OWASP
The Open Web Application Security Project
http://www.owasp.org

SELinux, Apache, and Tomcat – A Securely Implemented Web Application Server

Version 1.3 as of 8/11/06

Author: Russ McRee

russ@holisticinfosec.org

holisticinfosec.org

Contributors: Eric Sheridan, chcon details

eric.sheridan@owasp.org

Change notes v1.3:

Chcon additions to **SELinux configuration post httpd install**

Change notes v1.2:

Updated Fedora Core 4 to CentOS 4.3, Apache's httpd 2.0.52 to 2.0.58, apache-tomcat 5.5.12 to 5.5.17, mod_jk-ap20-1.2.14 to 1.2.15, and mod_security-1.8.7 to modsecurity-apache-1.9.4.

Added mod_evasive.

Updated links.

Future work:
httpd 2.2 and Jboss

Table of Contents

Introduction

As an enterprise strives to grow its online presence, its exposure to risk grows as Web applications are added. Thus, a secure as well as highly available Web application server becomes critical to help enterprises mitigate some of that risk.

Although no implementation offers perfect security, administrators can take a number of steps to dramatically improve the security of their Web app servers.

This paper describes methodology to build a secure Web application server utilizing the benefits of SELinux with a CentOS 4.3 distribution, along with the Apache httpd server and Apache-Tomcat communicating via mod_jk. Additionally, the use of iptables, mod_rewrite, and mod_security will be discussed.

In this paper, I will not cover SELinux or the modules utilized in detail; those topics are worthy of articles in and of themselves. Rather, I will present a server that I believe you can be comfortable with for public facing apps (assuming the apps are also well written).

You may have discovered that the documentation for implementing Apache's httpd and Tomcat servers is vast, but each reference typically encompasses individual services rather than implementing them holistically with security at the forefront. I will strive to describe a start-to-finish process to roll a well-secured Web app server that also offers user authentication and SSL options through the Apache Web server.

I'll make some assumptions for brevity's sake. First, we only have one server, and we're on a tight budget. Second, CentOS installations are straightforward and offer little challenge for most Linux users, so I'll describe only some key points in the install phase to ensure a more secure implementation. Third, I assume the reader has a reasonable understanding of the Apache Web server and its available modules, as well as managing iptables, and working with Apache-Tomcat. Finally, I assume that the basic principles of secure systems administration are also comfortable subject matter.

SELinux is innate in CentOS 4.3 and offers some excellent protections as, according to Wikipedia, "a version of the Linux kernel and utilities, which contains support for mandatory access controls based on the principle of least privilege. Primarily developed by the U.S. National Security Agency (NSA) it was released to the open source development community."

Iptables is also a default offering in CentOS distributions, allowing you to limit network traffic only to those ports and services you deem entirely necessary.

Installation

As you begin your installation from CentOS 4.3 media (assuming a clean install rather than an upgrade), follow the configuration process using default settings or those of your choice until you reach the Installation Type window. It is essential to choose Custom here. Resume defaults or personal settings until you reach the Firewall Configuration window. In this step, be certain to choose Enable firewall (default), check Web Server (HTTP, HTTPS) and Remote Login (SSH), and ensure SELinux is set to Active. You'll work through two more windows then arrive at the Package Group Selection window. Here you'll uncheck all options accept Minimal (very last check box at bottom of list).

The initial intent is to install nothing that is not required except basic OS functionality. You'll soon see, though, that one of the drawbacks of most RedHat-based distributions is that, even when the bare minimum is chosen, you will find a plethora of packages that serve no purpose on a hardened system. Reboot as prompted when the OS installation completes and log back on as root.

Here's where we start to clean house. Using rpm, we'll remove all the clutter that doesn't belong on a public-facing server. You'll note tools for bind, sendmail, printer libraries (cups), and a number of x11 libraries, even though we didn't install them. The gathering in **Listing 1** can be executed in one fell swoop at the command line, ensuring that you maintain a single space between each package name. You may find other unnecessary packages that you want to remove, but experiment carefully and understand the dependencies.

Listing 1 - RPM removal

```
rpm -e xorg-x11-libs xorg-x11-Mesa-libGL libtiff up2date system-config-mouse bind-utils bind-
libs ypbind yp-tools htmlview pinfo ppp rp-pppoe wvdial cups cups-libs redhat-lsb mdadm
portmap nfs-utils irda-utils isdn4k-utils pcmcia-cs NetworkManager pam_smb dos2unix
```

At this point, it is best to turn off the sendmail daemon. This is most easily done by editing /etc/sysconfig/sendmail and setting DAEMON=no. For an extensive methodology review of sendmail security, see:

http://www.deer-un.com/~hal/sysadmin/sendmail.html

Yum (Yellow dog Updater, Modified) will be used for system updates. In a default CentOS installation, the repository it draws from is /etc/yum.repos.d/CentOS-Base.repo, which in turn can be modified to allow fast, geographically based updates.

I prefer yum from an ease of use perspective because it requires only /etc/yum.conf and the /etc/yum.repos.d directory to manage packages. Some may note that rollback functionality does not exist in yum, but all indicators show that rollbacks are obsolesced and are not part of the rpm's future.

Install httpd

At the time of writing this release of *SELinux, Apache, and Tomcat – A Securely Implemented Web Application Server*, the current Apache 2.0 release is 2.0.58. Apache has released 2.2.2 but the 2.2 series indicates differences in modules we're using here, that will require more stringent testing than appropriate for this release. Suffice it to say 2.2 will be tested and described in future releases of this paper.

Installing 2.0.58 from RPMs will require downloading four files, including httpd-2.0.58 itself.

First, execute *yum install postgresql-libs*. This will answer one of three dependencies the httpd RPM will require.

Then, wget the following from an Apache.org mirror (<mirror> indicates the mirror of your choosing):

http://<mirror>/apache/httpd/binaries/rpm/i386/httpd-2.0.58-1.i386.rpm

You'll also need the APR files. Download the 0.9.12 series, as it most closesly matched the CentOS-preferred packages. 1.2.7 is available, but resist temptation until the next release of this paper.

Wget the following:

http://<mirror>/apache/apr/binaries/rpm/i386/apr-0.9.12-1.i386.rpm

http://<mirror>/apache/apr/binaries/rpm/i386/apr-util-0.9.12-1.i386.rpm

As root, cd to your download directory, and enter *rpm –Uvh httpd-2.0.58-1.i386.rpm apr-0.9.12-1.i386.rpm apr-util-0.9.12-1.i386.rpm*.

SELinux configuration post httpd install

NOTE: There is one critical step to execute that will ensure mod_security functionality. Execute */usr/sbin/setsebool –P httpd_disable_trans 1*. Mod_security will fail to load properly and httpd will die without this SELinux boolean setting enabled.

chcon

Administrators may run a situation wherein SELinux complains when Apache attempts to serve content not labeled 'httpd_sys_content_t'.

Consider the following commands, per content directory, if they are relevant to your installation:

htdocs:	chcon -R -t httpd_sys_content_t htdocs
cgi-bin:	chcon -R -t httpd_exec_t cgi-bin
logs:	chcon -R -t httpd_log_t logs

Add Tomcat user

At this point, you should build an account under which to run Tomcat. Running a server like Tomcat under root is certainly unnecessary and not recommended. First, do *groupadd <user>* where <user> is an account name you choose. Second, issue *useradd <user> -g <user> -d /home/<user>*, and third do *passwd <user>*.

Repeat these steps for a general user for administration. Later, I will show how to harden SSH access, which will prevent ssh logon as root. Thus, an additional user will be required to su or sudo root commands.

Add packages

The next step is to build a staging directory for packages that you'll need. As an example, do: mkdir /staging. You'll need packages to complete this platform:

1. Apache-Tomcat 5.5.17 from http://tomcat.apache.org/download-55.cgi

2. mod_jk-ap20-1.2.15-1 from http://www.jpackage.org/rpm.php?id=3345

3. mod_security-1.9.4 from http://www.jackal-net.at/tiki-read_article.php?articleId=22

4. mod_evasive will be covered in full on page 5 in the **mod_evasive** section. A production worthy RPM is not availabe at the time of writing this release.

5. JRE Version 5 Update 6 from http://java.com/en/download/manual.jsp and select the Linux RPM.

You can download these rpms on your admin workstation and copy them to CD or USB drive, transfer them over SSH with gftp or WinSCP, or simply wget them from the server you're building.

Regardless, ensure they're all in your staging directory then cd to it.

Install the modules first: *rpm -Uvh mod_jk-ap20-1.2.15-1jpp.i386.rpm modsecurity-apache-1.9.4-1.EL4.i386.rpm*

Install the JRE: *sh jre-1_5_0_06-linux-i586-rpm.bin*

This will install the JRE in /usr/java.

Install Tomcat

Finally, install Tomcat. Copy Apache-Tomcat-5.5.17.tar.gz from /staging to /usr/share (or a directory of your choosing):

cp apache-tomcat-5.5.12.tar.gz /usr/share

cd /usr/share

tar -zxvf apache-tomcat-5.5.17.tar.gz

rm -f apache-tomcat-5.5.17.tar.gz

Grant permission to <user> (the Tomcat user you created earlier) to all tomcat directories:

chown -R <user>:<user> /usr/share/jakarta-tomcat-5.5.17 then logout.

Log back in as <user> and *vi .bash_profile* to make some key changes to the user's environment variables.

Add *:/usr/java/jre1.5.0_06/bin* to the PATH reference, add *JAVA_HOME=/usr/java/jre1.5.0_06* after the PATH reference, and add *JAVA_HOME* to the export reference after PATH. Log out, then back in as <user> and build a couple of quick convenience scripts.

Next, do mkdir /home/<user>/bin, as it is already referenced in the <user> path, then cd /home/<user>/bin. You'll make tcatup and tcatdown here for quick start and stop of the Tomcat server.

Create tcatup with vi tcatup and enter two lines:

cd /usr/share/apache-tomcat-5.5.17/bin

./startup.sh

Tcatdown will be identical with the exception of ./shutdown.sh. Be sure to chmod a+x to make these scripts executable.

The next step is to clean up the default Tomcat installation. Do:

cd /usr/share/apache-tomcat-5.5.17/conf then rename or delete tomcat-users.xml, as it is not required. Delete manager and host-manager from usr/share/apache-tomcat- 5.5.17/server/webapps. From /usr/share/apache-tomcat-5.5.17/bin, do: rm-rf *.exe and rm-rf *.bat. No need for Windows files on a Linux server.

Finally, from /usr/share/apache-tomcat-5.5.17/webapps, remove all the example directories such as jsp-examples, balancer, tomcatdocs, etc., as well as ROOT.

Configuration

Configure mod_jk-ap20

Mod_jk is next and requires little more that the minimal setup described in Apache's documentation. You can build far more complex configurations with help from a plethora of available documentation.

Listing 2 is taken directly from: http://jakarta.apache.org/tomcat/connectors-doc/howto/quick.html

Listing 2 mod_jk

```
# Define 1 real worker using ajp13
worker.list=worker1
# Set properties for worker1 (ajp13)
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
worker.worker1.lbfactor=50
worker.worker1.cachesize=10
worker.worker1.cache_timeout=600
worker.worker1.socket_keepalive=1
worker.worker1.reclycle_timeout=300
```

Create Listing 2 as workers.properties in /etc/httpd/conf.

httpd.conf including mod_rewrite, mod_security, mod_jk-ap20, and mod_evasive

For a recent implementation similar to what this paper describes, Apache was only used to forward to Tomcat and served no other Web sites. Therefore, in my httpd.conf, I removed the vast majority of Section 2, including all references to DocumentRoot and its associated directory, UserDir, DirectoryIndex, Alias, ScriptAlias, WebDAV, Icons, cgi, AddLanguage, Error (handled by my app), and Proxy. Additionally, I deleted entire LoadModule references for modules I didn't want loaded, such as mod_proxy, mod_userdir, etc. We really only have heavy dependencies on modjk, mod_security, mod_auth, mod_rewrite, and a few others for this implementation. This may all seem a bit draconian, but the intent here is to strip the entire process down only to that which is explicitly required.

mod_rewrite

You can deter TRACE using mod_rewrite as well. I list mod_rewrite configurations before mod_security in my httpd.conf. See Listing 3.

mod_security

We've finally reached my favorite tool in this server setup, namely mod_security. If you want a great read on mod_security, see Shreeraj Shah's article, "Defending Web Services using mod_security", which says: "ModSecurity is an open source intrusion detection and prevention engine for Web applications (or a Web application firewall). Operating as an Apache Web server module or standalone, the purpose of ModSecurity is to increase Web application security, protecting Web applications from known and unknown attacks." The benefit of this module is obvious from its description. Imagine filtering SQL injection or XSS strings right in httpd.conf. I list my mod_security configurations before mod_jk configurations in my httpd.conf. I put the mod_security section before the mod_jk configurations described in Listing 3.

The mod_security section in Listing 3 is compiled of various examples and is by no means comprehensive, nor are these the only options available to you. You can and should experiment with adding SecFilter references for your needs by testing your site before final deployment using nikto from cirt.net or nessus.

mod_jk-ap20

Included in Listing 3's httpd.conf is the required mod_jk section.

I've commented out loading the module here because you should load it with other modules in the modules section of httpd.conf. Note that you should load mod_jk before mod_rewrite. Use JkMount to name the directory for the Web app you've placed in /usr/share/apache-tomcat-5.5.17/webapps. Use JkUnMount to prevent enumeration of files you do not want to be rendered directly. This functionality is only available in mod-jk-1.2.7 or later.

mod_evasive

This liitle gem comes from Jonathan Zdziarski at NuclearElephant.com. The current version is 1.10.1. Jonathon's site describes mod_evasive as "an evasive maneuvers module for Apache to provide evasive action in the event of an HTTP DoS or DDoS attack or brute force attack. It is also designed to be a detection and network management tool, and can be easily configured to talk to ipchains, firewalls, routers, and etcetera. mod_evasive presently reports abuses via email and syslog facilities."

Installation will be slightly tricky in our design, as our server has no compiler installed and no apxs facilities for module compilation.

I performed the following on an Apache server that included the necessary compilation tools including apxs:

tar –zxvf mod_evasive_1.10.1.tar.gz

cd mod_evasive

apxs –i –a –c mod_evasive20.c

This will install mod_evasive20.so in the default Apache modules directory. This may vary for you, as, in my case, I compiled the module on a Debian server, which is distinctly different than a RedHat derivative.

Copy mod_evasive20.c from the server you compiled it on to your new CentOS web app server and place it in /usr/lib/httpd/modules.

Add the following to httpd.conf: *LoadModule evasive20_module modules/mod_evasive20.so*

The mod_evasive section in Listing 3 is default but quite functional.

SELinux configuration post httpd install (repeated)

NOTE: There is one critical step to execute that will ensure mod_security functionality. Execute */usr/sbin/setsebool –P httpd_disable_trans 1*. Mod_security will fail to load properly and httpd will die without this SELinux boolean setting enabled.

Final steps

Restart httpd: */sbin/service httpd restart*

My httpd.conf looks largely as found below. It is a minimalist configuration attempting to refer only to the bare necessities.

Listing 3 - httpd.conf

```
### Section 1: Global Environment
ServerTokens OS
#
ServerRoot "/etc/httpd"
#
PidFile run/httpd.pid
#
Timeout 120
#
KeepAlive Off
#
MaxKeepAliveRequests 100
#
KeepAliveTimeout 15
##
## Server-Pool Size Regulation (MPM specific)
##
<IfModule prefork.c>
StartServers       8
MinSpareServers    5
MaxSpareServers   20
ServerLimit      256
MaxClients       256
MaxRequestsPerChild  4000
</IfModule>
# worker MPM
<IfModule worker.c>
StartServers         2
MaxClients         150
MinSpareThreads     25
MaxSpareThreads     75
ThreadsPerChild     25
MaxRequestsPerChild  0
</IfModule>
#
#Listen 12.34.56.78:80
Listen 80
#modules
LoadModule access_module modules/mod_access.so
LoadModule auth_module modules/mod_auth.so
LoadModule log_config_module modules/mod_log_config.so
#LoadModule logio_module modules/mod_logio.so
LoadModule mime_magic_module modules/mod_mime_magic.so
#LoadModule expires_module modules/mod_expires.so
LoadModule deflate_module modules/mod_deflate.so
LoadModule headers_module modules/mod_headers.so
LoadModule usertrack_module modules/mod_usertrack.so
LoadModule setenvif_module modules/mod_setenvif.so
LoadModule mime_module modules/mod_mime.so
#LoadModule info_module modules/mod_info.so
LoadModule jk_module modules/mod_jk.so
LoadModule rewrite_module modules/mod_rewrite.so
LoadModule security_module  modules/mod_security.so
LoadModule evasive20_module modules/mod_evasive20.so
LoadModule cache_module modules/mod_cache.so
LoadModule file_cache_module modules/mod_file_cache.so
LoadModule mem_cache_module modules/mod_mem_cache.so
#
# Load config files from the config directory "/etc/httpd/conf.d".
#
Include conf.d/*.conf
#
# User/Group:
User apache
Group apache
### Section 2: 'Main' server configuration
# ServerAdmin:
ServerAdmin <your email address>
#
ServerName <your server name>
# UseCanonicalName
UseCanonicalName Off
#
DocumentRoot "/var/www/html"
AccessFileName .htaccess
#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
#<Files ~ "^\.ht">
#    Order allow,deny
#    Deny from all
#</Files>
#
# TypesConfig describes where the mime.types file (or equivalent) is
# to be found.
#
TypesConfig /etc/mime.types
#
DefaultType text/plain
#
# The mod_mime_magic module allows the server to use various hints from the
# contents of the file itself to determine its type.  The MIMEMagicFile
# directive tells the module where the hint definitions are located.
#
<IfModule mod_mime_magic.c>
#   MIMEMagicFile /usr/share/magic.mime
    MIMEMagicFile conf/magic
</IfModule>
#
# HostnameLookups:
HostnameLookups On
# ErrorLog:
ErrorLog logs/error_log
#
# LogLevel:
LogLevel warn
#
# The following directives define some format nicknames for use with
# a CustomLog directive (see below).
#
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
# For a single logfile with access, agent, and referer information
# (Combined Logfile Format), use the following directive:
#
CustomLog logs/access_log combined
#
ServerSignature Off
#
AddDefaultCharset UTF-8
# If the AddEncoding directives above are commented-out, then you
# probably should define those extensions to indicate media types:
#
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
# The following directives modify normal HTTP response behavior to
# handle known problems with browser implementations.
#
#
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0
#
########################
#  mod_rewrite config  #
########################
RewriteEngine on
# Optional debug directives
RewriteLog          logs/mod_rewrite.log
RewriteLogLevel     1
RewriteCond %{REQUEST_METHOD} ^TRACE
RewriteRule .* - [F]
#
#########################
#  Mod_security config  #
#########################
<IfModule mod_security.c>
# Turn the filtering engine On or Off
SecFilterEngine On
SecFilterDefaultAction "deny,log,status:500"
SecFilterScanPOST On
# Make sure that URL encoding is valid
SecFilterCheckURLEncoding On
SecFilterCheckCookieFormat On
# Unicode encoding check
SecFilterCheckUnicodeEncoding On
# Only allow bytes from this range
SecFilterForceByteRange 1 255
# Only log suspicious requests
SecAuditEngine RelevantOnly
# The name of the audit log file
```

```
#SecAuditLog logs/audit_log
SecFilterSelective REQUEST_METHOD "!^GET$" chain
SecFilterSelective HTTP_Content-Type "!(^$|^application/x-www-form-
urlencoded$|^multipart/form-data)"
SecFilterSelective REQUEST_METHOD "^POST$" chain
SecFilterSelective HTTP_Content-Length "^$"
SecFilterSelective HTTP_Transfer-Encoding "!^$"
# Prevent path traversal (..) attacks
SecFilter "\.\./"
# Weaker XSS protection but allows common HTML tags
SecFilter "<( |\n)*script"
SecFilter "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
# Prevent XSS atacks (HTML/Javascript injection)
SecFilter "<(.|\n)+>"
SecFilter "/\"><img\x20src=\"javascript:alert(document.domain)\">"
# Crude filters to prevent SQL injection attacks
SecFilter "delete[[:space:]]+from"
SecFilter "insert[[:space:]]+into"
SecFilter "select.+from"
# Require HTTP_USER_AGENT and HTTP_HOST headers
SecFilterSelective "HTTP_USER_AGENT|HTTP_HOST" "^$"
</IfModule>
#
######################
# Mod_evasive config #
######################
#
#LoadModule evasive20_module modules/mod_evasive20.so #Loaded in modules section
<IfModule mod_evasive20.c>
   DOSHashTableSize    3097
   DOSPageCount        2
   DOSSiteCount        50
   DOSPageInterval     1
   DOSSiteInterval     1
   DOSBlockingPeriod   10
</IfModule>
#Optionally you can also add the following directives:
#    DOSEmailNotify you@yourdomain.com
#    DOSSystemCommand    "su - someuser -c '/sbin/... %s ...'"
    DOSLogDir                     "/var/log/mod_evasive"
#
########################
# Mod_jk Configuration #
########################
# mod_jk configuration
#LoadModule    jk_module  modules/mod_jk.so #Loaded in modules section
# Where to find workers.properties
JkWorkersFile /etc/httpd/conf/workers.properties
# Where to put jk logs
JkLogFile     /var/log/httpd/mod_jk.log
# Set the jk log level [debug/error/info]
JkLogLevel    info
# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y] "
# JkOptions indicate to send SSL KEY SIZE,
JkOptions     +ForwardKeySize +ForwardURICompat -ForwardDirectories
# JkRequestLogFormat set the request format
```

```
JkRequestLogFormat     "%w %V %T"
# Send everything for context /<your app> to worker named worker1 (ajp13)
JkMount  /<your app> worker1
JkMount  /<your app>/* worker1
JkUnMount /<your app>/*.jsp worker1
JkUnMount /<your app>/*.js worker1
JkUnMount /<your app>/*.xml worker1
JkUnMount /<your app>/*.xslt worker1
JkUnMount /<your app>/*.html worker1
JkUnMount /<your app>/*.cgi worker1
JkUnMount /<your app>/*.txt worker1
#
#<Location /<your app>>
#AuthType Basic
#AuthName "Authorized Users Only"
#AuthUserFile /usr/share/access/.authenticated
#require valid-user
#</Location>
#
### Section 3: Virtual Hosts
#
# VirtualHost
#
```

User Authentication

It seems that mod-jk with Apache and Tomcat requires utilizing Apache-based user authentication. There are additionally confusing search references that indicate .htaccess (mod_auth) will not work with mod_jk. The following works quite successfully, however. As root, do:

mkdir /usr/share/access

Following the htpasswd command syntax, issue:

htpasswd -c /usr/share/access/.authenticated <username> to create .authenticated.

To add users after the initial file creation, issue:

htpasswd /usr/share/access/.authenticated <username2>

You can build on this with digest authentication or LDAP, but for our purposes we'll keep it simple. See the Authenticate Users entry right before Section 3 in httpd.conf (see Listing 3).

Restart the Web server with service httpd restart and start a new browser session. You should be prompted for a username and password. Ensure that principal names are being successfully passed between Apache and Tomcat via mod_jk. For mod_jk, tomcatAuthentication="false" should be present in the <Ajp13Connector> configuration element in /usr/share/apachetomcat-5.5.12/conf/server.xml to ensure that the user's identity is passed from Apache to the servlet environment.

Harden SSH and Modify iptables

This is ideally performed last, after configuration is complete, because you might be building via ssh and initially want to establish a root shell without using su or sudo. You should also make these changes logged into the server at the local console, rather than remotely as you will need to modify your iptables configuration. <your port> represents an unused port number of your choosing. Assuming you allow only port 80 or 443 to your server in your DMZ, you'll likely only use SSH over the port you assign from your internal network.

Edit /etc/ssh/sshd-conf:

1. Uncomment Protocol 2 (accepts ssh2 only).

2. Uncomment Port 22 and change 22 to <your port> (modifies the ssh port for obfuscation purposes).

3. Uncomment LoginGraceTime.

4. Uncomment PermitRootLogin and change yes to no (prevents attempts to remotely brute-force the server as root). If root is needed while administering via ssh, utilize su or sudo.

5. Uncomment StrictModes.

6. Uncomment MaxAuthTries and change 6 to 3 (limits logon attempts to 3).

Iptables

Iptables, as configured automatically during the OS setup, is set to allow traffic over port 80 and the default SSH port 22.

First, *cd /etc/sysconfig/iptables* and *vi iptables*.

Be sure to delete the following as they are undesirable entries typical to the default installation:

```
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
```

Iptables must be modified to allow <your port> rather than 22:

Edit the line allowing 22 to <your port>; save and exit. Then issue

service iptables restart.

Listing 4 offers an example.

Listing 4 – iptables

```
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type echo-request -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type echo-reply -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport <your port> -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Troubleshooting

If you're having trouble pulling up your Web app through Apache after this, consider the following:

• SELinux policies can factor in the server's availability. As root, turn off SELinux policy enforcement by issuing: setenforce 0.

• setenforce 1 will restart SELinux enforcement. The status of SELinux enforcement can be checked by issuing: sestatus | grep -i mode.

• Don't forget to issue */usr/sbin/setsebool –P httpd_disable_trans 1* if httpd is hanging on mod_security.

• Iptables might factor as well. To be sure your iptables rules aren't working against you, issue: service iptables stop.

• Check /etc/sysconfig/iptables to see what may be the cause. If it seems as if file permissions issues are causing problems, issue the following:

chmod -R 755 /usr/share/apache-tomcat-5.5.17

Reduce these rights after you've solved the problem and retest.

Also issue:

chown -R tomcat:tomcat /usr/share/apache-tomcat-5.5.17

Notes on SSL

As you've probably spent a good deal of time with Apache, you are likely capable of configuring SSL. However, if not, you can do so by following Apache's documents or those at: http://www.linux-sxs.org/internet_serving/apache2.html

Summary

This paper hasn't really blazed any new trails; I've simply merged those blazed previously. I have shown, however, how to build a server that provides some sense of comfort in a savage environment.

In utilizing SELinux, deploying key Apache modules, activating iptables, removing unnecessary fluff from the OS and our configurations, and applying some basic good sense, I've presented steps that can be taken to provide Web applications with a safe home. It is my hope that your Web apps live a long and peaceful life. Just remember to check the code, too, and test, test, test.

Acknowledgments

Thanks to Ken Van Eyk for his extensive Java knowledge and for driving the project that led to this paper.

Thanks to the SMC team and my family for their endless support.

Resources

Apache HTTP server — http://httpd.apache.org/
Apache Portable Runtime – http://apr.apache.org/
CentOS — http://centos.org/
Modsecurity — http://modsecurity.org
SELinux — http://www.nsa.gov/selinux/index.cfm
Shah, Shreeraj. "Defending Web Services using mod_security" —
http://www.infosecwriters.com/text_resources/pdf/Defending-web-services.pdf
Tomcat — http://tomcat.apache.org/