Global Information Assurance Certification Paper

Password protected Semi-Anonymous Ftp Server Installation

Randy Brown
August 4, 2004

Abstract

I was tasked with installing a secure linux-based machine for the purpose of “password protected anonymous-like” ftp access so that scientists from the field could retrieve files from our office for analysis. The information being posted was not confidential in nature; however it was not intended to be publicly available yet either. The users accessing this data needed individual ftp logins and the ability to change there password at least every 90 days in order to comply with our company policy. This machine will also have traditional anonymous ftp functionality as well to serve the rest of the office. A specified group of users from our office would then need to be able to retrieve this data and store it in a location convenient for their analysis. There was also a requirement that no additional software be required for the end user.

In general, this paper will discuss my procedure for installing the operating system, the steps taken to secure the machine, the configuration of the FTP server, and the creation of user accounts. The end result will be a secure, “password protected anonymous-like” ftp server for our office.

Pre-planning

As in most offices, money is what determines how a particular problem will be approached. In this case, no money was available to buy hardware or software to solve this problem. So I had to locate an unused machine that could perform reliably and have enough “oomph” to get the job done. Fortunately, linux operating systems can run very well on older hardware. I was able to locate a 600Mhz machine with 256Mb of RAM. Although not ideal, this was the best I could find and will probably do the job just fine.

The decision for the operating system was a fairly easy one as we run the same version of linux on all of our machines. The OS chosen was Redhat Linux version 7.2. Also weighing into this decision was that this version has been very stable for us and our familiarity of it would be an advantage for system administration. The impending demise of support for this version was a concern, but it was decided that upgrading would be dealt with at a later time.

It was requested that a single account be created that could be shared by the 12 scientists that would be accessing this machine. We did not like the sound of that from the start, nor did it meet with our company policy. We are not allowed to use shared accounts. This meant that a separate ftp account per user would be needed along with an administrative account or group in our office to retrieve any uploaded data. Another requirement of our company policy is to change passwords at least every 90 days. This meant that we also needed a way by which the users could change their passwords using their ftp login and not giving them a full blown user account on the machine.

This machine would be located on the public internet (not behind a firewall) so securing this machine was extremely important. We needed to ensure that no extraneous services were running and that the machine was thoroughly patched. Most likely, the only services running with listening ports

would be sshd and ftpd. (All machines we have are installed securely regardless of whether or not they are behind a firewall. “Defense in Depth” is the key.)

Operating System Installation

I obtained the CD and began the installation process. (I am not going to cover all of the specifics of the OS installation; keyboard, mouse, partition sizes, etc, I will mention the steps that are of relevance to this paper.) I proceeded through the installation configuring the partitions as desired. I chose to breakout /var into its own file system so that I could more easily monitor the disk usage of where the logs were being written. I also made /home/ftp its own file system for similar reasons. This also allows me the option of increasing this disk space down the road with less impact on the rest of the system.

I did not configure the network during the install because I was not going to connect the machine to the network until which time I could confirm that only the services I wanted running were running and that no unnecessary TCP ports were listening. I set the root password on the machine when prompted and created a regular user account that will be used for general logging in to the machine. Being that this machine was going to be on the public internet, I ensured that the password was unique so in the event the machine was compromised, and the password stolen, it wouldn’t allow the attacker access to any other machine on our network.

The next stage was the package selection. Knowing that this machine is only going to be used for a specific purpose, I felt I could be very selective about which packages I needed to install. Basically, I needed ftp and ssh and not much else other than system essential packages. I have found that there are two schools of thought when it comes to package installing: Install only what you need, or, install everything to minimize dependency issues when patching the machine. I take more of the minimalist approach when it comes to installing packages. If I’m not using it, I don’t want it on the machine. I selected my minimal set of packages and proceeded to install.

When the machine finished the operating system installation, I rebooted and watched closely for any errors and for any services that may be starting which are not necessary and will need to be turned off.

The “Lock-down” Process

I first wanted to see what was running on this machine. Using:

```
chkconfig –list |grep on
```

I could see the services that were running and at which runlevel they were available. This list will vary depending on what you have installed. Using the command:

```
chkconfig –level 123456 <service> off
```

I turned off any unnecessary services. Some of the services I turned off were: portmap, autofs, gpm, anacron, identd, nfslock

This step doesn’t stop the selected services, but it does make them so they do not start up again during the boot process. This could have also have been done

by using the program /usr/sbin/setup and selecting “System Services” or, by renaming the start scripts in the /etc/rc.d/rcX.d directories (Replace X with the runlevel value)

In the /etc/passwd file, I wanted to disable certain accounts making them unusable to any would be intruder. The way I did this was to set the shell of these users to /bin/true. /bin/true returns an exit status of “0” which means it was successful, but nothing more. It can’t accept any arguments to modify its behavior either. Using this as a shell is an effective way of locking down accounts. I used this on the following accounts: shutdown, halt, nobody, xfs, and gdm. Normally, I would use this on the ftp user account too, but that is the account used for anonymous ftp access which this machine will need.

Now I took the machine to runlevel 1 (init 1) by issuing the `init 1` command from the command line. This shuts all services down taking the machine to single user mode. This is a faster option than a full reboot of the machine fits the purpose of what I am trying to accomplish; stop all running services. With the machine at runlevel 1, I log in and run `init 3`. This takes the machine to runlevel 3, multi-user mode, and restarts the services based on the changes I made using chkconfig. Now using the netstat program, I can see which services have ports open and listening.

```
netstat -na |grep tcp
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
```

Only ports 21 and 22 are listening. (ftp and ssh, respectively) I now want to make entries to the /etc/host.allow and /etc/host.deny files for access control. These ACLs (Access Control List) are used to control access for anything wrapped by the installed TCPWrappers package. (http://www.porcupine.org). Binaries like telnet and ftp can be wrapped by TCPWrappers. This launches the wrapper for the inetd connections which logs access attempts and verifies the attempt against the access control list. The access control lists for TCPWrappers are the /etc/hosts.allow and /etc/hosts.deny files. In the /etc/hosts.deny file I add a single entry:
all: all
This serves the purpose of blocking all access first then letting me use the /etc/hosts.allow file to allow access for specific hosts or networks as I see fit. I made the following entries to the /etc/hosts.allow file:
sshd: 111.222.333. 444.555.666.
in.ftpd: all
(The addresses are obviously bogus and represent the network addresses of our two internal networks from which the administrators and ftp administrators will need access). Although the use of the word all in an allow file of any type is a bit scary, it is necessary here due to the anonymous function of this machine.

Banners were added at all entry points. This is done by creating the /etc/issue, /etc/issue.net, /etc/motd files. These are ASCII text files which contain a

“legalese” warning message displayed when anyone accesses the system using, for example, telnet. The content of these files are identical in our case. We also added the line:
banner /etc/issue
to the /etc/ssh/sshd_config file so that the warning message is also displayed when the machine is accessed via ssh connections. The same warning message was also added in the Welcome section of the /etc/X11/gdm/gdm.conf file. This displays the warning message when logging in at the console at run level 5.

There are three files that must also to be locked down on this host. They are the .rhosts, .netrc, and /etc/hosts.equiv files. The .rhosts file controls the use of “r” commands like rlogin, rsh, and rcp, for example. They are very insecure and should be avoided whenever possible. Ssh can perform the same functions, but in a secure manner. The .netrc file allows you to predefine username and password information for ftp logins. This file is kept in the users home directory, most often, and is in plain ASCII text. The hosts.equiv file is used to define remote hosts ans users that are considered trusted. Trusted users are able to access the system without entering a password. This is similar to the function of the .rhosts file, but the hosts.equiv file is system wide, whereas each user can have an individual .rhosts file in their home directory. The following steps are taken to ensure that no one can create or modify these files:
/bin/touch /.rhosts /.netrc /etc/hosts.equiv
/bin/chmod 0 /.rhosts /.netrc /etc/hosts.equiv

Shadow passwords were enabled by running pwconv. This is an extremely important step. Enabling shadow passwords stores the users’ passwords in the /etc/shadow file which can only be accessed by root. This step protects passwords from easily being hacked and/or cracked.

We configure all of our machines to send logging data to a remote log host. This is desired for a couple of reasons. First and foremost, if the local machine is compromised, one of the things a hacker will do is cover his tracks by modifying the log files removing all traces of their access. Logging to a remote host means that the hacker must now gain access to a second machine, the log host, in order to cover his tracks. This is probably more than most hackers will want to have to deal with. Second, log files can sometimes get pretty big and logging to another host will avoid file systems filling up which can bring a machine to a slow crawl, and maybe even crash the machine.

Any extraneous accounts are also removed from the /etc/passwd file. Normally, the ftp account would be included in this group. But, since this machine will be used for anonymous ftp access, it is necessary to keep this account. Other accounts, such as; news, cron …were removed.

It is at this point that I am comfortable putting the machine on the network. I make the necessary entries in the /etc/sysconfig/network, /etc/sysconfig/network-scripts/ifcfg-eth0, /etc/hosts, and /etc/resolv.conf files:

/etc/sysconfig/network – turns networking on, sets gateway and hostname
etc/sysconfig/network-scripts/ifcfg-eth0 – Sets IP specific info – IP address, netmask, etc.
/etc/hosts – add entry for local machine's IP address and hostname
/etc/resolv.conf – Add domain name and IP addresses of DNS servers

I attached the machine to the network and restarted the network service using:
service network stop
service network start

The machine was now connected to the network.

The next step is to apply all necessary patches. There are a number of ways this can be done. We chose to use a package called Yum. (http://linux.duke.edu/projects/yum/) The rpm was downloaded and installed. Our company CIRT (Computer Incident Response Team) has pre-packaged the Yum program turning on GPG checking and also pointing the program to a repository maintained by our CIRT team.

When installing a Red Hat 7.2 box for the first time, yum will not install unless you first update rpm*, gnorpm, popt, and kdeadmin. This was the case for this installation. We also saw a conflict on Redhat 7.2 between recent kernels and the bcm5820 package. We erased the bcm5820 package and all was well. In addition, on Red Hat 7.2, yum requires gnupg-1.0.7 or later. We upgraded gnupg before installing yum.

We decided to not let Yum update the kernels. We prefer to test all kernel upgrades on non-operational machines first. This is done by editing the /etc/yum.conf file and adding the line:
exclude=kernel*
to the [main] section. The kernel was upgraded manually and the machine rebooted to the new kernel. Once all of the updates had been performed, the kernel had been upgraded, and the machine had been rebooted, the netstat and chkconfig commands, as described previously, were run again to ensure that no unwanted services were running and that no undesired ports were listening for connections as a result of the patching and reboot. It is always a good idea to repeat this step whenever patching or programs are added to your machine. Services can be inadvertently started by a patch and reboot and can open holes in your secured system. Sendmail is an example of a program that exhibits this behavior.

I had completed the installation of the operating system and had secured the machine to a desirable level. I now had to begin the planning of the FTP access configuration side of this project.

FTP configuration

This was the trickiest part for me as I had never set up an FTP server before. I needed to figure out exactly what I needed to be concerned with from a vulnerability standpoint. What makes and FTP server vulnerable? How are these vulnerabilities addressed? Can I provide the functionality the end users are requiring and still maintain an acceptable level of security? The end users are sympathetic to our security needs, for the most part, but I also knew that they do not want security to hinder their productivity. The solution had to be secure, for obvious reasons, but the security needed to be as transparent as possible to the end user. The end users had varying levels of computer skills and their operating systems could be anything from Microsoft to any one of the many unix based operating systems.

Vulnerabilities/Security Concerns of FTP

The biggest vulnerability we have to deal with is the fact that this machine will located on the public internet. Therefore, anyone will have access to it. We need to control the type of access different groups of users have. The general public will have access to the machine for anonymous ftp. There will also be our group of users that want to have anonymous-like access. (The anonymous-like access is access that will be available to a select group of users. However, each account will be used by multiple people at each contributing site. That is the reason for the use of the word anonymous here. It's anonymous in that the person contributing the data is unknown.) This select group will be able to change their password via ssh, but otherwise, have nothing but ftp upload access. We will restrict the access to the server through the use of ACLs (/etc/hosts.allow) as discussed earlier.

Anonymous ftp sites are also commonly used as “warez” sites for pirating software. These are sites on which people place software for the purpose of the general public being able to download it. Unfortunately, often times the software on these sites is commercially licensed software that being made available illegally. One way to avoid your site being used this way is to make the directories unreadable by the people accessing it with the exception of the people administrating the site. This way, unless you know the exact name of the files on the site, or have the proper permissions to see the files, you cannot see the files using `ls`, for example. Thus making it much more difficult to know what is there for downloading. Although this is not 100% fool proof, it will work well for deterring the majority of troublemakers.

The users accessing this machine will have no need to do anything other than place their data in a designated folder and then exit the machine. It is therefore important to limit the access of the users to a specific directory tree. Usually, /home/ftp/pub/ or something similar. This is accomplished by using the chroot command. This will ensure that the users cannot browse our system once they have logged in via their ftp account.

Because of the security risks involved with the operation of an ftp server, we have chosen to put this functionality on it's own machine. That way, if this machine were to be compromised, the rest of our network is not at risk. We will

keep a close watch on the log files for suspicious activity. We also know the type of data that being uploaded to this machine and should be able to tell if there are files being uploaded that are not of the proper type.

The FTP Server Configuration

The ftp user was setup in the /etc/passwd file pointing it to the proper home directory, /home/ftp, and ensuring that it could not be used as a normal login account. This was done by setting the shell to a value of /sbin/nologin and placing an asterisk in the password field. The finished entry looks like this:
ftp:*:14:50:FTP User:/home/ftp:/bin/false

A group needed to be created to which each of our end users would belong, allowing us to control the access to the upload directory. This group will be called “science.” This group was given the next available group ID, in this case, 262. The entry made in the /etc/group file was as follows:
science:x:262:
An additional group, ftpadmins was created for the purpose of maintaining the pub incoming directory contents. This is a select group of trusted users that have access to the contents of the /home/ftp/pub/incoming directory so that they can remove and/or move files as needed.

Also created was the /home/ftp/pub/incoming directory for the anonymous ftp function of this server. The /home/ftp/pub directory was configured with 555 permissions and the ownership set to root and the group to wheel. The incoming directory was created with 773 permissions. This prevents the contents of the incoming directory from being seen by the public making it more secure and unusable as a “warez” site as described previously. It was owned by root and the group was set to ftpadmins. The home directory for each of the science group users was then created. This directory would be called /home/ftp/pub/science. The permissions were set to 750. It is owned by root and is in group science. This allows only root or members of the science group to access the data in this directory.

Under the /home/ftp directory, there are bin, etc, lib, tmp, and usr directories created by default from the installation. Permissions on each were set to 555. They are owned by root and are in the wheel group. (The wheel group is used to increase the security of a system by preventing users that are not members of this group from using the su command to su to root.) These directories will act as the normal /bin, /etc, etc. directories as far as the ftp users are concerned. The bin directory will contain commands that are accessible by ftp users that have logged in. This is normally a very small group of commands. We chose to only include compress, ls, and tar. The permissions on these files were all set to 111. (only the execute bit was set) This prevents these files from being downloaded by the clients and possibly being studied for vulnerabilities. The etc directory will contain skeletal copies of the /etc/passwd and /etc/group files. The entries will look similar to:
root:*:0:0:::
<username>:*:XX:YY:::

where: * is the place holder for the encrypted password
XX is the UID
YY is the GID
No other information is needed as this file only serves the purpose of enabling the display of “human-friendly” names instead of numbers when referring to users and groups. Permissions were set to 444 on each file. The password field for each entry in the /etc/passwd file are replaced with asterisks. This was done to help improve the security of the passwords. Only necessary accounts will be included in the copy of the passwd and group files.

The default /etc/xinetd.d/wu-ftpd file was then modified removing the USERID parameter from both the log_on_success and log_on_failure lines. The USERID will only be available if the service identd is running. This is usually turned off, or more correctly, never turned on. The server_args parameters are important here. The -l option logs ftp sessions to syslog and the -a option enables the ftpaccess configuration file. This is the file that controls the server behavior. After any changes are made to the /etc/xinetd.d/wu-ftpd file, the xinetd service must be restarted. This is done using:
service xinetd restart

The ftpccess file was the next file to be configured. Our group needed to be added to allow access. The line:
guestgroup science
was added for this purpose. The user classes line was then set to:
class all real,guest,anonymous *
This creates a class called all which consists of real, guest, and anonymous users and is allowing access from any IP address. All IP addresses need to be allowed due to the anonymous nature of this server. The email address was then set to that of our tech support email account which is then forwarded to each of the members via a .forward file in the support accounts home directory. The loginfails parameter was set to 3. This is the number of times a user can fail at an attempt to login to your server before being disconnected. It is a very good idea to keep this number low. The next addition to this file was to add a banner line. We set this to display the /etc/issue file per our company policy. As an added precaution we also set the noretrieve parameter. This is a list of files that under no circumstances can be retrieved from this site. These files included the .forward, .rhosts, hosts, passwd, group, .notar, lost+found, .shosts, and hosts.equiv files. Finally we needed to configure the upload permissions for the site. We wanted no upload capability to the root directory of our ftp server, which is /home/ftp. We also wanted to allow uploading to the /home/ftp/pub/incoming directory. This was done by adding the lines:
upload /home/ftp * no
upload /home/ftp /pub/incoming yes root ftpadmin 0660 nodirs
The first line says that all directories (*) under /home/ftp have no upload permission. The second line states that the pub/incoming directory under /home/ftp has upload permission and that all files uploaded here will will be owned by root, have group assignment of ftpadmin, have permissions set to 660

(rw-rw-rw-) and the the person connecting cannot create directories under /home/ftp/pub/incoming.

Now the actual user accounts for the science group users need to be added to the /etc/passwd file. Each entry will have the format:
<username>:<password>:UID:GID:User info \
:/home/ftp/./pub/science:/usr/bin/passwd
NOTE this should be a continuous line. The \ indicates the line wrapping
Notice the "/./" in the home directory field. This is what causes this account to be chroot'd to /home/ftp. Chroot'ing the user keeps them from being able to browse the rest of your system. It makes /home/ftp appear as / to users ftping to your server. In other words, a user that runs the command cd / would not go to / on you machine but would instead go to /home/ftp which is mimicking / as a result of the chroot. The other thing of note in the /etc/passwd entries is the shell being set to /usr/bin/passwd. This is done so that the user can ssh into the machine for the specific purpose of changing their password, but nothing else. Upon the completion of the password change, the connection is dropped. An additional step though, is that /usr/bin/passwd needs to be added to the /etc/shells file for this work properly. A password was configured for each account which was passed on to the end user who was instructed to change their password upon their first attempt to log in.

The configuration is now complete and can be tested. We tested the machine as follows:

ftp <machine name>
Or: *ftp <IP address>*
and log in using one of the username/password pairs created. Issuing a pwd command should return something similar to:
ftp> pwd
257 "/pub/science" is current directory.
/pub/science is returned due to specifying it as the users home directory in the /etc/passwd file entry.
Issuing a cd / followed by a pwd command should return:
ftp> cd /
250 CWD command successful.
ftp> pwd
257 "/" is current directory.
Where "/" is actually /home/ftp. Doing and ls should show the same contents as doing an ls in /home/ftp. Issuing a cd /var or cd /sbin should return:
ftp> cd /sbin
550 /sbin: No such file or directory.
Or
ftp> cd /var
550 /var: No such file or directory.
unless there is a var or sbin directory under /home/ftp. Results, as shown above, verify that the users cannot get out of /home/ftp by cd'ing to a directory that does not exist under /home/ftp but does exist on the machine.

Final Analysis

The finished product is an individual machine that has been locked down by limiting the services running and ports listening for connection to only those required for the task at hand. The machine provides anonymous ftp upload access as well as the ability to allow a select group of user to access the machine using a valid username and password combination to download files posted by our department. An ftpadmin group was established for the purpose of maintaining the anonymous upload directory, /home/ftp/pub/incoming. This solution has been working well for the field and our office for awhile now.

As with any publicly accessible machine there will always be the possibility that it will someday be compromised. If that does happen, our regular monitoring of log files along with an intimate knowledge of the machine configuration and the data being stored on it, we can catch it early on and minimize the damage.

I learned a lot through this process. I had never before attempted a task like this, so every step had to be learned a long the way. I was fortunate enough to be involved in the SANS Security Essentials course when this task came up which helped in every aspect of the decision making throughout this project. The Securing Unix section in particular was very helpful as were others. I'm sure as my knowledge increases that I will find faults or things that could have been better, but I will chalk that up to experience and lessons learned and grow from there.

References:

Shah, Steve. Linux Administration a Beginner's Guide. Berkley: Osbourne/McGraw Hill, 2001. 305-333

Nemeth, Evi; Snyder, Garth; Seebass, Scott; Hein, Trent R. Unix System Administration Handbook. Upper Saddle River: Prentice Hall PTR, 2001. 696-697

Spitzner, Lance. “Armoring Linux: Preparing Your Linux Box for the Internet.” URL: http://www.justlinux.com/nhf/Security/Armoring_Linux.html (7 Jul. 2004).

O'Dell, Bob, Dr. “Ftp Security.” URL: http://www.thejournal.com/thefocus/featureprintversion.cfm?newsid=9 (3 Jul. 2004).

Planting, Alex. “Setting up an Secure FTP server.” 3 Sep. 2002. URL: http://www.linuxnetmag.com/en/issue7/m7secureftp1.html (25 Jun. 2004).