Basic Apache Security Considerations
John E. Grotevant, CISA
May 20, 2001

Introduction

Apache is the silent workhorse of the Internet. All the glamour and glory go to the big players of the Internet – Sun, Microsoft and Cisco. But Apache delivers almost all we see on the World Wide Web. The Internet might have not grown as large and quick if it was not for the Apache Web Server.

The success of the Open Source movement is most compelling with the Apache. Apache powers more than 62% of web servers on the Internet according to NetCraft.com's April Web Server Survey. That is almost 18 million web servers! What is more impressive is that the next most popular web servers, Microsoft's IIS and Netscape, comprise only 20% and 6% respectively. Furthermore, April statistics indicate Apache installations are increasing at a faster rate. Apache experienced an increase of 2.3% from February versus IIS with a .89% increase and Netscape with .03% increase.

Background

The origins of Apache came from the National Center for Supercomputing Applications (NCSA) HTTPd Web Server, developed by Rob McCool in the early 1990's. When the NCSA HTTPd project was discontinued, many webmasters had made their own modifications to the HTTPd software. With all the modifications, it became "a patchy" web server and from there the name Apache was contrived.

Eight core contributors created the foundation of the original Apache group in February 1995. The first public release of Apache (v 0.6.2) was released to the public in April 1995. Apache 1.0 was released on December 1, 1995 and surpassed the HTTPd web server as the #1 server on the Internet only 1 year after the group was formed.

The Apache Software Foundation now oversees the Apache HTTP Server Project. The project "is a collaborative software development effort aimed at creating a robust, commercial-grade, featureful, and freely-available source code implementation of an HTTP (Web) server "[1]. The Apache Web Server remains free under the requirements set in the Apache Software License.

Supporting Actors

Your house is only as secure as its (start your British Accent now) "weakest link". Your Apache installation will only be as secure as your operating system and its network

connectivity to the rest of the world. It does not matter if all the doors are locked when a basement window is wide open. Securing the operating system is a topic much too large for this discussion, but a very pertinent topic relating to the overall securing of Apache.

The operating system should be configured to provide the functionality that is required to operate Apache efficiently and securely. Basically, if you are creating a web server, you do not want to be running a variety of other services on the same machine. A web server is just that - your ftp, mail, DNS and any other services should all be run on other machines and secured accordingly, as well as unnecessary compilers and applications. Additionally, you do not need give every user access. Provide access based on responsibilities and educate those users on good passwords and other information security practices.

Network security also supports the security of the web server. Many times the Apache server will be connected to the Internet directly and not controlled by a screening router or firewall. If you are concerned about your site's integrity and reputation, a layer of network security should be considered. These security considerations will help in fending off Denial of Service and other network-based attacks, which ultimately affects your site's availability and security. If your budget does not support network security measures and you need to limit access to the web server, consider using the operating system to limit access. Tools such as TCP Wrappers, IPTables, SSH and Snort (links in Reference) will all give you "excellent" network layer security measures without the "excellent" cost overhead. You should also leverage some of the security options in Apache, which are discussed later in this paper.

Policy, Policy, Policy

Before beginning an installation in an organizational setting (Corporate, Educational, etc.), make sure you are aware of the established computing policy. During your installation, you may have to make decisions that will be affected by your established policy. And once your Apache web server is installed, revisit those policies, update them and create new ones for future Apache installations.

Assumptions

This paper will refer to Apache in a POSIX environment (i.e. Linux, UNIX, etc.), more specifically Red Hat Linux 7. Apache has been tailored over the years to operate effectively, efficiently and securely in the world of the Unices. Apache is but a newborn in the world of Microsoft. Therefore when considering operating Apache securely, consider proven ground the best ground to tread. Another assumption that will be made is that you are familiar with basic UNIX commands. Therefore the command detail will not be noted after lines such as "change file permissions 755" etc. I apologize to the newbies for this, but for the sake of brevity, I may omit some details.

Obtaining Apache

Before securely installing Apache, you need to obtain the software and ensure the code has retained its integrity. Obviously the best place to obtain the most recent release of the Apache software is at the web site of the Apache HTTP Server Project (httpd.apache.org). The web site provides the most recent stable version, all past versions and alpha releases. Although it is sometimes nice to have the latest and greatest, if you are concerned about security and availability, you should only download the most stable versions. As of April 1, 2001, Apache 1.3.19 is available and corrects some earlier versions security issues.

You most likely will obtain and install Apache when building a new server. Most Linux distributions include the web servers on their CD-ROMs. Always check the O/S vendor site to see if upgrades are available. Red Hat Linux 7 is shipped with version 1.3.12, however a more recent RPM is available at their web site.

To ensure the integrity of the software, you should always validate the download and the signature. The Apache download site includes the PGP and MD5 keys that can be used to verify and validate your download. If you are using an RPM, use the command 'rpm–K packagename.rpm' to make sure the package was not corrupted during the download process. If you have also installed and configured GnuPGP, which is released with most Linux distributions, you can verify the signature with the same command. This verification topic is broad for a general discussion, but a very important step in the secure installation of Apache.

Before beginning the installation, be sure to reference the Apache documentation http://httpd.apache.org/docs/ for general information, installation instructions and security related tips.

Installation

In this example, we used an Apache RPM for installation. For those customizing the Apache install, the source files should be downloaded and compiled. This will give you the most control over your installation.

Another nice feature of using an RPM installation is that you can query a package and obtain a listing of all the directories and files that have been installed. This is especially simple with a tool like gnorpm, however the command 'rpm–q apache –l |more' will get you the same information.

After querying the installation for this example, we can see that the directories used by apache are:

/etc/httpd
/etc/logratate
/etc/rc.d/init.d/httpd

/usr/bin/
/usr/lib/apache
/usr/sbin
/var/cache
/var/log
/var/www

Newer releases of Apache provide decent default directory and file permissions and ownership. The web server is run under the 'apache' user. Therefore this account does not require and by default does not have write access to the server directories (ServerRoot in this example /etc/httpd/), especially its control files (/etc/httpd/conf/*). Configuration files are owned by root and the apache server is only permitted read access through the world permissions, which can even be removed.

As for the web document directories, which will be discussed later in the configuration and in this example are in /var/www, these documents are, by default, owned by root and the apache server is limited to read and execute access only through world permissions. Since root does not and should not update these files, it is a good idea to change the ownership accordingly but not to the apache account. Furthermore, the truly paranoid might try to remove world read and execute access to the document directory, but the web server cannot serve what it cannot read, so this will not work. If the execute access is removed, you will not be granted access to the page, even if it is static.

And finally, do not forget any additional directories or files that you may point to with symbolic links. As we will discuss later in the configuration section, symbolic links can be followed or ignored by Apache. Therefore, before you create new symbolic links, check your Apache configuration.

Another file you may create is a user file (.htpasswd), which will contain your http users and passwords for authentication. This file is created when you execute the htpasswd command to add a user

htpasswd –cmb /username/path user password
where:
Flag – c – create new user file
Flag – m – Force MD5 encryption of passwords (highly recommended)
Flag – b – User password from command line
/username/path - in this example /etc/httpd/conf/.htpasswd

By default, the .htpasswd file was given the permissions of 644 and was owned by root and the root group. Note that by default it is readable by all users. Even though it is encrypted, it should not be world readable (remember that is why we have a shadow password file!) This allows Apache to read the file and if removed, Apache cannot authenticate a user. On Red Hat Linux 7, all users have a corresponding group. Therefore change the group ownership to apache ("chgrp apache .htpasswd") and then update permissions to 640. This prevents your general users on your server from viewing your Apache user file.

Configuration Considerations

Older versions of Apache use separate configuration files. The three files used for configuration are the access.conf, srm.conf and httpd.conf. These three files have been combined into the httpd.conf. Although the access.conf and srm.conf are still created, all their parameters are in the httpd.conf file.

The following parameters were taken from the httpd.conf file, in which they are described to some extent, therefore I will not repeat too much of that same information. Let us take a look at a few of the parameters that may have security repercussions:

ServerType
The Apache server can be run standalone or as part as the inetd, or in new versions of Red Hat xinetd. For the ease of configuration and security, it is best to leave the default and run as a standalone server.

ServerRoot
Whatever root you decide to use, make sure its related file permissions and ownership are secure. Just make sure it is not the root directory (/). In this example, the server root is /etc/httpd/.

ResourceConfig / AccessConfig
If you would revert back to using a separate file for access.conf and srm.conf, they can be activated with this parameter.

KeepAlive / MaxKeepAliveRequests / KeepAliveTimeout
Setting for persistent connections. These configurations will help avoid a denial of service instance if there are a large number of constant connections and your web server cannot keep up.

MinSpareServers / MaxSpareServers / StartServers / MaxClients / MaxRequestsPerChild
Although these settings do not affect security specifically, they do assist in optimizing server capacity. Security includes availability, therefore these parameters should be tweaked to ensure you are not restricting access because of too many servers running, thereby affecting capacity. The MaxClients maximum is 0 – which is infinite – which could definitely negatively impact your response time.

User / Group
As previously discussed, your web server will run under an account. By default this is the 'apache' account. Once you identify what account the server is using, you can then lock down the configuration files from that account as previously discussed.

ServerAdmin
This may be a small point, but do not include your email address on the server which is running the Apache server. As you will probably not be using that server as the primary source of email, it is prudent to have it sent someplace where you will identify it timely.

DocumentRoot
This directory holds your web content, in this example is /var/www/. Once again, use this parameter to identify or define your web documents directory and lock it down accordingly. As previously stated, newer releases of Apache have assimilated the traditional access.conf file into the httpd.conf file. Therefore to control access to any web directories, it would be defined after the DocumentRoot parameter. This can be done by basic authentication using the .htaccess files (change AllowOverride from None to AuthConfig) or you can use other allow or deny parameters to define access by keywords, full or partial domain names, full or partial IP addresses or network/netmask pair. (For efficiency, it is best to use IP addresses and not domain names, as resolution would need to take place) This is a nice way to block unwanted IP traffic if you do not have a firewall or router doing that for you.

Options FollowSymLinks / FollowSymLinksIfOwnerMatch
Directory and file permissions are important, however a symbolic link could point to a file or directory that provides different access. The FollowSymLinks option tells Apache to utilize a symbolic link regardless of the owner of the file linked to. FollowSymLinksIfOwnerMatch guides Apache to follow a link if the user ID that owns the destination file is the same as the original file. If links must be used, the latter option is a more prudent security option.

Options Indexes
If a URL request is made to a web directory, Apache can respond in one of three ways: return a file, display the directory list or return an error page that access is denied. If the Options Indexes directive is included, the directory contents will be returned. Additionally, if FancyIndexing is turned on, the information will include the files modified date, size and description. If you do not want users to browse your DocumentRoot Directory, remove the Indexes option. This option is included by default for your html and icons directories. Therefore you should remove that option to prevent the viewing of your files.

AccessFileName / Viewing
This parameter defines what access file is used to control access to Apaches documents. We are assuming this web server is for general information purposes only and not restricting access. However if you would like to limit access, this parameter defines the access file (default .htaccess) Additionally, by default .htaccess files cannot be viewed by visitors, however if the parameter is commented out, anyone can view these files by visiting your site.

ErrorLog / LogLevel / LogFormat /CustomLog
These parameters set your logging criteria. Once again, note the location of your logs and ensure file access is defined accordingly. Logging will help you trouble shoot problems, and possibly security breaches, so make sure you are capturing enough information. Based on your web traffic, you should also rotate the logs and archive them.

ScriptAlias
This provides an alias to what your CGI-Bin directory actually is. By default it is modified for you. Make sure the source file permissions are locked down and the directory is not owned by root. Also, here is an instance where the Symbolic Link options should not be used. If you are using CGI scripts, be sure to research manners to implement in a secure manner.

http put
By default it is not enabled, however uncommenting the parameter will allow http puts. This will be an unnecessary function for many installations, however if you must use it, be sure to secure it with the parameters and good file segregation.

Chroot Apache

To go the extra step in securing your Apache Web server, you may want to consider running the web server in a chroot environment. This will help you avoid future directory traversal attacks. What is chroot, you ask? Per the man pages:

> **Chroot** causes this directory to become the root directory, the starting point for path names beginning with ``/ "

You will create a new root directory that Apache will run within. All services and commands will run within this new root directory or sometimes called a jail. The steps to creating a chroot environment, particularly on Linux and Solaris, are well documented on the following sites:

http://www.linuxdoc.org/LDP/solrhe/Securing-Optimizing-Linux-RH-Edition-v1.3/chap29sec254.html
http://hoohoo.ncsa.uiuc.edu/docs/tutorials/chroot.html
http://penguin.epfl.ch/chroot.html

Just as good as the link above is for creating a chroot environment, the following link provides an overview of how you might break out of a chroot jail:

http://www.bpfh.net/simes/computing/chroot-break.html

By using both resources, you can balance the risk and overhead of using a chroot environment for securing your Apache web server.

SSL

This document will not go into detail on installing and configuring SSL. But here is a good reference for starters:
http://www.infosecuritymag.com/articles/april01/features1_web_server_sec.shtml.
SSL will provide encryption for your server. Before implementing consider revisiting your policy. Also consider the impact on any Intrusion Detection Systems (IDS) that you may have in place. Your SSL implementation may provide cover for some forms of intrusions.

Backup, Plan, Test, Monitor

Once you have configured your Apache server as secure as you see fit, perform a full backup of your server. These backups will be incredibly valuable in the event your server is compromised or experiences a hardware problem. Always have a full backup and incrementals after any change. Test your backups and your recovery plan. It is better to find out your backup process is not complete during a test instead of an actual outage.

Monitor your web server. Review the logs and look for anomalies, which may indicate some problems. Install TripWire (See References), which is a file integrity checker. If a file has been modified, TripWire will identify the change.

Finally monitor security updates from security organizations or Apache periodicals. Frequently visit a site such as the Carnegie Mellon Software Engineering Institute (http://www.cert.org) or the System Administration Networking and Security (SANS) Institute (http://www.sans.org) for security issues with Apache. SecurityFocus (http://www.securityfocus.com) provides a function to search its vulnerabilities for Apache related issues. Consider visiting these sites before your installation to avoid known problems ahead of time. And of course visit the Apache Specific Web Sites (http://www.apache.org), ApacheToday (http://www.apachetoday.com) and Apache on the O'Reilly Network (http://www.onlamp.com/apache/).

Accreditation and Certification

So you have successfully installed and secured you Apache web server. If your web server will be operating in an enterprise or ecommerce capacity, you may consider or be required to have the security of your site validated by a third party. This assessment will provide certification that your site has been configured securely. The following professional organizations provide this type of certification (REF)

WebTrust Certification – The American Institute of Certified Public Accountants (http://www.aicpa.org)
TruSecure – International Computer Security Association (http://www.trusecure.com)

Conclusion

The Apache Web Server is steadily growing in popularity. I feel that its use will continue to grow even more, especially as more bandwidth is available to home and small business users who begin to experiment with hosting their own websites. The versatility and customization will continue to allure the advanced users as well. All this with the power of the Open-Source movement will keep Apache on top.

REFERENCE:

Web References:

1 - About the HTTPd Server Project: How Apache Came to Be,
URL: http://httpd.apache.org/ABOUT_APACHE.html (04/01/2001)
(Apache original eight core contributors: Brian Behlendorf, Roy T. Fielding, Rob Hartill, David Robinson, Cliff Skolnick, Randy Terbush, Robert S. Thau, Andrew Wilson)

Netcraft Web Server Survey (April 2001), URL: http://www.netcraft.com/survey/ (05/21/2001)

World Wide Web Consortium (W3C) FAQ - http://www.w3.org/Security/Faq/www-security-faq.html#contents, Version 2.0.1, March 24, 2000

Print References:

Bill Ball, David Pitts, et al. Red Hat Linux 7 Unleashed; SAMS Publishing 2001;

Rich Bowen and Ken Coar, Apache Server Unleashed, SAMS Publishing 2000.

Anonymous, Maximum Linux Security –SAMS Publishing 2000

Complimentary Tools:

IPTables – Linux Firewall - http://netfilter.kernelnotes.org/
OpenSSH – Secure Shell - http://www.openssh.org/
Snort – Linux IDS – http://www.snort.org
TCPWrappers - ftp://ftp.uwsg.indiana.edu/pub/security/wrapbin/ (among many other reputable sites)
TripWire – http://www.tripwire.org/