Global Information Assurance Certification Paper

Securing Red Hat 8 to Run Anonymous FTP Server with VSFTPD

GIAC Certified UNIX Security Administrator (GCUX)
Practical Examination Version 1.9

William Souza

Securing Red Hat 8 to Anonymous FTP Server

Introduction

Anonymous FTP servers are key components on a network for file distribution to clients/users. Anonymous FTP servers are often difficult to secure due to the inherited security issues with the File Transfer Protocol (FTP) daemon. The FTPD should be turned off on all the servers, as part of your security measures, the only exception would be if you are running a FTP server.

Anonymous FTP server can be used as a centralized way to transfer software source code, patches, public documents, public reports and data analysis or any other information deem public.

The objective of this paper will be to lead you step by step through the process of building, securing and deploying an Anonymous FTP server that you will be able to deploy on your DMZ*, with minimal patch maintenance.

Description of the System

The environment I will be using will be a simulated DMZ (see figure 1) and the anonymous FTP server will be running Red Hat 8 as the Operating System with Vsftpd for our FTP functions on an Intel platform. This way I will be able to utilize our in house knowledge on the OS and have the support for the hardware by our vendor. I will try to make this process as easy and explanatory in a step by step format, so novice administrator would be able to deploy a secure server providing an anonymous FTP service.

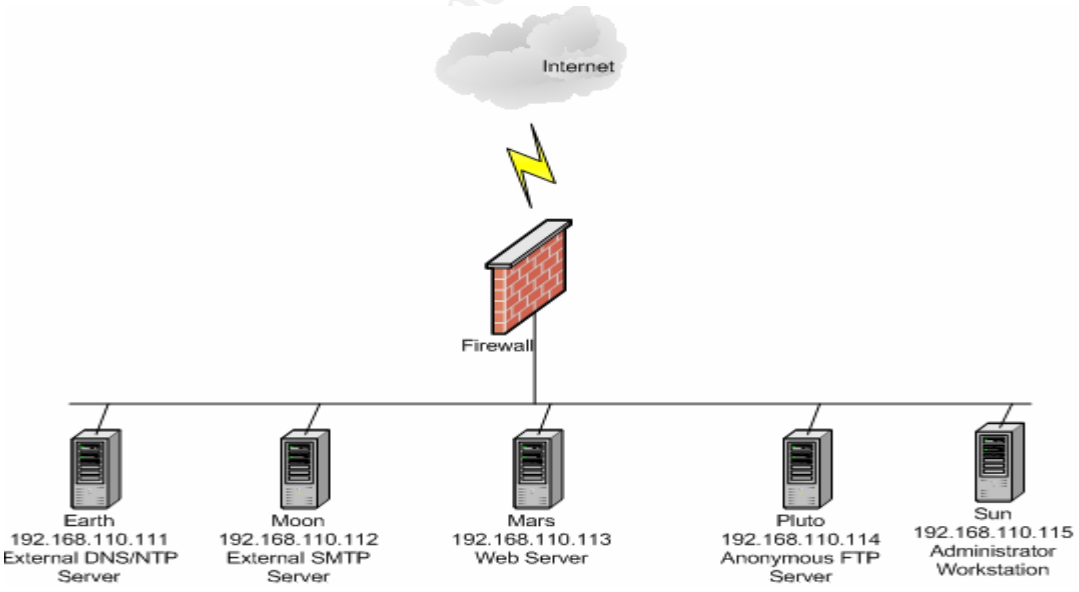


Figure 1: Simulated DMZ

* DMZ: Demilitarized Zone. The DMZ sits between the Internet and your company's corporate network.

The anonymous FTP server will be running on the DMZ, where by designed will have its own protections, such as, firewalls, bastion hosts, etc. The DMZ will allow access from the Internet to our anonymous FTP, which therefore our server will be subject to malicious attack. I will deploy also a host based firewall, on which I decided to use the one that comes with Red Hat 8, the major reason it's because, it is already part of the package and it's free.

One of several objectives that you, as the administrator, need to reach here is to minimize as much as you can, the processes running on the system, so you are less vulnerable to attacks and this will also easy the process of patching the system, which doesn't mean you should lack on your patch monitoring. There are several sites out there that you can get information on the latest vulnerabilities. Another reason I selected Red Hat 8 was that, Sudo and OpenSSH comes with Red Hat and if any vulnerability comes out, you will be able to get the RPM's from Red Hat web site and also eliminating the amount of third party software installed on the system. This system we will be installing ESM[*], so we can monitor our system for Standard compliance and vsftpd which is our FTP client.

This is how it will work, after the system build out is done, the anonymous FTP server will be serving the company's clients through the internet. Sudo[†] will be installed to address the *least privilege*, which gives users the exact amount of access to perform their duties; TCP Wrappers[‡] will also be installed, so we can log any unauthorized used, i.e. someone trying to ssh in to our DMZ through the FTP server; Tripwire[§] will be installed, so we can have a "snap shot" of things before final deployment, so this way we can track any changes on the systems files for example. Ports that will be open are SSH, NTP and FTP; our purpose here is to open only ports that are really needed. The final objective is to provide the company's customers a secure FTP site to conduct business.

The table below shows the software and hardware used to build the system.

CPU	1 – 3.06GHz Xeon processor
Memory	8GB
Network Card	1Gb
Hard Drive	2 x 73GB
Operating System	Red Hat 8.0
OpenSSH	OpenSSH 3.6.1
Sudo	Comes in the distribution
Vsftpd	Vsftpd 1.1.3

Table 1: Hardware/Software

[†] Sudo: Information on sudo can be found here - http://www.courtesan.com/sudo/other.html
[‡] TCP Wrappers: The home page is found here - ftp://ftp.porcupine.org/pub/security/
[§] Tripwire: The academic source release can be found here - http://www.tripwire.com/products/tripwire_asr/

Risk Analysis of the System

This anonymous FTP server will be a very important server in satisfying the customers' demand for a secure environment to conduct business, but at the same time FTP servers are notorious for their vulnerabilities. There are several malicious codes out on the Internet available for any novice attacker to use, so it is a matter of utmost importance to secure the FTP server from attackers.

If the server is not properly secure or configure, it can turn in to a central point for attackers to load and store DDoS* code, illegal music downloads, pornography, and much more. Also if the FTP server is compromise by a buffer overflow and the attacker gain root privileges on the system, the attacker can use this machine to attack your network or change the configuration of the FTP server to serve him/her.

Services running on the system must be kept to a minimal, as explain in an earlier section. This will limit the amount of exploit that can be used against the system and the chance of a success attack. The services that will be running are SSHd, vsftpd, ntpd and esm.

One item in security that most of the time gets overlooked is the physical security, and that is a great risk if not properly setup, because anyone could walk up to the system and have console access to the server, so prepare for everything.

Step by Step Guide

Assumption –

- Any software downloaded and installed on the system will have to be verified by checking its checksum and/or signature before installation, this MUST be done. There were several cases of Trojan software being distributed. Read the following advisories for more information:

http://www.cert.org/advisories/CA-2002-24.html
http://www.cert.org/advisories/CA-2002-28.html
http://www.cert.org/advisories/CA-2002-30.html

It is very important to check the checksum and/or signature, whenever is available.

Installing the Operating System

Red Hat 8 can be downloaded from the Red Hat web site on the following link or any other mirror site:

* DDoS: It is a Distribute Denial of Access.

http://ftp.redhat.com/pub/redhat/linux/8.0/es/iso/i386/

NOTE: On this page you have an MD5SUM, please check it, before installing.

The installation must be performed on an isolated network (test network) or completely disconnected from the network, because the system will be at a vulnerable state till we get everything installed, configure properly, tested and verified.

To start the installation, please insert CD #1 on the cdrom and boot the system. The Red Hat Install screen will come up and it will give you some options:

To install or upgrade Red Hat Linux in graphical mode, press the <enter> key.

To install or upgrade Red Hat Linux in text mode type: "linux text <enter>".

Select "graphical mode" by pressing the <enter> key;

The Red Hat "Welcome" screen will appear, click "next";

- Language Selection screen – accept default (English) click "next", unless you decide to install under a foreign language, and then choose properly;
- Keyboard Selection screen – accept default (U.S. English) click "next", unless otherwise, see above;
- Mouse Configuration screen – accept default (3 button mouse/PS2), click "next", unless you have another type, then select your choice;
- Installation Type screen – select "Custom" and click "next";
- Disk Partitioning Setup screen – you will be prompt with 3 choices and they are:

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

- Partitioning screen will appear for disk setup, which on the first half you will find a progress bar with all your free disk space and below that you will see several buttons:

  [New], [Edit], [Delete], [Reset], [RAID], [LVM]

With these buttons you will be able to configure your filesystem. Below the buttons, on the second half of the screen, you will see your devices (hard drives), which are divided by columns:

Device	Mount Point/RAID/ Volume	Type	Format	Size (MB)	Start	End

- Click on “New”;

A “add partition” screen will appear and you will have several selections to make and they are the following.

Mount Point – click on the down arrow to see your choices and select “/”;
File System Type – accept default “ext3”;
Allowable Drives – we will keep the default disk (‘sda1’ for SCSI or ‘hda1’ for IDE drives);
Size (MB) – we will set “/” at 520MB;
Click “next”;

Now you just have set your first partition on the disk. We need to follow these steps and set up all other partitions.

Click “New”;
Mount Point – select “/home”;
File System Type – “ext3”
Allowable Drives – accept default disk;
Size (MB) – set /home to 1700;
Click “next”;

Click “New”;
Mount Point – select “/tmp”;
File System Type – “ext3”;
Allowable Drives – accept default disk;
Size (MB) – set /tmp to 1000;
Click “next”;

Here we set /tmp in a separate file system, so we can have more control over the mount point settings.

Click “New”;
Mount Point – select “/usr”;
File System Type – “ext3”;

Allowable Drives – accept default disk;
Size (MB) – set /usr to 4000;
Click “next”;

Click “New”;
Mount Point – select “/usr/local”;
File System Type – “ext3”;
Allowable Drives – accept default disk;
Size (MB) – set /usr/local to 4000;
Click “next”;

The directory “/usr/local” will hold any third party software, which for this server’s purpose won’t be many, but the software you might find here will be “tripwire”, “TCP Wrapper”, etc …

Click “New”;
Mount Point – select “/var”;
File System Type – “ext3”;
Allowable Drives – accept default disk;
Size (MB) – set /var to 2000;
Click “next”;

The directory /var can be sized depending on how much logging you want to do on the system.

Click “New”;
Mount Point – skip this step
File System Type – choose swap on the drop down menu;
Allowable Drives – accept default disk;
Size (MB) – set at least the size of your memory.
Click “next”;

Click “New”;
Mount Point – type “/ftpcust”; this will be our ftp directory.
File System Type – “ext3”;
Allowable Drives – choose your second drive on the system;
Size (MB) – set /ftpcust” to the entire disk size;
Click “next”;

This will conclude our disk partitioning; at this point click “Next” at the bottom of the screen.

Boot Loader Configuration is the next screen that will appear and on this screen we will accept all default values. We will keep GRUB as our boot loader and we are not setting a password for the boot loader, click “next”.

- Network Configuration screen will be the next one to appear and here you will have a chance to configure the “Network Devices”, “Hostname” and “Miscellaneous Items”, as explained bellow.

- Network Devices

Click on “Edit” button; (The “Edit Interface eth0” will appear)
Uncheck [ ] Configure using DHCP; (This will activate the IP Address and Netmask options)
Type your IP address and Netmask;
Click “OK”; (This will take you back to the Network Configuration screen)
Type your “Hostname”;
Miscellaneous Items – setup your “Gateway”, “Primary DNS”, “Secondary DNS”, “Tertiary DNS”;
Click “next;

- Firewall Configuration is the next screen; you have several options here also.

Select – [x] No firewall
Click “next”;

NOTE: The reason you are selecting no firewall is because the firewall you have on the network will handle all packet filtering. Your best configuration here will be later implement IPTables, which we’ll go over in later sections.

- Additional Language Support screen will appear; Leave as default and click “next”.

- Time Zone screen will appear and at this point we will pick our time zone, but the only thing we need to make sure here is to set for “daylight savings” time.

Click on the tab “UTC Offset” at the bottom of the screen check the box, “Use Daylight saving time (US Only)”.

Click on the “Location” tab again and at the bottom of the screen check the box “System clock uses UTC”, then click “next”;

Account Configuration screen will appear; Set your “root” password and create your own account at this time. Click “next’

- Authentication Configuration screen will appear; Leave this at default, which will allow you to have MD5 passwords and shadow passwords. Click “next”;
- Package Group Selection will appear; here we will make several selections to secure our server.
  Uncheck everything, but “Network Servers” option;

Click on “details” under “Network Servers” to open a more detail screen for what will be installed under this option.

Uncheck “finger-server” option;
Uncheck “pxe”; we will not need this for our server, so we can go ahead and remove anything that we are not going to use it.
Uncheck “rsh-server”; this option is for all the “r-commands”, also know as “rsh, rlogin and rcp”, we will be using “ssh”;
Uncheck “talk-server”; this option allows you to perform a one-to-one Internet chatting, we are not going to need this installed.
Uncheck “telnet-server”; this will give you the “telnet” capability, but as mentioned before, we will be using “ssh” to satisfy our remote logins.
Uncheck “ypserv”; this option is for the NIS portion of the installation, but since we are not going to use NIS, we don’t need it.
Click “OK”;

You will get the “Package Group Selection” screen back, at this point, check the box at the bottom of the screen, where it says, “Select individual packages”, and then click “next”

- Individual Package Selection window will appear; at this point we will go over each individual package and check if we need them on our server, so we can even further secure our environment.

Amusements

Games:
Uncheck everything under this option, if any checked;
Graphics:
Uncheck everything under this option, if any checked;

Applications

Archiving:
“dump” – Keep checked;
“mnt” – Keep checked;
“unzip” – Keep checked;

Everything else we are going to uncheck, because we are not going to need it on our system.

NOTE: A explanation on each item can be obtain if you highlight the option, at the bottom you will see the package name, version and what it is this package do.

CPAN:

Make sure everything is unchecked here.

Communications:

"lrzsz" – Uncheck this item; this package is for modems.

"minicom" – Keep checked; this option is also for modems, but it also gives us a terminal emulator capabilities.

Make sure everything else is unchecked.

Databases:

Make sure everything is unchecked here.

Editors:

Make sure everything is unchecked here.

Emulators:

Make sure everything is unchecked here.

Engineering:

"bc" – Uncheck this item; this package has numbers handling capabilities and a text mode calculator.

Make sure everything else is unchecked.

File:

"bzip2" – Keep checked; this is a compression utility.

"slocate" – Uncheck this item; there is no immediate need for this utility, we can use "find".

Make sure everything else is unchecked.

Internet:

"finger" – Uncheck this item; we are not going to need this utility.

"lokkit" – Uncheck this item; this is a utility that provides user firewalling.

"mtr" – Uncheck this item; this item provides network diagnostic tool, useful for "telnet" (which we won't use it) and for GTK+ (which we won't be using X windows).

"rsh" – Uncheck this item; we are not going to be using "rsh", but "ssh" instead.

"talk" – Uncheck this item; this will provide one-to-one Internet talk.

"Telnet" – Uncheck this item; we will be using "ssh" for remote connections.

"whois" – Uncheck this item; this utility will provide the same style of queries as "finger".

"wget" – Uncheck this item; we don't need this utility on our ftp server.

"ftp" – Keep checked; this server will be an ftp server, so we need it.

"htmlview" – Keep checked; this is great to see files with html format.

"lftp" – Keep checked; this is a powerful utility that combines ftp & http file transfer.

"mailx" – Keep checked; this package will install the /bin/mail program.

"openssh" – Keep checked; this will install the core files for your "ssh".

"openssh-clients" – Keep checked; this will be necessary along with the above package.
"rsync" – Keep checked; this package provides you the ability to sync across the network.
"stunnel" – Keep checked; this utility will is a socked wrapper with SSL capabilities.
"tcpdump" – Keep checked; this utility is very useful to analyze network traffic among other things.
"traceroute" – Keep checked; this utility is good for troubleshooting.

Make sure everything else is unchecked.

Multimedia:

Make sure everything is unchecked.

Office:

Make sure everything is unchecked.

Productivity:

Make sure everything is unchecked.

Publishing:

"gruff" – Uncheck this item; this is a formatting system.

Make sure everything else is unchecked.

System:

"bind-utils" – Uncheck this item; this is some utilities for BIND.
"dosfstools" – Uncheck this item; this will give you DOS tools.
"irda-utils" – Uncheck this item; this is for infrared data association.
"isdn4k-utils" – Uncheck this item; this is tool for ISDN, which we are not going to be using.
"mtools" – Uncheck this item; this is a MS-DOS tool.
"net-snmp-utils" – Uncheck this item; this is for NET-SNMP management.
"parted" – Uncheck this item; we won't need all its capabilities and the ones we need it, we will use "dd" for it.
"rdate" – Uncheck this item; we will synchronize the clock through NTP.
"rdist" – Uncheck this item; we won't need its capabilities.
"time" – Uncheck this item; we won't be running "time".
"timeconfig" – Uncheck this item; we won't need this utility.
"ethtool" – Keep checked; utility for NIC.
"fbset" – Keep checked; utility to maintain frame buffer resolution.
"gnupg" – Keep checked; this is an encryption utility.
"logwatch" – Keep checked; this will give us the ability to monitor system logs.
"mt-st" – Keep checked; this provides tape utilities.
"netconfig" – Keep checked; text based tool for configuring NIC.
"pciutils" – Keep checked; this package is for PCI utilities on your system.
"setuptool" – Keep checked; gives you text mode menu for all configuration programs.
"statserial" – Keep checked; this will help to troubleshoot serial connections.
"sudo" – Keep checked; we will be using its capabilities.

"syslinux" – Keep checked; this is a boot loader for DOS floppies.

Make sure everything else is unchecked.

Text:

"aspell" – Uncheck this item; this is a spell checker.

"dos2unix" – Uncheck this item; this is a file conversion tool.

"pspell" – Uncheck this item; this is a library for any spell checker.

"unix2dos" – Uncheck this item; this is a file conversion tool.

Make sure everything else is unchecked.

Utilities:

Make sure everything is unchecked.

Development

Debuggers:

"lsof" – Keep checked; this is a good tool for forensic analysis.

Make sure everything else is unchecked.

Languages:

"perl" – Keep checked; this is a good programming language to use.

"python" – Keep checked; this is a good programming language also.

Make sure everything else is unchecked.

Libraries:

"perl-Filter" – Keep checked; we are going to need for Perl.

"pyOpenSSL" – Keep checked; we are going to need for python.

"python-optik" – Keep checked; we are going to need for python.

"rhnlib" – Keep checked; this file contains python libraries.

"rpm-python" – Keep checked;

Make sure everything else is unchecked.

System:

Make sure everything is unchecked.

Tools:

"make" – Keep checked; this utility allows users to build and install packages.

NOTE: this package can be removed from the server before it goes to production, which is my recommendation, for security reasons. If you have to install a software that requires it, just add this package back in to the system.

Make sure everything else is unchecked.

Documentation:

"man-pages" – Keep checked; this is good reference material when performing your duties.

"specspo" – Uncheck this item; this is used to internationalize packages.

Make sure everything else is unchecked.

Networking

Mail:

Make sure everything is unchecked.

Utilities:

Make sure everything is unchecked.

System Environment

Base:

"anacron" – Uncheck this item; this is a command scheduler.
"dhclient" – Uncheck this item; this is the DHCP client.
"mailcap" – Uncheck this item; this is used by the metamail program.
"nss_ldap" – Uncheck this item; we are not using LDAP.
"reiserfs-utils" – Uncheck this item; we are not using this filesystem.
"tmpwatch" – Uncheck this item; we are not going to use this utility.
"vixie-cron" – Uncheck this item; we are going to use the regular crontab.
"wireless-tools" – Uncheck this item; we won't use wireless.
"yp-tools" – Uncheck this item; we are not using NIS.
"acl" – Keep checked; it will provide you with acl capabilities.
"attr" – Keep checked; this provides tools to manipulate attributes.
"crontabs" – Keep checked;
"eject" – Keep checked; this will allow you to eject removable media.
"iptables" – Keep checked; this will be how we will secure our system.
"jfsutils" – Keep checked; this will allow you to check journal filesystem.
"kbdconfig" – Keep checked; provide you with keyboard mapping settings.
"logrotate" – Keep checked; simplify the administration of logs.
"man" – Keep checked; here you will find "man" tools.
"mkbootdisk" – Keep checked; allows you to create a standalone boot floppy.
"ntsysv" – Keep checked; allows another way to manage the runlevels.
"pam_krb5" – Keep checked; allows you to use PAM.
"pam_smb" – Keep checked; provides you to use external SMB server.
"pinfo" – Keep checked; information file viewer.
"quota" – Keep checked; administration tool.
"up2date" – Keep checked; update agent.
"utempter" – Keep checked; allow non-privileged program to have root.

Make sure everything else is unchecked.

Daemons:

"apmd" – Uncheck this item; power management for laptops.
"esound" – Uncheck this item; we are not using sound.
"net-snmp" – Uncheck this item; we are not using NET-SNMP.
"nfs-utils" – Uncheck this item; we are not using NFS.
"nscd" – Uncheck this item; we are not using NIS + with DNS.
"portmap" – Uncheck this item; there will not be NIS or NFS on the server.
"ppp" – Uncheck this item; we will not be using this protocol.
"procmail" – Uncheck this item; we won't be using this mail client.
"rp-pppoe" – Uncheck this item; this is also a PPP support item.
"wvdial" – Uncheck this item; this looks for modem and we don't have one.
"ypbind" – Uncheck this item; we won't be using NIS.
"at" – Keep checked;
"autofs" – Keep checked; automount.
"cipe" – Keep checked; IP Encapsulation.
"gpm" – Keep checked; mouse support.

"openssh-server" – Keep checked; we are loading the client and server.
"Orbit" – Keep checked; it enables communications between programs and objects.
"sendmail" – Keep checked; this will be our mail client.
"tcp_wrappers" – Keep checked; we will use to monitor our programs.

Make sure everything else is unchecked.

Kernel:

"kernel-pcmcia-cs" – Keep checked;

Make sure everything else is unchecked.

Libraries:

"audiofile" – Uncheck this item;
"Cyrus-sasl-plain" – Uncheck this item;
"gnome-libs" – Uncheck this item; we are not using gnome.

Accept all other defaults.

Shells:

"tcsh" – Keep checked;

Make sure everything else is unchecked.

User Interface

Desktops:

"redhat-menus" – Uncheck this item; we are not using gnome or kde.

Make sure everything else is unchecked.

X:

Make sure everything is unchecked.

X Hardware Support:

Make sure everything is unchecked.

Utilities

System:

Make sure everything is unchecked.

At this point we have finish going through all individual packages, so as soon as, you click next, the system will check for the dependencies.

Click "next";

The choices we made, the only dependencies were with "sendmail", mkbootdisk", "htmlview" and "man".
At the bottom of the screen, select "Install packages to satisfy dependencies" and click "next".

The next screen to appear is the "about to install" screen.

Click "next" to install;

You will see the installing packages screen and at this point you have to wait, but don't go away you might be asked to change CDs.

After the installation is complete you will have a chance to create a boot CD, it is my recommendation that you do so.

Click “next”;

You will get to the “Congratulations” screen, where it will ask you to remove all the media and click on “EXIT”. This will reboot your system to your new installed OS.

After the installation is complete, Red Hat leaves a file with all your configurations that were performed during installation; it’s called and located at:

```
[root@pluto root]# ls
anaconda-ks.cfg  install.log  install.log.syslog
```

It would be in your best interest to backup this file for future use in a safe place where it won’t get overwritten. Also at this point it would be recommended to make a list/copy of all your original RPM on your system, so to perform such task, please do the following:

```
rpm -qa | sort >/root/Orig_RPM_list
```

This will create a RPM list like this excerpt bellow:

```
[root@pluto root]# more Orig_RPM_list
acl-2.0.11-2
ash-0.3.8-5
at-3.1.8-31
attr-2.0.8-3
authconfig-4.2.12-3
autofs-3.1.7-33
basesystem-8.0-1
bash-2.05b-5
bdflush-1.5-21
bzip2-1.0.2-5
bzip2-libs-1.0.2-5
chkconfig-1.3.6-3
cipe-1.4.5-11
comps-8.0-0.20020910
cpio-2.4.2-28
```

PATCHES

Now that we have our Operating System up and running, we need to install the most up to date patches for our system, so we go to the source to get it. Remember, not all patches will apply to your system, since we did not install most of the RPM’s.

You can got to ftp://updates.redhat.com/ and navigate your way to your correct system, which for this system will be ftp://updates.redhat.com/8.0/en/os/i686/, where here I'll find several patches for my system, here are some examples of the patches you might find:

```
04/10/2003 09:07PM      3,967,254 glibc-2.3.2-4.80.6.i686.rpm
04/10/2003 09:08PM      9,313,016 glibc-debug-2.3.2-4.80.6.i686.rpm
05/13/2003 10:58PM     13,988,231 kernel-2.4.20-13.8.i686.rpm
05/13/2003 10:58PM     14,541,711 kernel-bigmem-2.4.20-13.8.i686.rpm
03/14/2003 11:21PM     13,864,655 kernel-debug-2.4.18-27.8.0.i686.rpm
05/13/2003 10:58PM     14,536,519 kernel-smp-2.4.20-13.8.i686.rpm
12/19/2002 01:03AM     24,721,979 kernel-uml-2.4.18-19.8.0.i686.rpm
03/26/2003 07:23PM      1,422,394 openssl-0.9.6b-33.i686.rpm
```

Now, you need to install all the RPM's that your system requires and not the other ones, and the best way to do that, will be to install with the command (`rpm -F`).

The command `rpm -F` is preferred because it will only install an RPM if there is an older version of that RPM already installed on the system, and that's what we are looking for, just to install the RPM's we have on the system.

NOTE: Make sure to install the RPM's for the kernel with the following command:

- "rpm –i" – This will make sure the old version is not deleted or overwritten and since Red Hat installed kernel with its version number on the directory name, we can have several versions on the same system.

You will choose which versions to boot from through the "`grup.conf`" file, where you can specify which kernel version you want to boot from it. This will give you more ways to fall back in case the RPM doesn't go well. After the new kernel is tested and you feel that you no longer need the old kernel, you can deleted the old kernel with the "`rpm -e`" command.

We will be automating all the patch process through the "`up2date`" utility, which is a Red Hat native tool.

Services

The next step after installing all the patches, which, by the way, protects you from known vulnerabilities, it is to stop unnecessary services that are running on the system, which will help you to stop undiscovered exploits.

Here is how my system looks like after the initial install:

```
# ps -ef | grep -v \\\[
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 08:53 ?        00:00:04 init
```

```
root       448     1  0 08:54 ?        00:00:00 syslogd -m 0
root       452     1  0 08:54 ?        00:00:00 klogd -x
root       508     1  0 08:54 ?        00:00:00 /sbin/cardmgr
root       573     1  0 08:54 ?        00:00:00 /usr/sbin/sshd
root       596     1  0 08:54 ?        00:00:00 sendmail:
accepting connections
smmsp      606     1  0 08:54 ?        00:00:00 sendmail: Queue
runner@01:00:00
root       616     1  0 08:54 ?        00:00:00 gpm -t ps/2 -m
/dev/mouse
daemon     625     1  0 08:54 ?        00:00:00 /usr/sbin/atd
root       634     1  0 08:54 ?        00:00:00 login - root
root       635     1  0 08:54 tty2     00:00:00 /sbin/mingetty
tty2
root       636     1  0 08:54 tty3     00:00:00 /sbin/mingetty
tty3
root       637     1  0 08:54 tty4     00:00:00 /sbin/mingetty
tty4
root       638     1  0 08:54 tty5     00:00:00 /sbin/mingetty
tty5
root       639     1  0 08:54 tty6     00:00:00 /sbin/mingetty
tty6
root       642   634  0 08:54 tty1     00:00:00 -bash
root       842   573  0 10:14 ?        00:00:00 /usr/sbin/sshd
root       844   842  0 10:14 pts/0    00:00:00 -bash
root       942   844  0 10:35 pts/0    00:00:00 ps -ef
```

The reason for the above command "`ps -ef | grep -v \\\[`" is for me to eliminate the processes like "[kswapd]", "[kjournald]", which are not processes form actual programs, but from bits of kernel code.

As you can see, there are not many process running on the system as it is, but you can see a couple that don't need to be there, such as, "`/sbin/mingetty tty*`", these entries are not necessary for this system. Another very important one would be "`sendmail: accepting connections`", we definitely don't want this process listening on any port.

Here is how we can eliminate these issues:

- Disabling the extra Daemons;

```
# vi /etc/inittab
# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
#2:2345:respawn:/sbin/mingetty tty2
#3:2345:respawn:/sbin/mingetty tty3
#4:2345:respawn:/sbin/mingetty tty4
#5:2345:respawn:/sbin/mingetty tty5
#6:2345:respawn:/sbin/mingetty tty6
```

The above is part of the "`inittab`" file were I commented it out the extra login daemons. Now to make this changes take effect please perform the following:

```
# /sbin/init q
```

- Disabling "Sendmail" from listening;

Prior versions of Red Hat and Sendmail, we had to stop the MTA* from listening on port 25 (smtp port), but on this version, Red Hat 8, with Sendmail 8.12.5-7, as you can see:

```
$ rpm -qa | grep sendmail
sendmail-8.12.5-7
```

We don't have to do this task anymore, because Sendmail has a statement on its sendmail.cf file, see below:
This file is located at `/etc/mail/sendmail.cf`

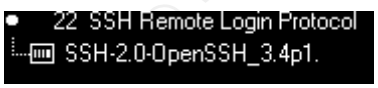
```
# SMTP daemon options

O DaemonPortOptions=Port=smtp,Addr=127.0.0.1, Name=MTA
```

As you can see the MTA is pointing to the loop back address, but if you still skeptical, here is the test I performed.

This is what I saw when I did a "`netstat -a`":

```
$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
State
tcp        0      0 *:ssh       *:*         LISTEN
tcp        0      0 pluto:smtp  *:*         LISTEN
```

Well, it seems that I have smtp listening on port 25, but this is what I got from an external scan:



It was used "SuperScan" to run this test.†

So, even if you see the port listening, on this particular case, it is not listening for external requests, but to further secure our environment we will disable this service from listening on the loop back port, because if an attacker breaks in to

* MTA: Mail Transport Agent
† Superscan can be found and downloaded at http://www.webattack.com/get/superscan.shtml - It is a freeware.

your system, the attacker can attack a sendmail vulnerability to gain root on your system.

There is no easy way to solve this problem on this version of sendmail, the only way will be to hack the boot script provided by sendmail. Here how it's done:

Edit /etc/init.d/sendmail and comment out the following line:

```
daemon /usr/sbin/sendmail -bd \ (place a # in the beginning of the line)
```

Restart the daemon:

```
# ./sendmail restart
```

Perform a "ps -ef", you will see only the queue daemon running:

```
smmsp     2130     1  0 17:18 ?        00:00:00 sendmail:
Queue runner@01:00:00
```

Perform a "netstat -a" and you won't see the smtp daemon listening on port 25:

```
tcp        0      0 *:ftp     *:*                     LISTEN
tcp        0      0 *:ssh     *:*                     LISTEN
```

These steps will further secure your environment against attacker that break in to your system, through other means and avoiding them to gain root through sendmail.

Now we have a much cleaner system, as you can see:

```
# ps -ef | grep -v \\\[
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 20:28 ?        00:00:04 init
root       457     1  0 20:28 ?        00:00:00 syslogd -m 0
root       461     1  0 20:28 ?        00:00:00 klogd -x
root       517     1  0 20:28 ?        00:00:00 /sbin/cardmgr
root       582     1  0 20:29 ?        00:00:00 /usr/sbin/sshd
root       605     1  0 20:29 ?        00:00:00 sendmail:
accepting connections
smmsp      615     1  0 20:29 ?        00:00:00 sendmail: Queue
runner@01:00:00
root       625     1  0 20:29 ?        00:00:00 gpm -t ps/2 -m
/dev/mouse
daemon     634     1  0 20:29 ?        00:00:00 /usr/sbin/atd
root       643     1  0 20:29 ?        00:00:00 login - root
root       646   643  0 20:29 tty1     00:00:00 -bash
root       810   582  0 20:47 ?        00:00:00 /usr/sbin/sshd
bsouza     812   810  0 20:47 ?        00:00:00 /usr/sbin/sshd
bsouza     813   812  0 20:47 pts/0    00:00:00 -bash
root       882   813  0 21:27 pts/0    00:00:00 su -
root       883   882  0 21:27 pts/0    00:00:00 -bash
root      1060   883  0 22:17 pts/0    00:00:00 ps -ef
```

SSH

The SSH program will be the one we will be using to remote login to our server, but certain default settings we might want to change, so we nee to take a closer look at it. The configuration file we need to look is located at "/etc/ssh/sshd_config", one default that we need to change is the "`PermitRootLogin`", which in RedHat is set to `yes`, and that is not a good idea if you want to keep track of "root" usage.

We will allow root access only by the usage of "su" or sudo, so this way we will have a better control over the "root" user.

Let's take a look on the /etc/ssh/sshd_config file:

```
# cd /etc/ssh
# vi sshd_config

# Authentication:

#LoginGraceTime 600
PermitRootLogin no  (change yes to no and remove comment)
#StrictModes yes

Protocol 2     (protocol make sure you set to 2 ONLY)
IgnoreRhosts yes    (uncomment this line)
RhostsRSAAuthentication no    (uncomment this line)
RhostsAuthentication no  (uncomment this line)
HostbasedAuthentication no    (uncomment this line)
```

The above was set to prevent:

- `PermitRootLogin` was set to "no", so we can force the user to login with his/her login ID and "su" to root or use "sudo" to gain root privileges.
- `Protocol 2` was set, so we don't fall back to "protocol 1", which is known to have vulnerabilities.
- `IgnoreRhosts` was set to "yes", so we can disable the use of .rhosts completely.
- `RhostsRSAAuthentication, RhistsAuthentication, HostbasedAuthentication` this is to further disable the authentication through .rhosts.

Security Kernel Settings

There is several kernel setting, but we will discuss and set some "security" kernel settings, which will address system and network resources that will help us prevent some "denial-of-service" attacks, buffer overflow attacks, etc …

Generally these settings are used to harden your system, but sometimes it might come to an inconvenience to others, for example, you want to limit "core dumps", due to its inherited problem of being "world readable", while some other members of your company might find that useful for troubleshooting, so you have to use common sense to resolve these issues.

The following are changes that we are going to perform on our system.

"`/etc/security/limits.conf`" – we will make a change on this file to prevent core dump, see below:

```
# /etc/security/limits.conf
#
#Each line describes a limit for a user in the form:
#
#<domain>        <type>  <item>  <value>

*               hard    core            0
```

To further improve our security settings we will make changes to `/etc/sysctl.conf` file, which we will set restrictions on "ip_forward", "log_martians"*, redirects, etc … See the changes below.

Here is our file before the alterations:

```
# more /etc/sysctl.conf
# Kernel sysctl configuration file for Red Hat Linux
#
# For binary values, 0 is disabled, 1 is enabled.  See
sysctl(8) and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 0
```

* Log Martians: logs packets with weird source addresses, which is an indication of "spoof attempts".

```
# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Controls the System Request debugging functionality of
the kernel
kernel.sysrq = 0

# Controls whether core dumps will append the PID to the
core filename.
# Useful for debugging multi-threaded applications.
kernel.core_uses_pid = 1
```

Now, lets make this more secure.

- We will add the capability to log all "weird" source address packets, which is an obvious attempt of spoofing, by adding the following line.

```
net.ipv4.conf.all.log_martians = 1
```

- We will add the capability of filtering by reverse path, to ensure the re-route will occur on the correct interface.

```
net.ipv4.conf.all.rp_filter = 1
```

- We also will turn off the following IP redirects from our system, so we don't start redirecting IP traffic.

```
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.tcp_syncookies = 1
```

- Also limit the `tcp_max_syn_backlog` to avoid any "syn flood attack" to your system.

```
net.ipv4.tcp_max_syn_backlog = 4096
```

Here is how our file looks like now:

```
# Kernel sysctl configuration file for Red Hat Linux
#
# For binary values, 0 is disabled, 1 is enabled.  See
sysctl(8) and
# sysctl.conf(5) for more details.
```

```
# Controls IP packet forwarding
net.ipv4.ip_forward = 0
net.ipv4.tcp_max_syn_backlog = 4096
# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Controls the System Request debugging functionality of
the kernel
kernel.sysrq = 0

# Controls whether core dumps will append the PID to the
core filename.
# Useful for debugging multi-threaded applications.
kernel.core_uses_pid = 1

# Interface-specific parameters
net.ipv4.conf.all.log_martians =1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.tcp_syncookies = 1
```

Here are some sample messages from the net.ipv4.conf.all.log_martians, this is an extract from `/var/log/messages`:

```
May 28 04:15:31 pluto kernel: martian source
255.255.255.255 from 111.22.333.444
, on dev eth0
May 28 04:15:31 pluto kernel: ll header:
ff:ff:ff:ff:ff:ff:00:03:77:42:1z.d2:08:
00
May 28 04:15:31 pluto kernel: martian source
255.255.255.255 from 111.22.333.444
, on dev eth0
May 28 04:15:31 pluto kernel: ll header:
ff:ff:ff:ff:ff:ff: 00:03:77:42:1z.d2:08:
00
May 28 04:15:31 pluto kernel: martian source
255.255.255.255 from 111.22.333.444
, on dev eth0
May 28 04:15:31 pluto kernel: ll header:
ff:ff:ff:ff:ff:ff: 00:03:77:42:1z.d2:08:
00
```

```
May 28 04:15:31 pluto kernel: martian source
255.255.255.255 from 111.22.333.444
, on dev eth0
May 28 04:15:31 pluto kernel: ll header:
ff:ff:ff:ff:ff:ff: 00:03:77:42:1z.d2:08:
00
May 28 04:15:31 pluto kernel: martian source
255.255.255.255 from 111.22.333.444
, on dev eth0
```

NOTE: All entries in this log were altered to hide identity.

Setting Up iptables

We will secure our server by creating an iptable, there are only two ports on our server listening, as you can see:

```
[root@pluto root]# netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
State
tcp        0      0 *:ftp    *:*                     LISTEN
tcp        0      0 *:ssh    *:*                     LISTEN
tcp        0    232 172.16.8.92:ssh         L22137:3027
ESTABLISHED
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node
Path
unix  2      [ ACC ]     STREAM     LISTENING     1538
/dev/gpmctl
unix  6      [ ]         DGRAM                    1221
/dev/log
unix  3      [ ]         STREAM     CONNECTED     1674
unix  3      [ ]         STREAM     CONNECTED     1673
unix  2      [ ]         DGRAM                    1505
unix  2      [ ]         DGRAM                    1454
unix  2      [ ]         DGRAM                    1333
unix  2      [ ]         DGRAM                    1236
```

What we will accomplish with the iptable is to block any outside traffic for port 22 our SSH port and make sure we accept all communications to port 21 FTP port. We will perform the following commands:

```
/sbin/iptables -A INPUT -d 192.168.110.114 -p tcp --dport
21 -j ACCEPT
```

```
/sbin/iptables -A INPUT -d 192.168.110.114 -p tcp --dport
22 -j DROP
```

The above commands will create the rule, but now we need to save this to a file, so we will perform the following command:

```
/sbin/service iptables save
```

This will save the rules to a file located at /etc/sysconfig/iptables, the next step will be to ensure this rule is in effect in all the run levels, by performing the following command:

```
/sbin/chkconfig –level 2345 iptables on
```

To verify the following, see below:

```
# /sbin/chkconfig --list iptables
iptables        0:off   1:off   2:on    3:on    4:on
5:on    6:off
```

As you can see, the rule is now in effect on run levels 2, 3, 4 and 5. Here is now the rule looks like:

```
# /sbin/iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             192.168.110.114
tcp dpt:ftp
DROP       tcp  --  anywhere             192.168.110.114
tcp dpt:ssh
```

The command breaks down like this:

/sbin/iptables –A (append) <INPUT packates> -d <destination host> -p <protocol> --dport <port the packets are coming to> -j <what do with packates>

File System Secure Settings

In this section we will cover how to protect our file system from unwanted changes, for example, when an attacker gain access to your system, they usually will put a “rootkit” on your system to cover their tracks, while their work and plant back-doors for easy access later. Since most of the binaries are located at /usr, so this directory is one of the primary directories for the attacker.

We can secure this directory by setting as `ro` (read only), but we need to use our judgment, since this directory will be used when you install a new product or upgrade any product on your system, but in our case, Linux can toggle back in forth with the mount permission, making this decision very easy.

Not everything is straight forward as it seems, take the "/" file system for example, we can not properly secure this file system. The following are the settings we can set to secure the file system:

`ro` – This option will set the file system read only.
`nosuid` – This option won't allow you to run set-UID programs on the file system.
`nodev` – This option don't allow any device file to work in the file system.
`noexec` – This option completely disable any executable in the file system.

Here are some problems we will encounter with the root system on our system, if you decide to secure it with any of the above options.

We cannot use `nosuid`, because Red Hat places `/bin/su` and other important set-UID programs under the / file system, while other OS place this file under `/usr/bin/su`, making it possible to lock it under a "`ro`" file system. Our objective is to secure all the binaries and libraries, so it cannot be used by any automated program to execute arbitrary commands.

We also cannot use "`nodev`" on the / file system, because we have `/dev`, and how good it is the system, if you cannot use any device. The option of "`noexec`", is definitely out of question, because you need to use executables under / file system.

Well, after all this explanation, what we will do is to, set "read only" on `/usr`, `/usr/local`, we also will set `/home`, as `nosuid`, `nodev`, this way we can avoid any user trying to run a set-UID program from their home directory, also since our `/tmp` directory is a separate file system we will set as `noexec, nosuid, nodev`. Everything else we will leave as default.

We will make the necessary changes by performing the following command:

`mount -o remount ro /usr` – this will turn `/usr` to read only.

Perform the above command to all the file system to achieve our objective. Remember we need to edit the `/etc/fstab`, so these changes can come back after a reboot, otherwise we will loose all the changes we just made to our file system.

You may also have notice an entry for the CDROM on the `/etc/fstab`, that is for the automounter, this feature was designed, so users don't have to run the mount command, besides to mount the CDROM, you need root access, so the automounter feature, solved that problem, but if you have a server in a public access place you might want to disable this feature, since people can place a CD with an set-UID program to gain root on your system.

Our system is in a secure facility with limited access through a badgering system, so we will not bother with this setting, but if you want to set you can delete all entries that refer to the CDROM and/or floppy under `/etc/security/console.perms`.

But what we'll do is to set our CDROM to `nosuid` and `nodev`.

System Log Settings

Red Hat logs all the messages going to "`authpriv`" facility, which are "private" messages that are available to root are logged in `/var/log/secure` file, but what the system doesn't do by default is to log messages sent to "`auth`" facility, so we need to make this alteration under `/etc/syslog.conf`, this way we can ensure that we log all the messages coming to "`auth`".

This is how the file looks like right now:

```
# The authpriv file has restricted access.
authpriv.*
/var/log/secure
```

This is how the file looks after our alteration:

```
# The authpriv file has restricted access.
auth,authpriv.*
/var/log/secure
```

I will be also setting up the kernel logging, because, we will have several messages from the kernel. To do that we will add the following lines:

```
# Kernel logging
kern.*;*.=crit                                  /dev/console
kern.*;*.=crit                                  root
```

I need also be notified of any emergency alerts, so for that I will add the following line.

```
# Emergency messages to a specific address
```

```
*.emerg                                          bsouza@abcd.com
```

NOTE: I will not be setting up “system accounting”, I will not be using or needing this data.

Access Control

As a security professional you should be aware and understand the threats, vulnerabilities, and risks associated with networked computers. Access control should be well implemented to avoid any and all the above and also to further improve the confidentiality of your system, by just allowing access to the appropriate personnel.

Under normal circumstances you should never allow the user “root” to remote login, or any other shared or system account for that matter. Any root access has to be obtained by logging in to the server under an unprivileged account and then gain root, by “su” or “sudo”.

We can start checking if `/etc/securetty` has the correct content and permissions, on Red Hat 8, it already has a restricted access by default, which looks like this:

```
-rw-------    1 root     root    114 Jun 13  2001 securetty
```

Permissions set to 600, you can further lock this with a permission of 400, but since the group and user permissions are set to root, we going to leave as it is on this file.

You should see the following on your `/etc/securetty` file:

```
tty1
```

This will guarantee that “root” will log only on the console. Remember that we set earlier on SSH, that root cannot login remotely, so this will further improve your security.

Red Hat has a function, where CTRL-ALT-Delete, key combination will shutdown the server, which is exactly what we don’t want, so to disable this function, we will have to edit the `/etc/inittab` file and comment out this particular line.

```
# Trap CTRL-ALT-DELETE
#ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

User Account Clean Up

UNIX System are know for their backwards comparability and historical reasons, so you'll find several accounts that you can delete, from the server, such as, "`games`", "`uucp`", etc… By deleting these accounts you will have a cleaner `/etc/passwd` file, that will be easier to maintain, accountability and security.

However, there are accounts that programs use, so if you remove them, these programs will not function correctly, as an example, the `bin` account, if removed several programs that need these binaries will die.

Here are the files that we will remove from our FTP server:

```
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
```

NOTE: before deleting these accounts, we will make a backup copy of `passwd, shadow, group` files.

We will use the /usr/sbin/userdel command to perform the account clean up and use /usr/sbin/groupdel command to clean up the groups these accounts belong.

Account and Password Aging

Red Hat by default doesn't expire accounts and doesn't set any limits for how long to keep the password before changing it, as you can see below:

```
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7
```

Here we will perform the following changes:

```
PASS_MAX_DAYS   99999 -> PASS_MAX_DAYS  45
PASS_MIN_DAYS   0   ->   PASS_MIN_DAYS  5
PASS_MIN_LEN    5   ->   PASS_MIN_LEN   8
```

We will set for all users created on this system a password aging to 45 days and the users won't be able to change it for the following 5 days and the passwords will have to a minimal of 8 character.

Now, we need to take care of any users that were created on this system before this rule was placed, which in this case, the only one will be myself, so here is an example of how you would address this issue.

Here is my account on `/etc/shadow` before:

```
bsouza:$1$Èbhñùe8p$g88ym2rO2EMGCzyuhMeMX0:12187:0:99999:7::
:
```

I will perform the following command:

```
# chage -M 45 -m 5 bsouza
```

Here is my account on `/etc/shadow` after the command execution:

```
bsouza:$1$Èbhñùe8p$g88ym2rO2EMGCzyuhMeMX0:12187:5:45:7:::
```

My account now will expire in 45 days, as any other that will be created in this system.

The ftpusers File

It is always a good practice to create the file `/etc/ftpusers`, even if you are not running `ftp` on your system, but in our case, since our server is an ftp server, it is must that we create this file and place all system accounts in it, specially the "root" account.

The "ftpusers" file it is a place where we place all users that are not allowed to ftp in our system, contrary that it may lead you. The root account needs to be always there, as for any other system user in our system, except the "ftp" user, since this will be an anonymous ftp server. All other regular users will be able to use the ftp functions of this server.

Here is our `/etc/ftpusers` file on our system:

```
# more /etc/ftpusers
root
bin
daemon
adm
lp
sync
shutdown
```

```
halt
mailx
news
nobody
vcsa
sshd
rpm
mailnull
smmsp
pcap
```

As you can see, the `ftp` user and my account are not here, allowing me to ftp to this server.

Single User Mode Password

We are going to add a single user password to our server, so when rebooting the server to single user mode you need to know the root password to get it. The process of adding this feature id the following:

- Add this line to the `/etc/inittab` file.

```
# Single User Mode.
sum:S:wait:/sbin/sulogin
```

Now, we have this extra layer of protection on our system.

Banners

We will be placing warning banners to anyone that logs in to our system, notifying them, that their behavior is being monitor. We will use a warning banner that was developed by the DOJ. Please, consult your legal department before placing, such banners.

Our banner will read:

```
This system is for the use of authorized users only.
Individuals using this computer system without authority,
or in excess of their
authority, are subject to having all of their activities on
this system\
monitored and recorded by system personnel.

In the course of monitoring individual improperly using
this system, or in the\
```

```
course of system maintenance, the activities of authorized
users may also be\
monitored.

Anyone using this system expressly consents to such
monitoring and is advised\
that if such monitoring reveals possible evidence of
criminal activity, system\
personnel may provide the evidence of such monitoring to
law enforcement\
officials.
```

Since we are using SSH for remote login, we need to create a directory and place a file with the above text, so on our server will be;

```
/etc/security/banner/warning_banner
```

After creating this file, we need to point our SSH configuration file to it and that is done on `/etc/ssh/sshd_config` file as shown below.

```
Banner /etc/security/banner/warning_banner
```

Now we need to restart the SSH daemon, so our changes can take effect and that is done in the following manner:

```
# cd /etc/init.d
# ./sshd restart
Stopping sshd:                                     [  OK  ]
Starting sshd:                                     [  OK  ]
```

Next time you remote login to the system, you will see our warning message.

Services Running at Boot Up

You can check what services suppose to be running on your system when it boots up by running the command "`/sbin/chkconfig –list`". Here is what is currently running on our system during boot up and which run level they suppose to run:

```
# /sbin/chkconfig –list
syslog          0:off   1:off   2:on    3:on    4:on
5:on    6:off
netfs           0:off   1:off   2:off   3:on    4:on
5:on    6:off
network         0:off   1:off   2:on    3:on    4:on
5:on    6:off
```

```
random          0:off   1:off   2:on    3:on    4:on
5:on    6:off
rawdevices      0:off   1:off   2:off   3:on    4:on
5:on    6:off
saslauthd       0:off   1:off   2:off   3:off   4:off
5:off   6:off
atd             0:off   1:off   2:off   3:on    4:on
5:on    6:off
gpm             0:off   1:off   2:on    3:on    4:on
5:on    6:off
autofs          0:off   1:off   2:off   3:on    4:on
5:on    6:off
keytable        0:off   1:on    2:on    3:on    4:on
5:on    6:off
kudzu           0:off   1:off   2:off   3:on    4:on
5:on    6:off
sshd            0:off   1:off   2:on    3:on    4:on
5:on    6:off
sendmail        0:off   1:off   2:on    3:on    4:on
5:on    6:off
iptables        0:off   1:off   2:on    3:on    4:on
5:on    6:off
rhnsd           0:off   1:off   2:off   3:on    4:on
5:on    6:off
pcmcia          0:off   1:off   2:on    3:on    4:on
5:on    6:off
```

Now, we will install our software, such as, TCP Wrappers, our ftp client (VSFTPD), etc. This will cause more services/processes to run and maybe some packages need to be installed to support the software, so we will have to re-evaluate the system again to further secure it.

Installing Vsftpd

Download your package; remember to always check for checksum and signature, where available. Now, remember, our filesystem is read only, so we will have to change that before we can install anything. Here is what we will do:

```
# mount –o remount rw /usr
# mount –o remount rw /usr/local
```

Place the vsftpd file on `/usr/local` directory and then:

```
# gunzip vsftpd-1.1.3.tar.gz
# tar –xvf vsftpd-1.1.3.tar
# cd vsftpd-1.1.3
```

At this point it is always recommended to read the README file and in this case also the INSTALL file under this directory, which will give us all the information to get this service installed, configure and working.

Next step after reading the file for us will be to run "make" on the directory, by doing the following:

```
# make
```

Well, on our system we will have some issues to address, since our installation was at a bare minimum, we will need to install some rpm to satisfy our installation and they are the following:

```
cpp-3.2-7
binutils-2.13.90.0.2-2
glibc-devel-2.2.93-5 (which also has a dependency)
glibc-kernheaders-2.4-7.20 (the above dependency)
```

After installing the above you can run make to prepare and install everything. You can verify this by doing the following:

```
# ls -l vsftpd
-rwxr-xr-x    1 root     root    65964 Jun  5 05:54 vsftpd
```

Other things you need to verify before running it are:

VSFTPD needs the user "nobody" on the default configuration.
VSFTPD needs the (empty) directory /etc/share/empty on the default configuration.
In case of "anonymous" ftp server, which is our case, we will need the "ftp" user with a valid home directory under /ftpdir/ftp.
Install all the config files, man pages, etc… by running "make install".
Copy the sample config file. i.e. cp vsftpd.conf /etc
On the config file set at the bottom this line:

```
listen=YES
```

Now, we need to start the service to test our installation, so we will run the following command:

```
# /usr/local/sbin/vsftpd &
```

NOTE: All this information and more is available on the INSTALL file, please read this file.

Running VSFTPD on xinetd Secure

You also, can find this information on `/usr/local/vsftpd-1.1.3/EXAMPLE/INTERNET_SITE` directory under the `README` file.

We will add more setting to the configuration file on `/etc/xinetd.d/vsftpd` this will improve security and performance.

Here is the file:

```
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/local/sbin/vsftpd
#       server_args             =
#       log_on_success          += DURATION USERID
#       log_on_failure          += USERID
        nice                    = 10
        disable                 = no
```

The above comes as default, but now we will improve upon this file. We will limit the amount of connection per IP address and the maximum of concurrent connections by adding the following to the file:

```
        per_sourc               = 5
        server                  = 200
```

In case we exceed our 200 concurrent connection we need to tell our customers what is happening, so we setup a banner to let them know and for this we will do the following:

```
        banner_fail             = /etc/vsftpd.busy_banner
```

The following two lines we will uncomment from the original file, this will allow us to log all information on log on failure or success under `/var/log/sercure` log file.

```
    log_on_success          += DURATION USERID
    log_on_failure          += USERID
```

Configuring vsftpd.conf File

Most of the file is already set by default, but the following we will verify or add to our /etc/vsftpd.conf file.

```
# Access rights
anonymous_enable=YES
local_enable=NO
write_enable=NO
anon_upload_enable=NO
anon_mkdir_write_enable=NO
anon_other_write_enable=NO
```

This makes sure the FTP server is in anonymous-only mode and that all write and upload permissions are disabled. Note that most of these settings are the same as the default values anyway – but where security is concerned, it is good to be clear.

```
# Security
anon_world_readable_only=YES
connect_from_port_20=YES
hide_ids=YES
pasv_min_port=50000
pasv_max_port=60000
```

These settings, in order
- Make sure only world-readable files and directories are served.
- Originates FTP port connections from a secure port – so users on the FTP server cannot try and fake file content.
- Hide the FTP server user IDs and just display “ftp” in directory listings. This is also a performance boost.
    226 Set a 50000-60000 port range for passive connections – may enable easier
firewall setup!

```
# Features
xferlog_enable=YES
ls_recurse_enable=NO
ascii_download_enable=NO
async_abor_enable=YES
```

In order,
- Enables recording of transfer stats to /var/log/vsftpd.log
- Disables “ls –R”, to prevent it being used as a DoS attack. Note – sites wanting to be copied via the “mirror” program might need to enable this.
    226 Disables downloading in ASCII mode, to prevent it being used as a DoS
attack (ASCII downloads are CPU heavy).
Enables older FTP clients to cancel in-progress transfers.

```
# Performance
one_process_model=YES
```

```
idle_session_timeout=120
data_connection_timeout=300
accept_timeout=60
connect_timeout=60
anon_max_rate=50000
```

In order,

226 Activates a faster “one process per connection” model. Note! To maintain
security, this feature is only available on systems with capabilities – e.g. Linux kernel 2.4.
- Boots off idle users after 2 minutes.
- Boots off idle downloads after 5 minutes.
- Boots off hung passive connects after 1 minute.
- Boots off hung active connects after 1 minute.
- Limits a single client to ~50kbytes / sec download speed.

You also want to set this option, so you can hide the FTP server version from the login prompt.

```
# You may fully  ustomize the login banner string:
ftpd_banner=Welcome to PLUTO FTP service.
```

When using this setting the FTP server version automatically is omitted.

Restart `xinetd`.

Your configuration is complete.

Testing the Configuration

We will run some test to see if the configuration is working as it suppose to, by performing several FTP commands.

- FTP from Windows computer (not localhost):

```
D:\Documents and Settings\souzaw>ftp 192.168.110.114
Connected to 192.168.110.114.
220 Welcome to PLUTO FTP service.
User (192.168.110.115☹none)): anonymous
331 Please specify the password.
Password:
230 Login successful. Have fun.
ftp>
```

We were able to login to the server now, we will test what we can do while logged in as anonymous.

- Attempt to delete files and directories:

```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
pub
226 Directory send OK.
ftp: 5 bytes received in 0.01Seconds 0.50Kbytes/sec.

ftp> rm pub
550 Permission denied.
ftp> rm –R pub
550 Permission denied.

ftp> cd pub
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
hello123
226 Directory send OK.
ftp: 10 bytes received in 0.00Seconds 10000.00Kbytes/sec.

ftp> rm hello123
550 Permission denied.
```

- Attempt to upload a file:

```
ftp> put upload_test.rtf
200 PORT command successful. Consider using PASV.
550 Permission denied.
```

- Attempt to download the file:

```
ftp> get hello123
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for hello123 (0
bytes).
226 File send OK.
```

Log Rotation

After all these logging on our system, we need to find a way to manage the quick growth of these log files. To address this issue we will be using a Red Hat facility called logrotate (/etc/logrotate.conf).

We will be running the logrotate script weekly to mitigate this issue of a growing log file. We will schedule the log rotation through "cron", but before we will need to create a cron file for "root", because by default this file does not exist, so you need to perform the following:

```
# mkdir /var/spool/cron
# chmod 700 /var/spool/cron
# cp /etc/crontab /var/spool/cron/root
```

Now, you have a "template of a cron file that looks like this:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

Now, we will edit this file, so we can run our logrotate script weekly, by doing the following:

```
# Log Rotation
22 4 * * 0 root log rotation /usr/sbin/logrotate
/etc/logrotate.conf
```


Secure by Designed

The vsftpd is an application secure by designed, here are some examples, also found on your vsftpd directory under the SECURITY directory.

- All parsing and acting on a possible malicious data is held by a process that runs as an unprivileged user and furthermore it runs on a "chroot" environment.
- All privilege operations are handle by the privilege parent process, that by designed is small for security reasons.
- This same privileged parent process makes use of capabilities and chroot(), to run with the least privilege required.

- VSFTPD uses its own "/bin/ls" program and libraries to avoid any calling or forking to any program that we cannot trust. All its program and utilities can be found in two places:
  "sysutil.c"
  "sysdeputil.c"
  This provides a convenient audit point for ascertaining which calls vsftpd trusts.
- The implementation is also conscious on coding errors to avoid buffer overflows by hiding the buffer handling code behind an API.

You can find a lot more information on this topic under `/usr/local/vsftpd-1.1.3/SECURITY`; because of this secure designed, light weight and fast handle on high traffic FTP servers, I chose to go with VSFTPD.

SUDO Configuration

As mentioned before SUDO will be used to address the less privilege issues on our system. As an example, if you have junior administrators creating user accounts and you don't want to give them the "root' password, you can set them with SUDO privileges.

In this system will be only myself to start, so I will show you how easy is to set up. The configuration file `/etc/sudoers` gives you examples, so the only thing you have to do is basically follow directions.

Here is the file before any alteration:

```
# less /etc/sudoers
# sudoers file.
#
# This file MUST be edited with the 'visudo' command as
root.
#
# See the sudoers man page for the details on how to write
a sudoers file.
#

# Host alias specification

# User alias specification

# Cmnd alias specification

# Defaults specification
```

```
# User privilege specification
root    ALL=(ALL) ALL

# Uncomment to allow people in group wheel to run all
commands
# %wheel        ALL=(ALL)       ALL


# Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL

# Samples
# %users  ALL=/sbin/mount /cdrom,/sbin/umount /cdrom
# %users  localhost=/sbin/shutdown -h now
```

As you can see this file is pretty much straight forward, but it gives you some Warning in the beginning the file, such as, only edit this file with “visudo” this will lock the file and ensure proper formatting, also directs you to the “man” pages for more information on how to build a proper sudo file.

Here is the file after my alterations, which in this case, as I mentioned before, the only administrator on this server is me, so you won’t see a huge file, but on a large site, this file can grow, and help you to manage all the users that want root’s password.

```
# sudoers file.
#
# This file MUST be edited with the 'visudo' command as
root.
#
# See the sudoers man page for the details on how to write
a sudoers file.
#

# Host alias specification

# User alias specification

# Cmnd alias specification

# Defaults specification

# User privilege specification
root    ALL=(ALL) ALL
bsouza  ALL=(ALL) ALL
```

```
# Uncomment to allow people in group wheel to run all
commands
# %wheel        ALL=(ALL)       ALL


# Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL

# Samples
# %users  ALL=/sbin/mount /cdrom,/sbin/umount /cdrom
# %users  localhost=/sbin/shutdown -h now
```

Ongoing Maintenance

To continuing keeping this system secure, we need to keep up with our maintenance, patches, software upgrades, log rotating, etc … As part of our plan of implementation, we decided to install, as little as possible, third party software, so we can alleviate the burden on monitoring for vulnerabilities and patch installation.

Since this server will be facing the internet, it is very important for us to keep this system as much updated as possible, our patches will be automated by Red Hat's up2date utility, which will facilitate the download of essential patches for our system.

Our software upgrade will be also a very important part of our ongoing maintenance; we need to keep up with all the revisions and upgrades, which in our case it is facilitated by the amount of software installed on our system. One of the major upgrades for any administrator is the operating system itself, so we need to pay attention for support dates.

I could not stress enough the importance of ongoing maintenance, otherwise all the hard work you just put on the server will be gone, because you didn't keep up with the latest vulnerabilities, which cause for you not to install the latest patch on the system, which will lead you to a possible break in, DoS, etc.

TCP Wrappers was not installed on this system, due to the fact, that we only have ftp running on xinetd and we are not setting any restriction to this service, since it is going to be an anonymous ftp site, so we don't have to maintain this information.

Daily monitoring of vulnerabilities is one of our ongoing maintenance routine, sites, such as, CERT, Bugtraq and other will be monitor every morning for any vulnerability that might possibly cause problems to us. If a vulnerability is found, we will address it, based on how much it will affect us, if will be an emergency patch or something that we can wait and schedule some outage windown.

This server will be scanned by an outside Nessus server to identify any port vulnerability that can be caught during this scan and address it right away. Not much vulnerability, but a port giving out too much information about the process that is running on that particular port. An example would be, if port 80 was giving our web server version and what web server we were using, such as, Apache, but this is only an example, we are not running any web server on this box.

Check Configuration

Here we will try to check for several setting that we set earlier in the installation, to see if they are configure correctly and working properly.

- "root" remote login:

```
login as: root
root@pluto's password:
Access denied
root@pluto's password:
```

Remote root login was denied when SSH was used. The command used above was "`ssh -l root pluto`".

- "telnet" login:

```
D:\>telnet pluto
Connecting To 172.16.8.92...Could not open a connection to
host on port 23 : Con
nect failed
```

Connection failed as it suppose, because we don't have telnet running on our server.

- Warning banner upon login:

```
$ ssh Pluto
#
#
#
This system is for the use of authorized users only.
Individuals using this computer system without authority,
or in excess of their authority, are subject to having all
of their activities on this system
monitored and recorded by system personnel.
```

```
In the course of monitoring individual improperly using
this system, or in the course of system maintenace, the
activities of authorized users may also be
monitored.

Anyone using this system expressly consents to such
monitoring and is advised
that if such monitoring reveals possible evidence of
criminal activity, system
personnel may provide the evidence of such monitoring to
law enforcement
officials.
#
#
#
bsouza@pluto's password:
```

It displayed the warning message upon login, as it suppose to.

- Sendmail not listening on port 25:

```
tcp        0      0 *:ftp     *:*                     LISTEN
tcp        0      0 *:ssh     *:*                     LISTEN
```

These are the only two processes listening on our system.

- FTP as a valid user:

```
D:\>ftp pluto
Connected to 172.16.8.92.
220 Welcome to PLUTO FTP service.
User (172.16.8.92:(none)): bsouza
530 This FTP server is anonymous only.
Login failed.
```

This server was setup to serve as an anonymous FTP server and it is acting upon, not allowing any user beside ftp and anonymous to login to the server. Further testing was done earlier in this documentation.

- SUDO – gain shell as root:

```
[bsouza@pluto bsouza]$ sudo bash
Password:
[root@pluto bsouza]# id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wh
eel)
```

I will add another user for testing purpose, so I can demonstrate how sudo works. The following is the user I created to test:

```
romeo:x:501:501::/home/romeo:/bin/bash
```

I will test to sudo to root without editing the `/etc/sudoers`, so I can see if the file is correct configured.

```
[romeo@pluto romeo]$ sudo bash

We trust you have received the usual lecture from the local
System
Administrator. It usually boils down to these three things:

        #1) Respect the privacy of others.
        #2) Think before you type.
        #3) With great power comes great responsibility.

Password:
romeo is not in the sudoers file.  This incident will be
reported.
```

The above is what happens if you are not in the `/etc/sudoers` file. Now the user will be removed from the system.

- File System Security:

```
/home type ext3 (rw,nosuid,nodev)
/tmp type ext3 (rw,noexec,nosuid,nodev)
/usr type ext3 (ro)
/usr/local type ext3 (ro)
```

The settings are in place and they will stay in place because we have this set in the file `/etc/fstab`, one more thing we have on this file that does not show here is the setting of our CD ROM, see following:

```
/dev/cdrom              /mnt/cdrom              iso9660
noauto,owner,kudzu,ro,nosuid,nodev 0 0
```

- SSH configuration:

SSH has been configured correctly on `/etc/ssh/sshd_config` file, all the setting state before are in fact applied and working properly.

- Kernel Settings:

All kernel settings stated before are in place and working properly. Here is some sample information for our `/var/log/messages` file.

```
Jun  8 17:40:51 pluto sysctl: net.ipv4.ip_forward = 0
Jun  8 17:40:51 pluto sysctl: net.ipv4.tcp_max_syn_backlog
= 4096
Jun  8 17:40:51 pluto sysctl:
net.ipv4.conf.default.rp_filter = 1
```

The above are just some examples to show you that our settings are taking place on the system.

- Logging;

Logging is working on our system, below you will see some examples form `/var/log/secure`, because the prior bullet you saw the `/var/log/messages`.

```
May 30 21:30:54 pluto sshd[929]: Failed password for root
from 192.168.110.115 port
2587 ssh2
May 30 21:31:00 pluto sshd[929]: Failed password for root
from 192.168.110.115 port
2587 ssh2
May 30 21:31:37 pluto sshd[931]: Accepted password for
bsouza from 192.168.110.115 p
ort 2590 ssh2
May 30 21:31:37 pluto sshd(pam_unix)[933]: session opened
for user bsouza by (ui
d=500)
May 30 21:31:43 pluto sshd(pam_unix)[933]: session closed
for user bsouza
```

As you can see the logging is working properly. (also, still don't know how to type)

Testing the log system.

Run the following commands to test the configuration of our /etc/syslog.conf file.

```
# logger -p daemon.info This is a test
# tail -l /var/log/messages
Jun 20 17:26:14 pluto bsouza: This is a test

# logger -p kern.panic kernel panic! Please log off NOW!
[root@pluto root]#
Message from syslogd@pluto at Fri Jun 20 17:28:59 2003 ...
```

```
pluto bsouza: kernel panic! Please log off NOW!
```

NOTE: The logger command is not a privileged command, so it is important for you to be able to identify messages coming from the logger command since users might try to fool you.

- Testing Log Rotation:

Run the command on the crontab directly on the command line and observe what happened on the `/var/log` directory.

```
# /usr/sbin/logrotate /etc/logrotate.conf
# ls /var/log
boot.log  ksyms.0  ksyms.3  ksyms.6  messages  up2date
cron      ksyms.1  ksyms.4  lastlog  secure    vsftpd.log
dmesg     ksyms.2  ksyms.5  maillog  spooler   wtmp
```

As you can see, all log files have a number after it, this is because the logrotate command ran and archived the current version and created a new log file empty on its place.

REFERENCES

1. RHCE - Red Hat Certified Engineer Linux Study Guide – by author Michael Jang, RHCE, Linux +, LCP – Osborne Certification Press.
2. The CISSP Prep Guide Gold Edition - by author Ronald L. krutz and Russell Dean Vines from Wiley press.
3. Very good source of information on http://www.puschitz.com/
4. Red Hat Errata website - http://www.redhat.com/apps/support/errata/
5. IBM QLX03 Class books on Linux Administration.