# VIEW BANNERS
####################################
echo "$POUND"
echo "$ISSUE_FILE"
echo "$POUND"
cat $ISSUE_FILE
echo
echo "$POUND"
echo "$MOTD_FILE"
echo "$POUND"
cat $MOTD_FILE
echo
exit 0
```

STEP 8: HARDEN SENDMAIL

Several of the processes and proprietary scripts installed on the SUSE Linux firewall have the need to send email in a secure manner. Thus, sendmail version 8.12.10 has been installed on this machine. At this point, however, we have a relatively insecure default installation, and one that is not suitable for an Internet facing device. This procedure will harden and secure sendmail specifically to the role of this machine. The following factors must be considered:

- The firewall should only be capable of sending mail, and not capable of receiving mail (**RMM:G:15**).

- The firewall will send mail directly to the mail relay server on the service network.

- The default configuration will start a sendmail MTA daemon listening on port 25/tcp.

- All sendmail versions at or above version 8.12.1 have a separate daemon process for the handling of locally generated mail (the MSP[24]), and the receiving and disposal of mail sent to it (the MTA[25]).

- It is not necessary for the MSP to communicate with the local MTA to successfully send or forward mail (**RMM:J:6**), and therefore the MTA listening on port 25/tcp may be disabled.

- By default, the SUSE Linux global system configuration file will control the configuration and startup of sendmail.

This hardening process will be based on these five needs or factors:

1. **Disable "sysconfig" sendmail Control**. Edit the /etc/sysconfig/mail file and make the following changes:

   ```
   MAIL_CREATE_CONFIG="no"
   SMTPD_LISTEN_REMOTE="no"
   ```

2. **Configure the MSP Client**. Create an /etc/mail/mspclient.mc file and add the following lines[26]:

   ```
   include(`/usr/share/sendmail/m4/cf.m4')
   divert(0)dnl
   define(`confCF_VERSION', `Submit')dnl
   define(`__OSTYPE__',`')dnl
   define(`_USE_DECNET_SYNTAX_', `1')dnl
   define(`confTIME_ZONE', `USE_TZ')dnl
   define(`confDONT_INIT_GROUPS', `True')dnl
   FEATURE(`msp', `[192.168.1.1]')dnl
   ```

[24] The MSP (Mail Submission Process) is a new feature added to sendmail version 8.12.1. MSP is a dedicated procedure used by local processes when generating email from the local machine.

[25] The familiar MTA (Mail Transport Agent) daemon performs the dual role. The MTA is responsible for listening on port 25/tcp for incoming mail, but it is also responsible for the periodic flushing of the system mail queue. In other words, the MTA receives mail and arranges for proper delivery of that mail.

[26] A complete example of the file is available in **Appendix E: /etc/mail/mspclient.mc File.**

Such a configuration is the equivalent of the “nullclient” feature in sendmail prior to version 8.12.1.

3. **Compile the MSP Client File**.

   m4 /etc/mail/mspclient.mc > /etc/mail/submit.cf

4. **Edit Sendmail Startup Script to Prevent MTA Daemon Startup**. Remove all lines containing the variables **SENDMAIL_ARGS** and **srvpid**. Specifically, remove the line containing:

   SENDMAIL_ARGS="-L sendmail -Am -bd -q30m -om"

   Change the variable **SENDMAIL_CLIENT_ARGS** to read:

   SENDMAIL_CLIENT_ARGS=”-L sendmail-client –Ac –qp30m”

   The completed file should appear similar to the one shown in **Appendix F: /etc/init.d/sendmailmsp File**.

5. Add the hostname of the SUSE Linux firewall to the **/etc/mail/local-host-names** file of the mail relay server on the service network. This is necessary or else the mail relay will refuse delivery.

6. **Edit the /etc/aliases File**. Remove all unnecessary alias listings (which means most of them).

7. Restart Sendmail: **rcsendmail restart**

STEP 9: HARDEN SSH

Withstanding the NTOP HTTP/HTTPS interface, all remote access (**RMM:K:2**) to the SUSE Linux firewall, and file transfer operations (**RMM:K:3**) to and from the firewall, will be accomplished with SSH. The more secure SSH Version 2 is the protocol of choice for operations on all segments of the network. The primary security concerns in configuring SSH are as follows:

- Reliance upon SSH Protocol 2 only (**RMM:K:1**).
- Permit access only from trusted administrative workstations (**RMM:J:17**).
- Permit no remote root logins (**RMM:G:7**).
- Permit no empty passwords (**RMM:F:4**).
- Permit RSA authentication (**RMM:K:1**).
- Disable R hosts authentication (**RMM:K:7**).
- Disable hosts based authentication (**RMM:K:7**).
- Disable X11 authentication (**RMM:J:12**).

- Disable X11 and TCP forwarding or tunneling (**RMM:K:7**).
- Set login warning banner (**RMM:F:14**).
- Set secure key regeneration interval (**RMM:K:1**).
- Set syslog level and facility (**RMM:F:14**).

Follow these steps to harden and secure SSH:

1. **Edit SSHD Configuration File**. Explicitly define all entries in the sshd_config file and do not rely upon "commented" default. An example sshd_config files is available in **Appendix J: /etc/ssh/sshd_config File**. However, the recommended entries are as follows:

```
Port 22
Protocol 2
ListenAddress 0.0.0.0
HostKey /etc/ssh/ssh_host_key
HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
KeyRegenerationInterval 3600
ServerKeyBits 768
SyslogFacility AUTHPRIV
LogLevel INFO
LoginGraceTime 120
PermitRootLogin no
StrictModes yes
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile  .ssh/authorized_keys
RhostsAuthentication no
IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no
IgnoreUserKnownHosts yes
PasswordAuthentication no
PermitEmptyPasswords no
ChallengeResponseAuthentication no
KerberosAuthentication no
KerberosOrLocalPasswd no
KerberosTicketCleanup no
AFSTokenPassing no
KerberosTgtPassing no
GSSAPIAuthentication no
GSSAPIKeyExchange no
GSSAPIUseSessionCredCache no
PAMAuthenticationViaKbdInt no
X11Forwarding no
X11DisplayOffset 10
X11UseLocalhost no
PrintMotd yes
PrintLastLog yes
KeepAlive no
UseLogin no
UsePrivilegeSeparation yes
PermitUserEnvironment no
Compression yes
MaxStartups 10
VerifyReverseMapping yes
Banner /etc/issue
Subsystem   sftp /usr/lib/ssh/sftp-server
```

2. **Edit SSH Configuration File**. The default configuration should prove adequate for security purposes. However, enhancements and restrictions, such as the control of command execution, may be added as needed or necessary.

3. **Configure TCPWrappers** (**RMM:J:5**). The OpenSSH Binary provided by the SUSE Linux 9.0 distribution is compiled with tcpwrappers library support. Thus, we must configure the hosts.deny and hosts.allow files for the SSH functionality needed. Edit the hosts.deny file and insert and "default deny" for ALL hosts and services:

```
####################################
# /etc/hosts.deny
####################################
ALL: ALL : DENY
```

Next, edit the hosts.allow file and add entries to permit sshd access from trusted administrative workstations (**RMM:J:17**) but deny X11 port forwarding:

```
###################################
# /etc/hosts.allow
###################################
sshd : 192.168.1.3, 192.168.0.6 : ALLOW
sshdfwd-X11 : ALL : DENY
```

4. **Add Firewall Rules**. Add restrictive rules to firewall script permitting access to tcp port 22 only from trusted workstations (**RMM:J:2**) (**RMM:J:17**).

5. **Accommodate Windows SSH Client**. One additional problem to resolve is the tendency for OpenSSH to reject (or fail to acknowledge) public keys generated by Windows based SSH clients. Since remote access to the SUSE firewall, and all remote file transfers and copies to and from the firewall will be with connections from trusted Windows clients, this issue must be resolved. PuTTy is used for our example:

   Step 1: Login to the Linux firewall and su to root. Generate a passphrase protected ssh2 key-pair with the "ssh-keygen" command and add the public key to the ~/.ssh/authorized_keys file:

   ```
   # ssh-keygen -t rsa -b 1024
   # cat id_rsa.pub >> authorized_keys
   ```

   Step 2: Log into the Linux firewall from the Linux firewall using the new key-pair and enter the passphrase manually to check that it works as expected:

   ```
   # ssh firewall1
   ```

   Step 3: On the trusted Windows workstation, run "puttygen" and import the private key (id_rsa) generated on the Linux firewall. Enter the passphrase and save as an ssh2 key in putty's own format (putty-rsa.PPK).

   Step 4: Run "pageant" and load the private key generated by "puttygen". Enter the passphrase when asked.

   Step 5: Test the key by SSH to Linux box. If working properly, no passphrase or password should be requested.

STEP 10: HARDEN NTP

Proper NTP configuration is critical for secure clock synchronization over the network infrastructure (**RMM:J:15**). The NTP architecture used for this network follows these general guidelines:

- Stratum 2 Servers: designated NTP Servers (through configuration) on Internet with which the central firewall and other stratum 3 servers are to synchronize.
- Stratum 3 Servers: the central SUSE Linux firewall and other NTP servers on the service and administrative network segments.
- Stratum 4 Servers: all Linux machines on the internal network segment, and the Windows First-Tree-In-Forest Domain Controller.
- Stratum 5 Servers: All Windows Machines except First-Tree-In-Forest Domain Controller.
- Stratum 3 servers set to synchronize with stratum 2 servers.
- Stratum 4 servers set to synchronize with stratum 3 servers.
- Stratum 5 servers set to synchronize with Windows First-Tree-In-Forest Domain Controller.

Harden this NTP configuration as follows. The complete ntp.conf file is available in **Appendix D: ntp.conf File**.

1. Select stratum 2 NTP servers to be used for network time synchronization[27].
2. Add stratum 2 NTP servers to the ntp.conf file using the "server" keyword. For example:

   server ntp2.usno.navy.mil prefer
   server tock.usno.navy.mil
   server tick.usno.navy.mil
   server cuckoo.nevada.edu

3. Add peer stratum 3 NTP servers to the ntp.conf file using the "peer" keyword. For example:

   peer 192.168.1.1
   peer 192.168.1.2
   peer 192.168.2.1
   peer 192.168.2.2
   peer 192.168.3.1
   peer 192.168.3.2

4. Set local clock as "pseudo clock" in the ntp.conf file:

[27] As a common courtesy, prior permission should be requested and obtained from the administrators of the selected stratum 2NTP servers.

```
server 127.127.1.0
fudge  127.127.1.0 stratum 5
```

5. Set “Security Policy” for NTP server access in the ntp.conf file:

```
#################
# SECURITY POLICY
#################
# default policy is to ignore all
# restrict default          ignore

# Permit synchronization with Stratum 2 NTP Servers
# but do not let them query this server or modify this configuration file
restrict 192.5.41.209     nomodify noquery
restrict 192.5.41.41      nomodify noquery
restrict 192.5.41.40      nomodify noquery
restrict 131.216.1.101    nomodify noquery

# Permit synchronization with peer Stratum 3 NTP Servers
# but do not let them query this server or modify this configuration file
restrict 192.168.1.1      nomodify noquery
restrict 192.168.1.2      nomodify noquery
restrict 192.168.2.1      nomodify noquery
restrict 192.168.2.2      nomodify noquery
restrict 192.168.3.1      nomodify noquery
restrict 192.168.3.2      nomodify noquery

# Allow administration, query, and debugging of Stratum 3 Server from the 192.168.0.0/24 network only
restrict 192.168.0.0 mask 255.255.255.0      nomodify

# place no restrictions on loopback address
restrict 127.0.0.1
```

6. Set authentication keys in the ntp.conf file (Optional).

7. Set syslog logging configuration and driftfile in the ntp.conf file:

```
driftfile /var/lib/ntp/ntp.drift
logfile   /var/log/ntp/ntp
```

8. Edit the /etc/sysconfig/xntp file and add the following startup options:

```
XNTPD_INITIAL_NTPDATE="AUTO-2"
XNTPD_OPTIONS="-U ntp"
```

   The "AUTO-2" startup value forces query of the first two NTP servers listed in the configuration file for the current time and date before the xntpd is started. The "-U" option sets the xntp daemon to run as user “ntp” instead of root.

9. Set boot time initialization script: **chkconfig –level 35 on**

10. Restart the xntp daemon: **./etc/init.d/xntp restart**.

STEP 11: HARDEN SUDO

The SUSE Linux firewall administrators, by policy, must execute all commands as root with SUDO. This authority must be granted by appropriate entries in the /etc/sudoers file. The following is an example that may be applied for the Linux firewall setup:

```
Host_Alias FIREWALLS = server3,server6
User_Alias FWADMINS = fwadmin1,fwadmin2
Cmnd_Alias     SHELLS = /bin/sh, /bin/csh, /bin/bash, /bin/ksh
Cmnd_Alias     BACKUP = /bin/mt, /sbin/restore, /sbin/dump
Cmnd_Alias     SHUT = /usr/sbin/shutdown, /usr/sbin/halt, /usr/sbin/reboot
Cmnd_Alias     FW = /usr/local/sbin/iptables, /usr/bin/iptables
root     ALL=(ALL) ALL
FWADMINS     FIREWALLS =(ALL)     ALL
FWADMINS     FIREWALLS =(ALL)     NOPASSWD: FW
```

A complete example **/etc/sudoers** file is available in **Appendix C: /etc/sudoers File**.

STEP 12: HARDEN SYSLOG

The hardening process for syslog consists of the three following tasks:

- Provide syslog with the ability to log remotely to a syslog server. In the event the firewall is compromised, infected, or intruded upon, remote logging will provide an additional set of log records for use or reference if needed (**RMM:I:2**).

- Prevent the syslog daemon on the firewall from accepting logging from any remote hosts. Failure to do so exposes the firewall to a potential DOS attack on the syslog service itself and the /var file system (**RMM:L:6**).

- Enable a logical and intuitive partitioning of logging records permitting both friendly and functional analysis (**RMM:G:19**) and (**RMM:J:13**).

Follow the steps below:

1. **Configure Remote Logging Capability**. To enable, open the /etc/sysconfig/syslog file and add the **–h** switch to the **SYSLOGD_PARAMS** variable.

   SYSLOGD_PARAMS="-h"

   Remove the **–r** switch (if present) within the same variable to disable logging from a remote host or server.

   Next, edit the /etc/syslog.conf file and add a line containing the host name or IP address of the remote syslog server on the administrative network segment:

   ***.* @192.168.2.1**

2. **Enable Partitioned Logging**. Edit the syslog configuration file for partitioned logging. A suggested format is available in **Appendix B: File /etc/syslog.conf**.

3. **Create Logging File System**. This guide suggests using a script similar to the following to accomplish this:

```
#!/bin/sh
#########################################################################
# set_logging.sh
#########################################################################
ROOT_DIR="/"
ROOT_LOG_DIR="/var/log"
STORAGE_LOG_DIR="/storage/logs"
FILE=""
DIR=""
#########################################################################
echo "Checking and Creating Needed Log Directories"
cd $ROOT_LOG_DIR

# snort
if [ ! -d $ROOT_LOG_DIR/snort ];
then
        mkdir snort
        chown root root snort
        chmod 640 snort
fi
# iptables
if [ ! -d $ROOT_LOG_DIR/iptables ];
then
        mkdir iptables
        chown root root iptables
        chmod 640 iptables
fi
# iptraf
if [ ! -d $ROOT_LOG_DIR/iptraf ];
then
        mkdir iptraf
        chown root root iptraf
        chmod 640 iptraf
fi
# tcpdump
if [ ! -d $ROOT_LOG_DIR/tcpdump ];
then
        mkdir tcpdump
        chown root root tcpdump
        chmod 640 tcpdump
fi
# mail
if [ ! -d $ROOT_LOG_DIR/mail ];
then
        mkdir mail
        chown root root mail
        chmod 640 mail
fi
# all
if [ ! -d $ROOT_LOG_DIR/all ];
then
        mkdir all
        chown root root all
        chmod 640 all
fi
# security
if [ ! -d $ROOT_LOG_DIR/security ];
then
```

```
        mkdir security
        chown root root security
        chmod 640 security
fi
# cron
if [ ! -d $ROOT_LOG_DIR/cron ];
then
        mkdir cron
        chown root root cron
        chmod 640 cron
fi
# ntp
if [ ! -d $ROOT_LOG_DIR/ntp ];
then
        mkdir ntp
        chown root root ntp
        chmod 640 ntp
fi
```

4. Restart the syslog daemon: **./etc/init.d/syslog restart**

PHASE IV: APPLY FIREWALL RULESET

This guide is not about building a firewall rule set[28] rather, it is about the securing of the Linux Firewall device itself. Consequently, it is not the intent of this guide to develop an actual rule set, or even recommend a rules set. However, this guide is about to present a detailed and comprehensive list of items that are either recommended good practices or essentials that should be applied in practice when developing or building the Linux firewall rule set.

Although this list is numbered, this is not to imply there is significance to their order. Each item should be considered for its individual importance and not its relative importance.

1. Understand that the Linux Firewall consists of two individual pieces of software, netfilter and iptables. Netfilter is the "kernel space" program that actually performs the firewall functions whether they are packet filtering, packet mangling, state tracking, or network address translation. Iptables is the "user space" program that interfaces with netfilter, and allows netfilter to be controlled.

2. Load the firewall from script. There are a number of good reasons for this:

   - The security mechanisms of a script can be controlled, just as they can be for any other script that is run on the system (**SMM:G:1**).

   - Scripts are flexible in how they organize, present, and process data.

[28] For information on firewall rule set development, I recommend reading either Building Internet Firewalls by Zwicky, Cooper, and Chapman, or Firewalls and Internet Security, 2nd Edition by William R. Cheswick. For a very quick tutorial on the subject, I recommend "Building Your Firewall Rule Base" by Lance Spitzner at **http://www.enteract.com/~lspitz/rules.html**.

- Scripts guarantee absolute repeatability.

- Scripts can be easily and quickly modified.

3. Source function libraries into the main firewall script to make it easier to read and manage, and to offload busy work.

4. Source pertinent security and protective functions, as discussed earlier in the **Hardening Environment** section, into the main script (**SMM:G:11**).

5. For ease of maintenance, utilize variables within the firewall script whenever possible, as opposed to referencing raw numbers or data. This is particularly important when a specific piece of data is to be repeatedly referenced within various firewall rules. The following is an example:

   - Assume the Linux firewall has four network interfaces, as is the case with the Linux firewall we are securing in this guide. On the physical level, the interfaces would be addressed as eth0, eth1, eth2, and eth3 by the system.

   - Assume each network interface is referenced in 200 rules, and then assume the network topology must be changed. Consequence: 800 firewall rules must be uniquely changed.

   - Now assume variables had been assigned to the four network interfaces as INET_IFACE=”eth0”, INTERNAL_IFACE=”eth1, SERVICE_IFACE=”eth2”, and ADMIN_IFACE=”eth3”, and then assume the network topology must be changed. Consequence: 4 variables must be changed and no firewall rules must be changed.

   When variables are used in conjunction with IF statements, the script will assume limited dynamic capabilities as well. For instance, the same script could be run on a firewall failover (backup) device containing a disparate set of IP addresses:

```
#######################################
# MAIN FIREWALL INTERFACES IP (SERVER6)
#######################################
if [ "$HOSTNAME" = "server6" ] && [ "$SERVER_NAME" = "server6" ];
then
      FW_LO_IP="127.0.0.1"
      FW_INTERNAL_IP="192.168.0.10"
      FW_SERVICE_IP="192.168.1.10"
      FW_ADMIN_IP="192.168.2.10"
      FW_HONEY_IP="192.168.3.10"
      FW_INET_IP="aaa.bbb.ccc.ddd"
      FW_GATEWAY_IP=" aaa.bbb.ccc.ddd "
      FW_PERIMETER_IP="172.16.0.10"
```

```
fi

#####################################
# BACKUP FIREWALL INTERFACES IP (SERVER3)
#####################################
if [ "$HOSTNAME" = "server3" ] && [ "$SERVER_NAME" = "server3" ];
then
        FW_LO_IP="127.0.0.1"
        FW_INTERNAL_IP="192.168.0.2"
        FW_SERVICE_IP="192.168.1.2"
        FW_ADMIN_IP="192.168.2.2"
        FW_HONEY_IP="192.168.3.2"
        FW_INET_IP="aaa.bbb.ccc.ddd"
        FW_GATEWAY_IP="aaa.bbb.ccc.ddd"
        FW_PERIMETER_IP="172.16.0.2"
fi
```

The identical principal of variables will apply to the network layer for such things as IP address assignments, TTL assignments, and TOS assignments for example, the transport layer with port assignments, and at the application layer for such things as logging assignments and string matching assignments. The possibilities are numerous, if not endless.

The moral is this. The failure to use variables equates to complexity, and complexity equates to error. The use of variables equates to simplicity, and simplicity equates to accuracy. When it comes to security, accuracy is the only acceptable result.

6. Implement a restrictive "Deny All" default security policy (-P) for the firewall FILTER table "FORWARD", "INPUT", and "OUTPUT" chains (**RMM:J:1**)[29]:

```
$IPT -t filter -P OUTPUT DROP
$IPT -t filter -P INPUT DROP
$IPT -t filter -P FORWARD DROP
```

7. **Critical**. To avoid major headaches, always set NAT tables with ACCEPT policies.

```
$IPT -t nat -P PREROUTING ACCEPT
$IPT -t nat -P OUTPUT ACCEPT
$IPT -t nat -P PREROUTING ACCEPT
```

[29] Firewall default policy theory describes both a "permissive" and a "restrictive" approach that can be used when setting policy defaults. The permissive approach essentially says to "permit any service unless it is expressly denied". The restrictive theory states that the policy should "deny any service unless it is expressly permitted". Firewalls that implement the permissive approach allow all services to pass into the site by default unless with the service-access policy has identified it as "disallowed". Firewalls that implement the restrictive approach deny all services by default, but then pass those services that have been identified as "allowed".

8. **Good Practice**. To avoid minor headaches, always FLUSH tables and ZERO counters prior to loading prior to loading firewall rule sets. This can easily be done by inserting the following function into the firewall script:

```
##########################
# FLUSH()
##########################
flush()
{
   if [ "$USE_RAW_TABLE" = "1" ];
    then
        for i in filter nat mangle raw
        do
            $ECHO "FLUSHING $i TABLE ....."
            $LOGGER -p warning "FLUSHING $i TABLE ....."
            $IPT -t $i -F
            $IPT -t $i –X
            $IPT -t $i -Z
        done
    fi

    if [ "$USE_RAW_TABLE" = "0" ];
    then
         for i in filter nat mangle
         do
            $ECHO "FLUSHING $i TABLE ....."
            $LOGGER -p warning "FLUSHING $i TABLE ....."
            $IPT -t $i -F
            $IPT -t $i –X
            $IPT -t $i -Z
         done
    fi
}
```

9. Lock down remote access to the Linux firewall by use of a tightly controlled INPUT chain rule set. Access should only be permitted from a few select trusted workstations (**RMM:J:2**).

10. Run all commands within the firewall script with SUDO (**RMM:F:12**). In doing so, the script can only executed by users with the correct permissions, and each command executed is recorded and logged by default. The following is an example:

```
IPT="/usr/sbin/iptables"
SUDO="/usr/bin/sudo"

$SUDO $IPT -A PORTSCAN -p tcp --tcp-flags ALL SYN,FIN -j LOG --log-level $LOG_LEVEL --log-prefix "SF SCAN”
$SUDO $IPT -A PORTSCAN -p tcp --tcp-flags ALL SYN,FIN -j REJECT --reject-with icmp-host-unreachable
```

9. Set tunable kernel TCP/IP stack related parameters from within the firewall script, but prior to loading the firewall rule sets (see section on **Hardening Linux Kernel and TCP/IP Stack**). This assures that the firewall policy itself has set the desired parameter, and that another system, user, or external process has not set it.

10. Disable ip_forwarding from within the firewall script prior to the loading of the firewall rule set:

```
# disable ip forwarding
echo "0" > /proc/sys/net/ipv4/ip_forward
```

This is to protect hosts on the network from being attacked while the rule set is loading and prior to the rule set policies taking effect. As no packet can be routed or forwarded, malicious packets will automatically be rejected. With PortSentry and TCPWrappers installed and properly configured, the Linux firewall itself is protect from attack during this same critical time period (see section on **PortSentry** installation).

Once the firewall rules have been loaded, enable packet-forwarding capability:

```
# enable ip forwarding
echo "1" > /proc/sys/net/ipv4/ip_forward
```

11. Utilize all of the "built-in" security that iptables provides. Each iptables rule permits one reference to each of the following:

- Incoming interface
- Outgoing interface
- Source IP address
- Destination IP address
- Source port
- Destination port

It comes down to this. If all six references are used within a single rule, the rule is "tight" and very secure. If only the minimum essential references are used to permit a packet to pass and "get the job done", then the rule is "loose" and very insecure.

For example, assume there is need for a rule to allow an email message to pass from the mail-relay on the service network to the exchange server on the internal network. This could be accomplished with the following rule set:

```
$IPT -t filter -A TCP_RULES -d $INTERNAL_SMTP1 -p tcp  --destination-port 25 -m state --state NEW -j LOG --log-level $LOG_LEVEL --log-prefix "MAIL: RELAY -> EXCHANGE: "

$IPT -t filter -A TCP_RULES -d $INTERNAL_SMTP1 -p tcp  --destination-port 25 -m state --state NEW -j ACCEPT
```

The firewall would pass this packet just fine and the rule would have accomplished its objective. However, this rule set would also pass packets that it

did not intend to pass. For instance, a spammer using “spoofed” IP addresses scanning for mail servers on the internal network just might be successful, and that is certainly not the intent of the rule.

The moral of this story then is “utilize all of the built-in security that iptables provides”. With that in mind, re-write that same rule, but in a manner that will “safely guide the desired packets” through the firewall, and foil scanning and spoofing at the same time:

```
$IPT -t filter -A TCP_RULES -i $FW_SERVICE_IFACE -o $FW_INTERNAL_IFACE -s $SERVICE_SMTP -d $INTERNAL_SMTP1 -p tcp --source-port $UNPRIVPORTS --destination-port 25 -m state --state NEW -j LOG --log-level $LOG_LEVEL --log-prefix "MAIL: RELAY -> EXCHANGE: "

$IPT -t filter -A TCP_RULES -i $FW_SERVICE_IFACE -o $FW_INTERNAL_IFACE -s $SERVICE_SMTP -d $INTERNAL_SMTP1 -p tcp --source-port $UNPRIVPORTS --destination-port 25 -m state --state NEW -j ACCEPT
```

12. Consider placing any rule or rule set that may to more than one “built-in” chain in its own “user-defined” chain. The fewer rules there are in a firewall rule set, the more efficiently the firewall will be able to process each packet. Therefore, it is much better to jump (-j) to a rule, then to re-write the rule several times.

13. Isolate firewall log messages into a separate or dedicated file. By default, netfilter logs to the **kern.info** syslog facility. This places all the firewall log messages into /var/log/messages along with all other kernel messages. This behavior is not exceedingly friendly for firewall log parsing and analysis. However, since the Linux kernel logs very little by default at the “debug” level, there is an easy solution. Follow these steps:

    - Set logging level to “debug” in the firewall script:

      LOG_LEVEL="debug"

    - Place the LOG_LEVEL variable in rule sets for all packets to be logged:

      -j LOG --log-level $LOG_LEVEL

    - Tell syslog to log only kernel.debug messages to the firewall log file:

      kern.=debug /var/log/iptables/iptables.log

    - Tell syslog not to place firewall messages into /var/log/messages:

      ***.*;kern.!=debug /var/log/messages**

    - Restart syslog: **./etc/init.d/syslog restart**

14. Whenever it is necessary to apply the same firewall rule a list of similar items, allow the firewall script to process and create the list from file, rather than repeatedly creating the same rule numerous times. For example, assume there are 500 IP addresses that need to be "blackholed" and dropped by the firewall. One way to do this would be to create a "DROP" rule for each address. Frankly, that may involve a bit of tedium, and then be a maintenance nightmare thereafter.

    Alternatively, place each of the 500 IP addresses, one per line, in an ASCII text file, and permit the script to process the file as such:

```
########################################################################
# BLACKHOLE CRACKERS & BAD GUYS
########################################################################
# file containing blacklisted addresses
BLACKHOLE_LIST="/usr/local/iptables/blackhole.txt"

if [ "$BLACKHOLE" = "1" ] && [ -e "$BLACKHOLE_LIST" ];
then
      while read BLACKHAT;
      do
            $IPT -A FORWARD -i $FW_INET_IFACE -s $BLACKHAT -m state --state NEW -j LOG --log-level
            $LOG_LEVEL --log-prefix "BLACKHOLE $BLACKHAT: "

            $IPT -A FORWARD -i $FW_INET_IFACE -s $BLACKHAT -m state --state NEW - REJECT --reject-with
            icmp-host-unreachable

if [ "$BLACKHOLE" = "1" ] && [ ! -e "$BLACKHOLE_LIST" ];
then
      $ECHO "UNABLE TO LOCATE OR READ $BLACKHOLE_LIST: RULES NOT LOADED"
      $LOGGER -p warning "UNABLE TO LOCATE OR READ $BLACKHOLE_LIST: NOT LOADED"
fi
```

    The moral of this story is this: It is much easier to change, revise, maintain and control a single file than it is several hundred firewall rules.

15. Utilize the "stateful" capabilities of netfilter/iptables for a more secure and efficient firewall. Filtering decisions will then be based on "policy" alone instead of reliance upon arbitrary port numbers or header flags. For instance, instead of permitting FTP connections by opening the complete range of high ports as such:

```
$IPT -A FORWARD -i $FW_INET_IFACE --destination $FTP_SERVER -p tcp --dport 1024:65535 -j ACCEPT
$IPT -A FORWARD --source $FTP_SERVER --source $FTP_SERVER -p tcp --dport 1024:65535 -j ACCEPT
```

    permit them with:

```
modprobe ip_conntrack_ftp
modprobe ip_nat_ftp
```

16. Utilize the internal counters of netfilter/iptables to view the firewall statistics and analyze the effectiveness and ordering of the firewall rules. For instance, the command:

iptables -vv -n -L -t filter

produces a full set of “rule match” statistics for rules in the FILTER table. The following script produces a nicely formatted statistical report for all four tables (filter, nat, mangle, and raw) and mails it to the firewall administrator. The generation of this report can easily scheduled with cron.

```
#!/bin/sh
######################################################################
# iptables_stats.sh
# packet filtering stats for iptables
######################################################################
##############################
# VARIABLES
##############################
HOSTNAME=`hostname`
SCRIPT_NAME="iptables_stats.sh"
MESSAGE="Option Not Selected"
REPORT_TITLE="IPTABLES STATS REPORT $HOSTNAME"
REPORT_HEADING=""
REPORT="/var/log/report/iptables_stats.txt"
MAILTO="administrator@network.com"
MAILFROM="root-security@network.com"
SUBJECT="IPTABLES Stats Report $DATE"
ATTACHMENT="$REPORT"
EMAIL_REPORT="1"
FILTER="1"
NAT="1"
MANGLE="1"
RAW="0"
POUND="######################################################################"

##############################
# EXECUTABLES
##############################
IPT="/usr/sbin/iptables"
LOGGER="/bin/logger"
ECHO="/bin/echo"
DATE="/bin/date"
MAIL="/bin/mail"

#########################
# BEGIN REPORT
#########################
$ECHO $POUND > $REPORT
$ECHO $REPORT_TITLE >> $REPORT
$DATE >> $REPORT
$ECHO $POUND >> $REPORT
$ECHO  >> $REPORT

#########################
# FILTER TABLE STATS
#########################
REPORT_HEADING="FILTER TABLE STATS"
$ECHO $POUND >> $REPORT
$ECHO $REPORT_HEADING >> $REPORT
$ECHO $POUND >> $REPORT
if [ "$FILTER" = "1" ];
then
        $ECHO "Getting FILT Table Stats"
        OPTIONS="-v -n -x -L -t filter"
        $IPT $OPTIONS >> $REPORT
```

```
        $ECHO $POUND >> $REPORT
fi

##########################
# NAT TABLE STATS
##########################
REPORT_HEADING="NAT TABLE STATS"
$ECHO $POUND >> $REPORT
$ECHO $REPORT_HEADING >> $REPORT
$ECHO $POUND >> $REPORT
if [ "$NAT" = "1" ];
then
        $ECHO "Getting NAT Table Stats"
        OPTIONS="-v -n -x -L -t nat"
        $IPT $OPTIONS >> $REPORT
        $ECHO $POUND >> $REPORT
fi

##########################
# MANGLE TABLE STATS
##########################
REPORT_HEADING="MANGLE TABLE STATS"
$ECHO $POUND >> $REPORT
$ECHO $REPORT_HEADING >> $REPORT
$ECHO $POUND >> $REPORT
if [ "$MANGLE" = "1" ];
then
        $ECHO "Getting MANGLE Table Stats"
        OPTIONS="-v -n -x -L -t mangle"
        $IPT $OPTIONS >> $REPORT
        $ECHO $POUND >> $REPORT
fi

##########################
# RAW TABLE STATS
##########################
REPORT_HEADING="RAW TABLE STATS"
$ECHO $POUND >> $REPORT
$ECHO $REPORT_HEADING >> $REPORT
$ECHO $POUND >> $REPORT
if [ "$RAW" = "1" ];
then
        $ECHO "Getting RAW Table Stats"
        OPTIONS="-v -n -x -L -t raw"
        $IPT $OPTIONS >> $REPORT
        $ECHO $POUND >> $REPORT
fi

################################
# MAIL REPORT
################################
if [ "$EMAIL_REPORT" = "1" ];
then
        $ECHO "Mailing $REPORT . . . . . "
        $CAT $REPORT | $MAIL -s "$SUBJECT" $MAILTO
fi
exit 0
```

PART 3: SYSTEM MAINTENANCE

Now that we the SUSE 9 Linux firewall has been installed, patched, tuned, and hardened, we shift our focus to system maintenance. Keeping a system “happy and

healthy" is perhaps an even more difficult task then making a system that is "happy and healthy". Proper system maintenance, at the very least, is unrelenting and intensive, and it can be quite time consuming if "automotive" tools and techniques are not used or employed.

System maintenance for the Linux firewall will include specific programs and procedures that deal with the following areas of concern:

- Backup and Disaster Recovery
- Patch Management
- Intrusion Detection and System Integrity
- Log Management
- Network Monitoring

BACKUP & DISASTER RECOVERY

mkCDrec

mkCDrec provides complete bare-metal system restore and recovery capabilities. For this reason, mkCDrec will be relied upon primarily for recovery from major system failure. Updated system ISO images should be burned to CD-ROM or DVD weekly, or whenever significant changes to the system are made (**RMM:C:5**). For this system:

- System backups will be scheduled with cron to occur every Sunday at midnight, and automated with the "**make CD-ROM"** command run from the mkcdrec source directory:

  0 0 * * 0 /usr/local/src/mkcdrec/make CD-ROM >> /dev/null

- To avoid unpleasant surprises, check each new backup for both "**bootability**" and "**restorability**". This is not a flawless process and a "usable" backup should not be assumed.

system_backup

Nightly backups of the most critical file systems and system files with the **system_backup.sh** script (**RMM:C:6**).

- The script, shown below, is flexible in that "what gets backed up" is configurable with the $DIR variable.

- This script will be scheduled to run automatically at 2200 hours with cron:

  0 22 * * * /root/bin/system_backup.sh >> /dev/null

- Each backups will be made in a compressed tar format to the /storage partition. The backup file will be purged after 30 days by an automated scripting process.

- Each backups will again be copied (using SSH-SCP) to a storage area on the internal network segment using an automated scripting process.

```
#!/bin/sh
#########################################################################
# system_backup.sh
# performs linux system backup
#########################################################################
#########################################################################
# INCLUDE FILES
#########################################################################
source /root/bin/utility.functions.sh

######################
# VERSION_INFO
######################
VERSION="1.0.1"
VDATE="03/24/04"

#########################################################################
# VARIABLES
#########################################################################
SERVER_NAME=`uname -n`
DATE=`date +%Y-%m-%d`
DAY=`date '+%d'`

HOSTNAME=`hostname`
SCRIPT_NAME="system_backup.sh"
MESSAGE="Starting System Backup on $HOSTNAME"
MESSAGE_DATE="DATE: "
REPORT_TITLE="SYSTEM BACKUP REPORT $HOSTNAME"
REPORT_HEADING=""
REPORT="/var/log/report/backup_report_$HOSTNAME.txt"

MAILTO="administrator@network.com"
MAILFROM="root-security@network.com"
SUBJECT="System Backup Report $HOSTNAME $DATE"
ATTACHMENT="$REPORT"
EMAIL_REPORT="1"

DESTBASE="/storage/backup/$HOSTNAME/$DATE"
DIRS="etc usr/local boot root, var/lib/tripwire"
NUM_DIRS="5"

TAR="/bin/tar"
LOGGER="/bin/logger"
DATE="/bin/date"
AWK="/bin/awk"
SED="/bin/sed"
ECHO="/bin/echo"
CAT="/bin/cat"
MAIL="/bin/mail"
SLEEP="/bin/sleep"
SENDMAIL="/usr/sbin/sendmail"

#########################################################################
# BEGIN SCRIPT
#########################################################################
$ECHO "Running $SCRIPT_NAME . . . . . "
$LOGGER -p warning "Running $SCRIPT_NAME . . . . . "

confirm_root
version_info "$VERSION" "$VDATE"

$ECHO $POUND > $REPORT
$ECHO $REPORT_TITLE >> $REPORT
```

```
$DATE >> $REPORT
$ECHO $POUND >> $REPORT
$ECHO  >> $REPORT

##########################
# ARCHIVE
##########################
DATE=`date +%Y-%m-%d`
$ECHO "Backup of $HOSTNAME on $DATE"
REPORT_HEADING="Backup of $HOSTNAME on $DATE"
print_heading_file $REPORT_HEADING $REPORT

# If the archive folder is not there, create it
[ -d $DESTBASE ] || mkdir -p $DESTBASE
cd /

for d in $DIRS;
do
      # Make sure directory even exists, or skip it
      if [ ! -d $d ];
      then
            $ECHO "Sorry, directory $d does not exist!"
            $ECHO "Sorry, directory $d does not exist!" >> $REPORT
            continue
      fi

      GIVEN=$d
      F=`$ECHO $GIVEN| $SED -e 's/\//-/'`
      SIZE=`du -sh $GIVEN | $AWK '{print $1}'`
      $ECHO "ARCHIVING . . . . . .  /$GIVEN"
      $ECHO ""
      $ECHO "Archiving /$GIVEN" >> $REPORT
      $ECHO "Size of $GIVEN is $SIZE"
      $ECHO "Size of $GIVEN is $SIZE" >> $REPORT
      $ECHO "$TAR --totals -czf $DESTBASE/$F.tgz $GIVEN"
      $ECHO "$TAR --totals -czf $DESTBASE/$F.tgz $GIVEN" >> $REPORT
      $TAR --totals -czf $DESTBASE/$F.tgz $GIVEN
done;

print_pound_file $REPORT

if [ "$EMAIL_REPORT" = "1" ];
then
      $ECHO "Mailing $REPORT . . . . . "
      $CAT $REPORT | mail -s "$SUBJECT" $MAILTO
fi
exit 0
```

SmartMonTools

SMART technology is used to monitor the health of a hard disk so the hard disk can be replaced during scheduled maintenance under controlled circumstances. Although the smart daemon runs and monitors hard disks continuously (**RMM:C:4**), a detailed report should be generated at least monthly and mailed to the firewall administrator. SMART data can be extracted from the hard disks with the "**smartctl**" utility. This guide recommends using the script "**smart_report.sh**" available in **Appendix Q**. The script can be scheduled to run under cron and will mail the firewall administrator a detailed report on all hard disks:

0 7 15 * * /root/bin/smart_report.sh >> /dev/null

PATCH MANAGEMENT

Software patches and system upgrades will be scheduled when the need has been identified. Refer to section on **Installing Patches and Bug Fixes** for procedures to be used.

INTRUSION DETECTION & INTEGRITY VERIFICATION

Tripwire

Tripwire will be the primary means for both system and file system integrity verification on the SUSE Linux firewall (**RMM:I:4**). However, a "tripwire" installation from the SUSE Linux 9.0 distribution only installs the tripwire binary and the basic file system structure wherein tripwire will live. Little else is provided. Thus, the administrator must write a "twinstall.sh" script and create configuration files from scratch, or obtain configuration file templates from elsewhere.

1. **Create twcfg.txt File**. The txcfg.txt file determines the overall functional capabilities and parameters for the tripwire binary. A recommended configuration is shown as follows:

```
##############################################################################
# twcfg.txt
# tripwire configuration file
##############################################################################
###############################################
# CONFIGURATION
###############################################
POLFILE                 = /etc/tripwire/tw.pol
DBFILE                  = /var/lib/tripwire/$(HOSTNAME).twd
REPORTFILE              = /var/lib/tripwire/report/$(HOSTNAME)-$(DATE).twr
SITEKEYFILE             = /etc/tripwire/site.key
LOCALKEYFILE            = /etc/tripwire/$(HOSTNAME)-local.key
EDITOR                  = /usr/bin/vi
LATEPROMPTING           = false
LOOSEDIRECTORYCHECKING  = false
MAILMETHOD              = SENDMAIL
MAILPROGRAM             = /usr/sbin/sendmail -oi -t
MAILNOVIOLATIONS        = true
EMAILREPORTLEVEL        = 4
REPORTLEVEL             = 4
SYSLOGREPORTING         = true
SYSLOGREPORTLEVEL       = 2
#####################################
```

   - Include file system variable assignments matching those created by the installation process.
   - "MAIL" parameters must be set for reports, warning, and notification to be emailed to the firewall administrator.
   - Set REPORTLEVEL to a value of 4 or 5 for maximum detail.

2. **Create twpol.txt File**. The twpol.txt file determines which files are monitored by tripwire. Tripwire, at a minimum, should monitor the following files and file systems on a Linux firewall:

   - Tripwire binaries
   - Tripwire data files
   - System binaries
   - Security control files such as /etc/shadow
   - Boot scripts
   - Boot and boot loader files
   - Login and environmental determinate scripts
   - Root account files and binaries
   - Critical device files
   - System configuration files

   A recommended example twpol.txt file is available in **Appendix O: txpol.txt File**.

3. **Create and Execute twinstall.sh File**. A recommended example twinstall.sh file is available in **Appendix P: twinstall.sh File**. This script performs the following actions:

   - Generates "site" encryption key
   - Generates "local" encryption key
   - Creates an encrypted configuration file from twcfg.txt
   - Creates an encrypted policy file from twpol.txt
   - Set tripwire directory permissions and ownership
   - Initializes tripwire and develops a baseline integrity database.
   - Removes the unencrypted version of the two configuration files.

4. **Schedule Regular Database Integrity Checks**. Using the cron daemon, schedule a daily tripwire system integrity check. This guide recommends the check be run in the early morning hours sometime between 0300 and 0600 hours. Although the command "tripwire –twcheck" can be run straight from cron, it is recommended the following "**twcheck.sh**[30]" script shown below be used. The script performs the following:

- Runs a system integrity check with the "**tripwire –twcheck**" command.

- Updates the tripwire database based on the results of the check with the "**tripwire –update –twfile**" command.

[30] Several lines of this code were taken from Linux Security Cookbook by Daniel J. Barrett, et al. O'Reilly Press, June 2003

```
#!/bin/sh
#########################################################################
# twcheck.sh
# runs tripwire integrity check from cron
#########################################################################
#########################################################################
# VARIABLES
#########################################################################
TW_DIR="/etc/tripwire"
TWDB_DIR="/var/lib/tripwire"
TWREPORT_DIR="/var/lib/tripwire/report"
SITE_KEY="$TW_DIR/site.key"
LOCAL_KEY="$TW_DIR/$HOSTNAME-local.key"
TW="/usr/sbin/tripwire"
TWADMIN="/usr/sbin/twadmin"
TWPRINT="/usr/sbin/twprint"
CHECK="1"
UPDATE="1"

################################
# RUN INTEGRITY CHECK
################################
if [ "$CHECK" = "1" ];
then
      if [ ! -e /var/lib/tripwire/$HOSTNAME.twd ] ;
      then
            $ECHO "**    Error: Tripwire database for $HOSTNAME not found.    **"
            $ECHO "** Run /etc/tripwire/twinstall.sh and/or tripwire --init. **"
      else
            $ECHO "Running Tripwire Integrity Check . . . . . "
            test -f $TW_DIR/tw.cfg && $TW --check --email-report
      fi
fi

################################
# UPDATE TW DB
################################
if [ "$UPDATE" = "1" ];
then
      LAST_REPORT=`ls -1t $TWREPORT_DIR/$HOSTNAME-*.twr | head -1`
      $ECHO "Updating Tripwire Database . . . . . "
      $TW --update --twrfile "$LAST_REPORT"
fi
```

Snort

Listening on the Internet facing network interface, the Snort IDS will be continuously monitoring for network intrusions and malicious traffic (**RMM:J:10**). Snort output will be configured to for logging as follows in the snort.conf file:

```
#################################################
# ALERTING & LOGGING
#################################################
output alert_syslog: LOG_LOCAL6 LOG_ALERT
output database: log, mysql, user=fw1 password=secret dbname=snort host=192.168.0.1 port=3306 sensor_name=server5
#########################################################################
```

Alerts will be logged through the local6 syslog facility to /var/log/snort/snort.log. The following line must be added to the syslog.conf file:

local6.=alert /var/log/snort/snort.log

The "alert logs" will be rotated daily and copied to a secure administrative workstation for daily analysis. Additionally, snort is configured to log to a remote MySQL database on the internal network segment. The information logged to the database will be analyzed with ACID and must be viewed and reviewed frequently by the firewall administrators.

Incident Response

All "undocumented" changes to the "Tripwire Database" or suspicious conditions detected through "log analysis" will be considered "Incidents", and each will be treated as an "**incident**[31]". The response to such incidents will consist of an initial damage assessment followed by an appropriate forensic investigation and report. Investigative tools used will be a special set of trusted static binaries burned to a CD-ROM.

LOG ROTATION & LOG ANALYSIS

In the section on Hardening Syslog and throughout this guide, we have meticulously configured logging to "compartmentalize" and "segregate" information so the information can be easily manage, extracted, and analyzed. However, log files must be properly and methodically managed to preserve disk space, prevent information loss, and avoid information overload. Additionally, a log file that is never reviewed or processed is also a log file that probably should never have been written in the first place. With these concepts in mind, this section discusses log rotation and log analysis for the SUSE 9 Linux firewall.

Logrotate

Log rotation is made simple by the Linux "**logrotate**" utility program. Additionally, logrotate is flexible and extensible, and it permits a very granular tuning structure through its configuration file **/etc/logrotate.conf**. The following options are available with logrotate.conf for log file management:

- **Configuration Extension:** file or directory inclusion:

  include /etc/logrotate.d

- **Compression:** compress, nocompress, or delaycompress.
- **New File Creation:** create mode owner group, or nocreate.
- **Rotation Interval:** daily, weekly, or monthly.
- **Rotation Count:** rotates the log "***count"*** times before being removed.
- **Rotation Size:** log files greater than size are removed.
- **Email Notification:** send errors to the address indicated.

[31] An "incident" refers to an adverse event in an information system or network, or the threat of the occurrence of such an event. "Incident" implies that either harm has been done to the system, or there has been an attempt to harm the system.

- **Command Execution:** delimits commands before and after the log is rotated.
- **Archival Directory:** old logs on the same physical device moved to this directory.
- **Empty or Missing Archival:** rotate the logs even if empty and ignore if missing.

The complete logrotate.conf file for our SUSE Linux firewall is located in **Appendix A: /etc/logrotate.conf File**. However, the default configuration, and the configuration for the firewall logs is show below:

```
#####################################
# INCLUDE
#####################################
Include /etc/logrotate.d

#####################################
# DEFAULTS
#####################################
compress
daily
rotate 10
maxage 120
size +9999k
dateext
create
missingok
notifempty

#####################################
# SECURITY & INTRUSION LOGS
#####################################
# iptables
/var/log/iptables/iptables.log {
    missingok
    notifempty
    rotate 30
    dateext
    daily
    nocompress
    create 700 root root
    postrotate
        /etc/init.d/syslog reload
    endscript
}
```

This file segment extends the logrotate.conf configuration to the logrotate.d[32] directory, sets the defaults for the system, and configures rotation for the "**iptables.log**" logs. By design, this configuration rotates iptables.log files daily, applies no compression, dates each log file, and retains each log for 30 days. It will also "reload" the syslog daemon and create the new file after rotation has occurred.

After rotation, the following logs are securely copied to an administrative workstation on the internal network for analysis:

- iptables.log
- iptraf.log
- snort.log
- alert.log
- tcpdump.dmp
- messages
- security
- sudo.log

[32] The logrotate.d directory normally contains individual configuration files for RPM installed packages.

- cron
- mail

Log copying is automated by **log_copy.pl**, a proprietary perl script. This script can be found in its entirety in **Appendix M: Script log_copy.pl**.

LOG ANALYSIS

A Linux firewall produces a large amount of log data and it would be unrealistic to believe that any normal human being could reasonably review, extract, and correlate useful data on a long term basis. Log processing for this Linux firewall will be highly automated, both off the system, and on the system. Logs transferred from the firewall to the administrative workstation with the **log_copy.pl** script (see **Log Rotation** above), will undergo a thorough parsing and correlation analysis, with a daily report generated for review by the firewall administrator.

The LogCheck Sentry tool will be assigned the task of ongoing on-system log analysis. As noted in the **LogCheck Installation** section, the effectiveness of LogCheck is based on the contents and configuration of four key files:

1. The **logcheck.sh**, the main configuration script, controls all processing and parses the log files with grep commands. **logcheck.sh** is executed on a timed basis from cron and sends a report to the firewall administrator.

2. The **logcheck.hacking** file contains keywords that indicate a possible attack on the system has transpired.

3. The **logcheck.violations** file contains keywords of system events that are usually seen as negative such as "denied" and "refused". Positive words such as 'su' are also placed in this list. Violations from this file are reported as "Security Violations" in the report.

4. The **logcheck.violations.ignore** file contains words that are reverse searched against the logcheck.violations file, essentially nullifying them.

The complete **logcheck.sh** file is available in **Appendix H: LogCheck Configuration File**. However, if a log file is to be reviewed by LogCheck, it must be properly set within the logcheck.sh file. Note the settings of the logs we wished to have parsed on the firewall:

```
$LOGTAIL /var/log/messages > $TMPDIR/check.$$
$LOGTAIL /var/log/mail/mail >> $TMPDIR/check.$$
$LOGTAIL /var/log/iptables/iptables.log >> $TMPDIR/check.$$
$LOGTAIL /var/log/snort/snort.log >> $TMPDIR/check.$$
$LOGTAIL /var/log/security/security.log >> $TMPDIR/check.$$
```

Additionally, be certain to set the SYSADMIN parameter to the username to whom reports generated by LogCheck are sent: **SYSADMIN=root**. If logcheck.sh is scheduled

to run hourly from cron, then a report on the log file findings will be sent to the root account each and every hour.

NETWORK MONITORING

NTOP

NTOP is a secure real-time network traffic monitoring system with SSL enabled reporting capabilities. Thus NTOP will be relied upon heavily to determine not only how the SUSE Linux firewall is performing and what the specific network conditions are during any slice-of-time. Utilizing an integrated packet sniffer, packet analyzer, and reporting engine, NTOP records and reports on the following:

- All IP unicast and multicast traffic
- TCP session history
- UDP traffic sorted by port
- Operating system identification
- Bandwidth usage statistics
- Traffic distribution by subnet and host name
- Network misconfiguration
- Service misuse or service abuse
- Portscans
- Trojan horse communication
- Routing bottlenecks
- Possible DOS attacks such as SYN flooding.

For a secure NTOP configuration, complete the following steps:

1. **Activate HTTP/HTTPS Server and Services**. Edit the **/etc/sysconfig/ntop** and set the following parameters to activate the web interface:

   NTOP_PORT="<firewall-internetfacing-ipaddress>:3000"
   NTOP_SSL_PORT="<firewall-internetfacing-ipaddress>:3001"
   NTOP_USER="wwwrun"

2. **Set NTOP Options**. Edit the /etc/init.d/ntop startup script and set the desired run options for NTOP:

   NTOP_OPTIONS="-E -P /var/lib/ntop"

   The -E options activates the interface for "external tools". Tools such as lsof can then be used to analyze network connection activity. The –P options activates a data storage database.

3. **Set Administrative Password**. Open a web browser, set URL to **<firewall-internetfacing-ipaddress>:3000**, and set the NTOP administrative password. The password is then encrypted and stored in the /var/sbin/passcheck file.

4. Add restrictive rules to firewall script permitting access to tcp ports 3000 and 3001 only from trusted workstations (**RMM:J:2**) (**RMM:J:17**).

5. Restart NTOP: **./etc/init.d/ntop reload**.

IPTraf

IPTraf intercepts packets using the built-in raw socket interface of the Linux kernel. IPTraf will be brought into use to intercept and record selected traffic when unusual network activity or network attack is detected. Packet and traffic data will be logged to the /var/log/iptraf/iptraf.log file and copied to a trusted administrative workstation for analysis (**RMM:J:16**).

TCPDump

All packets entering and exiting the network through the firewall will be "raw dumped" to file (/var/log/tcpdump/tcpdump.dmp). The dump files will be rotated daily and archived to a storage location on the internal network segment. The purpose for this action is to create a forensic record of all network activity should be data ever needed (**RMM:J:16**). TCPDump will be start at boot time with **OPTIONS="-i eth1 -n -vv -xX -s 1500"** by the script in **Appendix N: tcpdump Startup Script**.

FORENSIC ANALYSIS

A forensic analysis of the SUSE 9 Linux firewall will be done bi-monthly (or as needed) by running the shell script **forensic_report.sh**. This script performs the following tests, checks, and actions:

- Runs Rootkit Hunter checks for promiscuous NICS
- Checks all running processes with "**ps**"
- Checks utmp, wtmp, btmp files for login history
- Identifies all SUID & GUID files
- Identifies services started by INETD or XINETD
- Identifies all "world writable" directories
- Runs chkconfig to identify boot files activated for startup
- Runs netstat and identifies active port bindings
- Gathers bash command history for root
- Identifies all /dev/files (regular and special device)
- Runs integrity check on all RPM Files
- Runs "**lsof**" on systems current state
- Runs John the Ripper and identifies users with accounts with no logins
- Checks system warning banners
- Checks and tests permissions and ownerships
- Checks ftpusers file for root and others
- Checks all RHOSTS files for abnormalities
- Checks for user activity on system

The script **forensic_report.sh** will be scheduled to run with cron.

```
0 6 1,15 * * /root/bin/forensic_report.sh >> /dev/null
```

When completed, it will mail a neatly formatted report to the firewall administrator. This scrip[t can be found in **Appendix R: Script forensic_report.sh**.

PART 4: TESTING & VERIFICATION

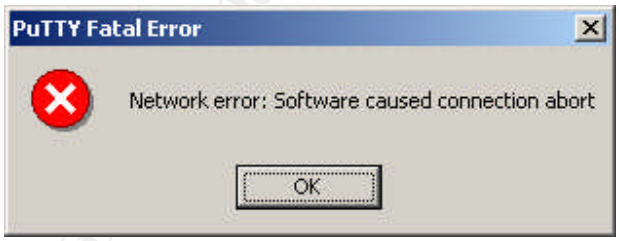
After a considerable amount of concern, time, and effort, the SUSE Linux 9 firewall is finally up and running. Congratulations might be in order, but the job is not quite done just yet. The configuration and hardening steps we took must be tested and the effectiveness of these measures verified. At this point, we only "think" we know the strength of the security we have in place. However, "thinking" and "knowing" are quite often two different things. The "knowing" is the level of confidence we must have for the system that is protecting the entire network.

Tests will be performed against the following areas of the system configuration to validate security:

1. Network Access
2. System Access
3. System Services and Activity
4. Logging
5. Vulnerability Assessment

Network Access

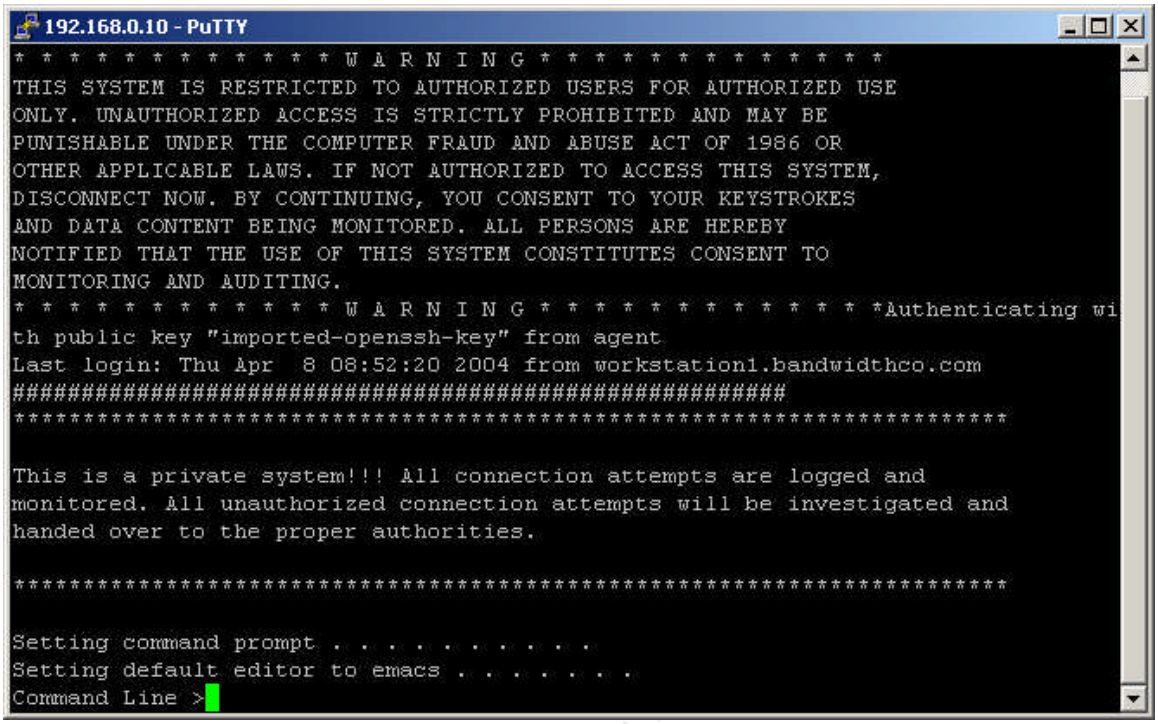
1. **Test: Remote Root Login Refused**
   Remote root login attempted from remote host using Putty. Login was refused and Putty received fatal error.



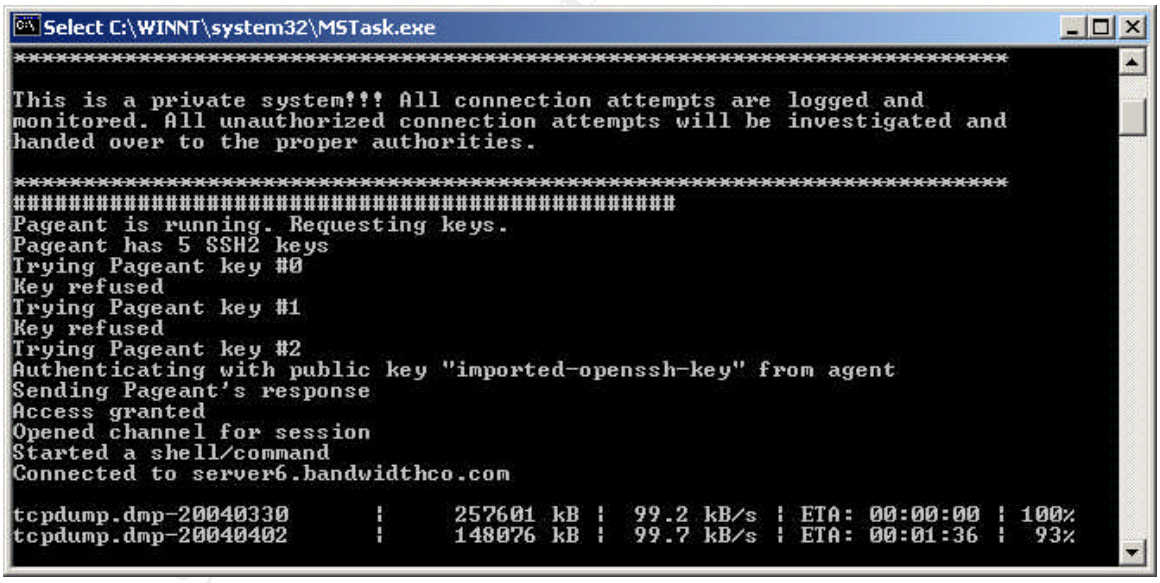
   Refusal confirmed by checking with the Linux firewall system logs:

   Apr 8 07:21:43 server6 sshd[24361]: ROOT LOGIN REFUSED FROM ::ffff:192.168.0.3
   Apr 8 07:21:43 server6 sshd[24361]: fatal: monitor_read: unsupported request: 32
   Apr 8 07:22:15 server6 sshd[24363]: ROOT LOGIN REFUSED FROM ::ffff:192.168.0.3

2. **Test: Warning Banners Functioning Properly**.
   The **/etc/issue banner** as set appeared correctly with remote ssh login:



The **/etc/motd banner** as set appeared correctly with remote scp file transfer:



System Access

1. **Test: SUDO Executed Commands Logged**.
   The sudo.log and sudolog confirms commands executed with SUDO are logged:

From sudo.log

```
Apr  8 07:16:08 server6 sudo:    root : TTY=pts/1 ; PWD=/root ; USER=root ; COMMAND=/bin/ls
Apr  8 07:27:21 server6 sudo:    root : TTY=pts/3 ; PWD=/root ; USER=root ; COMMAND=/usr/bin/crontab
```

```
Apr  8 07:27:32 server6 sudo:    root : TTY=pts/3 ; PWD=/root ; USER=root ; COMMAND=/usr/bin/crontab -e
Apr  8 07:33:12 server6 sudo:    root : TTY=pts/3 ; PWD=/root ; USER=root ; COMMAND=/bin/ls -ls /etc/init.d
Apr  8 07:33:27 server6 sudo:    root : TTY=pts/3 ; PWD=/root ; USER=root ; COMMAND=/bin/ls /etc/init.d
```

From sudolog

```
Apr  8 07:16:08 : root : TTY=pts/1 ; PWD=/root ; USER=root ; COMMAND=/bin/ls
Apr  8 07:27:21 : root : TTY=pts/3 ; PWD=/root ; USER=root ; COMMAND=/usr/bin/crontab
Apr  8 07:27:32 : root : TTY=pts/3 ; PWD=/root ; USER=root ;
 COMMAND=/usr/bin/crontab -e
Apr  8 07:33:12 : root : TTY=pts/3 ; PWD=/root ; USER=root ; COMMAND=/bin/ls -ls /etc/init.d
Apr  8 07:33:27 : root : TTY=pts/3 ; PWD=/root ; USER=root ; COMMAND=/bin/ls /etc/init.d
```

2. **Test: File System Permissions Properly Set**
   The “chkstat” utility is tested for proper function with the following code:

```
#!/bin/sh
#####################################
# permissions_test.sh
#####################################
#####################################
# VARIABLES
#####################################
CHKSTAT="/usr/bin/chkstat"
TEST_FILE="/etc/permissions-test"
LOCAL="/etc/permissions.local"
BEFORE=""
CHANGE=""
AFTER=""
POUND="############################"

#####################################
# TEST
#####################################
touch $TEST_FILE
chown root:root $TEST_FILE
chmod 700 $TEST_FILE
BEFORE=`ls -la $TEST_FILE`
chown bin:bin $TEST_FILE
chmod 777 $TEST_FILE
CHANGE=`ls -la $TEST_FILE`
$CHKSTAT --set $LOCAL
AFTER=`ls -la $TEST_FILE`

echo "$POUND"
echo "BEFORE TEST: $TEST_FILE"
echo "$POUND"
echo "$BEFORE"
echo "$POUND"
echo "CHANGE TEST: $TEST_FILE"
echo "$POUND"
echo "$CHANGE"
echo "$POUND"
echo "AFTER TEST: $TEST_FILE"
echo "$POUND"
echo "$AFTER"
exit 0
```

Output reveals the utility is functioning correctly:

```
######################################################################
PERMISSIONS TEST
######################################################################
```

```
########################################################################
BEFORE TEST: /etc/permissions-test
########################################################################
-rwx------   1 root     root           0 Apr 10 19:10 /etc/permissions-test
########################################################################
CHANGE TEST: /etc/permissions-test
########################################################################
-rwxrwxrwx   1 bin      bin            0 Apr 10 19:10 /etc/permissions-test
########################################################################
AFTER TEST: /etc/permissions-test
########################################################################
-rwx------   1 root     root           0 Apr 10 19:10 /etc/permissions-test
########################################################################
```

We now set the configured permissions and ownerships by running the following command:

chkstat –set /etc/permissions.secure

3. **Test: Root listed in ftpusers File**.
   We run the following command:

   cat /etc/ftpusers | grep root

   User "root" is returned.

System Services and Activity

1. **Test: Netstat For Listening Port**
   The output from **netstat –p** shows what was expected, and there were no surprises. We see the SSHD, X-windows, APC-agent, and NTOP services listening on their expected tcp ports. NTP port udp/123 availability is also expected.

   Output from netstat –p

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:6000            0.0.0.0:*               LISTEN
tcp        0      0 192.168.0.10:3000       0.0.0.0:*               LISTEN
apctcp     0      0 :::3052                 :::*                    LISTEN
tcp        0      0 :::22                   :::*                    LISTEN
udp        0      0 192.168.3.10:123        0.0.0.0:*
udp        0      0 192.168.2.10:123        0.0.0.0:*
udp        0      0 192.168.1.10:123        0.0.0.0:*
udp        0      0 66.14.166.45:123        0.0.0.0:*
udp        0      0 192.168.0.10:123        0.0.0.0:*
udp        0      0 127.0.0.1:123           0.0.0.0:*
udp        0      0 0.0.0.0:123             0.0.0.0:*
udp     4488      0 :::514                  :::*
raw        0      0 0.0.0.0:6               0.0.0.0:*               7
raw        0      0 0.0.0.0:17              0.0.0.0:*               7
raw        0      0 0.0.0.0:255             0.0.0.0:*               7
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ACC ]     STREAM     LISTENING     20877  /tmp/mcop-root/server6_bandwidthco_com-0997-406cee2a
unix  2      [ ACC ]     STREAM     LISTENING     9240   /tmp/.X11-unix/X0
unix  2      [ ACC ]     STREAM     LISTENING     20739  /tmp/ksocket-root/kdeinit-:0
unix  2      [ ACC ]     STREAM     LISTENING     20743  /tmp/.ICE-unix/dcop2436-1080880668
```

```
unix  2      [ ACC ]     STREAM     LISTENING     1956   /var/run/.resmgr_socket
unix  2      [ ACC ]     STREAM     LISTENING     20766  /tmp/ksocket-root/klauncheraiHaRa.slave-socket
unix  2      [ ACC ]     STREAM     LISTENING     20919  /tmp/.ICE-unix/2462
```

2. **Test: Programs With Bound Sockets**
   Test for executable programs with port bindings with **/usr/bin/lsof –l –n –P** command:

```
echo "The following programs have got bound sockets:"
/usr/bin/lsof -i -n -P | egrep 'UDP|TCP.*LISTEN' | sed 's/....[0-9]u  IP.*     /   /' | sed 's/  FD \
TYPE DEVICE SIZE NODE NAME/PROTO PORT/' | sed 's/ [0-9][0-9]* / /'|sed 's/ PID / /'|sort -u
```

   Again, the output confirms what was expected.

```
The following programs have got bound sockets:
X       root        TCP *:6000 (LISTEN)
sshd    root        TCP 192.168.3.1:22 (LISTEN)
ntop    wwwrun      TCP *:3000 (LISTEN)
```

3. **Test: INETD or XINETD Services**
   Test for services under the control of either inetd or xinetd, and test to see if the daemons are running on the system. The following script is used:

```
#!/bin/sh
########################################
# inetd_test.sh
########################################
iconf="/etc/inetd.conf"
xconf="/etc/xinetd.conf"
xdir="/etc/xinetd.d"
if [ -r $iconf ];
then
      echo "Services enabled in $iconf are:"
      grep -v '^#' $iconf | awk '{print "  " $1}'
      echo ""
      if [ "$($PS $OPTIONS | grep inetd | egrep -vE '(xinet|grep)')" = "" ];
      then
            echo "** warning: inetd does not appear to be running"
      fi
fi

if [ -r $xconf ];
then
      echo "Services enabled in $xdir are:"
      for service in $xdir/*
      do
            if ! $(grep disable $service | grep 'yes' > /dev/null) ;
            then
                  echo -n "  "
                  basename $service
            fi
      done
      if ! $($PS $OPTIONS | grep xinetd | grep -v 'grep' > /dev/null) ;
      then
            echo "** warning: xinetd does not appear to be running"
      fi
fi
```

Since neither inetd nor xinetd, the test confirms what was expected:

*** warning: inetd does not appear to be running**
*** warning: xinetd does not appear to be running**

4. **Test: Current User Activity On System**
   To check for current user activity on the system, the following script is used:

```
[ $Header = yes ] && {
      date
      uname -n
   }

who |
while read Name Tty Mon Day Time Host Rest
do
   [ -n "$User" -a "$User" != "$Name" ] && continue
   echo " $Tty $Name    $Time"
         case "$Tty" in
            *tty*)       T=`echo "$Tty" | sed -e 's:.*tty\(..\).*:\1:'`;;
            *)     T=`echo "$Tty" | sed -e 's:/dev/\(..\).*:\1:'`;;
         esac

            ps -ct"$T" | tail +2 |
            while read pid tty stat time command
            do
               echo "    $Tty     $pid $time      $command"
            done
done
```

Normal output returned:

```
18:56:47 up 15 min,  3 users,  load average: 1.00, 0.44, 0.22
USER     TTY      LOGIN@   IDLE   JCPU   PCPU WHAT
root     :0       18:42   ?xdm?  21.20s  0.02s -:0
root     pts/0    18:43   13:32   0.00s  0.42s kdeinit: kwrited
root     pts/1    18:51    1:33   0.21s  0.12s sh forensic_report.sh
```

5. **Test: Services Enabled at Boot**.
   We run the following command:

   chkconfig –list

   The following output is returned:

```
acpid                0:off   1:off   2:on    3:on    4:off   5:on    6:off
boot.clock           0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.crypto          0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.idedma          0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.ipconfig        0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.isapnp          0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.klog            0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.ldconfig        0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.loadmodules     0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.localfs         0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

```
boot.localnet           0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.lvm                0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.md                 0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.proc               0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.restore_permissions 0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.sched              0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.scpm               0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.scsidev            0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.swap               0:off   1:off   2:off   3:off   4:off   5:off   6:off
boot.sysctl             0:off   1:off   2:off   3:off   4:off   5:off   6:off
cron                  0:off   1:off   2:on    3:on    4:off   5:on    6:off
fbset                 0:off   1:on    2:on    3:on    4:off   5:on    6:off
hotplug                0:off   1:on    2:on    3:on    4:off   5:on    6:off
hwscan                  0:off   1:off   2:on    3:on    4:off   5:on    6:off
inetd                 0:off   1:off   2:off   3:on    4:off   5:on    6:off
kbd                   0:off   1:on    2:on    3:on    4:off   5:on    6:off
network                 0:off   1:off   2:on    3:on    4:off   5:on    6:off
ntop                  0:off   1:off   2:off   3:off   4:off   5:off   6:off
random                  0:off   1:off   2:on    3:on    4:off   5:on    6:off
raw                   0:off   1:off   2:off   3:off   4:off   5:off   6:off
resmgr                  0:off   1:off   2:on    3:on    4:off   5:on    6:off
sendmail                0:off   1:off   2:off   3:on    4:off   5:on    6:off
sshd                  0:off   1:off   2:off   3:on    4:off   5:on    6:off
syslog                0:off   1:off   2:on    3:on    4:off   5:on    6:off
xdm                   0:off   1:off   2:off   3:off   4:off   5:on    6:off
xntpd                 0:off   1:off   2:off   3:on    4:off   5:on    6:off
xinetd based services:
```

This verifies our hardening configuration.

Logging

1. **Test: Syslog Facilities & Levels Function**

   Execute **syslog_test.sh** script and check output in /var/log/messages. The test confirms the logging facility is performing correctly.

```
#!/bin/sh
#########################################################################
# syslog_test.sh
# test to make sure are facilities and levels of syslog are working correctly
#########################################################################

#########################################################################
# VARIABLES
#########################################################################
FACILITIES="auth authpriv cron daemon kern mail syslog local0 local1 local2 local3 local4 local5 local6 local7"
LEVELS="alert crit debug emerg err info notice warning"
MESSAGE=""
TAG="----TEST----"
LOGGER="/bin/logger"

echo "Starting SYSLOG test . . . . . "
echo $LOGGER -p warning "Starting SYSLOG test . . . . . "

for facility in $FACILITIES
do
      for level in $LEVELS
      do
            MESSAGE="**** TESTING SYSLOG $facility.$level ****"
            $LOGGER -t $TAG -p $facility.$level "$MESSAGE"
            echo "Testing $facility.$level . . . . . "
```

```
      done
done

echo "Ending SYSLOG test . . . . . "
echo $LOGGER -p warning "Ending SYSLOG test . . . . . "
exit 0
```

Output from test in /var/log/messages:

```
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG cron.alert ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG cron.crit ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG cron.debug ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG cron.emerg ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG cron.err ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG cron.info ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG cron.notice ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG cron.warning ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG daemon.alert ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG daemon.crit ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG daemon.debug ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG daemon.emerg ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG daemon.err ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG daemon.info ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG daemon.notice ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG daemon.warning ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG kern.alert ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG kern.crit ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG kern.debug ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG kern.emerg ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG kern.err ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG kern.info ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG kern.notice ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG kern.warning ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG syslog.alert ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG syslog.crit ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG syslog.debug ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG syslog.emerg ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG syslog.err ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG syslog.info ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG syslog.notice ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG syslog.warning ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG local0.alert ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG local0.crit ****
Apr 10 21:38:09 server6 ----TEST----: **** TESTING SYSLOG local0.debug ****
LINES REDACTED TO SAVE SPACE
Apr 10 21:38:10 server6 ----TEST----: **** TESTING SYSLOG local7.alert ****
Apr 10 21:38:10 server6 ----TEST----: **** TESTING SYSLOG local7.crit ****
Apr 10 21:38:10 server6 ----TEST----: **** TESTING SYSLOG local7.debug ****
Apr 10 21:38:10 server6 ----TEST----: **** TESTING SYSLOG local7.emerg ****
Apr 10 21:38:10 server6 ----TEST----: **** TESTING SYSLOG local7.err ****
Apr 10 21:38:10 server6 ----TEST----: **** TESTING SYSLOG local7.info ****
Apr 10 21:38:10 server6 ----TEST----: **** TESTING SYSLOG local7.notice ****
Apr 10 21:38:10 server6 ----TEST----: **** TESTING SYSLOG local7.warning ****
```

2. **Test: LogCheck Reports Generated and Mailed**
   Reports are being generated and mailed properly to firewall administrator. Below is a report segment:

```
Security Violations
=-=-=-=-=-=-=-=-=-=
Apr 10 17:36:08 server6 smartd[732]: Device: /dev/hda, SMART Prefailure Attribute: 8
Seek_Time_Performance changed from 248 to 247
```

```
Unusual System Events
=-=-=-=-=-=-=-=-=-=-=
Apr 10 16:20:36 server6 -- MARK --
Apr 10 16:34:20 server6 snort: TCP Data Offset(44) < longer than payload(20)!
Apr 10 16:59:00 server6 /USR/SBIN/CRON[21782]: (root) CMD ( rm -f
/var/spool/cron/lastrun/cron.hourly)
Apr 10 17:20:36 server6 -- MARK --
Apr 10 17:36:08 server6 smartd[732]: Device: /dev/hda, SMART Prefailure Attribute: 8
Seek_Time_Performance changed from 248 to 247 Apr 10 17:59:00 server6 /USR/SBIN/CRON[29010]:
(root) CMD ( rm -f /var/spool/cron/lastrun/cron.hourly)
Apr 10 18:20:36 server6 -- MARK --
Apr 10 18:40:36 server6 -- MARK --
Apr 10 18:59:00 server6 /USR/SBIN/CRON[3553]: (root) CMD ( rm -f
/var/spool/cron/lastrun/cron.hourly)
Apr 10 19:09:14 server6 logger: Running forensic_report.sh . . . . . .
```

3. **Test: Packets Logged to IPTables Log**

   Packets are being properly logged, and properly logged remotely. Below is a segment extracted from the remote syslog server on the administrative network:

2004-04-10,20:14:07,2004-04-10,20:14:07,192.168.0.3,0,7,kernel: DEFAULT NEW: UDP INT -> : IN=eth0 OUT= MAC=ff:ff:ff:ff:ff:ff:00:07:e9:81:ce:fc:08:00 SRC=192.168.0.3 DST=192.168.0.255 LEN=305 TOS=0x00 PREC=0x00 TTL=128 ID=26566 PROTO=UDP SPT=138 DPT=138 LEN=285

2004-04-10,20:14:09,2004-04-10,20:14:09,192.168.0.3,0,7,kernel: DEFAULT NEW: UDP INT -> : IN=eth0 OUT= MAC=ff:ff:ff:ff:ff:ff:00:07:e9:81:ce:fc:08:00 SRC=192.168.0.3 DST=192.168.0.255 LEN=305 TOS=0x00 PREC=0x00 TTL=128 ID=26591 PROTO=UDP SPT=138 DPT=138 LEN=285

2004-04-10,20:14:09,2004-04-10,20:14:09,192.168.0.3,0,7,kernel: BLACKHOLE BLOCKED TCP OUT: IN=eth0 OUT=eth1 SRC=192.168.0.6 DST=216.73.87.172 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=63587 DF PROTO=TCP SPT=4291 DPT=80 WINDOW=64240 RES=0x00 SYN URGP=0

2004-04-10,20:14:10,2004-04-10,20:14:10,192.168.0.3,0,7,kernel: DEFAULT NEW TCP INT -> : IN=eth0 OUT=eth2 SRC=192.168.0.1 DST=192.168.1.2 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=62382 DF PROTO=TCP SPT=2624 DPT=135 WINDOW=65535 RES=0x00 SYN URGP=0

2004-04-10,20:14:11,2004-04-10,20:14:11,192.168.0.3,0,7,kernel: DEFAULT NEW: UDP INT -> : IN=eth0 OUT= MAC=ff:ff:ff:ff:ff:ff:00:07:e9:81:ce:fc:08:00 SRC=192.168.0.3 DST=192.168.0.255 LEN=305 TOS=0x00 PREC=0x00 TTL=128 ID=26592 PROTO=UDP SPT=138 DPT=138 LEN=285

2004-04-10,20:14:12,2004-04-10,20:14:12,192.168.0.3,0,7,kernel: WEB ACCESS: INTERNAL ->: IN=eth0 OUT=eth1 SRC=192.168.0.6 DST=64.5.249.81 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=63699 DF PROTO=TCP SPT=4292 DPT=80 WINDOW=64240 RES=0x00 SYN URGP=0

2004-04-10,20:14:12,2004-04-10,20:14:12,192.168.0.3,0,7,kernel: INTERNAL SNAT OUT: IN= OUT=eth1 SRC=192.168.0.6 DST=64.5.249.81 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=63699 DF PROTO=TCP SPT=4292 DPT=80 WINDOW=64240 RES=0x00 SYN URGP=0

2004-04-10,20:14:12,2004-04-10,20:14:12,192.168.0.3,0,7,kernel: BLACKHOLE BLOCKED TCP OUT: IN=eth0 OUT=eth1 SRC=192.168.0.6 DST=216.73.87.172 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=63708 DF PROTO=TCP SPT=4293 DPT=80 WINDOW=64240 RES=0x00 SYN URGP=0

2004-04-10,20:14:12,2004-04-10,20:14:12,192.168.0.3,0,7,kernel: BLACKHOLE BLOCKED TCP OUT: IN=eth0 OUT=eth1 SRC=192.168.0.6 DST=216.73.87.172 LEN=48 TOS=0x00 PREC=0x00 TTL=127 ID=63710 DF PROTO=TCP SPT=4294 DPT=80 WINDOW=64240 RES=0x00 SYN URGP=0

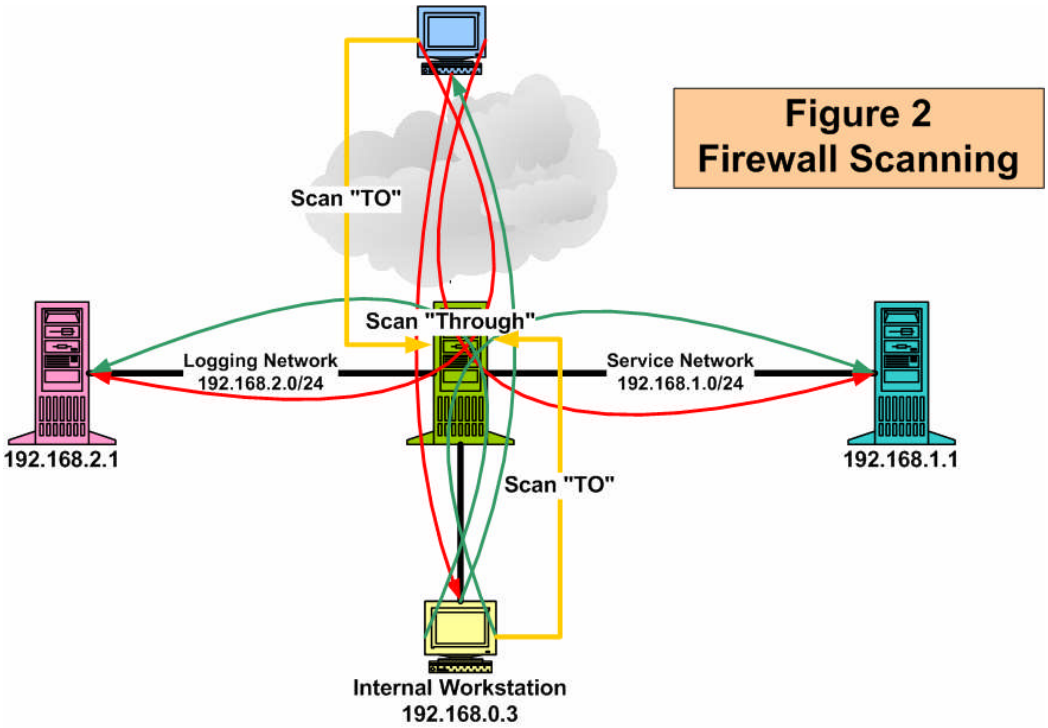
Vulnerability Assessment

1. **NMAP Scanning**

   Nmap is an excellent tool for not only checking the effectiveness of firewall rule set, but for checking the firewall box for vulnerabilities itself. The normal procedure for checking the firewall rule base normally involves multiple complex scans that attempt to “go through” the firewall, and not “get to” the firewall. Typically scans

should be done from all directions and through all combinations of the firewall network interface cards. **Figure 2 "Firewall Scanning"** below demonstrates the basic process.

However, since this guide is not about the effectiveness of the rule set, but the



security of the system itself, we will only run two scans through the firewall itself as a test of basic security. Additionally, we will target two scans at the firewall and its ports directly. The nmap commands for the four scans are as follows:

Test1: nmap -v -g53 -sS -sR -P0 -O -T 3 -p1-65000 -oN Q:\tests\test1.txt 192.168.0.10
Test2: nmap -v -g53 -sS -sR -PI -O -T 3 -p1-65000 -oN Q:\tests\test2.txt 192.168.0.10
Test3: nmap -v -g53 -sS -sR -P0 -O -T 3 -p1-65000 -oN Q:\tests\test3.txt 192.168.1.1
Test4: nmap -v -g53 -sS -sR -PI -O -T 3 -p1-65000 -oN Q:\tests\test4.txt 192.168.1.1

All 65,000 ports will be scrutinized by nmap. Test 1 and Test 2 will be directed at the firewall, while Test 3 and Test 4 will be directed at the sendmail relay on the service network. The results are as follows:

Test1

```
# nmap 3.50 scan initiated Thu Apr 08 12:53:19 2004 as: nmap -v -g53 -sS -sR -P0 -O -T 3 -p1-65000 -oN
Q:\tests\test1.txt 192.168.0.10
Interesting ports on server6.bandwidthco.com (192.168.0.10):
(The 64990 ports scanned but not shown below are in state: closed)
PORT      STATE    SERVICE          VERSION
22/tcp    open     ssh
3000/tcp  open     ppp
```

3052/tcp open PowerChute
6000/tcp open X11
Device type: general purpose
Running: Linux 2.4.X
OS details: Linux Kernel 2.4.19 - 2.4.25
Uptime 6.638 days (since Thu Apr 01 20:35:15 2004)
TCP Sequence Prediction: Class=random positive increments
Difficulty=1661678 (Good luck!)
IPID Sequence Generation: Incremental
Nmap run completed at Thu Apr 08 12:53:31 2004 -- 1 IP address (1 host up) scanned in 11.875 seconds

Test 2

nmap 3.50 scan initiated Thu Apr 08 12:53:31 2004 as: nmap -v -g53 -sS -sR -PI -O -T 3 -p1-65000 -oN Q:\tests\test3.txt 192.168.0.10
Interesting ports on server6.bandwidthco.com (192.168.0.10):
(The 64990 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE VERSION
22/tcp open ssh
3000/tcp open ppp
3052/tcp open PowerChute
6000/tcp open X11
Device type: general purpose
Running: Linux 2.4.X
OS details: Linux Kernel 2.4.19 - 2.4.25
Uptime 6.638 days (since Thu Apr 01 20:35:15 2004)
TCP Sequence Prediction: Class=random positive increments
Difficulty=3323416 (Good luck!)
IPID Sequence Generation: Incremental
Nmap run completed at Thu Apr 08 12:53:43 2004 -- 1 IP address (1 host up) scanned in 11.907 seconds

Test 3

nmap 3.50 scan initiated Thu Apr 08 12:53:43 2004 as: nmap -v -g53 -sS -sR -P0 -O -T 3 -p1-65000 -oN Q:\tests\test4.txt 192.168.1.1
Interesting ports on server5.bandwidthco.com (192.168.1.1):
(The 64989 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE VERSION
21/tcp open ftp
22/tcp open ssh
25/tcp open smtp
53/tcp open domain
80/tcp open http
111/tcp open rpcbind (rpcbind V2) 2 (rpc #100000)
6000/tcp open X11
10000/tcp filtered snet-sensor-mgmt
Device type: general purpose
Running: Linux 2.4.X
OS details: Linux Kernel 2.4.19 - 2.4.20
Uptime 39.065 days (since Sun Feb 29 10:20:18 2004)
TCP Sequence Prediction: Class=random positive increments
Difficulty=2974133 (Good luck!)
IPID Sequence Generation: Incremental
Nmap run completed at Thu Apr 08 12:54:00 2004 -- 1 IP address (1 host up) scanned in 17.172 seconds

Test 4

```
# nmap 3.50 scan initiated Thu Apr 08 12:54:00 2004 as: nmap -v -g53 -sS -sR -PI -O -T 3 -p1-65000 -oN
Q:\tests\test6.txt 192.168.1.1
Interesting ports on server5.bandwidthco.com (192.168.1.1):
(The 64989 ports scanned but not shown below are in state: closed)
PORT      STATE    SERVICE            VERSION
21/tcp    open     ftp
22/tcp    open     ssh
25/tcp    open     smtp
53/tcp    open     domain
80/tcp    open     http
3000/tcp  filtered ppp
6000/tcp  open     X11
10000/tcp filtered snet-sensor-mgmt
Device type: general purpose
Running: Linux 2.4.X
OS details: Linux Kernel 2.4.19 - 2.4.20
Uptime 39.065 days (since Sun Feb 29 10:20:18 2004)
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=2973786 (Good luck!)
IPID Sequence Generation: Incremental
# Nmap run completed at Thu Apr 08 12:54:18 2004 -- 1 IP address (1 host up) scanned in 17.828 seconds
```

The results confirm and verify the firewall configuration as expected. Nmap identified all the ports as "open" that should be open, and did not identify any as open that should not be open. Additionally, nmap successfully identified the correct operating system down to the "kernel version" in all four tests. Interestingly enough, nmap also triggered an alert picked up by LogCheck suggestion the security on the mail relay is performing correctly as well:

```
Active System Attack Alerts
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Apr 10 19:53:29 server5 sendmail-in[8661]: i3B2rT82008661: server6.bandwidthco.com [192.168.1.10]: EXPN
root [rejected] Apr 10 19:53:29 server5 sendmail-in[8661]: i3B2rT82008661: server6.bandwidthco.com
[192.168.1.10]: VRFY root [rejected]
```

2. **Nessus Scanning**

Nessus will provide a complete assessment of the firewall security and vulnerability. The report below was generated after selecting "All Non-DOS Pluggins[33]" and "All Port Scanners". Irrelevant information has been redacted:

```
NESSUS SECURITY SCAN REPORT
Created 11.04.2004          Sorted by host names
Session Name : Session1
Start Time   : 10.04.2004 19:51:40
Finish Time  : 10.04.2004 19:58:10
Elapsed Time : 0 day(s) 00:06:29
Preferences settings for this scan:

  max_hosts                 = 16
  max_checks                = 10
  log_whole_attack          = yes
  report_killed_plugins     = yes
```

[33] For Nessus version 1.2.7, this means approximately 1,030 pluggins were loaded for the scan, and thus the firewall was tested for approximately that number of known vulnerabilities.

```
  cgi_path                         = /cgi-bin
  port_range                       = 1-1024
  optimize_test                    = yes
  language                         = english
  per_user_base                    = /var/lib/nessus/users
  checks_read_timeout              = 5
  delay_between_tests              = 1
  non_simult_ports                 = 139, 445
  plugins_timeout                  = 320
  safe_checks                      = yes
  auto_enable_dependencies         = no
  use_mac_addr                     = no
  save_knowledge_base              = yes
  plugin_upload_suffixes           = .nasl
  admin_user                       = root
  ntp_save_sessions                = yes
  ntp_detached_sessions            = yes
  server_info_nessusd_version      = 1.2.7
  server_info_libnasl_version      = 1.2.7
  server_info_libnessus_version    = 1.2.7
  server_info_thread_manager       = fork
  server_info_os                   = Linux
  server_info_os_version           = 2.4.25
  reverse_lookup                   = no
  ntp_keep_communication_alive     = yes
  ntp_opt_show_end                 = yes
  save_session                     = yes
  detached_scan                    = no
  continuous_scan                  = no

Total security holes found : 11
        high severity : 0
        low severity : 2
        informational : 9

Scanned hosts:

Name                    High  Low   Info
-------------------------------------------------
192.168.0.10              0    2    9
Host: 192.168.0.10
Open ports:
  ssh (22/tcp)
  x11 (6000/tcp)
  ntp (123/udp)
Service: ssh (22/tcp)
Severity: Low

An unknown service is running on this port.
It is usually reserved for SSH

Remote SSH server banner :

###############################################################
*******************************************************************************

This is a private system!!! All connection attempts are logged and
monitored. All unauthorized connection attempts will be investigated and
handed over to the proper authorities.
```

```
**********************************************************************
##################################################################
Service: ntp (123/udp)
Severity: Low
Risk factor : Low

Service: ntp (123/udp)
Severity: Low

An NTP server is running on the remote host. Make sure that
you are running the latest version of your NTP server,
has some versions have been found out to be vulnerable to
buffer overflows.

*** Nessus reports this vulnerability using only
*** information that was gathered. Use caution
*** when testing without safe checks enabled.

If you happen to be vulnerable : upgrade
Solution : Upgrade
Risk factor : High
CVE : CVE-2001-0414
```

Again, the nessus report confirms the firewall configuration to be solid:

- "Open Ports" and services were reported as expected.
- Eleven potential vulnerabilities were found and reported. One was reported as "low risk", nine were "informational" only, and one was "high risk".
- The high risk vulnerability (CVE: CVE-2001-0414), associated with NTP, should further investigated. We find that this vulnerability is not present in the NTP version provided with the SUSE 9 installation.

Conclusion

The SUSE 9 Linux firewall has now been subjected to very rigorous testing and the results are highly favorable. We can now, at this point, feel very confident, that the firewall is highly secure and ready for "production" service. The time has come to say "mission accomplished"[34].

[34] The words of George Herbert Walker Bush from the deck of the USS Abraham Lincoln, May 1, 2003.

APPENDICES

Appendix A: /etc/logrotate.conf File

```
###############################################################################
# logrotate.conf
# global configuration file for logrotate
###############################################################################

#####################################
# INCLUDE
#####################################
Include /etc/logrotate.d

#####################################
# DEFAULTS
#####################################
compress
daily
rotate 10
maxage 120
size +9999k
dateext
create
missingok
notifempty

#####################################
# SECURITY & INTRUSION LOGS
#####################################
# iptables
/var/log/iptables/iptables.log {
      missingok
      notifempty
      rotate 30
      dateext
      daily
      nocompress
      create 640 root root
      postrotate
            /etc/init.d/syslog reload
      endscript
}

# iptraf
/var/log/iptraf/iptraf.log {
      missingok
      notifempty
      rotate 30
      dateext
      daily
      nocompress
}

# snort syslog
/var/log/snort/snort.log {
      missingok
      notifempty
      rotate 30
      dateext
```

```
        daily
        nocompress
        postrotate
                /etc/init.d/syslog reload
        endscript
}

# snort full
/var/log/snort/snort_full {
        missingok
        notifempty
        rotate 3
        dateext
        daily
        nocompress
        create 640
}

# tcpdump
/var/log/tcpdump/tcpdump.dmp {
        missingok
        notifempty
        rotate 3
        dateext
        daily
        nocompress
        create 640 root root
}

#####################################
# SYSTEM SPECIFIC LOGS
#####################################
# cron
/var/log/cron/cron {
        missingok
        notifempty
        rotate 10
        postrotate
                /etc/init.d/syslog reload
        endscript
}

# authpriv
/var/log/security/authpriv {
        create 600 root root
        nocompress
        weekly
        rotate 10
        postrotate
                /etc/init.d/syslog reload
        endscript
}

# auth
/var/log/security/authinfo {
        create 600 root root
        nocompress
        weekly
        rotate 10
        postrotate
                /etc/init.d/syslog reload
        endscript
}

# sudo
/var/log/security/sudo.log {
        create 600 root root
```

```
        nocompress
        weekly
        rotate 10
        postrotate
                /etc/init.d/syslog reload
        endscript
}

/var/log/security/sudolog {
        create 600 root root
        nocompress
        weekly
        rotate 10
}

/var/log/security/security {
        create 600 root root
        nocompress
        weekly
        rotate 10
        postrotate
                /etc/init.d/syslog reload
        endscript
}

# wtmp
/var/log/wtmp {
        monthly
        create 0664 root utmp
        rotate 10
}
########################################################################
```

Appendix B: /etc/syslog.conf File

```
########################################################################
# /etc/syslog.conf (firewall machines)
########################################################################

########################################################################
# REMOTE LOGGING
########################################################################
# must use -h to enable logging to remote server
# must use -r to accept logging from remote host
*.*    @192.168.0.3

########################################################################
# CONSOLE LOGGING
########################################################################
kern.warn;*.err;authpriv.none          |/dev/xconsole
kern.warn;*.err;authpriv.none          /dev/console

########################################################################
# EMERGENCIES
########################################################################
*.emerg                 *

########################################################################
# SECURITY
########################################################################
local5.*                /var/log/security/sudo.log
security.*              /var/log/security/security
auth.info               /var/log/security/authinfo
authpriv.info           /var/log/security/authpriv
```

```
#########################################################################
# WARNINGS & ERRORS
#########################################################################
*.=warn;*.=err;*.crit           /var/log/security/warn

#########################################################################
# BOOT
#########################################################################
local7.*                        /var/log/boot.log

#########################################################################
# CRON
#########################################################################
cron.*                          /var/log/cron/cron

#########################################################################
# FIREWALL
#########################################################################
kern.=debug                     /var/log/iptables/iptables.log

#########################################################################
# SNORT
#########################################################################
local6.=alert                   /var/log/snort/snort.log

#########################################################################
# MAIL
#########################################################################
mail.*                          /var/log/mail/mail

#########################################################################
# MESSAGES
#########################################################################
*.*;mail.none;auth, authpriv.none;security.none;kern.!=debug /var/log/messages

#########################################################################
# ALL MESSAGES
#########################################################################
*.*                             /var/log/all/allmessages

#########################################################################
# LOCAL MESSAGE FACILITIES
#########################################################################
local0,local1.*         /var/log/local/localmessages
local2,local3.*         /var/log/local/localmessages
local4,local5.*         /var/log/local/localmessages
local6,local7.*         /var/log/local/localmessages
#########################################################################
```

Appendix C: /etc/sudoers File

```
#########################################################################
# /etc/sudoers
#########################################################################
# NOTE: This file MUST be edited with the "visudo" command as root.
#########################################################################
#########################
# DEFAULTS
#########################
Defaults  syslog=local5, logfile=/var/log/security/sudolog, \
          mail_no_user, mail_no_host, mail_no_perms, mail_badpass, \
          mailto=administrator@network.com
```

```
#########################
# HOST ALIAS SPECIFICATION
#########################
Host_Alias     FIREWALLS = server3,server6
Host_Alias     MAILSERVERS = server4,server5
Host_Alias     DNSSERVERS = server4,server5
Host_Alias     WEBSERVERS = server4,server5
Host_Alias     NTPSERVERS = server3,server6
Host_Alias     MONSERVERS = server4,server5
Host_Alias     INTERNAL = 192.168.0.0/255.255.255.0
Host_Alias     SERVICE = 192.168.1.0/255.255.255.0
Host_Alias     ADMIN = 192.168.2.0/255.255.255.0

#########################
# USER ALIAS SPECIFICATION
#########################
User_Alias     FWADMINS = fwadmin1,fwadmin2
User_Alias     DNSADMINS = markee
User_Alias     DBA = markee

#########################
# CMND ALIAS SPECIFICATION
#########################
Cmnd_Alias     SHELLS = /bin/sh, /bin/csh, /bin/bash, /bin/ksh
Cmnd_Alias     SU = /usr/bin/su
Cmnd_Alias     BACKUP = /bin/mt, /sbin/restore, /sbin/dump
Cmnd_Alias     SHUT = /usr/sbin/shutdown, /usr/sbin/halt, /usr/sbin/reboot
Cmnd_Alias     PASS= /usr/sbin/vipw, /usr/bin/passwd
Cmnd_Alias     DB = /bin/mt, /sbin/restore, /sbin/dump
Cmnd_Alias     FW = /usr/local/sbin/iptables, /usr/bin/iptables
Cmnd_Alias     DNS = /usr/sbin/ndc, /usr/sbin/rndc

#########################
# RUNAS ALIAS SPECIFICATION
#########################
Runas_Alias = APACHEADMIN = wwwrun
Runas_Alias = SNORTADMIN = snort
Runas_Alias = MAILADMIN = mspclient
Runas_Alias = DNSADMIN = named
Runas_Alias = ORACLEADMIN = oracle

#########################
# USER PRIVILEGE RULE SPECIFICATIONS
#########################
# SYNTAX: username    host = (runas alias) command
root ALL=(ALL) ALL
FWADMINS    FIREWALLS =(ALL)      ALL
FWADMINS    FIREWALLS =(ALL)      NOPASSWD: FW
#########################
```

Appendix D: /etc/ntp.conf File

```
##############################################################################
# /etc/ntp.conf (stratum 3)
##############################################################################
##############################################################################
# NOTES
# Need to set NTP parameters in sysconfig
# Set on AUTO-2
##############################################################################
##############################################################################
# INTERNET STRATUM 2 SERVERS TO USE
```

```
##############################################################################
# Use the following Stratum NTP Servers on the Internet
# NTP1 = 192.5.41.209
server ntp2.usno.navy.mil prefer
# NTP2 = 192.5.41.41
server tock.usno.navy.mil
# NTP3 = 192.5.41.40
server tick.usno.navy.mil
# NTP4 = 131.216.1.101
server cuckoo.nevada.edu

##############################################################################
# PEER STRATUM 3 SERVERS
##############################################################################
peer 192.168.0.10
peer 192.168.1.10
peer 192.168.2.10
peer 192.168.3.10
peer 192.168.0.2
peer 192.168.1.2
peer 192.168.2.2

##############################################################################
# PSEUDO CLOCKS (UNDISCIPLINED LOCAL CLOCK)
##############################################################################
# The pseudo clock causes server to synchronize to its own system clock. The pseudo
# clock should be set at least two points higher than the stratum it usually operates at.

# local clock (LCL)
server 127.127.1.0

# uncomment the following line for Stratum 3 Servers
fudge  127.127.1.0 stratum 5

##############################################################################
# SECURITY POLICY
##############################################################################
# default policy is to ignore all
restrict default                ignore

# Permit synchronization with Stratum 2 NTP Servers
# but do not let them query this server or modify this configuration file
restrict 192.5.41.209           nomodify noquery
restrict 192.5.41.41            nomodify noquery
restrict 192.5.41.40            nomodify noquery
restrict 131.216.1.101          nomodify noquery

# Permit synchronization with peer Stratum 3 NTP Servers
# but do not let them query this server or modify this configuration file
restrict 192.168.0.10           nomodify noquery
restrict 192.168.1.10           nomodify noquery
restrict 192.168.2.10           nomodify noquery
restrict 192.168.3.10           nomodify noquery
restrict 192.168.0.2            nomodify noquery
restrict 192.168.1.2            nomodify noquery
restrict 192.168.2.2            nomodify noquery

# Allow administration, query, and debugging of Stratum 3 Server from the 192.168.0.0/24 network only
restrict 192.168.0.0 mask 255.255.255.0      nomodify

#################
# LOOPBACK POLICY
#################
# place no restrictions on loopback address
restrict 127.0.0.1


##############################################################################
```

```
# DRIFTFILE
#################################################################################
driftfile /var/lib/ntp/ntp.drift

#################################################################################
# LOGFILE
#################################################################################
logfile   /var/log/ntp/ntp

#################################################################################
# STATISTICS
#################################################################################
statsdir /tmp/
filegen peerstats  file peerstats  type day enable
filegen loopstats  file loopstats  type day enable
filegen clockstats file clockstats type day enable

#################################################################################
# AUTHENTIFICATION
#################################################################################
keys /etc/ntp.keys
trustedkey 1 2 3 4 5 6 14 15
requestkey 15
controlkey 15
#################################################################
```

Appendix E: /etc/mail/mspclient.mc File

```
divert(-1)
#################################################################
# mspclient.mc
#################################################################
# NOTE: When using this feature, the remote recieving MTA believes the mail has been
# generated locally, so the host or host IP must be listed in the "local-host-names"
# file of the receiving MTA.
# NOTE: You can also simply hack the submit.cf file provided by your vendor.
# Just look for the line that reads: D{MTAHost}[192.168.1.1]
#################################################################
include(`/usr/share/sendmail/m4/cf.m4')
divert(0)dnl
dnl #################################################################
dnl VERSION & OS TYPE DECLARATION
dnl #################################################################
VERSIONID(`$Id: submit.mc,v 8.6.2.4 2002/12/29 03:54:34 ca Exp $')
define(`confCF_VERSION', `Submit')dnl
define(`__OSTYPE__',`')dnl
define(`_USE_DECNET_SYNTAX_', `1')dnl
define(`confTIME_ZONE', `USE_TZ')dnl
define(`confDONT_INIT_GROUPS', `True')dnl
FEATURE(`msp', `[192.168.1.1]')dnl
#################################################################
```

Appendix F: /etc/init.d/sendmailmsp File

```
#!/bin/sh
#########################################################################
# Script:       sendmailmsp
# Description:  sendmail msp client startup script
# Author:       Mark E. Donaldson
```

```
# Date:          March 18, 2004
# Run Time:    Start the Sendmail MSP with no MTA
######################################################################
######################################################################
# TEST FOR SYSCONFIG SETTINGS
######################################################################
test -s /etc/sysconfig/mail && . /etc/sysconfig/mail
test -s /etc/sysconfig/sendmail && . /etc/sysconfig/sendmail

##############
# MSP
##############
if test -z "$SENDMAIL_CLIENT_ARGS" ;
then
      SENDMAIL_CLIENT_ARGS="-L sendmail-client -Ac -qp30m"
fi

######################################################################
# PID FILE VARIABLES
######################################################################
msppid=/var/spool/clientmqueue/sm-client.pid

######################################################################
# STARTUP SENDMAIL MSP
######################################################################
. /etc/rc.status
rc_reset
case "$1" in
   start)
      echo -n "Initializing sendmail MSP"
      rc_status
      startproc -f -p $msppid /usr/sbin/sendmail $SENDMAIL_CLIENT_ARGS
      rc_status -v
      ;;
   stop)
      echo -n "Shutting down sendmail MSP"
      killproc -p $msppid -TERM /usr/sbin/sendmail
      rc_status
      rc_status -v
      ;;
   try-restart)
      $0 stop && $0 start
      rc_status
      ;;
   restart)
      $0 stop
      $0 start
      rc_status
      ;;
   reload|force-reload)
      echo -n "Reload MSP sendmail"
      killproc -p $msppid -HUP /usr/sbin/sendmail
      rc_status -v
      ;;
   status)
      checkproc -p $msppid /usr/sbin/sendmail
      rc_status -v
      ;;
   probe)
      test /etc/mail/submit.cf -nt $msppid \
      && echo reload
      ;;
   *)
      echo "Usage: $0 {start|stop|status|try-restart|restart|force-reload|reload|probe}"
      exit 1
esac
rc_exit
```

Appendix G: Portsentry Configuration File

```
##################################################################
# /usr/local/psionic/portsentry/portsentry.conf
##################################################################
#####################
# Port Configurations
#####################
# Un-comment these if you are really anal:
TCP_PORTS="1,7,9,11,15,70,79,80,109,110,111,119,138,139,143,512,513,514,515,540,635,1080,1524,2000,2001,4000,4001,574
2,6000,6001,6667,12345,12346,20034,27665,30303,32771,32772,32773,32774,31337,40421,40425,49724,54320"
UDP_PORTS="1,7,9,66,67,68,69,111,137,138,161,162,474,513,517,518,635,640,641,666,700,2049,31335,27444,34555,32770,32
771,32772,32773,32774,31337,54321"

########################################
# Advanced Stealth Scan Detection Options
########################################
ADVANCED_PORTS_TCP="1024"
ADVANCED_PORTS_UDP="1024"
ADVANCED_EXCLUDE_TCP="113,139"
ADVANCED_EXCLUDE_UDP="520,138,137,67"

#####################
# Configuration Files
#####################
IGNORE_FILE="/usr/local/psionic/portsentry/portsentry.ignore"
HISTORY_FILE="/usr/local/psionic/portsentry/portsentry.history"
BLOCKED_FILE="/usr/local/psionic/portsentry/portsentry.blocked"

##############################
# Configuration Options
##############################
RESOLVE_HOST = "0"

#################
# Ignore Options
#################
# 0 = Do not block UDP/TCP scans.
# 1 = Block UDP/TCP scans.
# 2 = Run external command only (KILL_RUN_CMD)
BLOCK_UDP="1"
BLOCK_TCP="1"

##################
# Packet Filter
##################
KILL_ROUTE="/usr/local/bin/iptables -I INPUT -s $TARGET$ -j DROP"

###############
# TCP Wrappers
###############
KILL_HOSTS_DENY="ALL: $TARGET$"

##################
# External Command
##################
KILL_RUN_CMD_FIRST = "1"
KILL_RUN_CMD="/usr/local/banners/portsentry_killall.sh $TARGET$ $PORT$"

####################
# Scan trigger value
####################
```

```
SCAN_TRIGGER="0"

#####################
# Port Banner Section
#####################
PORT_BANNER="** UNAUTHORIZED ACCESS PROHIBITED *** YOUR CONNECTION ATTEMPT HAS BEEN LOGGED."
#########################################################################
```

Appendix H: LogSentry Configuration File

```
#!/bin/sh
#########################################################################
# logcheck.sh
# configuration file for logcheck
#########################################################################
#########################################################################
# CONFIGURATION SECTION
#########################################################################
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/ucb:/usr/local/bin
SYSADMIN="root"
LOGTAIL=/usr/local/bin/logtail
TMPDIR=/usr/local/etc/tmp
GREP=egrep
MAIL=mail
HACKING_FILE=/usr/local/etc/logcheck.hacking
VIOLATIONS_FILE=/usr/local/etc/logcheck.violations
VIOLATIONS_IGNORE_FILE=/usr/local/etc/logcheck.violations.ignore
IGNORE_FILE=/usr/local/etc/logcheck.ignore
HOSTNAME=`hostname`
DATE=`date +%m/%d/%y:%H.%M`
umask 077

rm -f $TMPDIR/check.$$ $TMPDIR/checkoutput.$$ $TMPDIR/checkreport.$$

if [ -f $TMPDIR/check.$$ -o -f $TMPDIR/checkoutput.$$ -o -f $TMPDIR/checkreport.$$ ];
then
      echo "Log files exist in $TMPDIR directory that cannot be removed. Possible attempt to spoof the log checker."  \
      | $MAIL -s "$HOSTNAME $DATE ACTIVE SYSTEM ATTACK!" $SYSADMIN
      exit 1
fi

#########################################################################
# LOG FILE CONFIGURATION SECTION
#########################################################################
$LOGTAIL /var/log/messages > $TMPDIR/check.$$
$LOGTAIL /var/log/mail/mail >> $TMPDIR/check.$$
$LOGTAIL /var/log/iptables/iptables.log >> $TMPDIR/check.$$
$LOGTAIL /var/log/snort/snort.log >> $TMPDIR/check.$$
$LOGTAIL /var/log/security/security.log >> $TMPDIR/check.$$

if [ "$HOSTNAME" = "server4" ] || [ "$HOSTNAME" = "server5" ];
then
      $LOGTAIL /var/lib/named/log/bind_log >> $TMPDIR/check.$$
fi

#########################################################################
# LOG CHECKING SECTION
#########################################################################
FOUND=0
ATTACK=0

if [ ! -s $TMPDIR/check.$$ ]; then
      rm -f $TMPDIR/check.$$
```

```
        exit 0
fi

if [ -f "$HACKING_FILE" ]; then
        if $GREP -i -f $HACKING_FILE $TMPDIR/check.$$ > $TMPDIR/checkoutput.$$; then
                echo >> $TMPDIR/checkreport.$$
                echo "Active System Attack Alerts" >> $TMPDIR/checkreport.$$
                echo "=-=-=-=-=-=-=-=-=-=-=-=-=-=" >> $TMPDIR/checkreport.$$
                cat $TMPDIR/checkoutput.$$ >> $TMPDIR/checkreport.$$
                FOUND=1
                ATTACK=1
        fi
fi

if [ -f "$VIOLATIONS_FILE" ]; then
        if $GREP -i -f $VIOLATIONS_FILE $TMPDIR/check.$$ |
          $GREP -v -f $VIOLATIONS_IGNORE_FILE > $TMPDIR/checkoutput.$$; then
                echo >> $TMPDIR/checkreport.$$
                echo "Security Violations" >> $TMPDIR/checkreport.$$
                echo "=-=-=-=-=-=-=-=-=-=" >> $TMPDIR/checkreport.$$
                cat $TMPDIR/checkoutput.$$ >> $TMPDIR/checkreport.$$
                FOUND=1
        fi
fi

if [ -f "$IGNORE_FILE" ]; then
        if $GREP -v -f $IGNORE_FILE $TMPDIR/check.$$ > $TMPDIR/checkoutput.$$; then
                echo >> $TMPDIR/checkreport.$$
                echo "Unusual System Events" >> $TMPDIR/checkreport.$$
                echo "=-=-=-=-=-=-=-=-=-=-=" >> $TMPDIR/checkreport.$$
                cat $TMPDIR/checkoutput.$$ >> $TMPDIR/checkreport.$$
                FOUND=1
        fi
fi

if [ "$ATTACK" -eq 1 ]; then
        cat $TMPDIR/checkreport.$$ | $MAIL -s "$HOSTNAME $DATE LOGCHECK: ACTIVE SYSTEM ATTACK!" $SYSADMIN
elif [ "$FOUND" -eq 1 ]; then
        cat $TMPDIR/checkreport.$$ | $MAIL -s "$HOSTNAME $DATE LOGCHECK: SYSTEM CHECK" $SYSADMIN
fi

rm -f $TMPDIR/check.$$ $TMPDIR/checkoutput.$$ $TMPDIR/checkreport.$$
######################################################################
```

Appendix I: HostSentry Configuration File

```
######################################################################
# /usr/local/abacus/hostsentry/hostsentry.conf
######################################################################
######################################################################
# CRITICAL FILE PATHS
######################################################################
IGNORE_FILE = "/usr/local/abacus/hostsentry/hostsentry.ignore"
ACTION_FILE = "/usr/local/abacus/hostsentry/hostsentry.action"
MODULE_FILE = "/usr/local/abacus/hostsentry/hostsentry.modules"
MODULE_PATH = "/usr/local/abacus/hostsentry/modules"
WTMP_FILE = "/var/log/wtmp"
DB_FILE = "/usr/local/abacus/hostsentry/hostsentry.db"
DB_TTY_FILE = "/usr/local/abacus/hostsentry/hostsentry.tty.db"

######################################################################
# WTMP FILE FORMAT
######################################################################
WTMP_FORMAT = "384/8:32/44:32/76:256"
```

Appendix J: /etc/ssh/sshd_config File

```
#########################################################################
# Script:        /etc/ssh/sshd_config
# Description: This is the sshd server system-wide configuration file.
#########################################################################

#########################################################################
# LISTENING PORTS & ADDRESSES
#########################################################################
Port 22
Protocol 2

#########################################################################
# HOST KEYS
#########################################################################
HostKey /etc/ssh/ssh_host_key
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
KeyRegenerationInterval 3600
ServerKeyBits 768

#########################################################################
# ACCESS CONTROL (Allowed Users, Groups, and Hosts)
#########################################################################
AllowUsers fwadmin1 fwadmin2

#########################################################################
# LOGGING
#########################################################################
SyslogFacility AUTHPRIV
LogLevel INFO

#########################################################################
# AUTHENTICATION
#########################################################################
LoginGraceTime 120
PermitRootLogin no
StrictModes yes
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile      .ssh/authorized_keys
RhostsAuthentication no
IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no
IgnoreUserKnownHosts no
PasswordAuthentication no
PermitEmptyPasswords no
ChallengeResponseAuthentication no
KerberosAuthentication no
KerberosOrLocalPasswd no
KerberosTicketCleanup no
AFSTokenPassing no
KerberosTgtPassing no
GSSAPIAuthentication no
GSSAPIKeyExchange no
GSSAPIUseSessionCredCache no
GSSAPICleanupCreds no
PAMAuthenticationViaKbdInt no

#########################################################################
```

```
# GLOBAL OPTIONS
#####################################################################
X11Forwarding no
X11DisplayOffset 10
X11UseLocalhost no
PrintMotd yes
PrintLastLog yes
KeepAlive yes
UseLogin no
UsePrivilegeSeparation yes
PermitUserEnvironment no
Compression yes
MaxStartups 10
VerifyReverseMapping yes

#####################################################################
# LOGON BANNER
#####################################################################
Banner /etc/issue

#####################################################################
# VERFSION ADDENDUM
#####################################################################
VersionAddendum For Authorized Users Only!!!!!!

#####################################################################
# OVERRIDES
#####################################################################
Subsystem     sftp  /usr/lib/ssh/sftp-server
```

Appendix K: Software Integrity Verification Procedure

Generally, software packages will offer one of three means of integrity checking:

- MD5 checksums
- Cryptographic signatures using GnuPG, the GNU Privacy Guard
- The built-in RPM integrity verification mechanism for RPM packages

The following procedure describes how to use these methods.

Verifying MD5 Checksum
After downloading a package signed with an MD5 checksum, make sure that the MD5 checksum matches the one provided by the corresponding signature file or download page. Each package has an individual checksum that you can verify with the following command, where package_name is the name of the package you downloaded:

shell> md5sum package_name

For example:

shell> md5sum mkCDrec_v0.7.8.tar.gz
948c553ef9cd4cc254705df974721d33 *mkCDrec_v0.7.8.tar.gz

The provided signature was: **md5sum=948c553ef9cd4cc254705df974721d33**

You should verify that the resulting checksum (the string of hexadecimal digits) matches the one provided for the respective package.

Signature Checking Using GnuPG
A more reliable method of verifying the integrity and authenticity of a package is to use GnuPG (GNU Privacy Guard) cryptographic signatures. To verify the signature for a specific package, you first need to obtain a copy of the

packages public GPG build key. You can usually either cut and paste it directly from the download site or signature file, or obtain it from a key server. For example, a typical key may look like the following:

Key ID:
pub 1024D/5072E1F5 2003-02-03
Fingerprint: A4A9 4068 76FC BD3C 4567 70C8 8C71 8D3B 5072 E1F5

Public Key (ASCII-armored):

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.0.6 (GNU/Linux)
Comment: For info see http://www.gnupg.org

mQGiBD4+owwRBAC14GIfUfCyEDSIePvEW3SAFUdJBtoQHH/nJKZyQT7h9bPlUWC3
RODjQReyCITRrdwyrKUGku2FmeVGwn2u2WmDMNABLnpprWPkBdCk96+OmSLN9brZ
fw2vOUgCmYv2hW0hyDHuvYlQA/BThQoADgj8AW6/0Lo7V1W9/8VuHP0gQwCgvzV3
BqOxRznNCRCRxAuAuVztHRcEAJooQK1+iSiunZMYD1WufeXfshc57S/+yeJkegNW
hxwR9pRWVArNYJdDRT+rf2RUe3vpquKNQU/hnEIUHJRQqYHo8gTxvxXNQc7fJYLV
K2HtkrPbP72vwsEKMYhhr0eKCbtLGfls9krjJ6sBgACyP/Vb7hiPwxh6rDZ7ITnE
kYpXBACmWpP8NJTkamEnPCia2ZoOHODANwpUkP43I7jsDmgtobZX9qnrAXw+uNDI
QJEXM6FSbi0LLtZciNlYsafwAPEOMDKpMqAK6IyisNtPvaLd8lH0bPAnWqcyefep
rv0sxxqUEMcM3o7wwgfN83POkDasDbs3pjwPhxvhz6//62zQJ7Q7TXlTUUwgUGFj
a2FnZSBzaWduaW5nIGtleSAod3d3Lm15c3FsLmNvbSkgPGJ1aWxkQG15c3FsLmNv
bT6IXQQTEQIAHQUCPj6jDAUJCWYBgAULBwoDBAMVAwIDFgIBAheAAAoJElxxjTtQ
cuH1cY4AnilUwTXn8MatQOiG0a/bPxrvK/gCAJ4oinSNZRYTnblChwFaazt7PF3q
zIhMBBMRAgAMBQI+PqPRBYMJZgC7AAoJEElQ4SqycpHyJOEAn1mxHijft00bKXvu
cSo/pECUmppiAJ41M9MRVj5VcdH/KN/KjRtW6tHFPYhMBBMRAgAMBQI+QoIDBYMJ
YiKJAAoJELb1zU3GuiQ/lpEAoIhpp6BozKI8p6eaabzF5MlJH58pAKCu/ROofK8J
Eg2aLos+5zEYrB/LsrkCDQQ+PqMdEAgA7+GJfxbMdY4wslPnjH9rF4N2qfWsEN/l
xaZoJYc3a6M02WCnHl6ahT2/tBK2w1QI4YFteR47gCvtgb6O1JHffOo2HfLmRDRi
Rjd1DTCHqeyX7CHhcghj/dNRIW2Z0l5QFEcmV9U0Vhp3aFfWC4Ujfs3LU+hkAWzE
7zaD5cH9J7yv/6xuZVw411x0h4UqsTcWMu0iM1BzELqX1DY7LwoPEb/O9Rkbf4fm
Le11EzIaCa4PqARXQZc4dhSinMt6K3X4BrRsKTfozBu74F47D8llbf5vSYHbuE5p
/1oIDznkg/p8kW+3FxuWrycciqFTcNz215yyX39LXFnlLzKUb/F5GwADBQf+Lwqq
a8CGrRfsOAJxim63CHfty5mUc5rUSnTslGYEIOCR1BeQauyPZbPDsDD9MZ1ZaSaf
anFvwFG6Llx9xkU7tzq+vKLoWkm4u5xf3vn55VjnSd1aQ9eQnUcXiL4cnBGoTbOW
I39EcyzgslzBdC++MPjcQTcA7p6JUVsP6oAB3FQWg54tuUo0Ec8bsM8b3Ev42Lmu
QT5NdKHGwHsXTPtl0klk4bQk4OajHsiy1BMahpT27jWjJlMiJc+IWJ0mghkKHt92
6s/ymfdf5HkdQ1cyvsz5tryVI3Fx78XeSYfQvuuwqp2H139pXGEkg0n6KdUOetdZ
Whe70YGNPw1yjWJT1IhMBBgRAgAMBQI+PqMdBQkJZgGAAAoJElxxjTtQcuH17p4A
n3r1QpVC9yhnW2cSAjq+kr72GX0eAJ4295kl6NxYEuFApmr1+0uUq/SlsQ==
=YJkx
-----END PGP PUBLIC KEY BLOCK-----

This key can be imported into your personal public GPG keyring by using **gpg --import**.

After downloading and importing the public build key, download the desired software package and the corresponding signature. The signature file has the same name as the distribution file with an .asc or .sig extension. For example:

Distribution File: **coreutils-5.2.1.tar.gz**
Signature File: **coreutils-5.2.1.tar.gz.sig**

A typical signature file may appear as follows:

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.2.4 (GNU/Linux)

iD8DBQBAUg6+/dLerNMzy6ERAsiSAKCTlAUv4vVWAdduLnopDfeNTKcSfgCeONxk
o2V4sUCxcIC1jDPw8FKbr7c=
=vHVK
-----END PGP SIGNATURE-----Version: GnuPG v1.0.6 (GNU/Linux)

Make sure that both files are stored in the same directory and then run the following command to verify the signature for the distribution file:

shell> gpg --verify package_name.sig

For example:

shell> gpg --verify coreutils-5.2.1.tar.gz.sig

gpg: Warning: using insecure memory!
gpg: Signature made Mon 03 Feb 2003 08:50:39 PM MET using DSA key ID 5072E1F5
gpg: Good signature from www.gnu.org. The "Good signature" message confirms the integrity of the package.

Signature Checking Using RPM
For RPM packages, there is no separate signature. RPM packages actually have a built-in GPG signature and MD5 checksum. You can verify a package by running the following command:

shell> rpm --checksig package_name.rpm

Appendix L: Function set_kernel_params()

```
############################
# SET_KERNEL_PARAMS()
############################
set_kernel_params()
{
      ######################
      # DEFAULT STATE CONNECTION TIMEOUTS
      ######################
      # set tcp connection timeouts
      $ECHO "SETTING STATE CONNECTION TIMEOUTS TO DEFAULT VALUES"
      $LOGGER -p warning "SETTING STATE CONNECTION TIMEOUTS TO DEFAULT VALUES"

      if [ -e "$GENERIC_STATE_TIMEOUT" ];
      then
            $ECHO "SETTING GENERIC STATE CONNECTION TIMEOUT"
            $LOGGER -p warning "SETTING GENERIC STATE CONNECTION TIMEOUT"
            # $ECHO "30" > $GENERIC_STATE_TIMEOUT
      else
            $ECHO "GENERIC STATE CONNECTION TIMEOUT NOT ON SYSTEM"
            $LOGGER -p warning "GENERIC STATE CONNECTION TIMEOUT NOT ON SYSTEM"
      fi

      if [ -e "$ICMP_STATE_TIMEOUT" ];
      then
            $ECHO "SETTING ICMP STATE CONNECTION TIMEOUT"
            $LOGGER -p warning "SETTING ICMP STATE CONNECTION TIMEOUT"
            # $ECHO "30" > $ICMP_STATE_TIMEOUT
      else
            $ECHO "ICMP STATE CONNECTION TIMEOUT NOT ON SYSTEM"
            $LOGGER -p warning "ICMP STATE CONNECTION TIMEOUT NOT ON SYSTEM"
      fi

      if [ -e "$TCP_STATE_TIMEOUT" ];
      then
            $ECHO "SETTING TCP STATE CONNECTION TIMEOUT"
            $LOGGER -p warning "SETTING TCP STATE CONNECTION TIMEOUT"
            # $ECHO "30" > $TCP_STATE_TIMEOUT
      else
            $ECHO "TCP STATE CONNECTION TIMEOUT NOT ON SYSTEM"
            $LOGGER -p warning "TCP STATE CONNECTION TIMEOUT NOT ON SYSTEM"
      fi
```

```
#####################
# TCP CONNECTION TIMEOUT (FIN)
#####################
# set tcp connection timeouts
if [ -e "$TIMEOUT" ];
then
      $ECHO "SETTING TCP CONNECTION TIMEOUT TO DEFAULT VALUE"
      $LOGGER -p warning "SETTING TCP CONNECTION TIMEOUT TO DEFAULT VALUE"
      # $ECHO "30" > $TIMEOUT
else
      $ECHO "TCP CONNECTION TIMEOUT NOT ON SYSTEM"
      $LOGGER -p warning "TCP CONNECTION TIMEOUT NOT ON SYSTEM"
fi

#####################
# NUMBER STATE TABLE ENTRIES
#####################
# set variable for controlling number of state table entries
if [ -e "$STATE_CONNECTIONS" ];
then
      $ECHO "SETTING STATE TABLE ENTRIES PERMITTED"
      $LOGGER -p warning "SETTING STATE TABLE ENTRIES PERMITTED"
      # $ECHO "30000" > $STATE_CONNECTIONS
else
      $ECHO "STATE TABLE ENTRIES CANNOT BE CONFIGURED ON SYSTEM"
      $LOGGER -p warning "STATE TABLE ENTRIES CANNOT BE CONFIGURED ON SYSTEM"
fi

#####################
# TCP KEEPALIVE INTERVAL
#####################
# set tcp keep alive
if [ -e "$KEEP_ALIVE" ];
then
      $ECHO "SETTING TCP KEEP ALIVE TO DEFAULT VALUE"
      $LOGGER -p warning "SETTING TCP KEEP ALIVE TO DEFAULT VALUE"
      # $ECHO "1800" > $KEEP_ALIVE
else
      $ECHO "TCP KEEP ALIVE INTERVAL CANNOT BE CONFIGURED ON SYSTEM"
      $LOGGER -p warning "TCP KEEP ALIVE INTERVAL CANNOT BE CONFIGURED ON SYSTEM"
fi

#####################
# IGNORE ICMP BROADCASTS
#####################
# ignore ICMP broadcasts
if [ -e "$ICMP_ECHO_BROADCASTS" ];
then
      $ECHO "SETTING IGNORE ICMP_ECHO_BROADCASTS TO TRUE"
      $LOGGER -p warning "SETTING IGNORE ICMP_ECHO_BROADCASTS TO TRUE"
      $ECHO "1" > $ICMP_ECHO_BROADCASTS
else
      $ECHO "IGNORE ICMP_ECHO_BROADCASTS CANNOT BE CONFIGURED ON SYSTEM"
      $LOGGER -p warning "IGNORE ICMP_ECHO_BROADCASTS CANNOT BE CONFIGURED ON SYSTEM"
fi

#####################
# ROUTE VERIFICATION
#####################
# set up route verification ( built-in egress & ingress filtering) apply to all interfaces on machine
if [ -e "$RP_FILTER" ];
then
      $ECHO "SETTING ROUTE VERIFICATION TO TRUE"
      $LOGGER -p warning "SETTING ROUTE VERIFICATION TO TRUE"
      for f in $RP_FILTER;
      do
            $ECHO "1" > $f
```

```
        done
else
        $ECHO "ROUTE VERIFICATION CANNOT BE CONFIGURED ON SYSTEM"
        $LOGGER -p warning "ROUTE VERIFICATION CANNOT BE CONFIGURED ON SYSTEM"
fi

######################
# ACCEPT SOURCE ROUTING
######################
# drop packets that are source routed
if [ -e "$ACCEPT_SOURCE_ROUTE" ];
then
        $ECHO "SETTING ACCEPT_SOURCE_ROUTE TO FALSE"
        $LOGGER -p warning "SETTING ACCEPT_SOURCE_ROUTE TO FALSE"
        for f in $ACCEPT_SOURCE_ROUTE;
        do
                $ECHO "0" > $f
        done
else
        $ECHO "ACCEPT_SOURCE_ROUTE CANNOT BE CONFIGURED ON SYSTEM"
        $LOGGER -p warning "ACCEPT_SOURCE_ROUTE CANNOT BE CONFIGURED ON SYSTEM"
fi

######################
# ACCEPT ICMP INCOMING REDIRECTS
######################
# drop ICMP incoming redirect packets
if [ -e "$REDIRECTS" ];
then
        $ECHO "SETTING ACCEPT INCOMING ICMP REDIRECTS TO FALSE"
        $LOGGER -p warning "SETTING ACCEPT INCOMING ICMP REDIRECTS TO FALSE"
        for f in $REDIRECTS;
        do
                $ECHO "0" > $f
        done
else
        $ECHO "INCOMMING REDIRECTS CANNOT BE CONFIGURED ON SYSTEM"
        $LOGGER -p warning "INCOMMING REDIRECTS CANNOT BE CONFIGURED ON SYSTEM"
fi

######################
# PERMIT ICMP OUTGOING REDIRECTS
######################
# drop ICMP outgoing redirect packets
if [ -e "$MAKE_REDIRECTS" ];
then
        $ECHO "SETTING ACCEPT OUTGOING ICMP REDIRECTS TO FALSE"
        $LOGGER -p warning "SETTING ACCEPT OUTGOING ICMP REDIRECTS TO FALSE"
        for f in $MAKE_REDIRECTS;
        do
                $ECHO "0" > $f
        done
else
        $ECHO "OUTGOING REDIRECTS CANNOT BE CONFIGURED ON SYSTEM"
        $LOGGER -p warning "OUTGOING REDIRECTS CANNOT BE CONFIGURED ON SYSTEM"
fi

######################
# LOG MATIANS
######################
# log packets with  bad or impossible addresses
if [ -e "$LOG_MARTIANS" ];
then
        $ECHO "SETTING LOG_MARTIANS TO TRUE"
        $LOGGER -p warning "SETTING LOG_MARTIANS TO TRUE"
        for f in $LOG_MARTIANS;
        do
```

```
            $ECHO "1" > $f
        done
else
        $ECHO "LOG MARTIANS CANNOT BE CONFIGURED ON SYSTEM"
        $LOGGER -p warning "LOG MARTIANS CANNOT BE CONFIGURED ON SYSTEM"
fi

######################
# SYN COOKIE PROTECTION
######################
# enable TCP SYN cookie protection
if [ -e "$SYN_COOKIES" ];
then
        $ECHO "SETTING TCP SYN COOKIE PROTECTION TO TRUE"
        $LOGGER -p warning "SETTING TCP SYN COOKIE PROTECTION TO TRUE"
        $ECHO "1" > $SYN_COOKIES
else
        $ECHO "SYN COOKIES CANNOT BE CONFIGURED ON SYSTEM"
        $LOGGER -p warning "SYN COOKIES CANNOT BE CONFIGURED ON SYSTEM"
fi

######################
# BOGUS ERROR
######################
# ignore bogus error
if [ -e "$IGNORE_BOGUS" ];
then
        $ECHO "SETTING IGNORE BOGUS ERROR TO TRUE"
        $LOGGER -p warning "SETTING IGNORE BOGUS ERROR TO TRUE"
        $ECHO "1" > $IGNORE_BOGUS
else
        $ECHO "IGNORE BOGUS ERROR CANNOT BE CONFIGURED ON SYSTEM"
        $LOGGER -p warning "IGNORE BOGUS ERROR CANNOT BE CONFIGURED ON SYSTEM"
fi

######################
# DENY PING
######################
# ping ($ECHO requests)
if [ -e "$DENY_PING" ];
then
        $ECHO "SETTING DENY PING TO FALSE"
        $LOGGER -p warning "SETTING DENY PING TO FALSE"
        $ECHO "0" > $DENY_PING
else
        $ECHO "DENY PING CANNOT BE CONFIGURED ON SYSTEM"
        $LOGGER -p warning "DENY PING CANNOT BE CONFIGURED ON SYSTEM"
fi

######################
# PROXY ARP
######################
# support for arp proxy
if [ -e "$ARP_PROXY" ];
then
        $ECHO "SETTING ARP_PROXY TO FALSE"
        $LOGGER -p warning "SETTING ARP_PROXY TO FALSE"
        $ECHO "0" > $ARP_PROXY
else
        $ECHO "ARP_PROXY CANNOT BE CONFIGURED ON SYSTEM"
        $LOGGER -p warning "ARP_PROXY CANNOT BE CONFIGURED ON SYSTEM"
fi

######################
# DYNAMIC ADDRESSING
######################
# support for dynamic addressing
```

```
        if [ -e "$DYNAMIC_IP" ];
        then
                $ECHO "SETTING DYNAMIC_IP TO FALSE"
                $LOGGER -p warning "SETTING DYNAMIC_IP TO FALSE"
                $ECHO "0" > $DYNAMIC_IP
        else
                $ECHO "DYNAMIC_IP CANNOT BE CONFIGURED ON SYSTEM"
                $LOGGER -p warning "DYNAMIC_IP CANNOT BE CONFIGURED ON SYSTEM"
        fi
}
```

Appendix M: Script log_copy.pl

```
#####################################################################
# Program:              log_copy.pl
# Programmer:           Mark E. Donaldson
# Date:                 March 21, 2004
# Description:          Copy Log files to central location for analysis
#####################################################################
#####################################################################
#       PRAGMAS
#####################################################################
# must predefine all variables and requires use of my $ declarations
use strict;
use warnings;

#####################################################################
#       MODULES
#####################################################################
use Win32;
use Getopt::Std;
use IO::File;
use File::Copy;

#####################################################################
#       VARIABLE DECLARATIONS
#####################################################################
my $CWD;
my $CONFIG_FILE;
my $FROM;
my $TO;
my $FILE;
my $NEW_FILE;
my $FH;
my $DATE = localtime(time);
my $system_command;
my $computer_name = Win32::NodeName();
my $message = "Firewall Log From $computer_name has been delivered.";

# Log Directory location variables
my $IPTABLES_LOGS;
my $IPT_STATS;
my $APACHE_LOGS;
my $IIS_LOGS;
my $MAIL_LOGS;
my $SYSLOG_LOGS;
my $WINSYS_LOGS;
my $SNORT_ALERT_LOGS;
my $SNORT_LOG_LOGS;
my $SNORT_SCAN_LOGS;
my $SNORT_OOS_LOGS;
my $SNORT_DUMP_LOGS;
```

```
my $SNMP_LOGS;
my $IPTRAF_LOGS;
my $TCPDUMP_LOGS;
my $WINDUMP_LOGS;

# Executables program variables
my $PUTTY;
my $PAGEANT;
my $PSCP;
my $PFTP;
my $PLINK;
my $PUTTYGEN;
my $PS;
my $PROCESS;

# Flags for program control
my $APACHE_LOG_FLAG = "0";
my $FW_LOG_FLAG = "0";
my $FW_STATS_FLAG = "0";
my $IIS_LOG_FLAG = "0";
my $MAIL_LOG_FLAG = "0";
my $SYSTEM_LOG_FLAG = "0";
my $WINSYSLOG_LOG_FLAG = "0";
my $SNORT_ALERT_FLAG = "0";
my $SNORT_SCAN_FLAG = "0";
my $SNORT_OOS_FLAG = "0";
my $SNORT_DUMP_FLAG = "0";
my $SNMP_LOG_FLAG = "0";
my $IPTRAF_FLAG = "0";
my $TCPDUMP_FLAG = "0";
my $WINDUMP_FLAG = "0";
my $AGEANT = "0";


my $POUND = "################################################################################\n";

#####################################################################
#     FUNCTION DECLARATIONS
#####################################################################
sub print_main_heading();
sub get_date();
sub check_agent();
sub get_files();
sub get_args();
sub convert_month( $ );
sub get_config();
sub check_variables();
sub usage();

#####################################################################
#     MAIN
#####################################################################
# make modules in Program Class Library available for use
BEGIN
{
      $CWD = Win32::GetCwd();

      # initialize system configuration file
      # $CONFIG_FILE = "$CWD\\log_copy.conf";
      $CONFIG_FILE = "C:\\log_copy.conf";
      unshift ( @INC, "$CWD\\" );
}

#########################################################################
# START
#########################################################################
$DATE = get_date();
```

```
print_main_heading();

# @ARGV = qw ( -f -l -t -F server2 -T workstation1 );

if ( defined ( @ARGV ) )
{
	# my @ARGS = @ARGV;
	get_args();
	print "$POUND";
}

else
{
	&usage();
}

get_config();
check_variables();
check_agent();
get_files();
exit(0);

#########################################################################
# SUBROUTINE GET_ARGS()
#########################################################################
sub print_main_heading()
{
	print "$POUND";
	print "PROGRAM FILE COPY\n";
	print "Copies and Moves Files For Analysis\n";
	print "DATE: $DATE\n";
	print "$POUND";
}

#########################################################################
# SUBROUTINE GET_DATE()
#########################################################################
sub get_date()
{
	my ( $DAY, $MONTH, $YEAR ) = (localtime) [3, 4, 5];
	# $MONTH = $MONTH + 1;
	# $MONTH = convert_month( $MONTH );
	# $YEAR = $YEAR + 1900;
	$DATE = sprintf ( "%02d-%02d-%04d", $MONTH + 1, $DAY, $YEAR + 1900 );
}

#########################################################################
# SUBROUTINE GET_FILES()
#########################################################################
# copies desired files using SCP
sub get_files()
{
	print "$POUND";
	print "Retrieving Files . . . . . \n";

	# get tcpdump logs
	if ( $TCPDUMP_FLAG == 1 )
	{
		$FILE = "/var/log/tcpdump/tcpdump.dmp";
		$NEW_FILE = "tcpdump_$DATE.dmp";
		$system_command = "$PSCP -v -l root $FROM:$FILE $TCPDUMP_LOGS\\$NEW_FILE";

		eval
		{
			system ( $system_command ) || warn "$FILE Copy Failed: $!";
		};
```

```
}

# get Apache server web logs
if ( $APACHE_LOG_FLAG == 1 )
{
	$FILE = "/var/log/httpd/access_log";
	$NEW_FILE = "access_log_$DATE.log";
	$system_command = "$PSCP -v -l root $FROM:$FILE $APACHE_LOGS\\$NEW_FILE";

	eval
	{
		system ( $system_command ) || warn "$FILE Copy Failed: $!";
	};

	if ( $@ )
	{
		print "$@\n";
	}

	$FILE = "/var/log/httpd/error_log";
	$NEW_FILE = "error_log_$DATE.log";
	$system_command = "$PSCP -v -l root $FROM:$FILE $APACHE_LOGS\\$NEW_FILE";

	eval
	{
		system ( $system_command ) || warn "$FILE Copy Failed: $!";
	};

	if ( $@ )
	{
		print "$@\n";
	}
}

# get firewall (IPTables) logs
if ( $FW_LOG_FLAG == 1 )
{
	$FILE = "/var/log/iptables/iptables.log";
	$NEW_FILE = "iptables_$DATE.log";
	$system_command = "$PSCP -v -l root $FROM:$FILE $IPTABLES_LOGS\\$NEW_FILE";

	eval
	{
		system ( $system_command ) || warn "$FILE Copy Failed: $!";
	};

	if ( $@ )
	{
		print "$@\n";
	}

}

# get firewall (IPTables) stats
if ( $FW_STATS_FLAG == 1 )
{
	# filter table stats
	$FILE = "/var/log/report/iptables_stats.txt";
	$NEW_FILE = "iptables_stats_$DATE.txt";
	$system_command = "$PSCP -v -l root $FROM:$FILE $IPT_STATS\\$NEW_FILE";

	eval
	{
		system ( $system_command );
	};

	if ( $@ )
```

```
        {
                print "$@\n";
        }
}

# get IIS server web logs
if ( $IIS_LOG_FLAG == 1 )
{

}

# get mail logs
if ( $MAIL_LOG_FLAG == 1 )
{
        $FILE = "/var/log/mail/mail";
        $NEW_FILE = "mail_$DATE.txt";
        $system_command = "$PSCP -v -l root $FROM:$FILE $MAIL_LOGS\\$NEW_FILE";

        eval
        {
                system ( $system_command ) || warn "$FILE Copy Failed: $!";
        };

        if ( $@ )
        {
                print "$@\n";
        }
}

# get system logs
if ( $SYSTEM_LOG_FLAG == 1 )
{
        $FILE = "/var/log/messages";
        $NEW_FILE = "messages_$DATE.txt";
        $system_command = "$PSCP -v -l root $FROM:$FILE $SYSLOG_LOGS\\$NEW_FILE";

        eval
        {
                system ( $system_command ) || warn "$FILE Copy Failed: $!";
        };

        if ( $@ )
        {
                print "$@\n";
        }
}

# get ucd-snmp logs
if ( $SNMP_LOG_FLAG == 1 )
{
        $FILE = "/var/log/ucd-snmpd.log";
        $NEW_FILE = "ucd-snmp_$DATE.log";
        $system_command = "$PSCP -v -l root $FROM:$FILE $SNMP_LOGS\\$NEW_FILE";

        eval
        {
                system ( $system_command ) || warn "$FILE Copy Failed: $!";
        };

        if ( $@ )
        {
                print "$@\n";
        }
}

# get snort alert logs (ALERT_FULL) & Syslog snort.log
if ( $SNORT_ALERT_FLAG == 1 )
```

```
{
        $FILE = "/var/log/snort/snort_full";
        $NEW_FILE = "snort_full_$DATE.txt";
        $system_command = "$PSCP -v -l root $FROM:$FILE $SNORT_ALERT_LOGS\\$NEW_FILE";

        eval
        {
                system ( $system_command ) || warn "$FILE Copy Failed: $!";
        };

        if ( $@ )
        {
                print "$@\n";
        }

        $FILE = "/var/log/snort/snort.log";
        $NEW_FILE = "snort.log_$DATE.txt";
        $system_command = "$PSCP -v -l root $FROM:$FILE $SNORT_LOG_LOGS\\$NEW_FILE";

        eval
        {
                system ( $system_command ) || warn "$FILE Copy Failed: $!";
        };

        if ( $@ )
        {
                print "$@\n";
        }
}

# get snort scan logs
if ( $SNORT_SCAN_FLAG == 1 )
{
        $FILE = "/var/log/snort/scan.log";
        $NEW_FILE = "snort_scan_$DATE.txt";
        $system_command = "$PSCP -v -l root $FROM:$FILE $SNORT_SCAN_LOGS\\$NEW_FILE";

        eval
        {
                system ( $system_command ) || warn "$FILE Copy Failed: $!";
        };

        if ( $@ )
        {
                print "$@\n";
        }
}

# get snort dump logs
if ( $SNORT_DUMP_FLAG == 1 )
{
        $FILE = "/var/log/snort/snort.dmp";
        $NEW_FILE = "snort_$DATE.dmp";
        $system_command = "$PSCP -v -l root $FROM:$FILE $SNORT_DUMP_LOGS\\$NEW_FILE";

        eval
        {
                system ( $system_command ) || warn "$FILE Copy Failed: $!";
        };

        if ( $@ )
        {
                print "$@\n";
        }
}

# get iptraf logs
```

```
	if ( $IPTRAF_FLAG == 1 )
	{
		$FILE = "/var/log/iptraf/iptraf.log";
		$NEW_FILE = "iptraf_$DATE.log";
		$system_command = "$PSCP -v -l root $FROM:$FILE $IPTRAF_LOGS\\$NEW_FILE";

		eval
		{
			system ( $system_command ) || warn "$FILE Copy Failed: $!";
		};

		if ( $@ )
		{
			print "$@\n";
		}
	}
}

#############################################################################
# SUBROUTINE CHECK_AGENT()
#############################################################################
# checks to see if SSH Agent program is running
sub check_agent()
{
	# create temp file for scratch pad
	my $SCRATCH_FILE = "C:\\Temp\\scratch.txt";
	my $FH_SCRATCH;
	my @lines;
	my $line;

	print "$POUND";
	print "Checking SSH Ageant . . . . . . . . . . \n";

	# delete existing scratch file if exists
	if ( -e $SCRATCH_FILE )
	{
		unlink( $SCRATCH_FILE );
	}

	# dump running processes to find if agent is running
	$system_command = "$PROCESS > C:\\Temp\\scratch.txt";
	system ( $system_command );
	sleep (1);

	# create file handles for scratch file
	eval
	{
		$FH_SCRATCH = new IO::File( "C:\\Temp\\scratch.txt" ) || warn "FATAL ERROR: Failed to open Scratch File
$SCRATCH_FILE.\n";
	};

	if ( $@ )
	{
		print "$@\n";
		print "FATAL ERROR: Failed to open Scratch File $SCRATCH_FILE.\n";
		print "EXITING PROGRMA NOW!!\n";
		print "Press [ENTER] to terminate Program.\n";
		<STDIN>;
		exit(1);
	}

	# search scratch file for pageant process
	if ( defined ( $FH_SCRATCH ) )
	{
		@lines = $FH_SCRATCH->getlines();

		foreach $line ( @lines )
```

```
        {
            chomp ( $line );

            if ( $line !~ /pageant.exe/ )
            {
                next;
            }

            elsif ( $line =~ /pageant.exe/ )
            {
                $AGEANT = 1;
                last;
            }
            else
            {
                next;
            }
        }
    }

    if ( $AGEANT == 1 )
    {
        print "SSH Agent is running.\n";
        print "$POUND\n";
    }

    else
    {
        print "The SSH Agent does not appear to be running.\n";
        print "You must start the SSH Agent and Load private keys.\n";
        print "EXITING PROGRAM NOW!!.\n";
        print "Press [ENTER] to terminate Program.\n";
        print "$POUND\n";
        <STDIN>;
        exit(1);
    }
}

#########################################################################
# SUBROUTINE CHECK_VARIABLES()
#########################################################################
# subroutine checks to make sure all critical variables have been properly
# initialized from configuration file, and exits with error if not.
sub check_variables()
{
    print "$POUND";
    print "Checking for Proper Initialization of Executables and Directory Variables . . . . . \n";

    #############
    # EXECUTABLES
    #############
    if ( !defined( $PUTTY ) )
    {
        print "FATAL ERROR:  Variable for \"PUTTY\" Executable not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
    }

    if ( !defined( $PAGEANT ) )
    {
        print "FATAL ERROR:  Variable for \"PUTTY AGEANT\" Executable not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
```

```
        <STDIN>;
        exit(1);
}

if ( !defined( $PSCP ) )
{
        print "FATAL ERROR:  Variable for \"PUTTY SC\" Executable not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
}

if ( !defined( $PFTP ) )
{
        print "FATAL ERROR:  Variable for \"PUTTY FTP\" Executable not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
}

if ( !defined( $PLINK ) )
{
        print "FATAL ERROR:  Variable for \"PUTTY Link\" Executable not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
}

if ( !defined( $PUTTYGEN ) )
{
        print "FATAL ERROR:  Variable for \"PUTTY Key Generator\" Executable not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
}

if ( !defined( $PS ) )
{
        print "FATAL ERROR:  Variable for \"PS\" Executable not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
}

if ( !defined( $PROCESS ) )
{
        print "FATAL ERROR:  Variable for \"PROCESS\" Executable not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
}

#############
# DIRECTORIES
#############
```

```
if ( !defined( $IPTABLES_LOGS ) )
{
	print "FATAL ERROR:  Variable for IPTables Logs not initialized.\n";
	print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
	print "This program will now terminate!!\n";
	print "PRESS [ENTER] to exit.\n";
	<STDIN>;
	exit(1);
}

if ( !defined( $IPT_STATS ) )
{
	print "FATAL ERROR:  Variable for IPTables Stats not initialized.\n";
	print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
	print "This program will now terminate!!\n";
	print "PRESS [ENTER] to exit.\n";
	<STDIN>;
	exit(1);
}

if ( !defined( $APACHE_LOGS ) )
{
	print "FATAL ERROR:  Variable for Apache Logs not initialized.\n";
	print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
	print "This program will now terminate!!\n";
	print "PRESS [ENTER] to exit.\n";
	<STDIN>;
	exit(1);
}

if ( !defined( $IIS_LOGS ) )
{
	print "FATAL ERROR:  Variable for IIS Logs not initialized.\n";
	print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
	print "This program will now terminate!!\n";
	print "PRESS [ENTER] to exit.\n";
	<STDIN>;
	exit(1);
}

if ( !defined( $MAIL_LOGS ) )
{
	print "FATAL ERROR:  Variable for Sendmail Logs not initialized.\n";
	print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
	print "This program will now terminate!!\n";
	print "PRESS [ENTER] to exit.\n";
	<STDIN>;
	exit(1);
}

if ( !defined( $SYSLOG_LOGS ) )
{
	print "FATAL ERROR:  Variable for Syslog Logs not initialized.\n";
	print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
	print "This program will now terminate!!\n";
	print "PRESS [ENTER] to exit.\n";
	<STDIN>;
	exit(1);
}

if ( !defined( $WINSYS_LOGS ) )
{
	print "FATAL ERROR:  Variable for Winsyslog Logs not initialized.\n";
	print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
	print "This program will now terminate!!\n";
	print "PRESS [ENTER] to exit.\n";
	<STDIN>;
```

```
        exit(1);
}

if ( !defined( $SNORT_ALERT_LOGS ) )
{
        print "FATAL ERROR:  Variable for Snort Alert Logs not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
}

if ( !defined( $SNORT_LOG_LOGS ) )
{
        print "FATAL ERROR:  Variable for Snort Log Logs not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
}

if ( !defined( $SNORT_SCAN_LOGS ) )
{
        print "FATAL ERROR:  Variable for Snort Scan Logs not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
}

if ( !defined( $SNORT_OOS_LOGS ) )
{
        print "FATAL ERROR:  Variable for Snort OOS Logs not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
}

if ( !defined( $SNORT_DUMP_LOGS ) )
{
        print "FATAL ERROR:  Variable for Snort DUMP Logs not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
}

if ( !defined( $SNMP_LOGS ) )
{
        print "FATAL ERROR:  Variable for UCD-SNMP Logs not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
}

if ( !defined( $IPTRAF_LOGS ) )
{
        print "FATAL ERROR:  Variable for IPTraf Logs not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
```

```
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
    }

    if ( !defined( $TCPDUMP_LOGS ) )
    {
        print "FATAL ERROR:  Variable for TCPdump Logs not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
    }

    if ( !defined( $WINDUMP_LOGS ) )
    {
        print "FATAL ERROR:  Variable for Windump Logs not initialized.\n";
        print "Please confirm that the configuration file \"log_copy.conf\" is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
        exit(1);
    }
}

#########################################################################
# SUBROUTINE CONVERT_MONTH()
#########################################################################
sub convert_month( $ )
{
  my $month = shift( @_ );

  if ( ( $month eq "01" ) || ( $month eq "1" ) )
  {
        $month = "Jan";
        return $month;
  }

  elsif ( ( $month eq "02" ) || ( $month eq "2" ) )
  {
        $month = "Feb";
        return $month;
  }

  elsif ( ( $month eq "03" ) || ( $month eq "3" ) )
  {
        $month = "Mar";
        return $month;
  }

  elsif ( ( $month eq "04" ) || ( $month eq "4" ) )
  {
        $month = "Apr";
        return $month;
  }

  elsif ( ( $month eq "05" ) || ( $month eq "5" ) )
  {
        $month = "May";
        return $month;
  }

  elsif ( ( $month eq "06" ) || ( $month eq "6" ) )
  {
        $month = "Jun";
```

```
            return $month;
    }

    elsif ( ( $month eq "07" ) || ( $month eq "7" ) )
    {
            $month = "Jul";
            return $month;
    }

    elsif ( ( $month eq "08" ) || ( $month eq "8" ) )
    {
            $month = "Aug";
            return $month;
    }

    elsif ( ( $month eq "09" ) || ( $month eq "9" ) )
    {
            $month = "Sep";
            return $month;
    }

    elsif ( $month eq "10"  )
    {
            $month = "Oct";
            return $month;
    }

    elsif ( $month eq "11"  )
    {
            $month = "Nov";
            return $month;
    }

    elsif ( $month eq "12"  )
    {
            $month = "Dec";
            return $month;
    }

    else
    {
            return "Date Conversion Error";
    }
}

######################################################################
# SUBROUTINE GET_ARGS()
######################################################################
sub get_args()
{
      our $opt_F;
      our $opt_T;
      our $opt_a;
      our $opt_c;
      our $opt_d;
      our $opt_f;
      our $opt_i;
      our $opt_l;
      our $opt_m;
      our $opt_n;
      our $opt_o;
      our $opt_p;
      our $opt_s;
      our $opt_t;
      our $opt_u;
      our $opt_w;
      our $opt_h;
```

```
print "Processing Command-line Arguments . . . . . \n";

#########################################################################
# ARGUMENT KEY
#########################################################################
# F = Computer to copy from
# T = Computer to copy to
# a = Apache web logs
# c = Snort scan logs
# d = TCPdump logs
# f = Firewall logs
# i = IIS web logs
# l = Snort alert logs
# m = Mail logs
# n = Snort dump logs
# o = Snort OOS logs
# p = IPTraf logs
# s = System logs
# t = Firewall stats
# u = UCD-SNMP logs
# w  = Windump logs
# h = Help menu
#########################################################################

getopts('acdfhilmnopstuwF:T:');

#########################################################################
# Getopt::Std Usage
#########################################################################
# getopt('oDI');                    -o, -D & -I take arg.  Sets opt_* as a side effect.
# getopt('oDI', \%opts);    -o, -D & -I take arg.  Values in %opts
# getopts('oif:');                -o & -i are boolean flags, -f takes an argument
#########################################################################

# get Apache server web logs
if ( $opt_a )
{
        $APACHE_LOG_FLAG = 1;
        print "Apache Web Logs selected . . . . . \n";
}

# get Snort scan logs
if ( $opt_c )
{
        $SNORT_SCAN_FLAG = 1;
        print "Snort Scan Logs selected . . . . . \n";
}

# get TCPdump logs
if ( $opt_d )
{
        $TCPDUMP_FLAG = 1;
        print "TCPDump Logs selected . . . . . \n";
}

# get firewall (IPTables) logs
if ( $opt_f )
{
        $FW_LOG_FLAG = 1;
        print "Firewall Logs (IPTables) selected . . . . . \n";
}

# help
if ( $opt_h )
{
        usage();
```

```
}

# get IIS server web logs
if ( $opt_i )
{
        $IIS_LOG_FLAG = 1;
        print "IISe Web Logs selected . . . . . . . . . \n";
}

# get Snort alert logs
if ( $opt_l )
{
        $SNORT_ALERT_FLAG = 1;
        print "Snort Alert Logs selected . . . . . . . . . \n";
}

# get mail logs
if ( $opt_m )
{
        $MAIL_LOG_FLAG = 1;
        print "Mail (sendmail) Logs selected . . . . . . . . . . \n";
}

# get Snort dump logs
if ( $opt_n )
{
        $SNORT_DUMP_FLAG = 1;
        print "Snort Dump Logs selected . . . . . . . . . \n";
}

# get Snort OOS logs
if ( $opt_o )
{
        $SNORT_OOS_FLAG = 1;
        print "Snort OOS Logs selected . . . . . . . . . \n";
}

# get IPTraf logs
if ( $opt_p )
{
        $IPTRAF_FLAG = 1;
        print "IPTraf Logs selected . . . . . . . . . \n";
}

# get system logs
if ( $opt_s )
{
        $SYSTEM_LOG_FLAG = 1;
        print "System Logs selected . . . . . . . . . . \n";
}

# get firewall stats
if ( $opt_t )
{
        $FW_STATS_FLAG = 1;
        print "Firewall Stats (IPTables) selected . . . . . . . . . \n";
}

# get ucd-snmp logs
if ( $opt_u )
{
        $SNMP_LOG_FLAG = 1;
        print "UCD-SNMP Logs selected . . . . . . . . . \n";
}

# get windump logs
if ( $opt_w )
```

```
    {
        $WINDUMP_FLAG = 1;
        print "Windump Logs selected . . . . . \n";
    }

    # copy from which computer
    if ( $opt_F )
    {
        $FROM = $opt_F;
        print "Copying to be done from $FROM . . . . . \n";
    }

    # copy to which computer
    if ( $opt_T )
    {
        $TO = $opt_T;
        print "Copying to be done to $TO . . . . . \n";
    }

    else
    {
        usage();
    }
}

##########################################################################
# SUBROUTINE GET_CONFIG()
##########################################################################
# this subrouting is necessary for program startup
# it locates and parses the analysis.conf file to set needed system parameters
# into varaiables.  If the program is unable to locate this file, or if the
# file contains erronious information, this is a fatal error and the program
# abruptly terminates with error.
sub get_config()
{
    my $line;
    my @lines;
    my $split;

    print "Retrieving system configuration information . . . . . \n";

    eval
    {
        # creaete a filehandle for fport dump
        $FH = new IO::File( $CONFIG_FILE ) || die "Failed to open the system configuration file.\n";
    };

    if ( $@ )
    {
        print "$@\n";
        print "FATAL ERROR:  Unable to locate system configuration files.\n";
        print "Please confirm that this files exists and is properly configured.\n";
        print "This program will now terminate!!\n";
        print "PRESS [ENTER] to exit.\n";
        <STDIN>;
    }

    if ( defined ( $FH ) )
    {
        @lines = $FH->getlines();

        foreach $line ( @lines )
        {
            chomp ( $line );

            if ( $line =~ /^\#/ )
            {
```

```
        next;
    }
    elsif ( $line =~ /^\[/ )
    {
        next;
    }
    elsif ( $line =~ /^\s/ )
    {
        next;
    }
    elsif ( $line =~ /^putty/ )
    {
        ( $split, $PUTTY ) = split ( /=/, $line );
        print "Initializing PuTTy.exe at $PUTTY . . . . . \n";
    }
    elsif ( $line =~ /^pageant/ )
    {
        ( $split, $PAGEANT ) = split ( /=/, $line );
        print "Initializing Pageant.exe at $PAGEANT . . . . . \n";
    }
    elsif ( $line =~ /^pscp/ )
    {
        ( $split, $PSCP ) = split ( /=/, $line );
        print "Initializing pscp.exe at $PSCP . . . . . \n";
    }
    elsif ( $line =~ /^pftp/ )
    {
        ( $split, $PFTP ) = split ( /=/, $line );
        print "Initializing Pftp.exe at $PFTP . . . . . \n";
    }
    elsif ( $line =~ /^plink/ )
    {
        ( $split, $PLINK ) = split ( /=/, $line );
        print "Initializing Plink.exe at $PLINK . . . . . \n";
    }
    elsif ( $line =~ /^keygen/ )
    {
        ( $split, $PUTTYGEN ) = split ( /=/, $line );
        print "Initializing Puttygen.exe at $PUTTYGEN . . . . . \n";
    }
    elsif ( $line =~ /^ps/ )
    {
        ( $split, $PS ) = split ( /=/, $line );
        print "Initializing Ps.exe at $PS . . . . . \n";
    }
    elsif ( $line =~ /^process/ )
    {
        ( $split, $PROCESS ) = split ( /=/, $line );
        print "Initializing Process.exe at $PROCESS . . . . . \n";
    }
    elsif ( $line =~ /^iptableslogs/ )
    {
        ( $split, $IPTABLES_LOGS ) = split ( /=/, $line );
        print "Initializing IPTable Log Directory $IPTABLES_LOGS\ . . . . . \n";
    }
    elsif ( $line =~ /^iptstats/ )
    {
        ( $split, $IPT_STATS ) = split ( /=/, $line );
        print "Initializing IPTable Stats Directory $IPT_STATS\ . . . . . \n";
    }
    elsif ( $line =~ /^apachelogs/ )
    {
        ( $split, $APACHE_LOGS ) = split ( /=/, $line );
        print "Initializing Apache Log Directory $APACHE_LOGS\ . . . . . \n";
    }
    elsif ( $line =~ /^iislogs/ )
    {
```

```
        ( $split, $IIS_LOGS ) = split ( /=/, $line );
        print "Initializing IIS Log Directory $IIS_LOGS\ . . . . . . . . . . \n";
}
elsif ( $line =~ /^maillogs/ )
{
        ( $split, $MAIL_LOGS ) = split ( /=/, $line );
        print "Initializing Sendmail Log Directory $MAIL_LOGS\ . . . . . . . . . . \n";
}
elsif ( $line =~ /^systemlogs/ )
{
        ( $split, $SYSLOG_LOGS ) = split ( /=/, $line );
        print "Initializing Syslog Log Directory $SYSLOG_LOGS\ . . . . . . . . . . \n";
}
elsif ( $line =~ /^winsyslogs/ )
{
        ( $split, $WINSYS_LOGS ) = split ( /=/, $line );
        print "Initializing Winsyslog Log Directory $WINSYS_LOGS\ . . . . . . . . . . \n";
}
elsif ( $line =~ /^snmplogs/ )
{
        ( $split, $SNMP_LOGS ) = split ( /=/, $line );
        print "Initializing UCD-SNMP Log Directory $SNMP_LOGS\ . . . . . . . . . . \n";
}
elsif ( $line =~ /^alert/ )
{
        ( $split, $SNORT_ALERT_LOGS ) = split ( /=/, $line );
        print "Initializing Snort Alert Log Directory $SNORT_ALERT_LOGS\ . . . . . . . . . . \n";
}
elsif ( $line =~ /^snortlog/ )
{
        ( $split, $SNORT_LOG_LOGS ) = split ( /=/, $line );
        print "Initializing Snort Log Log Directory $SNORT_LOG_LOGS\ . . . . . . . . . . \n";
}
elsif ( $line =~ /^oos/ )
{
        ( $split, $SNORT_OOS_LOGS ) = split ( /=/, $line );
        print "Initializing Snort OOS Log Directory $SNORT_OOS_LOGS\ . . . . . . . . . . \n";
}
elsif ( $line =~ /^scan/ )
{
        ( $split, $SNORT_SCAN_LOGS ) = split ( /=/, $line );
        print "Initializing Snort Scan Log Directory $SNORT_SCAN_LOGS\ . . . . . . . . . . \n";
}
elsif ( $line =~ /^snortdumplogs/ )
{
        ( $split, $SNORT_DUMP_LOGS ) = split ( /=/, $line );
        print "Initializing Snort Dump Log Directory $SNORT_DUMP_LOGS\ . . . . . . . . . . \n";
}
elsif ( $line =~ /^iptraflogs/ )
{
        ( $split, $IPTRAF_LOGS ) = split ( /=/, $line );
        print "Initializing IPTraf Log Directory $IPTRAF_LOGS\ . . . . . . . . . . \n";
}
elsif ( $line =~ /^tcpdumplogs/ )
{
        ( $split, $TCPDUMP_LOGS ) = split ( /=/, $line );
        print "Initializing TCPdump Log Directory $TCPDUMP_LOGS\ . . . . . . . . . . \n";
}
elsif ( $line =~ /^windumplogs/ )
{
        ( $split, $WINDUMP_LOGS ) = split ( /=/, $line );
        print "Initializing Windump Log Directory $WINDUMP_LOGS\ . . . . . . . . . . \n";
}
else
{
        next;
}
```

```
            }
        }
        close ( $FH );
}

#########################################################################
# SUBROUTINE USAGE()
#########################################################################
sub usage()
{
        my( $Script ) = ( $0 =~ /([^\\\/]*?)$/ );
        my( $Line ) = "-" x length( $Script );
        print "Program Usage\n";

        print <<ENDOFUSAGE;

        $Script
        $Line

        Syntax:
        perl $Script [-F] [-T] [-c] [-d] [-f] [-h] [-i] [-l] [-m] [-n] [-o] [-p] [-s] [-t] [-u] [-w]

                -F .....  Computer to copy log files FROM: -F server2.
                -T .....  Computer to copy log files TO: -T workstation1.
                -a .....  Retrieve APACHE web logs.
                -c .....  Retrieve SNORT SCAN logs.
                -d .....  Retrieve TCPDUMP logs.
                -f .....  Retrieve FIREWALL logs.
                -h .....  Print this help screen.
                -i .....  Retrieve IIS web logs.
                -l .....  Retrieve SNORT ALERT logs.
                -m ..... Retrieve MAIL logs.
                -n .....  Retrieve SNORT DUMP.
                -o .....  Retrieve SNORT OOS logs.
                -p .....  Retrieve IPTraf logs.
                -s .....  Retrieve SYSTEM logs.
                -t .....  Retrieve FIREWALL stats.
                -u .....  Retrieve UCD-SNMP logs.
                -w ..... Retrieve WINDUMP logs.

        Examples:
        $Line
        $Script -F server2 -T workstation1 -f -m
        $Script -F server2 -a


ENDOFUSAGE

        print "Press [ENTER] to exit program.\n";
        <STDIN>;

        exit(0);
}
```


Appendix N: tcpdump Startup Script

```
#########################################################################
# tcpdump startup script
#########################################################################
HOSTNAME=`hostname`
PID_FILE="/var/run/tcpdump.pid"
TCPDUMP_RAW_FILE="/var/log/tcpdump/tcpdump.dmp"
TCPDUMP="/usr/sbin/tcpdump"
```

```
OPTIONS="-i eth1 -n -vv -xX -s 1500"
LOGGER="/bin/logger"

echo "STARTING $SCRIPT_NAME!"
$LOGGER -p warning "STARTING $SCRIPT_NAME!"

test -x $TCPDUMP || exit 0

case "$1" in
    start)
      echo "Starting Packet Capture with TCPDUMP"
      $TCPDUMP $OPTIONS -w $TCPDUMP_RAW_FILE &
      ps -ef | $GREP tcpdump | $AWK '{print $2}' > $PID_FILE

      if [ "`pidof $TCPDUMP`" ];
      then
            $ECHO "TCPDUMP is up and running!"
      else
            $ECHO "TCPDUMP failed to start: EXITING!"
            exit 0
      fi
      echo -n "."
      ;;

    stop)
      $ECHO "Stopping Packet Capture with TCPDUMP"
      if [ "`pidof $TCPDUMP`" ] ;
      then
         kill -TERM `pidof $TCPDUMP`
         count=120
         numdots=0
         while ([ $count != 0 ]) do
            let count=$count-1
            if [ "`pidof $TCPDUMP`" ] ;
            then
               $ECHO -n .
               let numdots=$numdots+1
               sleep 1
            else
               count=0
            fi
         done
      else
         $ECHO "TCPDUMP is  not running!"
      fi
      ;;
    restart)
      $0 stop
      $0 start
      ;;
    *)
      $ECHO 'Usage: /etc/init.d/tcpdump {start|stop|restart}'
      exit 1
      ;;
esac
exit 0
;;
```

Appendix O: twpol.txt Configuration File

```
###########################################################################
# twpol.txt
# tripwire policy file
###########################################################################
################################################
# POLICY
################################################
#####################
# SECTION GLOBAL
#####################
@@section GLOBAL
TWROOT="/etc";
TWBIN="/usr/sbin";
TWPOL="/etc/tripwire";
TWCONFIG="/etc/tripwire";
TWDB="/var/lib/tripwire";
TWSKEY="/etc/tripwire";
TWLKEY="/etc/tripwire";
TWREPORT="/var/lib/tripwire/report";
HOSTNAME=`hostname`;

#####################
# SECTION FILESYSTEM
#####################
@@section FS
SEC_CRIT        = $(IgnoreNone)-SHa;     # Critical files - we can't afford to miss any changes.
SEC_SUID        = $(IgnoreNone)-SHa;     # Binaries with the SUID or SGID flags set.
SEC_TCB         = $(ReadOnly);           # Members of the Trusted Computing Base.
SEC_BIN         = $(ReadOnly);           # Binaries that shouldn't change
SEC_CONFIG      = $(Dynamic);            # Config files that are changed infrequently but accessed often.
SEC_LOG         = $(Growing);            # Files that grow, but that should never change ownership.
SEC_INVARIANT   = +pug;                  # Directories that should never change permission or ownership.
SIG_LOW         = 33;                    # Non-critical files that are of minimal security impact
SIG_MED         = 66;                    # Non-critical files that are of significant security impact
SIG_HI          = 100;                   # Critical files that are significant points of vulnerability

#####################
# TRIPWIRE BINARIES
#####################
(
    rulename = "Tripwire Binaries",
    severity = $(SIG_HI),
    emailto = administrator@network.com
)
{
    $(TWBIN)/siggen      -> $(ReadOnly);
    $(TWBIN)/tripwire    -> $(ReadOnly);
    $(TWBIN)/twadmin     -> $(ReadOnly);
    $(TWBIN)/twprint     -> $(ReadOnly);
}

#####################
# TRIPWIRE DATA FILES
#####################
(
    rulename = "Tripwire Data Files",
    severity = $(SIG_HI),
    emailto = administrator@network.com
)
{
```

```
        $(TWDB)                                 -> $(Dynamic) -i;
        $(TWPOL)/tw.pol                         -> $(SEC_BIN) -i;
        $(TWPOL)/tw.cfg                         -> $(SEC_BIN) -i;
        # $(TWLKEY)/$(HOSTNAME)-local.key -> $(SEC_BIN);
        $(TWSKEY)/site.key                      -> $(SEC_BIN);
        $(TWREPORT)                             -> $(Dynamic) (recurse=0);
}

######################
# SYSTEM INFORMATION BINARIES
######################
(
        rulename = "System Information Binaries",
        severity = $(SIG_HI),
        emailto = administrator@network.com
)
{
        /sbin/kernelversion             -> $(SEC_CRIT);
        /sbin/runlevel                  -> $(SEC_CRIT);
}

######################
# SECURITY CONTROL
######################
(
        rulename = "Security Control",
        severity = $(SIG_HI),
        emailto = administrator@network.com
)
{
        /etc/group                      -> $(SEC_CRIT);
        /etc/security/                  -> $(SEC_CRIT);
        /etc/securetty/                 -> $(SEC_CRIT);
        /var/spool/cron                 -> $(SEC_CRIT);
        /var/spool/cron/tabs            -> $(SEC_CRIT);
}

######################
# BOOT SCRIPTS
######################
(
        rulename = "Boot Scripts",
        severity = $(SIG_HI),
        emailto = administrator@network.com
)
{
        /etc/init.d                     -> $(SEC_CONFIG);
}

######################
# LOGIN SCRIPTS
######################
(
        rulename = "Login Scripts",
        severity = $(SIG_HI),
        emailto = administrator@network.com
)
{
        /etc/csh.cshrc                  -> $(SEC_CONFIG);
        /etc/csh.login                  -> $(SEC_CONFIG);
        # /etc/tsh_profile              -> $(SEC_CONFIG);
        /etc/profile                    -> $(SEC_CONFIG);
        /etc//skel                      -> $(SEC_CONFIG) (recurse=0);
}

######################
# CRITICAL SYSTEM BOOT FILES
```

```
#######################
(
 rulename = "Critical System Boot Files",
 severity = $(SIG_HI),
 emailto = administrator@network.com
)
{
      /boot                         -> $(SEC_CRIT);
      /sbin/lilo                    -> $(SEC_CRIT);
      !/boot/System.map;
      !/boot/module-info;
      !/boot/.config;
}

#######################
# ROOT ACCOUNT
#######################
(
      rulename = "Root Account Files",
      severity = 100,
      emailto = administrator@network.com
)
{
      /root                         -> $(SEC_CRIT);
      /root/Desktop                 -> $(SEC_CRIT);
      /root/bin                     -> $(SEC_CRIT);
      /root/.bash_history           -> $(SEC_LOG);
      /root/.forward                -> $(SEC_LOG);
      /root/.ssh                    -> $(SEC_LOG);
      /root/.xsession-errors        -> $(SEC_CONFIG);
      # /root/.gnome_private        -> $(SEC_CONFIG);
      # /root/.gnome-desktop        -> $(SEC_CONFIG);
      # /root/.gnome                -> $(SEC_CONFIG);
      /root/.Xauthority             -> $(SEC_CONFIG) -i;
      /root/.ICEauthority           -> $(SEC_CONFIG);
      /root/.exrc                   -> $(SEC_CONFIG);
}

#######################
# CRITICAL DEVICES
#######################
(
      rulename = "Critical Devices",
      severity = $(SIG_HI),
      recurse = false,
      emailto = administrator@network.com
)
{
      /dev/kmem                     -> $(Device);
      /dev/mem                      -> $(Device);
      /dev/null                     -> $(Device);
      /dev/zero                     -> $(Device);
      /proc/devices                 -> $(Device);
      /proc/net                     -> $(Device);
      /proc/sys                     -> $(Device);
      /proc/cpuinfo                 -> $(Device);
      /proc/modules                 -> $(Device);
      /proc/mounts                  -> $(Device);
      /proc/dma                     -> $(Device);
      /proc/filesystems             -> $(Device);
      /proc/pci                     -> $(Device);
      /proc/interrupts              -> $(Device);
      /proc/ioports                 -> $(Device);
      /proc/scsi                    -> $(Device);
      /proc/kcore                   -> $(Device);
      /proc/self                    -> $(Device);
      /proc/kmsg                    -> $(Device);
```

```
        /proc/stat                      -> $(Device);
        /proc/ksyms                     -> $(Device);
        /proc/loadavg                   -> $(Device);
        /proc/uptime                    -> $(Device);
        /proc/locks                     -> $(Device);
        /proc/version                   -> $(Device);
        /proc/mdstat                    -> $(Device);
        /proc/meminfo                   -> $(Device);
        /proc/cmdline                   -> $(Device);
        /proc/misc                      -> $(Device);
}

######################
# CONFIGURATION FILES
######################
(
        rulename = "Configuration Files",
        severity = $(SIG_HI),
        recurse = false,
        emailto = administrator@network.com
)
{
        /                               -> $(SEC_CONFIG) (recurse = 0);
        /etc                            -> $(SEC_CONFIG) (recurse = 0);
        /etc/hosts                      -> $(SEC_CONFIG);
        /etc/inetd.conf                 -> $(SEC_CONFIG);
        /etc/inittab                    -> $(SEC_CONFIG);
        /etc/resolv.conf                -> $(SEC_CONFIG);
        /etc/syslog.conf                -> $(SEC_CONFIG);
}

######################
# CRITICAL CONFIGURATION FILES
######################
(
        rulename = "Critical Configuration Files",
        severity = $(SIG_HI),
        emailto = administrator@network.com
)
{
        /etc/crontab                    -> $(SEC_BIN);
        /etc/cron.hourly                -> $(SEC_BIN);
        /etc/cron.daily                 -> $(SEC_BIN);
        /etc/cron.weekly                -> $(SEC_BIN);
        /etc/cron.monthly               -> $(SEC_BIN);
        /etc/default                    -> $(SEC_BIN);
        /etc/fstab                      -> $(SEC_BIN);
        /etc/exports                    -> $(SEC_BIN);
        /etc/group-                     -> $(SEC_BIN);
        /etc/host.conf                  -> $(SEC_BIN);
        /etc/hosts.allow                -> $(SEC_BIN);
        /etc/hosts.deny                 -> $(SEC_BIN);
        /etc/protocols                  -> $(SEC_BIN);
        /etc/services                   -> $(SEC_BIN);
        /etc/motd                       -> $(SEC_BIN);
        /etc/passwd                     -> $(SEC_CONFIG);
        /etc/passwd-                    -> $(SEC_CONFIG);
        /etc/profile.d                  -> $(SEC_BIN);
        # /etc/rpc                      -> $(SEC_BIN);
        /etc/sysconfig                  -> $(SEC_BIN);
        /etc/nsswitch.conf              -> $(SEC_BIN);
}
```

Appendix P: twinstall.sh File

```
#!/bin/sh
######################################################################
# twinstall.sh
# tripwire installation script
######################################################################
# declare a trusted path
PATH=/sbin:/usr/sbin:/usr/local/sbin:/bin:/usr/bin:/usr/local/bin
export PATH

# get hostname
HOSTNAME=`hostname`

# set script variable name
SCRIPT_NAME="twinstall.sh"

# tripwire directories
TW_DIR="/etc/tripwire"
TWDB_DIR="/var/lib/tripwire"
TWREPORT_DIR="/var/lib/tripwire/report"
SITE_KEY="$TW_DIR/site.key"
LOCAL_KEY="$TW_DIR/$HOSTNAME-local.key"

######################################################################
# EXECUTABLES
######################################################################
TW="/usr/sbin/tripwire"
TWADMIN="/usr/sbin/twadmin"
TWPRINT="/usr/sbin/twprint"
LOGGER="/bin/logger"
DATE="/bin/date"
MAIL="/bin/mail"
KILL="/bin/kill"
LSOF="/usr/bin/lsof"
FIND="/usr/bin/find"
WHO="/usr/bin/who"
OPTIONS=""

######################################################################
# BEGIN SCRIPT
######################################################################
$ECHO "Running $SCRIPT_NAME . . . . . "
$LOGGER -p warning "Running $SCRIPT_NAME . . . . . "

#################################
# GENERATE KEY FILES
#################################
$TWADMIN --generate-keys --site-keyfile $SITE_KEY
$TWADMIN --generate-keys --local-keyfile $LOCAL_KEY

#################################
# SIGN CONFIGURATION & POLICY FILES
#################################
$TWADMIN --create-cfgfile --cfgfile $TW_DIR/tw.cfg --site-keyfile $SITE_KEY $TW_DIR/twcfg.txt
$TWADMIN --create-polfile --cfgfile $TW_DIR/tw.cfg --site-keyfile $SITE_KEY $TW_DIR/twpol.txt

#################################
# SET PERMISSIONS
#################################
cd $TW_DIR
$CHOWN root:root $SITE_KEY $LOCAL_KEY tw.cfg tw.pol
$CHMOD 600 $SITE_KEY $LOCAL_KEY tw.cfg tw.pol
```

```
################################
# BUILD DATABASE & SIGN WITH LOCAL KEY
################################
$TW --init

################################
# REMOVE UNENCRYPTED TEXT FILES
################################
rm twcfg.txt twpol.txt

$ECHO "$SCRIPT_NAME has run successfully . . . . . "
$LOGGER -p warning "$SCRIPT_NAME has run successfully . . . . . "

#########################################################################
# MAIL REPORT
#########################################################################
exit 0
```

Appendix Q: smart_report.sh

```
#!/bin/sh
#########################################################################
# smart_report.sh
# SMART report generated by smartmontools
#########################################################################
SMART="/usr/sbin/smartctl"

# devices
ATA_DEVICE1="/dev/hda"
ATA_DEVICE2="/dev/hdb"
ATA_DEVICE3="/dev/hdc"
SCSI_DEVICE=""
ALL_DEVICES="1"

SMART_REPORT="/var/log/report/smart_report.txt"
MAILTO="administrator@network.com"
MAILFROM="root-security@network.com"
SUBJECT="Smart Report $DATE"
ATTACHMENT="$SMART_REPORT"
EMAIL_REPORT="1"

#########################################################################
# SMART REPORT
#########################################################################
echo $MESSAGE >> $SMART_REPORT
$DATE >> $SMART_REPORT
echo $POUND >> $SMART_REPORT

if [ -e $ATA_DEVICE1 ];
then
      echo "Report for $ATA_DEVICE1"  >> $SMART_REPORT
      $SMART -t long $ATA_DEVICE1 >> $SMART_REPORT
      $SMART -i $ATA_DEVICE1 >> $SMART_REPORT
      $SMART -Hc $ATA_DEVICE1 >> $SMART_REPORT
      $SMART -l error $ATA_DEVICE1 >> $SMART_REPORT
      $SMART -l selftest $ATA_DEVICE1 >> $SMART_REPORT
      echo "ALL SMART INFORMATION for $ATA_DEVICE1"  >> $SMART_REPORT
      $SMART -a $ATA_DEVICE1 >> $SMART_REPORT
else
      echo "NO DEVICE $ATA_DEVICE1 FOUND"  >> $SMART_REPORT
fi

if [ "$ALL_DEVICES" = "1" ] && [ "$ATA_DEVICE2" != "" ];
```

```
then
      echo "Report for $ATA_DEVICE2"  >> $SMART_REPORT
      $SMART -t long $ATA_DEVICE2 >> $SMART_REPORT
      $SMART -i $ATA_DEVICE2 >> $SMART_REPORT
      $SMART -Hc $ATA_DEVICE2 >> $SMART_REPORT
      $SMART -l error $ATA_DEVICE2 >> $SMART_REPORT
      $SMART -l selftest $ATA_DEVICE2 >> $SMART_REPORT
      echo "ALL SMART INFORMATION for $ATA_DEVICE2"  >> $SMART_REPORT
      $SMART -a $ATA_DEVICE2 >> $SMART_REPORT
else
      echo "NO DEVICE $ATA_DEVICE2 FOUND"  >> $SMART_REPORT
fi

if [ "$ALL_DEVICES" = "1" ] && [ "$ATA_DEVICE3" != "" ];
then
      echo "Report for $ATA_DEVICE3"  >> $SMART_REPORT
      $SMART -t long $ATA_DEVICE3 >> $SMART_REPORT
      $SMART -i $ATA_DEVICE3 >> $SMART_REPORT
      $SMART -Hc $ATA_DEVICE3 >> $SMART_REPORT
      $SMART -l error $ATA_DEVICE3 >> $SMART_REPORT
      $SMART -l selftest $ATA_DEVICE3 >> $SMART_REPORT
      echo "ALL SMART INFORMATION for $ATA_DEVICE3"  >> $SMART_REPORT
      $SMART -a $ATA_DEVICE3 >> $SMART_REPORT
else
      echo "NO DEVICE $ATA_DEVICE3 FOUND"  >> $SMART_REPORT
fi

if [ "$ALL_DEVICES" = "1" ] && [ "$SCSI_DEVICE" != "" ];
then
      echo "Report for $SCSI_DEVICE"  >> $SMART_REPORT
      $SMART -t long $SCSI_DEVICE >> $SMART_REPORT
      $SMART -i $SCSI_DEVICE >> $SMART_REPORT
      $SMART -Hc $SCSI_DEVICE >> $SMART_REPORT
      $SMART -l error $SCSI_DEVICE >> $SMART_REPORT
      $SMART -l selftest $SCSI_DEVICE >> $SMART_REPORT
      echo "ALL SMART INFORMATION for $SCSI_DEVICE"  >> $SMART_REPORT
      $SMART -a $SCSI_DEVICE >> $SMART_REPORT
else
      echo "NO DEVICE $SCSI_DEVICE FOUND"  >> $SMART_REPORT
      echo $POUND >> $SMART_REPORT
fi

######################################################################
# MAIL REPORT
######################################################################
if [ "$EMAIL_REPORT" = "1" ];
then
      echo "Mailing $SMART_REPORT . . . . . "
      $CAT $SMART_REPORT | $MAIL -s "$SUBJECT" $MAILTO
fi
exit 0
```

Appendix R: forensic_report.sh

```
#!/bin/sh
######################################################################
# forensic_report.sh
# Runs a forensic check on system and generates a report.
# Requires Rootkit Hunter 1.6 be installed on system
######################################################################
######################################################################
# SECURE ENVIRONMENT
######################################################################
```

```
# declare a trusted path and shell
PATH=/sbin:/usr/sbin:/usr/local/sbin:/bin:/usr/bin:/usr/local/bin
SHELL=/bin/sh
export PATH
export SHELL

########################################################################
# INCLUDE FILES
########################################################################
source /root/bin/utility.functions.sh

######################
# VERSION_INFO
######################
VERSION="1.0.1"
VDATE="03/24/04"

########################################################################
# VARIABLES
########################################################################
SERVER_NAME=`uname -n`
DATE=`date '+%b %d'`
HOSTNAME=`hostname`
SCRIPT_NAME="forensic_report.sh"

MESSAGE="Starting Forensic Check on $HOSTNAME"
MESSAGE_DATE="DATE: "
REPORT_TITLE="FORENSIC REPORT $HOSTNAME"
REPORT_HEADING=""
REPORT="/var/log/report/forensic_report_$HOSTNAME.txt"

MAILTO="administrator@bandwidthco.com"
MAILFROM="root-security@bandwidthco.com"
SUBJECT="Security Forensics Report $DATE"
ATTACHMENT="$REPORT"
EMAIL_REPORT="1"

MALWARE_CHECK="1"
PROCESS_CHECK="1"
LOGINS_CHECK="1"
NEVER_LOGGED_ON="1"
RUN_JOHN="1"
CHECK_BANNERS="1"
TEST_PERMISSIONS="1"
CHECK_FTPUSERS="1"
RHOSTS_CHECK="1"
SUID_GUID_CHECK="1"
INETD_CHECK="1"
WORLD_DIRS="1"
SNIFFER_CHECK="0"
BOOT_FILE_CHECK="1"
PORT_BINDING_CHECK="1"
ACTIVITY_CHECK="1"
RUN_HISTORY="0"
DEVICE_FILES="0"
RPM_FILES="0"
RUN_LSOF="0"
POUND="##############################################################################"

########################################################################
# EXECUTABLES
########################################################################
LOGGER="/bin/logger"
DF="/bin/df"
DU="/usr/bin/du"
GREP="/bin/grep"
AWK="/bin/awk"
```

```
SED="/bin/sed"
EXPECT="/usr/bin/expect"
ECHO="/bin/echo"
CAT="/bin/cat"
LSMOD="/sbin/lsmod"
DATE="/bin/date"
MAIL="/bin/mail"
KILL="/bin/kill"
LSOF="/usr/bin/lsof"
LDD="/usr/bin/ldd"
FILE="/usr/bin/file"
STRINGS="/usr/bin/strings"
SORT="/usr/bin/sort"
SLEEP="/bin/sleep"
NETSTAT="/bin/netstat"
IFCONFIG="/sbin/ifconfig"
PS="/bin/ps"
FIND="/usr/bin/find"
WHO="/usr/bin/who"
LAST="/usr/bin/last"
LASTB="/usr/bin/lastb"
CHOWN="/bin/chown"
CHMOD="/bin/chmod"
MAIL="/bin/mail"
SENDMAIL="/usr/sbin/sendmail"
CHKCONFIG="/sbin/chkconfig"
TAIL="/usr/bin/tail"
RPM="/bin/rpm"
FIND="/usr/bin/find"
RM="/bin/rm"
WHO="/usr/bin/who"
RKHUNTER="/usr/local/bin/rkhunter"
OPTIONS=""

#########################################################################
# BEGIN SCRIPT
#########################################################################
$ECHO "Running $SCRIPT_NAME . . . . . "
$LOGGER -p warning "Running $SCRIPT_NAME . . . . . "

#########################
# CONFIRM ROOT & VERSION
#########################
confirm_root
version_info "$VERSION" "$VDATE"

$ECHO $POUND > $REPORT
$ECHO $REPORT_TITLE >> $REPORT
$DATE >> $REPORT
$ECHO $POUND >> $REPORT
$ECHO  >> $REPORT

#########################
# MALWARE CHECK
#########################
if [ "$MALWARE_CHECK" = "1" ];
then
      $ECHO "Malware Check"
      REPORT_HEADING="RUNNING MALWARE CHECK"
      OPTIONS="-c --createlogfile --skip-keypress"
      RKHUNTER_LOG="/var/log/rkhunter.log"
      $ECHO $POUND >> $REPORT
      $ECHO $REPORT_HEADING >> $REPORT
      $ECHO $POUND >> $REPORT
      $RKHUNTER $OPTIONS
      $CAT $RKHUNTER_LOG >> $REPORT
      $ECHO $POUND >> $REPORT
```

```
        $ECHO >> $REPORT
fi

##########################
# RUNNING PROCESSES CHECK
##########################
if [ "$PROCESS_CHECK" = "1" ];
        then
        $ECHO "Processes Check"
        REPORT_HEADING="RUNNING PROCESSES CHECK"
        OPTIONS="-e"
        OPTIONS="-elf"
        OPTIONS="-el"
        $ECHO $POUND >> $REPORT
        $ECHO $REPORT_HEADING >> $REPORT
        $ECHO $POUND >> $REPORT
        $PS $OPTIONS >> $REPORT
        $ECHO $POUND >> $REPORT
        $ECHO >> $REPORT
fi

##########################
# LOGINS CHECK
##########################
if [ "$LOGINS_CHECK" = "1" ];
then
        ##########################
        # UTMP CHECK
        ##########################
        $ECHO "UTMP Check"
        REPORT_HEADING="UTMP CHECK"
        OPTIONS="-a"
        $ECHO $POUND >> $REPORT
        $ECHO $REPORT_HEADING >> $REPORT
        $ECHO $POUND >> $REPORT
        $WHO $OPTIONS >> $REPORT
        $ECHO $POUND >> $REPORT
        $ECHO >> $REPORT

        ##########################
        # WTMP CHECK
        ##########################
        $ECHO "WTMP Check"
        REPORT_HEADING="WTMP CHECK"
        $ECHO $POUND >> $REPORT
        $ECHO $REPORT_HEADING >> $REPORT
        $ECHO $POUND >> $REPORT
        $LAST >> $REPORT
        $ECHO $POUND >> $REPORT
        $ECHO >> $REPORT

        ##########################
        # BTMP CHECK
        ##########################
        $ECHO "BTMP Check"
        REPORT_HEADING="BTMP CHECK"
        OPTIONS="-adx"
        $ECHO $POUND >> $REPORT
        $ECHO $REPORT_HEADING >> $REPORT
        $ECHO $POUND >> $REPORT
        MESSAGE="Apparently Not Enabled in Linux"
        $ECHO $MESSAGE >> $REPORT
        $ECHO $POUND >> $REPORT
        $ECHO >> $REPORT
fi

##########################
```

```
# NEVER LOGGED ON
##########################
if [ "$NEVER_LOGGED_ON" = "1" ];
then
      $ECHO "Accounts Never Logged On"
      REPORT_HEADING="ACCOUNTS NEVER LOGGED ON"
      OPTIONS="-e"
      OPTIONS="-elf"
      OPTIONS="-el"
      $ECHO $POUND >> $REPORT
      $ECHO $REPORT_HEADING >> $REPORT
      $ECHO $POUND >> $REPORT

      for i in `lastlog 2> /dev/null | awk '/Never logged in/ {print$1}'`;
      do
         true = 0
         SHELL=`grep "^$i:" /etc/passwd | awk -F: '{print$7}'`
         test -z "$SHELL" && SHELL="/bin/sh"
         grep -q "^$SHELL" /etc/shells && export true=1
         test "$true" = 1 && {
            test -r /etc/shadow && {
               true=`awk -F:  "/^$i:/ "'{
                  if (length($2) > 12)
                     printf("2\n");
                  if ($2 == "")
                     printf("3\n");
               }' /etc/shadow`
               test "$true" = 2 && echo "Warning: user $i has a password and a valid shell but never logged in." >> $REPORT
               test "$true" = 3 && echo "Warning: user $i has got NO password and a valid shell." >> $REPORT
            }
            test -r /etc/shadow || {
                  echo "Warning: user $i never logged in." >> $REPORT
            }
         }
      done

      $ECHO $POUND >> $REPORT
      $ECHO >> $REPORT
fi

##########################
# SUID & SGID FILE CHECK
##########################
if [ "$SUID_GUID_CHECK" = "1" ];
then
      $ECHO "STANDARD SUID/SGID Check"
      REPORT_HEADING="STANDARD SUID & SGID FILE CHECK"
      $ECHO $POUND >> $REPORT
      $ECHO $REPORT_HEADING >> $REPORT
      $ECHO $POUND >> $REPORT
      $FIND / -type f \( -perm -004000 -o -perm -002000 \) >> $REPORT
      $ECHO $POUND >> $REPORT
      $ECHO >> $REPORT

      $ECHO "MODIFIED SUID/SGID Check"
      REPORT_HEADING="MODIFIED (Last 7 Days) SUID & SGID FILE CHECK"
      $ECHO $POUND >> $REPORT
      $ECHO $REPORT_HEADING >> $REPORT
      $ECHO $POUND >> $REPORT

      # how far back (in days) to check for modified cmds
      mtime="7"
      # by default, let's be quiet about things
      verbose=0

      if [ "$1" = "-v" ] ;
      then
```

```
        verbose=1
    fi

    for match in $(find /bin /usr/bin -type f -perm +4000 -print)
    do
        if [ -x $match ] ;
        then
            owner="$(ls -ld $match | awk '{print $3}')"
            perms="$(ls -ld $match | cut -c5-10 | grep 'w')"

            if [ ! -z $perms ] ;
            then
                echo "**** $match (writeable and setuid $owner)" >> $REPORT
            elif [ ! -z $(find $match -mtime -$mtime -print) ] ;
            then
                echo "**** $match (modified within $mtime days and setuid $owner)" >> $REPORT
            elif [ $verbose -eq 1 ] ;
            then
                lastmod="$(ls -ld $match | awk '{print $6, $7, $8}')"
                echo "     $match (setuid $owner, last modified $lastmod)" >> $REPORT
            fi
        fi
    done

    $ECHO $POUND >> $REPORT
    $ECHO >> $REPORT
fi

################################
# JOHN
################################
if [ "$RUN_JOHN" = "1" ];
then
    SEC_VAR="/var/lib/secchk"
    SEC_BIN="/usr/lib/secchk"
    $ECHO "Running John the Ripper"
    REPORT_HEADING="JOHN THE RIPPER"
    $ECHO $POUND >> $REPORT
    $ECHO "$REPORT_HEADING" >> $REPORT
    $ECHO $POUND >> $REPORT

    TMPDIR=`/bin/mktemp -d /tmp/.security.XXXXXX` ||
    {
        TMPDIR="/tmp/.security-weekly.sh.redhatshouldupdatetheirmktemp.$$"
        rm -rf "$TMPDIR"
        mkdir "$TMPDIR" || exit 1
    }

    trap 'rm -rf $TMPDIR; exit 1' 0 1 2 3 13 15
    OUT="$TMPDIR/security.out"
    TMP1="$TMPDIR/security.tmp1"
    TMP2="$TMPDIR/security.tmp2"

    if [ -x /usr/sbin/john -a -x /usr/sbin/unshadow ];
    then
        echo > $SEC_VAR/dict
        cat /usr/dict/* /var/lib/john/password.lst 2> /dev/null | sort | uniq >> $SEC_VAR/dict

        # Copy passwd file. Use unique name to avoid races when john takes very long
        SEC_PASSWD=$SEC_VAR/passwd.$$
        unshadow /etc/passwd /etc/shadow > $SEC_PASSWD
        nice -n 1 john -single "$SEC_PASSWD" 1> /dev/null 2>&1
        nice -n 1 john -rules -w:$SEC_VAR/dict "$SEC_PASSWD" 1> /dev/null 2>&1
        john -show "$SEC_PASSWD" | sed -n 's/:.*//p' > "$OUT"

        if [ -s "$OUT" ] ;
        then
```

```
                for i in `cat "$OUT"`;
                do
                    $MAILER "$i" << _EOF_
                    Subject: Please change your Password

Your password for account "$i" is insecure.
Please change it as soon as possible.

Yours,
    Password Checking Robot

_EOF_
                done

                printf "\nThe following user accounts have guessable passwords:\n" >> $REPORT
                cat "$OUT" >> $REPORT
            fi
    else
        echo -e "\nPassword security checking not possible, package "john" not installed." >> $REPORT
    fi
    rm -f $SEC_PASSWD

    $ECHO $POUND >> $REPORT
    $ECHO >> $REPORT

fi

################################
# CHECK BANNERS
################################
if [ "$CHECK_BANNERS" = "1" ];
then
    $ECHO "Banners Check"
    REPORT_HEADING="BANNERS CHECK"
    $ECHO $POUND >> $REPORT
    $ECHO "$REPORT_HEADING" >> $REPORT
    $ECHO $POUND >> $REPORT

    ISSUE_TEXT="/root/bin/issue.txt"
    MOTD_TEXT="/root/bin/motd.txt"
    ISSUE_FILE="/etc/issue"
    MOTD_FILE="/etc/motd"

    ###################################
    # SET BANNERS
    ###################################
    $CAT $ISSUE_TEXT > $ISSUE_FILE
    $CAT $MOTD_TEXT > $MOTD_FILE

    ###################################
    # VIEW BANNERS
    ###################################
    $ECHO "$POUND" >> $REPORT
    $ECHO "$ISSUE_FILE" >> $REPORT
    $ECHO "$POUND" >> $REPORT
    $CAT $ISSUE_FILE >> $REPORT
    $ECHO "$POUND" >> $REPORT
    $ECHO "$MOTD_FILE" >> $REPORT
    $ECHO "$POUND" >> $REPORT
    $CAT $MOTD_FILE >> $REPORT
    $ECHO $POUND >> $REPORT
    $ECHO >> $REPORT
fi

################################
# TEST PERMISSIONS
################################
```

```
if [ "$TEST_PERMISSIONS" = "1" ];
then
        $ECHO "Permissions TEST"
        REPORT_HEADING="PERMISSIONS TEST"
        $ECHO $POUND >> $REPORT
        $ECHO "$REPORT_HEADING" >> $REPORT
        $ECHO $POUND >> $REPORT

        CHKSTAT="/usr/bin/chkstat"
        TEST_FILE="/etc/permissions-test"
        LOCAL="/etc/permissions.local"
        BEFORE=""
        CHANGE=""
        AFTER=""

        touch $TEST_FILE
        $CHOWN root:root $TEST_FILE
        $CHMOD 700 $TEST_FILE
        BEFORE=`ls -la $TEST_FILE`
        $CHOWN bin:bin $TEST_FILE
        $CHMOD 777 $TEST_FILE
        CHANGE=`ls -la $TEST_FILE`
        $CHKSTAT --set $LOCAL
        AFTER=`ls -la $TEST_FILE`

        $ECHO "$POUND" >> $REPORT
        $ECHO "BEFORE TEST: $TEST_FILE" >> $REPORT
        $ECHO "$POUND" >> $REPORT
        $ECHO "$BEFORE" >> $REPORT
        $ECHO "$POUND" >> $REPORT
        $ECHO "CHANGE TEST: $TEST_FILE" >> $REPORT
        $ECHO "$POUND" >> $REPORT
        $ECHO "$CHANGE" >> $REPORT
        $ECHO "$POUND" >> $REPORT
        $ECHO "AFTER TEST: $TEST_FILE" >> $REPORT
        $ECHO "$POUND" >> $REPORT
        $ECHO "$AFTER" >> $REPORT
        $ECHO $POUND >> $REPORT
        $ECHO >> $REPORT
fi

################################
# CHECK FTPUSERS
################################
if [ "$CHECK_FTPUSERS" = "1" ];
then
        $ECHO "FTPUSERS Check"
        REPORT_HEADING="FTPUSERS CHECK"
        $ECHO $POUND >> $REPORT
        $ECHO "$REPORT_HEADING" >> $REPORT
        $ECHO $POUND >> $REPORT

        FTPUSERS="/etc/ftpusers"

        $ECHO "Users in ftpuser file are:" >> $REPORT
        $ECHO  >> $REPORT
        $CAT $FTPUSERS  >> $REPORT
        $ECHO $POUND >> $REPORT
        $ECHO >> $REPORT

fi

################################
# RHOSTS CHECK
################################
if [ "$RHOSTS_CHECK" = "1" ];
then
```

```
        $ECHO "FTPUSERS Check"
        REPORT_HEADING="FTPUSERS CHECK"
        $ECHO $POUND >> $REPORT
        $ECHO "$REPORT_HEADING" >> $REPORT
        $ECHO $POUND >> $REPORT

        OUT="/var/log/report/tempfile"
        list="/etc/hosts.equiv /etc/shosts.equiv /etc/hosts.lpd"
        for f in $list ; do
            if [ -s "$f" ] ;
            then
                awk '{
                    if ($0 ~ /^\+@.*$/)
                        next;
                    if ($0 ~ /^\+.*$/)
                        printf("\nPlus sign in the file %s\n", FILENAME) >> $REPORT;
                }' $f
            fi
        done

        awk -F: '{ print $1 " " $6 }' /etc/passwd |
        while read uid homedir;
        do
            for j in .rhosts .shosts;
            do
                if [ -s ${homedir}/$j ] ; then
                    rhost=`ls -lcdbg ${homedir}/$j|sed 's/[%\]/_/g'`
                    printf "$uid: $rhost\n" >> $REPORT
                    test -f "$j" && {
                       if egrep \\+ ${homedir}/$j > /dev/null ;
                       then
                          printf "\t(has got a plus (+) sign!)\n" >> $REPORT
                       fi
                    }
                fi
            done
        done > $OUT
        if [ -s "$OUT" ] ;
        then
            printf "\nChecking for users with .rhosts/.shosts files.\n" >> $REPORT
            cat "$OUT" >> $REPORT
        fi
        $ECHO $POUND >> $REPORT
        $ECHO >> $REPORT
    fi

    ################################
    # INETD - XINETD
    ################################
    # show what services are enabled with inetd and xinetd
    # and if they're available on the system.
    ################################
    if [ "$INETD_CHECK" = "1" ];
    then
        $ECHO "INETD-XINETD Check"
        REPORT_HEADING="INETD-XINETD CHECK"
        $ECHO $POUND >> $REPORT
        $ECHO "$REPORT_HEADING" >> $REPORT
        $ECHO $POUND >> $REPORT
        iconf="/etc/inetd.conf"
        xconf="/etc/xinetd.conf"
        xdir="/etc/xinetd.d"
        OPTIONS="-el"

        if [ -r $iconf ];
        then
            echo "Services enabled in $iconf are:" >> $REPORT
```

```
        grep -v '^#' $iconf | awk '{print "  " $1}' >> $REPORT
        echo "" >> $REPORT

        if [ "$($PS $OPTIONS | grep inetd | egrep -vE '(xinet|grep)')" = "" ];
        then
            echo "** warning: inetd does not appear to be running" >> $REPORT
        fi
    fi

    if [ -r $xconf ];
    then
        # don't need to look in xinietd.conf, just know it exists
        echo "Services enabled in $xdir are:" >> $REPORT

        for service in $xdir/*
        do
            if ! $(grep disable $service | grep 'yes' > /dev/null) ;
            then
                echo -n "  " >> $REPORT
                basename $service >> $REPORT
            fi
        done

        if ! $($PS $OPTIONS | grep xinetd | grep -v 'grep' > /dev/null) ;
        then
            echo "** warning: xinetd does not appear to be running" >> $REPORT
        fi
    fi

    $ECHO $POUND >> $REPORT
    $ECHO >> $REPORT
fi

################################
# FIND WORLD WRITABLE DIRECTORIES
################################
if [ "$WORLD_DIRS" = "1" ];
    then
    $ECHO "World Writable Directories Check"
    REPORT_HEADING="WORLD WRITABLE DIRECTORIES"
    $ECHO $POUND >> $REPORT
    $ECHO $REPORT_HEADING >> $REPORT
    $ECHO $POUND >> $REPORT
    $FIND / -xdev -perm +o=w ! \( -type d -perm +o=t \) ! -type l -print >> $REPORT
    $ECHO $POUND >> $REPORT
fi

#########################
# NIC & SNIFFER CHECK
#########################
if [ "$SNIFFER_CHECK" = "1" ];
then
    $ECHO "NIC & Sniffer Check"
    REPORT_HEADING="NIC & SNIFFER CHECK"
    OPTIONS="-a"
    $ECHO $POUND >> $REPORT
    $ECHO $REPORT_HEADING >> $REPORT
    $ECHO $POUND >> $REPORT
    $IFCONFIG $OPTIONS >> $REPORT
    $ECHO $POUND >> $REPORT
    $ECHO >> $REPORT
fi

#########################
# BOOT FILE CHECK
#########################
if [ "$BOOT_FILE_CHECK" = "1" ];
```

```
then
      $ECHO "Boot File Check"
      REPORT_HEADING="BOOT FILE CHECK (CHKCONFIG)"
      OPTIONS="--list"
      $ECHO $POUND >> $REPORT
      $ECHO $REPORT_HEADING >> $REPORT
      $ECHO $POUND >> $REPORT
      $CHKCONFIG $OPTIONS >> $REPORT
      $ECHO $POUND >> $REPORT
      $ECHO >> $REPORT
fi

##########################
# PORT BINDING CHECK
##########################
if [ "$PORT_BINDING_CHECK" = "1" ];
then
      $ECHO "Port Binding Check"
      REPORT_HEADING="PORT BINDING CHECK"
      OPTIONS="-a"
      $ECHO $POUND >> $REPORT
      $ECHO $REPORT_HEADING >> $REPORT
      $ECHO $POUND >> $REPORT
      $NETSTAT $OPTIONS >> $REPORT
      $ECHO $POUND >> $REPORT

      if [ -x /usr/bin/lsof ];
      then
            printf "\nThe following programs have got bound sockets:\n"  >> $REPORT
            /usr/bin/lsof -i -n -P | egrep 'UDP|TCP.*LISTEN' | sed 's/....[0-9]u  IP.*     /   /' | sed 's/  FD   TYPE DEVICE SIZE NODE
NAME/PROTO PORT/' | sed 's/ [0-9][0-9]* / /'|sed 's/ PID / /'|sort -u  >> $REPORT
      fi
      $ECHO $POUND >> $REPORT
      $ECHO >> $REPORT
fi

##########################
# ACTIVITY CHECK
##########################
if [ "$ACTIVITY_CHECK" = "1" ];
then
      $ECHO "System Activity Check"
      REPORT_HEADING="SYSTEM ACTIVITY CHECK"
      $ECHO $POUND >> $REPORT
      $ECHO $REPORT_HEADING >> $REPORT
      $ECHO $POUND >> $REPORT

      LongOutput=yes
      Header=yes

      if [ "$LongOutput" = no ]
      then
        [ $Header = yes ] && {
          date  >> $REPORT
          uname -n  >> $REPORT
        }

        who |
          while read Name Tty Mon Day Time Host Rest
          do
            [ -n "$User" -a "$User" != "$Name" ] && continue
            echo "
      $Tty $Name    $Time" >> $REPORT
            case "$Tty" in
              *tty*)       T=`echo "$Tty" | sed -e 's:.*tty\(..\).*:\1:'` >> $REPORT;;
              *)     T=`echo "$Tty" | sed -e 's:/dev/\(..\).*:\1:'` >> $REPORT;;
            esac
```

```
            ps -ct"$T" | tail +2 |
               while read pid tty stat time command
               do
                  echo "    $Tty      $pid $time      $command" >> $REPORT
               done
         done
      else
         # Long form: use "w" output format
         if [ $Header = yes ]
         then FirstLine=1
         else FirstLine=3
         fi
         if [ -z "$User" ]
         then
            w  >> $REPORT
         else
            w | grep "$User"
         fi | tail +$FirstLine
      fi
      $ECHO $POUND >> $REPORT
      $ECHO >> $REPORT
fi

############################
# BASH HISTORY CHECK
############################
if [ "$RUN_HISTORY" = "1" ];
then
      $ECHO "History Check"
      REPORT_HEADING="BASH HISTORY CHECK"
      $ECHO $POUND >> $REPORT
      $ECHO $REPORT_HEADING >> $REPORT
      $ECHO $POUND >> $REPORT
      $CAT /root/.bash_history >> $REPORT
      $ECHO $POUND >> $REPORT
      $ECHO >> $REPORT
fi

############################
# LSOF CHECK
############################
$ECHO "LSOF Check"
if [ "$RUN_LSOF" = "1" ];
then
      REPORT_HEADING="LSOF CHECK"
      OPTIONS="-l"
      $ECHO $POUND >> $REPORT
      $ECHO $REPORT_HEADING >> $REPORT
      $ECHO $POUND >> $REPORT
      $LSOF $OPTIONS >> $REPORT
      $ECHO $POUND >> $REPORT
      $ECHO >> $REPORT
fi

############################
# RPM FILE INTEGRITY CHECK
############################
$ECHO "RPM Check"
if [ "$RPM_FILES" = "1" ];
then
      REPORT_HEADING="RPM FILE INTEGRITY CHECK"
      OPTIONS="-Va"
      $ECHO $POUND >> $REPORT
      $ECHO $REPORT_HEADING >> $REPORT
      $ECHO $POUND >> $REPORT
      $RPM $OPTIONS >> $REPORT
```

```
        $ECHO $POUND >> $REPORT
        $ECHO >> $REPORT
fi

################################
# FIND ALL DEVICE SPECIAL FILES (BLOCK & CHARACTER)
################################
$ECHO "Device File Check"
if [ "$DEVICE_FILES" = "1" ];
then
        REPORT_HEADING="DEVICE SPECIAL FILES"
        $ECHO $POUND >> $REPORT
        $ECHO $REPORT_HEADING >> $REPORT
        $ECHO $POUND >> $REPORT
        $FIND / -xdev \( -type b -o -type c  \) -ls >> $REPORT
        $ECHO $POUND >> $REPORT
        $ECHO >> $REPORT
fi

################################
# FIND ALL REGULAR FILES IN DEV
################################
$ECHO "Regular Files DEV Check"
if [ "$DEVICE_FILES" = "1" ];
then
        REPORT_HEADING="REGULAR FILES IN DEV"
        $ECHO $POUND >> $REPORT
        $ECHO $REPORT_HEADING >> $REPORT
        $ECHO $POUND >> $REPORT
        $FIND /dev -type f ! -name MAKDEV -print >> $REPORT
        $ECHO $POUND >> $REPORT
        $ECHO >> $REPORT
fi

$ECHO "$SCRIPT_NAME has run successfully . . . . . "
$LOGGER -p warning "$SCRIPT_NAME has run successfully . . . . . "

################################
# MAIL REPORT
################################
if [ "$EMAIL_REPORT" = "1" ];
then
        $ECHO "Mailing $REPORT . . . . . "
        $CAT $REPORT | $MAIL -s "$SUBJECT" $MAILTO
fi
exit 0
```

REFERENCES

1. Allen, Bruce. "Monitoring Hard Disks with SMART". Linux Journal. January 2004. URL: **http://www.linuxjournal.com/article.php?sid=6983**.

2. Andreasson, Oskar. "Iptables Tutorial v.1.1.9". URL: **http://iptables-tutorial.frozentux.net.**

3. Andreasson, Oskar. "Ipsysctl Tutorial v.1.0.4". URL**:** **http://ipsysctl-tutorial.frozentux.net/ipsysctl-tutorial.html**.

4. Bandel, David A. "A NATural Progression." Linux Journal. June 2002. URL: **http://www.linuxjournal.com/article.php?sid=5839**.

5. Bandel, David A. "Taming the Wild Netfilter." Linux Journal. September 2001. URL: **http://interactive.linuxjournal.com/Magazines/LJ89/4815.html**.

6. Bandel, David A. "Netfilter 2: In the POM of Your Hands." Linux Journal. May 2002. URL: **http://interactive.linuxjournal.com/Magazines/LJ97/5660.html**.

7. Barrett, Daniel J., Richard Silverman, and Robert G. Byrnes. **Linux Security Cookbook**. O'Reilly Press, June 2003

8. Bauer, Mick. "Battening Down the Hatches with Bastille." Linux Journal. April 2001. URL**:** **http://www.linuxjournal.com/article.php?sid=4547**.

9. Bauer, Mick. "Designing and Using DMZ Networks to Protect Internet Servers." Linux Journal. March 2001.

10. Bauer, Mick. "Hardening Sendmail." Linux Journal. April 2002. URL: http://www.linuxjournal.com/article.php?sid=5753.

11. Chao, Thomas. "Linux XDMCP HOWTO". Revision v1.3. January 2, 2003.

12. Cinelli, Anthorny. "Using Port Sentry and LogCheck." SysAdmin Magazine. March 2002.

13. Cheswick, William R., et al. **Firewalls and Internet Security**. 2nd Edition. Addison-Wesley. 2003

14. Comer, Douglas E. **Internetworking with TCP/IP Volume I: Principles, Protocols, and Architecture**. Upper Saddle River, NJ: Prentice Hall. 2000.

15. Curley, Charles. "Linux Complete Backup and Recovery HOWTO." Revision 0.02. January 27, 2002.
URL: **http://www.ibiblio.org/mdw/HOWTO/Linux-Complete-Backup-and-Recovery-HOWTO/index.html**.

16. Deeths, David and Glenn Brunette. "Using NTP To Control and Synchronize System Clocks Part I: Introduction To NTP." Sun Microsystems, July 2001. URL: **http://www.sun.com/blueprints**.

17. Deeths, David and Glenn Brunette. "Using NTP To Control and Synchronize System Clocks Part II: Basic NTP Administration and Architecture." Sun Microsystems, August 2001. URL: **http://www.sun.com/blueprints**.

18. Deeths, David and Glenn Brunette. "Using NTP To Control and Synchronize System Clocks Part III: NTP Monitoring and Troubleshooting." Sun Microsystems, September 2001. URL: **http://www.sun.com/blueprints**.

19. Deri, Luca and Stephano Suin. "Improving Network Security Using Ntop". University of Pisa, Pisa, Italy.

20. Deri, Luca and Stephano Suin. "Ntop: beyond Ping and Traceroute". University of Pisa, Pisa, Italy.

21. Deri, Luca and Stephano Suin. "Effective Traffic Measurement Using Ntop". University of Pisa, Pisa, Italy.

22. Frisch, AEleen. **Essential System Administration**. Sebastipol, CA: O'Reilly & Associates, 1995.

23. Garfinkle, Simson and George Spafford. **Practical UNIX & Internet Security**. Sebastipol, CA: O'Reilly & Associates, 1996.

24. Hall, Eric A. **Internet Core Protocols: The Definitive Guide**. Sebastipol, CA: O'Reilly & Associates, 2000.

25. Hunt, Craig. **Linux Network Servers**. Alameda, CA: Sybex, 1999.

26. Hunt, Craig. **TCP/IP Network Administration**. Sebastipol, CA: O'Reilly & Associates, 1994.

27. Lowth, Chris. “The Hidden Treasures of iptables”. Linux Journal. April 2004. URL: http://www.linuxjournal.com/article.php?sid=7180.

28. Lucas, Michael. “Eliminating Root with Sudo”. ONLamp.com. August 29, 2002. URL: **http://www.onlamp.com/pub/a/bsd/2002/08/29/Big_Scary_Daemons.html**.

29. Lucas, Michael. “Sudo Aliases and Exclusions”. ONLamp.com. September 12, 2002.

30. Napier, Duncan. “IPTables/Netfilter – Linux’s Next-Generation Stateful Packet Filter.” Sys Admin Magazine. December 2001.

31. Nemeth, Evi et al. **UNIX System Administration Handbook**. Upper Saddle River, NJ: Prentice Hall. 2000.

32. Northcutt, Stephen et al**. Inside Network Perimeter Security: The Definitive Guide to Firewalls, Virtual Private Networks (VPN's), Routers, and Intrusion Detection Systems**. New Riders Publishing. 2002.

33. Pomeranz, Hal. “Improving Sendmail Security by Turning It Off”. SysAdmin Magazine. June 2003. URL:http://www.samag.com/documents/s=8228/sam0306a/0306a.htm.

34. Pras, Aiko. “NTOP – Network TOP An Overview”. University of Twente, The Netherlands.

35. Preston, Curtis W. **UNIX Backup and Recovery**. Sebastipol, CA: O’Reilly & Associates, 1999.

36. Russell, Rusty. “Linux 2.4 NAT HOWTO.” Revision 1.16. November 2001. URL: **http://netfilter.samba.org/unreliable-guides**.

37. Russell, Rusty. “Linux Networking Concepts HOWTO.” Revision 1.13. July 2001. URL: **http://netfilter.samba.org/unreliable-guides**.

38. Russell, Rusty. “Linux 2.4 Packet Filtering HOWTO.” Revision 1.22. November 2001. URL: **http://netfilter.samba.org/unreliable-guides**.

39. Russell, Rusty. “Linux Netfilter Hacking HOWTO.” Revision 1.11. October 2001. URL: **http://netfilter.samba.org/unreliable-guides**.

40. Spitzner, Lance. "Armoring Linux." September 19, 2000. URL: **http://www.enteract.com/~lspitz/linux.html**.

41. Spitzner, Lance. "Auditing Your Firewall Setup." December 12, 2000. URL: **http://www.enteract.com/~lspitz/rules.html**.

42. Spitzner, Lance. "Building Your Firewall Rule Base." January 26, 2000. URL: **http://www.enteract.com/~lspitz/rules.html**.

43. Stephens, James C. Iptables Connection Tracking".
URL: **http://www.sns.ias.edu/~jns/security/iptables/iptables_conntrack.html**

44. Stevens, W. Richard. **TCP/IP Illustrated Volume I: The Protocols**." Reading, MA: Addison-Wesley. January 1999.

45. "The SUSE Linux Unofficial FAQ". URL: http://susefaq.sourceforge.net.

46. Wallen, Jack. "Police Your Network with IPTraf". TechRepublic. April 30, 2002.
URL: http://insight.zdnet.co.uk/hardware/servers/0,39020445,2109393,00.htm.

47. Ziegler, Robert L and Carl B. Constantine. **Linux Firewalls**. Second Edition. Indianapolis, Indiana: New Riders Publishing. 2000.

48. Zwicky, Elizabeth D., Simon Cooper, and D. Brent Chapman**. Building Internet Firewalls**. Sebastipol, CA: O'Reilly & Associates, 2000.

.