Global Information Assurance Certification Paper

SANS GCUX Practical Assignment
Securing chrooted Snort on Slackware 10.0
Charles Pham
Oct 2004
Version 2.1 Option 1

Table of Contents

Abstract

This pager is the written to meet the practical assignment requirement for the SANS GIAC Certified UNIXAdministrator. The pager consists of four main parts.

The first part focused on the planning and risk analysis of the specified server. Specifications of the server and its role are discussed as well as the deployment strategies. In addition, plans are outlined to meet and mitigate the various risks vectors that would arise based on this kind of deployment.

The second part focused on the step-by-step instructions of how to obtain, verify for security, install, and configure the operating and the various required software packages. In addition, detailed instructions are given on how to secure the system such that it will abides by the security principle of least privilege.

The third part of the paper focused designing and implementing an ongoing operation maintenance process and procedures. This includes plans to backup, updates and assures that the system remains fully operation as the surround environment changes.

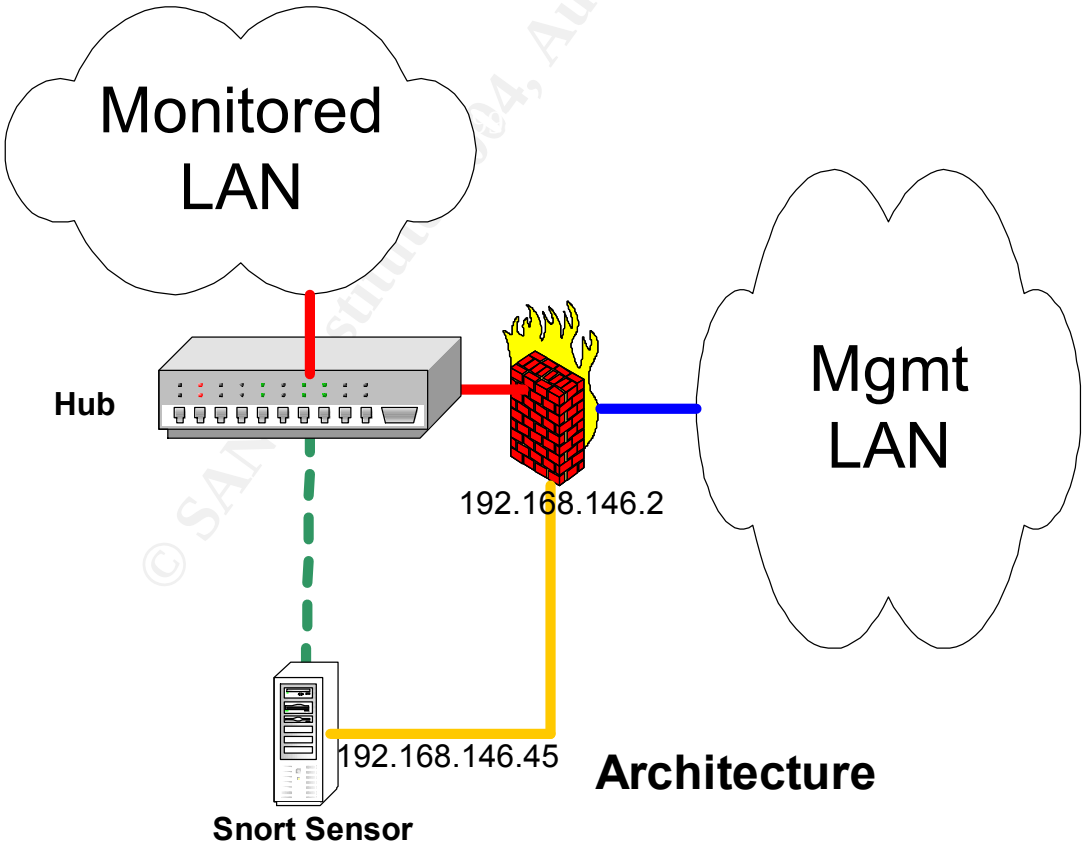
The final part focused on testing and verification of functionality and security of the operating system and the associated software based on the risks identified in the first part of the paper. Details of the steps taken to test and verify are both discussed and captured in their associated screenshots.

Part #1: Server Specification and Risk Mitigation Plan

Objective

To install, configure, and maintain a secure Snort IDS sensor running on Slackware Linux 10.0. Each Snort sensor has two network interfaces with one configured to run in "stealth" mode with no IP address assigned and the other with a static IP address. The "stealth" mode interface is connected to the network to be monitored and act as passive data collector. In addition, outbound access wires on the network cable will be cut to ensure the "stealth" mode operation. The other interface is connected to the management network providing secure shell services via OpenSSH 3.9p1 and secure file transfer via SCP. Deployment of the dual-homed IDS sensor provides a better security design compared to the single network interface sensor with the same degree of manageability.

The diagram below illustrates the ideal placement of this IDS sensor build. However, this build can also be deployed in variations with lower security posture.



The system build will abide by the principle of least privileges and hence only required applications and services will be installed. Slackware is a good choice for this purpose as it is a generic and is highly customizable distribution. In order to best secure the system, binary compiler will not be installed and the Snort IDS applications will be run in a chrooted environment. Applications requiring compilation will be built on a duplicate development machine and transferred to the IDS system through CDROM or secure file transfer. The second system can also serve as backup server for software upgrades, patches, and redundancy.

From a physical security perspective, this build is given the same treatment as a typical server, typically locked in a server cabinet and resides in a restricted server floor space. Hence the focus of securing this build will be on preventing unauthorized access from a network perspective.

System Description

The system, comparable to current low-end desktop, is a single purpose IDS sensor for a small to medium size network running 100Mb Ethernet. Scalability can be achieved through deployment of multiple sensors in strategic network chokepoints or through deployment of enterprise class hardware. However, the build should be modified to accommodate a centrally managed solution.

Hardware

Central Processing Unit: Single Pentium 1.6Ghz Processor
Storage: 20GB IDE Hard Drive
Memory: 256MB RAM
Video Adapter: SVGA
Network Interface Card: Two generic Linux compatible 10/100 Mb Ethernet
Removable Media: CD-RW/DVD Drive

Software

Open-source software, freely downloadable via internet sites, was deployed in order to effectively control cost. The following packages were used in this setup:

OS:	Slackware 10.0	http://www.slackware.com
OpenSSH:	OpenSSH 3.8.1p1	http://www.openssh.com
Kernel:	Linux – 2.4.27	http://www.kernel.org
IPtables	Iptables-1.2.10	http://www.netfilter.org
Snort	Snort-2.2.0	http://www.snort.org/

Risk Analysis

The generally accepted mathematical formula for risk is:

Risk = Threat * Vulnerability * Asset value[1]

For this particular setup, qualitative risk analysis is more applicable as the loss of an IDS sensor is heavily associated with intangible asset. The cost of replacing the hardware or rebuilding the sensor image is low since it is a low end machine and backup images are readily available.

Qualitative risk analysis starts with identifying the scenarios and its associated impact. Below are some of possible scenarios by which risk will need to be evaluated:

Physical Security

Threat: Loss of server due to physical means (i.e. Events cause by natural disaster, crime, hardware failures)

Vulnerability: Sensor's inability to maintain functionality when faced with physical threats

Asset value: Tangible - Value of hardware, labor associated with building the machine. Intangible - Value of IDS data confidentiality, integrity and availability.

Risk: The inability of the organization to detect intrusion

Mitigation:
1) Buy insurance to protect against natural disaster, theft
2) Place the machine is a secure location
3) Place a secondary IDS sensor in a contingency site.
4) Maintain good backup of the machine image.
5) Accept the loss

Placing the server in a secure facility such as a raised floor environment would lower most of the risk associated with physical threats. However, whatever physical security gaps remains associated with the secure facility will also be inherited by the server. In most situations, the security requirements for other servers in the secure facility are normally higher than the IDS sensor and hence are normally more than adequate.

[1] Henry, Kevin. "Risk Management and Analysis". Chapter 3.4 of Reference #14

Trojan Binaries

Threat: Binaries installed might have backdoor allowing malicious users to control the machine.

Vulnerability: Blindly trust downloadable binaries.

Asset value: Intangible – Value of IDS data confidentiality, integrity and availability. Potential for lawsuit if machine is involved in high tech crime. Reputation and credibility.

Risk: The inability of the organization to detect intrusion. Staging ground for further attack. Financial loss due to lawsuit, reputation or credibility.

Mitigation: 1) Check the validity of the binaries through cryptographic signature or one way hash.
2) Compile the code after examining the source
3) Buy insurance to protect against lawsuit

The architecture associated with the IDS sensor placement allows further mitigations against this type of threat. Since there is no direct access to the network from the sensor, a trojaned binary will not be able to "call-home". Management access to the sensor needs to pass through a firewall that has been specifically configured to allow required traffic to and from a specific management segment on a private IP address range. Hence the only real loss in the event of a breach is to the integrity and availability of the IDS sensor. Furthermore, attack of the management network by a compromised IDS sensor should be detectable through the firewall logs.

Configuration Error

Threat: Poorly implemented configuration during initial deployment or through scheduled changes.

Vulnerability: Lack of configuration validation.

Asset value: Intangible – Value of IDS data confidentiality, integrity and availability. Potential for lawsuit if machine is involved in high tech crime. Reputation and credibility.

Risk: The inability of the organization to detect intrusion. Errors leave the machine vulnerable to compromise and their associated losses. Staging ground for further attack. Financial loss due to lawsuit, reputation or credibility.

Mitigation: 1) Validate all configuration and their changes
2) Buy insurance to protect against lawsuit

Similar to the description mentioned in the trojan binaries threats section. The only loss in the even of configuration error is to the integrity and availability of the IDS sensor. The firewall also provides a second layer of validation in the event that the sensor was configured incorrectly.

Vulnerabilities in Network Services

Threat: Exploitation of previously undiscovered vulnerabilities in network services

Vulnerability: Programming errors associated with deployed network services.

Asset value: Intangible – Value of IDS data confidentiality, integrity and availability. Potential for lawsuit if machine is involved in high tech crime. Reputation and credibility.

Risk: The inability of the organization to detect intrusion. Staging ground for further attack. Financial loss due to lawsuit, reputation or credibility.

Mitigation: 1) Limit network access to trusted source
2) Maintain currency with security patches
3) Buy insurance to protect against lawsuit

There is limited threat associated with vulnerabilities in the network services. Attacks originating from the "monitor" network might be able to compromise the sensor. However, the firewall should be able to detect and prevent the attack from going any further. Successful exploitation from the "monitor" network might result in the loss of integrity and availability of the IDS sensor. Attacks originating from the "management" network might be able to compromise the sensor but should not spread any further as there is no other available line of trust or connectivity. If the attacks did originate from the "management" network, there is bigger problem to worry about. Successful exploitation from the "management" network might result in the loss of integrity, confidentiality and availability of the IDS sensor.

Deployment of integrity protection software such as Tripwire and AIDE might be of limited value in this scenario. However, given existing controls, the cost of installation, configuration and maintenance of such software is not justifiable.

Denial of Service

Threat: Network attack aimed at overloading or shutdown the system.

Vulnerability: Sensor's inability to maintain functionality

Asset value: Intangible - Value of IDS data integrity and availability.

Risk: The inability of the organization to detect intrusion

Mitigation: 1) Have an alternative secondary IDS sensor in place.
2) Employ DoS filtering firewall and routers
3) Accept the temporary loss
4) Employ good disk partition practices

Denial of Service attacks from the “monitor” or “management” network might corrupt the IDS data and overflow the sensor log space. The loss of availability and integrity of the data is acceptable if such attacks exceeded the mitigation threshold level. This is because it is virtually impossible to prevent a well-designed DoS attack that is based on bandwidth consumption.

Part #2: Steps to Install and Harden the Server

Obtaining and Verifying the Software

Download the following software from the site listed in the software section above or through one of its mirror:

Slackware 10.0 ISO
Key: http://slackware.com/gpg-key
Fingerprint: EC56 49DA 401E 22AB FA67 36EF 6A44 63C0 4010 2233
CD1
Image: slackware-10.0-install-d1.iso
Signature: slackware-10.0-install-d1.iso.asc
CD2 (However, this setup will only required CD1)
Image: slackware-10.0-install-d2.iso
Signature: slackware-10.0-install-d2.iso.asc

Linux Kernel – 2.4.27
Key: http://www.kernel.org/signature.html
Fingerprint: C75D C40A 11D7 AF88 9981 ED5B C86B A06A 517D 0F0E
Software: linux-2.4.27.tar.bz2
Signature: linux-2.4.27.tar.bz2.sign

Snort-2.2.0
Key: http://www.snort.org/public-key.html
Fingerprint: F504 0CEF 9B3E C272 36D2 1EC7 9449 15EA 1946 E4A1
Software: snort-2.2.0.tar.gz
Signature: snort-2.2.0.tar.gz.asc

Verification of integrity of the downloaded software can be done through cryptographic signature or a one time hash. This is done to prevent a malicious 3rd party from introducing trojan into existing software packages.

The keys are then imported into gpg one at a time as followed:

>gpg --import keyfile.asc

Then, to get KEY ID:
>gpg --list-keys

The next step is to assign trust to the key. This is accomplished by going into edit mode:

>gpg --edit-key KEY_ID

Verify that it is the correct public key by checking the fingerprint against public key server or through Google searches on the Internet. The following command to show the key fingerprint:

command>fpr

Once a comfort level has been reached, the key should be sign using the following command:

command>sign

Answer the questions and enter the passphrase for the key being used to sign. Optionally, the key can also be added to the trusted database with the following command:

command>trust

Answer the question and exit with the following command:

command>quit

Save the changes. To check the software download, both the signature file and the software package need to be in the same directory. The command is as followed:

>gpg --verify signature_file software_pkg

The results of verified software should be “Good signature”.

There is only one identified role for the IDS sensor, that of an administrator. Access is limited to the person assigned to the task of administrating this box and since root access is only feasible by means of sudo, a non-privileged account is created to allow this administrator to log in. The risk associated with privileges escalation attack is fairly low as no other users are allowed on this system.

Installing Slackware 10.0

Burn the Slackware ISO images to CDs and use them to boot. However, make sure that the network cable is not connected to the machine prior to turning it on. Following strict guidelines, the machine must be hardened and secured before getting connected to the network.

Booting up with Slackware ISO image CD1 will bring up a splash screen with the boot: prompt. Press <ENTER> to continue since there is no special hardware in

this setup. Press <ENTER> again at the keyboard map prompt will bring up the Slackware login: prompt. Type in “root” <ENTER> to continue.

Partitioning

Good partitioning scheme can withstand most DoS attacks that are based on disk space consumption. As an IDS sensor, it is expected that most of the disk space will be used by the log data. Hence, the disk space allocation will be as followed:

hda1: / = 200 MB
hda2: /home = 10GB
hda3: SWAP = 512 MB (2 times the physical memory)
hda4: /var = remainder of disk

Separating the root (/) partition from the rest of the disk is a good practice to prevent SUID exploitation. To start the partitioning process type the following commands:

root@slackware:/# cfdisk /dev/hda

Since this is a new setup, type “y” to start with a “zero table”. Select new -> primary -> 200 -> beginning -> bootable for the root (/) partition.

Select free space -> new -> primary -> 10000 -> beginning for the /home partition. The snort log will be under /home/snort/var/log/snort since it will be running under chroot.

Select free space -> new -> primary -> 512- > beginning -> Type - > <ENTER> -> <ENTER> to change swap partition to type 82 (Linux SWAP).

Select free space -> new -> primary -> <ENTER> to assign the remainder of the disk to the /var partition.

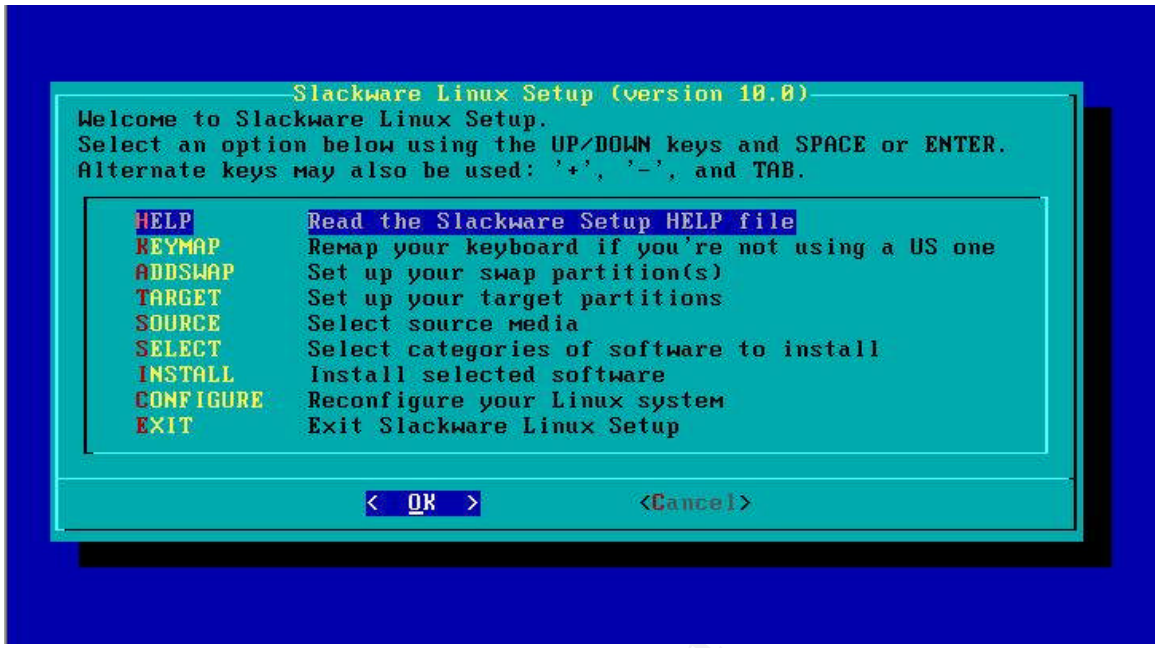
Select write -> <ENTER> -> yes -> quit to save and exit.

Setup

Back at the root prompt, type the following to enter the setup screen:

root@slackware:/# setup

will display the following screen:



Select ADDSWAP -> <ENTER> -> <ENTER> to setup the swap partition. Once configured hit <ENTER> to continue.

Select /dev/hda1 for root (/) partition and follow the prompt to format the partition using ReiserFS.

Select /dev/hda2 for /home partition and follow the prompt to format with ReiserFS and mount.

Select /dev/hda4 for /var partition and follow the prompt to format with ReiserFS and mount. Once completed, they will be added to /etc/fstab.

Installation will be from the Slackware CD-rom and let it auto scan for the CD/DVD drive which will be mounted to /dev/hdc. Next step will be package selection.

Select package series A (base), AP (additional application), D (development), and N (networking). For prompting mode, select menu.

Remove the following from the default package A: cpio, cups, floppy, isapnptools, jfsutils, kbd, loadlin, lprng, minicom, pcmcia-cs, and xfsprogs.

Select only the following from the package AP: diffutils, lsof, sudo, and vim.

From the package D, select only PERL as it is useful for log analysis.

From the package N, select only the following: iptables, openssh, tcpdump, and tcpip. Follow the prompt to continue with the package installation.

At the Install Linux Kernel prompt, leave the default as from CD and continue. It is recommended that a boot disk be made as it will be handy in the event of a crash. Select No modem and No to Hotplug subsystem at boot. Install simple lilo with the default frame buffer console with MBR as the destination.

Select ps2 for mouse and yes to gpm for cut and paste ability. Skip the network configuration as this will be done manually. Leave the default as enable for the rc.syslog and perhaps the SSH service for startup. Hardware clock will be set to local time as per selected time zone. After setting a secure root password (i.e. first character of each word in a long and unique passphrase), select exit, remove the boot media, and reboot.

After the reboot, go into pkgtool and remove the following non-essential package (Check the package series tagfiles to verify that "REQUIRED" package are not accidentally removed. These are identified with the keywords ADD): acpid, apmd, genpower, and pciutils.

Configuring the network manually starting the hostname with the following commands:

echo "gcux.sans.net" > /etc/HOSTNAME
/bin/hostname gcux

Setting the interface for eth0 connected to the "monitor" network to "stealth" as followed:

vi /etc/rc.d/rc.local

Add the following lines :

echo "Setting eth0 to stealth…"
/sbin/ifconfig eth0 0.0.0.0 promisc

The steps for setting the IP addresses for the interfaces for eth1 and the gateway as followed:

vi /etc/rc.d/rc.inet1.conf

Under the eth1 section, set the static IP address and netmask:

IPADDR[1]="192.168.146.45"
NETMASK[1]="255.255.255.0"
USE_DHCP[1]="no"

Under the default gateway section, set the IP address:

GATEWAY=”192.168.146.2”

Kernel Upgrade

On the development machine, download, and verify the kernel linux-2.4.27.tar.bz2. Decompress and unpack the package with the following commands:

bunzip2 linux-2.4.27.tar.bz2
tar xvf linux-2.4.27.tar

Change directory and configure the kernel build with the following commands:

cd linux-2.4.27
make mrproper
make menuconfig

Make the following changes to the default options:

Loadable module support: Enable loadable module support (N)
Processor type and features: Pentium-4
Processor type and features: Symmetric multi-processing support (N)
General Setup -> PCMCIA/CardBus support: PCMCIA/CardBus support (N)
General Setup -> Power Management support (N)
Networking options: Network packet filtering (replaces ipchains) (Y)
Networking options -> TCP/IP networking:
    IP: multicasting (N)
    IP: TCP syncookie support (disabled per default) (Y)
Networking options -> IP: Netfilter configuration:
    Connection tracking (Y)
    Iptables support (Y)
        limit match support (Y)
        MAC address match support (Y)
        Connection state match support (Y)
        Packet filtering (Y)
        LOG target support (Y)
SCSI support: SCSI support (N)
Network device support -> Ethernet (10 or 100Mbit)
    AMC PCnet32 PCI support (Y)
    EtherExpressPro/100 (N)
File systems: Reiserfs support (Y)
File systems -> Network File Systems:
    NFS file system support (N)
    NFS server support (N)

Sound: Sound card support (N)
USB support: Support for USB (N)

Select Exit and save the new kernel configuration to the default file.

Disabling the Loadable module support help prevent attack via loadable kernel level rootkit. Power management is turn off as the sensor needs to be on 24x7. Netfilter and the various security options are enabled to support IPtables firewall. IP multicasting is disabled as the system is not expected to participate in this type of activity. Filesystem in use is Reiserfs on IDE hard drive. The remaining settings are set based on devices available on the system.

To start the compilation process the following commands are executed:

make dep clean bzImage

The bootable kernel bzImage is located in the arch/i386/boot directory once the compilation finished, Copy this file onto CD and transfer to the Snort sensor system. Mount the CD, copy the new kernel to the boot directory and make it readable to root only using the following command:

mount /dev/hdc /mnt/cdrom
cp /mnt/cdrom/bzImage /boot/vmlinux
chmod 400 vmlinux

Update the LILO configuration by running the vi editor:

vi /etc/lilo.conf

The new configuration is as followed:

```
image = /boot/vmlinux
  root = /dev/hda1
  label = Linux
  read-only
image = /boot/vmlinuz
  root = /dev/hda1
  label = backup
  read-only
```

The old kernel is relabeled as backup in case of problem with booting the new kernel. Having a backup kernel that is one build behind is useful when it come to updating with new kernel image. There is an option to setup the LILO password to protect the bootup process. However, this is not necessary in this build as

there is no restriction associated with the differing kernel image and security set via the BIOS password is deemed sufficient. Update the LILO with:

/sbin/lilo

and reboot and test the new kernel.

To prevent people from rebooting from the keyboard using the "three finger salute", edit the /etc/inittab and comment out the following line:

ca::ctrlaltdel:/sbin/shutdown -t5 -r now

Next step is to setup the logging functionality on the system. Enable the kernel message by editing the syslog.conf:

vi /etc/syslog.conf

Modify the line for kern.* to:

kern.* -/var/log/kernel

Note the (-) sign to prevent resync after every logging to improve performance at the cost of lost data. To create the kernel file and set the permissions on the file:

touch /var/log/kernel
chmod 640 /var/log/kernel

In addition, edit the /etc/logrotate.d/syslog to include /var/log/kernel to ensure that it will be taken care of by logrotate as well. The changes to syslogd will take place after next reboot or if necessary be forced through sighup.

OS Hardening

The first step to harden the OS is to setup a user account that is not root. Since shadow suite is installed by default, the following commands are used to create a new account:

 # /usr/sbin/useradd cpham –g users –s /bin/bash –m

The commands will create a userid cpham belonging to the users group with bash shell and make a directory in /home/cpham. Next set the directory permission for the user with:

chmod 700 /home/cpham

and set the password with:

passwd cpham

Checking the /etc/passwd, a number of defaults accounts are setup and will not be needed for this particular system. Hence, the following accounts are deleted using the userdel command:
games, news, uucp, ftp, rpc, mysql, operator, and gdm.

It should be noted that of the remaining accounts, only the cpham and root account has a shell account. From this point onward, access to root privileges should be done through su. Log out of root account and log in with the new user account.

To prevent direct root login into the system, edit the /etc/securetty file and put a pound sign (#) in front of all the listing in the file. By default, root can log in from the console and tty1 through tty6.

In addition, su attempts should be log to a separate file by removing the comment (#) from the following entry from the /etc/login.defs file:

SULOGFILE /var/log/sulog

To create the sulog file and set the permissions on the file:

touch /var/log/sulog
chmod 640 /var/log/sulog

In addition, edit the /etc/logrotate.d/syslog to include /var/log/sulog to ensure that it will be taken care of by logrotate as well. In addition insert the following lines before the sharedscripts line to have the logs rotated daily and compressed for 4 weeks cycle:

daily
rotate 28
compress

Next, set the banners as a proactive measure against unauthorized access as followed:

****
WARNING: Access to this system is monitored and is only for authorized users. Unauthorized usage is prohibited and is subject to prosecution.
****

This message should go into the /boot/boot_message.txt (must run /sbin/lilo for the update to work), the /etc/motd and the /etc/issue.

Next up are the network services. Check for running services with the following command:

netstat –a

As expected, the only running service is SSH. Review the /etc/inetd.conf, /etc/rc.d/rc.inet2, /etc/rc.d/rc.M and /etc/rc.d/rc.S for other possible options. It should be noted that IPv4 forwarding is controlled by the script /etc/rc.d/rc.ip_forward and is disabled by default.

In addition to a firewall, TCP Wrappers is a good tool to use to limit IP based access to the system. Configuration is fairly straightforward and should start with denying all connections by adding the following entry to the /etc/hosts.deny file:

ALL: ALL

To grant SSH access to the system from the management network, the following entry is added to the /etc/hosts.allow file:

sshd: 192.168.146.

To verify that the setup is successful, the following command is executed:

tcpdchk

this resulted in the following error:

warning: /etc/hosts.allow, line 11: sshd: no such process name in /etc/inetd.conf

This does not affect the operation and is easily remedy by adding an entry for sshd in /etc/inetd.conf.

NetFilter Firewall

To really control access to the system, a host based firewall provides flexibility and stronger enforcement policy. NetFilter firewall and its options were previously built into the new kernel and provide another layer of protection through stateful packet filtering and comprehensive logging of traffic for the IDS sensor.

There are three main rules (INPUT, OUTPUT, and FORWARD) associated with the firewall and their ordering affects security filtering and performance. The

rules are processed from a top down approach and are designed to follow the principle of least privileged. The firewall is automatically loaded at bootup by inclusion of the following startup script into the /etc/rc.d/rc.local:

echo “Shields up…”
/etc/rc.d/rc.firewall

On the fly changes to the script is also possible by executing rc.firewall.

As a host based firewall, the FORWARD rule is set to deny. The INPUT rule is set to:

- allow SSH traffic on eth1 interface
- be stateful aware
- allow localhost communications
- allow sniffing on eth0 interface

The OUTPUT rule is set to:

- be stateful aware
- allow localhost communications

In addition, the firewall should block malicious activities and log them. The rules below are adopted by a sample NetFilter rule set written by Robert L. Ziegler. A copy of this sample rule set is attached in the Appendix.

```
#####################################################################
#!/bin/sh
# /etc/rc.d/rc.firewall
# Invoked from /etc/rc.d/rc.local
#####################################################################

# Flush all old rules on restart
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD

# Default policies to drop all packets
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Reject source routed packets
for x in /proc/sys/net/ipv4/conf/*/accept_source_route; do
  echo 0 > $x
done
```

```
# Reject redirections
for x in /proc/sys/net/ipv4/conf/*/accept_redirects; do
  echo 0 > $x
done

# Enable TCP SYN cookie, ICMP broadcast echo, ICMP bad error message
# protection
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
echo 1 > /proc/sys/net/ipv4/ICMP_echo_ignore_broadcasts
echo 1 > /proc/sys/net/ipv4/ICMP_ignore_bogus_error_responses

# Enable IP spoofing protection
for x in /proc/sys/net/ipv4/conf/*/rp_filter; do
  echo 1 > $x
done

# Enable logging of spoofed, source routed and redirect packets
for x in /proc/sys/net/ipv4/conf/*/log_martians; do
  echo 1 > $x
done

# Declare environment variables
MGMT_IFACE=eth1
IPADDR=192.168.146.45
MGMT_NET=192.168.146.0/24
SNORT_IFACE=eth0
LOOPBACK=127.0.0.1

# --------INPUT CHAIN--------
#
# Define the access policy for all traffic destined to the system itself.

# Block and log spoofed source IP packets
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 172.16.0.0/12 -d 0/0 -j LOG --log-
prefix " 172.16 spoof "
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 172.16.0.0/12 -d 0/0 -j DROP
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 10.0.0.0/8 -d 0/0 -j LOG --log-
prefix " 10.0 spoof "
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 10.0.0.0/8 -d 0/0 -j DROP
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 0.0.0.0/32 -d 0/0 -j LOG --log-
prefix " 10.0 spoof "
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 0.0.0.0/32 -d 0/0 -j DROP
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 127.0.0.0/8 -d 0/0 -j LOG --log-
prefix " 127.0.0.1 spoof  MGMT iface"
```

```
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 127.0.0.0/8 -d 0/0 -j DROP
iptables -A INPUT -p ALL -i $SNORT_IFACE -s 127.0.0.0/8 -d 0/0 -j LOG --log-
prefix " 127.0.0.1 spoof  SNORT iface"
iptables -A INPUT -p ALL -i $SNORT_IFACE -s 127.0.0.0/8 -d 0/0 -j DROP
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 224.0.0.0/8 -d 0/0 -j LOG --log-
prefix " 224.0.0.0 spoof "
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 224.0.0.0/8 -d 0/0 -j DROP
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 169.254.0.0/16 -d 0/0 -j LOG --
log-prefix " Autoconf spoof "
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 169.254.0.0/16 -d 0/0 -j DROP
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 240.0.0.0/4 -d 0/0 -j LOG --log-
prefix " 240.0.0.0 spoof "
iptables -A INPUT -p ALL -i $MGMT_IFACE -s 240.0.0.0/4 -d 0/0 -j DROP
iptables -A INPUT -p ALL -i $SNORT_IFACE -s $MGMT_NET -d 0/0 -j LOG --
log-prefix " MGMT spoof on SNORT iface "
iptables -A INPUT -p ALL -i $SNORT_IFACE -s $MGMT_NET -d 0/0 -j DROP

# Log and drop where applicable, illegal or suspicious packets on the MGMT
# interface.
iptables -A INPUT -p tcp --tcp-flags ALL SYN,FIN -i $MGMT_IFACE -j LOG --log-
prefix " SFScan "
iptables -A INPUT -p tcp --tcp-flags ALL SYN,FIN -i $MGMT_IFACE -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL SYN,ACK -i $MGMT_IFACE -j LOG --
log-prefix " SAScan "
iptables -A INPUT -p tcp --tcp-flags ALL SYN,ACK -i $MGMT_IFACE -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL FIN -i $MGMT_IFACE -j LOG --log-prefix
" FSscan "
iptables -A INPUT -p tcp --tcp-flags ALL FIN -i $MGMT_IFACE -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL NONE -i $MGMT_IFACE -j LOG --log-
prefix " NUScan "
iptables -A INPUT -p tcp --tcp-flags ALL NONE -i $MGMT_IFACE -j DROP
iptables -A INPUT -p tcp --tcp-flags ALL FIN,PSH,URG -i $MGMT_IFACE -j LOG
--log-prefix " XMAS"
iptables -A INPUT -p tcp --tcp-flags ALL FIN,PSH,URG -i $MGMT_IFACE -j
DROP
iptables -A INPUT -p icmp -f -i $MGMT_IFACE -j LOG --log-prefix " ICMPFRAG "
iptables -A INPUT -p icmp -f -i $MGMT_IFACE -j DROP

# Allow sniffing/promiscuous mode on eth0
iptables -A INPUT -i $SNORT_IFACE -j ACCEPT

# Allow loopback communications
iptables -A INPUT -s $LOOPBACK -d $LOOPBACK -j ACCEPT

# Allow system to be managed via SSH
```

```
iptables -A INPUT -p tcp -i $MGMT_IFACE -s $MGMT_NET --dport 22 -j
ACCEPT

# Allow established sessions through
iptables -A INPUT -m state --state ESTABLISH,RELATED -j ACCEPT

#Log and drop the rest as per default policy
iptables -A INPUT -s 0/0 -j LOG --log-prefix " DROP_INPUT "


# --------OUTPUT CHAIN--------
#
# Define the access policy for all traffic originating from the firewall itself.

# Allow loopback communications
iptables -A OUTPUT -s $LOOPBACK -d $LOOPBACK -j ACCEPT

# Allow established sessions through
iptables -A OUTPUT -m state --state ESTABLISH,RELATED -j ACCEPT

#Log and drop the rest as per default policy
iptables -A OUTPUT -s 0/0 -j LOG --log-prefix " DROP_OUTPUT "
```

Packaged Software

OpenSSH

OpenSSH is a secure replacement for clear text remote service such ftp or telnet. Version 3.8.1p1 is available as part of the Slackware package. The current version 3.9p1 offers no security fix or significant new features and hence is not worth the upgrade.

During installation, SSH would create a public and private system/host keys that are used to protect password authentication over the network. Configuration of SSH starts with editing the /etc/ssh/sshd_config.

Set SSH to only use protocol 2 with by inserting the following line:
Protocol 2

Disable root login with the following line:
PermitRootLogin no

Disable motd
PrintMotd no

Enable print last login
PrintLastLog yes

Set banner to /etc/issue
Banner /etc/issue

To increase the security
After saving the changes, stop sshd with:
 # /etc/rc.d/rc.sshd stop

and restart it with:
 # /etc/rc.d/rc.sshd start

Optionally, a more secured way of connecting to SSH is via PKI. However, it does not scale well due to the tedious task of key exchange and management. For this purpose, using password authentication is deemed sufficient and connectivity from a remote machine can be initiated with the following command:

ssh -l userid ip_address_of_SSH_server or
ssh userid@ip_address_of_SSH_server

Where the userid is the local username on the SSH server. Connectivity from a remote machine for the first time will yield a authenticity warning if the host public key information is not contain in /etc/ssh/ssh_known_hosts or the $HOME/.ssh/known_hosts.

SSH connect are by default logged under the info facility in syslog.

Snort

To start setting up Snort running in chroot, download the source code onto the development system where the compilation can take place. For this setup, Snort version 2.2.0 was downloaded and the steps once the source files are downloaded are as followed:

tar –xvzf snort-2.2.0.tar.gz
cd snort-2.2.0
./configure
make

A compiled version of Snort is available in the src directory. Copy the file over to the IDS sensor using scp with the following command:

scp -p ./src/snort <userid>@SSHd_IP:/home/<userid>

It will prompt for password and if successful, would copy the file over. The (-p) options would preserve the timestamps and modes of the file copied. We will also need the snort and threshold configuration located in the etc directory. The threshold configuration is not needed in the initial setup but will be useful after the IDS has been in operation for a period of time. The following command to copy them over:

tar -cvzf etc.tgz etc
scp -p etc.tgz <userid>@SSHd_IP:/home/<userid>

In addition to the configuration files, the rules files are also needed. They are located in the rules directory and it is easier to tarball them before sending them over. The following commands were used:

tar -cvzf rules.tgz rules
scp -p rules.tgz <userid>@SSHd_IP:/home/<userid>

There is one additional file which is optional but is useful for the maintenance of log is the snort.logrotate file located in the contrib/rpm directory. Copy it over with the following command:

scp -p ./contrib/rpm/snort.logrotate <userid>@SSHd_IP:/home/<userid>

On the IDS sensor system, extract the etc.tgz and rules.tgz with the following commands:

tar xvzf etc.tgz
tar xvzf rules.tgz

Now su - to root and setup snort to run under user snort and group snort. This will prevent access to root should snort get compromised. The commands are as followed:

 # /usr/sbin/groupadd snort
 # /usr/sbin/useradd -g snort snort –m
 # cd /home/snort
 # mkdir -p usr/local/bin
 # mkdir -p var/log/snort
 # vi /etc/shadow

Change the "!" in the snort field to "*" to disable the account and prevent sign in. Do the same for the /etc/gshadow file. Place the snort binaries in the /home/snort/usr/local/bin and assign access:

 # cp /home/<username>/snort /home/snort/usr/local/bin/snort

 # chmod 700 /home/snort/usr/local/bin/snort

For the configuration and rules files, the following commands are executed:

 # mkdir -p etc/snort
 # cd etc/snort
 # cp /home/<username>/etc/* .
 # rm Make*
 # chmod 600 *
 # mkdir rules
 # cd rules
 # cp /home/<username>/rules/* .
 # chmod 600 *

In order to run as user snort, the configuration and logs must be owned and readable by user snort and group snort. For the log directory /home/snort/var/log/snort, the permission is set to full for user and read execute for group. To complete the logs setup, performs the following commands:

 # cd /home
 # chown –R snort.snort snort
 # chmod 730 /home/snort/var/log/snort
 # cd /etc/logrotate.d
 # cp /home/<username>/snort.logrotate snort

Edit the snort file to as followed:

```
# /etc/logrotate.d/snort
# $Id: snort.logrotate,v 1.3 2003/12/12 02:05:51 cazz Exp $

/home/snort/var/log/snort/alert /home/snort/var/log/snort/*log
/home/snort/var/log/snort/*/alert /home/snort/var/log/snort/*/*log  {
    daily
    rotate 28
    missingok
    compress
    postrotate
        /etc/rc.d/rc.snortd
    endscript
}
```

The directories are updated to correspond to the chrooted snort and the logs are set to be rotated daily and compress for 4 weeks. Now for the snort configurations, enter the editor using vi:

 # vi /etc/snort.conf

Change the RULE_PATH from "../rules" to "/home/snort/etc/snort/rules". In addition, enable all the rules by removing the comment (#) besides all the include statement for rules.

For the test run with full alert, application data dump, under snort user and group, on eth0 interface, chrooted to /home/snort, execute the following command:

 # /home/snort/usr/local/bin/snort -c /home/snort/etc/snort/snort.conf -A full -d -g snort -u snort -i eth0 -l /home/snort/var/log/snort -t /home/snort -T

It should display a number of different lines with the following line as one of them upon exit:

"Snort successfully loaded all rules and checked all the rule chains!"

Create a /etc/rc.d/rc.snortd script to simplify the task of starting and stopping snortd. It should contain the following entry:

```
#!/bin/sh
killall –q snort
/home/snort/usr/local/bin/snort -c /home/snort/etc/snort/snort.conf -A full -d -g
snort -u snort -i eth0 -l /home/snort/var/log/snort -t /home/snort –D
```

Save the file and change the permission to 755. To start Snort automatically during bootup, the following lines are added to /etc/rc.d/rc.local

```
echo " Starting the pig…"
/etc/rc.d/rc.snortd
```

This completes the chroot snort installation. Check the /home/snort/var/log/snort directory for the logs. Monitoring of the logs can further be simplified with scripts found in the contribute directory. As the final step, remove the scp copied Snort files in the /home/<user> directory.

Final Lockdown

Now that all the software has been configured and installed, the system will required to be fully lockdown.

Patches:

Checking the list of security advisories at http://www.slackware.com/security/list.php?l=slackware-security&y=2004, there is no applicable patch that should be applied.

However, in the event where an upgrade is required, the following steps should be followed:

1) Download the patch to the development server from the required site and check against the md5 signature using md5sum or cryptographic signature using gpg.
2) Transfer via scp to the IDS sensor
3) Go into single user mode with the command: telinit 1 as root
4) Login as usual and su to root
5) Execute upgradepkg package.tgz
6) Go back into multi user mode with the command: telinit 3 as root

Filesystem security:

SUID and SGID files give the user the ability to temporary run as the owner of the files and on occasion can be exploited to give root access. Check for SUID file with the following command as root:

 # find / -perm 4000

Similarly, the command to check for SGID file is:

 # find / -perm 2000

The results indicate the few files that has SUID/SGID are located in the /bin or the /usr directory. Now that the software install has been completed, these directories should be protected. To lock down the system, edit the /etc/fstab for the following changes:

change the defaults field for / to ro
change the defaults field for /home and /var to defaults, nosuid

Save the changes and shutdown. At this time, it is a good idea to make a backup of the system and use it as a baseline for contingency purpose. Using dd in single user mode or other alternative bit-to-bit copy software or device is highly recommended.

Hardware security:

BIOS:

At boot up opportunity, go into the BIOS to set a user and superuser password forcing a password to be entered upon booting and also prevent changes to the system BIOS without the superuser password. In addition, the order to booting

should be changed so that the system will automatically boot up from the hard drive first and to also disable hard drive auto-detect. This prevents the system from booting alternative media which can be exploited to gain root access to the system hard drive. It should be noted that the above measures can be circumvented should an attacker have unlimited physical access to the system. The BIOS settings will be loss with a CMOS reset.

NETWORK CABLE:

As part of the snort setup, the cable used on eth0 is set to receive only mode. This can be achieved by shorting pin 1 and 2 on the system connectoid. Pin 3 and 6 are set as straight through. Lastly, on the hub connectoid, pin 1 is connected to pin 3, and pin 2 is connected to pin 6.

Part #3: Design and Implement Ongoing Maintenance Procedures

From a change perspective, there are scheduled changes and there out-of-cycle changes. Scheduled changes take place as part of the periodic review and updates and are usually triggered via controlled plan. Out-of-cycle changes occur when an external event trigger the need to either schedule a change or react to it if there is insufficient time.

Proactive/Scheduled change:

From a proactive perspective a number of activities can be planned for. These periodic activities are: scheduled change and updates, review of logs, network vulnerability assessment, and tracking of security trends.

Maintenance:

On a regular basis, there might be a need to make changes to the configuration, updates the IDS signatures, make a backup, apply patches, or upgrades the system software. Changes that occur here will most likely required filesystem security mentioned above to be reset before hand and set back after the fact. As part of the change management framework, there is always a backout plan to undo the changes in the event of failure. In order to minimize cost and maximize production uptime, the following change schedules are implemented (out-of-cycle changes are discussed later):

- On a monthly basis, during the specified maintenance window of the last Sunday of the month between the hours of 12am to 6am, the system can undergo configuration changes, updates to IDS signatures, and backup. Snort IDS signature are downloadable from www.snort.org. Backup that occurs in this cycle are limited to the logs data. There should be offloaded via the tar utility onto CDs, tape drive, or remote file server.
- On a 3 months basis, during the specified maintenance window of the last Sunday of the month between the hours of 12am to 6am, the system can undergo software patching. The patches must be tested on a backup system and signed off as being ready at least a week prior. The procedures for patches and packages upgrades were described in the patches section above. For backup, the system should be backup using utils such as dd, Symantec Ghost, or specialized hardware that are capable to perform bit-to-bit copy. The following software must be checked for patches: Slackware distribution, Linux kernel, Snort IDS, OpenSSH, and IPtables. The patches must be evaluated for security,

features and balance them against the effort required to patch. For instance, it might be possible to forgo a minor kernel release version if such release does not address any system impact problem.

- On an annual basis, during the specified maintenance window of the last Sunday of the month between the hours of 12am to 6am, the system can undergo software upgrades. The new software must be tested on a backup and signed as being ready at least two weeks prior. Similar to the 3 months cycle, the upgrades are for Slackware distribution, Linux kernel, Snort IDS, OpenSSH, and IPtables. Similarly to the patch cycle, the new software version must be evaluated on the merit of effort cost versus security and features. New products implementation can also be introduce in this cycle as they represents significant changes to existing configurations and processes. For example, implementation of log review product that can automate the existing log review process.

Log Reviews:

On a regular basis, there is a need to make reviews the various logs and configurations associated with the installed software. The following schedules are implemented:

- On a 6 hours basis, the snort logs are reviewed for sign of intrusion. This can also be enhanced by implementing a near real-time alert via email or pagers. In addition, the system logs are quickly reviewed for sign of problems. Things such as wtmp review using the last command which showed user login activities. In addition, secure and sulog logs should be checked using the tail/more commands.
- On a weekly basis, the messages, kernel, and syslog logs are reviewed using tail/more commands for sign of system problem. In addition, verification that logrotation is working by checking that the compress log files are up-to-date and are sequential. Lastly, the system health using ps and top commands, disk space using df commands, and time using date command are within desired boundaries.

There are products which can ease the review efforts. However, at this time, none has been chosen. In the long term, it would make sense to deploy such product as it would significantly reduce the review time and hence cost.

Vulnerability Assessment:

On a 6 months basis, a network vulnerability assessment should take place to ensure that the system is still secured. Tools deployed can consist of freely available software such as nmap or nessus or commercial tools such as foundscan and retina. The activity should be planned for such that change

records are raised and approved at least 1 week prior to the actual assessment. This is to ensure that negative side-effects such as service shutdown can be responded to.

Security Trends:

On a daily basis, a check of current security trends relating to new vulnerabilities, threats and attacks trends will allow administrator to better plan for out-of-cycle changes. This might or might not result in a scheduled change cycle. Sites such as Slackware security, Bugtraq, OpenSSH security, Kernel security, and Linux security should be reviewed as a minimum.

Reactive/Unscheduled change:

From a reactive perspective, the changes that occur here can not be planned for or is poorly planned for. These activities are: new vulnerabilities affecting system viability, vulnerabilities identified as part of the planned assessment activities, hardware failure, and natural disaster.

For the most part, these changes are best provisioned for through contingency planning. Activities associated with natural disaster should be assessed and planned for during the project deployment phase. Activities associated with hardware failure are best mitigated through system and data backup as well as spare hardware that can be re-introduced in a relatively short time.

For activities associated with vulnerabilities, there is a need to analyze and assess the risks that are introduced. Through assessment, a plan should emerge to address the problems identified. This might means that an identified vulnerability remains un-patched for several months until sufficient resources are available or if exploitation occurred and incident response process is initiated.

Part #4: Test and Verify the Setup

Checking and verification can be broken down into two parts, that of security and functionality. However, it is difficult to break them out into separate components since some the tools installed can only be considered functional if they pass the security tests. To simplify the situation, the approach taken will be from a network to local perspective.

NMAP:

Nmap 3.50 was installed on the development system as part of the Slackware package installed. The latest version 3.70 can be downloaded from http://www.insecure.org/nmap/. The new version do not have changes which would prove significant for this purpose and hence the test will proceed with version 3.50.

Nmap will be run a computer on the same network residing between the firewall and the eth1 interface on the IDS system. Connectivity is provided via a hub.

Start the scan with TCP SYN scan covering the defaults ports in nmap-services without pinging and reverse DNS resolution

```
# nmap 3.50 scan initiated Mon Oct 11 13:00:31 2004 as: nmap -vv -sS -F -n -P0
-oN scan1 192.168.146.45
Interesting ports on 192.168.146.45:
(The 1216 ports scanned but not shown below are in state: filtered)
PORT   STATE SERVICE
22/tcp open  ssh

# Nmap run completed at Mon Oct 11 13:15:13 2004 -- 1 IP address (1 host up)
scanned in 881.674 seconds
```

Now for UDP covering the default ports in nmap-services (around 1008) scans:

```
# nmap 3.50 scan initiated Mon Oct 11 13:21:05 2004 as: nmap -vv -sU -F -n -P0
-oN scan3 192.168.146.45
All 1008 scanned ports on 192.168.146.45 are: filtered

# Nmap run completed at Mon Oct 11 13:41:19 2004 -- 1 IP address (1 host up)
scanned in 1213.561 seconds
```

The results confirmed that the firewall does filter these types of packets and only the port for SSH is accessible. Looking through the /var/log/syslog and /var/log/kernel revealed the scanning activities as logged by IPtables.

```
root@gcux:/var/log# tail -n 5 syslog
Oct 11 21:03:23 gcux kernel:  DROP_INPUT IN=eth1 OUT= MAC=00:0c:29:75:b1:9d:00:0
c:29:5a:5e:b3:08:00 SRC=192.168.146.138 DST=192.168.146.45 LEN=60 TOS=0x00 PREC=
0x00 TTL=64 ID=12429 DF PROTO=TCP SPT=2362 DPT=555 WINDOW=5840 RES=0x00 SYN URGP
=0
Oct 11 21:03:23 gcux kernel:  DROP_INPUT IN=eth1 OUT= MAC=00:0c:29:75:b1:9d:00:0
c:29:5a:5e:b3:08:00 SRC=192.168.146.138 DST=192.168.146.45 LEN=60 TOS=0x00 PREC=
0x00 TTL=64 ID=42421 DF PROTO=TCP SPT=2363 DPT=556 WINDOW=5840 RES=0x00 SYN URGP
=0
Oct 11 21:03:23 gcux kernel:  DROP_INPUT IN=eth1 OUT= MAC=00:0c:29:75:b1:9d:00:0
c:29:5a:5e:b3:08:00 SRC=192.168.146.138 DST=192.168.146.45 LEN=60 TOS=0x00 PREC=
0x00 TTL=64 ID=32552 DF PROTO=TCP SPT=2364 DPT=557 WINDOW=5840 RES=0x00 SYN URGP
=0
Oct 11 21:03:23 gcux kernel:  DROP_INPUT IN=eth1 OUT= MAC=00:0c:29:75:b1:9d:00:0
c:29:5a:5e:b3:08:00 SRC=192.168.146.138 DST=192.168.146.45 LEN=60 TOS=0x00 PREC=
0x00 TTL=64 ID=61190 DF PROTO=TCP SPT=2365 DPT=558 WINDOW=5840 RES=0x00 SYN URGP
=0
Oct 11 21:03:23 gcux kernel:  DROP_INPUT IN=eth1 OUT= MAC=00:0c:29:75:b1:9d:00:0
c:29:5a:5e:b3:08:00 SRC=192.168.146.138 DST=192.168.146.45 LEN=60 TOS=0x00 PREC=
0x00 TTL=64 ID=15974 DF PROTO=TCP SPT=2366 DPT=559 WINDOW=5840 RES=0x00 SYN URGP
=0
root@gcux:/var/log# _
```

Nessus:

Nessus version 2.0.12 can be downloaded from http://ftp.nessus.org/nessus/nessus-2.0.12/src/. In order to get it up and running, download and compile the packages in the following order:

- Nessus-libraries
- Libnasl
- Nessus-core
- Nessus-plugins

Also download the MD5 file to validate the files integrity using md5sum. After validation, start the install with the following commands:

```
# tar xvzf nessus-libraries-2.0.12.tar.gz
# cd nessus-libraries
# ./configure
# make
# make install
```

For the libnasl package :

```
# tar xvzf libnasl-2.0.12.tar.gz
# cd libnasl
# ./configure
# make
# make install
```

For the core package, disable gtk to enable command line only:

```
# tar xvzf nessus-core-2.0.12.tar.gz
# cd nessus-core
# ./configure –disable-gtk
# make
# make install
```

And the plugins

```
# tar xvzf nessus-plugins-2.0.12.tar.gz
# cd nessus-plugins
# ./configure
# make
# make install
```

Next, create update the library and create the user :

```
# cd /sbin
# ./ldconfig
# cd /usr/local/sbin
# ./nessus-adduser
```

Follow the prompt and make sure to remember the username and password as they will be needed to run the scan. Next, create the SSL certificate as followed:

```
# ./nessus-mkcert
```

Follow the prompt as required. Download the latest pluggins at http://www.nessus.org/scripts.php and performs a manual updates by extracting the all-2.0.tar.gz to a temporary directory and then copying them over to the /usr/local/lib/nessus/plugins directory. Start Nessus in daemon mode:

```
# /usr/local/bin/nessusd –D
```

To run the client, change back to the home directory and do as followed:

```
# cd
# touch target
# echo “192.168.146.45” >> target
# nessus -c .nessusrc -T html_pie -q 127.0.0.1 1241 <username> <password>
target results
```

Descriptions of the specific command are in the man page. The basic explanation is to run nessus with the configuration file .nessusrc with output result html_pie on target and return the output to results.

After executing the above command, go through the prompt the SSL certificate and wait for the scan to complete. Once completed, kill the nessusd daemon and review the results through a web browser.

The results from the scan are as followed:

List of open ports :

- *ssh (22/tcp) (Security notes found)*
- *general/udp (Security notes found)*
- *general/tcp (Security warnings found)*

Information found on port ssh (22/tcp)

An ssh server is running on this port

Nessus ID : 10330

Information found on port ssh (22/tcp)

Remote SSH version : SSH-2.0-OpenSSH_3.8.1p1

Nessus ID : 10267

Information found on port ssh (22/tcp)

The remote SSH daemon supports the following versions of the SSH protocol :

. 1.99
. 2.0

SSHv2 host key fingerprint :
47:1b:2d:df:d5:6c:ea:04:8d:cc:0a:76:7e:c5:74:14

Nessus ID : 10881

Information found on port general/udp

For your information, here is the traceroute to 192.168.146.45 :
192.168.146.138
192.168.146.45

Nessus ID : 10287

Warning found on port general/tcp

The remote host might be vulnerable to a sequence number approximation
bug, which may allow an attacker to send spoofed RST packets to the remote
host and close established connections.

This may cause problems for some dedicated services (BGP, a VPN over TCP, etc...).

Solution : See http://www.securityfocus.com/bid/10183/solution/
Risk factor : Medium
CVE : CAN-2004-0230
BID : 10183
Other references : OSVDB:4030, IAVA:2004-A-0007
Nessus ID : 12213

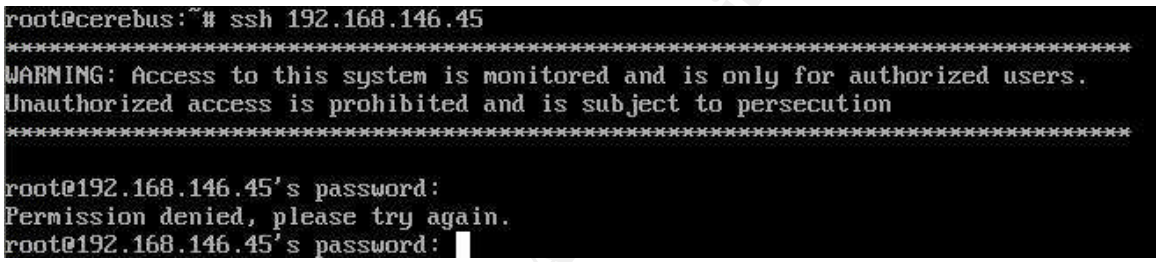
Nessus reported one “warning” and four “information items. The info on SSH

validated the configuration for SSH as it was setup to accept only protocol version 2. The only item of concern is the sequence number approximation bug. Based on further from http://www.securityfocus.com/bid/10183/info/, the bug has a severe affect on routing platforms.

Perhaps disabling tcp_window_scaling might address the issue. There seems to be no patch yet for kernel 2.4.27

Check SSH:

The first test for SSH involved an attempt to login as root. This should be disabled as per configuration setup. The following screenshot support the finding.



Notice the warning displayed as per banner setting in /etc/ssh/sshd_config. Now, test of a working account returned the following result:

As expected, the attempt was a success. Note the last login time displayed as proof that the PrintLastLog setting is functional. The last part to check is the log entry. Since ssh authentication are logged under the info facility, the entries are logged in the messages file. Looking at the messages file revealed the entries as expected:

```
root@gcux:/var/log# tail messages
Oct 11 21:43:24 gcux sshd[583]: Server listening on 0.0.0.0 port 22.
Oct 11 21:44:11 gcux sshd[584]: Accepted password for cpham from 192.168.146.138
 port 4796 ssh2
Oct 11 21:44:56 gcux sshd[598]: Accepted password for cpham from 127.0.0.1 port
32770 ssh2
Oct 11 21:48:08 gcux sshd[583]: Received signal 15; terminating.
Oct 11 21:48:10 gcux sshd[644]: Server listening on 0.0.0.0 port 22.
Oct 11 21:48:20 gcux sshd[646]: Failed password for cpham from 127.0.0.1 port 32
771 ssh2
Oct 11 21:48:23 gcux sshd[646]: Accepted password for cpham from 127.0.0.1 port
32771 ssh2
Oct 11 21:50:08 gcux sshd[659]: Failed password for root from 192.168.146.138 po
rt 4797 ssh2
Oct 11 21:51:14 gcux sshd[661]: Accepted password for cpham from 192.168.146.138
 port 4798 ssh2
Oct 11 21:51:28 gcux sshd[674]: Accepted password for cpham from 192.168.146.138
 port 4799 ssh2
root@gcux:/var/log# _
```

Last but not least, checking netstat for running process revealed that only SSH service is active on the system. The output as below:

```
root@gcux:/var/log# netstat -atp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 *:ssh                   *:*                     LISTEN
151/sshd
root@gcux:/var/log# _
```

Check Snort:

To ensure that snort is running as expected, log files and process table were checked. The output from checking the process table as followed:

```
root@gcux:/etc/rc.d# ps -w U snort
  PID TTY      STAT   TIME COMMAND
  744 ?        Ss     0:00 /home/snort/usr/local/bin/snort -c /home/snort/etc/sn
ort/snort.conf -A full -d -g snort -u snort -i eth0
root@gcux:/etc/rc.d# _
```

No surprise here. The entry corresponds to the command entered in the /etc/rc.d/rc.snortd file. The capture also confirmed the process is running in a chroot environment under the user snort.

To check that snortd is running in IDS mode, the content of /home/snort/var/log/snort was listed below:

```
root@gcux:/home/snort/var/log/snort/192.168.146.138# ls -l |more
total 320
-rw-------  1 snort snort  1615 Oct 11 21:31 ICMP_ADDR
-rw-------  1 snort snort  3004 Oct 11 21:32 ICMP_ECHO
-rw-------  1 snort snort   639 Oct 11 21:31 ICMP_ECHO_REPLY
-rw-------  1 snort snort  1650 Oct 11 21:31 ICMP_TIMESTAMP
-rw-------  1 snort snort  1360 Oct 11 21:32 TCP:10004-22
-rw-------  1 snort snort  1356 Oct 11 20:53 TCP:1481-161
-rw-------  1 snort snort  1344 Oct 11 20:53 TCP:1482-162
-rw-------  1 snort snort  1017 Oct 11 20:53 TCP:1505-161
-rw-------  1 snort snort  1008 Oct 11 20:53 TCP:1506-162
-rw-------  1 snort snort  1017 Oct 11 20:54 TCP:1529-161
-rw-------  1 snort snort  1008 Oct 11 20:54 TCP:1530-162
-rw-------  1 snort snort   284 Oct 11 21:30 TCP:24779-0
-rw-------  1 snort snort  1050 Oct 11 21:31 TCP:4699-0
-rw-------  1 snort snort  1053 Oct 11 21:31 TCP:4727-0
-rw-------  1 snort snort   411 Oct 11 21:42 TCP:4794-21
-rw-------  1 snort snort   283 Oct 11 21:30 TCP:7025-0
-rw-------  1 snort snort   283 Oct 11 21:30 TCP:9675-0
-rw-------  1 snort snort  5058 Oct 11 21:30 UDP:1027-161
-rw-------  1 snort snort  5040 Oct 11 21:30 UDP:1028-161
-rw-------  1 snort snort  2490 Oct 11 21:30 UDP:1029-161
-rw-------  1 snort snort  2490 Oct 11 21:30 UDP:1030-161
-rw-------  1 snort snort  2502 Oct 11 21:30 UDP:1031-161
root@gcux:/home/snort/var/log/snort/192.168.146.138# _
```

There is an abnormal amount of activities coming from 192.168.146.138. This is as expected since Nessus was sending attacks packets across this network segment. Checking the content of the alert revealed the following:

```
root@gcux:/home/snort/var/log/snort# tail -n 20 alert
[**] [1:519:6] TFTP parent directory [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
10/11-21:32:35.649753 192.168.146.138:1168 -> 192.168.146.45:69
UDP TTL:64 TOS:0x0 ID:22405 IpLen:20 DgmLen:50 DF
Len: 22
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-1209][Xref => http:/
/cve.mitre.org/cgi-bin/cvename.cgi?name=1999-0183][Xref => http://www.whitehats.
com/info/IDS137]

[**] [1:384:5] ICMP PING [**]
[Classification: Misc activity] [Priority: 3]
10/11-21:32:39.059714 192.168.146.138 -> 192.168.146.45
ICMP TTL:64 TOS:0x0 ID:55748 IpLen:20 DgmLen:29
Type:8  Code:0  ID:1   Seq:1  ECHO

[**] [1:553:7] POLICY FTP anonymous login attempt [**]
[Classification: Misc activity] [Priority: 3]
10/11-21:42:04.799792 192.168.146.138:4794 -> 192.168.146.1:21
TCP TTL:64 TOS:0x10 ID:32072 IpLen:20 DgmLen:62 DF
***AP*** Seq: 0x253049AB  Ack: 0x70E7DD02  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 1062251 1461712

root@gcux:/home/snort/var/log/snort# _
```

As shown above, the alerts are working and the finger is pointing to 192.168.146.138 as to where the malicious activities are coming from.

Check Authentication Log:

In the system setting, authentication log entry should be log to the /var/log/secure file. The entries should consist of all login and sudo attempts. The screen captured below validates the system configuration as warranted.

```
root@gcux:/var/log# tail secure
Oct 11 14:38:46 gcux su[246]: - tty1 cpham-root
Oct 11 14:38:49 gcux su[247]: + tty1 cpham-root
Oct 11 15:48:46 gcux login[210]: invalid password for 'cpham'  on 'tty1'
Oct 11 15:49:28 gcux su[226]: Authentication failed for root
Oct 11 15:49:28 gcux su[226]: - tty1 cpham-root
Oct 11 15:49:31 gcux su[227]: + tty1 cpham-root
Oct 11 15:58:24 gcux su[229]: + tty1 cpham-root
Oct 11 16:03:37 gcux su[225]: + tty1 cpham-root
Oct 11 16:06:24 gcux su[225]: + tty1 cpham-root
Oct 11 16:52:23 gcux su[251]: + tty1 cpham-root
root@gcux:/var/log# _
```

In addition, the sudo info is also captured separately in the sulog file as followed:

```
root@gcux:/var/log# tail sulog
SU 10/11 14:38 - tty1 cpham-root
SU 10/11 14:38 + tty1 cpham-root
SU 10/11 14:43 + tty1 cpham-root
SU 10/11 15:49 - tty1 cpham-root
SU 10/11 15:49 + tty1 cpham-root
SU 10/11 15:58 + tty1 cpham-root
SU 10/11 15:59 + tty1 cpham-root
SU 10/11 16:03 + tty1 cpham-root
SU 10/11 16:06 + tty1 cpham-root
SU 10/11 16:52 + tty1 cpham-root
root@gcux:/var/log# _
```

Entries between the two logs correspond accordingly and the sulog file should be viewed as a backup for alternative for the secure log file. Reviewing both logs on a regular basis is probably not necessary as the secure log file also duplicate the entries in the sulog file

Validate Software:

One of the identified risks associated with software that are downloadable on the Internet is the possibility of a trojan being introduced into the system. As previously demonstrated, this risk is somewhat mitigated through verification of the cryptographic signature or the one way hash associated with the download. Below is the typical output of checking md5 with rpm:

```
root@cerebus:~# rpm -K --nosignature snort-2.1.3-1.i386.rpm
snort-2.1.3-1.i386.rpm: sha1 md5 OK
root@cerebus:~#
```

Checking with cryptographic signature is done through gpg and the typical output is as followed:

```
root@cerebus:~# gpg --verify snort-2.2.0.tar.gz.asc snort-2.2.0.tar.gz
gpg: Signature made Wed 11 Aug 2004 03:57:23 PM EDT using DSA key ID 1946E4A1
gpg: Good signature from "Snort.org releases <releases@snort.org>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: F504 0CEF 9B3E C272 36D2  1EC7 9449 15EA 1946 E4A1
root@cerebus:~#
```

Observations:

Overall, the system is functioning as expected with a minor deficiency in the predictable sequence number as identified by Nessus. Mitigation of this problem might take some cycle and for the time being is an acceptable risk. The only possible attack vector is between the firewall and eth1 interface. It is not likely that an attacker will get this access from existing physical security measures. It is highly unlike that an attacker would go through all the trouble of bypassing the physical security measures just to DoS the IDS system. The cost and risk of discovery for carrying out this attack is very high for the attacker and hence has an extremely low probability of taking place.

Appendix

```
#!/bin/bash

# This script is designed to be run as:  /etc/dhclient-enter-hooks

################################################################
# Copyright (C) 1997 - 2003  Robert L. Ziegler
#
#  Permission to use, copy, modify, and distribute this software and
its
#  documentation for educational, research, private and non-profit
purposes,
#  without fee, and without a written agreement is hereby granted.
#  This software is provided as an example and basis for individual
firewall
#  development.  This software is provided without warranty.
#
#  Any material furnished by Robert L. Ziegler is furnished on an
#  "as is" basis.  He makes no warranties of any kind, either expressed
#  or implied as to any matter including, but not limited to, warranty
#  of fitness for a particular purpose, exclusivity or results obtained
#  from use of the material.
################################################################


# Load the FTP connection state helper module.
modprobe ip_conntrack_ftp

# Load the FTP NAT module.
modprobe ip_nat_ftp

# ------------------------------------------------------------------------
-------
#  Some definitions for easy maintenance.
#
#  EDIT THESE TO SUIT YOUR SYSTEM AND ISP.

# Toggle firewall service rules on/off

NAT_ENABLE="1"                    # 0=disable; 1=SNAT; 2=MASQUERADE
DHCP_CLIENT="1"
LAN_ACCESS="0"                     # ssh & ftp access from firewall
to lan

DNS_CACHE="0"                      # caching nameserver for LAN
ACCEPT_AUTH="0"
SMTP_SERVER="0"
SSH_SERVER="0"
FTP_SERVER="0"
WEB_SERVER="0"
NTP_CLIENT="0"
NTP_SERVER="0"                     # LAN server (not public)
MULTICAST_ENABLE="0"               # must subscribe to multicast
```

```
INTERNET="ethn"                         # network interface to the DMZ
LAN="ethn+1"                            # network interface to the LAN
LOOPBACK_INTERFACE="lo"                 # however your system names it

INTERNET_IPADDR="your.IP.address"       # gateway firewall - public IP

LAN_ADDR="some.address"                 # LAN IP address
LAN_ADDRESSES="address.range"           # LAN IP address range
LAN_NETWORK="network address"           # DMZ subnet base address
LAN_BROADCAST="some.address"            # DMZ broadcast address
LAN_NETMASK="255.255.255.0"

NAMESERVER_1="your.name.server"

MAIL_SERVER="any/0"                     # address of a remote mail gateway
POP_SERVER="your.pop.server"
NEWS_SERVER="your.news.server"

TIME_SERVER="your.time.server"

DHCP_SERVER="any/0"                    # some ISPs tell you the address

LOOPBACK="127.0.0.0/8"                  # reserved loopback address range
CLASS_A="10.0.0.0/8"                    # class A private networks
CLASS_B="172.16.0.0/12"                 # class B private networks
CLASS_C="192.168.0.0/16"                # class C private networks
CLASS_D_MULTICAST="224.0.0.0/4"         # class D multicast addresses
CLASS_E_RESERVED_NET="240.0.0.0/5"      # class E reserved addresses

BROADCAST_SRC="0.0.0.0"                 # broadcast source address
BROADCAST_DEST="255.255.255.255"        # broadcast destination address

UNPRIVPORTS="1024:65535"                # unprivileged port range

############################################################

# WARNING:

# The following section is written for dhclient.
# This section demonstrates what needs to be done
# to dynamically modify the IP address and name servers.

# See the "dhclient-script" man page
# and the "dhclient.conf" man page for details.

if [ x$reason = xBOUND ] || [ x$reason = xRENEW ] || [ x$reason =
xREBIND ]; then

    IPADDR=$new_ip_address

    # Some ISPs use more than one DHCP server.
    # In that case, you can leave DHCP_SERVER set to any/0,
    # or you can hard-code duplicate DHCP rules that
    # reference the specific server IP addresses.

    DHCP_SERVER=$new_dhcp_server_identifier
```

```
elif [ x$reason = xPREINIT ] || \
     [ x$reason = xEXPIRE ] || [ x$reason = xFAIL ] || [ x$reason =
xTIMEOUT ]; then

    IPADDR="any/0"
    DHCP_SERVER="any/0"

fi

################################################################

# Enable TCP SYN Cookie Protection
echo 1 > /proc/sys/net/ipv4/tcp_syncookies

# Enable broadcast echo Protection
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts

# Enable bad error message  Protection
echo 1 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses

# Drop Spoofed Packets coming in on an interface, which if replied to,
# would result in the reply going out a different interface.
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
    echo 1 > $f
done

# Disable ICMP Redirect Acceptance
for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
    echo 0 > $f
done

# Don't send Redirect Messages
for f in /proc/sys/net/ipv4/conf/*/send_redirects; do
    echo 0 > $f
done

# Disable Source Routed Packets
for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
    echo 0 > $f
done

# Log packets with impossible addresses.
# Includes Multicast src, Class E src/dst, Loopback src/dst,
# Zero net src/dst
for f in /proc/sys/net/ipv4/conf/*/log_martians; do
    echo 1 > $f
done

################################################################

# Remove any existing rules from all chains
iptables --flush
iptables -t nat --flush
iptables -t mangle --flush

# Set the default filter table policy
```

```
iptables --policy INPUT   DROP
iptables --policy OUTPUT  DROP
iptables --policy FORWARD DROP

# Remove any pre-existing user-defined chains
iptables --delete-chain
iptables -t nat --delete-chain
iptables -t mangle --delete-chain

# Create any user-defined chains
iptables -N tcp-state-flags
iptables -N log-tcp-state
iptables -N icmp-in
iptables -N icmp-out

# Unlimited traffic on the loopback interface
# Do immediately in case of firewall script errors!
iptables -A INPUT  -i $LOOPBACK_INTERFACE -j ACCEPT
iptables -A OUTPUT -o $LOOPBACK_INTERFACE -j ACCEPT

################################################################
# Enable Source NAT

if [ $NAT_ENABLE = "1" ]; then
    iptables -t nat -A POSTROUTING -o $INTERNET \
             -j SNAT --to-source $INTERNET_IPADDR
elif [ $NAT_ENABLE = "2" ]; then
    iptables -t nat -A POSTROUTING -o $INTERNET \
             -j MASQUERADE
fi

################################################################
# Stealth Scans and TCP State Flags

iptables -A log-tcp-state -m limit -j LOG --log-prefix "Bad TCP state
flags: "
iptables -A log-tcp-state -j DROP


# All of the bits are cleared
iptables -A tcp-state-flags -p tcp --tcp-flags ALL NONE -j log-tcp-
state

# SYN and FIN are both set
iptables -A tcp-state-flags -p tcp --tcp-flags SYN,FIN SYN,FIN -j log-
tcp-state

# SYN and RST are both set
iptables -A tcp-state-flags -p tcp --tcp-flags SYN,RST SYN,RST -j log-
tcp-state

# FIN and RST are both set
iptables -A tcp-state-flags -p tcp --tcp-flags FIN,RST FIN,RST -j log-
tcp-state

# FIN is the only bit set, without the expected accompanying ACK
```

```
iptables -A tcp-state-flags -p tcp --tcp-flags ACK,FIN FIN -j log-tcp-
state

# PSH is the only bit set, without the expected accompanying ACK
iptables -A tcp-state-flags -p tcp --tcp-flags ACK,PSH PSH -j log-tcp-
state

# URG is the only bit set, without the expected accompanying ACK
iptables -A tcp-state-flags -p tcp --tcp-flags ACK,URG URG -j log-tcp-
state

iptables -A INPUT   -p tcp -j tcp-state-flags
iptables -A OUTPUT  -p tcp -j tcp-state-flags
iptables -A FORWARD -p tcp -j tcp-state-flags

##############################################################
# Using Connection State to By-pass Rule Checking

iptables -A INPUT  -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

##############################################################
# Network Ghouls
# Deny access to jerks

# /etc/rc.d/rc.firewall.blocked contains a list of
# iptables -A INPUT -s address -j DROP
# rules to block from any access.

# Refuse any connection from problem sites
if [ -f /etc/rc.d/rc.firewall.blocked ]; then
    . /etc/rc.d/rc.firewall.blocked
fi

##############################################################
# Source Address Spoofing and Other Bad Addresses

# Refuse spoofed packets pretending to be from you
iptables -A INPUT -s $INTERNET_IPADDR -j DROP
iptables -A INPUT -s $LAN_ADDR -j DROP

iptables -A FORWARD -s $INTERNET_IPADDR -j DROP
iptables -A FORWARD -s $LAN_ADDR -j DROP

# Refuse packets claiming to be from a Class A private network
iptables -A INPUT  -i $INTERNET -s $CLASS_A  -j DROP

# Refuse packets claiming to be from a Class B private network
iptables -A INPUT  -i $INTERNET -s $CLASS_B  -j DROP

# Refuse packets claiming to be from a Class C private network
iptables -A INPUT  -i $INTERNET -s $CLASS_C  -j DROP

# Refuse Class D reserved IP addresses  (never source addresses)
iptables -A INPUT   -s $CLASS_D_MULTICAST -j DROP
iptables -A FORWARD -s $CLASS_D_MULTICAST -j DROP
```

```
# Refuse Class E reserved IP addresses
iptables -A INPUT   -i $INTERNET -s $CLASS_E_RESERVED_NET -j DROP
iptables -A FORWARD -i $INTERNET -s $CLASS_E_RESERVED_NET -j DROP

# Refuse packets claiming to be from the loopback interface (martian)
iptables -A INPUT    -s $LOOPBACK -j DROP
iptables -A FORWARD  -s $LOOPBACK -j DROP

# refuse other addresses defined as reserved by the IANA
# 0.*.*.*             - Can't be blocked for DHCP users.
# 169.254.0.0/16      - Link Local Networks
# 192.0.2.0/24        - TEST-NET

if [ $DHCP_CLIENT = "0" ]; then
    iptables -A INPUT -i $INTERNET -s 0.0.0.0/8 -j DROP
fi
iptables -A INPUT -i $INTERNET -s 169.254.0.0/16 -j DROP
iptables -A INPUT -i $INTERNET -s 192.0.2.0/24 -j DROP

iptables -A FORWARD -i $INTERNET -s 0.0.0.0/8 -j DROP
iptables -A FORWARD -i $INTERNET -s 169.254.0.0/16 -j DROP
iptables -A FORWARD -i $INTERNET -s 192.0.2.0/24 -j DROP

# Refuse broadcast DEST address SRC packets
iptables -A INPUT    -s $BROADCAST_DEST -j DROP
iptables -A FORWARD  -s $BROADCAST_DEST -j DROP

# Refuse broadcast SRC address DEST packets
iptables -A INPUT    -d $BROADCAST_SRC  -j DROP
iptables -A OUTPUT   -d $BROADCAST_SRC  -j DROP
iptables -A FORWARD  -d $BROADCAST_SRC  -j DROP

# Don't forward broadcasts in either direction
iptables -A FORWARD -m pkttype --pkt-type broadcast -j DROP

if [ "$DHCP_CLIENT" = "0" ]; then

    # Refuse broadcast packets from the Internet
    # Note: the LAN uses broadcasts internally.
    iptables -A INPUT -i $INTERNET -m pkttype --pkt-type broadcast -j
DROP
fi

if [ $MULTICAST_ENABLE = "1" ]; then
    # Multicast traffic is UDP traffic.
    iptables -A FORWARD -i $INTERNET -p udp -d $CLASS_D_MULTICAST \
             -m pkttype --pkt-type multicast -j ACCEPT
fi

############################################################

if [ $DHCP_CLIENT = "0" ]; then
    # Don't need to check my source addresses in future output rules:
    iptables -A OUTPUT -o $INTERNET -s ! $INTERNET_IPADDR -j DROP
fi
```

```
iptables -A OUTPUT -o $LAN -s ! $LAN_ADDR -j DROP

# Don't need to check LAN source address in future input/forward rules:
iptables -A INPUT    -i $LAN -s ! $LAN_ADDRESSES -j DROP
iptables -A FORWARD  -i $LAN -s ! $LAN_ADDRESSES -j DROP

#############################################################
# DNS Name Server
# NOTE: BIND 8.1+ uses an unprivileged source port for the query.

iptables -A OUTPUT -o $INTERNET -p udp \
        --sport $UNPRIVPORTS \
        -d $NAMESERVER_1 --dport 53 \
        -m state --state NEW -j ACCEPT

iptables -A OUTPUT -o $INTERNET -p tcp \
        --sport $UNPRIVPORTS \
        -d $NAMESERVER_1 --dport 53 \
        -m state --state NEW -j ACCEPT

# ------------------------------------------------------------

if [ $DNS_CACHE = "1" ]; then
    # DNS LAN clients to private caching server (53)

    iptables -A INPUT  -i $LAN -p udp \
             --sport $UNPRIVPORTS \
             -d $LAN_ADDR --dport 53 \
             -m state --state NEW -j ACCEPT

    iptables -A INPUT  -i $LAN -p tcp \
             --sport $UNPRIVPORTS \
             -d $LAN_ADDR --dport 53 \
             -m state --state NEW -j ACCEPT
else
    iptables -A FORWARD -o $INTERNET -p udp \
             --sport $UNPRIVPORTS \
             -d $NAMESERVER_1 --dport 53 \
             -m state --state NEW -j ACCEPT

    iptables -A FORWARD -o $INTERNET -p tcp \
             --sport $UNPRIVPORTS \
             -d $NAMESERVER_1 --dport 53 \
             -m state --state NEW -j ACCEPT
fi

#############################################################
# Filtering the AUTH User Identification Service (TCP Port 113)

# Outgoing Local Client Requests to Remote Servers

iptables -A OUTPUT -o $INTERNET -p tcp \
        --sport $UNPRIVPORTS \
        --dport 113 -m state --state NEW -j ACCEPT

#-------------------------------------------------------------
# Incoming Remote Client Requests to Local Servers
```

```
if [ "$ACCEPT_AUTH" = "1" ]; then
    iptables -A INPUT  -i $INTERNET -p tcp \
             --sport $UNPRIVPORTS \
             -d $INTERNET_IPADDR --dport 113 \
             -m state --state NEW -j ACCEPT

    iptables -A FORWARD  -i $INTERNET -o $LAN -p tcp \
             --sport $UNPRIVPORTS \
             -d $LAN_ADDRESSES --dport 113 \
             -m state --state NEW -j ACCEPT
else
    iptables -A INPUT -i $INTERNET -p tcp \
             --sport $UNPRIVPORTS \
             -d $INTERNET_IPADDR --dport 113 \
             -j REJECT --reject-with tcp-reset

    iptables -A FORWARD -i $INTERNET -o $LAN -p tcp \
             --sport $UNPRIVPORTS \
             -d $LAN_ADDRESSES --dport 113 \
             -j REJECT --reject-with tcp-reset
fi

# reject in all cases
iptables -A INPUT -i $LAN -p tcp \
         --dport 113 -j REJECT --reject-with tcp-reset

iptables -A OUTPUT -o $LAN -p tcp \
         --dport 113 -j REJECT --reject-with tcp-reset

################################################################
# Sending outgoing Mail

if [ $SMTP_SERVER = "1" ]; then
    iptables -A OUTPUT -o $INTERNET -p tcp \
             --sport $UNPRIVPORTS --dport 25 \
             -m state --state NEW -j ACCEPT
else
    iptables -A OUTPUT -o $INTERNET -p tcp \
             --sport $UNPRIVPORTS \
             -d $MAIL_SERVER --dport 25 \
             -m state --state NEW -j ACCEPT
fi

################################################################
# Receiving/Relaying Mail from the LAN as the local Mail Server (TCP
Port 25)

if [ $SMTP_SERVER = "1" ]; then
    iptables -A INPUT -i $LAN -p tcp \
             --sport $UNPRIVPORTS \
             -d $LAN_ADDR --dport 25 \
             -m state --state NEW -j ACCEPT
else
    iptables -A FORWARD -o $INTERNET -p tcp \
             --sport $UNPRIVPORTS \
             -d $MAIL_SERVER --dport 25 \
```

```
            -m state --state NEW -j ACCEPT
fi

#############################################################
# Receiving Mail from the Internet as the local Mail Server (TCP Port
25)

if [ $SMTP_SERVER = "1" ]; then
    iptables -A INPUT -i $INTERNET -p tcp \
            --sport $UNPRIVPORTS -d $INTERNET_IPADDR --dport 25 \
            -m state --state NEW -j ACCEPT
fi

#############################################################
# Retrieving Mail as a POP Client

if [ $SMTP_SERVER = "1" ]; then
    iptables -A INPUT  -i $LAN -p tcp \
            --sport $UNPRIVPORTS \
            -d $LAN_ADDR --dport 110 \
            -m state --state NEW -j ACCEPT
else
    iptables -A FORWARD  -i $LAN -o $INTERNET -p tcp \
            -s $LAN_ADDRESSES --sport $UNPRIVPORTS \
            -d $POP_SERVER --dport 110 \
            -m state --state NEW -j ACCEPT

    iptables -A OUTPUT -o $INTERNET -p tcp  \
            -s $INTERNET_IPADDR --sport $UNPRIVPORTS \
            -d $POP_SERVER --dport 110 \
            -m state --state NEW -j ACCEPT
fi

#############################################################
# Accessing Usenet News Services (TCP NNTP Port 119)

iptables -A FORWARD -i $LAN -o $INTERNET -p tcp \
        --sport $UNPRIVPORTS -d $NEWS_SERVER --dport 119 \
        -m state --state NEW -j ACCEPT

#############################################################
# SSH (TCP Port 22)

if [ $SSH_SERVER = "1" ]; then
    iptables -A INPUT  -i $LAN -p tcp \
            --sport $UNPRIVPORTS -d $LAN_ADDR --dport 22 \
            -m state --state NEW -j ACCEPT
fi

if [ "$LAN_ACCESS" = "1" ]; then
    iptables -A OUTPUT  -o $LAN -p tcp \
            --sport $UNPRIVPORTS -d $LAN_ADDRESSES --dport 22 \
            -m state --state NEW -j ACCEPT
fi

#############################################################
# ftp (TCP Ports 21, 20)
```

```
# The secondary rules are not necessary if the ip_conntrack_ftp is
used.

# LAN to remote servers
iptables -A FORWARD -o $INTERNET -p tcp \
        --sport $UNPRIVPORTS --dport 21 \
        -m state --state NEW -j ACCEPT

# firewall to remote servers
iptables -A OUTPUT -o $INTERNET -p tcp  \
        --sport $UNPRIVPORTS --dport 21 \
        -m state --state NEW -j ACCEPT

if [ $FTP_SERVER = "1" ]; then
    # LAN to firewall server
    iptables -A INPUT -i $LAN -p tcp \
            --sport $UNPRIVPORTS -d $LAN_ADDR --dport 21 \
            -m state --state NEW -j ACCEPT
fi

if [ "$LAN_ACCESS" = "1" ]; then
    # firewall to LAN servers
    iptables -A OUTPUT -o $LAN -p tcp  \
            --sport $UNPRIVPORTS -d $LAN_ADDRESSES --dport 21 \
            -m state --state NEW -j ACCEPT
fi

################################################################
# HTTP Web Traffic (TCP Port 80)

iptables -A FORWARD  -o $INTERNET -p tcp \
        --sport  $UNPRIVPORTS --dport 80 \
        -m state --state NEW -j ACCEPT

iptables -A OUTPUT  -o $INTERNET -p tcp \
        --sport  $UNPRIVPORTS --dport 80 \
        -m state --state NEW -j ACCEPT

if [ $WEB_SERVER = "1" ]; then
    iptables -A INPUT  -i $INTERNET -p tcp \
            --sport  $UNPRIVPORTS -d $INTERNET_IPADDR --dport 80 \
            -m state --state NEW -j ACCEPT

    iptables -A INPUT  -i $LAN -p tcp \
            --sport  $UNPRIVPORTS -d $LAN_ADDR --dport 80 \
            -m state --state NEW -j ACCEPT
fi

################################################################
# SSL Web Traffic (TCP Port 443)

iptables -A OUTPUT  -o $INTERNET -p tcp \
        --sport  $UNPRIVPORTS --dport 443 \
        -m state --state NEW -j ACCEPT

iptables -A FORWARD  -i $LAN -o $INTERNET -p tcp \
```

```
        --sport  $UNPRIVPORTS --dport 443 \
        -m state --state NEW -j ACCEPT

################################################################
# whois (TCP Port 43)

iptables -A OUTPUT -o $INTERNET -p tcp  \
        --sport $UNPRIVPORTS --dport 43 \
        -m state --state NEW -j ACCEPT

iptables -A FORWARD  -i $LAN -o $INTERNET -p tcp \
        --sport $UNPRIVPORTS --dport 43 \
        -m state --state NEW -j ACCEPT

################################################################
# Accessing Network Time Server (UDP 123)
# Note: some clients and servers use source port 123
# when querying a remote server on destination port 123.
# Note: some (local) clients broadcast the request.

if [ $NTP_CLIENT = "1" ]; then
    iptables -A OUTPUT  -o $INTERNET -p udp \
            --sport $UNPRIVPORTS -d $TIME_SERVER --dport 123 \
            -m state --state NEW -j ACCEPT

    iptables -A OUTPUT  -o $INTERNET -p udp \
            --sport 123 -d $TIME_SERVER --dport 123 \
            -m state --state NEW -j ACCEPT
fi

if [ $NTP_SERVER = "1" ]; then
    # Note: Some LAN devices might broadcast the request to
$BROADCAST_DEST
    iptables -A INPUT  -i $LAN -p udp \
            --sport $UNPRIVPORTS --dport 123 \
            -m state --state NEW -j ACCEPT

    iptables -A INPUT  -i $LAN -p udp \
            --sport 123 --dport 123 \
            -m state --state NEW -j ACCEPT
fi

################################################################
# DHCP client (67, 68)

# Some broadcast packets are explicitly ignored by the firewall.
# Others are dopped by the default policy.
# DHCP tests must precede broadcast-related rules, as DHCP relies
# on broadcast traffic initially.

if [ "$DHCP_CLIENT" = "1" ]; then
    # Initialization or rebinding: No lease or Lease time expired.

    iptables -A OUTPUT -o $INTERNET -p udp \
            -s $BROADCAST_SRC --sport 68 \
            -d $BROADCAST_DEST --dport 67 -j ACCEPT
```

```
    # Incoming DHCPOFFER from available DHCP servers

    iptables -A INPUT  -i $INTERNET -p udp \
             -s $BROADCAST_SRC --sport 67 \
             -d $BROADCAST_DEST --dport 68 -j ACCEPT

    # Fall back to initialization
    # The client knows its server, but has either lost its lease,
    # or else needs to reconfirm the IP address after rebooting.

    iptables -A OUTPUT -o $INTERNET -p udp \
             -s $BROADCAST_SRC --sport 68 \
             -d $DHCP_SERVER --dport 67 -j ACCEPT

    iptables -A INPUT  -i $INTERNET -p udp \
             -s $DHCP_SERVER --sport 67 \
             -d $BROADCAST_DEST --dport 68 -j ACCEPT

    # As a result of the above, we're supposed to change our IP
    # address with this message, which is addressed to our new
    # address before the dhcp client has received the update.
    # Depending on the server implementation, the destination address
    # can be the new IP address, the subnet address, or the limited
    # broadcast address.

    # If the network subnet address is used as the destination,
    # the next rule must allow incoming packets destined to the
    # subnet address, and the rule must preceed any general rules
    # that block such incoming broadcast packets.

    iptables -A INPUT  -i $INTERNET -p udp \
             -s $DHCP_SERVER --sport 67 \
             --dport 68 -j ACCEPT

    # Lease renewal

    iptables -A OUTPUT -o $INTERNET -p udp \
             -s $INTERNET_IPADDR --sport 68 \
             -d $DHCP_SERVER --dport 67 -j ACCEPT

    iptables -A INPUT  -i $INTERNET -p udp \
             -s $DHCP_SERVER --sport 67 \
             -d $INTERNET_IPADDR --dport 68 -j ACCEPT
fi

#############################################################
# ICMP Control and Status Messages

iptables -A icmp-in -p icmp \
         --icmp-type source-quench -j ACCEPT

iptables -A icmp-in -p icmp \
         --icmp-type parameter-problem -j ACCEPT

iptables -A icmp-in -p icmp \
         --icmp-type destination-unreachable -j ACCEPT
```

```
# Intermediate traceroute responses
iptables -A icmp-in -i $INTERNET -p icmp \
        --icmp-type time-exceeded -j ACCEPT

# ------------------------------------------------------------

iptables -A icmp-out -p icmp \
        --icmp-type source-quench -j ACCEPT

iptables -A icmp-out -p icmp \
        --icmp-type parameter-problem -j ACCEPT

iptables -A icmp-out -p icmp \
        --icmp-type fragmentation-needed -j ACCEPT

# allow outgoing pings to anywhere
iptables -A icmp-out -p icmp \
        --icmp-type echo-request \
        -m state --state NEW -j ACCEPT

iptables -A icmp-out -o $LAN -p icmp \
        --icmp-type destination-unreachable -j ACCEPT

# Don¹t log dropped outgoing ICMP error messages
iptables -A icmp-out -o $INTERNET  -p icmp \
        --icmp-type destination-unreachable -j DROP

# ------------------------------------------------------------

iptables -A INPUT -p icmp -j icmp-in

# allow incoming pings from trusted hosts
iptables -A INPUT  -i $LAN -p icmp \
        --icmp-type echo-request \
        -m state --state NEW -j ACCEPT

iptables -A OUTPUT -p icmp -j icmp-out
iptables -A FORWARD -o $LAN -p icmp -j icmp-in
iptables -A FORWARD -o $INTERNET -p icmp -j icmp-out

#############################################################
# Logging Dropped Packets

# A Rule without a "-j LOG" target increments the packet counters
# without writing a log message.
# This is convenient for people who want to log dropped packets but
# don't want their log file filling up due to the most common packet
# types.

iptables -A OUTPUT  -j LOG
iptables -A FORWARD -j LOG
iptables -A INPUT   -i $LAN -j LOG

iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport 0:20
-j LOG
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport 21:23
```

```
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport 24 -j
LOG
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport 25
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport 26:78
-j LOG
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport
81:112 -j LOG
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport
114:134 -j LOG
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport
135:139
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport
140:442 -j LOG
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport 443
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport 444 -
j LOG
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport 445
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport
446:553 -j LOG
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport 554
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport
555:900 -j LOG
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport 901
iptables -A INPUT -i $INTERNET -p tcp -d $INTERNET_IPADDR --dport
902:1023 -j LOG

# UDP
# not echo old_pcAnywhere sunrpc snmp mountd pcAnywhere BackOrifice

iptables -A INPUT -i $INTERNET -p udp -d $INTERNET_IPADDR --dport 0:21
-j LOG
iptables -A INPUT -i $INTERNET -p udp -d $INTERNET_IPADDR --dport 22

iptables -A INPUT -i $INTERNET -p udp -d $INTERNET_IPADDR --dport 23:79
-j LOG
iptables -A INPUT -i $INTERNET -p udp -d $INTERNET_IPADDR --dport 80
iptables -A INPUT -i $INTERNET -p udp -d $INTERNET_IPADDR --dport
81:134 -j LOG
iptables -A INPUT -i $INTERNET -p udp -d $INTERNET_IPADDR --dport
135:139
iptables -A INPUT -i $INTERNET -p udp -d $INTERNET_IPADDR --dport
140:1023 -j LOG

# ICMP
iptables -A INPUT -i $INTERNET -p icmp --icmp-type ! echo-request -j
LOG
```

References

1) Fyodor "Nmap Stealth Port Scanner 3.70" August 2004
URL: http://www.insecure.org/nmap/ (Oct 12, 2004)

2) Harriss, Jeff. "GCUX Practical Assignment." GIAC Certified UNIX Administrator (GCUX #174). December 1, 2002. URL: http://www.giac.org/practical/GCUX/Jeff_Harriss_GCUX.pdf (Oct 12, 2004)

3) Hartsuijker, Maarten. "GCUX Practical Assignment." GIAC Certified UNIX Administrator (GCUX #131). October 26, 2001. URL: http://www.giac.org/practical/Maarten_Hartsuijker_GCUX.doc (Oct 12, 2004)

4) IPtables 1.2.10. "NetFilter/IPtables" June 2004. URL: http://www.netfilter.org/ (Oct 12, 2004)

5) Kernel 2.4.27. "The Linux Kernel Archive". August 2004. URL: http://www.kernel.org/ (Oct 12, 2004)

6) Murdoch, Don. "GCUX Practical Assignment." GIAC Certified UNIX Administrator (GCUX #225). February 28, 2004. URL: http://www.giac.org/practical/GCUX/Don_Murdoch_GCUX.pdf (Oct 12, 2004)

7) Nall, Andrew. "GCUX Practical Assignment." GIAC Certified UNIX Administrator (GCUX #147). May 23, 2002. URL: http://www.giac.org/practical/Nall_Andrew_GCUX.doc (Oct 12, 2004)

8) Nessus 2.0.12. "The Nessus Project" July 2004. URL: http://www.nessus.org/ (Oct 12, 2004)

9) OpenSSH 3.8.1p1. "Portable OpenSSH" March 2004. URL: http://www.openssh.com/portable.html (Oct 12, 2004)

10) SecurityFocus. 1999. URL: http://www.securityfocus.com/ (Oct 12, 2004)

11) Seth DeVries, Peter. "GCUX Practical Assignment." GIAC Certified UNIX Administrator (GCUX #117). October 31, 2001. URL: http://www.giac.org/practical/Seth_DeVries_GCUX.doc (Oct 12, 2004)

12) Slackware 10.0. "The Slackware Linux Project" June 2004. URL: http://www.slackware.com/ (Oct 12, 2004)

13) Snort. "The Open Source Network Intrusion Detection System 2.2.0" August 2004. URL: http://www.snort.org (Oct 12, 2004)

14) Tipton, Harold F.; Krause, Micki. CRC Press LLC. “Information Security Management Handbook”. 2002.

15) Ziegler, Robert L. “Linux LAN & Internet Firewall Security FAQ”. URL: http://www.linux-firewall-tools.com/linux/faq/index3.html (Oct 12, 2004)