Securing Apache 2.0.44 running under Red Hat 8.0 for the Home Network
GCUX Practical Version 1.9

Sean Heare

February 12, 2003

Securing Apache 2.0.44 under Red Hat 8.0 for the Home Network

1. Abstract

Web servers are now ubiquitous. They provide a facile means of distributing information to parties without human intervention. Web Servers often prove difficult to secure. Insecurity is often discovered only after the site is defaced. As time goes on the security risks are being outweighed by the desire for the home user to control content in a more convenient and less costly manner than an ISP or ASP will provide.

This practical will describe a step-by-step process for securing an Apache 2.0.44 web server residing on a Red Hat 8.0 i686 box for a home or small business that requires a web presence solely to provide information. Preparing a server for e-commerce is outside of the scope of this practical.

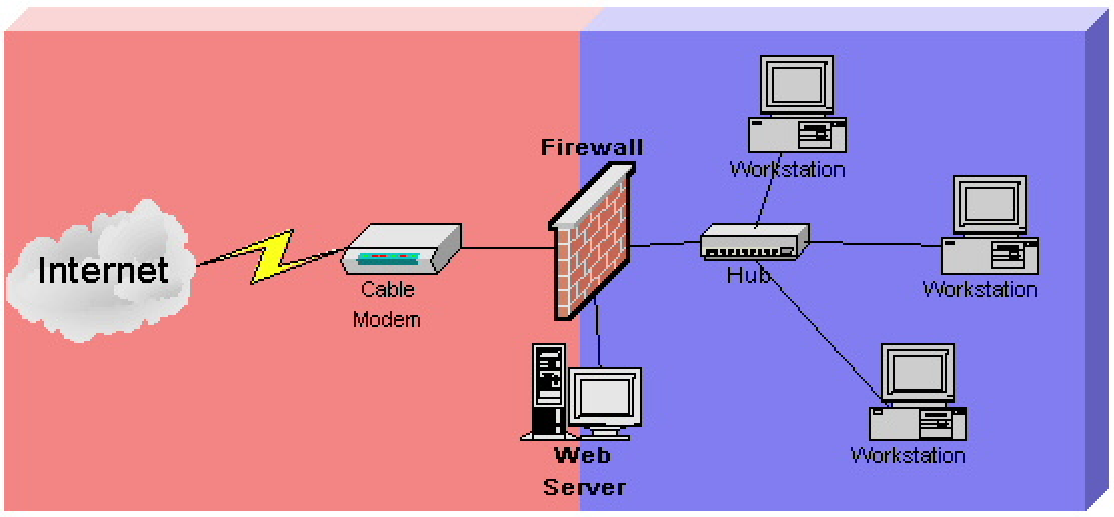
2. Description of the Systems

2.1 Environment

The web server will be running on a home network. Home networks have their own physical and network security issues that will be glossed over. Almost any server in a home environment is by definition physically insecure.

Many home networks now have wireless access points. This one does not. If a wireless network were to be included in this environment it could potentially give access to the web server without the firewall first filtering the traffic. If the wireless network were placed before the firewall that placement would place restrictions on its utility by applying the firewall rules to the wireless network.

In this case the home network looks like this.



A Cable Modem provides uplink to the service provider. Behind the cable modem is a Linux Firewall running IPTables. The Firewall has three NICs, one that connects to the outside world, one for the internal network and one to connect to the web server.

The internal network consists of a couple of Windows PCs, and a Linux workstation. The Web Server is isolated on it's own network connected to the firewall's third NIC via a crossover cable.

Securing the Firewall

Although securing the firewall and the other computers on the network is outside of the scope of this paper, some basic functionality and security regarding the configuration of the firewall in relation to the web server must be discussed. An excellent full treatment on using IPTables to create a secure perimeter firewall along with a sample script for a IPTables firewall utilizing three interfaces can be found at http://iptables-tutorial.frozentux.net/iptables-tutorial.html#INCLUDE.RCDMZFIREWALL. For the purposes of setting up basic connectivity to the Internet through the firewall a few lines will be brought up here.

The firewall is performing IP Masquerading (1 to many network address translation) so that the privately addressed web server can be accessed from the Internet. The following command must be in the firewall script so that the firewall enables masquerading.

```
iptables -t nat -A POSTROUTING -o INET_IFACE -j MASQUERADE
```

This line says to append a rule (-A) into the NAT table (-t nat) after routing (POSTROUTING) for all packets going out INET_IFACE, which is the Ethernet device associated with uplink to the service provider (in my case this is eth0). Finally it says to MASQUERADE the connection (-j MASQUERADE). IP Forwarding also has to be enabled so that packets can be forwarded from one interface to another. That line should look like this.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Once those lines are in place the private networks behind the firewall can reach the Internet masquerading as the public IP address given by the service provider. These next lines have to be added to allow the Internet to be able to access the web server.

```
iptables -A PREROUTING -t nat -p tcp -d INET_IP --dport 80 -j DNAT --to
DMZ_HTTP_IP:80
iptables -A FORWARD -p tcp -d DMZ_HTTP_IP --dport 80 -o DMZ_IFACE-j
ACCEPT
```

These lines say that any TCP traffic inbound on port 80 of the INET_IP should be forwarded to the DMZ_HTTP_IP on the same port. The second line says to forward packets destined for the DMZ_HTTP_IP on port 80 to the DMZ_IFACE, which is where the web server resides.

```
iptables -A FORWARD -p tcp -i LAN_IFACE-d DMZ_HTTP_IP --dport 22 -o
DMZ_IFACE-j ACCEPT
```

Since SSH will only be allowed from the Internal interface and both the server and the internal network are utilizing private addressing the only line needed for SSH to connect from the internal network to the web server is one forwarding the packets destined for the DMZ_HTTP_IP from the LAN_IFACE on port 22 to the DMZ_IFACE.

Again the commands mentioned above are only the commands required to configure access to the web server. The IPTables tutorial referenced above is an excellent starting point for writing a full firewall script.

Being reached from the outside world
Service providers vary in the network services that they offer. In this example, the home is connected to a cable modem. The cable modem provides one public IP address assigned by the provider via DHCP. The IP address remains stable between leases and hence can be assumed to be the one assigned semi-permanently by the cable provider to the house.

So the IP address is relatively stable, still it is best practice to give friends and loved ones something easier to remember than the IP address. If the cable provider is not interested in registering a domain for the IP address there are services on the Internet that will.

Dynamic Domain Name Services (DDNS) provide domain names for hosts whose IP addresses change often and are not registered elsewhere. A list of DDNS providers is maintained at http://www.technopagan.org/dynamic/#TheList

Setting up an account for the web server with a DDNS provider is generally a straightforward affair. The items to have on hand to set up the account are the domain name (or subdomain name) in mind and the public address assigned by the cable company.

A DDNS provider should provide a password for your account and a secure (encrypted) means of registering the domain and changing the password on the account. The password selected for the account should be hard to guess and not dictionary crackable.

2.2 Hardware

The Web Server has the following hardware:
Tyan Thunder 400 BX (BIOS Version AMIBios1.18.02) Motherboard
Pentium II 400 MHz Processor
256 MB SDRAM
Onboard 10/100 NIC
Adaptec AIC-7895 SCSI Controller

Seagate 10 GB HDD (SCSI)
Mitsumi 3.5" FDD
40x CD-ROM
SVGA Video Card
PS/2 2-button Mouse
Generic 105 key Keyboard

This is a run of the mill spare computer that a home user may have on hand. An informational website for a small business should probably be run on a faster server with fault tolerant features like redundant power supply, and RAID support.

2.3 Software

The Web Server will use the following software:

Red Hat 8.0 (psyche)	http://www.redhat.com/downloads/mirrors.html
Apache 2.0.44	http://httpd.apache.org/download.cgi
OpenSSH 3.5.p1	http://www.openssh.com/portable.html

Obviously not all of the Red Hat distribution will be installed (Apache 2.0.40 and OpenSSH 3.4p1 are in the distribution and will not be used)[1]. As the OS installation is performed the packages required will become evident. OpenSSH will be required for remote administration of the web server.

To test the security of the web server the following software will be run from one of the workstations:

Nikto 1.2.3	http://www.cirt.net/code/nikto.shtml
nmapwin 1.3.1	http://www.insecure.org/nmap/nmap_download.html
winpcap 2.3	http://winpcap.polito.it/install/default.htm

All the software used to operate and test the web server is open source. Check the websites for updates to the software. All these links are current as of February 12, 2003.

Nikto is a vulnerability scanner tuned specifically for web servers. nmapwin is the windows port of nmap, a port scanner. Winpcap is a device driver that attaches to windows and adds the ability for windows to promiscuously capture packets off the network. In order for nmapwin to run Winpcap needs to be installed.

1. Red Hat, "redhat.com | Included Packages." URL: http://www.redhat.com/software/linux/technical/packages.html (20 January 2003).

3. Risk Analysis of the System

Key Security Concerns

Since this is a dedicated web server emphasis should be placed on hardening the host to prevent unauthorized access to the host. A hacker with unauthorized access to the host could deface the site, delete all files from the host, or use the host as a launch site for attack on other hosts. The web server will be operating an HTTP and a SSH daemon running on it. SSH will allow the Webmaster a convenient and secure means of copying (scp) new files onto the website.

The server must be accessible by users all over the Internet while at the same time not exposing the host or it's network to loss of availability or information. Confidentiality is not a paramount factor since the information on the server is free to access. Key security concerns for the web server include Trojans, Passwords, Physical Security, Poor Administration, Privilege Escalation and DoS.

Trojans

Since the machine is not being used for any function except serving pages, the main concern regarding Trojans would be in downloading and installing a binary from an open source site that was actually a Trojan. In 2002, hackers compromised at least three open source sites. The hackers placed Trojans on the compromised FTP sites and mirrors.[2] The Trojans, downloaded by unsuspecting users of the software, compromised those users systems in turn.

Where available, the PGP key distributed with the software should be validated. Failing that the MD5 checksum of the software downloaded should be checked against the MD5 checksum listed for the software on the website. This alone is not sufficient precaution to verify that the software is in fact legitimate, the MD5 sum listed on the website could have been changed as the software on the site was hijacked.[3] The MD5 verification plus testing for unexpected open ports on the host the binary was installed on should be sufficient testing for the level of security required.

2. CERT, "CERT® Advisory CA-2002-24 Trojan Horse OpenSSH Distribution." 2 August 2002. URL: http://www.cert.org/advisories/CA-2002-24.html (20 January 2003).

CERT, "CERT® Advisory CA-2002-28 Trojan Horse Sendmail Distribution." 14 October 2002. URL: http://www.cert.org/advisories/CA-2002-28.html (20 January 2003).

CERT, "CERT® Advisory CA-2002-30 Trojan Horse tcpdump and libpcap Distributions." 13 November 2002. URL: http://www.cert.org/advisories/CA-2002-30.html (20 January 2003).

3. CERT, "CERT® Incident Note IN-2001-06." 8 June 2001. URL: http://www.cert.org/incident_notes/IN-2001-06.html (20 January 2003).

Password Policy
Passwords need to be changed at least monthly and should not be dictionary crackable. This means the password should have symbols, numbers and letters in it in such as way as to not make a word that could be guessed. Since this is a home system password management should be trivial and no password auditing should be necessary unless other users are allowed access to the server.

Unexpected Service Vulnerability
Both OpenSSH and Apache have had numerous vulnerabilities exposed in the past year. Often these vulnerabilities provide hackers access to more privilege than which they are entitled. The vendor will often have a patch or update to the vulnerable software ready before point and click exploit software is available. The home user who operates a website should stay updated as to the latest patches and updates made to the software the website runs.

The chances of a home user site being exposed to a zero-day exploit are relatively low due to the nature of the site. However if this were a concern installing a copy of the IDS database for this computer on a separate floppy would be suggested before placing this computer on the network.

Physical Security
The small business owner has more to be concerned about regarding physical security. If someone illegally entered your house or stole your laptop then it is unlikely that the website would be your paramount concern. In either case, the main concern should be theft of a workstation, laptop or backup. If the password or secret shared key used to authenticate SSH is stored on the computer used to access the website then anyone using that computer would be able make changes to your site.

Hardware failures also fall under the purview of physical security. Regular backups of the site should be maintained. This allows the Webmaster to recover quickly from any hardware related failure (or website defacement).

Poor Administration
Poor administration of a server can be its downfall. Leaving services open that are not in use, poorly configured firewall lists, and a failure to maintain updates can all lead to a security breach on the host. The web server should be tested when it is initially brought up to verify that the system is indeed performing exactly what is expected of it and only what is expected of it.

Every time a patch or an update is installed, every time a new service is launched on a server the entire configuration should again be tested. The quote at the top of the cirt.net website says it best “Suspicion breeds confidence.”

Privilege Escalation
If a malicious user succeeds in breaching the outer defenses of the server through gaining local access to the server, exploiting a vulnerability in an application, cracking a local user password, or an error in administration then inner defenses need to be capable of deterring further access to system resources such as root. To implement this layered defense or defense-in-depth strategy, attention needs to be given to what methods a malicious user would use to escalate privilege from an ill gotten user account to the root account for the system. These methods include but are not limited to: exploiting a race condition in the code of a binary or vulnerability in a running local process to change permissions on arbitrary files, running an SUID/SGID binary from the local users home directory to reset the local user's group, abusing cgi scripts on the web server utilizing the scripts to execute malicious code, or breaking out of the chroot()ed jail.

A race condition occurs when two or more programs compete for the same memory resources. A local malicious user could take advantage of such a condition existing between an SUID setting binary and another application to overwrite the memory space of the SUID setting binary with commands that extend root privilege to the malicious user.

Abuse to a local SUID/SGID binary can be mitigated in several ways. Restricting local users umask setting. Disallowing the operation of SUID/SGID binaries from the users home directories through using the nosuid option in /etc/fstab. Possibly using noexec and nodev on the home partitions as well. Finding and knowing what SUID binaries exist on the system and disabling those not in use. Eliminating unused users and groups. Setting the immutable bit on files that control access would further increase the difficulty of exploiting the server. These measures would prevent the local user from using regions they control to gain control over root.

If CGI is not going to be used then the cgi-scripts should be removed. If CGI scripts must be used then avoid allowing script users to use characters that interact with the shell (such as ` > | ; 0x00 & <). Parsing them with a check on all input, escaping input like the characters above should be a good start.

If Perl is used to write the CGI script then avoid utilizing the eval command as it may allow an outside user to execute arbitrary commands. All Perl scripts should be run in tainted mode so that any data outside users give the script is treated in such a way as it cannot take actions to affect the system outside of the script.[4]

4. Callendar, John. "Running a Guestbook", URL:http://www.lies.com/begperl/guestbook.html. (2 February 2003).

The chroot() by it's nature is already hardened most of the OS is missing. However making certain that there are no users in the chroot() besides those needed to run the web server, making certain those users have no ability to login to the system remotely, keeping the web server up to date and only having binaries in the chroot() that are needed to run the chroot() should aid in defending against this threat vector.

DoS Attack

DoS (Denial of Service) attacks are extremely difficult to defend against especially when on a stub network like the home user or small business owner. Although there is no magic bullet to defend against DoS there are several things that can be done to mitigate the risk. Increasing connection tables and decrease TCP timeout times set on web servers.[5] Inserting entries like these into IPTables on the firewall can limit SYN flood and Ping of Death attacks.[6]

```
iptables -A INPUT -p tcp --syn -m limit --limit 1/s -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 1/s
-j ACCEPT
```

If you are subject to a DoS attack then logging the attack will be crucial toward understanding the nature of the attack and then implementing a fix.

5. Scheneider, Bill, "P R E S S R E L E A S E." 14 February 2000. URL:http://www.arena.no/nyheter/wsa-ddos.htm (20 January 2003).

6. Russell, Rusty. "Linux 2.4 Packet Filtering HOWTO: Using iptables." 20 November 2001. URL:http://www.netfilter.org/unreliable-guides/packet-filtering-HOWTO/packet-filtering-HOWTO.linuxdoc-7.html (20 January 2003).

4. Step by step guide

The first step in setting up the server is making sure it is not connected to the network. In a small business setting placing the server on a test network is recommended. In either case, the server (and any configured network device) should not be placed on the live network until it is ready for production. There have been incidents where hosts have been attacked and successfully owned within minutes of being placed onto the network.[7]

Software that needs to be downloaded to the server can either be loaded via CD-ROM or transferred onto the machine via crossover cable. Either way the files to be loaded onto the server need to have their integrity verified before they are placed on the server. It is recommended that a copy of the software be burned to CD as an archive.

4.1 File Verification using PGP and GPG

Each site that the server needs software from provides a public key with which to verify software integrity. You can use either PGP or GPG to verify the software. PGP and GPG both need the same two things to verify the software, the developer's public key and the PGP signature for the file.

Once the actual distributions have been downloaded from the sites the signature for each file downloaded should be retrieved along with the public key for each site. The latest versions of Apache and OpenSSH should be downloaded directly from the site. Apache is the web daemon and OpenSSH is a secure tool for remotely administering the server.

Apache 2.0.44
Signature: http://www.apache.org/dist/httpd/httpd-2.0.44.tar.gz.asc
Public Key: http://www.apache.org/dist/httpd/KEYS

OpenSSH
Signature: ftp://mirrors.rcn.net/pub/OpenBSD/OpenSSH/portable/openssh-3.5p1.tar.gz.sig (mirror)
Public Key: ftp://mirrors.rcn.net/pub/OpenBSD/OpenSSH/portable/DJM-GPG-KEY.asc (mirror)

Once the signature and public key for a file are downloaded to the temp directory the following commands should be issued. The GPG examples here are from a

7. Honeynet Project, "Know Your Enemy: Statistic." 22 July 2001. URL:http://project.honeynet.org/papers/stats/ (20 January 2003).

Windows 98 workstation (denoted C:\gnupg>)[8]. The PGP examples are on a Linux workstation (denoted %).
First import the public key (Apache is the example here)

```
C:\gnupg>gpg --import C:\temp\KEYS

% pgp -ka \tmp\KEYS
```

If you download a public key once you should not have to download it again for a different version of the same kind of software, not if the same person authored the software. Next, verify the signature.

```
C:\gnupg>gpg --verify C:\temp\httpd-2.043.tar.gz.asc

% pgp \tmp\apache_1.3.24.tar.gz.asc
```

In the Windows GNUpg distribution the output looks like this:

```
gpg: Signature made 12/13/02 04:00:52  using RSA key ID FFFFE0F5
gpg: Good signature from "Somebody <someone@somewhere.net>"
gpg:                 aka "somebody@somewhere.net"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
owner.
Primary key fingerprint: FF 14 9F FF FC 12 34 09  DD 6A DB DF FE FA FF
D7
```

PGP will come back with output that looks like this. The output does not have the fingerprint, but should say it was a Good signature.

```
Good signature from user "Somebody <someone@somewhere.net>".
Signature made 2002/12/13 04:00 GMT using 1024-bit key, key ID FFFFE0F5

WARNING: Because this public key is not certified with a trusted
signature, it is not known with high confidence that this public key
actually belongs to: "Somebody <someone@somewhere.net>".
```

To find the fingerprint for the key in PGP type in this command.

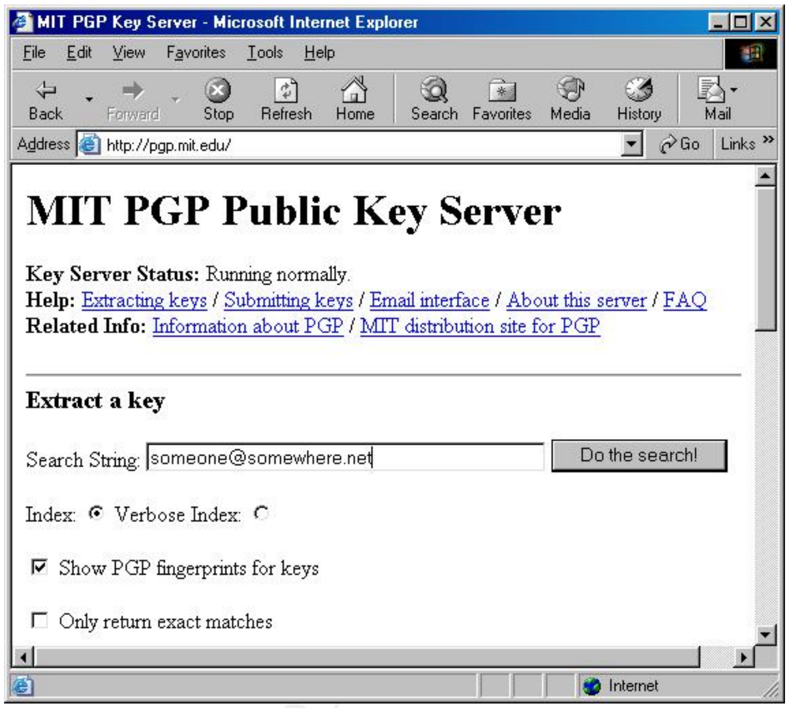
```
% pgp –kvc someone@somewhere.net
```

Somewhere in the output should be a line that reads.

```
Key fingerprint = FF 14 9F FF FC 12 34 09  DD 6A DB DF FE FA FF D7
```

8. A Linux Treatment of GPG can be found at http://www.gnupg.org/gph/en/manual.html#AEN84

Fingerprint Verification
This fingerprint still needs to be verified against an independent third party. Once such party is the MIT PGP Public Key Server (http://pgp.mit.edu/). Take the e-mail address of the developer and place it into the search string field. Check the box marked "Show PGP fingerprints for keys." The page should look like this:


MIT PGP Key Server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail
Address http://pgp.mit.edu/
Go Links
Internet

MIT PGP Public Key Server

Key Server Status: Running normally.
Help: Extracting keys / Submitting keys / Email interface / About this server / FAQ
Related Info: Information about PGP / MIT distribution site for PGP

Extract a key

Search String: someone@somewhere.net [Do the search!]

Index: [✓ ●] Verbose Index: ○

[✓] Show PGP fingerprints for keys

☐ Only return exact matches

Click the "Do the Search!" key. The output that appears should have a signature that matches the signature from the verify command.

```
Public Key Server -- Index ``someone@somewhere.net ''
Type bits /keyID    Date       User ID
pub  2048R/FFFFE0F5 2002/12/13 Somebody <somebody@somewhere.net>
Primary key fingerprint: FF 14 9F FF FC 12 34 09  DD 6A DB DF FE FA FF
D7
                               someone@somewhere.net
```

File Verification using MD5
Red Hat performs file integrity checks a little differently. They have you download MD5SUMs for the distributions that are signed using Red Hat's PGP key. The procedure changes to:

Download the MD5SUM and Red Hat's public key.

For RH 8.0 the Signed MD5SUM is found at:
ftp://ftp.redhat.com/pub/redhat/linux/8.0/en/iso/i386/MD5SUM

Red Hat's Public Key can be copied from this page:
http://www.redhat.com/solutions/security/news/publickey.html

Import the public key and verify that the signature fingerprint matches the fingerprint found on the public PGP key server (http://pgp.mit.edu for example) So far, so good the next part is where you actually use the MD5SUM. For Windows you will have to download an MD5SUM program. Most distribution of Linux have one pre-installed. The one used here runs from the command line in MS-DOS.

```
C:\temp>md5sum psyche-i386-disc1.iso
```

After awhile the PC will output a line that looks like this.

```
d7b16b081c20708dc0dd7d41793a4177 *psyche-i386-disc1.iso
```

Compare it with the MD5SUM in the PGP signed file called MD5SUM from Red Hat. The numbers should match. If they do, congratulations you have verified the distribution is from Red Hat.

4.2 Installing Red Hat 8.0 (Psyche)

During installation most users will install as many packages as possible and allow the software dependencies to be sorted out by the installation program. In building this web server as few packages as possible should be installed. Installing the bare minimum provides two advantages: Less packages to maintain and keep updated with the latest patches and less packages to investigate when something goes wrong with the system.

Media Verification

Once the Linux Distribution is installed on the CD-ROM, the disk should be placed in the server's CD-ROM Drive and the server should be turned on. Once the Installation splash screen comes up there should be a boot prompt at the bottom of the screen. Type "linux mediacheck" into the prompt and hit enter.

```
boot: linux mediacheck
```

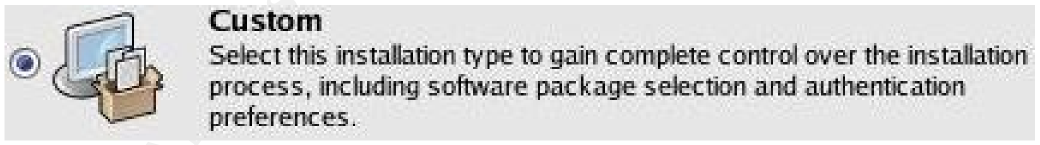
The installation will then verify that the media was not corrupted by the CD-R.

Installation

Once Anaconda, the Red Hat GUI system installer activates there will be a welcome splash screen. Click Next. After that the next three screens will have configuration information that may vary depending on the hardware used.

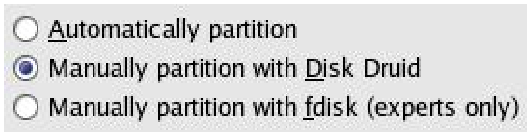
In the case of the hardware listed above, the following choices were made. In each case highlight the selection then click next:

Language Selection: English (English)
Keyboard Configuration: U.S. English
Mouse Configuration: Generic: 2-Button Mouse (PS/2)



Custom
Select this installation type to gain complete control over the installation process, including software package selection and authentication preferences.

The next screen will ask for the Installation Type. In this case the Radio Button marked Custom should be selected. The installation needs to be customized in order to select as few packages as possible.



Automatically partition
[✓ Manually partition with Disk Druid]
Manually partition with fdisk (experts only)

The Disk Partitioning Setup screen will present three options: Automatically Partition, Manually Partition with Disk Druid or Manually Partition with fdisk. Select Manually Partition with Disk Druid. Disk Druid is more forgiving, graphical and easy to use than fdisk while providing the same level of customization. This is how the disk was partitioned on the server. The /boot partition is set at 75MB per the recommendations in the Red Hat installation guide.[9] The <swap> is set at twice the amount of RAM in the system. The /usr partition will be set to mount as read-only after all the installations are completed. The root partition takes the remainder of the drive.

Device	Mount	Space
/dev/sda1/	/boot	75MB
/dev/sda3/	/var	512MB
	<swap>	512MB
/dev/sda2/	/tmp	256MB
/dev/sda4/	/usr	1024MB
/dev/sda6/	/home	256MB
/dev/sda5/	/	

Once the hard drive is configured click next. The Boot Loader Configuration Screen will appear.

LILO vs. GRUB

In normal operation, after the server successfully passes through POST (Power On Self Test), the boot loader performs a set of instructions that allow the operating system to bootstrap. The Red Hat Linux installation offers two different kinds of boot loaders: LILO and GRUB.[10]

GRUB and LILO are pretty much neck and neck in feature offerings, but GRUB will be used on this server for two reasons. With LILO, failing to remember to run LILO after updating the kernel may have serious consequences for the user (they may not be able to access the workstation without a boot disk). Second, LILO

9. Red Hat, "Partitioning Your System." URL:http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/install-guide/s1-diskpartitioning.html (20 January 2003).

10. Keane, Justin, "Madirish.net." 2002 URL:http://www.madirish.net/tech.php?article=95&section=5 (20 January 2003).

seeks a certain physical address on the hard drive to find the boot image. GRUB seeks out the files containing the boot image.[11]

Red Hat recommends using a boot loader password to protect the server.

Without a boot loader password, physical access to the server is all that is needed to pass alternate boot options (like single-user mode) to the kernel that can compromise system security. With a boot loader password in place, the password must first be entered before selecting any non-standard boot options.[12]

Network Configuration
The Network Configuration screen should display only eth0, Highlight eth0 and click Edit. The Edit Interface eth0 window will pop up. Since this web server is on a home network it will have a private address. Uncheck use DHCP then input the IP address. Once these parameters are placed in click OK and then next.

IP Address:	10.0.0.2
Subnet Mask:	255.255.255.0
Hostname:	www.yourdomain.net

Firewall Configuration
This should be disabled. Choose No firewall. The firewall on the edge of the network will handle packet filtering. IPTables will be used on this server.

The language configuration should be left on default. The time zone should be set to local time.

Account Configuration
The root password should be at least 8 characters long and include at least one non-alphanumeric character. Passwords of lesser strength (shorter string, dictionary crackable), can be broken with much greater ease via a brute force attack. A non-root user account should be set up with the same password policy in mind.

Click Next, then Click next again. Since the Authentication Configuration screens defaults of enable MD5 passwords and enable Shadow Passwords are acceptable nothing on that screen needs to be changed.

11. Dobani, Abid Ali, “More About RPM.” URL:http://www.student.math.uwaterloo.ca/~aadobani/Specifics.htm. (1 February 2003).

12. Red Hat, “Boot Loader Configuration.” URL:http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/install-guide/s1-x86-bootloader.html (20 January 2003).

Package Group Selection
Check the Minimal check box on this screen (Scroll Down) and then click the check box for Select Individual Packages. Click Next. On the Select Individual Packages screen follow the following instructions.

Uncheck the following:
Applications > Communications > lrzsz
Applications > Communications > minicom
Applications > Internet > finger
Applications > Internet > ftp
Applications > Internet > lftp
Applications > Internet > lokkit
Applications > Internet > mtr
Applications > Internet > openssh (installing a later package)
Applications > Internet > openssh-clients
Applications > Internet > rsh
Applications > Internet > stunnel
Applications > Internet > talk
Applications > Internet > wget
Applications > Internet > whois
Applications > Systems > isdn4k-utils
Applications > Systems > rdate
Applications > Systems > statserial
Applications > Text > aspell
Applications > Text > pspell
Development > Languages > python
Development > Libraries > pyOpenSSL
Development > Libraries > python-optik
Development > Libraries > rhnlib
Development > Libraries > rpm-python
Documentation > specspo
System Environment > Base > nss_ldap
System Environment > Base > pam_krb5
System Environment > Base > reiserfs-utils
System Environment > Base > up2date
System Environment > Base > yp-tools
System Environment > Daemon > apmd
System Environment > Daemon > esound
System Environment > Daemon > finger-server
System Environment > Daemon > net-snmp
System Environment > Daemon > nfs-utils
System Environment > Daemon > ORBit
System Environment > Daemon > portmap

System Environment > Daemon > ppp
System Environment > Daemon > rp-pppoe
System Environment > Daemon > wvdial
System Environment > Daemon > ypbind
System Environment > Libraries > audiofile
System Environment > Libraries > gnome-libs
System Environment > Libraries > gtk+
System Environment > Libraries > imlib
System Environment > Libraries > krbafs
System Environment > Libraries > libjpeg
System Environment > Libraries > libpng
System Environment > Libraries > libpng10
System Environment > Libraries > libtiff
System Environment > Libraries > libungif
System Environment > Libraries > libwvstreams
System Environment > Libraries > XFree86-libs
System Environment > Libraries > XFree86-Mesa-libGL

The installation should be slightly less than 450 Megabytes. After clicking Next, the next screen will test for any dependencies that are missing. There should not be any. Click next to install. Once installation is complete there will be a prompt to create a bootdisk floppy in case of emergencies. If you already have a bootdisk select no and click next, and then click exit. The system will reboot.

4.3 Installing Apache

Once the server has come back up, login as root and download the Apache software and OpenSSH software. They can be downloaded either by crossover cable or from CD-ROM but not directly from the Internet because the server should still be on a test network if connected to a network at all. Once downloaded the files will need to be extracted from each distribution
Where <filename> is the name of the file downloaded (e.g. httpd-2_0_44.tar.gz).

```
tar -zxvf <filename>
```

Once extracted their will be a folder named httpd-2.0.44 in the directory. Enter that directory (cd httpd-2.0.44). Next Apache will be configured for the system. Type in the following command:

```
./configure --prefix=/usr/local/apache2 –-enable-rewrite
```

The value to which prefix is set is the path where the distribution (in this case Apache) is to be installed. --prefix does not need to be set. Each distribution has a default location to where it will install (apache's is /usr/local/apache2). The --

enable-rewrite option allows the mod_rewrite module to be included in the build. It will be used later.

```
make
```

Building should take some time depending upon your machine. When it finishes compiling the software it is ready to install.

```
make install
```

The steps above are pretty standard for compiling any software from source distributions. To recap, download, verify signatures then:

```
tar –zxvf <filename>
./configure
make
make install
```

4.4 Installing and Setting up OpenSSH

Installing OpenSSH

Once the OpenSSH tarball is downloaded and verified it is ready to be installed. Enter the OpenSSH-3.5p1 directory and issue the following command.

```
./configure --with-tcp-wrappers=PATH
```

This command instructs the script to make the OpenSSH makefile such that it will run using tcp_wrappers. Tcp_wrappers should already be installed on the system and will be configured in a minute to work with SSH. Tcp_wrappers provides security, by acting as a middleman between SSH and the outside world, deciding by IP address who should be allowed in and denying and logging IP addresses that do not match the profile.

```
make && make install
```

The commands above should complete the installation of OpenSSH.

Configuring tcp_wrappers

The tcp_wrappers application uses the hosts.allow and hosts.deny files in the etc directory to determine access to certain applications on the machine. Unlike a firewall tcp_wrappers implicitly allows traffic to pass through that fails to match any of the rules.[13] Due to this fact the hosts.deny should look like this:

13 Red Hat, "Server Security." 2002. URL: http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/security-guide/ch-server.html#S1-SERVER-TCPW-XINETD. (2 February 2003).

```
# hosts.deny   This file describes the names of the hosts which are
#              *not* allowed to use the local INET services, as decided
#                 by the '/usr/sbin/tcpd' server.
ALL: ALL
sshd: ALL: spawn (/bin/echo `date` %c >> /var/log/sshd.log) &
```

This denies everyone and logs attempts to gain access to the server. The hosts.allow should allow the internal network to access the web server. In the case of this server just one workstation on the internal network can access this server (10.0.0.3). Just out of habit log the logins from there too.

```
# hosts.allow  This file describes the names of the hosts which are
#                 allowed to use the local INET services, as decided
#                 by the '/usr/sbin/tcpd' server.
#
ALL: 127.0.0.1
sshd: 10.0.0.3: spawn (/bin/echo `date` %c >> /var/log/sshd.log) &
```

When sshd runs it should be secured from the outside network. The log generated by access should look like this:

```
Sun Feb 2 17:55:54 EST 2003 10.0.0.3
```

Configuring OpenSSH

Before the OpenSSH daemon is run for the first time the /etc/sshd_config must be configured. Protocol 1 fallback should be disallowed because of The SSH CRC32 Compensation Attack Detector Vulnerability[14]

The following line in the sshd_config should be modified from protocol 2,1 to protocol 2:

```
# Allowable protocols - "Protocol 2,1" allows version 1 fallback
Protocol 2
```

This line needs to be changed to allow logging in with keys to work.

```
AuthorizedKeysFile      .ssh/authorized_keys2
```

These settings will make exploiting the system more difficult. Users will have to use the su command to gain privilege. Users with empty passwords are not permitted to access the server via sshd.

```
PermitRootLogon no
```

14. CIAC, "CIACTech02-001: Understanding SSH Exploits." 9 May 2002. URL: http://www.ciac.org/ciac/techbull/CIACTech02-001.shtml (20 January 2003)

```
PermitEmptyPasswords no
UseLogin no
```

Once the sshd_config file is saved, the key pair must be generated.

```
ssh-keygen -t dsa -b 1024
```

After the server generates the keys it will ask where to place the public key:

Enter file in which the save the key (/foo/.ssh/id_dsa):

Hit enter then type in a password to use this key. If no password is typed in then if the key is somehow captured by a hacker (stolen laptop, copied off a workstation) the server will be accessible.

Navigate to the /foo/.ssh directory. Copy the id_dsa.pub to authorized_keys2. Copy the id_dsa file to a floppy to place on computer that will be used to access the server via SSH.

4.5 Hardening the Server

Apply the latest patches

When exploits are discovered in the software installed on a server it is only a matter of time before a hacker will take advantage of those exploits. Fortunately, in most cases, once an exploit is widely known a patch fixing that vulnerability is quick to come forth.

The administrator's responsibility is to subscribe to mailing lists where vulnerabilities and upgrades are announced. Patches should be promptly, downloaded, verified and installed.

Here is the list of mailing lists to subscribe to regarding the software installed during this practical:

https://listman.redhat.com/mailman/listinfo/redhat-announce-list
announce-subscribe@httpd.Apache.Org
http://www.mindrot.org/mailman/listinfo/openssh-unix-announce

Red Hat places all their security updates on this page. Patches are available directly below the exploit descriptions.

https://rhn.redhat.com/errata/rh8-errata-security.html

Keeping server applications fresh with the latest patches and updates is one of the best defenses against the casual hacker. With the following commands the latest RPMs can be downloaded and installed:[15]

```
rpm -qa --qf "%{NAME}-*\n" > updated_rpm_list.tmp
```

The -qa in the parameter lists all the RPMs currently installed. The --qf "%{NAME}-*\n" parameter changes the query format. In this case, the query is formatted to take the name of the archive stripped of it's version number with an asterisk placed on the end. Opening up the updated_rpm_list.tmp you should see a list of filenames with * appended to the end of them. This file needs to be copied to the other linux box for the next step.

The next commands issued will retrieve the RPMs from a mirror.[16] Change the URL in the command to an URL that is close to you. A list of mirrors can be found at http://www.redhat.com/download/mirror.html

15. Kugelberg, Thorsten. "RE: [suse-autoinstall] How to save package sets in suse 8.1?" 22 October 2002. URL: http://lists.suse.com/archive/suse-autoinstall/2002-Oct/0045.html (24 January 2003).

```
for i in `cat updated_rpm_list.tmp`
do
                echo > $i
                ncftpget 
ftp://ftp.tux.org/distributions/redhat/releases/redhat-
8.0/en/os/i386/SRPMS/$i

done
```

The output of running this command should look like this:

```
… lots of files…
tcsh-*
tcsh-6.12-2.src.rpm:                                    815.35 kB
60.68 kB/s
traceroute-*
traceroute-1.4a12-6.src.rpm:                             89.88 kB
55.20 kB/s
unzip-*
unzip-5.50-5.src.rpm:                                     1.02 MB
65.17 kB/s
…many more files…
```

Once the download is complete, verify the files and freshen the installed RPMs

```
rpm -v --checksig *.rpm > validation.report
```

This drops all the checked signatures into a file. All the signatures that checked out OK should look like this.

```
basesystem-8.0-1.src.rpm:
    Header V3 DSA signature: OK, key ID db42a60e
    Header SHA1 digest: OK (5e99680b904092abeef1a658fcd92cfdb2337ccf)
    MD5 digest: OK (5c3e521d45211daf8ae8fe2b704ecc10)
```

If any of them do not list as OK for all the checked signatures then do not freshen the installation using the package. For all those that do, this will freshen the packages:

```
rpm -F *.rpm
```

Freshening upgrades only those packages already installed. If the script has a bug it may download packages that are not actually installed. Freshening will not install these unwanted packages but will update those packages on the system.

16. Hariss, Jeff. "Securing a Linux FreeS/Wan Gateway for Home Use" 1 December 2002. URL: http://www.giac.org/practical/GCUX/Jeff_Harriss_GCUX.pdf (24 January 2003).

Tripwire should be checked before files are freshened and then updated once the freshening is completed.

BIOS Settings
Now that the operating system and all the distributions required to operate the web server are installed, the next step in hardening the server is to prevent unauthorized installations of software. To that end the BIOS must now be accessed. On the hardware listed above pressing the delete key during system startup will access the BIOS. Once in the BIOS three settings need to be changed. Depending on the BIOS you may encounter different settings and some settings may not be available.

Set an administrator password (it might be called a Supervisor Password) to prevent others from accessing the BIOS and making changes.

Set a boot password. This password will appear on boot-up preventing the system from operating until the password is entered. This will prevent an unauthorized user from succeeding in accessing the server by rebooting it. If this password is set an authorized user has to be present every time the server reboots to put in the correct password. Otherwise the server will boot to this prompt and do nothing.

Change the boot order. By default the boot order on the motherboard listed above is CD-ROM, Floppy, Hard Drive. This should be changed to boot up for Hard Drive only.

Securing GRUB
Assuming the BIOS password is defeated the next line of defense would be at the bootloader. From the bootloader an attacker can gain access to root in single user mode, change its configuration by editing the command's interface or use the cat command to reconnoiter.[17] Run /sbin/grub. This will drop into the GRUB interface. The prompt will change to grub>. Type in md5crypt:

```
grub> md5crypt
Password: ********
Encrypted: $4$fGkl/shuj9Rw43x2hoTRfslSlkly7
```

Copy the encrypted hash from the encrypted line. Quit GRUB, then edit the /boot/grub/grub.conf file. Add the line password --md5 <encrypted hash> to the file right below the timeout parameter. It should look like this:

17. Red Hat "BIOS and Boot Loader Security." URL: http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/security-guide/s1-wstation-boot-sec.html#S3-BOOTLOADER-GRUB. (22 January 2003).

```
default = 0
timeout = 10
password --md5 $4$fGkl/shuj9Rw43x2hoTRfslSlkly7
```

Save the file.

Configuring IPTables
IPTables is a packet filter. It provides tighter control over what packets get filtered and how so it will be used instead of the Firewall offered in Firewall Configuration. It also provides stateful connection tracking which means that using parameters inside the packet IPTables can discern whether the packet is part of an established connection, possibly related to an established connection, a new connection, or invalid if it fails to meet any of these states.

When setting up a firewall the first thing to consider is what ports need to be opened and what IP ranges require access to them. In this case the ports that need to be opened are HTTP (TCP 80), and SSH (TCP 22) and they need to opened from anywhere.

The following script will be loaded on the server. Scott Morizot's "Easy Firewall Generator for IPTables" (version 1.10) generated portions of this script.[18] His script generator uses principles based on Oskar Andreasson's IPTables tutorial.[19]

```
#!/bin/sh
#
# The network this web server is on already has a firewall the main
# purpose of placing IPTables on this server is to drop traffic that
# has no business bothering the web server but might exist on the
# network segment anyway even after passing through a filter.
#

# Internet Interface
INTERNET_INTERFACE="eth0"
INTERNET_IP="10.0.0.2"

# Localhost Interface
LOCALHOST_INTERFACE="lo"
LOCALHOST_IP="127.0.0.1"

# Save and Restore arguments handled here
if [ "$1" = "save" ]
```

18. Morizot, Scott. "Easy Firewall Generator for iptables" 24 January 2003. URL: http://morizot.net/firewall/gen/ (24 January 2003).

19. Andreasson, Oskar. "Iptables Tutorial 1.1.16." 2002. URL: http://iptables-tutorial.frozentux.net/iptables-tutorial.html (24 January 2003).

```
then
                echo -n "Saving firewall to /etc/sysconfig/iptables ...
"
                iptables-save > /etc/sysconfig/iptables
                echo "done"
                exit 0
elif [ "$1" = "restore" ]
then
                echo -n "Restoring firewall from /etc/sysconfig/iptables
... "
                iptables-restore < /etc/sysconfig/iptables
                echo "done"
                exit 0
fi

# Enable Anti-IP Spoofing
echo "1" > /proc/sys/net/ipv4/conf/eth0/rp_filter

echo "Flushing Tables ..."
# Open policies wide open (So they can be locked down later)
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT

# Flush all rules
iptables -F

# Erase all non-default chains
iptables -X

if [ "$1" = "stop" ]
then
                echo "Firewall completely flushed!  Now running with no
firewall."
                exit 0
fi


# Set Policies

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# User-Specified Chains

# The following user-specified chains reduce the number of rules each
# packet must traverse.  This improves performance.

echo "Create and populate custom rule chains ..."

#
# The iptables –N command followed by a name creates a new
# custom chain.  The chains here are placed in this order
# to dismiss undesired traffic first.  Note the egress
# filter on udp and tcp.
#

iptables -N bad_packets
iptables -N bad_tcp_packets
iptables -N icmp_packets
iptables -N udp_inbound
iptables -N udp_outbound
iptables -N tcp_inbound
iptables -N tcp_outbound

###########################################################
# Populate User Chains
#

# bad_packets chain

# Stateful filtering at work.
# The bad_packets user chain logs, marks and drops packets deemed
# invalid by the state filter.

# Hackers reported by Swatch to be using known attack sigs.
# Further explanation in the Swatch section
for i in `cat /chroot/usr/local/apache2/logs/hosts.deny`
do
iptables -A INPUT -p ALL -s $i -j LOG \
   --log-prefix "Hacker:"
iptables -A INPUT -p ALL -s $i -j DROP
echo "Services denied to $i for hacking."

done

iptables -A bad_packets -p ALL -m state --state INVALID -j LOG \
    --log-prefix "Invalid packet:"
iptables -A bad_packets -p ALL -m state --state INVALID -j DROP

# This line forwards all TCP packets dropped here into the
# bad_tcp_packets chain for further scrutiny.
iptables -A bad_packets -p tcp -j bad_tcp_packets

# Send everything else back to the main chain.
iptables -A bad_packets -p ALL -j RETURN

# bad_tcp_packets chain

# Log, mark and drop any new connection
#without a SYN packet leading it
iptables -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j LOG
\
    --log-prefix "New not syn:"
iptables -A bad_tcp_packets -p tcp ! --syn -m state --state NEW -j DROP
```

```
# Rate Limiting new SYN connections in an attempt to limit SYN Flood
iptables -A INPUT -p tcp --syn -m limit --limit 1/s \
--destination-port 80 -j ACCEPT
iptables -A INPUT -p tcp --syn -m limit --limit 1/s \
--destination-port 22 -j ACCEPT
iptables -A INPUT -p tcp --syn -m limit --limit 1/s \
--destination-port 443 -j ACCEPT

# Send everything else back to the main chain.
iptables -A bad_tcp_packets -p tcp -j RETURN

# icmp_packets chain
# Limit these connections to 1/sec an attempt to circumvent DoS
# attacks.
iptables -A INPUT -p icmp --icmp-type echo-request -m limit \
--limit 1/s -j ACCEPT
iptables -A icmp_packets -p ICMP -s 0/0 --icmp-type 11 -m limit \
--limit 1/s -j ACCEPT

# No reason to give anyone any information they don't need.
iptables –A icmp_packets –p ICMP -j REJECT --reject-with \
icmp-host-unreachable

# Just in case the above rule gets modified in the future it is
# good practice to mop up any packets not soaked into the chain
# and return them the main chain.
iptables -A icmp_packets -p ICMP -j RETURN

#
# TCP & UDP
#

# udp_inbound chain
# Drop netbios calls. Windows machines are chatty.
iptables -A udp_inbound -p UDP -s 0/0 --destination-port 137 -j DROP
iptables -A udp_inbound -p UDP -s 0/0 --destination-port 138 -j DROP

# Send everything else back to the main chain.
iptables -A udp_inbound -p UDP -j RETURN

# udp_outbound chain
# If this server were compromised it may be used for DDoS attack.  An
# egress filter here on commonly chosen ports may prevent it from
# flooding the network.

# No match, so ACCEPT
iptables -A udp_outbound -p UDP -s 0/0 -j ACCEPT

# tcp_inbound chain
# Web Server (HTTP)
iptables -A tcp_inbound -p TCP -s 0/0 --destination-port 80 -j ACCEPT

# Uncomment the next line if you are using a Secure Web Server (HTTPS)
```

```
#iptables -A tcp_inbound -p TCP -s 0/0 --destination-port 443 -j ACCEPT

# sshd
iptables -A tcp_inbound -p TCP -s 0/0 --destination-port 22 -j ACCEPT

# Send everything else back to the main chain.
iptables -A tcp_inbound -p TCP -j RETURN

# tcp_outbound chain
# Egress filtering for TCP.

# No match, so ACCEPT
iptables -A tcp_outbound -p TCP -s 0/0 -j ACCEPT

#####################################################
#
# INPUT Chain
#
echo "Process INPUT chain ..."

# Allow all on localhost interface
iptables -A INPUT -p ALL -i $LOCALHOST_INTERFACE -j ACCEPT

# Drop bad packets
iptables -A INPUT -p ALL -j bad_packets

# Inbound Internet Packet Rules
# Accept Established Connections
iptables -A INPUT -p ALL -i $INTERNET_INTERFACE -m state --state
ESTABLISHED,RELATED \
     -j ACCEPT

# Route the rest to the appropriate user chain
iptables -A INPUT -p TCP -i $INTERNET_INTERFACE -j tcp_inbound
iptables -A INPUT -p UDP -i $INTERNET_INTERFACE -j udp_inbound
iptables -A INPUT -p ICMP -i $INTERNET_INTERFACE -j icmp_packets

# Drop broadcasts without logging.
iptables -A INPUT -p ALL -d 255.255.255.255 -j DROP

# Anything that reaches here in the input chain should be logged
# labeled and limited.
iptables -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
     --log-prefix "INPUT packet died at end of chain: "

#######################################################
#
# OUTPUT Chain
#
echo "Process OUTPUT chain ..."

# Generally trust the firewall on output
# However, invalid icmp packets need to be dropped
# to prevent a possible exploit.
iptables -A OUTPUT -m state -p icmp --state INVALID -j DROP

iptables -A OUTPUT -p TCP $LOCALHOST_INTERFACE -j tcp_outbound
iptables -A OUTPUT -p UDP $LOCALHOST_INTERFACE -j udp_outbound
iptables -A OUTPUT -p ALL -s $LOCALHOST_IP -j ACCEPT
iptables -A OUTPUT -p ALL -o $INTERNET_INTERFACE -j ACCEPT

# Anything that reaches here in the input chain should be logged
# labeled and limited.

iptables -A OUTPUT -m limit --limit 3/minute --limit-burst 3 -j LOG \
     --log-prefix "OUTPUT packet died at end of chain: "
```

To load the script into the iptables copy the script above to a text file and save it as rc.firewall. Make certain to change the IP address listed in the INTERNET_INTERFACE parameter to the IP for your server. Copy it onto the server into the /etc/rc.d/ directory. Run the following commands; the first to remove carriage returns that Windows applies but Unix may not like, the second to adjust permissions:

```
dos2unix /etc/rc.d/rc.firewall
chmod u+x /etc/rc.d/rc.firewall
```

The rc.local file has to be modified.

```
echo "Loading firewall rules…"
/etc/rc.d/rc.firewall
```

To manually launch it without reloading run the following command:

```
./rc.firewall
```

The following output should look like the following:

```
Flushing Tables…
Creating and populate custom rules and chains...
Services denied to 10.0.117.153 for hacking.
Process INPUT Chain…
Process OUTPUT Chain…
```

There may not as yet be any hosts denied access for hacking by Swatch as of yet. Running iptables –L should yield a list of the chains currently in place. A copy of the results of that command on the server is in Appendix A.

Sendmail
Mail will not be sent to the server. Since this is the case sendmail needs to be set so that it does not run as a daemon but that it does send out the logs as e-mail. The /etc/sysconfig/sendmail configuration file must be edited to read.[20]

```
Daemon=no
Queue=4h
```

This will result in sendmail sending any mail in queue once every four hours. Also it closes an unnecessary server. To make certain it does not get reactivated it should be removed from the rc files by the following command:

```
chkconfig --del sendmail
```

Password Aging
Whether there are other users or not, passwords for accounts allowing access to the server should be changed periodically. If a hacker has managed to access one of the passwords on the server then forcing the password to change will once again deny the hacker access to the system.

To do this, issue the chage command. For example:

```
chage -M 30 foo
```

Forces the foo's password to expire once every 30 days.

Tripwire
Tripwire is an intrusion detection system. Tripwire verifies the integrity of the existing File System against a baseline set by the administrator. This digitally signed database consists of encrypted information regarding the various system files, system binaries and various other important files and directories that you wanted to protect.

Tripwire must be installed and must create a baseline snapshot before the server is connected to the production network. This will maintain full assurance that a hacker has not compromised the file system before the baseline was taken. The snapshot is based on rules that are set in the policy file.

Tripwire is distributed with Red Hat 8.0 (on the second CD) and thus available as an RPM on the disks or the website. To install it from the CD issue the following command.[21]

20. Berninger, John. "NCSU Realm Kit for Red Hat Linux 7.3 Guide - Securing the Machine." 2002 URL: http://www.linux.ncsu.edu/realmkit/usersguide/x197.html (20 January 2003).

```
rpm -Uvh /mnt/cdrom/RedHat/RPMS/tripwire-2.3.1-2.rpm
```

Once the installation is complete run the /etc/tripwire/twinstall.sh script. It will prompt for a site key passphrase and a local key passphrase. The site key is used to sign Tripwire policy and software configuration files. The local key signs the database files. According to Red Hat the following should be considered when selecting these passphrases.[22]

- ✓ Use at least eight alphanumeric and symbolic characters, but for each password do not exceed 1023.
- ✓ Do not use quotes in a password.
- ✓ Make the Tripwire passwords completely different from the root or any other password for the system.
- ✓ Use unique passwords for both the site key and the local key.

Great care must be taken not to forget the passphrases. Without the passphrases, the database must be reinitialized because the tripwire files are unusable.

Once the passphrases have been selected Tripwire will prompt for the site passphrase to be entered and will generate a tw.cfg for inspection purposes. This should be deleted after inspection to prevent hackers from gleaning information from it. Tripwire will prompt for the site passphrase once again and generate a tw.pol. This file will be edited later but a test run should be completed first. Enter the following command.

```
tripwire --init > output.txt
```

There will be many errors. The minimal configuration installed causes the error messages. The tripwire default policy is based on a configuration loaded with everything. Initializing at this point lends the opportunity to identify which entries in the policy file can be trimmed to reduce error messages. The output.txt stores the full report within which is the list of files that should be removed. Here is a perl script that will parse those filenames out of the output.txt

```
#!/usr/bin/perl

use IO::File;
```

21. Red Hat, "Install the Tripwire RPM." URL: http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/ref-guide/s1-tripwire-install-rpm.html (20 January 2003).
22. Red Hat, "Customizing Tripwire." URL: http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/ref-guide/s1-tripwire-initialize.html (20 January 2003).

```
$fh_twreport = new IO::File $ARGV[0], "r";
$fh_list = new IO::File "> list";
@report = <$fh_twreport>;
@filenames = grep /Filename:/, @report;
foreach $line(@filenames){
        @array = split(/:/, $line);
        @badfilearray[x] =  $array[1];
        print $x.":".$badfilearray[x];
        print $fh_list $array[1];
        $x++;
}
```

Save this perl script as parseoutput.pl then run the following command:

```
parseoutput.pl output.txt
```

A new file named list should have a list of the files that need to be removed from the twpol.txt

Once the unneeded entries have been removed from the twpol.txt file then the extra entries of files that need to be monitored but are not included in the standard list are added. Apache, OpenSSH and firewall files specifically need to be added to the database.

For this installation the following files were selected so that any changes in the essential files inside the chroot() would be noticed. You may need to add more or less files to this list. Note the emailto parameter is optional.

```
#Apache Web Server chroot

(
   rulename = "Apache Web Server Files",
   severity = $(SIG_HI),
   emailto  = yourname@yourdomain.com
)

{

/chroot/bin/sh                                  ->$(SEC_CRIT);
/chroot/dev/null                                ->$(SEC_CRIT);
/chroot/etc/                                    ->$(SEC_CRIT);
/chroot/etc/group                               ->$(SEC_CRIT);
/chroot/etc/hosts                               ->$(SEC_CRIT);
/chroot/etc/localtime                           ->$(SEC_CRIT);
/chroot/etc/passwd                              ->$(SEC_CRIT);
/chroot/etc/resolv.conf                         ->$(SEC_CRIT);
/chroot/etc/shadow                              ->$(SEC_CRIT);
/chroot/usr/local/apache2/bin/apachectl         ->$(SEC_CRIT);
/chroot/usr/local/apache2/bin/httpd             ->$(SEC_CRIT);
/chroot/usr/local/apache2/conf/httpd.conf       ->$(SEC_CONFIG);
/chroot/usr/local/apache2/htdocs/index.html     ->$(SEC_CONFIG);
#You may wish to place other files in here that are
#important to your website
}

#OpenSSH
(
   rulename = "OpenSSH Files",
   severity = $(SIG_HI),
   emailto  = yourname@yourdomain.com
)

{

/usr/local/sbin/sshd                            ->$(SEC_CRIT);
/etc/log.d/conf/services/sshd.conf              ->$(SEC_CRIT);
/usr/local/etc/sshd_config                      ->$(SEC_CRIT);

}

#Firewall
(
   rulename = "Firewall",
   severity = $(SIG_HI),
   emailto  = yourname@yourdomain.com
)

{

/etc/rc.d/rc.firewall                           ->$(SEC_CRIT);

}
```

Once the entries have been pruned and added the Tripwire database needs to be updated so that it has a list of what it should be monitoring. To do this issue the following command:[23]

```
tripwire -m p -Z low twpol.txt
```

Run an integrity check.

```
tripwire --m c > output.txt
```

This output.txt should be free of errors. This means that a baseline snapshot of the database is ready to be copied off the computer. It should be transferred onto media that can be write-protected either a CD or a floppy. Failing that a copy should be transferred to another computer. Tripwire writes the database to

23. Williams, C. "Lab 3 | Final Part." URL: http://www.eecis.udel.edu/~cwilliam/cis479/Lab-IDS-B.htm (20 January 2003).

/var/lib/tripwire/[hostname].twd The Tripwire binary should be placed on the other media as well to ensure that if the server is compromised a copy of the binary and the database can be dropped onto the server.

```
/mount/floppy
cp /var/lib/tripwire/[hostname].twd /mnt/floppy
mv /etc/twpol.txt /mnt/floppy
chmod 700 /mnt/floppy/[hostname].twd /mnt/floppy/twpol.txt
umount /mnt/floppy
```

Note that the permissions on the files on the floppy have been changed so that only root can read them. Do not forget to write protect the floppy by switching the write protect tab to the write protect state. Place the floppy in a safe place, as it will be used in the maintenance cycle later.

SUID/SGID
Programs on a Unix system often require root privileges when performing some activities. If the programs are not being run by root then they will need to temporarily escalate the privileges of the user running them to root. These programs are set to have their set user ID (SUID) or set group ID (SGID) bit set. In a directory entry they look like this. Notice the s under execute for root.[24]

```
-rwsr-xr-x    1 root     root        19132 Aug 29 16:56 su
```

Hackers often exploit these programs to gain root access. As many of them as possible should be disabled. To view a list of them run the following command:[25]

```
find / -perm -4000 > suid 2>&1 ; find / -perm -2000 > sgid 2>&1
```

The first part asks find to look in root on down for files with the suid bit set (4000). The second part does the same for the sgid bit (2000). The full permissions list is furnished for academic curiosity.[26]

```
---s--s--t 7000 setuid, setgid, sticky
---s--s--- 6000 setuid, setgid
---s-----t 5000 setuid, sticky
---s------ 4000 setuid
------s--t 3000 setgid, sticky
---s------ 2000 setgid
---------t 1000 sticky
---------- 0000 none
```

24. Schneider, Neil. "SUID." 2002 URL: http://www.linuxgeek.net/index.pl/suid. (4 February 2003).

25. Gateways. "http://onicrom.com/info/security.txt." URL:http://onicrom.com/info/security.txt. (4 February 2003).

26. volatile, "Setuid/Setgid Tutorial." 15 January 2002. URL: http://neworder.box.sk/newsread.php?newsid=2380. (4 February 2003).

This command will create two files suid and sgid in the directory in that the command was run. In the suid file is a list of programs with the set user ID bit set, and in the sgid file is a list of programs with the set group ID bit set. The lists should look like this:

```
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/newgrp
/usr/bin/at
/usr/bin/passwd
/usr/bin/sudo
/usr/bin/crontab
/usr/libexec/pt_chown
/usr/sbin/usernetctl
/usr/sbin/userhelper
/usr/sbin/traceroute
/usr/local/libexec/ssh-keysign
/bin/mount
/bin/umount
/bin/su
/sbin/pwdb_chkpwd
/sbin/unix_chkpwd
```

Remove the suid or sgid bit as needed. To remove the setuid bit on traceroute (or any file):[27]

```
chmod u-s /usr/sbin/traceroute
```

To remove the setgid bit on /usr/bin/slocate:

```
chmod g-s /usr/bin/slocate
```

After running that command an ls –ll will reveal that the bit was removed (the s should be an x). If you remove the permission and it is actually needed replacing it as simple as replacing the – in the above commands with a +.

As a general rule if you don't use a program and it has one of these bits set then the bit should be removed. If there are other users on the system then disable the s bits on program that they should not run as root.

Check that tripwire has each of the files that remain with s bits enabled listed in the twpol.txt. If they are not then manually add them under a custom category.

27. Schneider, Neil. "SUID." 2002 URL: http://www.linuxgeek.net/index.pl/suid. (4 February 2003).

Securing /etc/passwd and /etc/group
The /etc/passwd file has several users within it that can be disabled. Disabling the unused users in /etc/passwd impedes the ability of hackers to exploit those unused accounts.

This is what the /etc/passwd file looks like on this server before editing. The first action taken should be to save a copy of the unedited /etc/passwd file to /etc/passwd.old and a copy of the /etc/group to /etc/group.old

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
...
```

This is /etc/group note each one of these files been truncated for presentation here.

```
root:x:0:0:root:/root:/bin/bash
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
tty:x:5:
disk:x:6:root
lp:x:7:daemon,lp
mem:x:8:
kmem:x:9:
wheel:x:10:root
mail:x:12:mail
news:x:13:news
...
```

Several of these users and groups can be removed safely but many are important, so please use caution when deleting users and groups. Since printing, FTP, gopher, news, and X-windows are not going to be used on this server the users lp, ftp, gopher, news and games can be removed without harm.

```
userdel gopher
```

The command userdel gopher would delete the user gopher. The groups associated with these users can also be removed. In the case of the users listed above these would be lp, ftp, gopher, news and games as well. The groupdel command removes the group gopher.

```
groupdel gopher
```

Be very judicious in deleting users and groups. The following script will yield some idea as to what files on the server are owned by whom but even if a user owns no files it does not mean that the user does not have some other function. Also the more files and functionality that are installed on the server the more likely it is that the server may need one of these default names.

```
for name in $(cut -d: -f1 /etc/passwd) ; do
 echo -n "Number of objects with owner $name: "
 find / -user $name –xdev | wc -l
done
```

Running this script should yield results as follows.

```
Number of objects with owner root:   48006
Number of objects with owner bin:       0
Number of objects with owner daemon:       0
...
```

If all else fails the /etc/passwd.old and /etc/group.old should be moved off the server onto a floppy for safekeeping. This way, in the event catastrophe strikes and an unintentionally deleted group or user is unable to be restored then restoring these files can restore the server. Try using the adduser and addgroup commands first.

Finally, the chattr or change attribute command can add an immutable bit to files preventing their being overwritten or deleted. To protect the group and passwd files and their shadows issue the following commands.[28]

```
chattr   +i /etc/passwd
chattr   +i /etc/shadow
chattr   +i /etc/group
```

28. Madhusudan and Mourani, G. "Special Accounts." 2000 URL: http://www.tldp.org/LDP/solrhe/Securing-Optimizing-Linux-RH-Edition-v1.3/chap5sec42.html (6 February 2003).

```
chattr   +i /etc/gshadow
```

/etc Directory Permissions
Since there is no reason for users without root privilege to see the inner workings of the server the /etc directory should be set such that users can only execute from it.

```
chmod 711 /etc
```

This way only root will be able to see the /etc directory.

Core Dumps
Core dumps normally occur when an application writes to an out-of-bound memory area. A hacker may initiate a dump on purpose using a buffer overflow to obtain data in clear text such as passwords.[29]

Core dumps pose a threat to the server. Since no development that would involve debugging with a core dump occurs on this server, the allowable core file size should be set to 0. To do this, utilize the ulimit command.

```
ulimit –c 0
```

To verify that the change went through issue the ulimit –a command, the core file size should be zero:

```
core file size        (blocks, -c) 0
data seg size         (kbytes, -d) unlimited
file size             (blocks, -f) unlimited
max locked memory     (kbytes, -l) unlimited
max memory size       (kbytes, -m) unlimited
open files                    (-n) 1024
pipe size          (512 bytes, -p) 8
stack size            (kbytes, -s) 8192
cpu time             (seconds, -t) unlimited
max user processes            (-u) 1536
virtual memory        (kbytes, -v) unlimited
```

Mounting /usr as Read-Only and setting /home to nosuid
Once all the software is installed no changes should be made in /usr. Mounting /usr as read-only adds another step that a hacker must overcome before being able to make changes to /usr.

29. Chung, Adrian. "Core Dump Files and What To Do About Them." 2002. URL: http://startlinux.co.nz/articles/article_153.php. (4 February 2003).

The hacker wants to do this because most of the tools that conceal a hacker's presence (rootkits) replace binaries on the server that lurk in the /usr partition. To mount /usr as read only the first step involves opening the /etc/fstab file. The one for this server looks like this:

```
LABEL=/           /                   ext3    defaults         1 1
LABEL=/boot       /boot               ext3    defaults         1 2
none              /dev/pts            devpts  gid=5,mode=620   0 0
LABEL=/home       /home               ext3    defaults         1 2
none              /proc               proc    defaults         0 0
none              /dev/shm            tmpfs   defaults         0 0
LABEL=/tmp        /tmp                ext3    defaults         1 2
LABEL=/usr        /usr                ext3    defaults         1 2
... (There's more to this)
```

Since /usr is the partition to be changed that line needs the word defaults changed to ro. ro tells the system to mount the partition with that tag as read-only. The /usr line in the file should now look like this.

```
LABEL=/usr        /usr                ext3    ro               1 2
```

/home should be set to disallow the running of SUID binaries from the home directories of users. This prevents a hacker who has broken into a user account on the server from using it and an SUID binary to escalate the compromised user to root. To edit the /home line; change the word default on that line to nosuid. It should then look like this:

```
LABEL=/home       /home               ext3    nosuid           1 2
```

Save the /etc/fstab file. Upon the next reboot /usr should come up as read-only. This is great until the next patch or binary needs to be installed then /usr will need to be writable again. In order to write on the /usr partition issue this command:

```
mount –o remount,rw /usr
```

Once the installation is completed, remount /usr as read-only again:

```
mount –o remount,ro /usr
```

4.6 Hardening Apache

CGI
According to HP, poorly written CGI code is the #1 vulnerability allowing web server break-ins.[30] Since the server being built is an informational server with no changing information on it no CGI is required. A short discussion of why CGI should be disabled unless you absolutely need it is in order.

CGI (Common Gateway Interface) allows a Webmaster to hook a web page to an internal application for the world to use. Since it is accessible from the world it is a logical place from which to launch an attack since the hacker already has a foothold. If the user happens to post a string into the CGI that would execute a command within the system then if the data does not have such characters properly escaped it may actually execute the command posted.

Essentially when writing a CGI script make certain that it treats all data it receives as if it is bad data. With Perl compiling with the –t command will treat all data as if it has been tainted. Regular expressions will have to escape any potential characters that may issue a command before the data will be used within the server.

CGI should be turned off if it is not needed. Hardening the httpd.conf will accomplish that. After that CGI and other files installed by default will be removed to further protect the server.

Hardening http.conf
The httpd.conf needs to be modified so that Apache presents the smallest possible opportunity to hackers. By default, Apache reveals it's version number, the OS it is using, its IP and port:

```
Apache/2.0.44 (Unix) Server at 10.0.0.2 Port 80
```

By editing the ServerToken parameter from its default of Full to read:

```
ServerToken Prod
```

And then restarting the server Apache will yield the least information possible.

```
Apache Server at 10.0.0.2 Port 80
```

But more can be done, this server will not allow Apache to generate custom error messages that reveal the anything about the server. Right below the ServerToken parameter in the httpd.conf there is the ServerSignature parameter.

30. Hewlett-Packard. "HP WebWise MPE/iX Secure Web Server." 10 December 2002. URL: http://jazz.external.hp.com/src/webwise/. (3 February 2003).

Leaving this on allows Apache to generate error screens. This also means Apache will at a minimum reveal that the server is running Apache on port 80 and at what IP address. Change this parameter to read:

```
ServerSignature Off
```

Once that is done custom error pages must be generated. Creating custom web pages with error messages is outside of the scope of the paper. Once they are created though Apache must be told where to find them. For each error generated web page the term ErrorDocument, the HTTP Error Number and the path to the error generated web page must be issued. For example:

```
ErrorDocument 404 "/error/404.html"
```

This will point 404 messages to a custom error page I created in the error directory of Apache2. A full list of errors can be found at http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html. If performing this step create an error page for 403,404 and 500 at a minimum. They are the most common errors.

The Options directive configures what features are available to users in a particular directory. Since there is no CGI on this server and the DocumentRoot directory should be as hardened as possible the following lines need to be changed:

```
<Directory "/usr/local/apache2/htdocs">
               Options None
               AllowOverride None
               Order allow,deny
               Allow from all

</Directory>
```

Setting the Options Directive to None forbids the server from using SSI, CGI, Symlinks, DirectoryIndexing or Content Negotiated Multi-views.[31] Each of these could potentially be used to either learn more about the file system or attack the server itself.

AllowOverride is set to None. This forbids htaccess files in subdirectories from overriding the Options directives set in the httpd.conf.

31 Apache. "core – Apache HTTP Server." URL:http://httpd.apache.org/docs-2.0/mod/core.html#options. (3 February 2003).

The Order directive and the Allow directive are from the mod_access module as opposed to the core module. They determine who can access the directory in question. Order allow, deny means that allow statements are read in first followed by an implicit deny, much like an access list. Reversing them would result in implicitly accepting connections and reading deny entries first.

Allow from all means that access is opened to everyone. The Allow and Deny directives can be restricted to a domain name, IP address, IP range or environment variable.

Mod_auth can provide basic authentication if it is needed. Since this server has no need for authentication it won't be covered here. Basic examples of using both mod_auth and mod_access are located in Apache's documentation.

In the aliases section there are other directories, which provide the manual and fancy indexing. These should simply be commented out in general the less information yielded by the server the better. To do this place the pound sign at the beginning of each line of the directive. For example:

```
#Alias /manual "/usr/local/apache2/manual"

#<Directory "/usr/local/apache2/manual">
#    Options Indexes FollowSymLinks MultiViews IncludesNoExec
#    AddOutputFilter Includes html
#    AllowOverride None
#    Order allow,deny
#    Allow from all
#</Directory>
```

File Permissions in htdoc
All the files in the htdoc directory should have read-only permissions set. Verify this using the ls –ll command.

```
drw-r--r--    2 root  root     4096 Jan 27 02:09 error
drw-r--r--    2 root  root     4096 Jan 27 02:01 images
-rw-r--r--    1 root  root      111 Jan 27 02:03 index.html
```

The letters on the left are permission settings. The d indicates that an entry is actually a directory. An r, w, or x indicates permission to read, write or execute. The first time it is listed is for the owner of the file (in this case root). The second time it is listed is for users in the owner's group and the last time is for the world or global setting. Here everything is set to read-only for the world. If it weren't then the following command would adjust the file:

```
chmod 644 filename
```

Read = 4, Write = 2 and Execute = 1. To combine these functions together add them. The permission 644 makes the file readable to everyone but only allow the owner to write to it.

Removing Unneeded Files
The following files should be removed from the server at this point. The /cgi-bin directory and all of it's contents should be deleted. The contents within the htdocs directory should also be removed. The deleted files are well known for giving hackers more information about the server than they would normally have a right to get. The new web server content will replace the files deleted from htdocs. The /manual directory should be removed. The files in the error directory should be replaced by custom error messages.

chroot()
The chroot() command cloisters a process inside a restricted filesystem substituting a different directory tree for root. The process and users tied to that process are only able to obtain access to the files the system administrator put in place. The advantage to this is that an exploit found in the software in the chroot()ed environment will only effect that environment and not grant the hacker root access and control of the entire system.

The disadvantages are it is difficult to set up, isn't easy to use once it is going, and is easy to break out of if the wrong files are placed within it. Still all and all it is best to provide as little to hackers as possible.

Create a directory tree where Apache and its restricted version of the OS will reside. The following directories should be created:

```
mkdir /chroot
mkdir /chroot/dev
mkdir /chroot/lib
mkdir /chroot/etc
mkdir /chroot/bin
mkdir /chroot/usr
mkdir /chroot/usr/local
```

The mknod command is used to create special files in Unix. Create a /dev/null for the chroot()ed filesystem.

```
mknod -m 666 /chroot/dev/null c 1 3
mknod –m 666 /chroot/dev/random  c 1 8
mknod –m 666 /chroot/dev/urandom c 1 9
```

Copy everything under the apache home directory over to the (/usr/local/apache2/ by default) over to the /chroot/usr/local directory. The –rp means to recursively copy all files under this part of the directory tree and to preserve file permission when they are copied.

The files should be copied instead of moved to facilitate adding new modules later. If a new module were to be installed in the chroot()ed environment compilation may be difficult because much of the filesystem is missing.

```
   cp -rp /usr/local/apache2/ /chroot/usr/local
   cp /bin/sh /chroot/bin
```

Certain libraries need to be copied over as well. The ldd command lists dynamic dependencies of the program toward which it is pointed.

```
   ldd /usr/local/apache2/bin/httpd
```

Take the output from the ldd command and copy each of the libraries it displays into the /chroot/lib directory. During the installation with the hardware above these were the dependencies revealed by ldd.

```
   cp /lib/libm.* /chroot/lib/
   cp /lib/libgdbm.* /chroot/lib
   cp /lib/libdb.* /chroot/lib
   cp /lib/libdl.* /chroot/lib
   cp /lib/libc.* /chroot/lib
```

The libnss_files is required without it the uid will not map to the username. libnss_dns is required in order to lookup hosts.[32]

```
   cp /lib/libnss* /chroot/lib
```

The /etc files required need to be copied over. They will be edited in a minute to only have the username that Apache needs to start.

```
   cp /etc/passwd /chroot/etc
   cp /etc/shadow /chroot/etc
   cp /etc/group /chroot/etc
   cp /etc/resolv.conf /chroot/etc
   cp /etc/hosts /chroot/etc
   cp /etc/localtime /chroot/etc
   cp /etc/localtime /chroot/etc
   cp /etc/ld.so.* /chroot/etc
```

32. Holcomb, William. "Apache Setup." URL: http://odin.himinbi.org/website_via_cvs/apache.html (20 January 2003).

At this point Apache should come up in chroot(). Testing it now before cleaning up the chroot()ed jail isolates whether the fault is in configuration performed after this point or if a library or dependency is missing. Please note that the /usr/local/apache2/bin/apachectl referred to in the chroot command is actually /chroot/usr/local/apache2/bin/apachectl.

```
chroot /chroot /usr/local/apache2/bin/apachectl start
```

If it comes up without errors (when you hit return it returns you to the prompt with no response) then congratulations you have chroot()ed Apache and should issue the next command

```
chroot /chroot /usr/local/apache2/bin/apachectl stop
```

HINT: If Apache does come up with an error at this point run this command:[33]

```
strace /sbin/httpd 2>&1 | grep "No such file"
```

This should provide a list of any dependencies that are missing.

Apache needs to be given a unique user and group that won't be used outside the chroot()ed tree. Leaving the old /etc files in place in the chroot() would give hackers an exact copy of the user structure, groups and shadowed passwords in the root directory. If the same user or group exists outside the chroot()ed tree it could be exploited by a hacker to gain privilege in the root portion of the tree.

The following commands will overwrite the old files in /chroot/etc with a unique user and group for Apache to use.

```
echo "www:x:80:80:Web Account:/chroot/home/www:/usr/bin/False" >
/chroot/etc/passwd
echo "www:*:10882:-1:99999:-1:-1:-1:134537804" > /chroot/etc/shadow
echo "www:x:80:www" > /chroot/etc/group
```

The file permissions of the /etc files should be made read-writable by root only.

```
chmod 600 /chroot/etc/passwd shadow group
```

The file permissions of the httpd file should be set so that anyone can execute it but only root can read it and no one can write to it.[34]

33. Ibid.

34. Apache Project. "Security Tips - Apache HTTP Server." URL:http://httpd.apache.org/docs-2.0/misc/security_tips.html (20 January 2003).

```
   chmod 511 /chroot/usr/local/apache2/bin/httpd
```

Now Apache's httpd.conf must be modified to reflect these changes. Edit /chroot/usr/local/apache/conf/httpd.conf. Find the lines that read User nobody and Group #-1 and change them to the following.

```
   User www
   Group www
```

Finally edit the /etc/rc.d/rc.local and add the following lines

```
   echo "Starting Apache in chroot()ed tree "
   chroot /chroot /usr/local/apache2/bin/apachectl start
```

The commands above automatically spawn httpd.

5. Ongoing Maintenance

5.1 Backups

Frequent backups facilitate restoring a system to full capacity in the case of compromise or hardware failure. The first backup should be performed once the system is hardened and ready to be placed onto the Internet, but before it is actually placed on the Internet. As with the Tripwire database this provides assurance that the data being backed up is unaltered.

```
tar cfz backup.tar.gz / --verify –-label=webserver
```

The backup will commence it will compress the archive with gzip automatically verify all the files in the archive and label the archive. Since there is no backup device on this server the archive will have to be copied to another computer.

Since this is a single home web server it will be manually backed up whenever the site is changed or once a week if it is changed with greater frequency. If this server were at a small business weekly full backups and daily incremental backups should be instituted with a tape drive attached to the server or over the wire to a backup server.

Things to keep in mind: Running Tripwire before performing the incremental backup may be a good idea to verify that the only files that have changed on the server are files that were suppose to be changed. Also test the site to verify the changes made are actually the desired changes before running through the backup process.

5.2 Update RPMs

As stated earlier, applying the latest patches to the server software greatly reduces the chance of server compromise because known vulnerabilities are the number one hacker exploit.[35] To automate the process of updating the server as much as possible without opening up the vulnerability of installing software without verifying it, a shell script[36] will be used to download and verify the files.

```
#!/bin/bash

cd /updates
rpm -qa --qf "%{NAME}-*\n" > updated_rpm_list.tmp
for i in `cat updated_rpm_list.tmp`
do
               echo $I
               ncftpget
ftp://ftp.tux.org/distributions/redhat/releases/redhat-
8.0/en/os/i386/SRPMS/$i

done

rm -f updated_rpm_list.tmp
rpm -v --checksig *.rpm > validation.report
```

The updates will still have to be freshened manually after examining the validation report to ensure that all the signatures are in order.

```
rpm -F *.rpm
```

In order for the script to run automatically it will have to be copied to the /etc/cron.weekly directory. Note that the freshening will set off Tripwire since the files have changed. Once a month or so Tripwire's database should be updated to synchronize the file signature database with the files. (This will be explained later).

5.3 Updating Apache

Apache was not downloaded as an RPM package; it was downloaded as a tarball. As such it needs to be updated manually. The mailing list in section 4.5 should send e-mail when patches are made available for the Apache software. Patches can be downloaded from http://www.apache.org/dist/httpd/patches/apply_to_2.0.44/ via ftp. Once the patch is downloaded install it using the following command:

35. Johnston Margaret. "Security guru says known vulnerabilities are No. 1 hacker exploit." 16 December 1999. URL:http://www.infoworld.com/articles/en/xml/99/12/16/991216enhackers.xml. (24 January 2003).

36. Hariss, Jeff. "Securing a Linux FreeS/Wan Gateway for Home Use." 1 December 2002. URL: http://www.giac.org/practical/GCUX/Jeff_Harriss_GCUX.pdf (24 January 2003).

```
patch -s < name.of.patch
```

The compiler should issue some warnings. This is expected. Once the installation is complete test the bugs the patch is suppose to fix to verify that they work.

5.4 Scans

Nikto

Nikto is capable of updating itself over the wire. This is good since vulnerabilities are being exposed at an ever-increasing rate[37]. This should be done before testing with Nikto every time. The web server should be tested against Nikto at least once a month. The commands for updating and running Nikto can be found in section 6.3 along with the output of a vulnerability scan.

Tripwire

Tripwire should be configured by default to run once a day. The reports generated are found in the /var/lib/tripwire/report/ directory. The reports look like this:

```
Tripwire(R) 2.3.0 Integrity Check Report

Report generated by:          root
Report created on:            Thu 23 Jan 2003 12:58:00 AM EST
Database last updated on:     Fri 24 Jan 2003 10:24:00 AM EST
=======================================================================
 Report Summary:
=======================================================================

Host name:                    host.somewhere.com
Host IP address:              10.0.0.2
Host ID:                      None
Policy file used:             /etc/tripwire/tw.pol
Configuration file used:      /etc/tripwire/tw.cfg
Database file used:           /var/lib/tripwire/host.somewhere.com.twd
Command line used:            tripwire -m c

=======================================================================
 Rule Summary:
=======================================================================
-----------------------------------------------------------------------
Section: Unix File System
-----------------------------------------------------------------------
Rule Name                  Severity Level     Added    Removed  Modified
---------                  --------------     -----    -------  --------
```

37. CERT. "CERT/CC Statistics 1988-2002." 21 January 2003. URL:http://www.cert.org/stats/#vulnerabilities. (24 January 2003).

```
  Invariant Directories           66            0         0         0
  Temporary directories           33            0         0         0
  Tripwire Data Files             100           0         0         0
  Critical devices                100           0         0         0
  User binaries                   66            0         0         0
  Tripwire Binaries               100           0         0         0
```

In order to actually see the reports though you will have to issue the command.

```
twprint -m r --twrfile /var/lib/tripwire/report/<filename>.twr
```

You may wish to redirect the output to a file. Since the server is pretty static and the files Tripwire is monitoring are not prone to change, none of these categories in the report should show signs of change. If they do then the files that have been changed should be mentioned elsewhere in the report. Test the files and change them or re-install the programs if necessary. Even though the logs may be suspect search them around the time the file was last accessed to see if you can spot any unusual activity on the system. Double check to ensure the latest patches are installed.

Once the files have been replaced, the next thing to do would be to get the secured copy of the Tripwire database that was made at the end of Section 4.5 and perform an integrity check. Ensure the floppy is write-protected before placing it in the floppy drive. Mount the floppy as before and copy the database from the floppy back onto the server and then run a new integrity check.

```
cp /mnt/floppy/[hostname].twd /var/lib/tripwire
tripwire --m c > output.txt
```

In a small business scenario stronger incident handling policies are required. If it is suspected that the server has been compromised a couple of disk images of the server should be taken one for investigation and one to keep as evidence. The full policy is outside of the scope of the practical.

5.5 Log Monitoring

Rotatelogs

Apache provides system administrators with rotatelogs. Rotatelogs allows administrators to pipe logs to timestamped backup files at a number of seconds to be set by the administrator.[38] In order to use this feature the httpd.conf file must be edited. httpd.conf is found in the conf directory of the Apache installation.

38. Apache Foundation. "Log Files." URL: http://httpd.apache.org/docs/logs.html (24 January 2003).

```
CustomLog "|/usr/local/apache2/bin/rotatelogs
/usr/local/apache2/log/access_log 86400" common

# make certain to comment out the other error log line.

ErrorLog "|/usr/local/apache2/bin/rotatelogs
/usr/local/apache2/log/error_log 86400" common
```

Every day (86400 seconds), the log files for the web server are sent to access_log, error_log, and new logs are created. The active log files will have names such as access_log.1043539200 and error_log.1043539200. The server will need to be restarted for the httpd.conf to reload. This isolates each days logs and prevents the web server from having to stop to change logs. The smaller the logs the more likely administrators are to find the information they seek.

Apache logs are written inside the chroot. Since this is the case, copying rotated logs out of the chroot on a regular basis might lend forensic evidence as to when the chroot has been penetrated.

The following script mirrors the rotated logs into the /var/log/ directory in the non-chrooted portion of the computer.

```
#!/bin/bash

if [ $1 = "-install" ]; then

   echo Building Directories
   mkdir /var/log/apache
   for i in Mon Tue Wed Thu Fri Sat Sun
   do
      mkdir /var/log/apache/$i
   done

else

   dayofweek=`date | awk '$1 ~ /[A-Z]/ {print $1}'`
   echo Copying files for $dayofweek
   cp –f /chroot/usr/local/apache2/logs/* /var/log/
   cp -f /chroot/usr/local/apache2/logs/* /var/log/apache/$dayofweek

fi
```

Save this script as copyapachelog then run it with the –install parameter the first time it is run. It will not copy anything but it will create the directory structure with /var/log for the mirror.

Next run it without any parameters and it should copy the logs from the apache folder into the folder for today. Note that it is also copying the today's files into the /var/log directory. This allows logwatch to pick up the logs while keeping a copy of the last six days of logs safe in case of break-in. Copy the script into the /etc/cron.daily folder and it will copy the logs once a day automatically.

Logwatch

While not wanting to overload the administrator with too many log files coming in on a day-to-day basis there are some levels of information worth sending daily via e-mail - to do this edit the /etc/log.d/logwatch.conf file.[39]

```
# Default person to mail reports to.  Can be a local account or a
# complete email address.
MailTo = root@host.somewhere.com
```

By default, logwatch sends the e-mail to users requested once a day. This provides a central repository to check all logs on a daily basis if the need presents itself.

Swatch

Swatch will watch logs real-time and send alerts to the console (and via other mediums) when an event of interest is added to the logs. Swatch can be downloaded from:

ftp://ftp.cerias.purdue.edu/pub/tools/unix/logutils/swatch/swatch-3.0b5.tar.gz

Running ./configure will reveal that three (actually four) libraries need to be added to perl before the Swatch installation can proceed successfully.

Bit::Vector
http://search.cpan.org/CPAN/authors/id/S/ST/STBEY/Bit-Vector-6.3.tar.gz

Date::Calc
http://search.cpan.org/CPAN/authors/id/S/ST/STBEY/Date-Calc-5.3.tar.gz

Date::Parse
http://www.perl.com/CPAN-local/authors/id/G/GB/GBARR/TimeDate-1.14.tar.gz

File::Tail
http://search.cpan.org/CPAN/authors/id/M/MG/MGRABNAR/File-Tail-0.98.tar.gz

39. Bauer Kirk, "www.logwatch.org." June 1998. URL: http://www.logwatch.org/tabs/docs/logwatch.8.html (24 January 2003).

Bit::Vector is required by Date::Calc and should be installed first. The others can be installed in any particular order. To install a perl module run the following commands in the directory created after the tarball is extracted:

```
perl Makefile.pl
make
make install
```

Once completed the same commands need to be applied to install Swatch since it too is a perl script.

Swatch needs to be configured so that it will tail certain logs searching for certain events. The swatch configuration files for this server looks like this:[40]

```
#
# swatch -c /etc/swatchrc -t /var/log/messages
#
#Reload swatch with this entry uncommented to run tests
#ignore           /10.0.0.3/

### Kernel problems or system reboots
watchfor       /shutting down for system reboot|Linux version/
               echo bold
               mail addresses=someone@somewhere.com,subject=System
reboot!

watchfor       /file system full/
               echo bold
               mail addresses=someone@somewhere.com,subject=File system
Full
               throttle 01:00
               bell

watchfor       /su:/
               echo bold
               mail addresses=someone@somewhere.com,subject=Someone
sued to root access

watchfor       /Accepted password for root from/
               echo bold red
               bell
```

The watchfor command literally watches for that phrase to pass through the log it is monitoring. If multiple phrases are put in place to trip off that alert they must be separated by a pipe ("|") else it will pile them all into one entry.

40 Spitzner, Lance. "http://www.spitzner.net/swatchrc.txt." 7 April 2000. URL: http://www.spitzner.net/swatchrc.txt. (1 February 2003).

Watchfor also is exact in matching phrases. For example an alert /htdocs/c could match c and cgi-bin equally well.[41]

The echo command echoes the offending log phrase to the console. The bell command sets off a beep.

Notice in general that the entries chosen are critical problems that one would want to know about immediately. A balance should be struck between overloading the console with messages and missing critical problems between log audits.

The web server has it's own swatch script as well:

```
#
# Swatch configuration for Apache Server
#
#
#swatch -c /etc/swatchwebrc -t /chroot/usr/local/apache2/logs/error_log

#Reload swatch with this entry uncommented to run tests
#ignore /10.0.0.3/

### overcome with requests
watchfor /consider raising the MaxClients/
        echo=red,
        bell,
        mail addresses=someone@somewhere.com,subject=Maxclients Reached

### A hacker thinks we are a Windows web server
watchfor /[Mm][Ss][Aa][Dd][Cc]|_vti_bin|_mem_bin|scripts/
        echo=red,
        bell 3,
        mail addresses=someone@somewhere.com,subject=Possible Attempt
to exploit IIS,
        exec perl /myperl/addhackerlist.pl $0

### Someone is scanning us with Nikto
watchfor /passwd|win.ini|Nikto/
        echo=red,
        bell 3,
        mail addresses=someone@somewhere.com,subject=Possible Nikto
Scan,
        exec perl /myperl/addhackerlist.pl $0


### RFC 2616 and malformed host header
```

41 Bates, Marion and Stearns, William. "Setting up automatic alerting in your Unix environment." 26 January 2001. URL: http://www.ists.dartmouth.edu/IRIA/knowledge_base/swatch.htm. (1 February 2003).

```
watchfor /RFC2616|Client sent malformed Host header/
        echo=red,
        bell 3,
        mail addresses= someone@somewhere.com,subject=Possible
Malformed Header Attack,
        exec perl /myperl/addhackerlist.pl $0

## Someone looking for cgi-bin (it was deleted on this server)
watchfor /cgi-bin/
        echo=red,
        bell 3,
        exec perl /myperl/addhackerlist.pl $0,
        mail addresses=someone@somewhere.com,subject=Possible CGI Scan
```

Notice the exec command in the web server swatch configuration file. It takes the offending line in the log, ascertains the offending IP address and places it in a table that the firewall is written to pull. The $0 at the end of the statement means to place the offending log statement at the end of the command.

Cron could be set to restart the firewall once a day, but in this server's configuration the firewall is restarted manually to afford the administrator the opportunity to examine why the IP addresses are being restricted. This ensures that legitimate traffic is not blocked by an overzealous swatch rule.

The addhackerlist.pl file should be placed in a directory named /myperl off the root directory else the swatch attacks above will have to be configured to the path in which the perl script is placed. The perl script is here:

```
#!/usr/bin/perl
#addhackerlist.pl - Adds hackers utilizing attacks alerted to by swatch
to the list of sites to deny.

use IO::File;

foreach $word(@ARGV){
        $line = $line." ".$word;
}

#parse out the hacker's IP address from the log entry
@linearray = split /\[/, $line;
@line2array = split /\]/, $linearray[3];
@hackerip = split / /, $line2array[0];

$ip_addr_file = new IO::File
"/chroot/usr/local/apache2/logs/hosts.deny", "r";
@in = <$ip_addr_file>;
foreach $line(@in){
        if($line == $hackerip[1]){
                $notunique = 1;
        }
}
undef $ip_addr_file;

if ($notunique == 0){
        $ip_addr_file = new IO::File ">>
/chroot/usr/local/apache2/logs/hosts.deny";
        print $ip_addr_file $hackerip[1]."\n";
        print "$hackerip[1] added to deny table. Next time firewall is
restarted host will be blocked.\n";
        undef $ip_addr_file;
} else {
        print "$hackerip[1] already exists in deny table. Host will be
denied after restarting firewall if not blocked already.\n";
}
```

Once the two swatch configuration files have been saved to /etc/swatchrc and /etc/swatchwebrc then they will require testing. To load them issue the following commands:

```
swatch -c /etc/swatchrc -t /var/log/messages &
swatch -c /etc/swatchwebrc -t /chroot/usr/local/apache2/logs/error_log
&
```

You should see the following response:

```
*** swatch-3.0.4 (pid:1776) started at Sat Feb  1 16:22:39 EST 2003
```

Try logging into the server and issuing both the proper and bad passwords after swatch is running both at the console and over SSH:

```
Feb  1 16:22:51 cosmic sshd[1725]: Accepted password for root from
10.0.0.3 port 4083 ssh2
Feb  1 16:23:38 cosmic sshd[1780]: Failed password for root from
10.0.0.3 port 4093 ssh2
Feb  1 16:23:40 cosmic sshd[1780]: Failed password for root from
10.0.0.3 port 4093 ssh2
Feb  1 16:25:17 cosmic login[1660]: FAILED LOGIN 1 FROM (null) FOR
root, Authentication failure
Feb  1 16:25:22 cosmic login[1660]: FAILED LOGIN 2 FROM (null) FOR
root, Authentication failure
Feb  1 16:26:38 cosmic  -- root[1794]: ROOT LOGIN ON tty3
```

This script appears to be working. Testing the web script: involves launching attacks on the webserver.

```
[Sat Feb 01 17:05:07 2003] [error] [client 10.0.0.3] File does not
exist: /usr/local/apache2/htdocs/cgi-bin
10.0.0.3 added to deny table. Next time firewall is restarted host will
be blocked.
[Sat Feb 01 17:05:08 2003] [error] [client 10.0.0.3] File does not
exist: /usr/local/apache2/htdocs/Nikto-1.23-dtqP3h2pVjii.htm
```

```
10.0.0.3 already exists in deny table. Host will be denied after
restarting firewall if not blocked already.
[Sat Feb 01 17:05:08 2003] [error] [client 10.0.0.3] File does not
exist: /usr/local/apache2/htdocs/fcgi-bin
10.0.0.3 already exists in deny table. Host will be denied after
restarting firewall if not blocked already...
```

It goes on like that for a bit when Nikto is run against the server. The ignore statement is in the script to allow for vulnerability testing while running Swatch. Replace the IP address in the ignore statement with the IP address of the host from which you launch your vulnerability tests. Whenever actually performing vulnerability tests with Nikto it might be a good idea to restart Swatch with the ignore statement uncommented so as not to be flooded by these messages and most importantly so that the testing host is not inadvertently cut off from the web server.

Once the scripts have tested successfully entries to start them automatically need to be placed in the rc.local file. Place the swatch entries before the entry that launches the chroot()ed web server.

```
swatch -c /etc/swatchrc -t /var/log/messages &
echo "Swatch monitoring /var/log/messages..."
swatch -c /etc/swatchwebrc -t /chroot/usr/local/apache2/logs/error_log
&
echo "Swatch monitoring /chroot/usr/local/apache2/logs/error_log"
```

Test Swatch once a month to make certain that it is still working while conducting an audit of the server's logfiles. When performing that audit seek out error messages that should be tracked by Swatch that are not tracked now. Consider placing a throttle on a message seen too often.

5.6 Change Procedure for the Web Server

The change procedure for a web server is integral to maintaining its availability and its security. The following is a list step by step of how to make a change on the website while maintaining its security:

- ✓ Verify that the last known good backups are accessible.
- ✓ Create and test web content on a test bed machine.
- ✓ Run Tripwire to verify that the server is unchanged since the last update
- ✓ Transfer content and place into the website accordingly.
- ✓ Test content to ensure that it is working correctly.
- ✓ Update Tripwire database to account for new content.
- ✓ Backup Tripwire database to a floppy.
- ✓ Perform a full backup on the server to preserve changes.

5.7 Verify Core Dump and SUID/SGID settings.

Changes in either the core dump file size or the SUID/SGID file settings could indicate an intrusion has occurred on the system. These settings should be checked weekly. This script will write a report with all the information regarding these parameters to a file.

```
#!/bin/bash

echo ###################### > /var/log/suidreport
echo #Environment Settings# >> /var/log/suidreport
echo ###################### >> /var/log/suidreport

ulimit –a >> /var/log/suidreport 2>&1

echo ##################### >> /var/log/suidreport
echo #Set-UID permissions# >> /var/log/suidreport
echo ##################### >> /var/log/suidreport

find / -perm -4000 >> /var/log/suidreport 2>&1

echo ##################### >> /var/log/suidreport
echo #Set-GID permissions# >> /var/log/suidreport
echo ##################### >> /var/log/suidreport

find / -perm -2000 >> /var/log/suidreport 2>&1
```

Placing this script in /etc/cron.weekly will update the file once a week.

5.8 Test passwords against password policy

If there are other legitimate users (i.e. people) who use this server then testing that their passwords conform to the password policy is crucial. John the Ripper, a password cracking tool used while hardening the server must be run against the passwords right after they expire. In the case of this server JTR should be run once every password change. You may wish to run it to test the strength of your own passwords as well.

6. Verifying the configuration

6.1 Verify that only authorized Daemons are running

First the server must be tested for any unauthorized daemons. A service that is open that an administrator is unaware of is akin to a castle having the drawbridge up but having ladders in several places. In other words it is an easy vulnerability to check for and fix. To do so run:

```
netstat -ap
```

The output should look like this:

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address  Foreign Address   State    PID/Program name
tcp        0      0 *:http         *:*               LISTEN   587/httpd
tcp        0      0 *:ssh          *:*               LISTEN   674/sshd

Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node PID/Program name
Path
unix  2      [ ACC ]     STREAM     LISTENING     1269   550/gpm
/dev/gpmctl
unix  6      [ ]         DGRAM                    803    395/syslogd
/dev/log
unix  2      [ ]         DGRAM                    1280   559/crond
unix  2      [ ]         DGRAM                    1252   540/clientmqueue
unix  2      [ ]         DGRAM                    958    480/apmd
unix  2      [ ]         DGRAM                    811    399/klogd
```

Under Active Internet Connection (servers and established) only http (tcp port 80) and ssh (udp port 22) should be running. This verifies that only the authorized daemons are running.

But outside verification is standard practice. In order to get independent verification from another computer first take down the firewall (it will rate limit most port scans). If using the script from earlier the following command should bring down the firewall.

```
/etc/rc.d/rc.firewall stop
```

Then from one of the windows computers on the network running nmap issue the following command:

```
C:\nmap>nmap -sS -PT -PI  -p 1-65535 -O -T 3 10.0.0.2
```

The output should match the output from the netstat command:

```
Starting nmap V. 3.00 ( www.insecure.org/nmap )
Interesting ports on  (10.0.0.2):
(The 65533 ports scanned but not shown below are in state: closed)
Port       State       Service
22/tcp     open        ssh
80/tcp     open        http
Remote OS guesses: Linux Kernel 2.4.0 - 2.5.20, Linux 2.4.19-pre4 on
Alpha
Nmap run completed -- 1 IP address (1 host up) scanned in 23 seconds
```

Once this has been verified restore the firewall with the command:

```
/etc/rc.d/rc.firewall
```

6.2 Verify that Apache is chroot()ed

It is critical that the chroot()ed jail be tested to verify that it is indeed chroot()ed.[42] Run the ps –ax | grep "httpd" command. The ps (process statistics) command shows information on the processes running on the server. The –a tag shows all processes. The x tag limits it to just processes not running on ttys (i.e. daemons and other background processes). Grep pulls out just the processes having the string "httpd" in them. The results should look like this:

```
      558 ?        S      0:00 /usr/local/apache2/bin/httpd -k start
      566 ?        S      0:00 /usr/local/apache2/bin/httpd -k start
      567 ?        S      0:00 /usr/local/apache2/bin/httpd -k start
      568 ?        S      0:00 /usr/local/apache2/bin/httpd -k start
      569 ?        S      0:00 /usr/local/apache2/bin/httpd -k start
      570 ?        S      0:00 /usr/local/apache2/bin/httpd -k start
      626 ?        S      0:00 /usr/local/apache2/bin/httpd -k start
```

Now run the ls –la /proc/[number of the process]/root. ls –l should be familiar as the list command with long entries. –a lists all files even those hidden behind a period. The process number is any one of those numbers in the first column of the ps –ax | grep httpd command above.

```
      ls -la /proc/566/root
      lrwxrwxrwx    1 root     root            0 Jan 24 01:32
      /proc/566/root -> /chroot
```

The /proc/566/root -> /chroot means that as far as that process is concerned the root directory is /chroot which is what it was set to earlier in the practical.

42. Mourani, Gerald. "Securing and Optimizing Red Hat Linux." 2000 March 25
URL:http://www.packetstormsecurity.org/papers/unix/Securing-Optimizing-RH-Linux-1_2.pdf (24 January 2003).

6.3 Vulnerability scanning the web server using Nikto

Nikto is a vulnerability scanner that specializes in testing web servers. It tests for a wide variety of vulnerabilities. It is a perl script so perl has to be installed on the scanning host. It downloads as a tarball from http://www.cirt.net/source/nikto-current.tar.gz.

Installing Nikto to a windows box with Winzip is practically automatic. Once it is installed it's database must be updated to do this run the command

```
C:\nikto> perl nikto.pl --update
```

Once the vulnerability database is updated Nikto is ready to run. One of the windows boxes on the practical networks is outfitted with Nikto.

```
C:\nikto> perl nikto.pl --host 10.0.0.2
```

The output of the test should come up like this:

```
-----------------------------------------------------------------------
- Nikto v1.23  - www.cirt.net - Fri Jan 24 00:15:01 2003
-----------------------------------------------------------------------
+ Target IP:       10.0.0.2
+ Target Hostname: ?? (Unable to resolve)
+ Target Port:     80
-----------------------------------------------------------------------
- Scan is dependent on "Server" string which can be faked, use -g to
override
+ Server: Apache
+ Allowed HTTP Methods: GET,HEAD,POST,OPTIONS,TRACE
+ HTTP method 'TRACE' may allow client XSS or credential theft. See
http://www.cgisecurity.com/whitehat-mirror/WhitePaper_screen.pdf for
details.
+ / - TRACE option appears to allow XSS or credential theft. See
http://www.cgisecurity.com/whitehat-mirror/WhitePaper_screen.pdf for
details (TRACE)
- 1408 items checked, 1 item found on remote host
```

There is a cross-site scripting/cookie theft vulnerability open on the host. Apache has given no confirmation of an adequate resolution to this date. The only unconfirmed fix to this vulnerability is to use the mod_rewrite module (built into this server) to institute a fix.[43] Since TRACE cannot be circumvented using the limit command add the following lines to the end of the httpd.conf:

43. Grossman, Jeremiah. "Cross-Site Tracing (XST):The New Techniques and Emerging Threats to Bypass Current Web Security Measures Using Trace and XSS." 20 January 2003. URL: http://www.cgisecurity.com/whitehat-mirror/WhitePaper_screen.pdf. (20 January 2003).

```
  RewriteEngine on
  RewriteCond %{REQUEST_METHOD} ^(TRACE)$
  RewriteRule .* - [F]
```

Restart the server after the httpd.conf has been saved. Running Nikto again should yield no errors.

6.4 Testing GRUB and BIOS Security

Rebooting the computer will verify that the BIOS and GRUB security changes made are in place. First place a Linux boot floppy in the floppy drive and a Red Hat Linux disk in the CD-ROM. Next reboot the system. If the changes listed in section 4.5 were made the system should prompt for a password before yielding any other options. Enter the password, the system should ignore the floppy and CD-ROM and boot directly to the hard drive.

Once the machine enters the GRUB menu try hitting the 'e' key for editing the commands or the 'c' key to enter a command-line interface. Neither of these should work. Hitting the 'p' key should cause GRUB to prompt for a password.

6.5 Verify IP Tables

To verify that the proper firewall policy is in place run the iptables –L command. The resulting output should look much like the output in Appendix A. In order to independently verify that the firewall policy put in place is working re-run the nmap command run in section 6.1 (a SYN Stealth scan on all ports). If the firewall is running and the configuration is like the one listed in the script earlier in the practical the output this time should read.

```
Starting nmap V. 3.00 ( www.insecure.org/nmap )
Interesting ports on  (10.0.0.2):
(The 65533 ports scanned but not shown below are in state: closed)
Port       State      Service
22/tcp     open       ssh
80/tcp     open       http
Remote OS guesses: Linux Kernel 2.4.0 - 2.5.20, Linux 2.4.19-pre4 on
Alpha
Nmap run completed -- 1 IP address (1 host up) scanned in 80 seconds
```

Next test the services. Launch a web browser toward the server. The index.html web page should load. Launch an ssh client at the server and you should be able to log in via SSH. Since these are the only services allowed through the firewall.

6.6 Password Audit

Password auditing should take place to verify that the passwords chosen are sufficiently strong to thwart a reasonable attack. This is especially the case if there are other users accessing the system (i.e. people who are not you). To do this a password cracker should be run against the user's passwords to check for password strength.

The Official Red Hat Linux Security Guide lists John the Ripper as a "fast and flexible password cracking program."[44] The source code for John can be downloaded from http://www.openwall.com/john/. The stable version as of this writing is 1.6.

```
tar –zxvf john-1.6.tar.gz
```

Once the source is downloaded untar it per instructions above. This will generate a directory named john-1.6. Enter the src directory found under that directory. If you are running gcc version 3 or later (and if you followed the installation directions you are) changes will need to be made to the Makefile in this directory before installation will succeed. The Makefile has a deprecated line in it. In the case of the server listed above the following lines had to be changed from:

```
linux-x86-any-elf:
               $(LN) x86-any.h arch.h
               $(MAKE) $(PROJ) \
                  JOHN_OBJS="$(JOHN_OBJS) x86.o" \
                  CFLAGS="$(CFLAGS) -m486"
```

to:

```
linux-x86-any-elf:
               $(LN) x86-any.h arch.h
               $(MAKE) $(PROJ) \
                  JOHN_OBJS="$(JOHN_OBJS) x86.o" \
                  CFLAGS="$(CFLAGS) -mcpu=i486"
```

Once that change is completed run make.and a list of configurations of OS and architecture should appear. Type make SYSTEM where SYSTEM is the configuration of your system on the list of systems issued with the last make command. In the case of this server the command looks like this:

```
make linux-x86-any-elf
```

44. Red Hat. "Password Security." URL: http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/security-guide/s1-wstation-pass.html. (1 February 2003).

This should install a binary named john into the /john-1.6/run directory. According to the John the Ripper documentation, the run directory can be copied anywhere in the directory tree.

```
mv –f /john-1.6/run /sbin/john
chown root /sbin/john
chmod 700 /sbin/john
```

Those commands complete the installation of John the Ripper. The last two commands ensure that root owns the directory it is in and that only root users may manipulate the items therein.

Once installed the following commands can be run to test the passwd file. First copy the password file to the root directory and ensure that only users with root access can manipulate it.

```
cp /etc/passwd /root/passwd
chown root /root/passwd
chmod 600 /root/passwd
```

Next, run John:

```
/sbin/john/john –single /root/passwd
```

If the passwords are shadowed you should receive the following message:

```
Loaded 0 passwords, exiting...
```

So unshadowing the passwd file, grabbing the data we need and reshadowing should look like this.

```
pwunconv
cp /etc/passwd /root/passwd #(overwrite the old one)
pwconv
```

To verify that the password list is once again shadowed:

```
cp /etc/passwd /root/passwd2 #(just to verify re-shadowing)
/sbin/john/john –single /root/passwd2

Loaded 0 passwords, exiting...
rm /root/passwd2
```

and now the moment of truth:

```
/sbin/john/john –single /root/passwd

guesses: 0 time: 0:00:00:07 100% ...
```

was the response this server yielded. If JTR guessed your password at this point you definitely need a stronger password. Guessed passwords look like this:

```
guesses: 1  time: 0:00:00:06 ...
foo            (fighters)
```

So next John should be run to actually test the passwords. A word of warning, running a password cracker is resource intensive it will gobble system resources and hence should only be run when you are certain the server will not be in use. (You could run the nice command but this next command will take long enough at full speed).

```
/sbin/john/john -i:all /root/passwd
```

This mode is incremental mode. Besides performing a brute force check it is the longest mode to run with John the Ripper. It cycles through all permutations of all the characters on the keyboard.[45] How long you keep this running is up to you. For this server running John in this mode for a full day did not yield any passwords. Once the test is complete, delete the /root/passwd file.

6.7 Testing the Web Site

Once the web server is up and connected go to another computer outside the home network and attempt to connect to the website. First attempt to use the IP address assigned by the ISP. Next test the domain registered with the DynDNS provider. In both cases simply launch the web browser, fill in the URL and hit enter. If all is going well the website should load.

While testing, check out the 404 message to make certain that if it is the default message provided by Apache that it does not display the version number and type of server. If you created a custom 404 error message it should not reveal this information either.

45. k4mts, "JTR Tutorial." URL: http://www.mindfuk.darkg.com/tutorials/jtr.html. (1 February 2003).

Works Cited

1. Red Hat, “redhat.com | Included Packages.” URL: http://www.redhat.com/software/linux/technical/packages.html (20 January 2003).

2. CERT, “CERT® Advisory CA-2002-24 Trojan Horse OpenSSH Distribution.” 2 August 2002. URL: http://www.cert.org/advisories/CA-2002-24.html (20 January 2003).

CERT, “CERT® Advisory CA-2002-28 Trojan Horse Sendmail Distribution.” 14 October 2002. URL: http://www.cert.org/advisories/CA-2002-28.html (20 January 2003).

CERT, “CERT® Advisory CA-2002-30 Trojan Horse tcpdump and libpcap Distributions.” 13 November 2002. URL: http://www.cert.org/advisories/CA-2002-30.html (20 January 2003).

3. CERT, “CERT® Incident Note IN-2001-06.” 8 June 2001. URL: http://www.cert.org/incident_notes/IN-2001-06.html (20 January 2003).

4. Callendar, John. “Running a Guestbook”, URL:http://www.lies.com/begperl/guestbook.html. (2 February 2003).

5. Scheneider, Bill, “P R E S S R E L E A S E.” 14 February 2000. URL:http://www.arena.no/nyheter/wsa-ddos.htm (20 January 2003).

6. Russell, Rusty. “Linux 2.4 Packet Filtering HOWTO: Using iptables.” 20 November 2001. URL:http://www.netfilter.org/unreliable-guides/packet-filtering-HOWTO/packet-filtering-HOWTO.linuxdoc-7.html (20 January 2003).

7. Honeynet Project, “Know Your Enemy: Statistic.” 22 July 2001. URL:http://project.honeynet.org/papers/stats/ (20 January 2003).

8. A Linux Treatment of GPG can be found at http://www.gnupg.org/gph/en/manual.html#AEN84

9. Red Hat, “Partitioning Your System.” URL:http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/install-guide/s1-diskpartitioning.html (20 January 2003).

10. Keane, Justin, “Madirish.net.” 2002 URL:http://www.madirish.net/tech.php?article=95&section=5 (20 January 2003).

11. Dobani, Abid Ali, "More About RPM." URL:http://www.student.math.uwaterloo.ca/~aadobani/Specifics.htm. (1 February 2003).

12. Red Hat, "Boot Loader Configuration." URL:http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/install-guide/s1-x86-bootloader.html (20 January 2003).

13. Red Hat, "Server Security." 2002. URL: http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/security-guide/ch-server.html#S1-SERVER-TCPW-XINETD. (2 February 2003).

14. CIAC, "CIACTech02-001: Understanding SSH Exploits." 9 May 2002. URL: http://www.ciac.org/ciac/techbull/CIACTech02-001.shtml (20 January 2003)

15. Kugelberg, Thorsten. "RE: [suse-autoinstall] How to save package sets in suse 8.1?" 22 October 2002. URL: http://lists.suse.com/archive/suse-autoinstall/2002-Oct/0045.html (24 January 2003).

16. Hariss, Jeff. "Securing a Linux FreeS/Wan Gateway for Home Use" 1 December 2002. URL: http://www.giac.org/practical/GCUX/Jeff_Harriss_GCUX.pdf (24 January 2003).

17. Red Hat "BIOS and Boot Loader Security." URL: http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/security-guide/s1-wstation-boot-sec.html#S3-BOOTLOADER-GRUB. (22 January 2003).

18. Morizot, Scott. "Easy Firewall Generator for iptables" 24 January 2003. URL: http://morizot.net/firewall/gen/ (24 January 2003).

19. Andreasson, Oskar. "Iptables Tutorial 1.1.16." 2002. URL: http://iptables-tutorial.frozentux.net/iptables-tutorial.html (24 January 2003).

20. Berninger, John. "NCSU Realm Kit for Red Hat Linux 7.3 Guide - Securing the Machine." 2002 URL: http://www.linux.ncsu.edu/realmkit/usersguide/x197.html (20 January 2003).

21. Red Hat, "Install the Tripwire RPM." URL: http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/ref-guide/s1-tripwire-install-rpm.html (20 January 2003).

22. Red Hat, “Customizing Tripwire.” URL: http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/ref-guide/s1-tripwire-initialize.html (20 January 2003).

23. Williams, C. “Lab 3 | Final Part.” URL: http://www.eecis.udel.edu/~cwilliam/cis479/Lab-IDS-B.htm (20 January 2003).

24. Schneider, Neil. “SUID.” 2002 URL: http://www.linuxgeek.net/index.pl/suid. (4 February 2003).

25. Gateways. “http://onicrom.com/info/security.txt.” URL:http://onicrom.com/info/security.txt. (4 February 2003).

26. volatile, “Setuid/Setgid Tutorial.” 15 January 2002. URL: http://neworder.box.sk/newsread.php?newsid=2380. (4 February 2003).

27. Schneider, Neil. “SUID.” 2002 URL: http://www.linuxgeek.net/index.pl/suid. (4 February 2003).

28. Madhusudan and Mourani, G. “Special Accounts.” 2000 URL: http://www.tldp.org/LDP/solrhe/Securing-Optimizing-Linux-RH-Edition-v1.3/chap5sec42.html (6 February 2003).

29. Chung, Adrian. “Core Dump Files and What To Do About Them.” 2002. URL: http://startlinux.co.nz/articles/article_153.php. (4 February 2003).

30. Hewlett-Packard. “HP WebWise MPE/iX Secure Web Server.” 10 December 2002. URL: http://jazz.external.hp.com/src/webwise/. (3 February 2003).

31. Apache. “core – Apache HTTP Server.” URL:http://httpd.apache.org/docs-2.0/mod/core.html#options. (3 February 2003).

32. Holcomb, William. “Apache Setup.” URL: http://odin.himinbi.org/website_via_cvs/apache.html (20 January 2003).

33. Ibid.

34. Apache Project. “Security Tips - Apache HTTP Server.” URL:http://httpd.apache.org/docs-2.0/misc/security_tips.html (20 January 2003).

35. Johnston Margaret. “Security guru says known vulnerabilities are No. 1 hacker exploit.” 16 December 1999.

URL:http://www.infoworld.com/articles/en/xml/99/12/16/991216enhackers.xml. (24 January 2003).

36. Hariss, Jeff. “Securing a Linux FreeS/Wan Gateway for Home Use.” 1 December 2002. URL: http://www.giac.org/practical/GCUX/Jeff_Harriss_GCUX.pdf (24 January 2003).

37. CERT. “CERT/CC Statistics 1988-2002.” 21 January 2003. URL:http://www.cert.org/stats/#vulnerabilities. (24 January 2003).

38. Apache Foundation. “Log Files.” URL: http://httpd.apache.org/docs/logs.html (24 January 2003).

39. Bauer Kirk, “www.logwatch.org.” June 1998. URL: http://www.logwatch.org/tabs/docs/logwatch.8.html (24 January 2003).

40. Spitzner, Lance. “http://www.spitzner.net/swatchrc.txt.” 7 April 2000. URL: http://www.spitzner.net/swatchrc.txt. (1 February 2003).

41. Bates, Marion and Stearns, William. “Setting up automatic alerting in your Unix environment.” 26 January 2001. URL: http://www.ists.dartmouth.edu/IRIA/knowledge_base/swatch.htm. (1 February 2003).

42. Mourani, Gerald. “Securing and Optimizing Red Hat Linux.” 2000 March 25 URL:http://www.packetstormsecurity.org/papers/unix/Securing-Optimizing-RH-Linux-1_2.pdf (24 January 2003).

43. Grossman, Jeremiah. “Cross-Site Tracing (XST):The New Techniques and Emerging Threats to Bypass Current Web Security Measures Using Trace and XSS.” 20 January 2003. URL: http://www.cgisecurity.com/whitehat-mirror/WhitePaper_screen.pdf. (20 January 2003).

44. Red Hat. “Password Security.” URL: http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/security-guide/s1-wstation-pass.html. (1 February 2003).

45. k4mts, “JTR Tutorial.” URL: http://www.mindfuk.darkg.com/tutorials/jtr.html. (1 February 2003).

Appendix A – Output from iptables –L

The final list of chains should look like this if the script listed above was used. The site evil.hacker.org was blocked by the perl script swatch runs that adds the offending host to the firewall list when it is restarted.

```
Chain INPUT (policy DROP)
target     prot opt source               destination
 LOG        all  --  evil.hacker.org    anywhere           LOG level
warning prefix `Hacker:'
DROP       all  --  evil.hacker.org    anywhere
ACCEPT     tcp  --  anywhere             anywhere           tcp
flags:SYN,RST,ACK/SYN limit: avg 1/sec burst 5
ACCEPT     icmp --  anywhere             anywhere           icmp echo-
request limit: avg 1/sec burst 5
ACCEPT     all  --  anywhere             anywhere
bad_packets  all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere           state
RELATED,ESTABLISHED
tcp_inbound  tcp  --  anywhere             anywhere
udp_inbound  udp  --  anywhere             anywhere
icmp_packets  icmp --  anywhere             anywhere
DROP       all  --  anywhere             255.255.255.255
LOG        all  --  anywhere             anywhere           limit: avg
3/min burst 3 LOG level warning prefix `INPUT packet died eoc: '

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere           state
INVALID
ACCEPT     all  --  localhost.localdomain  anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
LOG        all  --  anywhere             anywhere           limit: avg
3/min burst 3 LOG level warning prefix `OUTPUT packet died eoc: '

Chain bad_packets (1 references)
target     prot opt source               destination
LOG        all  --  anywhere             anywhere           state
INVALID LOG level warning prefix `Invalid packet:'
DROP       all  --  anywhere             anywhere           state
INVALID
bad_tcp_packets  tcp  --  anywhere             anywhere
RETURN     all  --  anywhere             anywhere

Chain bad_tcp_packets (1 references)
target     prot opt source               destination
```

```
LOG        tcp  --  anywhere             anywhere           tcp
flags:!SYN,RST,ACK/SYN state NEW LOG level warning prefix `New not
syn:'
DROP       tcp  --  anywhere             anywhere           tcp
flags:!SYN,RST,ACK/SYN state NEW
RETURN     tcp  --  anywhere             anywhere

Chain icmp_packets (1 references)
target     prot opt source               destination
ACCEPT     icmp --  anywhere             anywhere           icmp time-
exceeded limit: avg 1/sec burst 5
REJECT     icmp --  anywhere             anywhere           reject-with
icmp-host-unreachable
RETURN     icmp --  anywhere             anywhere

Chain tcp_inbound (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere           tcp
dpt:http
ACCEPT     tcp  --  anywhere             anywhere           tcp dpt:ssh
RETURN     tcp  --  anywhere             anywhere

Chain tcp_outbound (0 references)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere

Chain udp_inbound (1 references)
target     prot opt source               destination
DROP       udp  --  anywhere             anywhere           udp
dpt:netbios-ns
DROP       udp  --  anywhere             anywhere           udp
dpt:netbios-dgm
RETURN     udp  --  anywhere             anywhere

Chain udp_outbound (0 references)
target     prot opt source               destination
ACCEPT     udp  --  anywhere             anywhere
```