Global Information Assurance Certification Paper

Secure Red Hat 9 to Run a Trouble Ticket System in Chroot Environment

GIAC Certified UNIX Security Administrator (GCUX)
Practical Assignment Version 1.9
Securing Unix Step by Step

C. Cliff Liu

03/15/2004

Introduction

This paper describes the author's step-by-step process of building a trouble ticket system in a "chroot" environment on a RedHat 9 desktop. Please note that the information contained herein is not an official document and there are no warranties to the accuracy of the data.

The trouble ticket system, referenced in this paper as "Bug Tracking Server", is a freeware Bugzilla from www.mozilla.org. The Bug Tracking Server uses a web server (Apache) to provide user interfaces. It also uses software packages such as the MySQL database server, Perl programming language, and Perl modules. The table below lists version (the most recent at this time) and source of each of the software. Note that HTTP, instead of HTTPS (HTTP over SSL), is used for this project.

Software	Version	Source
Bugzilla	2.16.4	www.bugzilla.org
Apache	2.0.48	www.apache.org
MySQL	3.23.58	RedHat 9 release
Perl	5.8.0	RedHat 9 release
Perl modules	Latest as of January,2004	CPAN

Besides the Bug Tracking Server, the only other service provided by this host is SSH. The SSH server is OpenSSH (version 3.5p1-11) from RedHat 9 release, newer version can also be downloaded from www.openssh.org. The SSH server provides remote administration for system administrators.

The platform used for the Bug Tracking Server is a Dell Precision 330 desktop running Linux Red Hat 9 operating system. The Dell desktop has a 1.4 GHZ Intel Pentium 4 CPU with 384 MB memory and an 18 GB disk. The Bug Tracking Server, when running in the "chroot environment", needs 82 MB of disk space initially, and will grow to over 100 MB as its database becomes bigger.

This server is connected to the company network behind the company firewall, which blocks the HTTP port (port 80) and many other privileged ports (TCP/UDP port number below 1024). Access to the Bug Tracking Server is restricted to the trusted client hosts, or the authorized users on intranet as described later during system hardening.

Risk analysis

The Bug Tracking Server enforces a strict access control to the Apache web server interfaces via Apache's configuration file so that only internal people may access. However any web server on the Internet (or even on intranet) is susceptible to security exploits such as buffer overflow problems in the web server software, or the security holes in CGI scripts run by the web server. Some older versions of MySQL database server also had buffer overflow issues that could cause denial of service attacks, or even gaining root privilege by a local or remote user. Much of these risks can be avoided by keeping software up-to-date as described later in section "Configure up2date service" and "keep system up-to-date". However to mitigate the previously undiscovered vulnerabilities, it's decided that the Bug Tracking Server shall be run inside a chroot environment.

Chroot environment, also known as "chroot jail", is meant to provide another layer of security when running high-risk programs such as the Apache web server. When a program runs in a chroot environment it is running relative to a new root directory, instead of the typical Unix root directory (/). This program can only see the files under this new root directory tree that contains only a very small set of files that are needed by programs running in chroot environment.

There are many benefits of running programs in a chroot environment, one example is that a malicious CGI script will not able to email important files such as the encrypted password file (/etc/shadow) to the potential intruder because theyt are not in the new root directory tree. Another example is that local and remote privilege elevations will not be able to get outside of the chroot jail unless the chroot environment contains setuid-root programs (the Bug Tracking Server doesn't contain any setuid-root programs) that have security holes. As mentioned earlier a chroot environment is meant to provide another layer of protection, while it may not deter the most determined attackers, it makes such attacks much more difficult.

There are two TCP ports (HTTP and SSH) that are open to the network on this host. HTTP port is considered secure because it's blocked by company firewall, has strict intranet access control via Apache configuration, and the service binded to it is running in a chroot environment. The SSH port also has a strict access control enforced by TCPWrapper. As will be shown later in local firewall configuration, there are a few other TCP and UDP ports that are open to only specific remote hosts for the necessary services such as DNS name service, Networker backup service and NTP service. These hosts are secure servers so the risk of these ports being open is considered minimum.

Building Bug Tracking Server step by step

Building the Bug Tracking Server consists of three major tasks. First, a Red Hat 9 OS (operating system) is installed and hardened, then a complete Bug Tracking Server is installed within the original file systems environment and tested, finally all Bug Tracking Server's processes, system resources and data files needed are copied into a chroot environment. If you already have a secured operating system in place you can skip the first task, or even the first two tasks if you simply want to know how to get Bugzilla to work in a chroot environment.

Install and secure Red Hat 9 operating system

Although efforts have been made to include as much information as possible in this section, it should be noted that OS installation and hardening is often dictated by site policy, and the applications running on the host. Below are the steps that were taken to install and secure this server.

Power up the system and immediately insert Red Hat 9 CD 1 into the cdrom drive.

When Red Hat Install screen comes up, press <enter> to select graphical mode.

Click next to select English language and U.S. English keyboard (both are default).

Select mouse model based on the type of mouse used. This system uses Microsoft IntelliMouse (PS/2) mouse.

Select Perform a new Red Hat Linux installation.

Select Custom Installation Type.

Select Manually partition with Disk Druid.

Partitioning:

> There are many different opinions regarding how systems should be partitioned. Perhaps the best strategy is to partition the system in a way that will better mitigate the risks that may result from the applications the system is running while trying to maximize the use of available disk space. In general on a Unix server it's a good idea to have a separate /tmp partition to avoid denial of service to occur when local users (or remote

attackers) fill up the root file system. This is not done here for the following reasons: 1. This server does not have general user accounts. 2. This server does not have a large disk so the maximization of disk space is desired. 3. The Bug Tracking Server is running in choot environment and can not access the regular /tmp file system.

However a separate /usr partition is made so different security options can be applied on this file system later in the hardening process. Also there is a /home partition for system administrator's home directory, downloaded software updates and the chroot directory tree.

Swap = 800 MB
/boot = 75 MB
/ = 500 MB
/usr = 4 GB
(Red Hat and the subsequent applications installation consume quite a bit of /usr space so at least 4 GB is recommended)
/home = remaining disk space

Boot Loader Configuration: Use the default GRUB boot loader, which is easier to maintain than LILO.

Network Configuration: click Edit, unselect Configure using DHCP, enter IP Address for this server, Netmask, Gateway, and Primary/Secondary DNS servers.

Firewall Configuration: select High and Customize. Disable Trusted devices. The firewall will be customized later to meet the needs of this host.

Additional Language Support and Time Zone Selection: set as appropriate.

Set root password.

Authentication Configuration: use default selection – enable MD5 passwords and shadow passwords.

Package Group Selection:

Package selections are based on individual needs and preferences so it may vary with each individual installation, however the rule of thumb is that if a given package is not needed, don't install it. If after the OS installation, additional packages are needed they can be installed from RedHat CDs or various web sites (for example, www.rpmfind.net). Please note that in the list below, packages for desktop environment, office productivity, graphics and sound, etc. should not be selected if the host is a server only. They are

selected here because the system administrator will log onto it locally for various maintenance tasks.

Desktops: X Window System, GNONE Desktop Environment.
Applications: Editor (Xemacs and vim), Graphical Internet, Text-based Internet (pine), Office/Productivity (openoffice, xpdf), Sound and Video, Authoring and Publishing, Graphics.
Servers: only select SQL Database Server (mysql-server), Apache web server will be built later.
Development: Development Tools, Kernel Development.
System: Administration Tools, System Tools, Printing Support (LPRng, unselect cups). Note we prefer LPRng to Cups, but it's a personal choice.
Miscellaneous: none selected.
Check Select individual packages box to view and add or delete package. Add perl-TimeDate package in CPAN.

Click Next to start installation.

During the installation the Graphical Interface is automatically configured.

Click Exit to restart system.

Continue setup after system restarts, set up user account, set Date and Time or enable Network Time Protocol and enter server name. (Sound card is automatically detected during the restart.)

Now that the basic OS installation is complete. Before hardening the system, it's recommended to update software and patches first. This is to prevent the configuration that will be modified during hardening to be overwritten by the update process. Please note that RedHat's automatic packages update program "up2date" can be configured so that no packages that would change configuration data are automatically installed. However in general it's a good idea that we check the integrity of the configuration after each regular patch updates. This checking can be accomplished by reviewing the tripwire report to be described later. To manually install patches perform the following steps:

1. Since firewall has been set to highest security during OS installation, issue the following commands to open FTP source port 20 and 21 to allow updates.redhat.com.

   ```
   iptables -I INPUT -p tcp -m tcp ! --syn -s updates.redhat.com --sport 21 --dport 1024:65535 -j ACCEPT
   iptables -I INPUT -p tcp -m tcp -s updates.redhat.com --sport 20 --dport 1024:65535 -j ACCEPT
   ```

   Note the above commands do not affect the permanent configuration.

2. Retrieve all updates for RedHat 9:

```
mkdir /home/patches
cd /home/patches
wget ftp://updates.redhat.com/9/en/os/noarch/\*.rpm
wget ftp://updates.redhat.com/9/en/os/`uname -m`/\*.rpm
wget ftp://updates.redhat.com/9/en/os/i386/\*.rpm
```

3. Install patches:

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

However cares should be taken if there are kernel updates (kernel*) among the downloaded updates. These kernel updates should be moved to a subdirectory (eg. ./kernel), cd to the subdirectory and use 'rpm –i' to install it. This will keep the old versions (-F option updates if older version exists and will remove older version). There are different kernel types such as Kernel-smp… is for multi-processor systems, kernel-bigmem is for systems with large amount of memory, etc. Use command 'rpm –qa |grep kernel' to find out what type of kernel was installed on this host during OS installation.

Reboot to make sure the system comes up and proceed with system hardening. Note that if the system can't boot with the new kernel, the old kernel can be selected to boot at the GRUB menu.

Network security:

To view the current network security settings we can go to the virtual file system /proc/sys/net/ipv4 to see them. For example, RedHat disables IP forwarding by default, so 'cat ip_forward' should return 0.

To increase overall network security, edit /etc/sysctl.conf to match the following kernel option settings. Note that some of the settings listed below might already been set as default by RedHat, they are nevertheless listed here for information.

```
#Disable IP Forwarding.  This setting should come before other setting since
#changing it will reset reset other parameters to their default values
net.ipv4.ip_forward = 0

#Disable IP Source Routing which may allow attackers to bypass normal
#network router and firewalls
net.ipv4.conf.all.accept_source_route = 0

#Enable SYN flood protection. Setting a limit to prevent denial-of-service
#attack on TCP connections
net.ipv4.tcp_max_syn_backlog = 4096

#Enable IP Spoofing protection which rejects incoming packets if their source
```

```
#address doesn't match the network interface they are arriving on. Note this
#only protects against spoofed internal address.
net.ipv4.conf.all.rp_filter = 1

#Enable TCP SYN Flood protection
net.ipv4.tcp_syncookies = 1

#Log packets that have source addresses with no known route to the kernel
#log
net.ipv4.conf.all.log_martians =1

#Disable the ability to send ICMP Redirects
net.ipv4.conf.all.send_redirects = 0

#Disable acceptance of ICMP Redirect messages
net.ipv4.conf.all.accept_redirects = 0

#Disable ICMP Redirect acceptance
net.ipv4.conf.default.accept_redirects = 0
```

If any changes are made, issue '/etc/init.d/network restart' as root to activate them .

Configure OpenSSH:

OpenSSH and TCP Wrapper (providing access controls for OpenSSH) were installed during OS installation from RedHat CDs. TCPWrapper searches two files, first /etc/hosts.allow and then /etc/hosts.deny, but stops at the first match. Access will be granted when a match is found in /etc/hosts.allow, otherwise access will be denied when a match is found in /etc/hosts.deny, and finally if no matches is found access will be granted. On this site we use only /etc/hosts.allow in which hosts that are allowed to this system are listed, followed by a global deny statement. See below for an example (refer to hosts_access manpage for the format of this file):

```
sshd: 69.138.xxx.xxx
sshd: …
sendmail: localhost.localdomain
ALL: ALL: DENY
```

Note that the sendmail statement in the above example allows the host to send email.

Verify that the SSH server configuration file (/etc/ssh/sshd_config) has the following settings (with comments following each statement).

```
Protocol            2                   #Use only protocol 2
SyslogFacility AUTHPRIV                 #Log ssh security messages in syslog
PermitRootLogin no                      #Root user can not login via ssh
PermitEmptyPasswords no                 #Disallow login to empty password
```

```
                                    #   account
Banner /etc/issue                   #Display warning banner before login
IgnoreRhosts yes                    #Disable use of .rhost and .shost
RhostsAuthentication no             #Disable rhost authentication
RhostsRSAAuthentication no          #Protocol 1 option, same as protocol 2
                                    #option HostbasedAuthentication below
HostbasedAuthentication no          #Disable host public key authentication
```

Note that Protocol 1 have had many known security holes, even though they have been fixed it's common nowadays to only use protocol 2 (the default protocol setting "Protocol 2,1" falls back to protocol 1 if protocol 1 client connects the server).

The "Banner" option will cause a warning banner containing legal notification to be displayed before a user logs into the system. This legal notification may become useful if a civil or criminal prosecution will be pursued if a hacker breaks into this system. To create such warning banner, enter the legal notification in /etc/issue. Below is an example of /etc/issue.

```
NON PUBLIC RESOURCE WITH EXTERNAL CONNECTIONS

This resource is for authorized use only.

If not authorized to access this resource, disconnect now.
Unauthorized use of, or access to this resource may subject you to
        disciplinary action or criminal prosecution.

By accessing and using this resource, you are consenting to
        monitoring, keystroke recording or auditing.
```

Restart sshd daemon (/etc/init.d/sshd restart) to activate the configuration changes.

Sendmail Configuration:

Sendmail has been installed during RedHat 9 OS installation. Sendmail on this host runs as a client to only listen on the local loopback address 127.0.0.1. It does not accept emails from the external network. This is achieved by the line "DnMAILER-DAEMON" in /etc/mail/sendmail.cf. If it's not there, modify /etc/mail/sendmail.mc to ensure that the line "DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl" is not commented out. Then run 'make –C /etc/mail' to recreate sendmail.cf, and '/etc/init.d/sendmail restart' to activate the change.

Modify Firewall Configuration:

RedHat 9's default firewall is iptables. Iptables contain rules to filter input, forward and output packets. Since the high security firewall setting (selected

during the OS installation) blocks most of the ports, the firewall configuration needs to be modified to open some ports that the host uses to communicate with the outside world. As seen below in the firewall configuration file (/etc/sysconfig/iptables), both INPUT and FORWARD chains are filtered by the same rule set (identified as RH-Lokkit-0-50-INPUT), while the OUTPUT chain is not filtered. All incoming traffics are blocked except for ones that are specifically allowed in this configuration. Port 22 (SSH port) and 80 (HTTP port) on this host are open for TCP connections. TCP packets from port 53 (DNS port) of the two name servers are allowed to enter this host. Similarly, the TCP packets for LEGATO Networker and UDP packets for NTP daemon are allowed from specific servers. More information about iptables can be found in man page (man iptables), or the reference (6) at the end of this paper.

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Lokkit-0-50-INPUT - [0:0]
-A INPUT -j RH-Lokkit-0-50-INPUT
-A FORWARD -j RH-Lokkit-0-50-INPUT
# allow SSHD
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT
# allow HTTP
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp --dport 80 --syn -j ACCEPT
# allow LEGATO Networker traffic
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp -s xxx.xxx.xxx.xxx --dport 7937 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp -s xxx.xxx.xxx.xxx --dport 7938 --syn -j ACCEPT
# unlimited lo traffic
-A RH-Lokkit-0-50-INPUT -i lo -j ACCEPT
# allow our DNS servers
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s xxx.xxx.xxx.xxx --sport 53 -d 0/0 -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s xxx.xxx.xxx.xxx --sport 53 -d 0/0 -j ACCEPT
# allow NTP servers
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s xxx.xxx.xxx.xxx --sport 123 -d 0/0 --dport
1023:65535 -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s xxx.xxx.xxx.xxx --sport 123 -d 0/0 --dport
1023:65535 -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s xxx.xxx.xxx.xxx --sport 123 -d 0/0 --dport
1023:65535 -j ACCEPT
# Reject anything else
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp --syn -j REJECT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -j REJECT
COMMIT
```

Restart iptables daemon to reflect the changes: /etc/init.d/iptables restart

Restrict root login to the system from local console:

Other than logging in from the local console, the only other way of logging into this system is remotely via ssh. To disable root login via ssh add "PermiRootLogin no" to sshd_config and restart sshd as described above.

Require root password when entering "Single User Mode":

Linux provides a "Single User Mode" that allows someone to boot into it and log in physically at console to perform system maintenance. By default Linux doesn't require root password to log into single user mode. Even though this system is in a room that requires key to enter, it's still necessary to protect single user mode with a root password. To do this, add "~~:S:wait:/sbin/sulogin" to /etc/inittab, then run '/sbin/init q' to activate the change.

Some user accounts and groups are not needed for this server to function. It's best to keep the /etc/passwd file as small as possible so it's easier to detect unauthorized additions. To be on the safe side, before removing these accounts make a copy of /etc/passwd and /etc/group in case any account needs to be restored later. Run this command to remove user "uucp" "operator" "games" "mail" "news" "ftp" and "gopher" from /etc/passwd:

```
for user in uucp operator games gopher mail news ftp;
    do /usr/sbin/userdel $user;
done
```

Run this command to remove group "dip" "gopher" "games" "uucp" "mail" "ftp" and "news" from /etc/group:

```
for group in dip gopher games uucp mail news ftp;
    do /usr/sbin/groupdel $group;
done
```

Some other user accounts that are used by system service or daemon do not require interactive login. For example, user lp is needed for print daemon (lpd) to run. Run this script to lock accounts "bin", "daemon", "adm", "sync", "lp" and "nobody".

```
for user in bin daemon adm sync lp nobody;
    do /usr/sbin/usermod –L –s /dev/null $user;
done
```

Verify that no accounts in /etc/passwd have an empty password.

Check to ensure that the second field (fields are separated by ":") of /etc/shadow is non-blank.

Disable unnecessary services:

To reduce the chances of being attacked, any service that is not needed by this host should be stopped, and configured not to start at boot time. First use the command 'chkconfig --list' to list all system services. Then check each of them (read its man page or startup script at /etc/init.d) to determine if the service is needed. If a service is not needed for the server use the following commands to stop it and turn it off.

```
/etc/init.d/<service> stop
/sbin/checkconfig –level 0123456 <service> off
```

The software package for a service can also be removed completely by 'rpm –e <package>' so it will not get new updat.

Some of the services will not be started even if they are turned on because of missing configuration file (one example is /etc/sysconfig/isdncard for isdn). They should be turned off anyway to avoid confusion. Below is a list of services that are turned off on this server:

saslauthd: SASL authentication server that handles plaintext authentication requests on behalf of cyrus-sasl library.
Irda: Infrared Data Association, a starndard for wireless infrared communication between devices.
pcmcia: PCMCIA support is usually to support things like Ethernet and modems in laptops.
Nscd: Name switch cache daemon, used if naming services like NI S, NIS+, LDAP, or hesiod is used.
Isdn: This host doesn't have ISDN (Integrated Services Digital Network) card.
portmap: portmapper is only needed for NFS and NIS servers.
rhnsd: Red Hat Network Daemon that periodically connects to Red Hat Network servers to check for updates. This host uses a local up2date server so it doesn't need this
nfs, nfslock: NFS services that are need only if this host needs to mount NFS file systems. This host does not need to mount NFS file systems.
winbind: This host does not run Samba.
canna: Canna Japanese Conversion Engine.
snmpd, snmptrapd: Simple Network Management Protocol (SNMP) Daemon is used to monitor hosts connected to network. This host doesn't provide to this function.
postgresql: PostgreSQL backend daemon that handles all database requests. This host uses MySQL database server.
lisa: It provides something like a "network neighbourhood" that is not use on this host.

All xinetd services are turned off as shown from the output of 'chkconfig --list'.

```
xinetd based services:
```

```
chargen-udp:    off
rsync:  off
chargen:        off
daytime-udp:    off
daytime:        off
echo-udp:       off
echo:   off
services:       off
servers:        off
time-udp:       off
time:   off
cups-lpd:       off
sgi_fam:        off
ktalk:  off
```

Configure up2date service:

To mitigate security risks it's most important to keep the system up to date with software updates and security patches. Red Hat provides the automatic update service (up2date) that runs rhnsd daemon every 120 minutes to check for the updates. Red Hat however only provides it free for a limited demo period. If you do not plan to buy an extended license you can build an open-source implementation of a local up2date server. The building of a local up2date server is beyond the scope of this paper, but interested readers can go to http://current.tigris.org for software download and installation information. If a local up2date server is used the rhnsd daemon can be turned off as described in "Disable unnecessary services " above.

Below are some of the directives and comments in the configuration file (/etc/sysconfig/rhn/up2date) that may require further explanation.

```
retrieveOnly[comment]=Retrieve packages only
retrieveOnly=1

storageDir[comment]= Where to store packages and other data when they are retrieved
storageDir=/var/spool/up2date

sslCACert[comment]=The CA cert used to verify the ssl server
sslCACert=/usr/share/rhn/LOCAL-CA-CERT

serverURL[comment]=Remote server URL
serverURL=https://local.up2date.server/XMLRPC

noReplaceConfig[comment]=When selected, no packages that would change configuration
data are automatically installed
noReplaceConfig=1
```

The above configuration tells up2date to only download packages (retriveOnly=1) without automatically installing them. This way each new package can be tested on a test host before installation. The downloaded

packages will be in /var/spool/up2date defined by “storageDir”. “SslCACert” (copy this file from the local up2date server) and “severURL” will only need to be changed if a local up2date server is used. “noReplaceConfig” (if retrieveOnly is not set to 1) tells up2date not to automatically install packages that will change configuration.

We can then register this host to the up2date server:

```
up2date  --register
```

Then set up a cron job to run up2date nightly and send an email to administrator containing a list of packages that have been downloaded:

```
/usr/sbin/up2date --nox –u -d >> /var/log/current-up2date
/bin/mail -s "Up2date: `hostname`" cliff < /var/log/current-up2date
```

--nox: do not attempt to display the gui.
-u : Completely update the system.
-d : Download only, do not install.

Install Networker client software:

A reliable backup is necessary to ensure the system’s availability. For a secure server it’s preferred to use local storage devices such as a tape device to do the backup. This avoids system data travel on network unencrypted when a networked backup scheme is used. However due to the lack of local storage device, this host uses a Legato Networker server to do the backup. The Networker server resides on the intranet within the company firewall and has been securely configured.

The Networker server only allows root user to make any configuration changes. It’s configured by default that a client can only recover its own data so that if hackers broke into one client they can’t gain access to the data on any other clients. Ideally the Networker server will be set up to do backup via the SSH tunnel so data traveling on the network will be encrypted. Below are steps for installing and starting the Networker client software on this host.

```
rpm -i  lgtoclnt-6.1.3-1.i386.rpm lgtoman-6.1.3-1.i386.rpm
echo “Networker server host name” > /nsr/res/servers
/etc/init.d/networker start
```

Verify that the Networker daemon is running by ‘ps –ef’ command.

```
root      841     1  0 Mar04 ?        00:00:00 /usr/sbin/nsrexecd
root      867   841  0 Mar04 ?        00:00:00 /usr/sbin/nsrexecd
```

To add this client to the server, run ‘nwadmin’ GUI on the server, get client/setup window, create this client and select backup schedule (full backup once every four weeks, incremental back in the meantime).

Install Tripwire:

Tripwire checks the integrity of file systems by alerting administrators when critical files on the system have been changed. The free Linux rpm package can be downloaded from searching at rpmfind.net. After the rpm package (tripwire-2.3.1-17 is used on this host) is installed the setup instruction can be found in /usr/share/doc/tripwire-2.3.1/quickstart.txt. Below are steps taken on this host to setup tripwire.

The Tripwire policy file (/etc/tripwire/twpol.txt) describes what system objects are to be monitored by Tripwire. The comments in this file and the Tripwire Policy Guide (/usr/share/doc/tripwire-2.3.1/policyguide.txt) provide information on how to tailor this file. Because the Bug Tacking Server is in a chroot environment we need to add files under the chroot directory tree to the appropriate places in the policy file. For example, add /home/chroot/bash to the list of files for rulename “Shell Binaries", and /home/chroot/usr/local/apache2/conf/httpd.conf to rulename “Critical configuration files", and so on. The default configuration file (/etc/tripwire/ twcfg.txt) can be used without modification.

Now run the configuration script to create binary tripwire policy and configuration files:

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

(The script will prompt you to enter “site keyfile passphrase” and “local keyfile passphrase” of your choice that will be used to sign these two binary files.)

Run the command to initialize tripwire database (/var/lib/tripwire/ <hostname>.twd):

```
/usr/bin/tripwire --init
```

While the default policy file was designed for Linux, it might contain files that do not exist on a RedHat 9, or files not installed on this host. As a result, this database initialization script will most likely generate warnings about certain files can’t be found. The text policy file (twpol.txt) should be modified to comment out those files according to these warnings. Please note that after the text policy file is changed we need to recreate the binary policy file (/etc/tripwire/tw.pol) by running ‘twadmin - -m P /etc/tripwire/twpol.txt’ or rerunning the configuration script (twinstall.sh). We will also need to reinitialize the tripwire database.

Tripwire integrity checking should be run regularly and report the result to administrators. The email capability is enabled by adding "emailto = " in each of the "rulename" section in the policy file like the example below.

```
(
rulename = "Shell Binaries",
severity = $(SIG_HI),
emailto = <email address>
)
```

Below is an example of crontab statement for running nightly Tripwire integrity check.

```
05 1 * * * /usr/sbin/tripwire --check –email-report 1> /dev/null 2>&1
```

Please keep in mind that because this host runs up2date nightly that if updates were done the next tripwire report will contain violations related to the files affected by these updates. So it's a good idea to run tripwire database initialization after the up2date process to avoid the unnecessary violations being reported.

File system /usr can be mounted as read-only as a protection against any Trojan binaries being installed by hacker.

Modify /usr mount statement in /etc/fstab:

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

Use the following commands to remount it to allow write access before installing rpm or other applications, and change it back to read only after installation is completed.

```
mount –o remount,rw /usr
mount –o remount,ro /usr
```

Logging:

SYSLOG-NG will be installed in "Ongoing Maintenance" later to be used to better manage and filter log messages. But in case the regular SYSLOG daemon is desired, its configuration is also presented here. Note that log rotation described here applies to either regular SYSLOG daemon or SYSLOG-NG is used. Below is SYSLOG daemon's configuration file (/etc/syslog.conf).

```
# Log all info or higher messages, except facilities that use their own log
*.info;authpriv,auth,mail,cron,kern,local7.none        /var/log/messages
```

```
# log security related messages (for example, authentication messages)
# in /var/log/secure
auth,authpriv.*                         /var/log/secure

# Send mail messages to a separate file.
mail.*                                  /var/log/maillog

# Send crond and atd messages to a separate file.
cron.*                                  /var/log/cron

# Send kernel messages to a separate file. Note that this will
# include messages generated by iptables about blocked network traffic.
kern.*                                  /var/log/kernel

# Everybody gets emergency messages
*.emerg
# Save boot messages also to boot.log
local7.*                                /var/log/boot.log

# Remote logging host
kern.*        @log-server
authpriv.*    @log-server
mail.*        @log-server
```

When a system is compromised one of the first things the intruder does is to hide the traces of illegal activities either by modifying or erasing system logs. So to protect the log's integrity under those circumstances, the last three lines of the above configuration tell SYSLOG daemon to also forward log messages to a remote log host.

Note that SYSLOG daemon will create the log files specified in the configuration if they don't exist already when it starts. This is preferred than creating these log files manually because log files created by daemon has more secured permissions that only root can read and write, while the permissions of manually created log files depend on the setting of environment variable "umask" (often set to 022 to allow group and world to read). Allowing world to read log files that may contain important information is considered a security risk.

Red Hat 9 comes with a log rotation program that moves old log files (and compress if we set so) away and creates new ones to prevent log files growing too big and filling up the file system. The log rotation configuration file (/etc/logrotate.conf, see below) should be modified to rotate log files less frequently (monthly instead of default weekly) and that the old logs retained longer (yearly instead of default monthly).

```
monthly         # rotate log monthly
rotate 12       # keep a year's logs
compress        # compress old log to conserve space
```

Make sure that all log files specified in /etc/syslog.conf (shown below) are in the first line of /etc/logrotate.d/syslog to allow rotation.

```
/var/log/messages /var/log/secure /var/log/maillog /var/log/boot.log /var/log/cron
/var/log/kernel {
    sharedscripts
    postrotate
        /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true
    endscript
}
```

Configure the Network Time Protocol (NTP) daemon:

As computers and networks grow more inter-connected, especially in the client-server mode, an accurate system time becomes vitally important. During the forensic investigation of a computer security incident, it's also very important that the system records events with the correct time tag so that event messages among different systems at a central log host can be correlated.

NTP (version 4.12 on RedHat 9) daemon enables a computer's system clock to synchronize itself to UTC (Universal Time Coordinated) as close as possible by polling time information from other NTP servers. To run NTP more reliably it's best not to use broadcast mode for both the NTP servers and clients (without "broadcast" in server's configuration file and "broadcast client" in the client configuration file). The client may miss the broadcast updates and may be unable to determine the delay between the server and itself. Also to avoid an attacker impersonates one of the NTP servers and provides clients with bogus time information (which will be thrown out by NTP daemon) more than one servers are normally specified. Below is an example of the configuration file (/etc/ntp.conf).

```
driftfile /etc/ntp/drift

server x.x.x.x
server x.x.x.x
server x.x.x.x

restrict default ignore
restrict x.x.x.x nomodify noquery
restrict x.x.x.x nomodify noquery
restrict x.x.x.x nomodify noquery
restrict 127.0.0.1
```

Notes: 1. x.x.x.x is either the hostname or IP address of the local NTP servers, or public NTP server. A list of the public NTP servers can

be found at http://www.eecis.udel.edu/~mills/ntp/servers.html. If one of the local NTP server has a local reference clock it should be defined as preferred by adding "prefer" after hostname/IP address of that server. NTP daemon put more weight on the preferred server in its synchronization algorithm.

2. "driftfile" directive specifies a file that contains the frequency offset of the local clock oscillator calculated by NTP daemon.
3. Access control from the remote hosts is done via the "restrict" statements. The first "restrict" line sets the default that no machines can synchronize with this host and all administrative requests are ignored. The next three lines allow this host to synchronize time with the three NTP servers listed above but does not permit them to modify this host's configuration. The last "restrict" line allows the local administrator to communicate and make changes to the running NTP daemon.

Restrict users who can run cron and at jobs.

Unix allows users (including root) to schedule jobs to be run at a regular basis (cron), or once at a specific time (at). By default, any user can run both cron and at jobs. To prevent incidents like denial of service caused by a local attacker scheduling malicious cron jobs it's best to only allow ones who need to run the scheduled jobs to use this capability. This is controlled by /etc/cron(at).allow and /etc/cron(at).deny files. The deny files list the users who are denied to use cron or at. But normally the deny files are removed and only use allow files. For example on this host, /etc/cron.allow contains only "root" so that only "root" user can run cron jobs.

Install Bug Tracking Server

The Bug Tracking Server has a web interface so the first step of this process is to install and configure an Apache web server, then the main engine Bugzilla and its components are installed.

Install Apache Web server

Download httpd-2.0.48.tar.gz and httpd-2.0.48.tar.gz.md5 from http://www.apache.org. Run 'md5sum –c httpd-2.0.48.tar.gz.md5' to verify the checksum.

Build httpd-2.0.48 to the default location, /usr/local/apache2:

```
CFLAGS=-I/usr/kerberos/include ./configure --disable-isapi --disable-status --disable-userdir --disable-cgid
make
```

make install

The “CFLAGS=-I/usr/kerberos/include” above is a work-around for a known problem building Apache 2.0 on RedHat 9. This server is not using kerberos authentication. Note that by default Apache includes a limited number of modules that have status as “Base” (check the list of modules from the documentation at www.apache.org) in the executable. In the above ‘configure’ command the following base modules that are not needed are disabled:

Module isapi – This module handles ISAPI extensions for Windows. It’s not applicable to Unix.
Module status – This module provides server administrator with information on server activity and performance. It’s decided not to include this module to avoid this information being accessed illegally.
Module userdir – This module allows user-specific directory being accessed. This server has no such needs.
Module cgid – This module executes CGI scripts using an external CGI daemon. It’s for performance improvement for a multi-threaded server. This server has a single CPU so this module doesn’t apply.

Note that another approach to building Apache is to disable all modules first, then enable only the ones that are necessary. Please be aware that what modules are needed for a web server often depend on the applications (CGI programs) running on that server.

Modify Apache main configuration file (/usr/local/apache2/conf/httpd.conf):

Create a new Unix user and group “bugzilla” in /etc/passwd and /etc/group respectively to be used in httpd.conf.

```
$groupadd bugzilla
$useradd –g bugzilla –d /usr/local/bugzilla –s /bin/false bugzilla
```

Set the following configuration directives in httpd.conf as shown below.

```
ServerName < canonical name of Bug.Tracking.Server>
UseCanonicalName on
User bugzilla
Group bugzilla
ServerAdmin <email address>
ServerSignature off
ServerTokens Prod
ServerRoot "/usr/local/apache2"
DocumentRoot "/usr/local/bugzilla/"
AddHandler cgi-script .cgi
<Directory />
   Options None
```

```
    Order allow,deny
    Deny from all
    Satisfy all
    AllowOverride None
</Directory>
```

The “ServerName” and “UseCanonicalName on” directives tell the web server always use the canonical name defined in “ServerName”. The e-mail address defined by “ServerAdmin” will be appear in web server’s error messages. “ServerSignature off” and “ServerTokens Prod” hide the Apache’s version in footer messages and header.

Between <Directory/> and </Directory> contains the default policy of the web server. This very restricted policy (not allowing any access, etc.) applies to the entire web server directory structure (from / and down), while an individual directory can be defined later to a less restricted setting (see /usr/local/bugzilla below for an example). This default policy can prevent mistakes such as a directory did not define any access control.

With this configuration, when httpd receives a request on port 80, it will fork a child process with an effective user ID and group ID as “bugzilla” to process this request. All actions taken in response to the request are done with the privilege of the unprivileged “bugzilla” user and group.

The “DocumentRoot”, /usr/local/bugzilla, will contain the Bugzilla software to be installed later. “AddHandler cgi-script .cgi” tells the web server to recognize scripts with file name extension “.cgi” as CGI scripts. Below shows how /usr/local/bugzilla is defined in httpd.conf:

```
<Directory "/usr/local/bugzilla/">
    DirectoryIndex index.html index.cgi
    Options ExecCGI
    AllowOverride None
    AuthType Basic
    AuthName "Bug Tracking Server"
    AuthUserFile /usr/local/apache2/conf/authenticate/users
    Require valid-user
    Order deny,allow
        Allow from x.x.x.x
        …
    Satisfy any
    Deny from all
</Directory>
```

“DirectoryIndex” tells web server to use index.cgi if index.html does not exist in this directory to render the page when this directory is requested (Bugzilla distribution only has index.cgi). Option “ExecCGI” allows CGI scripts to run in this directory. The “AllowOverride None” directive tells Apache to ignore another configuration file .htaccess and only use the authentication method as described in this file. “AuthType

Basic" is required for "AuthUserFile" and "Require" to work. "AuthName" sets the string that will be displayed in a window when Apache prompts for user name and password for authentication ("Check the system configuration" near the end of this paper includes a figure of this window). "Require valid-user" tells Apache to allow the users listed in "AuthUserFile". The Apache "Order" directive often causes confusions as to whether access is allowed or denied by default. The above setting ignores the default and force Apache to allow access immediately if the client host matches a host in the "Allow from" directives, otherwise prompt for user name and password for authentication. Access is denied if none of the above succeeds. Regardless of how the "Order" is defined thorough testing is required to ensure that it's behaved as expected.

To add each user's login ID and password to the "AuthUserFile", use this command '/usr/local/apache2/bin/htpasswd -f /usr/local/apache2/conf/authenticate/users <username>'.

Start up Apache server:

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

Enter the URL of the host to see the default Apache home page.
Now stop the Apache server and continue installation.

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

Install Bugzilla

Download bugzilla-2.16.4.tar.gz from http://www.bugzilla.org.

Untar this file at /usr/local, this creates directory /usr/local/bugzilla2.16.4.

```
tar xzf bugzilla-2.16.4.tar.gz
```

Change directory name /usr/local/bugzilla-2.16.4 to /usr/local/bugzilla used as document root in Apache configuration. Since all bug tracking requests are handled by user and group "bugzilla, this directory doesn't need to allow any permission for the world with the following permissions. Change the owner/group of this directory to bugzilla:bugzilla.

```
mv /usr/local/bugzilla-2.16.4 /usr/local/bugzilla
chmod 750 /usr/local/bugzilla
chown -R bugzilla:bugzilla /usr/local/bugzilla
```

Below is a brief description of how to complete the rest of the Bugzilla installation. More detailed information can be found in section "Step-by-step Install" at the above Bugzilla web site or the local HTML file /usr/local/bugzilla/docs/html/index.html.

Perl

Perl programming language comes with Red Hat distribution. It can be downloaded from www.perl.org if it was not installed during the Red Hat 9 installation. Use 'rpm –qa' to verify that perl-5.8.0-88.3 is installed.

Bugzilla also needs additional Perl modules that can be retrieved from CPAN (Comprehensive Perl Archive Network) with this command.

```
perl –MCPAN –e 'install "Bundle::Bugzilla"'
```

Make a symbolic link to /usr/bonsaitools/bin/perl from /usr/bin/perl to reflect the location of Perl executable to avoid modifying all the Bugzilla .cgi files to correct the location of perl.

```
ln -s /usr/bin/perl /usr/bonsaitools/bin/perl
```

MySQL

MySQL should also already be installed during the Red Hat 9 installation (alternatively it can be downloaded from www.mysql.com). Use 'rpm –qa' to verify that mysql-3.23.58-1.9 and mysql-server-3.23.58-1.9 are installed.

Set up the MySQL database:

Fist start mysqld, '/etc/init.d/mysqld start'.

Set user root password:

```
# mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD('<new_password>') WHERE
user='root';
mysql> FLUSH PRIVILEGES;
```

Set user bugs' permission and password:

```
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,INDEX,
ALTER,CREATE,DROP,REFERENCES ON bugs.* TO bugs@localhost
IDENTIFIED
mysql> FLUSH PRIVILEGES;
```

Note that the above MySQL GRANT command gives privileges (SELECT, INSERT, UPDATE…etc.) to "bugs" user (the Bugzilla user) to operate on the "bugs" database (the Bugzilla database) from localhost only. When someone accesses the Bug Tracking Server from a remote machine, Bugzilla issues queries to the MySQL database as this "bugs" user (it's not a regular Unix user in /etc/passwd) at localhost. This is why while MySQL listens on TCP port 3306 it doesn't accept remote connections. Also please be reminded that in the local firewall setup, the destination port 3306 is closed. This can be proven by running 'telnet <bug tracking server> 3306' from a remote host and getting no replies.

Now run /usr/local/bugzilla/checksetup.pl script. This script will make sure Bugzilla files and directories have reasonable permissions, set up the data directory, and create all the MySQL tables. The first time it runs it will create /usr/local/bugzilla/localconfig. Edit localconfig to add password (set earlier during setting up MySQL) to user bugs.

'su' to bugzilla and run checksetup.pl again and this time it will ask you to enter the name, e-mail address and password for the administrator of bugzilla.

```
administrator's email:
Enter the real name of the administrator:
password for the administrator account:
```

Test Run Bug Tracking Server

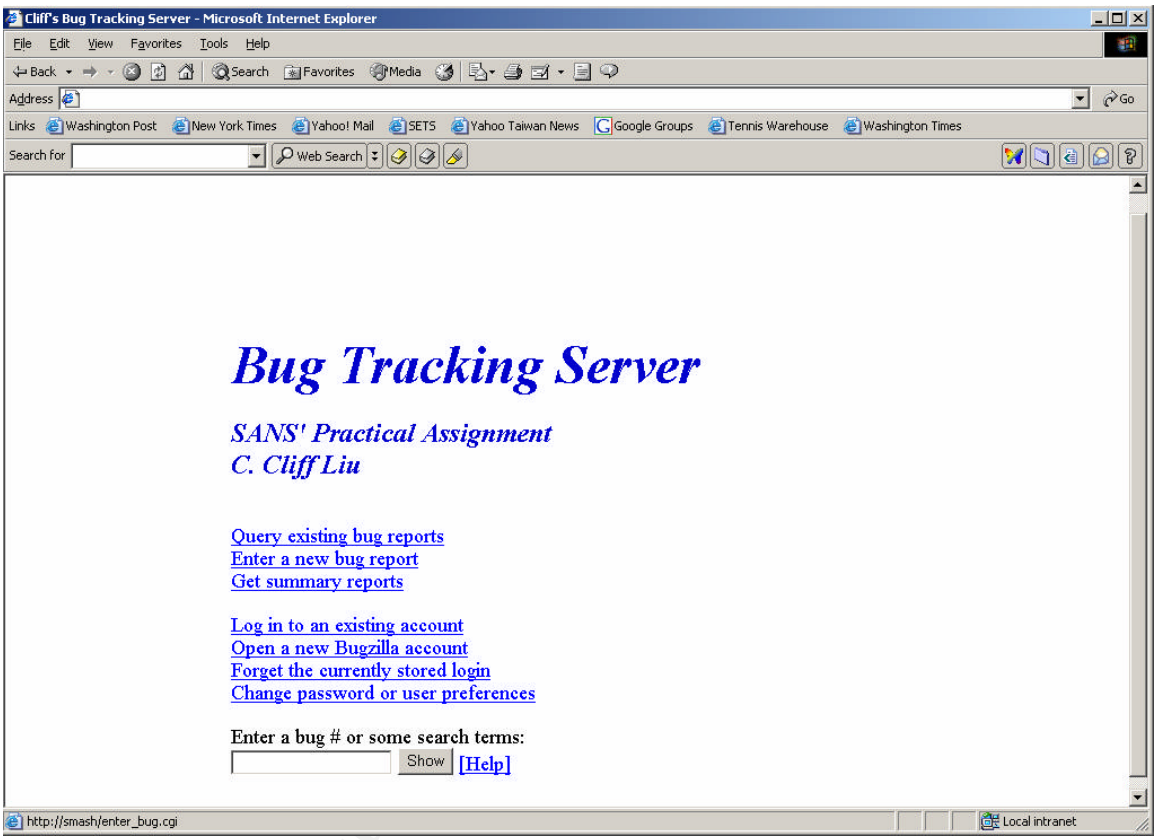
Check if the MySQL server is already started in the previous step.

```
$ps -ef |grep mysql

root      2199     1  0 Mar07 ?        00:00:00 /bin/sh /usr/bin/safe_mysqld --defaults-
file=/etc/my.cnf
mysql     2220  2199  0 Mar07 ?        00:00:00 /usr/libexec/mysqld --defaults-
file=/etc/my.cnf --basedir=/usr --datadir=/var/lib/mysql --user=mysql --pid-
file=/var/run/mysqld/mysqld.pid --skip-locking
```

If MySQL server has not been started, started it with '/etc/init.d/mysqld start'. Start the Apache server with '/usr/local/apache2/bin/apachectl start'.

Enter the hostname of the Bug Tracking Server to get its home page. Click "Log in to an existing account" and enter the Bugzilla administrator's e-mail address and password set above. Go to the bottom of the new page and click Edit parameters. These parameters concern the local administration of the Bugzilla server and are not related to the security of the system. They

include the email address of the person responsible for this site, the base URL address of the site, the content of email sent to user after an account is created, etc. Please refer to “5.1. Bugzilla Configuration” in the document mentioned above for more detail. Click Submit Changes button after changes are made. Below shows the Bug Tracking Server’s home page.



Move Bug Tracking Server to chroot environment

When the Bug Tracking Server runs in the chroot jail it can only access the resources that are available in the new root directory. First the new root directory tree (/home/chroot/…) is created, then all programs, files, libraries and shell commands the Bug Tracking Server needs are copied into this directory structure. Finally, start-up scripts are modified to start these processes via “chroot” command.

This is going to be a laborious process because not only libraries needed by main daemons such as httpd and mysqld are copied into the chroot directory, all other libraries referenced by the already copied libraries also need to be copied. This process starts with creating a new root directory structure:

```
mkdir -p /home/chroot/dev
mkdir -p /home/chroot/lib
mkdir -p /home/chroot/etc
mkdir -p /home/chroot/var/run
mkdir -p /home/chroot/usr/lib
mkdir -p /home/chroot/usr/libexec
mkdir -p /home/chroot/kerberos/lib
mkdir -p /home/chroot/usr/local/apache2/bin
mkdir -p /home/chroot/usr/local/apache2/logs
mkdir -p /home/chroot/usr/local/apache2/conf
mkdir -p /home/chroot/usr/local/apache2/lib
mkdir -p /home/chroot/usr/local/bugzilla
```

(More directories will be created later as needed.)

First copy the httpd daemon:

```
cp /usr/local/apache2/bin/httpd /home/chroot/usr/local/apache2/bin/
```

Use ldd to find out what libraries httpd needs.

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

```
libaprutil-0.so.0 => /usr/local/apache2/lib/libaprutil-0.so.0 (0x40150000)
libgdbm.so.2 => /usr/lib/libgdbm.so.2 (0x40168000)
libdb-4.0.so => /lib/libdb-4.0.so (0x4016f000)
libpthread.so.0 => /lib/libpthread.so.0 (0x40218000)
libexpat.so.0 => /usr/lib/libexpat.so.0 (0x40269000)
libapr-0.so.0 => /usr/local/apache2/lib/libapr-0.so.0 (0x40289000)
librt.so.1 => /lib/librt.so.1 (0x402ac000)
libm.so.6 => /lib/libm.so.6 (0x402be000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x402df000)
libdl.so.2 => /lib/libdl.so.2 (0x401f1000)
libc.so.6 => /lib/libc.so.6 (0x401f5000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Based on the output, copy these libraries into the respective directories under chroot directory structure:

```
mkdir -p /home/chroot/usr/local/apache2/lib

cp /usr/local/apache2/lib/libaprutil-0.so.0 /home/chroot/usr/local/apache2/lib
cp /usr/lib/libgdbm.so.2 /home/chroot/usr/lib
cp /lib/libdb-4.0.so /home/chroot/lib
cp /lib/libpthread.so.0 /home/chroot/lib
cp /usr/lib/libexpat.so.0 /home/chroot/usr/lib
cp /usr/local/apache2/lib/libapr-0.so.0 /home/chroot/usr/local/apache2/lib
cp /lib/librt.so.1 /home/chroot/lib
cp /lib/libm.so.6 /home/chroot/lib
cp /lib/libcrypt.so.1 /home/chroot/lib
cp /lib/libnsl.so.1 /home/chroot/lib
cp /lib/libdl.so.2 /home/chroot/lib
cp /lib/libc.so.6 /home/chroot/lib
```

```
mkdir -p /home/chroot/kerberos/lib

cp /usr/kerberos/lib/libgssapi_krb5.so.2 /home/chroot/usr/kerberos/lib
cp /usr/kerberos/lib/libkrb5.so.3 /home/chroot/usr/kerberos/lib
cp /usr/kerberos/lib/libcom_err.so.3 /home/chroot/usr/kerberos/lib
cp /usr/kerberos/lib/libk5crypto.so.3 /home/chroot/usr/kerberos/lib
cp /lib/libresolv.so.2 /home/chroot/lib
cp /usr/lib/libz.so.1 /home/chroot/usr/lib
cp /lib/ld-linux.so.2 /home/chroot/lib
```

Besides libraries, there are other files opened by httpd. Use the following commands to find them:

```
strace -o/tmp/dbin -f -t /usr/local/apache2/bin/httpd
```

(This will start a number of httpd processes. Let them run for a minute, then kill them with '')

The above strace command traces system calls made by, and signals received by, the httpd process and send them to output file /tmp/dbin. Since we are only interested in the files being opened so we use this command to list them:

```
grep open /tmp/dbin  | grep -v ENOENT | \
sed -n 's/.*open[(]["]\([^"]*\)["].*/\1/p'  | sort –u
```

This will produce the output like this:

```
/dev/null
/dev/urandom
/etc/ld.so.cache
/etc/group
/etc/host.conf
/etc/localtime
/etc/nsswitch.conf
…
```

This list shows that two device files need to be created in chroot. Each of the device files should have the same major and minor device number as it does in /var directory.

```
ls -l /dev/null
crw-rw-rw-    1 root     root       1,   3 Jan 30  2003 /dev/null
ls -l /dev/urandom
crw-r--r--    1 root     root       1,   9 Mar  4 18:20 /dev/urandom

mknod /home/chroot/dev/null c 1 3
mknod /home/chroot/dev/urandom c 1 9
chmod 666 /home/chroot/dev/null
chmod 644 /home/chroot/dev/urandom
```

Also according this list more files need to be copied:

```
cp /etc/group  /home/chroot/etc
cp /etc/host.conf /home/chroot/etc
cp /etc/hosts /home/chroot/etc
cp /etc/ld.so.cache /home/chroot/etc
cp /etc/localtime /home/chroot/etc
cp /etc/nsswitch.conf /home/chroot/etc
cp /etc/passwd /home/chroot/etc
cp /etc/resolv.conf /home/chroot/etc
cp /lib/libnss_files.so.2 /home/chroot/lib
cp /usr/local/apache2/conf/ssl.conf /home/chroot/usr/local/apache2/conf
```

Remove all accounts from /home/chroot/etc/passwd and group except root, mysql, bugzilla and smmsp.

The passwd file now looks like this:

```
root:x:0:0:root:/root:/bin/bash
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
bugzilla:x:503:503::/usr/local/bugzilla:/sbin/nologin
smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin
```

The group file looks like this:

```
root:x:0:root
mysql:x:27:
bugzilla:x:503:
smmsp:x:51:
```

To allow syslogd to also get logs from chroot environment, tell syslogd to also listen on another socket and restart syslogd to see the socket being created.

Modify /etc/sysconfig/syslog to have this line:

```
SYSLOGD_OPTIONS="-m 0 -a /home/chroot/dev/log"
```

Restart syslogd: '/etc/init.d/syslog restart'.

```
$ls –l /home/chroot/dev/log
srw-rw-rw-    1 root     root           0 Mar 10 16:20 log
```

Next copy the mysql files:

```
cp /usr/libexec/mysqld /home/chroot/usr/libexec
cp /usr/bin/safe_mysqld /home/chroot/usr/bin
cp /usr/bin/mysql /home/chroot/usr/bin
cp /etc/my.cf /home/chroot/etc
mkdir -p /home/chroot/var/log
mkdir -p /home/chroot/var/lib
```

```
cp -rp /var/lib/mysql /home/chroot/var/lib/
cp -rp /usr/lib/mysql /home/chroot/usr/lib/
```

Same as httpd above, run the ldd and strace commands to find more libraries and files and copy them into chroot.

```
ldd /usr/libexec/mysqld

cp /usr/lib/libstdc++.so.5 /home/chroot/usr/lib/
cp /lib/libgcc_s.so.1 /home/chroot/lib/
cp /lib/libtermcap.so.2 /home/chroot/lib/

strace -o/tmp/dbin -f -t /usr/bin/safe_mysqld
grep open /tmp/dbin  | grep -v ENOENT | \
        sed -n 's/.*open[(]["]\([^"]*\)["].*/\1/p'  | sort -u

mknod /home/chroot/dev/tty c 5 0
cp /etc/services /home/chroot/etc/services
mkdir /home/chroot/tmp
chmod 777 /home/chroot/tmp
mkdir -p /home/chroot/usr/share/locale
cp /usr/share/locale/locale.alias /home/chroot/usr/share/locale/
cp -rp /usr/share/mysql /home/chroot/usr/share/
```

Copy sh shell to chroot since MySQL startup script /usr/bin/safe_mysqld is a sh script.

```
cp /bin/sh /home/chroot/bin
```

Test run MySQL to see if more files are needed:

```
chroot /home/chroot /usr/bin/safe_mysqld --defaults-file=/etc/my.cnf
```

The above command causes /usr/bin/safe_mysqld to be executed relative to the new root, /home/chroot. However at this time it produced the following error messages on standard output (the shell window).

```
/usr/bin/safe_mysqld: line 1: my_print_defaults: command not found
/usr/bin/safe_mysqld: line 1: /bin/hostname: No such file or directory
/usr/bin/safe_mysqld: line 187: touch: command not found
/usr/bin/safe_mysqld: line 187: chown: command not found
Starting mysqld daemon with databases from /var/lib/mysql
/usr/bin/safe_mysqld: line 1: date: command not found
/usr/bin/safe_mysqld: line 243: rm: command not found
…
```

So more files need to be copied into the choot environment.

```
cp /usr/bin/my_print_defaults /home/chroot/usr/bin/
cp /bin/hostname /home/chroot/bin/
cp /bin/touch /home/chroot/bin/
```

```
cp /bin/chown /home/chroot/bin/
cp /bin/date /home/chroot/bin/
cp /bin/rm /home/chroot/bin/
cp /usr/bin/tee /home/chroot/usr/bin/
cp /bin/sed /home/chroot/bin/
cp /usr/bin/nohup /home/chroot/usr/bin/
```

Run the above safe_mysqld again. Run 'ps -ef |grep mysql' to show that mysqld is running.

```
mysql     6502     1  0 20:37 pts/9    00:00:00 /usr/libexec/mysqld --defaults-
file=/etc/my.cnf --basedir=/usr --datadir=/var/lib/mysql --user=mysql --pid-
file=/var/lib/mysql/.pid --skip-locking
```

Next copy the bugzilla and perl.

```
cp -rp /usr/local/bugzilla /home/chroot/usr/local
cp /usr/bin/perl /home/chroot/usr/bin
cp -rp /usr/lib/perl5 /home/chroot/usr/lib/
```

List the libraries Perl needs.

```
ldd /usr/bin/perl
```

Most of the libraries listed here have already been copied earlier except one.

```
cp /lib/libutil.so.1 /home/chroot/lib
```

Since all bugzilla perl scripts reference /usr/bonsaitools/bin/perl instead of /usr/bin/perl for Red Hat, to avoid modifying each script, make the following symbolic link:

```
ln –s  /usr/bin/perl /home/chroot/usr/bonsaitools/bin/perl
```

The Bugzilla uses e-mails to communicate with users about the account information and bug status, etc. it's necessary to copy the sendmail program and the associated files into the chroot environment.

Copy sendmail program:

```
cp /usr/lib/sendmail /home/chroot/usr/lib
```

Copy libraries:

Run 'ldd /usr/lib/sendmail' and copy the following libraries according to the output:

```
cp /usr/lib/libwrap.so.0 /home/chroot/usr/lib/libwrap.so.0
cp /usr/lib/libldap.so.2 /home/chroot/usr/lib/libldap.so.2
```

```
cp /usr/lib/liblber.so.2 /home/chroot/usr/lib/liblber.so.2
cp /usr/lib/libsasl.so.7 /home/chroot/usr/lib/libsasl.so.7
cp /usr/lib/libhesiod.so.0 /home/chroot/usr/lib/libhesiod.so.0
cp /lib/libpam.so.0 /home/chroot/lib/libpam.so.0
cp -rp /usr/lib/locale /home/chroot/usr/lib
```

Copy configuration files:

```
cp /etc/mail/sendmail.cf /home/chroot/etc/mail
cp /etc/mail/submit.cf /home/chroot/etc/mail
cp /etc/mail/local-host-names /home/chroot/etc/mail/local-host-name
cp /etc/mail/trusted-users /home/chroot/etc/mail/trusted-users
cp /etc/aliases.db /home/chroot/etc/aliases.db
cp /lib/libnss_dns.so.2 /home/chroot/lib/libnss_dns.so.2
```

Create spool directories:

```
mkdir -p /home/chroot/var/spool/mqueue
mkdir /home/chroot/var/spool/clientmqueue
chown smmsp:smmsp /home/chroot/var/spool/clientmqueue
chmod 770 /home/chroot/var/spool/clientmqueue
```

Finally create two chroot start-up scripts in /etc/init.d for mysqld and httpd as shown below. If it's desired to start Bug Tracking Server at system boot time, two symbolic links should be made from the run level startup directory to these two scripts. For example if the default run level is 5 (the first line in /etc/inittab is "id:5:initdefault"), make the following links:

```
ln –s /etc/init.d/mysql.sh /etc/rc5.d/ S78mysqld
ln –s /etc/init.d/ apache.sh /etc/rc5.d/ S94httpd
```

mysql.sh:

```
 #!/bin/sh
CHROOT=/home/chroot/
MYSQLD=/usr/bin/safe_mysqld
PIDFILE=/var/run/mysqld/mysqld.pid

case "$1" in
start)
 /usr/sbin/home/chroot $CHROOT $MYSQLD --defaults-file=/etc/my.cnf >/dev/null 2>&1 &
        ;;
stop)
 /bin/kill `cat ${CHROOT}/${PIDFILE}  2> /dev/null ` > /dev/null 2>&1
        ;;
*)
        echo ""
        echo "Usage: `basename $0` {start|stop}" >&2
        exit 64
        ;;
esac
exit 0
```

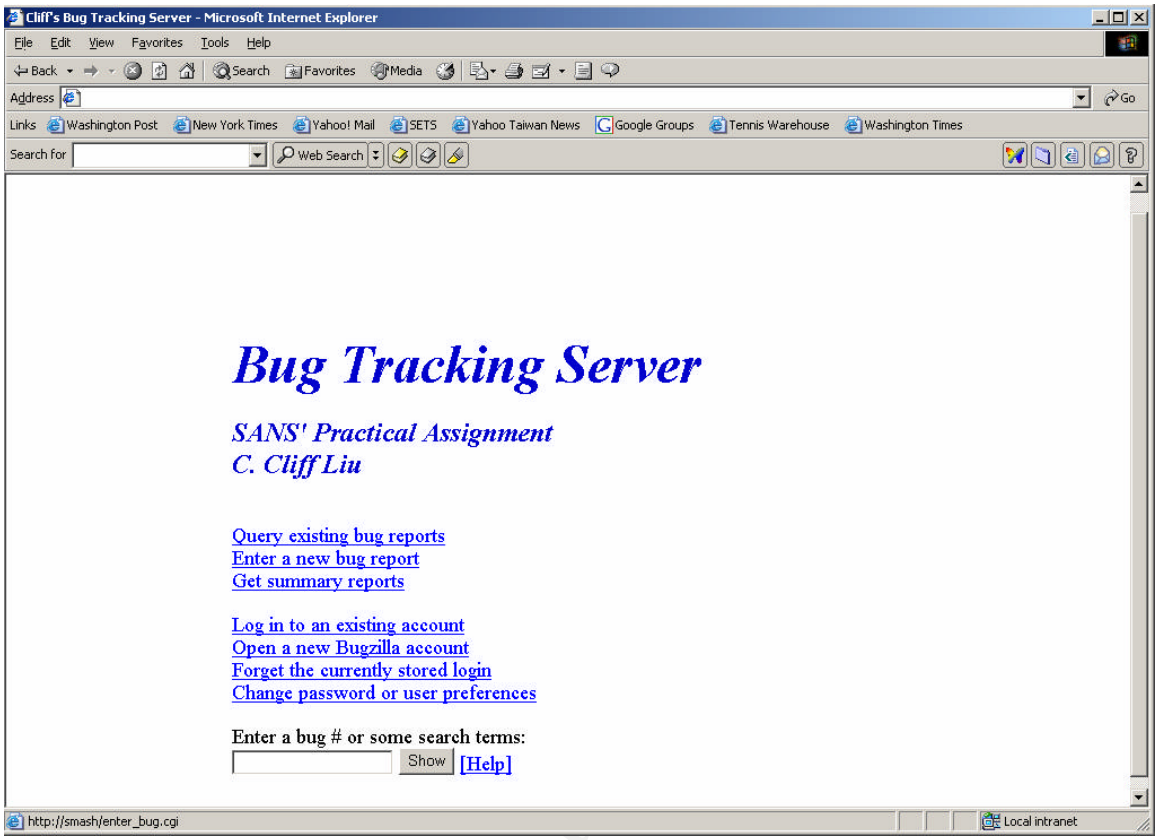
apache.sh:

```
#!/bin/sh
CHROOT=/home/chroot/
HTTPD=/usr/local/apache/bin/httpd
PIDFILE=/usr/local/apache/logs/httpd.pid

echo -n " apache"

case "$1" in
start)
        /usr/sbin/home/chroot $CHROOT $HTTPD
        ;;
stop)
        kill `cat ${CHROOT}/${PIDFILE}`
        ;;
*)
        echo ""
        echo "Usage: `basename $0` {start|stop}" >&2
        exit 64
        ;;
esac
exit 0
```

To start the Bug Tracking Server, run these two commands:

/etc/init.d/mysql.sh start
/etc/init.d/apache.sh start

On a web browser, enter the server's URL to get the home page:



As this point the Bug Tracking Server is working in a chroot environment and so it concludes the project. The Bug Tracking Server will likely need some user interface customizations, for example modifying the HTML pages to add the organization specific information. These additional customizations are beyond the scope of this project.

Ongoing maintenance

Keep system up-to-date:

It's most important to perform software and security patch updates regularly to keep the system secure. This is achieved by running the up2date service nightly as described in "Install and secure Red Hat 9 operating system" earlier. However, this only updates the regular Red Hat 9 environment while Bug Tracking Server is running in a chroot environment. Even though security holes are not as dangerous in a chroot

environment as when they are in a regular environment because of the limited resources available in a chroot environment, it's still a good idea to keep these two environments synchronized.

One way to do this is to cut and paste all the 'cp' commands (see below for files that should be excluded) used in above "Move the Bug Tracking Server to chroot environment" into a script file and rerun it as needed. Another way is to use 'rsync' commands, for example to keep /lib and /home/chroot/lib synchronized, 'rsync -aLvx --existing /lib /home/chroot' can be used. Cautions should be taken when using 'rsync' command (test first) to avoid unexpected outcome. In the above rsync command, option –L tells rsync to copy the file referenced by symbolic link instead of the link itself, option "--existing" tells rsync to copy only files that exist on the destination.

Also keep in mind that some directories that contain the database or customized configurations should not be copied from the regular environment to chroot environment. These directories are listed below:

/home/chroot/var/log – containing mysqld log file.
/home/chroot/usr/local/bugzilla – containing bugzilla, update can be done here with 'cvs update' command (see bugzilla web site in references for more detail)
/home/chroot/etc – containing passwd, group files.
/home/chroot/var/run – containing running mysqld pid.
/var/lib/mysql – containing mysql database.
/home/chroot/dev – containing device files
/home/chroot/usr/local/apache2/conf – containing Apache configuration files.

Analyze Log:

Also very important for the ongoing maintenance is to monitor system logs that contain important security and health information about the system. Because the Bug Tracking Server is running in a chroot environment some logs, such as the MySQL and Apache logs, are scattered in different directories so for easier management we'd like to have all logs in the regular log directory - /var/log. Further more, since these system logs contain a lot of information that sometimes the important information we need to pay attention to promptly were buried in this massive messages and were ignored. So we'd like to filter these potentially important messages out and store them in separate files that will be checked regularly. We also like to get email notification immediately when emergency conditions occur. To be able to do the above we installed syslog-ng, available at http://www.balabit.com/products/syslog_ng.

Download the latest stable software syslog-ng-1.6.2, and libol-0.3.13 syslog-ng uses, from this site and follow the simple installation instructions in file INSTALL to install them. The “contrib.” directory in the source tree contains the sample startup script and configuration file. Copy init.d.RedHat-7.3 to /etc/init.d/syslog-ng, syslog-ng.conf.RedHat to /etc/syslog-ng.conf. Edit /etc/sysconfig/syslog-ng so it contains SYSLOGNG_OPTIONS="-f /etc/syslog-ng.conf".

Similar to syslogd, syslog-ng can filter messages based on the facility/priority pairs and generate separate log files, and forward log messages to a remote host. In addition syslog-ng can use a boolean expression to filter log messages based on the content of these messages. More work needs to be done to fine tune the filter expressions to capture the events we are most interested. Below shows a simple configuration file, /etc/syslog-ng.conf:

```
options { sync (0);
        time_reopen (10);
        log_fifo_size (1000);
        long_hostnames (off);
        use_dns (yes);
        use_fqdn (no);
        create_dirs (no);
        keep_hostname (yes);
      };

source src { pipe ("/proc/kmsg" log_prefix("kernel: ")); unix-stream ("/dev/log"); internal();
};

destination d_cons  { file("/dev/console"); };
destination d_kern  { file("/var/log/kernel"); };
destination d_mesg  { file("/var/log/messages"); };
destination d_auth  { file("/var/log/secure"); };
destination d_mail  { file("/var/log/maillog"); };
destination d_spol  { file("/var/log/spooler"); };
destination d_boot  { file("/var/log/boot.log"); };
destination d_cron  { file("/var/log/cron"); };
destination d_http  { file("/var/log/httpd"); };
destination d_mysql { file("/var/log/mysqld"); };
destination d_ssh   { file("/var/log/sshd"); };
destination d_tcp   { tcp(“xxx.xxx.xxx.xxx”); };
destination d_mlal  { usertty("*"); };
destination d_maillog { program(“/usr/local/bin/syslog-mail” ) };
filter f_kern       { facility(kern); };
filter f_mesg       { level(info) and
                  not (facility(mail)
                  or facility(authpriv) or facility(cron)); };
filter f_auth       { facility(auth, authpriv); };
filter f_mail       { facility(mail); };
filter f_emergency  { level(emerg); };
filter f_cron       { facility(cron); };
```

```
filter f_http       { facility(local5); };
filter f_mysql      { facility(local6); };
filter f_ssh        { match("Failed") or match("refused"); };

log { source(src); filter(f_kern); destination(d_kern); };
log { source(src); filter(f_mesg); destination(d_mesg); };
log { source(src); filter(f_auth); destination(d_auth); destination(d_tcp); };
log { source(src); filter(f_mail); destination(d_mail); };
log { source(src); filter(f_emergency); destination(d_mlal); };
log { source(src); filter(f_alert); destination(d_maillog); };
log { source(src); filter(f_filter6); destination(d_spol); };
log { source(src); filter(f_cron); destination(d_cron); };
log { source(src); filter(f_mysql); destination(d_mysql); };
log { source(src); filter(f_http); destination(d_http); };
log { source(src); filter(f_mysql); destination(d_mysql); };
log { source(src); filter(f_ssh); destination(d_ssh); };
```

Please see syslog-ng.conf manpage or on-line documentation at the above site for information about the configuration file. Note that each log file specified in the configuration file will be created by syslog-ng daemon if it does not already exist. Again it's preferred to let daemon create these log files due to the permission issue described before.

In the above configuration the httpd and mysqld logs are filtered based on facility local5 and local6 respectively. For that to work we need to run the following two scripts.

To feed the Apache log file access_log into syslog-ng, run this script to send its content to facility local5:

```
#!/bin/sh
tail -f /home/chroot/usr/local/apache2/logs/access_log | logger -p local5 -t httpd &
```

To feed the MySQL log file mysqld.log into syslog-ng, run this script to send its content to facility local6:

```
#!/bin/sh
tail -f /home/chroot/var/log/mysqld.log | logger –p local6 –t mysqld &
```

Additional notes about the above configuration:

1. In the above configuration the sshd log cotains only the ssh connections that are rejected by TCPWrapper or password authentication.
2. The authpriv facility messages are still in /var/log/secure and a copy of it is sent to a loghost (see destination (d_tcp)).
3. The messages of priority alert and emerg are sent to system administrator via email. The shell script /usr/local/bin/syslog-mail looks like this (from http://www.campin.net/perl-mail.txt):

```
#!/bin/sh
```

```
while read line; do
        echo $line | /bin/mail -s "log alert" administrator
done
```

4. The new log files need to be appendeded to the first line of /var/logrotate.d/syslog (see “Logging” during system hardening):

```
/var/log/messages /var/log/secure /var/log/maillog /var/log/boot.log /var/log/cron
/var/log/kernel /var/log/httpd /var/log/mysqld /var/log/sshd {
   sharedscripts
   postrotate
      /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true
   endscript
}
```

Finally to run syslog-ng instead of syslog, turn off syslog and stop its daemon, then trun on syslog-ng and start its daemon:

```
chkconfig syslog off
/etc/init.d/syslog stop
chkconfig syslog-ng on
/etc/init.d/syslog-ng start
```

Keep Reliable Backup:

A complete backup is necessary to ensure the availability of this system. This system is backed up daily via Networker to a tape library as described in “Install and secure Red Hat 9 operating system”. To verify the integrity of the backup, do regular drills such as using Networker recover program (nwrecover) to restore some files from tapes and verify that the correct files are recovered.

Check the system configuration

This Linux system only provides two Internet services, httpd and sshd, so client applications such as telnet, ftp and finger run from remote sites to this system will be timed out at remote site. Also as described in “MySQL” section that it can be proven (via telnet) that MySQL doesn’t accept connection on port 3306 (MySQL default port) from remote hosts. Five other tests are performed here to verify the system configuration.

Check Firewall Using Nmap

A port-scanning program can be used to check the firewall (Red Hat 9 uses iptables) configuration. Nmap (www.insecure.org/nmap), a popular open

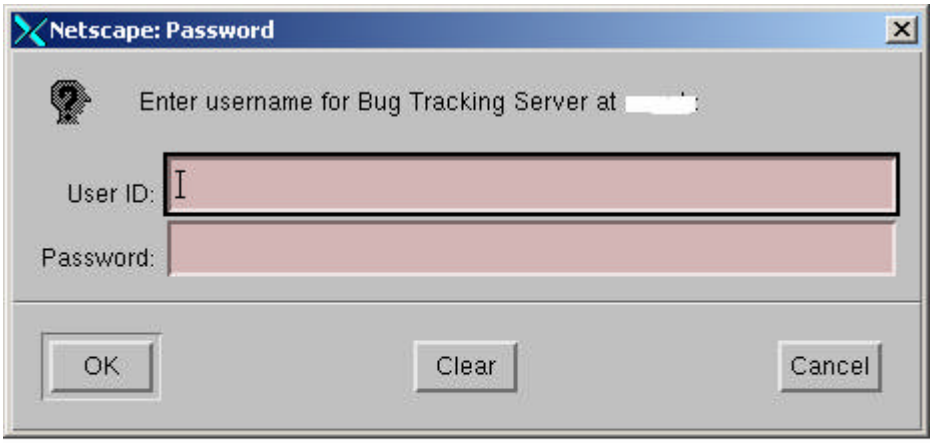
source utility for security auditing, is selected to perform this verification. Download and install the most recent Nmap on another Linux host. Below is the nmap command issued from a remote host:

```
# nmap -P0 -sS -sU <Bug Tracking Server hostname> |grep open
(-P0: since we know the server is up, don't bother ping it first
 -sS : scan TCP ports
 -sU:  scan UDP ports
 see http://www.insecure.org/nmap/nmap_doc.html for more options)

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-02-28 12:30 EST
Interesting ports on xxx.xxx.xxx.xxx (nnn.nnn.nnn.nnn):
(The 1657 ports scanned but not shown below are in state: filtered)
PORT   STATE  SERVICE
22/tcp open   ssh
80/tcp open   http

Nmap run completed -- 1 IP address (1 host up) scanned in 71.055 seconds
```

This proves that the only open TCP ports are 22 and 80 (MySQL TCP port 3306 is closed). This also shows that while some other ports are open for DNS, NTP they are only for specific hosts.

Check Web Server (Apache) Authentication

Apache allows users from trusted client hosts that are listed in the configuration file (httpd.conf) to access the Bug Tracking Server's home page directly. People trying to access the server from any other hosts will get a login screen (shown below which has hostname sanitized) that asks for user ID and password for authentication purpose.



If incorrect user ID and password are entered, a page showing "**Authorization Required**" is displayed.

Check TCP Wrapper Host Access Control

The access control for sshd is configured using TCP Wrapper's host access control file /etc/hosts.allow. This file only contains the allowed hosts for sshd and a DENY for all at the end:

sshd: xxx.xxx.xxx.xxx
.
.
ALL: ALL: DENY

To test access control, an ssh connection to this system is attempted from a host on intranet that's not on /etc/hosts.allow. The ssh connection was timed out at this remote host with the following output:

ssh_exchange_identification: Connection closed by remote host

While the /var/log/sshd on this server shows:

sshd[4098]: refused connect from x.x.x.x

If a remote host is on /etc/hosts.allow a banner will be displayed before login prompt:

NON PUBLIC RESOURCE WITH EXTERNAL CONNECTIONS

```
This resource is for authorized use only.

If not authorized to access this resource, disconnect now.
Unauthorized use of, or access to this resource may subject you to
      disciplinary action or criminal prosecution.

By accessing and using this resource, you are consenting to
      monitoring, keystroke recording or auditing.

cliff  password:
```

Check Chroot Environment

To demonstrate that in a “chroot” environment the file system access is limited to a small root directory, /bin/bash and /bin/ls are copied to /home/chroot. We then issued the following commands:

```
[cliff]# chroot /home/chroot
 bash-2.05b# pwd
 /
 bash-2.05b# ls
 bin  dev  etc  lib  tmp  usr  var
 bash-2.05b# cd dev
 bash-2.05b# ls
 log  null  tty  urandom
```

Check Up2date Process

To ensure that the software package and patch updates are done regularly so in the nightly cron job of up2date an email is sent to administrator after the update:

```
0 1 * * * /usr/sbin/up2date --nox -u -d >> /var/log/current-up2date; /bin/mail -s "Up2date:
`hostname`" cliff < /var/log/current-up2date
```

Below is an example of the email:

```
Thu Mar 11 01:00:00 EST 2004

Fetching package list for channel: redhat-9.0-i386...

Fetching Obsoletes list for channel: redhat-9.0-i386...

Fetching rpm headers...

Testing package set / solving RPM inter-dependencies...
gdk-pixbuf-0.22.0-6.1.0.i38  Retrieved.
```

gdk-pixbuf-gnome-0.22.0-6.1 Retrieved.
kdelibs-3.1-13.i386.rpm: Retrieved.

In summary, a chroot environment enclosed the vulnerable applications in an area where most of the important system resources are unavailable. A break in via the security holes in these applications will be much more difficult to cause damage to the overall system.

References

1. http://www.bugzilla.org/, Bugzilla software and documentation
2. http://httpd.apache.org/, Apache Web Server software and documentation
3. The World Wide Web Security FAQ, http://www.w3.org/Security/
4. SANS Step-by-Step Series Securing Linux Version 1.0
5. http://www.securityfocus.com/infocus/1694: Securing Apache: Step-by-Step
6. “Iptables Tutorial 1.1.19” by Oskar Adreasson (http://iptables-tutorial.frozentux.net/iptables-tutorial.html).
7. http://current.tigris.org/, open-source up2date server software and documentation
8. http://www.tripwire.com/, Tripwire software and documentation
9. http://www.insecure.org/nmap/lamont-nmap-guide.txt , helpful introduction to Nmap.
10. Notes on setting up a NTP subnet, http://www.eecis.udel.edu/~mills/ntp/html/notes.html.

Last Updated: October 17th, 2017

Upcoming Training

SANS 2018	**Orlando, FL**	**Apr 03, 2018 - Apr 10, 2018**	**Live Event**
SANS OnDemand	**Online**	**Anytime**	**Self Paced**
SANS SelfStudy	**Books & MP3s Only**	**Anytime**	**Self Paced**