Global Information Assurance Certification Paper

GCUX Practical (v1.9):
Red Hat Java Application Server Step-by-Step

Mike Armstrong
February 15, 2004

Table of Contents

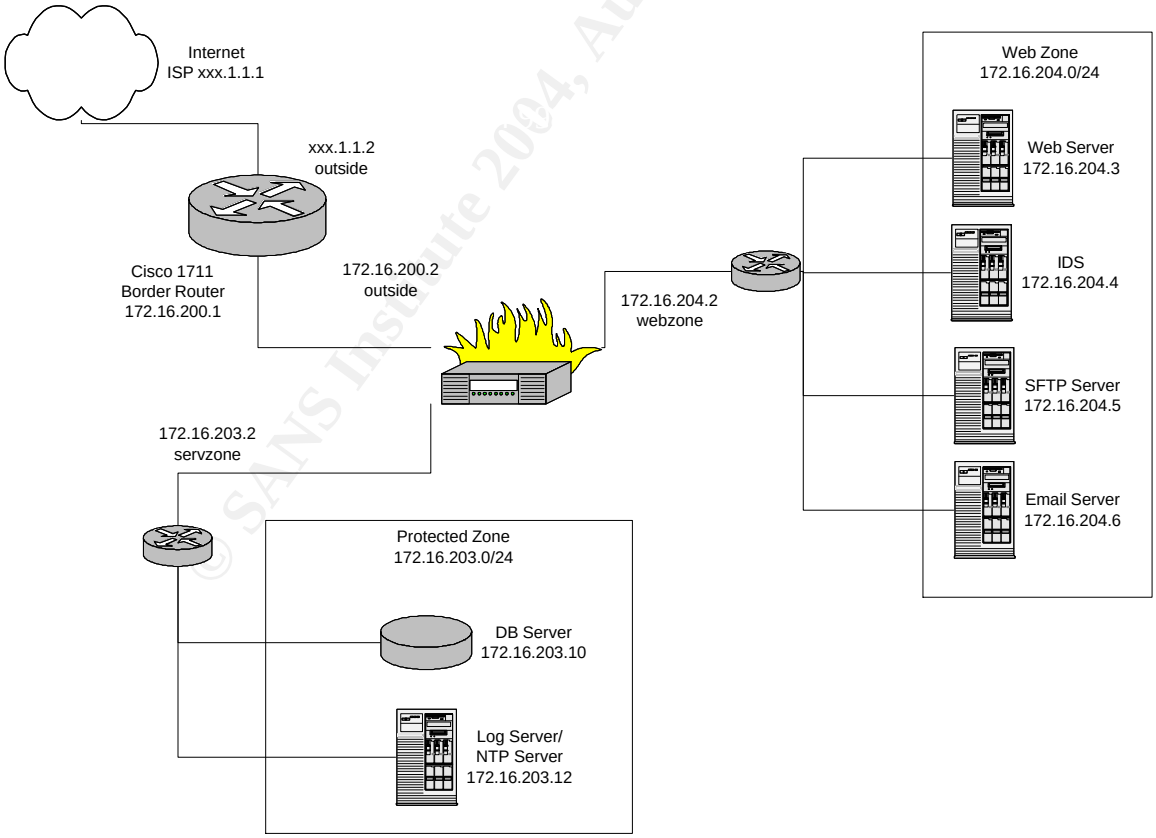
Abstract

This paper provides detailed instructions for setting up a Red Hat Linux system used as a Java application server. The instructions include installation of Red Hat Linux, installation and configuration of Apache, and the installation and configuration of Apache Tomcat as the Java servlet engine. Throughout the installation of the various software packages, system hardening is discussed in order to ensure that the resulting system is as secure from outside attack as possible.

Description of the system

Purpose

The purpose of this system is to provide a secure Java application server to be used for delivering Java web applications to customers of the company. The web server will be Internet-accessible through a firewall, with the firewall providing Network Address Translation (NAT) to help protect this server. The server will reside in a secure DMZ and will have a non-Internet routable address. A diagram showing the layout of the relevant network elements follows:



The company represented in this paper is a new small business with less than 25 employees – a startup less than six months old. This company uses technology as its primary means of generating income. Budgets are tight and the owners of the company have chosen to use open source software as a means of controlling costs. The company's development platform is Java and Apache Tomcat is used on the developers' desktops as well as on the test and QA servers. This server is the first production server being built for the company. As such, the company plans on creating a comprehensive step-by-step document describing how the production server has been built in order to ease the creation of subsequent production servers.

Environment

The server will reside in a DMZ protected by a dedicated hardware firewall. Within this DMZ there will be three other servers: an SFTP server, an Email server running sendmail, and an Intrusion Detection System (IDS) running Snort 2.0 to provide monitoring for the DMZ. A database server will reside in another protected zone accessible from the DMZ. There is a separate internal firewall between the production server and this database server. In the same protected zone as the database server are the company's NTP server and logging server, both of which the production server must access.

Access to the Java application server from the Internet is protected by the firewall. The IP address is not Internet-routable, so Network Address Translation (NAT) is used to allow connections to the Java application server from the Internet.

Hardware

The production server is being built on a Relion 1X rackmount server available from http://www.penguincomputing.com. It has a 2.8GHz Pentium 4 processor, 1GB of DDR RAM, and an 80GB hard drive. This server comes with dual integrated 10/100 Ethernet ports. The system came pre-loaded with Red Hat 9 with the boxed set so as to have the manuals on hand. However, as this system will be used as a secure server, the operating system will be installed from scratch in order to eliminate any unnecessary software packages.[1]

Software

The server will be built to run Red Hat 9 (available at www.redhat.com), with the Apache web server for serving static web content, and Apache Tomcat as the Java servlet engine. The Java version installed on this platform is the Java 2 Runtime Environment, Standard Edition 1.4.2_03, available from Sun Microsystems.[2] Apache 2.0.48 will be

1 Penguin Computing

2 Sun Microsystems

installed[3], along with Apache Tomcat 4.1.29[4], both the most recent releases as of this writing. The Java applications will access a MySQL database running in the company's protected zone as depicted in the network diagram above.

Risk analysis of the system

Risks to the system include remote compromise of the operating system, local compromise of the operating system, remote compromise of the Apache server, remote compromise of the Apache Tomcat server, denial of service attacks on the web server, and privilege escalation by both local and remote users.

Remote compromise of the operating system involves an external malicious user gaining unauthorized access to the server. If access is gained, that user can attack the operating system itself or can exploit known vulnerabilities in any of the software running under the operating system. Such a user could attempt to access confidential company data, or alter or delete that data. The intruder also could attempt to use the company's compromised system to launch attacks against other systems, either inside or outside of this company's network.[5]

Local compromise of the system would indicate an internal user of the company has gained unauthorized access to the system. The same vulnerabilities could be exploited, leading to compromise of the system or to opening up the system to external attack.

The Apache web server and Apache Tomcat server could be attacked in order to gain further entry to the system. Additionally, malicious users could attempt to alter the applications running under these software packages in an attempt to subvert the system to the malicious user's needs or to harass the company by presenting embarrassing or possibly illegal content.

A Denial of Service (DoS) attack against the system could prevent the company from conducting its legitimate business. Such attacks could cost this new company its entire business, as a startup such as this may never recover from the loss of reputation as well as the loss of revenue.

Step by step guide

Gathering Information

Prior to beginning installation, the following information must be gathered:

Information	Value for this Server

3 Apache

4 Apache Jakarta

5 Toxen, pg 8

IP Address for this server	172.16.204.3
IP Address of the database server	172.16.203.10
IP Address of the NTP server	172.16.203.12
Primary DNS server (ISP provided)	xxx.1.1.5
Secondary DNS server (ISP provided)	xxx.1.1.6
System/Server Name	CMSP001
Primary Gateway	172.16.204.1
Domain Name	www.the-company-name.com

Installation Steps

The installation steps will be broken down into the following steps:

1. Partition the hard drives
2. RPM
3. Verify the software packages
4. Install Red Hat
5. Configuring Red Hat and Hardening the System
6. Install Apache
7. Install Java
8. Install Apache Tomcat
9. Configure Apache and Tomcat
10. Set Java/Tomcat Security

Partitioning the Hard Drive

The 80 GB hard drive has an actual logical capacity of capacity of 76.32 GB. The difference in the advertised capacity and the logical capacity comes from the manner in which the GB of storage is calculated. For marketing purposes 1GB = 1,000 MB. In translation to actual hard drive space, 1 GB = 1,024 MB. The hard drive will be partitioned as follows:

Mount Point	File System	Size	% of drive	Notes
/boot	ext3	200 MB	< 1%	
/	ext3	1.0 GB	1.3%	
/home	ext3	4.0 GB	5.2%	
/tmp	ext3	2.0 GB	2.6%	
/usr		6.0 GB	7.9%	
/var	ext3	61.20 GB	80.2%	Allowed to grow to fill the drive
swap	swap	2.0 GB	2.6%	

Linux hard drive partitions create virtual drives within the physical drive. The boot partition stores the Linux startup commands and is kept separate so that if another part

of the file system is corrupted, it will still be possible to boot the server (assuming, of course, that the boot partition has not been corrupted). The swap partition provides an area to swap the computer's RAM to the hard drive. The general recommendation is to create a swap partition at the same size as the computer's memory or up to twice that size. This partition is set to twice the memory size to allow for the expansion of the internal memory to 2GB, the maximum that can be installed on the provided server.[6]

The other defined partitions and their purposes are:

- The / partition, or root partition, is the top level partition under Linux. All other partitions are a subset of root.
- The /tmp partition is for storing temporary file generated by running applications. This partition is kept separate as these files change often and could potentially corrupt other portions of the system. Additionally, by creating a separate /tmp partition, the size of the temporary files can be controlled.[7]
- The /home directory is used for the server's user home, or default, directories. These are the directories allocated for each user to store files. This partition is large in order to accommodate moving out new production pushes of the company's software for installations.
- The /usr partition is the default partition where most software packages are installed. The intent is to have global executables here, along with much of the documentation for the installed executables. After installation and system hardening, this partition will be set to read-only.
- The /var partition is where system log files go, as well as spool directories for print jobs. In particular, this is where the error files will be stored on this server.

RPM

RPM, or the Red Hat Package Manager, is Red Hat's means of packaging and installing applications. The package manager provides an application that can be used for verifying, installing, uninstalling, updating, querying, and building packages for Red Hat (and other) systems. Packages contain a complete and fully tested application, along with the application's configuration information.[8]

The syntax for installing a package is:

```
rpm –i packagename.rpm
```

or

```
rpm –I --percent packagename.rpm
```

6 LeBlanc, pg 2

7 LeBlanc, pg2

8 Tacket 7 Burnett, pg 149

which will provide the percentage complete of the install, allowing the installer to track progress.

To uninstall:

```
rpm –e packagename
```

To update a package:

```
rpm –U packagename.rpm
```

As packages are installed, RPM checks the integrity of the package against an GnuPG signature provided with the package. This signature must be checked against the GnuPG key supplied by the package vendor. The keys can be obtained from the media on which the software was distributed or from the vendor's web site. If pulling the keys from the vendor's web site, due care should be taken to ensure that the web site is valid.

GnuPG is a freeware replacement for PGP. It provides a cryptography engine which can be used, among other things, for signing rpm packages. The signature can later be verified against a trusted public key as a means of validating the integrity of the package.[9]

Verify the Software Packages

The Red Hat 9 CD-ROMs come as part of a boxed set with the server. This software will be installed as a trusted installation application. Software downloaded from any web site must be verified before installing it on this server.

Every package that is downloaded from an Internet site for this system also provides an MD5 checksum that can be compared against an MD5 checksum computed locally to determine if the downloaded package was corrupted during the download process. This checksum also provides partial protection against the package itself having been replaced with a modified package by a potentially malicious user. Of course, if the package has been replaced, there is good reason to believe the MD5 checksum was replaced as well. To verify a package against its MD5 checksum, run the command:

```
md5sum filename
```

The packages being installed on this system also come with an electronic signature file which can be used to verify the packages integrity. This signature is a GnuPG key which can be used to verify the packages signature. The signature from the trusted site from which the package is downloaded must be imported prior to validating the package. Do not download signature files from mirror sites – only trust signatures from

9 GnuPG

the primary site. The GPG key can be imported into Red Hat RPM using the following command:

```
rpm --import /path/to/key/KEY-FILE-NAME
```

The Red Hat key is provided on the installation CDs. The command to see all keys available to RPM is:

```
rpm -qa gpg-pubkey*
```

Once the key has been imported, a package's signature can be verified using the command:

```
rpm --checksig -v filename
```

RPM automatically checks each package being installed for the proper key as well.

Install Red Hat

Red Hat will be installed from the CD-ROM media provided with the Relion server as well as with the boot floppy disk provided. This installation procedure will cover installing Red Hat using the command prompt rather than the GUI interface. The installation steps are:[10]

1. Insert the boot floppy disk in the 3.5" floppy drive.
2. Power up the system
3. At boot up, you will receive the following prompt:

```
boot:
```

4. Press the return key to begin.
5. The installation program will prompt you for the CD. Insert the CD into the Relion's CD-Drive. You will be presented with the "Welcome to Red Hat Linux", which is purely informational. Click the "Next" button to continue (show as ->Next throughout the remainder of these instructions.
6. Select your language ("English" in this case) by clicking the language name with the mouse. -> Next.
7. Select the keyboard configuration as "U.S. English" -> Next.
8. Select the mouse -> Next.
9. Select Custom installation. This will allow you to install only the packages required for this system. -> Next.
10. Under Disk Partitioning, select "Manually Partition with Disk Druid" -> Next.
11. Define the partitions as listed in the table above -> Next.

10 Red Hat

12. Select the grub boot loader. Select “Use a boot loader password” and change the password to the default company boot password provided. Deselect “Advanced” options and -> Next.
13. Under “Network Configuration”, select each network device and click “Edit”
    a. For eth0, deselect “Configure with DHCP” and select “Activate on Boot”. Enter the following information:

IP address	172.16.204.3
Default Gateway	172.16.204.1
Primary DNS Server	xxx.1.1.5
Secondary DNS Server	xxx.1.1.6

    b. For eth1, deselect “Activate on Boot”.

14. Under “Firewall Configuration”
    a. Select “Customize”
    b. Select under “Allow Incoming”:
        i. WWW (HTTP)
        ii. SSH
    c. Under other ports, enter 22 for SFTP and 443 for HTTPS.
15. For “Additional Language Support”, select “English (US) -> Next.
16. For “Time Zone Selection”, select “America/New York” for Eastern Time -> Next.
17. Enter the assigned root password -> Next.
18. On the next screen, add the assigned default user and password.
19. For “Authentication Configuration”, select “Enable MD5 passwords” and “Enable shadow passwords” -> Next.
20. On the packages screen, select “Customize the set of packages to be installed” -> Next.
21. Under “Package Group Selection”, install only the following selections:
    a. Applications/File – select stat, a utility for obtaining file information.
    b. Applications/Internet
        i. Select ethereal, a network monitor.
        ii. Select lynx for testing Internet connectivity locally
    c. Applications/System
        i. Select arpwatch to monitor arp requests
        ii. Select autorun for auto-mounting floppies and CD's. As this machine is in a physically secure area, this should present minimal risk.
        iii. Select tripwire to monitor system changes
        iv. Select psacct to install system accounting.
        v. Select procinfo to display system status gathered from /proc
    d. System Environment/Daemons
        i. Select openssh-server
        ii. Select openssh_clients
        iii. Select apache
        iv. Select mod_ssl

e. Under System Environment/Libraries
    i. Select openssl
f. System Environment/shells
    i. Select bash
g. Documentation – select bash-doc for bash shell references.
h. -> Next.

22. Under “Unresolved Dependencies”, select “Install packages to satisfy dependencies” if any unresolved dependencies are listed. -> Next.
23. The software will begin installing at this point. Put CD’s #2 and #3 in as prompted.
24. Create a boot diskette after install, label the diskette with the server name, and ensure the diskette is secured in the installation media cabinet.
25. Remove any CDs and re-boot the system as prompted.
26. This completes the initial installation.

Configuring Red Hat and Hardening the System

Updating the system

Once the system reboots, enter the root username and password to continue with installation. These following steps must be performed as root as they involve system configuration changes and software updates. The first task is to check for updated packages from red Hat. To do so, you must first configure up2date, the Red Hat update manager, for this system. At the command prompt, type:

```
up2date --config --nox
```

to run up2date with a text interface. You will be presented with a numbered list of currently installed and selected options. After updating the configuration, type:

```
up2date --update --nox
```

to check for updates to any installed packages. This will update everything necessary and automatically resolve any dependencies.[11]

Login Banner

Setting a login banner provides a message to all users describing the security policies of this company as well as the monitoring activities associated with use of this server. It also provides a warning as to the expected actions should these policies be ignored. this banner is important from a legal perspective as it may be difficult to prosecute a malicious user if the banner is not present. The company has obtained a banner from their lawyers and it is included on the CD-ROM provided for installing third-party

11 Red Hat #2

software packages. This banner must be copied to /etc/motd to replace the default message of the day.[12]

User Accounts

By default, Red Hat creates unnecessary user accounts. To find a list of these accounts, check /etc/passwd for a list of users created. To determine if a user is active or needs access to a shell, run the following command:

```
find / -user username –print
```

This will list all files and directories owned by that user. If that user has only a home directory, it is a safe bet that the user can be disabled by setting that user's default shell to /bin/false.[13]

Logging

Logging must be configured to enable system administrators (SA's) to search for anomalies and signs of attack or compromise. This company maintains a separate Logging Server in the Protected Server Zone. All output from syslog will be routed to this server. All entries in /etc/syslog.conf have the action @172.16.203.12.

/etc/syslog.conf should be set as follows:

```
# Log anything of level info or higher to the local log files.
# Don't log private authentication messages
#
*.info;mail.none;news.none;authpriv.none;cron.none    /var/log/messages
#
# log the following to the remote log server
#
*.notice    @172.16.203.12
*.warning   @172.16.203.12
*.crit      @172.16.203.12
*.alert     @172.16.203.12
*.emerg     @172.16.203.12
```

SSH

SSH will be the transport layer for administering the system over the network. Only system administrator's desktops have access through SSH to the production servers. These network rules are enforced at the firewall level and also on the local server. Additionally, root will not be allowed to login under SSH; all users must log in under their assigned user names and su to root. This will ensure that all changes made are logged under the user name of the user making the changes. A user using su to change to

12 Chapple, pg 392

13 Bauer, pg 59-60

another user still has all actions logged under the user's personal account.[14] openssh, the SSH implementation used by Red Hat 9, provides the ability to use both the SSH1 and SSH2 protocols. For this implementation, SSH2 will be the only allowed protocol. Edit the /etc/ssh/sshd_config file and add/modify the following lines:

```
#Protocol 2,1
Protocol 2
#PermitRootLogin yes
PermitRootLogin no
Banner /etc/motd
```

This also defines a banner file to be displayed on login. This banner is the same as the message of the day replacement banner discussed above. The commented lines indicate that if the value exists already in the config file, that the existing value should be commented out.

To generate the keys needed for SSH, run:

```
/usr/bin/ssh-keygen -t rsa
```

To check that the SSH server daemon will run, user the following command:

```
chkconfig sshd –list
```

The output should be:

```
sshd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

If sshd is not set to run, use

```
chkconfig --level 2345 linuxconf on
```

Local firewall rules

IP Tables will be used to maintain a local firewall. Netfilter is the part of the Linux kernel that provides hooks for registering callback functions with the network stack. These callback functions allow each packet to be filtered as it passes through the network stack. IP Tables provides those callback functions as well as a rule-set for processing the packets that pass through.[15] IP Tables/netfilter provides a stateful inspection firewall as well as Network Address Translation (NAT). It has the capability to mangle IP headers and can filter based on MAC addresses and on specific network interfaces.[16] There are three default filter chains within IP Tables – INPUT, OUTPUT, and FORWARD. The INPUT chain checks each packet destined for the local machine; the OUTPUT chain checks each packet which originates locally, and the FORWARD chain

14 Chapple, pg 387

15Coulson, pg 1

16Bauer, pg 66

checks each packet sent to the local machine but destined for another host. Other chains can be created if needed.[17]

As the packet moves through the different filter chains, it is checked against the rules defined by that chain in the order in which the rules are defined. If the packet matches a rule, the rule may log it and have it continue passing through the chain; the rule can ACCEPT or DROP it, or the rule can transfer the packet to a different chain. If a packet passes through the entire chain with no match, then the default rule for that chain is applied. For the INPUT, OUTPUT, and FORWARD chains, the default rule is APPLY.[18] The default policy is to DROP all packets which do not map a rule.

IP Tables is configured with a script on system startup. In this script, the different network interfaces are defined as well as all the rules applying to each interface and chain. The IP Tables script is stored in the source code repository running CVS. At regular intervals, the script on the firewall host is compared to the script in CVS to detect any changes. If there are changes, the changes must be investigated to track who made the change and why.

To start with defining the IP Tables ruleset, define the network interface:

```
#!/bin/sh
# Firewall ruleset for GIACE Development Firewall
#
# Define aliases for utilities
#
# External interface
EXTIF=eth0
#Internal interface
INTIF=eth1
IPT=iptables
```

Load the appropriate modules

```
modprobe ip_tables
modprobe ip_conn_track_ftp
```

The second step is to clean out any existing rules by dropping all rules and flushing the tables:

```
# Reset Default Policies
$IPT -P INPUT ACCEPT
$IPT -P FORWARD ACCEPT
$IPT -P OUTPUT ACCEPT
$IPT -t nat -P PREROUTING ACCEPT
$IPT -t nat -P POSTROUTING ACCEPT
$IPT -t nat -P OUTPUT ACCEPT
$IPT -t mangle -P PREROUTING ACCEPT
$IPT -t mangle -P OUTPUT ACCEPT
```

17Coulson, pg 83

18Bauer, pg 67-8

```
# Flush all rules
$IPT -F
$IPT -t nat -F
$IPT -t mangle -F

# Erase all non-default chains
$IPT -X
$IPT -t nat -X
$IPT -t mangle -X

# Set Default Policies to DROP
$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP
```

The next rules allow all loopback interface packets. Many applications communicate using the loopback address locally to pass data over the TCP/IP stack.

```
$IPT -A INPUT -i lo -j ACCEPT
$IPT -A OUTPUT -i lo -j ACCEPT
```

Anti-spoofing policies match against addresses that are non-Internet-routable. It also checks against packets that supposedly originate from the firewall host itself.

```
# Anti-spoofing policies
$IPT -A INPUT -s 255.0.0.0/8 -j LOG -log-prefix "Spoofed IP"
$IPT -A INPUT -s 255.0.0.0/8 -j LOG --log-prefix "Spoofed source IP! "
$IPT -A INPUT -s 255.0.0.0/8 -j DROP
$IPT -A INPUT -s 0.0.0.0/8 -j LOG --log-prefix "Spoofed source IP! "
$IPT -A INPUT -s 0.0.0.0/8 -j DROP
$IPT -A INPUT -s 127.0.0.0/8 -j LOG --log-prefix "Spoofed source IP! "
$IPT -A INPUT -s 127.0.0.0/8 -j DROP
$IPT -A INPUT -s 192.168.0.0/16 -j LOG --log-prefix "Spoofed source IP! "
$IPT -A INPUT -s 192.168.0.0/16 -j DROP
$IPT -A INPUT -s 172.16.0.0/12 -j LOG --log-prefix "Spoofed source IP! "
$IPT -A INPUT -s 172.16.0.0/12 -j DROP
$IPT -A INPUT -s 10.0.0.4 -j ACCEPT
$IPT -A INPUT -s 10.0.0.0/8 -j LOG --log-prefix "Spoofed source IP! "
$IPT -A INPUT -s 10.0.0.0/8 -j DROP
$IPT -A INPUT -s 172.16.204.1 -j LOG --log-prefix "Spoofed Host! "
$IPT -A INPUT -s 172.16.204.1 -j DROP
```

Invalid TCP wrappers indicate a stealth scan attempt:

```
# Block bad TCP wrappers
$IPT -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN FIN,SYN -j DROP
$IPT -A INPUT -p tcp -m tcp --tcp-flags SYN,RST SYN,RST -j DROP
$IPT -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -j DROP
$IPT -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG
FIN,SYN,RST,ACK,URG -j DROP
$IPT -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG
FIN,SYN,RST,PSH,ACK,URG -j DROP
$IPT -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,PSH,URG -
j DROP
```

Force each TCP session to begin with a FYN as protection against stealth scans:

```
# Anti-stealth-scanning rule
$IPT -A INPUT -p tcp ! Syn -m state --state NEW -j LOG --log-prefix "Stealth
Scan Attempt:"
$IPT -A INPUT -p tcp ! Syn -m state --state NEW -j DROP
```

All inbound, outbound, and forwarded packets that are part of previously-ACCEPTed session should be allowed:

```
# Accept inbound, outbound, forward packets that are part of
# a previously-ACCEPTed sessions
$IPT -A INPUT   -j ACCEPT -m state --state ESTABLISHED,RELATED
$IPT -A OUTPUT  -j ACCEPT -m state --state ESTABLISHED,RELATED
$IPT -A FORWARD -j ACCEPT -m state --state ESTABLISHED,RELATED
```

Our SAs are allowed to connect to the internal development servers using SSH.

```
# Accept inbound packets which initiate SSH sessions SAs
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.201.200 --dport 23 -m state --state
NEW
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.201.201 --dport 23 -m state --state
NEW
```

Our SAs are allowed to connect to the internal development servers using SFTP.

```
# Accept inbound packets which initiate SFTP sessions from SAs
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.201.200 --dport 22 -m state --state
NEW
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.201.201 --dport 22 -m state --state
NEW
```

All internal users can connect to the HTTP/S servers in the development zone.

```
# Accept inbound packets which initiate HTTP/S sessions - accepted from all
# to support testing by other groups
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.201.0/24 --dport 80 -m state --state
NEW
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.201.0/24 --dport 443 -m state --
state NEW
```

Apache needs to talk to Tomcat over the defined listener ports:

```
#
# Accept connections to the Tomcat listeners
#
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.204.3 --dport 8007 -m state --state
NEW
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.204.3 --dport 8009 -m state --state
NEW
```

Log anything not accepted in the INPUT chain:

```
# Log anything not accepted above
$IPT -A INPUT -j LOG --log-prefix "Dropped by default:"
```

Begin the OUTPUT chain, allowing DNS and NTP queries, syslogd as well as outbound pings:

```
# Output

# If part of an already approved connection
$IPT -I OUTPUT 1 -m state --state RELATED,ESTABLISHED -j ACCEPT

# Allow outbound ping for troubleshooting
$IPT -A OUTPUT -p icmp -j ACCEPT --icmp-type echo-request

# Allow outbound DNS queries
$IPT -A OUTPUT -p udp --dport 53 -m state --state NEW -j ACCEPT

# Allow outbound NTP queries
$IPT -A OUTPUT -o $dev -p udp -dport 123 -d 172.16.203.12 -j ACCEPT

# Allow outbound syslogd connections
$IPT -A OUTPUT -p udp --dport 514 -m state --state NEW -j ACCEPT
```

Log anything not accepted in the OUTPUT chain:

```
# Log everything else
$IPT -A OUTPUT -j LOG --log-prefix "Dropped by default: "
```

Hardware Configuration Changes

Now that the install is complete, reboot the system again. On startup, enter the CMOS settings and add the assigned CMOS password. Disable the floppy drive to prevent any malicious user from booting from a floppy to gain access should physical security be breached. also disable all serial and parallel ports.[19] Edit `/etc/inittab` and ensure the following line is commented out:

```
#no C-A-D:  ca::ctrlaltdel:/sbin/shutdown –t5 –rfn now
```

This will prevent the Ctrl-Alt-Del key sequence from shutting down the server.[20]

Re-boot the server to allow all of these changes to take place.

Cron

19 Toxen, pg 127-30

20 Toxen, pg 130

To prevent any user other than root from creating cron or at jobs, create the files `/etc/cron.allow` and `/etc/at.allow` with the following entry:

```
root
```

this will allow cron jobs to be run by other users, but only created by root.

To add NTP synchronization on an hourly basis, create file /etc/cron.hourly/time.cron with the following entry. This will run ntpdate as a cron job every hour.

```
/usr/bin/ntpdate -b 172.16.203.12
```

Set the permissions on the file using the command:

```
chmod a+x /etc/cron.hourly/time.cron
```

Install Apache

The Apache web server is part of the standard installation discussed above. At this point, the goal is to ensure Apache is up-to-date and configured in as secure a manner as is possible. The Red Hat update manager, up2date, was run in the previous steps to update Apache to the latest version. The current version of Apache is 2.0.48 (as of this writing).

Create Apache User

In order to better control Apache security, create a user and a group for Apache, both named *apache*. The group will have access to the Apache documents and web files being served and the web server (running as user *apache*) will have access to serve those files. The only users in the group *apache* will be the user *apache* and the system administrators for this server. To set up file permissions, run `umask 137` and `chmod 660` for each content file and directory in order to allow the users in the group *apache* to read and write the Apache content files. Write permissions are allowed so that any system administrator for this server can update the files as needed. The directories to be modified (include all sub-directories of each) are:

File type	Directory	Owner	File Mode
Apache Utilities	/usr/sbin	root	Directory 755 Files 755
CGI Programs	/var/www/cgi-bin	Directory owned by root; files owned by apache	Directory 755 Files 750
Static Content	/var/www/html	apache	Directory 470 Files 660
Config files	/etc/http/conf	root	Directory 755 Files 644

Initialization script	/etc/init.d	root	Directory 755 Files 755
Logs	/etc/httpd/logs	root	Directory 755 Files 644

Configuration

In /etc/httpd/conf/access.conf, ensure the following entry exists:

```
<Directory /var/www>
    Options SymLinksIfOwnerMatch IncludesNoExec
    AllowOverride none
    Order deny,allow
    Allow from 127.0.0.1
    Allow from 172.16.204.3
</Directory>
```

This will effectively disable CGI scripts and allow symbolic link use in the subdirectories if the target and the link have the same owner. As this server will run as a Java application server, there is no need for CGI.

Delete all default example files installed by Apache in the /var/www/cgi-bin and /var/www/html directories. These files will not be used be the current Java application running on this server and any files not used should be removed to lessen the chance of introducing a vulnerability on the server.

Certificates

Copy the existing key file to:

```
/etc/httpd/conf/ssl.key/server.key
```

Copy the existing certificate file to:

```
/etc/httpd/conf/ssl.crt/server.crt
```

Install Java

This application server will run the Java J2SE version 1.4.2.03. It was downloaded from Sun Microsystems (available at http://java.sun.com/j2se/1.4.2/download.html as of 14 February 2004) as an RPM and placed on a CD-ROM after being verified for integrity and security. To install the RPM:

1. Copy the file from the CD-ROM to your home directory.

2. Change directory to your home directory and set the permissions on the copied file to executable to all:

```
chmod 777 j2re-1_4_2_<version>-linux-i586-rpm.bin
```

3. Run the binary to extract the RPM file:

```
./j2re-1_4_2_03-linux-i586-rpm.bin
```

4. Note that the initial "./" is required as "." is not and should not be in your PATH environment variable be default.

5. The will create the file `j2re-1_4_2_03-linux-i586.rpm` in the current directory.

6. su to root

7. Run the rpm command to install the Java 2 Runtime Environment

```
rpm -iv j2re-1_4_2_03-linux-i586.rpm
```

8. Change directories to /usr/java and set permissions on the Java home directory

```
chmod 755 j2re-1_4_2_03
```

9. Create a symbolic link to the Java installation:

```
cd /usr/java
ln -s j2re-1_4_2_03 java
```

10. Delete the rpm and bin files in your home directory.

Install Apache Tomcat

Apache Tomcat version 4.1.29 was downloaded as an RPM from http://www.jpackage.org. This package has been installed on the development and QA servers and thoroughly tested in these environments to ensure that it can be trusted for production. Tomcat's log files will be stored in `/var/tomcat4/logs/`.

1. Copy the RPM from the supplied CD-ROM to your home directory.
2. Change directories to your home directory and run the following command:

```
rpm –iv
```

3. su to root
4. Move Tomcat to its destination directory and change to that directory:

```
cd /usr/share
```

5. Check the permissions on the Tomcat directory. If the permissions are not set correctly, run the following:

```
chmod 755 tomcat4
```

6. Modify the Tomcat configuration file at `/etc/tomcat4/conf/tomcat4.conf`.

7. Change the JAVA_HOME reference to

```
JAVA_HOME=/usr/java/java
```

8. Set the CATALINA_HOME variable:

```
CATALINA_HOME=/usr/share/tomcat4
```

9. Set the classpath:

```
CLASSPATH=$CLASSPATH:/var/tomcat4/lib:/usr/share/pgsql:/usr/java/java/jre/lib
:/usr/java/ava/lib:
```

Install mod_jk2

mod_jk2 is the connector between Apache 2 and Tomcat that allows Apache to serve static content but server JSPs and servlets using Tomcat. The mod_jk2 RPM was also downloaded from http://www.jpackage.org.

Configure Apache and Tomcat

To configure Apache to run with Tomcat:[21]

1. Modify the file /etc/httpd/conf/workers.properties:

```
workers.tomcat_home=/var/tomcat4
workers.java_home=/usr/java/java
ps=/
worker.list=ajp12, ajp13

#
# Define a worker named ajp12 and of type ajp12
# Note that the name and the type do not have to match.
#
worker.ajp12.port=8007
worker.ajp12.host=localhost
worker.ajp12.type=ajp12
worker.ajp12.lbfactor=1
```

[21] Ippolito

```
#
# Define a worker named Ajp13
#
worker.ajp13.port=8009
worker.ajp13.host=localhost
worker.ajp13.type=ajp13
worker.ajp13.lbfactor=1

worker.loadbalancer.type=lb
worker.loadbalancer.balanced_workers=ajp12, ajp13

#
# Defining a worker named inprocess and of type jni
#
worker.inprocess.type=jni

#
# Additional class path components.
#
worker.inprocess.class_path=$(workers.tomcat_home)$(ps)server$(ps)lib$(ps)tom
cat-ajp.jar    - Changed this line to match location of library.
worker.inprocess.cmd_line=start
worker.inprocess.class_path=$(workers.java_home)$(ps)lib$(ps)tools.jar   -
Added this line based on something I read. (Also modified to match
installation)

#
# Setting the place for the stdout and stderr of tomcat
#
worker.inprocess.stdout=$(workers.tomcat_home)$(ps)logs$(ps)inprocess.stdout
worker.inprocess.stderr=$(workers.tomcat_home)$(ps)logs$(ps)inprocess.stderr
```

This configuration is used to define socket listeners and communication protocols between Apache and Tomcat.

2. Modify /var/tomcat4/conf/server.xml by adding within the following XML tags:

```
<Service name="Tomcat-Standalone">
...
</Service>
```

Define a listener for Apache by adding definitions for connectors on ports 8007 and 8009 for workers:

```
    <Connector className="org.apache.ajp.tomcat4.Ajp13Connector"
               port="8007" minProcessors="5" maxProcessors="75"
               acceptCount="10" debug="0"/>

    <Connector className="org.apache.ajp.tomcat4.Ajp13Connector"
               port="8009" minProcessors="5" maxProcessors="75"
               acceptCount="10" debug="0"/>
```

3. Modify /etc/httpd/conf/httpd.conf.

a. Add the following line after all other LoadModule statements:

```
LoadModule jk_module         modules/mod_jk.so
```

b. Add the following line after the line ClearModuleList and after all AddModule statements:

```
AddModule mod_jk.c
```

c. Add the following:

```
<IfModule mod_jk.c>

    JkWorkersFile /etc/httpd/conf/workers.properties
    JkLogFile /var/log/httpd/mod_jk.log
    JkLogLevel info

    # Root context mounts for Tomcat
    # Format: JkMount URL_PREFIX WORKER_NAME
    JkMount /*.jsp ajp13
    JkMount /servlet/* ajp13

    # The following line makes apache aware of the location of the /examples
contextAlias /examples "/var/tomcat4/webapps/examples"
    <Directory "/var/tomcat4/webapps/examples">
        Options Indexes FollowSymLinks
    </Directory>

    # The following line mounts all JSP files and the /servlet/ uri to tomcat
    JkMount /examples/servlet/* ajp13
    JkMount /examples/*.jsp ajp13

    # The following line prohibits users from directly access WEB-INF
    <Location "/examples/WEB-INF/">
        AllowOverride None
        deny from all
    </Location>
</IfModule>
```

4. Modify file /etc/httpd/conf/mod_jk.conf. Change all references of "tomcat" to "tomcat4"

5. Create the startup script for Tomcat as follows (/etc/rc.d/init.d/tomcat):

```
#!/bin/bash
#
# chkconfig
# description: Startup script for Tomcat
# Source function lib
. /etc/init.d/functions

RETVAL=$?
export JAVA_HOME=/usr/java/java
```

```
export CATALINA_HOME=/usr/share/tomcat

case "$1" in
    start)
        if [ -f $CATALINA_HOME/bin/startup.sh ];
            then
                echo $"Starting Tomcat"
                /bin/su tomcat $CATALINA_HOME/bin/startup.sh -security
        fi
        ;;
    stop)
        if [ -f $CATALINA_HOME/bin/shutdown.sh ];
            then
                echo $"Stopping Tomcat"
                /bin/su tomcat $CATALINA_HOME/bin/ shutdown.sh
        fi
        ;;
    *)
        echo $"Usage: $0 {start|stop}"
        exit 1;
        ;;

esac

exit $RETVAL
```

The above-listed script starts up Tomcat with the Tomcat Security Manager rather than the inherited Java security manager. This allows Tomcat's security parameters to be configured through Tomcat, allowing finer-grained control over web applications.[22]

1. Modify the Apache ssl.conf file as follows in order to support Tomcat:[23]

```
<VirtualHost _default_:443>
# General setup for the virtual host
DocumentRoot "/usr/sbin/htdocs/secure/myhost"
ServerName www.some-company-name.com:443
ServerAdmin webmaster@some-company-name.com
ErrorLog logs/error_log
TransferLog logs/access_log


# Static files
Alias /appname"/usr/share/tomcat/webapps/appname"


<Directory "/usr/share/tomcat/webapps/appname">
  Options Indexes FollowSymLinks
  DirectoryIndex index.jsp
</Directory>

# Deny direct access to WEB-INF and META-INF
```

22 Chopra, et al, pg 62

23 Carrillo

```
<Location "/appname/WEB-INF/*">
  AllowOverride None
  deny from all
</Location>

<Location "/appname/META-INF/*">
  AllowOverride None
  deny from all
</Location>

JkMount /appname/*.do ajp13
JkMount /appname/*.jsp ajp13

JkMount /appname ajp13
JkMount /appname/* ajp13

</VirtualHost>
```

Set Java/Tomcat Security

The Tomcat server runs under the security settings defined by the servlet container administrator. The administrator can control access to the local file system, to other network hosts, to system properties, and to certain Java classes which could be used by rogue code to compromise the Java application server. In order to ensure only that access needed, the Tomcat Security Manager will be configured with the defaults provided with Tomcat with the exceptions noted below:[24]

1. Read-only access to the default system properties defining the Java environment
2. Read/Write access to the file used as the Java web applications log file

```
grant codeBase "file:${catalina.home}/webapps/appname/WEB-INF/classes/
com.some-company-name.logging/-"
{
    permission java.io.FilePermission "$(catalina.home}/logs/appname/-",
"read, write";
};
```

3. Permissions for the Java application to connect to the database server on port 3306 (MySQL server) through the MySQL jar file.

```
grant codeBase "jar:file:${catalina.home}/webapps/appname/WEB-
INF/lib/mysql.jar"
{
    permission java.net.SocketPermission "172.16.203.10:3306", "connect";
};
```

Tripwire

24 Chopra, et al, pg 63-76

Tripwire (available from http://www.tripwire.org) is an open source tool that computes MD5 checksums on each file you want to track for changes. Once the MD5 checksum has been created, any change to that file will cause a change to the checksum value. Checking these values over time will indicate which files have been modified and can be used to point out attempts to alter the integrity of this server. At the end of the initial installation, and after every production push, a Tripwire checksum will be computer for all files in /etc, /var/www, and /usr/share/tomcat to establish a baseline for later comparisons. These checksums will be stored on CD-ROM along with the image created for backups (covered in the section on Ongoing Maintenance).[25]

Tripwire was installed with the initial operating system install. To setup Tripwire, su to root and issue the following command:

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

When asked, enter the strong passwords supplied for the site and local Tripwire passphrases. Tripwire is installed by default into `/usr/sbin/tripwire`. An entry is also added to cron to run Tripwire daily. This entry will be left enabled in order to have a daily snapshot of any file changes.

Edit the twpol.txt file at /etc/tripwire to enter the Apache and Tomcat configuration files to be tracked.

```
# Apache config files
/etc/httpd/conf/access.conf
/etc/httpd/conf/apache.conf
/etc/httpd/conf/ssl.conf
# Tomcat policy file
/usr/share/tomcat/conf/catalina.policy
```

Create the Tripwire policy file by running:

```
/usr/sbin/twadmin --create-polfile twpol.txt
```

Initialize the Tripwire database by running:

```
/usr/sbin/tripwire --init
```

Run an integrity check with the following command:

```
/usr/sbin/tripwire --check
```

Ongoing maintenance

25 Toxen, pg 651-3

Tracking Vulnerabilities

New vulnerabilities are discovered daily. In order to provide the highest level of protection possible to this and all other servers running in this company, operating system and application patches must be checked for, downloaded, and installed on a regular basis. Checks are performed each day, with patches downloaded for verification and testing. Once the patches have been tested, the changes to the production environment are scheduled based on the severity of the perceived threat. Known sources for discovering new vulnerabilities are:

- Computer Emergency response Team Coordination Center (CERT/CC) at http://www.cert.org/advisories
- Red Hats errata list at http://www.redhat.com/support/errata/.
- Apache's web Site at http://httpd.apache.org/
- Tomcat's web Site at http://jakarta.apache.com
- The Apache Bug Database at http://nagoya.apache.org/bugzilla/query.cgi?product=Tomcat%204.
- Common Vulnerabilities and Exposures at http://cve.mitre.org/.

Change Control/Patches

Keeping the system up to date has been greatly simplified with the commercial version of Red Hat 9. The utility up2date allows system administrators to connect to Red Hat through the Internet and automatically update their systems with the latest patches from Red Hat. However, no patch should be applied until it has been thoroughly tested in other environments. The change control process for this company consists of the following steps:

1. Identify the changes required. As was mentioned above, there are many sources for determining required changes. Web sites and email lists are the ones most often used. Once the need for a change, or patch, has been identified, it should be logged in the company change control database.
2. Download the patches to a development/test server located internally.
3. Verify the patch using its MD5 hash as well as the GnuPG signature.
4. Apply the patch to the development server.
5. Test the patch.
6. The patch will then move through the same change control process as for internal software updates, to include:
    a. Install in QA
    b. Test in QA
    c. Schedule production push
    d. Install in production
        i. Backup production configuration files
        ii. Apply patches
        iii. Restart services/server as necessary

e. Verify in production
f. Monitor production

Backups

No dynamic application content is stored on this production server. All dynamic content is derived from the database, with all application audit logging stored there as well. The log files for the server itself are sent to the Logging Server in the Protected Server Zone. As such, backup procedures for the production server include backing up the contents of the Apache static web files, .the Tomcat webapps, and the Apache and Tomcat configuration files. These backups are created after each production push through the current build process. The Apache and Tomcat applications are burned to a CD-ROM with each subsequent push producing a new CD-ROM. The source code control server also is backed up on a nightly basis, allowing the company to re-create and changes necessary to keep the production server up to date (i.e., hot fixes).

Should this server go down or be compromised, the recovery time to re-build the server from scratch is estimated to be four hours. The management of the company has decided that that is an acceptable time to be down. As the company progresses, this policy will be re-evaluated and additional measures taken to ensure rapid recovery.

Check your configuration

General Checks

After installation is complete, run the following general checks to ensure the system is properly configured:

1. List all open ports and the processes which are running on these ports. The only open ports should be 80, 443, 22, 23, 8007, and 8009:

```
netstat –punta
netstat –nlp
```

2. List any RPC services that are running -- there should be no RPC services running. As root, run:

```
rpcinfo -p localhost
```

3. Verify NTP connectivity to the Protected Server Zone:

```
/usr/bin/ntpdate -b 172.16.203.12
```

4. Verify Apache configuration by running the following command. The response should be `Syntax OK`. If not, check the Apache configuration files for proper syntax.

```
/usr/sbin/apachectl configtest
```

5. Verify Tomcat listeners by connecting to http://172.16.203.3/examples/servlet/MyHelloWorld. A successful Apache/Tomcat response indicates that Tomcat and Apache are configured correctly. Connecting to http://172.16.203.3/ should return static content from Apache.

nmap

A port scan using nmap (available at http://www.insecure.org/nmap/) will show which ports are open on this server. The syntax is:

```
nmap -sTUR -F -Po -O 172.16.204.3
```

which will conduct a TCP Connect scan against all privileged ports (0 – 1023) plus the more common registered ports. The -Po option indicates no ping, as a failed ping will terminate the program. The -O option attempts an OS Fingerprinting just as a sanity check as to how much information is being exposed. The options under -s provide for TCP Connect, UDP, and RPC scans.[26]

nmap must be run from another system on the same subnet as this server. Running from the same subnet ensures there are no routers or firewalls blocking attempts to open ports on this server. The goal is to check that this server is secure without firewall intervention. On running nmap, the only open ports that should be found are port 80 for HTTP requests and port 443 for HTTPS requests. The SSH (port 23) and SFTP (port 22) ports should not be seen as the IP Tables configuration should only allow connections from the 172.16.201.0 network. Ports 8007 and 8009 (Tomcat listeners) should not be seen as they are only available from the Java application server itself.

If a port is listed that is not expected, run the command `cat /etc/services | grep portnum` on the Java application server to see if the port matches one of the list of known ports. For example, if port 8080 shows up on the list of exposed ports (from nmap), run the command:

```
/etc/services | grep 8080.
```

If the previous command does not provide the information, try:

```
lsof -i | grep 8080
```

26 Bauer, pg 83

which will display which processes are using that port. You can then check the processes shown for proper configuration.

A successful run should show output similar to this:

```
PORT       STATE   SERVICE   VERSION
 80/tcp    open    http      Apache httpd 2.0.48 (mod_ssl/...)
 443/tcp   open    ssl/http  Apache httpd 2.0.48 ((Unix) mod_gzip/ ...)
```

For this installation, the nmap scan checks that:

- Ports 80 and 443 are open and accepting connections, which indicates a properly configured Apache and Tomcat installation.
- The fact that the nmap scan fails on ports 22 and 23 shows that IP Tables is properly configured.
- nmap failing on ports 8007 and 8009 shows that IP Tables is properly configured to only allow connections from the Java application server itself.

Nessus

A Nessus scan will indicate security holes in the server's configuration. Nessus not only scans for open ports on the server, but attempts to check these ports to see what software is running on them, to include the version of that software. This information is then compared against a list of known vulnerabilities for each software package found. As vulnerabilities are uncovered, the proper steps for updating the required software packages must be taken immediately -- before this server goes into production. Nessus also must be run from another system on the same subnet for the same reasons mentioned under the nmap description. Nessus is available at http://www.nessus.org/.[27]

To run Nessus against this server, select the open ports found with the nmap scan (ports 80 and 443). Enter the server's IP Address in the target list and begin the scan. The results should show any vulnerabilities that might exist in the installed software.

27 Bauer, pg 84-5

Appendix A: IP Tables ruleset

```
#!/bin/sh
# Firewall ruleset for GIACE Development Firewall
#
# Define aliases for utilities
#
# External interface
EXTIF=eth0
#Internal interface
INTIF=eth1
IPT=iptables
#
# Load the appropriate modules
#
modprobe ip_tables
modprobe ip_conn_track_ftp
#
# Reset Default Policies
$IPT -P INPUT ACCEPT
$IPT -P FORWARD ACCEPT
$IPT -P OUTPUT ACCEPT
$IPT -t nat -P PREROUTING ACCEPT
$IPT -t nat -P POSTROUTING ACCEPT
$IPT -t nat -P OUTPUT ACCEPT
$IPT -t mangle -P PREROUTING ACCEPT
$IPT -t mangle -P OUTPUT ACCEPT
#
# Flush all rules
$IPT -F
$IPT -t nat -F
$IPT -t mangle -F
#
# Erase all non-default chains
$IPT -X
$IPT -t nat -X
$IPT -t mangle -X
#
# Set Default Policies to DROP
$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP
#
$IPT -A INPUT -i lo -j ACCEPT
$IPT -A OUTPUT -i lo -j ACCEPT
#
# Anti-spoofing policies
$IPT -A INPUT -s 255.0.0.0/8 -j LOG -log-prefix "Spoofed IP"
$IPT -A INPUT -s 255.0.0.0/8 -j LOG --log-prefix "Spoofed source IP! "
$IPT -A INPUT -s 255.0.0.0/8 -j DROP
$IPT -A INPUT -s 0.0.0.0/8 -j LOG --log-prefix "Spoofed source IP! "
$IPT -A INPUT -s 0.0.0.0/8 -j DROP
$IPT -A INPUT -s 127.0.0.0/8 -j LOG --log-prefix "Spoofed source IP! "
$IPT -A INPUT -s 127.0.0.0/8 -j DROP
$IPT -A INPUT -s 192.168.0.0/16 -j LOG --log-prefix "Spoofed source IP! "
```

```
$IPT -A INPUT -s 192.168.0.0/16 -j DROP
$IPT -A INPUT -s 172.16.0.0/12 -j LOG --log-prefix "Spoofed source IP! "
$IPT -A INPUT -s 172.16.0.0/12 -j DROP
$IPT -A INPUT -s 10.0.0.4 -j ACCEPT
$IPT -A INPUT -s 10.0.0.0/8 -j LOG --log-prefix "Spoofed source IP! "
$IPT -A INPUT -s 10.0.0.0/8 -j DROP
$IPT -A INPUT -s 172.16.204.1 -j LOG --log-prefix "Spoofed Host!"
$IPT -A INPUT -s 172.16.204.1 -j DROP
#
# Block bad TCP wrappers
$IPT -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN FIN,SYN -j DROP
$IPT -A INPUT -p tcp -m tcp --tcp-flags SYN,RST SYN,RST -j DROP
$IPT -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -j DROP
$IPT -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG
FIN,SYN,RST,ACK,URG -j DROP
$IPT -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG
FIN,SYN,RST,PSH,ACK,URG -j DROP
$IPT -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,PSH,ACK,URG FIN,PSH,URG -
j DROP
#
# Anti-stealth-scanning rule
$IPT -A INPUT -p tcp ! Syn -m state --state NEW -j LOG --log-prefix “Stealth
Scan Attemp:”
$IPT -A INPUT -p tcp ! Syn -m state --state NEW -j DROP
#
# Accept inbound, outbound, forward packets that are part of
# a previously-ACCEPTed sessions
$IPT -A INPUT   -j ACCEPT -m state --state ESTABLISHED,RELATED
$IPT -A OUTPUT  -j ACCEPT -m state --state ESTABLISHED,RELATED
$IPT -A FORWARD -j ACCEPT -m state --state ESTABLISHED,RELATED
#
# Accept inbound packets which initiate SSH sessions SAs
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.201.200 --dport 23 -m state --state
NEW
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.201.201 --dport 23 -m state --state
NEW
#
# Accept inbound packets which initiate SFTP sessions from SAs
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.201.200 --dport 22 -m state --state
NEW
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.201.201 --dport 22 -m state --state
NEW
#
# Accept inbound packets which initiate HTTP/S sessions - accepted from all
# to support testing by other groups
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.201.0/24 --dport 80 -m state --state
NEW
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.201.0/24 --dport 443 -m state --
state NEW

#
# Accept connections to the Tomcat listeners
#
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.204.3 --dport 8007 -m state --state
NEW
$IPT -A INPUT -p tcp -j ACCEPT -s 172.16.204.3 --dport 8009 -m state --state
NEW
```

```
#
# Log anything not accepted above
$IPT -A INPUT -j LOG --log-prefix "Dropped by default:"
#
# Output
#
# If part of an already approved connection
$IPT -I OUTPUT 1 -m state --state RELATED,ESTABLISHED -j ACCEPT
#
# Allow outbound ping for troubleshooting
$IPT -A OUTPUT -p icmp -j ACCEPT --icmp-type echo-request
#
# Allow outbound DNS queries
$IPT -A OUTPUT -p udp --dport 53 -m state --state NEW -j ACCEPT
#
# Allow outbound NTP queries
$IPT -A OUTPUT -o $dev -p udp -dport 123 -d 172.16.203.12 -j ACCEPT
#
# Allow outbound syslogd connections
$IPT -A OUTPUT -p udp --dport 514 -m state --state NEW -j ACCEPT
#
# Log everything else
$IPT -A OUTPUT -j LOG --log-prefix "Dropped by default: "
```

Appendix B: Catalina Policy File

```
// ============================================================================
// catalina.corepolicy - Security Policy Permissions for Tomcat 4.0
//
// This file contains a default set of security policies to be enforced (by
the
// JVM) when Catalina is executed with the "-security" option.  In addition
// to the permissions granted here, the following additional permissions are
// granted to the codebase specific to each web application:
//
// * Read access to the document root directory
//
// $Id: catalina.policy,v 1.28 2002/09/30 19:59:47 glenn Exp $
// ============================================================================


// ========== SYSTEM CODE PERMISSIONS =========================


// These permissions apply to javac
grant codeBase "file:${java.home}/lib/-" {
  permission java.security.AllPermission;
};

// These permissions apply to all shared system extensions
grant codeBase "file:${java.home}/jre/lib/ext/-" {
  permission java.security.AllPermission;
};

// These permissions apply to javac when ${java.home] points at
$JAVA_HOME/jre
grant codeBase "file:${java.home}/../lib/-" {
  permission java.security.AllPermission;
};

// These permissions apply to all shared system extensions when
// ${java.home} points at $JAVA_HOME/jre
grant codeBase "file:${java.home}/lib/ext/-" {
  permission java.security.AllPermission;
};


// ========== CATALINA CODE PERMISSIONS ============


// These permissions apply to the server startup code
grant codeBase "file:${catalina.home}/bin/bootstrap.jar" {
  permission java.security.AllPermission;
};

// These permissions apply to the servlet API classes
// and those that are shared across all class loaders
// located in the "common" directory
grant codeBase "file:${catalina.home}/common/-" {
```

```
  permission java.security.AllPermission;
};

// These permissions apply to the container's core code, plus any additional
// libraries installed in the "server" directory
grant codeBase "file:${catalina.home}/server/-" {
  permission java.security.AllPermission;
};

// These permissions apply to the jasper page compiler.
grant codeBase "file:${catalina.home}/shared/lib/jasper-compiler.jar" {
  permission java.security.AllPermission;
};

// These permissions apply to the jasper JSP runtime
grant codeBase "file:${catalina.home}/shared/lib/jasper-runtime.jar" {
  permission java.security.AllPermission;
};

// These permissions apply to the privileged admin and manager web
applications
grant codeBase "file:${catalina.home}/server/webapps/admin/WEB-INF/classes/-"
{
  permission java.security.AllPermission;
};

grant codeBase "file:${catalina.home}/server/webapps/admin/WEB-
INF/lib/struts.jar" {
  permission java.security.AllPermission;
};

// ========== WEB APPLICATION PERMISSIONS ===============


// These permissions are granted by default to all web applications
// In addition, a web application will be given a read FilePermission
// and JndiPermission for all files and directories in its document root.
grant {
  // Required for JNDI lookup of named JDBC DataSource's and
  // javamail named MimePart DataSource used to send mail
  permission java.util.PropertyPermission "java.home", "read";
  permission java.util.PropertyPermission "java.naming.*", "read";
  permission java.util.PropertyPermission "javax.sql.*", "read";

  // OS Specific properties to allow read access
  permission java.util.PropertyPermission "os.name", "read";
  permission java.util.PropertyPermission "os.version", "read";
  permission java.util.PropertyPermission "os.arch", "read";
  permission java.util.PropertyPermission "file.separator", "read";
  permission java.util.PropertyPermission "path.separator", "read";
  permission java.util.PropertyPermission "line.separator", "read";

  // JVM properties to allow read access
  permission java.util.PropertyPermission "java.version", "read";
  permission java.util.PropertyPermission "java.vendor", "read";
  permission java.util.PropertyPermission "java.vendor.url", "read";
  permission java.util.PropertyPermission "java.class.version", "read";
```

```
  permission java.util.PropertyPermission "java.specification.version",
"read";
  permission java.util.PropertyPermission "java.specification.vendor",
"read";
  permission java.util.PropertyPermission "java.specification.name", "read";

  permission java.util.PropertyPermission "java.vm.specification.version",
"read";
  permission java.util.PropertyPermission "java.vm.specification.vendor",
"read";
  permission java.util.PropertyPermission "java.vm.specification.name",
"read";
  permission java.util.PropertyPermission "java.vm.version", "read";
  permission java.util.PropertyPermission "java.vm.vendor", "read";
  permission java.util.PropertyPermission "java.vm.name", "read";

  // Required for getting BeanInfo
  permission java.lang.RuntimePermission "accessClassInPackage.sun.beans";
  permission java.lang.RuntimePermission "accessClassInPackage.sun.beans.*";

  // Required for sevlets and JSP's
  permission java.lang.RuntimePermission
"accessClassInPackage.org.apache.catalina.util";
  permission java.lang.RuntimePermission
"accessClassInPackage.org.apache.catalina.util.*";
  permission java.lang.RuntimePermission
"defineClassInPackage.org.apache.catalina.util";
  permission java.lang.RuntimePermission
"defineClassInPackage.org.apache.catalina.util.*";

  // Required for running servlets generated by JSPC
  permission java.lang.RuntimePermission
"accessClassInPackage.org.apache.jasper.runtime";
  permission java.lang.RuntimePermission
"accessClassInPackage.org.apache.jasper.runtime.*";

  // Required for OpenJMX
  permission java.lang.RuntimePermission "getAttribute";

  // Allow read of JAXP compliant XML parser debug
  permission java.util.PropertyPermission "jaxp.debug", "read";
};


// You can assign additional permissions to particular web applications by
// adding additional "grant" entries here, based on the code base for that
// application, /WEB-INF/classes/, or /WEB-INF/lib/ jar files.
//
// Different permissions can be granted to JSP pages, classes loaded from
// the /WEB-INF/classes/ directory, all jar files in the /WEB-INF/lib/
// directory, or even to individual jar files in the /WEB-INF/lib/ directory.
//
// For instance, assume that the standard "examples" application
// included a JDBC driver that needed to establish a network connection to
the
// corresponding database and used the scrape taglib to get the weather from
// the NOAA web server.  You might create a "grant" entries like this:
```

```
//
// The permissions granted to the context root directory apply to JSP pages.
// grant codeBase "file:${catalina.home}/webapps/examples/-" {
//   permission java.net.SocketPermission "dbhost.mycompany.com:5432",
"connect";
//   permission java.net.SocketPermission "*.noaa.gov:80", "connect";
// };
//
// The permissions granted to the context WEB-INF/classes directory
// grant codeBase "file:${catalina.home}/webapps/examples/WEB-INF/classes/-"
{
// };
//
// The permission granted to your JDBC driver
// grant codeBase "file:${catalina.home}/webapps/examples/WEB-
INF/lib/driver.jar" {
//   permission java.net.SocketPermission "dbhost.mycompany.com:5432",
"connect";
// };
// The permission granted to the scrape taglib
// grant codeBase "file:${catalina.home}/webapps/examples/WEB-
INF/lib/scrape.jar" {
//   permission java.net.SocketPermission "*.noaa.gov:80", "connect";
// };

grant codeBase "file:${catalina.home}/webapps/appname/WEB-INF/classes/
com.some-company-name.logging/-"
{
    permission java.io.FilePermission "$(catalina.home}/logs/appname/-",
"read, write";
};
grant codeBase "jar:file:${catalina.home}/webapps/appname/WEB-
INF/lib/mysql.jar"
{
    permission java.net.SocketPermission "172.16.203.10:3306", "connect";
};
```

References

Apache, “The Apache Software Foundation”, 14 Feb 2004 , http://www.apache.org/

Apache Jakarta, “The Apache Jakarta Project”, 14 Feb 2004, http://jakarta.apache.org/

Bauer, Michael D., Building Secure Servers with Linux, Sebastopol, CA: O'Reilly& Associates, Inc, 2003

Carrillo, Oscar, "HOWTO : Installing Web Services with Linux / Tomcat / Apache / Struts / Postgresql / OpenSSL / JDBC / JNDI", 2 Feb 2004, http://www.linuxjava.net/howto/webapp/

Chapple, Mike, The GSEC Prep Guide, Indianapolis, IN: Wiley Publishing, Inc, 2003

Coulson, David, “Mastering IP Tables”, Linux Format, May 2001 (82-87)

Dhanjani, Nitesh, HackNotes Linux and Unix Security Portable Reference, Emeryville, CA: McGraw-Hill/Osborne, 2003

GnuPG, “The GNU Privacy Guard”, 14 Feb 2004, http://www.gnupg.org/(en)/index.html

Ippolito, Greg, “YoLinux Tutorial: Java Servlets, JSP, Jakarta-Tomcat, a Database (PostgreSQL or MySQL), Apache and Linux”, 2002, http://www.yolinux.com/TUTORIALS/LinuxTutorialTomcat.html

LeBlanc, De-Ann, “Linux Partitions: A Primer”, 20 Jun 2002, http://www.linuxplanet.com/linuxplanet/tutorials/4269/1/”

The Mitre Corporation, “The Key to Information Sharing”, 14 Feb 2004, http://cve.mitre.org/docs/docs2000/key_to_info_shar.pdf

Peikari, Cyrus & Chuvakin, Anton, Security Warrior, Sebastopol, CA: O'Reilly& Associates, Inc, 2004

Penguin Computing, “Relion 1X Server”, 14 Feb 2004, http://www.penguincomputing.com/store/configurator.php

Red Hat, Red Hat Linux 9: Red Hat Linux x86 Installation, Guide, Raleigh, NC: Red Hat, Inc. 2003

Red Hat #2, “Tips and Tricks. Featured Article: Using the Update Agent (up2date) from the command line”, Apr 2003, http://www.redhat.com/advice/tips/up2date.html

Sun Microsystems, “Download Java 2 Platform, Standard Edition, v 1.4.2 (J2SE)”, 14 Feb 2004, http://java.sun.com/j2se/1.4.2/download.html

Tacket Jr, Jack & Burnett, Steven, Special Edition Using Linux Fourth Edition, Que Corporation, 1999

Theroux, Jean-Francois, “Firewall/packet filters rules”, 9 Oct 2003 http://www.digitalized.ca/scripts/ruleset.txt

Toxen, Bob , Real World Linux Security, Upper Saddle, NJ: Pearson Education Inc as Prentiss Hall PTR, 2003