Installation of a Red Hat 9.0 server with DNS services, emphasising security

Mark Chandler

GSEC Practical Assignment 1.4b
Option 2 – Case Study in Information Security

Installation of a Red Hat 9.0 server with DNS services, emphasising security

Mark E. Chandler

14 July 2003

Version 1.0.0

Table of Contents

1 Abstract

This paper seeks to provide an edited account of the work done by the author to create a minimal-install, primary DNS server based on a Linux platform. The document includes some discussion as to why certain decisions were made and the reasons for the method used to build the system.

There is a preliminary summary of this document, which outlines the rest of the documents content. Each section, that details the instructions for building the system, has information and discussion about the actions and decisions taken that are relevant to that section. However, the document is also designed to be a set of build instructions that can be followed to create a simple DNS server with security as a focus.

2 Before, During and After

2.1 Previous environment and reasons for change

The motivation behind building a new primary DNS server was simply that the previous one was out-of-date. The older system was running BIND 8 on Red Hat Linux 6.2. It was also running on Pentium II based hardware. Due to several vulnerabilities already known on BIND 8 and the recommendation from the Internet Software Consortium to upgrade to version 9, I decided that it was time to undertake a complete rebuild of the system.

The DNS server was considered to be critical to the IT infrastructure of the company. It controls the DNS records for the company, its subsidiaries, and several clients. It was also considered to be more vulnerable than most systems as it existed in the DMZ. It was deemed necessary that the server be constructed using the latest techniques available in securing the system from unauthorised access and abuse.

2.2 Decisions taken for implementing upgrade

Several principles and techniques learned from the SANS GSEC training, were used in generating the instructions for building the server. To begin with the server was built with what was considered to be the minimal set of operating system components required to support the application (BIND). The fewer elements used, the easier it would be to test and fewer security weaknesses would be introduced. Also, the during the build and test process, services or components that were not understood, were removed or deactivated in order to see what effect this would have.

The operating system chosen was Red Hat 9. This, at the time of the project, was the latest version of Red Hat Linux. It contained several upgraded components, not least of which was a kernel that included “capability” restrictions that could be used to tighten the system to very high levels. Although there are often many bugs to be

ironed out from freshly released software, it was considered that the project would run for a sufficiently long period, that many of these bugs would surface and be fixed before a build was finalised.

Several components were built from source in order to make them as efficient and secure as possible. OpenSSL, OpenSSH, BIND and the Linux kernel were chosen as components that required compilation instead of using packages provided by Red Hat. Not all components were selected for compilation, as a certain amount of simplicity was required in the build process. This is so that in the event of a catastrophic failure, the instructions could be used to get the system operational without a great deal of technical knowledge required. Also, some packages did not seem to install well from source, so the precompiled versions were used instead.

2.2.1 System considerations

The system needed to satisfy three requirements:

a) could perform DNS services

b) allowed authorised users to login and maintain DNS information

c) was secure to a reasonable level

The first requirement we satisfied by installing Red Hat 9 and building BIND 9.2.2 for that platform. I chose Red Hat as it was a well-known operating system, with regular updates, that allowed the ability to modify the kernel for security purposes. BIND 9.2.2 was, at the time of writing, the latest version of the ISC's DNS software and recommended by them as the level to upgrade to. As becomes evident from some of the decisions that were already made, several components are needed to support the construction and configuration of others. We intend to remove components that are not needed for day-to-day running of the system, once it is operational.

The second requirement could be satisfied, by allowing people to login at the console as “root” and performing maintenance. This would eliminate the need for other user profiles and more complex permissions. However, as it was discovered during the course of this project, system design is often a compromise between three things: simplicity, security, and flexibility. Disallowing other users would have made the system simpler and more secure, however it would have made it more difficult for people to use it. I decided to allow remote logins for ease of administration. Therefore, in order to allow remote logins that were secure, I needed to implement OpenSSH.

The third requirement required significant investigation. Each part of the installation process was analysed and techniques for securing them were researched. However, as already noted, ultimate security can be to the detriment of ease of use. It is well known in the software-engineering world that a system that is inflexible or difficult to use will often be ignored or discarded even though it may have superior features. Securing the system to a “reasonable level” was required as opposed to securing without regard for its administration.

2.2.2 Outline of build

The basic process involved was the following:

- install what we need

- remove anything we don't need
- secure what is left
- test the results
- reconfigure based on tests
- repeat the last two steps until requirements are satisfied

The server is a generic Pentium III-based system that contains 128MB of RAM, a 4.3GB internal IDE disk, a 10/100Mb Ethernet card, a CD-ROM, and 1.44MB floppy disk drive. It also has a keyboard, monitor and mouse. The server exists in a subnet of private IP's and is connected to the Internet via a firewall.

The following sections detail the method used to build the system and the decisions taken at each step.

3 Basic Red Hat 9.0 Install

3.1 Design Decisions

The previous system had been installed in a quite minimalist fashion already. X-Windows was never installed and, on a bundle level, only basic services and BIND were installed. However, this still left several packages lurking on the system. Obviously dangerous services, such as Telnet and FTP were not installed. NFS was deactivated, etc. However, to be truly comfortable with the security level of this new system, I wanted to know what was being installed at a package level. This is one of the distinct advantages of OS's like Red Hat Linux: each of the OS components is a separate package that can be investigated and pulled apart to see how it works – if you want. It was impractical to do this with *every* package, but the option was there. This is rarely an option with proprietary platforms.

At installation time, I deselected all default packages that weren't mandatory. I then manually inspected the list of optional packages and installed those that I considered relevant to the build requirements, e.g. crontab, iptables, logrotate, logwatch, sendmail, tripwire. These components required other components to support them. Some of these dependencies have been listed in the package list that's included in this chapter.

A few packages were included with the intention of using their functionality later, although this was determined impractical. For example, the "acl" package is installed, but ACL's are not fully implemented in Red Hat Linux. Also, "stunnel" is included as I hoped to create secure syslogging to a separate server. This could not be achieved within the timeframe of the project.

The previous server had only two filesystems. This is practical for a single service system, however there are advantages in carving up the disk space as different filesystems so that we can apply different mount options to each one. I applied this to the disk layout portion of the install. I explain the format used in more detail later on.

3.2 Method

At the outset I made sure the system being built was not physically connected to a network, until the last step was completed. This way, maximum security was achieved at all times. I never left the server unattended and logged in. I created the following set of instructions to install the base operating system.

Boot from Red Hat 9 Install CD 1

Hit <enter> when initial options are shown, to run fresh install.

Hit OK to test the CD when the option is offered

At “Welcome” hit enter.

Choose English Language

Choose U.S. English keyboard

Choose "2 Button Mouse (PS/2)" + "Emulate 3 Buttons"

Installation Type: Custom

Choose "Manually Partition with Disk Druid"

Remove existing partitions if they exist.

Basic Disk Layout on 4.3 GB disk. Ignore the mount options to the right of the filesystem size.

<swap>	256	
/boot	128	n, nodev, nosuid, ro
/var	512	n, nodev, noexec, nosuid, rw
/chroot/named/var	128	n, nodev, noexec, nosuid, rw
/	<rest>	n, ro

It's important to create separate filesystems so we can mount them with different options.

Use GRUB; Install Boot Loader on MBR, Default boot image

Use a GRUB password

Click Edit

DHCP:no, Activate on boot:yes

Supply IP settings

Gateway	<gateway address>
Primary DNS	<this server's IP>
Secondary DNS	<secondary's IP>

Hostname should be "ns1"

Firewall Configuration: High

Customize: Trusted devices = <none>

Allow incoming= SSH

Other ports = 53:tcp,53:udp

Choose "English (USA)" as the language

Time Zone "*Your Timezone*" System Clock uses UTC = no

Enter root password & enter a new user

Enable MD5 + Shadow passwords

leave NIS, LDAP, Kerberos 5, and SMB

Select individual packages and click Next

Select “all packages” in the tree-view and then click unselect all in group

Select the following packages

(comments to the right of a “<=” can be ignored):

acl-2.2.3-1

anacron-2.3-25

apmd-3.0.2-18

at-3.1.8-33

attr-2.2.0-1

bc-1.06-12

bzip2-1.0.2-8

crontabs-1.10-5

cyrus-sasl-plain-2.1.10-4

dosfstools-2.8-6 <= mkbootdisk

ethtool-1.6-5

gnupg-1.2.1-3

groff-1.18.1-20 <= man

hesiod-3.0.2-26 <= sendmail

iptables-1.2.7a-2

krbafs-1.1.1-9 <= pam_krb5

libcap-1.10-15 <= ntp

libstdc++-3.2.2-5

libtool-libs-1.4.3-5

libwvstreams-3.70-8

libxml2-2.5.4-1

libxslt-1.0.27-3

lockdev-1.0.0-23

logrotate-3.6.8-1		
logwatch-4.3.1-2		
lsof-4.63-4		
m4-1.4.1-13	<=	sendmail
mailx-8.1.1-28		
make-3.79.1-17	<=	quota
man-1.5k-6		
man-pages-1.53-3		
mkbootdisk-1.5.1-1		
mtr-0.52-2		
netconfig-0.8.14-2		
ntp-4.1.2-0.rc1.2		
pciutils-2.1.10-7		
perl-5.8.0-88	<=	gnupg
perl-Filter-1.29-3	<=	perl
procmail-3.22-9	<=	sendmail
quota-3.06-9	<=	detect capacity changes
sendmail-8.12.8-4	<=	for mailing alerts
sendmail-cf-8.12.8-4	<=	for configuring sendmail
setuptool-1.12-1		
stunnel-4.04-3	<=	possible SSL syslogging
syslinux-2.00-4	<=	mkbootdisk
tcp_wrappers-7.6-34	<=	stunnel
traceroute-1.4a12-9		
tripwire-2.3.1	<=	intrusion detection
unzip-5.50-7		
vixie-cron-3.0.1-74	<=	anacron
zip-2.3-16		

Install Size should be around 429MB

Skip boot disk creation (if you want)

4 Additional Software Installation

4.1 Design Decisions

As already mentioned, some packages are best installed by retrieving their source and compiling them on the target system. This not only can give you a more efficient application, but sometimes a more secure application, as certain options that can benefit are only available at compile time. Also, I like getting as much in-depth and hands-on experience with the applications as possible. By understanding the software, I can better understand how to defend it.

Obviously, to build the software, we need compiler tools installed on the system. I deliberately did not include any of these tools at OS install time, as I wanted to make it easier to identify what is required. This means that we know what we can remove when we're finished with them. It's also useful if we need to put them back on again to compile updated source.

4.2 Install Filesets required for compilation

As Red Hat packages are installed manually, RPM will complain about the validity of the key used to sign the packages. This is a minor annoyance, but as security is of major concern here, I decided it needed fixing. The Red Hat 9 online manuals [26] tell us that the Red Hat key needs to be imported before packages signed with that key are automatically authenticated.

import Red Hat's GPG key

```
rpm --import /usr/share/doc/Red Hat-release-9/RPM-GPG-KEY
```

I knew from experience that I needed at least binutils, gcc, and make in order to start compiling. GCC version 3 is now a generic compiler interface that different languages can be accessed from. So, I also needed to add in the gcc-c++ package. RPM reminded me of other dependencies and I discovered that in order to use "make menuconfig" for the kernel configuration, I also needed the ncurses development package. Eventually the following list of packages evolved.

install RPMS (in this order, due to dependencies) using the following format

```
mount /mnt/cdrom
rpm --install /mnt/cdrom/Red Hat/RPMS/<rpm-file>
```

From Red Hat 9.0 install CD 1

binutils-2.13.90.0.18-9.i386.rpm

cpp-3.2.2-5.i386.rpm

glibc-kernheaders-2.4-8.10.i386.rpm

glibc-devel-2.3.2-11.9.i386.rpm

diffutils-2.8.1-6.rpm (required for "make test of" OpenSSL)

From Red Hat 9.0 install CD 2

zlib-devel-1.1.4-8.i386.rpm

gcc-3.2.2-5.i386.rpm

libstdc++-devel-3.2.2-5.rpm

gcc-c++-3.2.2-5.rpm Note: may need a force reinstall

ncurses-devel-5.3-4 (required for "make menuconfig")

During the course of this project patches were released for OpenSSL before the current version was available. Initially, I was unable to apply the patches, as I had not installed the following patch tools:

patch-2.5.4-14

patchutils-0.2.14-3

These will be necessary to patch and recompile, if patches are released to any of the software compiled from source. It's in our best interests to know how this can be done in case vulnerability is discovered and a fix is only available via a patch.

4.3 Freshen Filesets

Several packages were updated during the course of the project. I thought it best to stay current with the packages that had been selected for the system. I downloaded all the updates from a local mirror onto a separate system and then wrote them to a CD. I then put this CD into the target system, and used the following command to apply them.

```
mount /mnt/cdrom
rpm --freshen /mnt/cdrom/*
```

The list of packages that I updated using RPM's is in Appendix E. More of the packages that I use in the initial installation may have been updated since the writing of this document.

5 Upgrading Kernel

5.1 Design decisions

The main reason why I decided to compile the kernel, instead of using an RPM delivered one, was to guard against malicious kernel loadable modules. Even though you can now turn off module loading without having to compile your kernel, I still decided to investigate this option with the intention of understanding the kernel better. I found a paper from OpenNA [5] which gave me much of the information I needed. As the kernel had been updated since the writing of that paper, I also read

some of the documentation that comes with the kernel source.

As with the installation of the operating system, my plan for the configuration of the kernel was to enable only the bare-minimum set of features required, to have the system running and secure.

5.2 Prepare for compilation

I obtained a baseline of the currently used modules. This is useful when determining what support is required to be built into the kernel [5].

```
lsmod > /tmp/modules
```

You can read this file to determine what devices have been detected during the installation. These are the only devices that you need to enable in the kernel at configuration time. Of course, this can mean long term problems if, for example, the network card fails and you replace it with a different brand or model. The kernel won't support it and it may not work. I get around this problem in my situation by having a duplicate system on standby. While the duplicate is running, I can build a new system that supports the different hardware.

I downloaded the latest kernel source package from Red Hat (2.4.20-18.9 at time of writing), burned it to CD and copied it onto the target system. I then used the following commands to begin the preparation for compilation.

```
rpm --install kernel-source-2.4.20-18.9.i386.rpm
cd /usr/src/linux-2.4
make clean
make mrproper
```

In some of the mailing lists that I encountered during my research, there seemed to be some debate as to whether it was valid to compile new kernels under /usr/src. Although this point is of minor relevance within the scope of this paper I wished to raise it so as not to blindly perpetuate only one opinion in this matter. The issue may only be important for those working on development kernels.

I decided to use the kernel-source packages from Red Hat as I had trouble using the source found at kernel.org, and I was never completely sure that I had the latest fixes incorporated into it. Even though there was a certain amount of lag between patches to the kernel and when Red Hat back-ported them into their versions, I considered this acceptable.

5.3 Set kernel options for behavior and support

I recompiled the kernel with all modules as static & no module loading (KLM) allowed

Setting the compile options was done using the following command:

```
make menuconfig
```

See Appendix D for a table of the options I used. These are not suitable for all

systems.

If you are attempting this process for yourself, then be careful about adding SCSI support or not, and pay attention to the network card being used.

5.4 Compile and install new kernel

I used instructions from the The Linux Documentation Project's "Kernel-HOWTO" [4] and OpenNA [5].to compile new kernel.

```
make dep
make bzImage

cp .config /boot/config-2.4.20-18.9
cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.20-18.9
cp System.map /boot/System-map-2.4.20-18.9
cd /boot
ln -sf System.map-2.4.20-18.9 System.map
```

I found out what partition "root" was on using "df" and then added the following to /boot/grub/grub.conf to provide a second boot option that I used to start the system with the new kernel.

```
title Red Hat Linux (2.4.20-18.9)
     root (hd0,0)
     kernel /vmlinuz-2.4.20-18.9 ro root=/dev/hda6 nofirewire
nomodules nousb nosound
```

From analysing the boot scripts, I realised that I could add several options to the kernel statement in the boot loader that would disable other features (see above). As those features are already disabled in the kernel it's not very important, but it does mean the boot process is cleaner and quieter. It's also useful to know in case I have a need to turn these features off on a kernel I can't recompile.

5.5 Test New Kernel

Reboot system

```
shutdown -r now
```

At the Grub bootloader menu, I selected the second option to boot with the new kernel

I encountered some problems with the boot process at this stage. The rc.sysinit script contains a section for starting a process that looks for a keypress. For some reason, that I don't fully understand, with the new kernel the process is invisible to the script later on, so it cannot be killed. This hangs the script.

I knew that I could just press “C” to continue, but I created a better work-around that can be found in section 18.1 .

Apart from this, the system seemed to boot successfully with the new kernel.

6 IPtables (firewalling)

6.1 Design Decisions

The previous DNS server had no firewall. It relied on a hardware-based firewall within the communications infrastructure. This was a design flaw that I intended to rectify, as there were many good reasons to implement one. “Defense-in-depth” is the most obvious reason: if for some reason the hardware firewall is bypassed, then iptables may provide some protection. Another reason was that it was good educationally: by learning how to properly configure iptables, I learned more about firewalls and TCP/IP. This allowed me to have more productive discussions with the communications team.

I started my reading on the subject with Rusty Russel’s Packet Filtering HOWTO [11]. Knowing that egress filtering was almost as important as ingress filtering, I did some searching and found an article by Carla Schroder [12]. This, amongst other things, showed how to add logging rules that would show what traffic had been filtered. It also advocated dropping packets by default. Whilst I noticed on the mailing lists and in some articles, that there is contention on whether to drop or reject packets, I decided to check with corporate policy on the matter. Policy dictated that unwanted packets were to be dropped.

I left the rudimentary ruleset added at installation time by Lokkit, just in case this section was missed somehow. It would offer some protection in case the system was attached to the network. At this point, though, we remove the existing rules and replace them with something more tailored to the environment.

I consulted a paper from CERT [19] and the FAQ from the comp.protocol.tcp-ip.domains newsgroup [20], to better understand the port usage by BIND so I could create better filters. I only allow incoming connection traffic for BIND and OpenSSH. Any other incoming traffic must be because of a valid outgoing connection for the other services I support, which at this point is Sendmail. Later on I added NTP support and amended the firewall rules accordingly. I’ve left that in separate section in case someone wants to use this paper as build instructions, but does not want to use NTP.

I decided not to allow ICMP as this can give away information about the system. I have left some tools on the system that use ICMP (traceroute, ping) with the intention that if a network issue needs diagnosis, then the firewall can be modified to allow ICMP traffic in the short-term.

6.2 Set default policies

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
```

```
iptables -P FORWARD DROP
```

6.3 Clear existing rules

```
iptables -F RH-Lokkit-0-50-INPUT
iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD
iptables -X RH-Lokkit-0-50-INPUT
```

6.4 Allow internal traffic

Carla Schroder's article [12], alerted me to the fact that I needed to allow internal traffic to occur to allow some programs to talk to each other.

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

6.5 Deny invalid packets

As per this Neohapsis article [23] and this CERT [24] alerts, the firewall configuration needs to protect the system from packets created with invalid TCP flag setting. This was reported in the first pass of Nessus testing on the completed system (see section 23.6.2).

```
iptables -A INPUT -i eth0 -p tcp --tcp-flags SYN,FIN SYN,FIN -m limit\
--limit 10/m -j LOG --log-prefix="bogus packet"

iptables -A INPUT -i eth0 -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
```

6.6 Allow SSH traffic

One of the early decisions I made was to allow remote administration using OpenSSH. Clearly, I had to add rules to allow that type of traffic. I've attempted to minimize the risk by only allowing connections from systems within the company. Also, I use stateful packet inspection to check to see whether outgoing traffic belongs to an already established connection.

```
iptables -A INPUT -s <DMZ subnet> -i eth0 -p tcp -m tcp \
--sport ! 0:1023 --dport 22 -j ACCEPT

iptables -A OUTPUT -o eth0 -p tcp -m tcp -d <DMZ subnet>\
--sport 22 --dport ! 0:1023 -m state --state ESTABLISHED -j ACCEPT
```

6.7 Allow DNS traffic

The majority of the rules I implemented here were created from the filtering recommendations in CERT's "Securing an Internet Server" [19]. I made some modifications to the rule list after running some tests and monitoring the logs. The only major design change was the additional rules to allow DNS traffic from systems in the DMZ. These rules have no port restrictions for the clients. This is because many of the clients are going through port address translation and being given source ports under 1024. Normally, client requests are made from port values over 1023. I noticed the denials in the logs and made the adjustments in response.

From discussions with the communications team, I already knew that PAT was

happening, however I didn't realise that port values under 1024 were being used until I moritored the logs.

If I'd had more time, I would've also added some state inspection to the rules. Some of the rules could possibly use restrictions based on whether a connection has already been established.

```
     iptables -A INPUT -i eth0 -p udp -m udp --sport ! 0:1023 --dport 53 -j
ACCEPT
     iptables -A INPUT -i eth0 -p udp -m udp --sport 53 --dport ! 0:1023 -j
ACCEPT
     iptables -A INPUT -i eth0 -p udp -m udp --sport 53 --dport 53 -j
ACCEPT
     iptables -A INPUT -i eth0 -p tcp -m tcp --sport 53 --dport 53 -j
ACCEPT
     iptables -A OUTPUT -o eth0 -p udp -m udp --sport ! 0:1023 --dport 53 -
j ACCEPT
     iptables -A INPUT -i eth0 -p tcp -m tcp --sport ! 0:1023 --dport 53 -j
ACCEPT
     iptables -A INPUT -i eth0 -p tcp -m tcp --sport 53 --dport ! 0:1023 -j
ACCEPT
     iptables -A OUTPUT -o eth0 -p tcp -m tcp --sport ! 0:1023 --dport 53 -
j ACCEPT
     iptables -A OUTPUT -o eth0 -p udp -m udp --sport 53 --dport ! 0:1023 -
j ACCEPT
     iptables -A OUTPUT -o eth0 -p tcp -m tcp --sport 53 --dport ! 0:1023 -
j ACCEPT
     iptables -A OUTPUT -o eth0 -p udp -m udp --sport 53 --dport 53 -j
ACCEPT
     iptables -A OUTPUT -o eth0 -p tcp -m tcp --sport 53 --dport 53 -j
ACCEPT
     iptables -A INPUT -i eth0 -s <DMZ-subnet> --dport 53 -j ACCEPT
     iptables -A OUTPUT -o eth0 -d <DMZ-subnet> --sport 53 -j ACCEPT
```

6.8 Allow Outgoing SMTP (mail) traffic

In order to support the mailing of alerts and reports, I allow outgoing SMTP connections.

```
     iptables -A INPUT -i eth0 -p tcp -m tcp --dport ! 0:1023 --sport 25 \
     -m state --state ESTABLISHED -j ACCEPT

     iptables -A OUTPUT -o eth0 -p tcp -m tcp --dport 25 --sport ! 0:1023 \
     -m state --state NEW, ESTABLISHED -j ACCEPT
```

6.9 Log all denied traffic

```
     iptables -A INPUT -j LOG
     iptables -A OUTPUT -j LOG
     iptables -A FORWARD -j LOG
```

6.10 Save new rules & reload

```
     iptables-save > /etc/sysconfig/iptables
```

After applying these rules, I restarted iptables with the following command:

```
/etc/init.d/iptables restart
```

7 SSH (secure telnets)

7.1 Design Decisions

As I'd used OpenSSH on several systems before, including the predecessor DNS server, there wasn't too much to think about in terms of planning the implementation. The only alterations I made to the configuration was the restriction of the SSH protocol to version 2 only. I came across several references to SSH version 1 being "broken" and so decided to only allow version 2 connections. An example of the arguments against SSH version 1 can be found here: http://www.ayahuasca.net/ssh/ssh-faq-1.html#ss1.12

As usual, I only allow regular users to login. Root can only login directly at the console.

I also decided to use the rc-scripts that come with the Red Hat 9 distribution to control the service. I ran into some problems in early testing with the host key not matching the key being presented when I connected using a SSH client. I traced this to the fact that the rc-scripts use different directories for storing the keys than the default directories used by the compiled programs. I modified the rc-scripts accordingly (see below) which fixed the problem.

7.2 OpenSSL

Open SSL is a pre-requisite of OpenSSH. I downloaded the latest source bundle from OpenSSL.org. At the time of writing, this was

openssl-0.9.7b.tar.gz

As with most packages I download, I download the source from one mirror server, and download two separate versions of whatever verification file is being used (e.g. MD5 checksum) from two other mirrors. I compare the two verification files, and if they're the same, then I use them to verify the package. As per other downloads, I transferred it onto the target system using CDROM.

From /tmp I unpacked the source file

```
cd /tmp
tar xzvf /download-dir/openssl-0.9.7b.tar.gz
```

I then ran make & installed openssl (instructions are on openssl.org or in the INSTALL doc)

```
cd /tmp/openssl-0.9.7b
./config
make
make test
```

```
make install
```

7.3 OpenSSH

I downloaded the latest source bundle from OpenSSH.org. At the time of writing, this was

openssh-3.6p2.tar.gz

From /tmp I unpacked the source file

```
cd /tmp
tar xzvf /download-dir/openssh-3.6.p2.tar.gz
```

I then created the required user & group before install, otherwise the “make install” complains.

```
groupadd -g 74 sshd
mkdir /var/empty
useradd -u 74 -g 74 -d /var/empty/sshd \
-c “Privilege-separated SSH” -s /sbin/nologin sshd
```

Next was the make & install of openssh (instructions are on openssh.org or in the INSTALL doc within the source)

```
cd /tmp/openssh-3.6p2
./configure
make
make install
```

I noted the server’s key fingerprint as generated in the “make install” process for later comparison when signing in using SSH.

```
cd /usr/local/etc
ssh-keygen -lf ssh_host_rsa_key
```

I then extracted the rc-script to start and stop the sshd daemon. After mounting the Red Hat 9 install disk 1, I ran the following commands.

```
cd /
rpm2cpio /mnt/cdrom/Red Hat/RPMS/openssh-server-3.5p1 | \
cpio -iv ./etc/rc.d/init.d/sshd
```

The default settings in sshd point to the executables at /usr/bin & usr/sbin, so I changed the macros in /etc/rc.d/init.d/sshd to be more appropriate for my installation.

```
KEYGEN=/usr/local/bin/ssh-keygen
SSHD=/usr/local/sbin/sshd
RSA1_KEY=/usr/local/etc/ssh_host_key
RSA_KEY=/usr/local/etc/ssh_host_rsa_key
DSA_KEY=/usr/local/etc/ssh_host_dsa_key
```

I hashed out the reference to /etc/sysconfig/sshd.

I changed the configuration file in /usr/local/etc/sshd_config so the following lines exist.

```
Protocol 2
Banner /usr/local/etc/warn
PermitRootLogin no
```

I created a suitable warning message, as per our corporate policy, to be displayed when SSH connections are made This was stored in /usr/local/etc/warn.

I also created links for starting and stopping the SSHD service within various run levels.

```
cd /etc/rc.d
ln -s ../init.d/sshd rc3.d/S60sshd
ln -s ../init.d/sshd rc2.d/K40sshd
ln -s ../init.d/sshd rc1.d/K40sshd
```

8 Network Configuration

8.1 Set up domain

I used vi on /etc/resolv.conf and added the following:

```
domain    <your domain>
nameserver <ip of this server>
```

The system's IP address is added as a nameserver as it will resolve its own requests.

8.2 ZEROCONF

I noticed early on, that there seemed to be a spurious routing table entry for 169.254/12 network range. After some investigation, I found some web-based discussion about how this network is used on Macintoshes for network discovery. I decided to deactivate it, as it was unnecessary.

Deactivating it was achieved by adding the following to the /etc/sysconfig/network file, and then rebooting.

```
NOZEROCONF=”yes”
```

The routing table can be checked with the following command:

```
netstat –r
```

9 System Configuration

9.1 Design Decisions

Vulnerabilities can occur in TCP/IP stacks that have weak configuration. I attempted to thoroughly investigate the parameters that could be tuned. Unfortunately, it was difficult to find any one comprehensive source of information. Often I found that the Red Hat default configuration already complied with advice I found.

A great deal of information came from the "Linux Advanced Routing and Traffic Control HOWTO" [8] and Oskar Andreasson's "Ipsysctl Tutorial" [9]. Rob Thomas' "UNIX IP Stack Tuning Guide" [15] was also a starting place for me, although it is rather old at this stage. The kernel source documentation (ip-sysctl.txt) [10] also provides some good explanations on the tunable parameters.

9.2 IP stack tuning

I added the following into /etc/sysctl.conf

```
# Socket queue defense against SYN attacks
net.ipv4.tcp_max_syn_backlog=1280
net.ipv4.tcp_syncookies=1

#Disable IP redirects
net.ipv4.conf.all.send_redirects=0
net.ipv4.conf.all.accept_redirects=0

#Don't respond to directed broadcasts (SMURF)
net.ipv4.icmp_echo_ignore_broadcasts=1
```

9.3 User limits

I also added the following into /etc/security/limits.conf

```
*               soft    core            0
*               hard    core            0
```

As pointed out in the CISecurity template for Linux 1.0.0 [25], this will hinder users from being able to create core files, which can be used to gain access to information within running programs.

10 Services Configuration

10.1 Design Decisions

From the previous DNS server design, I knew that inetd could cause huge problems for security. In the previous system, I had disabled the daemon and commented out all configuration information. This time, I decided to go one step further and just not install the xinetd package. This is obviously something that affects the package list in chapter 3, but I mention it here, as it's more appropriate for this chapter.

Next, I manually inspected all the rc-scripts in /etc/init.d to see if there were any redundant services that could be disabled or removed. The only service I could find

that didn't need to be there was "netfs".

10.2 Daemon Configuration

I removed the last reference to xinetd

```
rmdir /etc/xinetd.d
```

From the rc-driectory shown in colum one of the table below, I removed the corresponding files/links on that row.

Directory	Filename
rc1.d	
rc2.d	K75netfs
rc3.d	S25netfs
rc4.d	S25netfs
rc5.d	S25netfs
rc6.d	K75netfs
init.d	netfs

11 Sendmail

11.1 Design Decisions

As already mentioned, I planned to allow alerts and reports to be sent from the DNS server to a team of people. The default service for sending mail in Red Hat 9 is Sendmail. Sendmail has been plagued with security problems in its history. Recently, however, a change in design has allowed greater security, especially within the model that I needed. I elected to stick with it as I had more experience with it and felt confident that as long as I only used it for outgoing emails and properly configured the firewall, then my risks would be low.

I followed the documentation I found on the Sendmail.org site, especially the "Secure Install" guide [27]. This has instructions for setting up Sendmail as a local server only; it doesn't accept mails from external sources. The only alterations I needed to make were to make sure that log files were being written in a filesystem that I had mounted with write permissions.

I did have some trouble initially with mails not being delivered. After consulting the communications team, it transpired that I would not be able to directly email our mail server. I worked around this problem by directing email to a relay.

11.2 Configuration Changes

I removed /etc/sysconfig/sendmail as it contains default settings for sendmail to be a listening server.

As per "Secure Install" guide, I used submit.cf settings. I also changed the /etc/init.d/sendmail file so that the following lines exist.

```
DAEMON=yes
QUEUE=30m
```

I moved the “status” file to a filesystem that allows changes

```
mv /etc/mail/statistics /var/spool/mail/statistics
```

and changed the following in the /etc/mail/sendmail.mc file:

```
dnl define(`STATUS_FILE', `/etc/mail/statistics')dnl
```

Becomes

```
dnl define(`STATUS_FILE', `/var/spool/mail/statistics')dnl
```

I checked to see if the /etc/mail/submit.cf existed, and was not empty. To recreate it, I ran the command:

```
make -C /etc/mail
```

12 Inittab

12.1 Design Decisions

I knew that there were some changes that could be made in the inittab to improve system security. The previous DNS server did have the “ctrlaltdel” line removed to stop unauthorised people from rebooting the server, but nothing else.

I used ideas in Stephen Gibson’s GCUX paper [1] as a starting point. But I decided that removing lines was better than commenting them out. As I was not allowing an X-Windows interface, I removed the line that starts X and replaced it with a line used to start a script of my own.

In trying to design the system so that the root filesystem is read-only, I realised that I could only mount it as read-only after most of the system initialisation had completed. It seemed that I needed another script that was run after everything else. I created a new script (rc.final) that I invoked from the inittab as the very last entry. This script is detailed in section 18.3 .

12.2 Configuration

I removed from /etc/inittab

```
ca::ctrlaltdel:/sbin/shutdown -t 3 -r now
```

and added

```
# Force login even in single-user mode
~~:S:wait:/sbin/login
```

after the entry for “si::sysinit:/etc/rc.d/rc.sysinit.”

I replaced:

```
x:5:respawn:/etc/X11/prefdm -nodaemon
```

with

```
#fn:3:wait:/etc/rc.final
```

We'll uncomment this as part of the last few steps in configuring the system. Otherwise it will restrict the system too much.

I then told init to re-read the inittab so it could see the changes.

```
telinit q
```

12.3 Permissions

As per the Gibson's paper, I set the inittab as immutable. Even though the entire filesystem will be made read-only, this file is very important so I prefer the extra protection.

```
chattr +i /etc/inittab
```

13 NTP (accurate timekeeping)

13.1 Design Decisions

The previous server had no time synchronisation ability. This is important to us not only for accurate logs of system activity, but also if we want to implement time sensitive security measures such as TSIG within BIND.

As I considered this an important and possibly vulnerable application, I attempted to compile it from source that was available from www.ntp.org. This didn't seem to work, and rather than spend too much time trying to discover what was wrong, I installed the Red Hat package instead.

Initially I tried to use an internal timeserver as advised by the comm.s team. This turned out to be unusable, so I switched to using three "stratum 2" systems available on the Internet. I configured the service according to documentation found through the ntp.org site [22].

I made sure that the files that require writing to were located in the /var filesystem that was mounted as writable. I also tightened the permissions on these and other files. The capabilities of the timeservers were restricted, so that they could not affect the DNS server too greatly.

My initial set of firewall rules only partially worked, and by monitoring the logs I saw that I had to add further rules to allow traffic with a source and destination port of 123. Eventually, using "ntpq –p" I could see good connection values for the three servers. I then checked the date and compared that to a "speaking clock" telephone service. The two matched, showing that ntpd was working.

13.2 Configure NTP

NTP should already be installed at this stage, so I only needed to configure and start it.

I changed the /etc/ntp.conf file so that it contained the following

```
# Adelaide
restrict 129.127.40.3 mask 255.255.255.255 nomodify notrap noquery
# Cornell
restrict 132.236.56.250 mask 255.255.255.255 nomodify notrap noquery
# Singapore
restrict 202.56.133.180 mask 255.255.255.255 nomodify notrap noquery

server  ntp.adelaide.edu.au     # University of Adelaide,
                                # South Australia
server  ntp0.cornell.edu        # Cornell University, Ithaca, NY
server  ntp.shim.org            # Singapore

driftfile /var/lib/ntp/drift

authenticate no
```

The restriction directives only appeared to work when using an IP address. Conversely, the “server” directives only appeared to work with names, not IP’s. This could cause problems in the future if the IP addresses change for any of the servers listed.

13.3 Move writable files location & enable

I created a new directory in /var for the NTP drift file that will change over time

```
mkdir /var/lib/ntp
chgrp ntp /var/lib/ntp
chmod 770 /var/lib/ntp              # so temp files can be created
mv /etc/ntp/drift /var/lib/ntp
chmod 600 /var/lib/ntp/drift
chmod 600 /etc/ntp.conf
```

I didn’t enable time synchronisation at install time, so at this point I added the NTP daemon into the startup list for runlevel 3.

```
cd /etc/rc.d/rc3.d
ln -s ../init.d/ntpd S26ntpd
```

13.4 Firewall Ammendments

The firewall rules required some changes to allow the NTP traffic to flow. I monitored the firewall logs to determine what the NTP traffic requirements were and then created appropriate rules.

```
    iptables -I OUTPUT 7 -o eth0 -p udp -m udp --sport ! 0:1023 --dport
123 -j ACCEPT
    iptables -I OUTPUT 8 -o eth0 -p udp -m udp --sport 123 --dport 123 -j
ACCEPT
    iptables -I INPUT 10 -i eth0 -p udp -m udp --sport 123 --dport !
```

```
0:1023 -m state --state ESTABLISHED -j ACCEPT
     iptables -I INPUT 11 -i eth0 -p udp -m udp --sport 123 --dport 123 -m
state --state ESTABLISHED -j ACCEPT
```

14 Syslogd Configuration

For simplicity, I left the syslogd configuration as is except for removing unmonitored services, as below. This is how the old server operated.

I removed the following lines:

```
# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler
```

15 Log Rotation

15.1 Design Decisions

Whilst the log rotation scheme employed by Red Hat 9 is adequate for most systems, I planned to improve the security on the logs by making the active log files append-only, and the history files immutable. This creates a problem for the log rotation daemon, as it can't do its job with these attributes set.

A quick look at the possible configuration available to me, confirmed that I could perform some operations on the files before and after log rotation. Essentially, I remove the attributes on the files and then replace them when the rotations are complete.

15.2 Configuration

I changed /etc/logrotate.d/syslog to be the following

```
/var/log/messages /var/log/secure /var/log/maillog /var/log/spooler
/var/log/boot.log /var/log/cron {
    sharedscripts
    prerotate
        /usr/bin/chattr -i /var/log/messages.? /var/log/secure.?
/var/log/maillog.? /var/log/spooler.? /var/log/boot.log.? /var/log/cron.?
        /usr/bin/chattr -a /var/log/messages /var/log/secure
/var/log/maillog /var/log/spooler /var/log/boot.log /var/log/cron
    endscript
    postrotate
        /usr/bin/chattr +i /var/log/messages.? /var/log/secure.?
/var/log/maillog.? /var/log/spooler.? /var/log/boot.log.? /var/log/cron.?
        /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null
|| true
        /usr/bin/chattr +a /var/log/messages /var/log/secure
/var/log/maillog /var/log/spooler /var/log/boot.log /var/log/cron
    endscript
```

```
}
```

16 File Permissions

16.1 Design Decisions

I already knew that files existed with sgid & suid permissions. I had eliminated some of these in the previous DNS server build. This time, I created a comprehensive list of those that existed on the Red Hat 9 build I had created. I also tracked down the purpose of each file to determine whether I needed it or not. The result was that I was able to remove the permissions from over half the files found. Given more testing, I think more could be removed.

I also needed to make sure that the wtmp and utmp files had only read permissions, except for root. These files are used to log user sessions and are important in tracking down changes on the system. They are also priority targets for hackers trying to cover their tracks.

16.2 SUID & SGID files

I listed all files with SUID & SGID permissions as per Stephen Gibson's paper [1].

```
find / -type f \( -perm -04000 -o -perm -02000 \)
```

The following is the list of files produced by the above command with indicators and comments as to whether they should keep their SUID/SGID status. I checked the system man pages for the purpose of each file to determine whether it was needed or not.

File path		Keep?	Comments
/usr/bin/chage	u	no	only -l option (what's my age) requires suid
/usr/bin/gpasswd	u	no	suid so that group admins can use it
/usr/bin/wall	g	no	not really required by users other than root
/usr/bin/chfn	u	no	to change your finger information
/usr/bin/chsh	u	no	to change your login shell
/usr/bin/newgrp	u	no	we are not going to be jumping groups
/usr/bin/write	g	no	not a multi-user system - not required
/usr/bin/passwd	u	yes	basic user function
/usr/bin/at	u	yes	probably not required, but fundamental
/usr/bin/crontab	u	yes	probably not required, but fundamental
/usr/sbin/ping6	u	no	not using IPv6
/usr/sbin/traceroute6	u	no	not using IPv6
/usr/sbin/usernetctl	u	no	root only. can't delete: embedded in scripts
/usr/sbin/userhelper	u	yes	GUI helper for passwd operations
/usr/sbin/lockdev	g	yes	lock devices. only sgid for group lock
/usr/sbin/traceroute	u	no	diags only done by root
/usr/local/libexec/ssh-keysign	u	yes	required by sshd
/bin/ping	u	no	diags only done by root
/bin/mount	u	no	root only
/bin/umount	u	no	root only

File path		Keep?	Comments
/bin/su	u	yes	basic user function
/sbin/pam_timestamp_check	u	yes	helper for pam db
/sbin/pwdb_chkpwd	u	yes	helper for pam db
/sbin/unix_chkpwd	u	yes	helper for pam db
/sbin/netreport	g	no	checks for status of network links
/usr/sbin/sendmail.sendmail	g	yes	sendmail service
/usr/bin/lockfile	g	yes	creates semaphores

All the files in this list marked “no” had their suid/sgid permissions removed by using the following command:

```
chmod a-s <path>
```

Where <path> is the path of the program in the left-hand column.

16.3 Utmp & wtmp

I ensured that permissions on /var/log/wtmp* and /var/run/utmp* were 644

17 User profile changes

17.1 Design Decisions

From the Linux Security HOWTO [2], I knew that loose path-settings could allow programs inserted by hackers to be run by mistake. I examined all the scripts I could identify that might contribute settings to the PATH variable. The old DNS server did have a similar trimming of path settings, but not to the level performed here.

From early testing, I knew that I was going to have trouble with command history. Either it was going to conflict with the read-only root filesystem, or it would cause Tripwire to alarm. To remove this problem altogether, I found how to disable bash command history.

17.2 Path Changes

I removed superfluous entries in PATH for all users, especially root.

From /etc/profile, I removed the following lines

```
pathmunge /usr/X11R6/bin after
```

From /etc/csh.login, I replaced

```
setenv PATH "${PATH}:/usr/X11R6/bin"
```

with

```
setenv PATH "${PATH}"
```

and

```
setenv PATH "/bin:/usr/bin:/usr/local/bin:/usr/X11R6/bin"
```

with

```
setenv PATH "/bin:/usr/bin:/usr/local/bin"
```

From /root/.bash_profile, the following was replaced

```
PATH=$PATH:$HOME/bin
```

with

```
PATH=$PATH
```

This could just be deleted altogether.

I did the above for other user directories that existed.

From /root/.cshrc and /root/.tcshrc, I replaced

```
setenv PATH "/usr/local/sbin:/usr/sbin:/sbin:${PATH}:${HOME}/bin"
```

with

```
setenv PATH "/usr/local/sbin:/usr/sbin:/sbin:${PATH}"
```

I then checked the new PATH settings for root. I ensured that each directory in it was not writable.

17.3 Disable command history

Changes to the ".bash_history" file in user directories will create false alarms from Tripwire, unless we add in specific directives to ignore them. I prefer to disable the saving of history. History will still work in a session; only history from previous sessions will be lost.

I added the following to the bottom of `/etc/bashrc`

```
#
# Unset variable pointing to history file, so no history is saved
unset HISTFILE
```

18 RC-Scripts (startup scripts)

18.1 Design Decisions

As mentioned in section 5.5 , changes to the kernel meant that the boot process could hang waiting for a keypress. To fix this problem, I came up with two alternatives: set the hanging process to exit after thirty seconds, or disable the prompting section of the rc.sysinit script. I opted for the latter, as even though the system is in a secure location, I don't want anyone intefering with the boot process. I left the other solution documented here in case it's useful for someone else.

In section 12.2 , I introduce the idea of a script used to perform the final security measures after all the other services have been started. The main reason for doing this was that the rc.sysinit script remounts the root filesystem several times, and

specifically in read-write mode. I needed the system to finish its boot process with the root filesytem in read-only mode.

I could have fixed this by changing the behavior of the rc.sysinit script. But, one of the choices I've been trying to avoid is modifying established scripts or components, unless I absolutely have to. Otherwise the system configuration becomes very obscure and difficult to administer, and upgrades wipe out the changes.

I decided to create a separate, more obvious, script instead. This allowed me to add anything else that I needed. I did consider using the script to remove the capability of mounting filesystems, thereby eliminating the chance that the root filesystem could be remounted in read-write mode. This, however, causes difficulty with the shutdown process and removes many other capabilities at the same time.

Instead, I added a section which checks that mingetty processes are running before remounting the root filesystem. This is necessary as the remounting can happen too quickly and the mingetty processes do not get created, leaving the system without any console devices. In this case it's still possible to login using SSH, but not from the system itself.

18.2 Modifications to rc.sysinit

Process observation in 2.4.20 seems to be different, and the getkey process cannot be killed because the process id cannot be found by *pidof*. We can tell it to only wait 30 seconds before continuing. To do this I could've changed the *getkey* use at the end of /etc/rc.sysinit to include "`--wait=30`"

However, I turned off this feature by ammending the /etc/sysconfig/init file as follows:

```
PROMPT=no
```

The "30-second" work-around is included in case prompting needs to be turned on at some point.

18.3 Addition of rc.final

I created the file /etc/rc.d/rc.final.

```
cd /etc/rc.d
touch rc.final
ln -s rc.d/rc.final ../rc.final
chown root.root rc.final
chmod 755 rc.final
```

I then added the following contents

```
#!/bin/sh
#
# This script will be executed at the end of the inittab list.
# This ensures that all other inittab initiated scripts are run first.
# After execution of this script, many things will not work as normal.

# Check if the mingettys are setup
NUMGETTYS=6   # make sure this number is the same as the number of
              # mingettys specified in the inittab

if [ $( /sbin/pidof mingetty | wc -w ) -le $NUMGETTYS ]; then
        sleep 3
```

```
fi

# Remount root filesystem as Read-Only
# Once the /dev part of the root fs is Read-Only, mingetty cannot create
# new terminals

mount -n -o remount,ro /
```

19 BIND 9.2.2 (DNS service)

19.1 Design Decisions

The named service was going to need the greatest amount of protection. This is the only network service that has to allow unfettered access to it, and therefore is the most likely point of attack. I considered using another resolver service other than BIND as vulnerabilities in previous versions have been exploited. D.J. Bernstein's "djbdns" [28] is a well regarded replacement for BIND. However, I had to decide against it, as my priorities were to upgrade the previous system quickly in order to reduce risk to the company, and preserve existing knowledge of the DNS server. I couldn't afford a steep learning curve in implementing new technology, at this point. My plan was to upgrade and then work on migrating when time was available.

Bernstein's site provides excellent reading on the subject of DNS services in general. One of the considerations mentioned is the setting up of a caching only server as well as an authoritative server. Unfortunately, the addition of extra hardware into the environment was denied. The DNS server remains fulfilling both tasks. However, I attempted to mitigate the risk by disabling recursion and caching in order to stop cache poisoning. Authorised requests are also forwarded to our ISP, as per their recommendations, which passes the responsibility of resolution. Essentially, though, the problem remains.

I knew that putting the named service into a chroot jail was central to the implementation. I consulted several articles on the matter starting with The Linux Documentation Project's HOWTO [18]. I followed this up with Rob Thomas' paper on securing BIND [14], which pointed me to Steve Friedl's site [16] on the subject. Friedl also has some other information specific to chroot jails on the same website[17].

I found one strange phenomenon with the chroot named service when I was testing it. Initially, I had forgotten to create the chrooted /dev/log device. However, named was able to log to the messages file. I confirmed that the service was chrooted, and could not explain how it was able to use the root filesystem /dev/log. I posted several questions on mailing lists, forums and the bind newsgroup. Most of these were not answered, or were not answered to my satisfaction. My only conclusion is that on startup, named binds to /dev/log before it gives up its root authority.

The previous design of DNS lacked several security measures. It was a "vanilla" install of BIND 8 from Red Hat RPM's. There was no split-horizon model to separate internal and external resolution. The system performed both caching and authoritative services. There was no masking of the service version. The service was not chrooted. Foreign clients were allowed to use the server as a resolver. Zone transfers were restricted. Maintainers of the DNS and the DNS service had similar

rights to files. Only root could restart the service.

Apart from the known issue of separating caching and authoritative services, I planned to fix all of the above flaws. However, I could not implement one of the security features I did want: TSIG. TSIG provides a way of proving that records transferred between DNS servers are authentic. The secondary server we use is outsourced to another company. I contacted them to see if TSIG was supported. I was informed that it was not. Hence, I could not enable it on the primary server.

19.2 Unpack and Compile Source

The first task was to transfer the BIND 9.2.2 source onto the system and into /tmp/Source, and then unpack it.

```
cd /tmp
tar zxf /tmp/Source/bind-9.2.2.tar.gz
```

I then made and installed BIND, using some advice from the ISC website for the configuration stage. The BIND FAQ (http://www.isc.org/products/BIND/FAQ.html) mentions a problem with dropping root privileges if threading is enabled. This is within the context of installing on a Linux 2.2 kernel and does not specifically talk about 2.4 kernels. To be safe, I disabled thread use.

I also disabled IPv6 support as I'm not using it on this system.

```
cd bind-9.2.2
./configure --prefix=/usr/local --disable-ipv6 --disable-threads
make
make install
```

19.3 Creating chroot jail

I created a jail directory for named to exist in. This was part of the read-only root filesystem, but had the separate filesystem /chroot/named/var mounted on it in order to be able to have configuration and zone files that can change. The advantage of having it as a separate filesystem, is that I not only can I have it read-write, but I can also restrict it from having executable or device files. Even if someone can place their own files into this filesystem, it will be more difficult for them to be used.

One of the characteristics of the old DNS server was that DNS record maintainers were in the same group as the named service. This meant that the named service had write permissions for all the files it accessed. As the named service is the most accessible component of the server, I was concerned that it could be manipulated into changing these files. To combat this, I decided to only allow the named service read access to all configuration files. Unfortunately, if I gave the named service ownership of the files, then perhaps the file permissions could be changed anyway which would be no protection at all.

The best method I could determine was to create a single user for maintaining the files. This user would then own the files and it would have read-write permissions to them. The named service would have group-ownership of the files and read-only permissions to them. This creates a problem with accountability: anybody with the

password could log in as “dns” and make changes. It would then be more difficult to trace who had made those changes. I planned to force people to login using their specific profiles and then “su” to “dns”, and the “dns” user would be not be allowed to login. This would create an accountability trail. But, it was deemed too cumbersome a procedure.

To truly fix this problem, we need to wait for ACL’s to be fully implemented in the ext3 filesystems that are used. This, I believe, is planned for the 2.6 kernel release. With ACL’s we can assign two group-ownerships (dns & named) to the files and give them different permissions. Then the files can be owned by root. Maintainers would then belong the “dns” group and have permission to modify the files.

I created a "named" user to be used for chrooted named daemon. I also created a “dns” user and group, for maintenance.

I used groupadd to add the new named and dns groups

```
groupadd -g 200 named
groupadd -g 201 dns
```

I then used useradd to add the new named and dns users. I also locked the named profile so it could not be used for login.

```
useradd -u 200 -g 200 -d /chroot/named -c Nameserver -s /bin/false
named
passwd -l named
useradd -u 201 -g 201 -G 200,201 -c "DNS admin" dns
```

The “dns” user is used to maintain DNS zone files and the named.conf

I removed the unnecessary files from the home directory that had been setup during the useradd

```
umount /chroot/named/var
rm -rf /chroot/named
mkdir -p /chroot/named/var
mount /chroot/named/var
```

I then created the jail directory structure. I decided to create all directories with writable files under the var directory, as it is the only one that is on a writable filesystem. I created links at the /chroot/named level to the directories under var to avoid confusion.

```
cd /chroot/named
mkdir dev
cd var
mkdir conf
mkdir etc
mkdir logs
mkdir run
cd ..
```

```
ln -s var/conf conf
ln -s var/etc etc
```

I created the necessary devices, by first confirming the major & minor numbers I needed to use

```
ls -lL /dev/zero /dev/null /dev/random
```

I then made the devices within the chroot jail. These are necessary, as named shouldn't be able to access the devices in the normal root filesystem. As noted above, this didn't seem to be the case for the logging device.

```
mknod dev/null c 1 3
mknod dev/random c 1 8
mknod dev/zero c 1 5
```

Finally, I copied the timezone file & made it immutable, as per the HOWTO.

```
cp /etc/localtime var/etc
chattr +i var/etc/localtime
```

19.4 BIND configuration

The next step was to create the named.conf file and place it in /chroot/named/var/etc. The file is reproduced in Appendix A. The file is a modified version of Rob Thomas' template [14]. It contains features such as ACL's for determining access to DNS information, a "bogon" list for filtering requests from invalid IP's, and a section for enabling rndc control of the named service.

I then found the latest copy of the "rootcache" using another system and the following command

```
dig @a.root-servers.net . ns > db.rootcache
```

I transferred the file onto the target system (not over the network) into the directory /chroot/named/var/conf/

I set its permissions using the named.perms script, which is created in the next section.

19.5 Create the localhost zones

I created a new file as /chroot/named/var/conf/db.localhost with the following, again as per Steve Friedl's documentation.

```
;
; db.localhost
;
$TTL    86400
```

```
@       IN SOA   @ root (
                         42              ; serial (d. adams)
                         3H              ; refresh
                         15M             ; retry
                         1W              ; expiry
                         1D )            ; minimum

        IN NS        @
        IN A         127.0.0.1
```

I also created a new file as /chroot/named/conf/db.127.0.0 with the following:

```
;
; db.127.0.0
;
$TTL    86400
@       IN      SOA     localhost. root.localhost.  (
                            1 ; Serial
                            28800       ; Refresh
                            14400       ; Retry
                            3600000     ; Expire
                            86400 )     ; Minimum
        IN      NS      localhost.
1       IN      PTR     localhost.
```

19.6 Init scripts

In order to start the compiled named service, I again used Red Hat init scripts, but modified them for my particular situation. I discovered the rpm2cpio command that allowed me to extract just the files that I needed.

The following files were restored from the bind-9.2.1-9.rpm on Red Hat 9 Install CD 1

```
/etc/rc.d/init.d/named
/etc/sysconfig/named
```

using the following commands.

```
mount /mnt/cdrom
cd /
rpm2cpio /mnt/cdrom/Red Hat/RPMS/bind-9.2.1-16.i386.rpm | \
cpio -iv ./etc/rc.d/init.d/named
rpm2cpio /mnt/cdrom/Red Hat/RPMS/bind-9.2.1-16.i386.rpm | \
```

```
cpio -iv ./etc/sysconfig/named
```

I modified the scripts behavior by adding the following lines into /etc/sysconfig/named. This sets up the chroot jail location and tells named where to find its configuration file relative to the jail's "root" path.

```
ROOTDIR="/chroot/named"
OPTIONS="-c /var/etc/named.conf"
```

I also changed the following in /etc/rc.d/init.d/named, so the scripts can find the configuration file and executable files.

```
${ROOTDIR}/etc/named.conf => ${ROOTDIR}/var/etc/named.conf
/usr/sbin/named => /usr/local/sbin/named
/usr/sbin/rndc  => /usr/local/sbin/rndc
```

I could've just linked the files by using the following, but I thought the former more obvious.

```
cd /
ln -s /usr/local/sbin/named /usr/sbin/named
ln -s /usr/local/sbin/rndc /usr/rndc
```

Finally, I linked the init script to rc-level directories

```
cd /etc/rc.d/
ln -s ../init.d/named rc2.d/S96named
ln -s ../init.d/named rc3.d/S96named
ln -s ../init.d/named rc2.d/K04named
ln -s ../init.d/named rc1.d/K04named
ln -s ../init.d/named rc0.d/K04named
ln -s ../init.d/named rc6.d/K04named
```

I made this the penultimate service to start so it starts after supportive services such as random, atd, anacron, crond (that control tripwire, logrotate, etc.). It's also the first service to stop on shutdown.

I created a /chroot/named.perms script to manage the permissions in the chroot jail. This can be found in Appendix B. It's a modified form of Friedl's script. I made some of the permissions more restrictive in removing "other" permissions, and tailored it for the dns & named ownership I've implemented. I ran this file from the /chroot directory using the following command

```
sh named.perms
```

19.7 rndc configuration

One of the previous flaws in the old system was that to reload any changes to the DNS tables, the root user had to restart the named service. This meant that the maintainers would have to request the administrators to perform this task.

Now, I've given permission for the maintainers to use the rndc command, which

allows them control over the named service. I do have some reservations about handing this control over, however, as I stated before, security is often compromised by flexibility.

Instructions for setting this up were taken mostly from Friedl's pages[16], but I've increased the number of bits used in the key.

I created the new file /chroot/named/var/etc/rndc.conf that contained the following:

```
#
# /chroot/named/var/etc/rndc.conf
#

options {
        default-server  127.0.0.1;
        default-key     "rndckey";
};

server 127.0.0.1 {
        key     "rndckey";
};

key "rndckey" {
        algorithm       "hmac-md5";
        secret          "private-key goes here";
};
```

I then ran the following commands

```
cd /tmp
/usr/local/sbin/dnssec-keygen -a HMAC-MD5 -b 512 -n HOST rndc
```

This created two files

Krndc.+157+*something*.key

Krndc.+157+*something*.private

I copied the key value from Krndc.+157+*something*.private and put it into the relevant section in /chroot/named/var/etc/rndc.conf

I then linked the chrooted rndc.conf into /usr/local/etc so that rndc knew where to find it

```
ln -s /chroot/named/var/etc/rndc.conf /usr/local/etc/rndc.conf
```

A controls section needed to be added into named.conf

```
controls {
        inet 127.0.0.1 allow { 127.0.0.1; } keys { rndckey; };
};

key "rndckey" {
        algorithm       "hmac-md5";
```

```
                secret          "private-key goes here";
        };
```

This gives only local access to the named service through rndc. We don't want remote access.

19.8 Extras

19.8.1 File permissions

I removed “other” permissions on sbin files and allowed users in the “dns” group to administer the service.

```
chgrp dns /usr/local/sbin/rndc
chmod o= /usr/local/sbin/*
```

I also changed permissions on /var/log/messages file, so that the “dns” group could monitor the named service when it's reloaded.

```
chgrp dns /var/log/messages
chmod g+r /var/log/messages
```

19.8.2 User profile

Finally, I changed the “dns” user profile so that named commands could be accessed more easily. I edited the /home/dns/bash_profile and replaced “/home/dns/bin” with “/usr/local/sbin” in the PATH settings.

20 Post Install Procedures

20.1 Remove Developer Software

Now that I’d built everything that I needed, I could remove the tools that were used.

```
rpm -e binutils gcc gcc-c++ kernel-source
rpm -e cpp glibc-kernheaders glibc-devel
rpm -e diffutils zlib-devel libstdc++-devel ncurses-devel
```

I also removed the source used to construct software

```
cd /tmp
rm -Rf Source
rm -Rf bind-9.2.2
rm -Rf openssh-3.6.1p2
rm -Rf openssl-0.9.7b
rm modules
```

20.2 Remove Old Kernel

The new kernel seemed to be behaving itself, so I switched to using it by default, and removed the one that was used at installation time.

20.2.1 GRUB

I deleted the following lines from /etc/grub.conf, which leaves the second option as the default now.

```
title Red Hat Linux (2.4.20-8)
     root (hd0,0)
     kernel /vmlinuz-2.4.20-8 ro root=LABEL=/
     initrd /initrd-2.4.20-8.img
```

20.2.2 Kernel Files

I also removed the old kernel files from the system using this command:

```
rm /boot/*2.4.20-8*
```

This removed the following files

```
config-2.4.20-8
module-info-2.4.20-8
vmlinux-2.4.20-8
initrd-2.4.20-8.img
System.map-2.4.20-8
vmlinuz-2.4.20-8
```

I could also remove the “module-info” file, as the kernel no longer uses modules

```
rm /boot/module-info
```

Lastly, I relinked the vmlinuz file to new kernel

```
cd /boot
ln -sf vmlinuz-2.4.20-18.9 vmlinuz
```

20.3 Filesystem mount options

Here’s where I made some departures from normal filesystem layouts. As with the previous system, I knew that there was no need to modularise the filesytem usage in order to cater for future expansion or manage space usage. So, in that case I had a very flat layout with boot and root as the only major filesystems created.

With this system, I wanted to use the mount options to better protect it. Any filesystem that contained executable, device, or suid files was to become read-only. This way, no modifications to existing executables and device files could be made. Also, no new files could be created.

The exact opposite rule applies to the other filesystems; any filesystem that could be written to, was not allowed to contain executables, device, or suid files. With the boot

filesystem, though, I could be even more stringent. It's a static filesystem that has no devices or suid files.

The fstab listing below shows the mount options I employed to improve security.

```
LABEL=/                 /                 ext3    ro
1 1
LABEL=/boot             /boot             ext3    nodev,nosuid,ro
1 2
LABEL=/chroot/named/va  /chroot/named/var ext3    nodev,noexec,nosuid,rw
1 2
none                    /dev/pts          devpts  gid=5,mode=620
0 0
none                    /proc             proc    defaults
0 0
none                    /dev/shm          tmpfs   defaults
0 0
LABEL=/var              /var              ext3    nodev,noexec,nosuid,rw
1 2
/dev/hda3               swap              swap    defaults
0 0
/dev/cdrom              /mnt/cdrom        udf,iso9660
noauto,owner,kudzu,ro 0 0
/dev/fd0                /mnt/floppy       auto    noauto,owner,kudzu
0 0
```

20.3.1 File and directory changes

With the fresh install of Red Hat 9, the standard location for /tmp is in the root filesystem. As we're mounting root as read-only, applications that use /tmp for creating temporary files are denied access and can fail. To alleviate this, I relocated /tmp into var, which is a writable filesystem.

```
cd /
mv /var/tmp/* /tmp
rmdir /var/tmp
mv tmp var
ln -s /var/tmp tmp
```

20.4 Prepare to seal the system

At this point, I was ready to start closing down access to the system from further modification. I'd already created the rc.final in section 18.3 to facilitate mounting the root filesystem as read-only. The next reboot should perform that action, so I uncommented the line in the inittab that invokes the rc.final script.

First I removed the immutables attribrute on the inittab so I can edit it.

```
chattr -i /etc/inittab
```

I uncommented the last line in /etc/inittab. Then I re-established the immutable attribute on the inittab .

```
chattr +i /etc/inittab
```

20.5 Setup Tripwire

Intrusion Detection Systems are essential for todays systems that are deemed critical. I chose Tripwire as it is featured in SANS GSEC training material. It's also available through the Red Hat 9 install. I attempted to compile from source but ran into some difficulty. Rather than risk an uncertain install, I chose to use the Red Hat package.

I examined the default policy file and trimmed it significantly. The way I had designed the system was so that very few components should change, therefore I shouldn't need a complex policy to cover the different files and their profiles. I began by creating sections for each filesystem and then tailoring the files from there. Most of the exceptions I needed to add in were for device files that changed on a reboot.

20.5.1 Customisation

Eventually a new policy file (/etc/tripwire/twpol.txt) and a new configuration file (/etc/tripwire/twcfg.txt) evolved, which can be found in Appendix C.

I created a new temporary area for tripwire as the default configuration uses /etc, which is in a read-only filesystem now.

```
mkdir /var/tmp/tripwire
chmod 700 /var/tmp/tripwire
```

This in inline with the customisation already in /etc/tripwire/twcfg.txt

```
TEMPDIRECTORY=/var/tmp/tripwire
```

20.5.2 Initialise Tripwire

I then added a crontab entry to schedule a tripwire check twice a day. This emails a report on the system integrity to a group email address, twice per day. A group email address is better than a single contact. In the case that one of the people in the group is not receiving their email for some reason, the others can still receive the notifications.

```
0 1,13 * * *    /usr/sbin/tripwire --check --email-report
```

I removed the tripwire entry in the /etc/crontab.daily directory as I preferred to be able to schedule more than one tripwire check per day. I noted that an entire system check took about ten minutes and didn't seem to adversely affect system performance too greatly. This being the case, I decided to schedule two checks per day, thus giving a smaller window within which something could be changed.

```
rm /etc/cron.daily/tripwire-check
```

I then ran the Tripwire install script.

```
/etc/tripwire/twinstall.sh
```

I entered the two passwords as requested, then recorded them somewhere secure. Company policy dictates that passwords are available to more than one person in case one is unavailable.

Next, I created the tripwire database with this command

```
tripwire --init
```

I then removed the plain-text copies of configuration files twcfg.txt & twpol.txt from the machine.

20.6 Seal the system

Finally, I rebooted the system, which performs the last step of mounting the root filesystem as read-only.

21 Post Install Checks

After the system build was completed, I checked basic functionality using the following list. It's outside the scope of this document to provide comprehensive instructions on how this was performed.

21.1 Functional Checks

Checklist of required functions

- shutdown & reboot of server
- sign on via ssh
- resolve a fqdn to address for an owned zone
- resolve an IP address to a fqdn for an owned zone
- resolve a fqdn to address for an unknown zone
- resolve an IP address to a fqdn for an unknown zone
- creation of a new zone file
- restart and reload of named service
- alteration of zone file
- deletion of a zone
- mail to an internal address
- mail to an external address
- detection of a changed file via tripwire

- reception of email tripwire notification
- check NTP service is operating correctly
- test filesystem attributes:
  - attempt to create and use devices & suid files in filesystems where they are forbidden
  - attempt to write to files in read-only filesystems

21.2 Integrity Checks

I tested and compared the system security level using the following methods. For the nmap and Nessus tests, results are listed in Appendix F.

1. Use nmap to check the network integrity
2. Use CISecurity gold standard to compare with my system setup
3. Use Nessus to attempt to find weaknesses

22 Bibliography

References

1. Gibson, Stephen. “Installing and Securing a Shell Access Server Using Red Hat 6.2 Linux.” http://www.giac.org/practical/Stephen_Gibson_GCUX.doc (9 July 2003)
2. Fenzi, Kevin. “Linux Security HOWTO.” The Linux Documentation Project. V2.0. 11 June 2002. URL:http://www.tldp.org/HOWTO/Security-HOWTO/index.html (10 July 2003)
3. Wreski, Dave. “Linux 2.4: Next Generation Kernel Security.” 03 January 2001. URL:http://www.linuxsecurity.com/feature_stories/kernel-24-security-printer.html (13 July 2003)
4. Vasudevan, Alavoor. “The Linux Kernel HOWTO.” The Linux Documentation Project. v6.3. 4 July 2003. URL:http://www.tldp.org/HOWTO/Kernel-HOWTO/index.html
5. Mourani, Gerhard. “How to Build, Install, Secure and Optimize a Monolithic Kernel.” v2.0. 22 October 2002. URL:http://www.openna.com/documentations/articles/kernel/index.php?e=0,1
6. Seifried, Kurt. “Linux Administrator’s Security Guide.” 1 October 2001. URL:http://www.seifried.org/lasg (13 July 2003)
7. Onsight Inc. “Onsight Links.” URL:http://www.onsight.com/links (13 July 2003)
8. Hubert, Bert. “Linux Advanced Routing and Traffic Filtering HOWTO.” V1.0.0. 5 July 2003. URL:http://lartc.org/howto/lartc.kernel.obscure.html
9. Adreasson, Oskar. “Ipsysctl Tutorial.” V1.0.4. 21 May 2003. URL:http://ipsysctl-tutorial.frozentux.net/ipsysctl-tutorial.html
10. Kuznetsov, Alexey. “ip-sysctl.txt.” v1.5. 19 August 2002. URL:http://iptables-tutorial.frozentux.net/other/ip-sysctl.txt
11. Russel, Rusty. “Linux 2.4 Packet Filtering HOWTO.” V1.26. 24 January 2002. URL:http://www.netfilter.org/documentation/HOWTO//packet-filtering-HOWTO.html
12. Schroder, Carla. “Egress Filtering: Fencing in the Bad Guys.” 20 March 2003. URL:http://networking.earthweb.com/netsysm/article.php/10951_2168251_1
13. Steadman, Jody. “Securely implementing a BIND DNS server on RedHat Linux.” URL: http://www.giac.org/practical/GSEC/Jody_Steadman_GSEC.pdf (13 July 2003)
14. Thomas, Rob. “Secure BIND Template.” V4.0. 8 April 2003. URL:http://www.cymru.com/Documents/secure-bind-template.html (10 July 2003)
15. Thomas, Rob. “UNIX IP Stack Tuning Guide.” V2.7. 3 December 2000. URL: http://www.cymru.com/Documents/ip-stack-tuning.html
16. Friedl, Steve. “Building and configuring BIND 9.” Unixwiz.net Tech Tips.

URL: http://www.unixwiz.net/techtips/bind9-chroot.html (10 July 2003)

17. Friedl, Steve. “Best Practices for UNIX chroot() Operations.” Unixwiz.net Tech Tips. 18 January 2002. URL:http://www.unixwiz.net/techtips/chroot-practices.html (10 July 2003)
18. Wunsch, Scott. “Chroot-BIND HOWTO.” v1.5. 1 December 2001. URL:http://en.tldp.org/HOWTO/Chroot-BIND-HOWTO.html (10 July 2003)
19. Householder, Allen King, Brian. “Securing an Internet Name Server.” August 2002. URL:http://www.cert.org/archive/pdf/dns.pdf
20. Peckham, Chris (maintainer). “comp.protocol.tcp-ip.domains Frequently Asked Questions.” V1.27. 17 June 1999. URL:http://www.faqs.org/faqs/internet/tcp-ip/domains-faq/
21. Internet Software Consortium. “BIND 9 Administrator Reference Manual.” 2001. URL:http://www.nominum.com/content/documents/bind9arm.pdf
22. Mills, Dr. David. “Quick Start.” The Network Time Protocol (NTP) Distribution. 11 January 2003. URL:http://www.eecis.udel.edu/~mills/ntp/html/quick.html
23. Starzetz, Paul. “Ambiguities in TCP/IP - firewall bypassing.” 18 October 2002. URL:http://archives.neohapsis.com/archives/bugtraq/2002-10/0266.html
24. Finlay, Ian A.”Vulnerability Note VU#464113 - TCP/IP implementations handle unusual flag combinations inconsistently.” CERT/CC Vulnerability Notes Database. 30 May 2003. URL:http://www.kb.cert.org/vuls/id/464113
25. The Center for Internet Security. “Linux Benchmark.” V1.0.0. 16 February 2002. URL:http://www.cisecurity.org/bench_linux.html *
26. Fuller, Johnray Ha, John. “Red Hat Linux 9: Red Hat Linux Security Guide.” 2002. URL: http://www.Red Hat.com/docs/manuals/linux/RHL-9-Manual/security-guide/s1-security-updates-website.html
27. Sendmail Consortium. “Secure Install.” Sendmail Documentation. V1.50. 29 March 2002. URL:http://www.sendmail.org/secure-install.html
28. Bernstein, Dan J. “dbjdns: Domain Name System tools.” URL:http://cr.yp.to/djbdns.html (10 July 2003)

* This document is only accessible through a free registration process.

23 Appendices

23.1 Appendix A

The censored named.conf used in the final build of the DNS server. This is a tailored version of the template found on Rob Thomas' website [14].

```
// @(#)named.conf 02 OCT 2001 Rob Thomas robt@cymru.com
// Set up our ACLs
// In BIND 8, ACL names with quotes were treated as different from
// the same name without quotes. In BIND 9, both are treated as
// the same.
// Modified by Mark Chandler to suit our environment
acl "xfer" {
      // Allow transfers to our secondary
      xx.xx.xx.xx;
      xx.xx.xx.xx;
};

acl "trusted" {


      // Place our internal and DMZ subnets in here so that
      // intranet and DMZ clients may send DNS queries.  This
      // also prevents outside hosts from using our name server
      // as a resolver for other domains.
      //
      xx.xx/xx;        // internal public subnets
      xx.xx.xx/yy;     // external private subnets
      xx.xx.xx/yy;     // client subnet
      xx.xx.xx/yy         // ditto
      localhost;

};

acl "bogon" {
    // Filter out the bogon networks.  These are networks
    // listed by IANA as test, RFC1918, Multicast, experi-
    // mental, etc.  If you see DNS queries or updates with
    // a source address within these networks, this is likely
    // of malicious origin. CAUTION: If you are using RFC1918
    // netblocks on your network, remove those netblocks from
    // this list of blackhole ACLs!
    0.0.0.0/8;
    1.0.0.0/8;
    2.0.0.0/8;
    5.0.0.0/8;
    7.0.0.0/8;
    10.0.0.0/8;
    23.0.0.0/8;
    27.0.0.0/8;
    31.0.0.0/8;
    36.0.0.0/8;
    37.0.0.0/8;
    39.0.0.0/8;
    41.0.0.0/8;
    42.0.0.0/8;
    49.0.0.0/8;
```

```
50.0.0.0/8;
58.0.0.0/8;
59.0.0.0/8;
70.0.0.0/8;
71.0.0.0/8;
72.0.0.0/8;
73.0.0.0/8;
74.0.0.0/8;
75.0.0.0/8;
76.0.0.0/8;
77.0.0.0/8;
78.0.0.0/8;
79.0.0.0/8;
83.0.0.0/8;
84.0.0.0/8;
85.0.0.0/8;
86.0.0.0/8;
87.0.0.0/8;
88.0.0.0/8;
89.0.0.0/8;
90.0.0.0/8;
91.0.0.0/8;
92.0.0.0/8;
93.0.0.0/8;
94.0.0.0/8;
95.0.0.0/8;
96.0.0.0/8;
97.0.0.0/8;
98.0.0.0/8;
99.0.0.0/8;
100.0.0.0/8;
101.0.0.0/8;
102.0.0.0/8;
103.0.0.0/8;
104.0.0.0/8;
105.0.0.0/8;
106.0.0.0/8;
107.0.0.0/8;
108.0.0.0/8;
109.0.0.0/8;
110.0.0.0/8;
111.0.0.0/8;
112.0.0.0/8;
113.0.0.0/8;
114.0.0.0/8;
115.0.0.0/8;
116.0.0.0/8;
117.0.0.0/8;
118.0.0.0/8;
119.0.0.0/8;
120.0.0.0/8;
121.0.0.0/8;
122.0.0.0/8;
123.0.0.0/8;
124.0.0.0/8;
125.0.0.0/8;
126.0.0.0/8;
127.0.0.0/8;
169.254.0.0/16;
172.16.0.0/12;
173.0.0.0/8;
```

```
    174.0.0.0/8;
    175.0.0.0/8;
    176.0.0.0/8;
    177.0.0.0/8;
    178.0.0.0/8;
    179.0.0.0/8;
    180.0.0.0/8;
    181.0.0.0/8;
    182.0.0.0/8;
    183.0.0.0/8;
    184.0.0.0/8;
    185.0.0.0/8;
    186.0.0.0/8;
    187.0.0.0/8;
    189.0.0.0/8;
    190.0.0.0/8;
    192.0.2.0/24;
    192.168.0.0/16;
    197.0.0.0/8;
    223.0.0.0/8;
    224.0.0.0/3;
};


// Using default logging
// This will mean a syslog that is difficult to read,
// but LogWatch should make this task easier.



// Set options for security
options {
    directory          "/var/conf";
    pid-file           "/var/run/named.pid";
    statistics-file    "/var/run/named.stats";
//  memstatistics-file "/named.memstats";
    dump-file          "/var/run/named.db";

    forwarders {
                 xx.xx.xx.xx;   // forward DNS requests to ISP
                 xx.xx.xx.xx;   // primary and secondary servers
               };

    // hide our "real" version number
    version        "[secured]";

    zone-statistics yes;

    // Prevent DoS attacks by generating bogus zone transfer
    // requests.  This will result in slower updates to the
    // slave servers (e.g. they will await the poll interval
    // before checking for updates).
    // notify no;
    // Changed so that more timely updates can be done.
    notify yes;

    // Generate more efficient zone transfers.  This will place
    // multiple DNS records in a DNS message, instead of one per
    // DNS message.
    transfer-format many-answers;
```

```
    // Set the maximum zone transfer time to something more
    // reasonable.  In this case, we state that any zone transfer
    // that takes longer than 60 minutes is unlikely to ever
    // complete.  WARNING:  If you have very large zone files,
    // adjust this to fit your requirements.
    max-transfer-time-in 60;

    // We have no dynamic interfaces, so BIND shouldn't need to
    // poll for interface state {UP|DOWN}.
    interface-interval 0;

    allow-transfer {
        // Zone tranfers limited to members of the
        // "xfer" ACL.
        xfer;
    };

    allow-query {
        // Accept queries from our "trusted" ACL.  We will
        // allow anyone to query our master zones below.
        // This prevents us from becoming a free DNS server
        // to the masses.
        trusted;
    };

    blackhole {
        // Deny anything from the bogon networks as
        // detailed in the "bogon" ACL.
        bogon;
    };
};


view "internal-in" in {
    // Our internal (trusted) view. We permit the internal networks
    // to freely access this view. We perform recursion for our
    // internal hosts, and retrieve data from the cache for them.

    match-clients { trusted; };
    recursion yes;
    additional-from-auth yes;
    additional-from-cache yes;

      zone "." in {
          // Link in the root server hint file.
          type hint;
          file "db.cache";
      };

      zone "localhost" {
          // Localhost forward domain
          type master;
          file "db.localhost";
          notify no;

          allow-query {
              any;
          };

          allow-transfer {
              none;
```

```
        };
    };

    zone "0.0.127.in-addr.arpa" in {
        // Allow queries for the 127/8 network, but not zone transfers.
        // Every name server, both slave and master, will be a master
        // for this zone.
        type master;
        file "db.127.0.0";
        notify no;

        allow-query {
            any;
        };

        allow-transfer {
            none;
        };
    };

         // other zones are placed here but have been
         // removed for confidentiality & security reasons.


};

// Create a view for external DNS clients.
view "external-in" in {
    // Our external (untrusted) view. We permit any client to access
    // portions of this view. We do not perform recursion or cache
    // access for hosts using this view.

    match-clients { any; };
    recursion no;
    additional-from-auth no;
    additional-from-cache no;

      // Link in our zones
      zone "." in {
          type hint;
          file "db.cache";
      };

    //
    // KEEP THESE DOMAINS IN ALPHABETICAL ORDER
    //

         // Internet zones are placed here, but have been
         // removed for confidentiality & security reasons.

};

// Create a view for all clients perusing the CHAOS class.
// We allow internal hosts to query our version number.
// This is a good idea from a support point of view.

/* view "external-chaos" chaos {
    match-clients { any; };
    recursion no;

         zone "." {
```

```
            type hint;
            file "/dev/null";
        };

        zone "bind" {
            type master;
            file "master/db.bind";

            allow-query {
                trusted;
            };
            allow-transfer {
                none;
            };
    };


};
*/

controls {
        inet 127.0.0.1 allow { 127.0.0.1; } keys { rndckey; };
        };

key "rndckey" {
                algorithm       "hmac-md5";
                secret          "removed";
        };
```

23.2 Appendix B

named.perms script with ammendments

```
#
# named.perms
#
#   Set the ownership and permissions on the named directory
#


cd /chroot/named



# By default, root owns everything and only root can write, but dirs
# have to be executable too. Note that some platforms use a colon
# instead of a dot between user/group in the chown parameters}
# - we want to give as few permissions to group named, because the named
# - process is connected to the Internet. If it is compromised, then we
# - want it to be able to do as little as possible afterwards.
# - This, unfortunately, stops us from being able to add other non-root
```

```
# - users to the named group for administrative purposes.


chown -R root.named .

find . -type f -print | xargs chmod u=rw,g=r,o=     # regular files
find . -type d -print | xargs chmod u=rwx,g=x,o=   # directories

# allow generic DNS admin non-root user to modify DNS info.
chown dns.named var/etc
chown dns.named var/etc/named.conf

# the named.conf and rndc.conf must protect their keys
chmod o= var/etc/*.conf

# allow generic DNS admin non-root user to modify DNS info.
chown dns.named var/conf
find var/conf -type f -maxdepth 1 -print | xargs chown dns.named

# the "secondaries" directory is where we park files from
# master nameservers, and named needs to be able to update
# these files and create new ones.

touch var/conf/secondaries/.empty  # placeholder
find var/conf/secondaries/ -type f -print | xargs chown named.named
find var/conf/secondaries/ -type f -print | xargs chmod ug=r,o=

chown root.named var/conf/secondaries/
chmod ug=rwx,o=  var/conf/secondaries/

# the var/run business is for the PID file
#chown root.root  var/            Changed from original
chown root.named  var/
#chmod u=rwx,g=x,o= var/          Changed from original
chmod u=rwx,g=x var/

chown root.named  var/run/
chmod ug=rwx,o= var/run/

# named has to be able to create logfiles
# - but doesn't need to be able to read them,
chown root.named  var/logs/
chmod u=rwx,g=wx var/logs/
```

23.3 Appendix C

23.3.1 twpol.txt contents

```
###############################################################################
###
 #
##
###############################################################################
### #
#
# #
#                           DNS Server Policy File
# #
#                                  V0.1.0
# #
#                       a tailored version of the default
# #
#
# #
#                       Policy file for Red Hat Linux 7.0
# #
#                                  V1.0.0
# #
#                              July 18, 2000
# #
#
##
###############################################################################
###



###############################################################################
###
 #
##
###############################################################################
### #
#
# #
#
# #
# The example policy file is best run with 'Loose Directory Checking'
# #
# enabled. Set LOOSEDIRECTORYCHECKING=TRUE in the Tripwire Configuration
# #
# file.
# #
#
# #
# Email support is not included and must be added to this file.
# #
# Add the 'emailto=' to the rule directive section of each rule (add a
comma # #
# after the 'severity=' line and add an 'emailto=' and include the email
# #
```

```
# addresses you want the violation reports to go to).  Addresses are
# #
# semi-colon delimited.
# #
#
##
###############################################################################
###




###############################################################################
###
 #
##
###############################################################################
### #
#
# #
# Global Variable Definitions
# #
#
##
###############################################################################
###

@@section GLOBAL
TWROOT="/usr/sbin";
TWBIN="/usr/sbin";
TWPOL="/etc/tripwire";
TWDB="/etc/tripwire";
TWSKEY="/etc/tripwire";
TWLKEY="/etc/tripwire";
TWREPORT="/var/lib/tripwire/report";
HOSTNAME=ns1;

@@section FS
SEC_CRIT      = $(IgnoreNone)-SHa ;  # Critical files that cannot change
SEC_SUID      = $(IgnoreNone)-SHa ;  # Binaries with the SUID or SGID flags
set
SEC_BIN       = $(ReadOnly) ;        # Binaries that should not change
SEC_CONFIG    = $(Dynamic) ;         # Config files that are changed
infrequently but accessed often
SEC_LOG       = $(Growing) ;         # Files that grow, but that should
never change ownership
SEC_INVARIANT = +tpug ;              # Directories that should never change
permission or ownership
SIG_LOW       = 33 ;                 # Non-critical files that are of
minimal security impact
SIG_MED       = 66 ;                 # Non-critical files that are of
significant security impact
SIG_HI        = 100 ;                # Critical files that are significant
points of vulnerability


# Tripwire Binaries
(
  rulename = "Tripwire Binaries",
  severity = $(SIG_HI),
  emailto  = supportgroup@company.com
```

```
)
{
  $(TWBIN)/siggen                       -> $(SEC_BIN) ;
  $(TWBIN)/tripwire                     -> $(SEC_BIN) ;
  $(TWBIN)/twadmin                      -> $(SEC_BIN) ;
  $(TWBIN)/twprint                      -> $(SEC_BIN) ;
}

# Tripwire Data Files - Configuration Files, Policy Files, Keys, Reports,
Databases
(
  rulename = "Tripwire Data Files",
  severity = $(SIG_HI),
  emailto  = supportgroup@company.com
)
{
  # NOTE: We remove the inode attribute because when Tripwire creates a
backup,
  # it does so by renaming the old file and creating a new one (which will
  # have a new inode number).  Inode is left turned on for keys, which
shouldn't
  # ever change.

  # NOTE: The first integrity check triggers this rule and each integrity
check
  # afterward triggers this rule until a database update is run, since the
  # database file does not exist before that point.

  $(TWDB)                               -> $(SEC_CONFIG) -i ;
  $(TWPOL)/tw.pol                       -> $(SEC_BIN) -i ;
  $(TWPOL)/tw.cfg                       -> $(SEC_BIN) -i ;
  $(TWLKEY)/$(HOSTNAME)-local.key       -> $(SEC_BIN) ;
  $(TWSKEY)/site.key                    -> $(SEC_BIN) ;

  #don't scan the individual reports
  $(TWREPORT)                           -> $(SEC_CONFIG) (recurse=0) ;
}




  ##########################
 #                        ##
########################## #
#                       # #
# Temporary directories # #
#                       ##
##########################
(
  rulename = "Temporary directories",
  recurse = false,
  severity = $(SIG_LOW)
)
{
  /usr/tmp                              -> $(SEC_INVARIANT) ;
  /var/tmp                              -> $(SEC_INVARIANT) ;
  /tmp                                  -> $(SEC_INVARIANT) ;
}

   ################################################
  ################################################
```

```
  # Everything under /boot should never change as  ##
  # as it's mounted readonly.                      ##
  ################################################
(
  rulename = "Boot filesystem",
  severity = $(SIG_HI),
  emailto  = supportgroup@company.com
)
{
     /boot                              -> $(SEC_CRIT) ;
}

   ################################################
  ################################################
  # Everything under / should never change as it's ##
  # mounted readonly. Except for those listed.     ##
  # They may change on bootup.                     ##
  ################################################
(
  rulename = "Root filesystem",
  severity = $(SIG_HI),
  emailto  = supportgroup@company.com
)
{
     /                                  -> $(SEC_CRIT) ;
     /dev/kmem                          -> $(Device) ;
     /dev/mem                           -> $(Device) ;
     /dev/null                          -> $(Device) ;
     /dev/ptmx                          -> $(Device) ;
     /dev/zero                          -> $(Device) ;
     /dev/log                           -> $(SEC_CONFIG) ;
     /dev/cua0                          -> $(SEC_CONFIG) ;
     /dev/console                       -> $(SEC_CONFIG) -u ; # User ID may
change on console login/logout.
     /dev/tty2                          -> $(SEC_CONFIG) ; # tty devices
     /dev/tty3                          -> $(SEC_CONFIG) ; # are extremely
     /dev/tty4                          -> $(SEC_CONFIG) ; # variable
     /dev/tty5                          -> $(SEC_CONFIG) ;
     /dev/tty6                          -> $(SEC_CONFIG) ;
     /dev/urandom                       -> $(SEC_CONFIG) ;
     /dev/initctl                       -> $(SEC_CONFIG) ;
}


   ################################################
  ################################################
  # These files change during system lifetime      ##
  ################################################
(
  rulename = "Var filesystem",
  severity = $(SIG_HI),
  emailto  = supportgroup@company.com
)
{
     /var                               -> $(SEC_CRIT) ;
     /var/cache/man                     -> $(SEC_CONFIG) ; # whatis updates
     /var/lib/logrotate.status          -> $(SEC_CONFIG) ;
     /var/lib/ntp/drift                 -> $(SEC_CONFIG) ;
     /var/lock                          -> $(SEC_CONFIG) ;
     /var/lock/subsys                   -> $(SEC_CONFIG) ;
     /var/lock/subsys/random            -> $(SEC_CONFIG) ;
```

```
     /var/lock/subsys/network          -> $(SEC_CONFIG) ;
     /var/lock/subsys/syslog           -> $(SEC_CONFIG) ;
     /var/lock/subsys/atd              -> $(SEC_CONFIG) ;
     /var/lock/subsys/crond            -> $(SEC_CONFIG) ;
     /var/lock/subsys/local            -> $(SEC_CONFIG) ;
     /var/lock/subsys/named            -> $(SEC_CONFIG) ; #Uncomment when
this file exists
     /var/lock/subsys/sendmail         -> $(SEC_CONFIG) ;
     /var/lock/subsys/sm-client        -> $(SEC_CONFIG) ;
     /var/lock/subsys/sshd             -> $(SEC_CONFIG) ;
     /var/lock/subsys/anacron          -> $(SEC_CONFIG) ;
     /var/lock/subsys/iptables         -> $(SEC_CONFIG) ;
     /var/lock/subsys/keytable         -> $(SEC_CONFIG) ;
     /var/lock/subsys/kudzu            -> $(SEC_CONFIG) ;
     /var/run                          -> $(SEC_CONFIG) ; # daemon PIDs
     /var/log                          -> $(SEC_CONFIG) -i ; # log-rotation
     /var/spool/anacron                -> $(SEC_CONFIG) ;
     /var/spool/clientmqueue           -> $(SEC_CONFIG) ;
     /var/spool/mail                   -> $(SEC_CONFIG) ;
     /var/spool/mqueue                 -> $(SEC_CONFIG) ;
}

   ################################################
  ################################################
  # Chroot Jail Var: only very few of these files  ##
  # should change during system lifetime.          ##
  ################################################
(
  rulename = "Chroot Jail Var",
  severity = $(SIG_HI),
  emailto  = supportgroup@company.com
)
{
     /chroot/named/var                 -> $(SEC_CRIT) ;
     /chroot/named/var/run             -> $(SEC_CONFIG) ;
     /chroot/named/log                 -> $(SEC_CONFIG) ;
}

  ####################
 #                  ##
#################### #
#                  # #
# Critical devices # #
#                  ##
####################
(
  rulename = "Critical devices",
  severity = $(SIG_HI),
  emailto  = supportgroup@company.com,
  recurse  = false
)
{
     #/proc/devices                     -> $(Device) ;
     #/proc/net                         -> $(Device) ;
     #/proc/sys                         -> $(Device) ;
     #/proc/cpuinfo                     -> $(Device) ;
     #/proc/modules                     -> $(Device) ;
     #/proc/mounts                      -> $(Device) ;
     #/proc/dma                         -> $(Device) ;
     #/proc/filesystems                 -> $(Device) ;
     #/proc/pci                         -> $(Device) ;
```

```
      #/proc/interrupts                -> $(Device) ;
      #/proc/rtc                       -> $(Device) ;
      #/proc/ioports                   -> $(Device) ;
      #/proc/scsi                      -> $(Device) ;
      #/proc/kcore                     -> $(Device) ;
      #/proc/self                      -> $(Device) ;
      #/proc/kmsg                      -> $(Device) ;
      #/proc/stat                      -> $(Device) ;
      #/proc/ksyms                     -> $(Device) ;
      #/proc/loadavg                   -> $(Device) ;
      #/proc/uptime                    -> $(Device) ;
      #/proc/locks                     -> $(Device) ;
      #/proc/version                   -> $(Device) ;
      #/proc/mdstat                    -> $(Device) ;
      #/proc/meminfo                   -> $(Device) ;
      #/proc/cmdline                   -> $(Device) ;
      #/proc/misc                      -> $(Device) ;
}

#==========================================================================
===
#
# Copyright 2000 Tripwire, Inc. Tripwire is a registered trademark of
Tripwire,
# Inc. in the United States and other countries. All rights reserved.
#
# Linux is a registered trademark of Linus Torvalds.
#
# UNIX is a registered trademark of The Open Group.
#
#==========================================================================
===
#
# Permission is granted to make and distribute verbatim copies of this
document
# provided the copyright notice and this permission notice are preserved on
all
# copies.
#
# Permission is granted to copy and distribute modified versions of this
# document under the conditions for verbatim copying, provided that the
entire
# resulting derived work is distributed under the terms of a permission
notice
# identical to this one.
#
# Permission is granted to copy and distribute translations of this
document
# into another language, under the above conditions for modified versions,
# except that this permission notice may be stated in a translation
approved by
# Tripwire, Inc.
#
# DCM
```

23.3.2 twcfg.txt contents

```
ROOT          =/usr/sbin
POLFILE       =/etc/tripwire/tw.pol
DBFILE        =/etc/tripwire/$(HOSTNAME).twd
REPORTFILE    =/var/lib/tripwire/report/$(HOSTNAME)-$(DATE).twr
SITEKEYFILE   =/etc/tripwire/site.key
LOCALKEYFILE  =/etc/tripwire/ns1-local.key
EDITOR        =/bin/vi
LATEPROMPTING =false
LOOSEDIRECTORYCHECKING =false
MAILNOVIOLATIONS =true
EMAILREPORTLEVEL =3
REPORTLEVEL   =3
MAILMETHOD    =SENDMAIL
SYSLOGREPORTING =false
MAILPROGRAM   =/usr/lib/sendmail -oi -t
TEMPDIRECTORY=/var/tmp/tripwire
```

23.4 Appendix D

Kernel Configuration Settings

Kernel Configuration Option or Section	Value
•Code Manturity level option	
Prompt for development...	Off
•Loadable module support	
Enable loadable module support	Off
•Processor Type and Features	
Processor Family	Pentium III/Celeron
Toshiba Laptop Support	Off
Dell Laptop Support	Off
/dev/cpu/microcode	Off
/dev/cpu/*/msr	Off
/dev/cpu/*/cpuid	Off
High Memory Support	Off
MTRR	Off
•General Setup	
ISA Bus Support	Off
PCI device name database	Off
EISA support	Off
Support for hot-pluggable devices	Off
BSD process accounting	Off
Kernel support for a.out binaries	Off
Power Management Support	
•Advanced Power Management BIOS support	Off
Plug & Play configuration	
ISA Plug and Play support	Off
Block devices	

Kernel Configuration Option or Section	Value
Company SMART2 Support	Off
Company SMART2 Array 5xxx support	Off
Mylex DAC960/DAC1100 PCI Raid controller support	Off
Loopback device support	Off
Network block device support	Off
RAM disk support	Off
Per partition statistics in /proc/partition	Off
Multi-device support (RAID and LVM)	
Multiple devices driver support (RAID and LVM)	Off
Networking options	
Network packet filtering debugging	ON
IP:multicasting	Off
IP:advanced router	Off
IP:tunneling	Off
GRE tunnels over IP	Off
Netfilter Configuration	
•IP tables support <everything is ON except Full NAT>	
•ipchains (2.2-style)	Off
•ipfwadm (2.0-style)	Off
802.1Q VLAN Support	Off
The IPX Protocol	Off
Appletalk Protocol	Off
DECnet Support	Off
802.1d Ethernet Bridging	Off
QoS and/or fair queueing	
•QoS and/or fair queueing	Off
Telephony Support	
Linux telephony support	Off
ATA/IDE/MFM/RLL support	
IDE, ATA and ATAPI Block devices	
•Include IDE/ATAPI TAPE support	Off
•Include IDE/ATAPI FLOPPY support	Off
SCSI support	
SCSI Support	Off
I2O device support	
I2O Support	Off
Network Device Support	
Dummy Net driver support	Off
Bonding driver support	Off
EQL driver support	Off
Universal TUN/TAP device driver support	Off
Ethernet (10 or 100 Mbit) <everything off except...>	
•EISA, VLD, PCI and onboard comtrollers	
•EtherExpressPro/100 support (eepro100,original Becker...)	ON
•EtherExpressPro/100 support (e100,Alternate Intel...)	ON
Ethernet (1000 Mbit) <everything off>	OFF
FDDI driver support	Off
PLIP driver support	Off

Kernel Configuration Option or Section	Value	
PPP driver support	Off	
SLIP driver support	Off	
Wireless LAN (non-hamradio)		
•Wireless LAN (non-hamradio)	Off	
Token Ring devices		
•Token Ring driver support	Off	
Fibre Channel driver support	Off	
WAN interfaces		
•WAN interfaces support	Off	
ATM devices <everything off>	OFF	
Amateur Radio support		
Amateur Radio support	Off	
IrDA (infrared) support		
IrDA subsystem support	Off	
ISDN subsystem		
ISDN support	Off	
Load all symbols for debugging	Off	
Input core support		
Input core suport	Off	
Character Devices		
Support for console on serial port	Off	
Extended dumb serial driver support	Off	
Non-standards serial port support	Off	
Maximum number of UNIX98 PTY's in use		128
I2C support		
•I2C Support	Off	
Mice		
•Bus Mouse support	Off	
•Mouse Support	Off	
IPMI top-level message handler	Off	
Watchdog Cards		
•Watchdog Timer Support	Off	
AMD 768 Random Number Generator	Off	
Intel i8x0 Random Number Generator	Off	
AMD 76x native power management (Experimental)	Off	
/dev/nvram support	Off	
Enhanced Real Time Clock Support	Off	
Double Talk PC internal speechcard support	Off	
Semiens R3964 line discipline	Off	
Ftape, the floppy tape device driver		
•Ftape (QIC-80/Travan) support	Off	
/dev/agpgart (AGP Support)	Off	
Direct Rendering Manager (Xfree86 DRI support)	Off	
Gencom/Advent laptop battery support	Off	
Multimedia Devices		
Video for Linux	Off	
Crypto Hardware Support		
Crypto Hardware Accelerator Support	Off	
Filesystems		

Kernel Configuration Option or Section	Value
Kernel automounter support	Off
Kernel automounter support version 4 support	Off
Reiserfs support	Off
JFS filesystem support	Off
Minix filesystem support	Off
FreeVxFS filesystem support	Off
System V/Xenix/V7/Coherent filesystem support	Off
UDF File System support (read-only)	Off
UFS File System support (read-only)	Off
Network File Systems	
Coda file system support	Off
NFS file system support	Off
NFS server support	Off
SMB filesystem support	Off
NCP filesystem supprt	Off
Partition types	
Advanced Partition selection	Off
Sound	
Sound card support	Off
USB support	
Support for USB	Off
Bluetooth Supprt	
Bluetooth subsystem support	Off
Kernel Hacking	
Kernel debugging	Off

23.5 Appendix E

CDROM Contents

RPMS

glibc-utils-2.3.2-27.9

glibc-profile-2.3.2-27.9

glibc-2.3.2-27.9

glibc-common-2.3.2-27.9

glibc-debug-2.3.2-27.9

glibc-devel-2.3.2-27.9

gnupg-1.2.1-4

initscripts-7.14-1

krb5-libs-1.2.7-14

nscd-2.3.2-27.9

sendmail-8.12.8-5.90

sendmail-cf-8.12.8-5.90

Source

bind-9.2.2

kernel-source-2.4.20-18.9.rpm

openssh-3.6.1p2

openssl-0.9.7b

23.6 Appendix F

23.6.1 Nmap testing

Output of nmap V 3.00 from using the following command to invoke it.

```
nmap -sT -P0 -O -d -vv -T 3 o1.o2.o3.o4
```

This is the final section of output. Previous entries have been culled for brevity.

```
Done with round 2
The Connect() Scan took 2201 seconds to scan 1601 ports.
Warning:  OS detection will be MUCH less reliable because we did not find
at least 1 open and 1 closed TCP port
Wait time is 550ms
For OSScan assuming that port 22 is open and port 34364 is closed and
neither are firewalled
The avg TCP TS HZ is: 100.000000
Interesting ports on blink (xx.xx.xx.xx):
(The 1600 ports scanned but not shown below are in state: filtered)
Port       State       Service
22/tcp     open        ssh
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
OS Fingerprint:
TSeq(Class=RI%gcd=1%SI=1E2079%IPID=Z%TS=100HZ)
T1(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)
T2(Resp=N)
T3(Resp=Y%DF=Y%W=16A0%ACK=S++%Flags=AS%Ops=MNNTNW)
T4(Resp=Y%DF=Y%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)
Uptime 2.010 days (since Tue May 27 16:1
7:02 2003)
TCP Sequence Prediction: Class=random positive increments
                         Difficulty=1974393 (Good luck!)
TCP ISN Seq. Numbers: 1EAF29DE 1E9D54A0 1ECC29BB 1F38D34B 1EED7C7A 1EADAD53
IPID Sequence Generation: All zeros
Final times for host: srtt: 110000 rttvar: 110000  to: 550000
al number of queries to          5 to account for firewalling
Raising ideal number of queries to          5 to account for firewalling
This message repeats 57 times
Raising ideal number of queries to          5 to account for firewalling

Nmap run completed -- 1 IP address (1 host up) scanned in 2207 seconds
```

23.6.2 Nessus testing: first attempt

As shown below, the first run of Nessus against my server, showed up a couple of issues. It detected that the OpenSSH version was down-level, and that the firewall configuration was not complete. I fixed both these problems and ran another test, the results of which are in section 23.6.3 .

Output of Nessus 2.0.6a with default parameters

Nessus Scan Report
This report gives details on hosts that were tested and issues that were found. Please follow the recommended steps and procedures to eradicate these threats.

Scan Details

Hosts which where alive and responding during test	1
Number of security holes found	0
Number of security warnings found	2

Host List

Host(s)	Possible Issue
xx.xx.xx.xx	Security note(s) found

[return to top]

Analysis of Host

Address of Host	Port/Service	Issue regarding Port
xx.xx.xx.xx	ssh (22/tcp)	Security warning(s) found
xx.xx.xx.xx	domain (53/udp)	Security notes found
xx.xx.xx.xx	general/tcp	Security warning(s) found

Security Issues and Fixes: xx.xx.xx.xx

Type	Port	Issue and Fix
Warning	ssh (22/tcp)	You are running OpenSSH-portable 3.6.1p1 or older. If PAM support is enabled, an attacker may use a flaw in this version to determine the existence or a given login name by comparing the times the remote sshd daemon takes to refuse a bad password for a non-existant login compared to the time it takes to refuse a bad password for an existant login. An attacker may use this flaw to set up a brute force attack against

		the remote host. *** Nessus did not check whether the remote SSH daemon is actually *** using PAM or not, so this might be a false positive Solution : Upgrade to OpenSSH-portable 3.6.1p2 or newer Risk Factor : Low CVE : CAN-2003-0190 Nessus ID : 11574
Informational	ssh (22/tcp)	An ssh server is running on this port Nessus ID : 10330
Informational	ssh (22/tcp)	Remote SSH version : SSH-2.0-OpenSSH_3.6.1p1 Nessus ID : 10267
Informational	ssh (22/tcp)	The remote SSH daemon supports the following versions of the SSH protocol : . 1.99 . 2.0 Nessus ID : 10881
Informational	domain (53/udp)	A DNS server is running on this port. If you do not use it, disable it. Risk factor : Low Nessus ID : 11002
Informational	domain (53/udp)	The remote bind version is : [secured] Nessus ID : 10028
Warning	general/tcp	The remote host does not discard TCP SYN packets which have the FIN flag set. Depending on the kind of firewall you are using, an attacker may use this flaw to bypass its rules. See also : http://archives.neohapsis.com/archives/bugtraq/2002-10/0266.html http://www.kb.cert.org/vuls/id/464113 Solution : Contact your vendor for a patch Risk factor : Medium BID : 7487 Nessus ID : 11618
Informational	general/tcp	Remote OS guess : FreeSCO 0.27 (Linux 2.0.38 kernel)

CVE : CAN-1999-0454
Nessus ID : 11268

This file was generated by Nessus, the open-sourced security scanner.

23.6.3 Nessus testing: second attempt

This second run was much more satisfactory. I'm a little puzzled by the fact that the DNS service was not noted as it was in the first run. As no warnings were given against the DNS service in the first run, I'm not too concerned about this.

Output of Nessus 2.0.6a with default parameters

Nessus Scan Report
This report gives details on hosts that were tested and issues that were found. Please follow the recommended steps and procedures to eradicate these threats.

Scan Details

Hosts which where alive and responding during test	1
Number of security holes found	0
Number of security warnings found	0

Host List

Host(s)	Possible Issue
xx.xx.xx.xx	Security note(s) found

[return to top]

Analysis of Host

Address of Host	Port/Service	Issue regarding Port
xx.xx.xx.xx	ssh (22/tcp)	Security notes found
xx.xx.xx.xx	general/tcp	Security notes found

Security Issues and Fixes: xx.xx.xx.xx

Type	Port	Issue and Fix
Informational	ssh (22/tcp)	An ssh server is running on this port Nessus ID : 10330
Informational	ssh (22/tcp)	Remote SSH version : SSH-2.0-OpenSSH_3.6.1p2 Nessus ID : 10267
Informational	ssh (22/tcp)	The remote SSH daemon supports the following versions of the SSH protocol : . 1.99 . 2.0

		Nessus ID : 10881
Informational	general/tcp	Remote OS guess : FreeSCO 0.27 (Linux 2.0.38 kernel)
		CVE : CAN-1999-0454
		Nessus ID : 11268

This file was generated by Nessus, the open-sourced security scanner.

Last Updated: July 18th, 2019

Upcoming SANS Training

Click here to view a list of all SANS Courses

SANS Pen Test Hackfest Europe Summit & Training 2019	Berlin, DE	Jul 22, 2019 - Jul 28, 2019	Live Event
DFIR Summit & Training 2019	Austin, TXUS	Jul 25, 2019 - Aug 01, 2019	Live Event
SANS Riyadh July 2019	Riyadh, SA	Jul 28, 2019 - Aug 01, 2019	Live Event
SANS Boston Summer 2019	Boston, MAUS	Jul 29, 2019 - Aug 03, 2019	Live Event
SANS July Malaysia 2019	Kuala Lumpur, MY	Jul 29, 2019 - Aug 03, 2019	Live Event
SANS Crystal City 2019	Arlington, VAUS	Aug 05, 2019 - Aug 10, 2019	Live Event
SANS Melbourne 2019	Melbourne, AU	Aug 05, 2019 - Aug 10, 2019	Live Event
Security Awareness Summit & Training 2019	San Diego, CAUS	Aug 05, 2019 - Aug 14, 2019	Live Event
SANS London August 2019	London, GB	Aug 05, 2019 - Aug 10, 2019	Live Event
Supply Chain Cybersecurity Summit & Training 2019	Arlington, VAUS	Aug 12, 2019 - Aug 19, 2019	Live Event
SANS Prague August 2019	Prague, CZ	Aug 12, 2019 - Aug 17, 2019	Live Event
SANS Minneapolis 2019	Minneapolis, MNUS	Aug 12, 2019 - Aug 17, 2019	Live Event
SANS San Jose 2019	San Jose, CAUS	Aug 12, 2019 - Aug 17, 2019	Live Event
SANS MGT516 Beta Three 2019	Arlington, VAUS	Aug 19, 2019 - Aug 23, 2019	Live Event
SANS Amsterdam August 2019	Amsterdam, NL	Aug 19, 2019 - Aug 24, 2019	Live Event
SANS Virginia Beach 2019	Virginia Beach, VAUS	Aug 19, 2019 - Aug 30, 2019	Live Event
SANS Chicago 2019	Chicago, ILUS	Aug 19, 2019 - Aug 24, 2019	Live Event
SANS Tampa-Clearwater 2019	Clearwater, FLUS	Aug 25, 2019 - Aug 30, 2019	Live Event
SANS New York City 2019	New York, NYUS	Aug 25, 2019 - Aug 30, 2019	Live Event
SANS Copenhagen August 2019	Copenhagen, DK	Aug 26, 2019 - Aug 31, 2019	Live Event
SANS Hyderabad 2019	Hyderabad, IN	Aug 26, 2019 - Aug 31, 2019	Live Event
SANS Philippines 2019	Manila, PH	Sep 02, 2019 - Sep 07, 2019	Live Event
SANS Brussels September 2019	Brussels, BE	Sep 02, 2019 - Sep 07, 2019	Live Event
SANS Munich September 2019	Munich, DE	Sep 02, 2019 - Sep 07, 2019	Live Event
SANS Canberra Spring 2019	Canberra, AU	Sep 02, 2019 - Sep 21, 2019	Live Event
SANS Network Security 2019	Las Vegas, NVUS	Sep 09, 2019 - Sep 16, 2019	Live Event
SANS Oslo September 2019	Oslo, NO	Sep 09, 2019 - Sep 14, 2019	Live Event
SANS Dubai September 2019	Dubai, AE	Sep 14, 2019 - Sep 19, 2019	Live Event
SANS Rome September 2019	Rome, IT	Sep 16, 2019 - Sep 21, 2019	Live Event
SANS Paris September 2019	Paris, FR	Sep 16, 2019 - Sep 21, 2019	Live Event
SANS Raleigh 2019	Raleigh, NCUS	Sep 16, 2019 - Sep 21, 2019	Live Event
Oil & Gas Cybersecurity Summit & Training 2019	Houston, TXUS	Sep 16, 2019 - Sep 22, 2019	Live Event
SANS San Francisco Summer 2019	OnlineCAUS	Jul 22, 2019 - Jul 27, 2019	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced