Global Information Assurance Certification Paper

Protecting the Apache HTTP Server
General Security & Protection From HTTP DoS Attacks, Buffer Overflows, and Root Access

SANS GIAC Level 1 Security Essentials
Class Project Version 1.2b

Kevin J. Martin

April 4th, 2001

Purpose

This paper will explain factors involved in the secure installation and configuration of the Apache HTTP Server on a UNIX platform. We will focus on UNIX because the Windows Version of Apache Server is primarily in the testing phase. This paper will concentrate on several issues regarding web server security. It will be assumed that the administrator has selected the modules pertinent to the requirements of their site and that they were able to configure, build, and troubleshoot the server. Additionally, it is the intent of this paper to serve as a guide to assist in securing the Apache Server.

The Apache HTTP Server is the most popular web server globally in use today and presents itself as a secure and very well designed server application. However, with any application there are vulnerabilities. Three of the vulnerabilities that this paper will focus on are denial of service (DoS) attacks using the HTTP protocol, buffer overflows, and root exploits. It is important to note that the proper configuration of Apache will help protect the server against several types of attacks, however a DoS attack at the network layer cannot be stopped by any Apache configuration or directive. The DoS attack referred to in this paper is an attack directed at the HTTP protocol.

Major Apache Vulnerabilities

HTTP Dos: The attacker generates an HTTP object from the server. This causes an increase in CPU and memory usage. As a result of this attack the system will slow and perhaps stop functioning completely.

Buffer Overflow: An exception condition created by a poorly designed program. This program uses static memory allocation and is overflowed easily by a very large input string. An example would be a Perl script program designed to request user input. Once the overflow condition has taken place the attacker is able to execute code on the server. Sometimes the system will come to a halt as a result of the buffer overflow condition.

Root Exploit: The attacker gains root access through the Apache Server application, which is running as a root process, then has the capability of gaining full control of the system.

Acquiring the Latest Apache Software

It is important to be running the most secure version of the Apache HTTP Server available. You may download latest secure versions of the Apache Server software from their Web Site located at www.Apache.org. The most recent compressed tar file apache_1.3.19.tar.z can be downloaded from http://httpd.apache.org/dist/httpd. A compressed tar is similar to a Winzip

file. This contains Apache HTTP Server version 1.3.19. Read the announcements.txt file as it contains important information as well as possible security changes. Also, be sure to view the change_1.3 file as it contains additional security information. After you have downloaded the software then go to http://httpd.apache.org/docs/ for general documentation. The installation instructions are at http://httpd.apache.org/docs/install.html . The documentation will explain how to compile and install Apache On UNIX platforms. There is also an important security document located at http://httpd.apache.org/docs/misc/security_tips.html.

Installation and File Structure Concerns - Root Exploit
The following file protection settings are recommended by the Apache documentation links mentioned above. Their purpose is explained here in further detail.

ServerRoot is a directive which allows you to set the location where the Apache server will reside. The Apache HTTP Server documentation suggests you set the ServerRoot to the /usr/local/apache directory and set these directory protections:

chown 0 . bin conf logs
chgrp 0 . bin conf logs
chmod 755 . bin conf logs

The chown and chgrp commands are being used to set the ownership and group of the listed directories to root. This allows root to search, alter, and change to those directories. Additionally, these settings allow non root users only search capabilities on these directories The chmod command is being used to allow only root to modify the files in these directories. It is important never to allow others to have write access to these directories or you will open the server to root compromise.

The documentation also suggests that the /usr/local/apache/bin/httpd directories should be protected as follows.

chown 0 /usr/local/apache/bin/httpd
chgrp 0 /usr/local/apache/bin/httpd
chmod 511 /usr/local/apache/bin/httpd

Again the chown and chgrp commands are setting the ownership and group of the httpd directory to root. The chmod command is allowing root to search and change to that directory. Only search privilege is given to non root users.

Following is a detailed description of the UNIX file ownership and protection commands.
chown: Used to change the owner of a file. Entering chown 0 *filename* will change the owner to root.
chgrp: Used to change the group of the files and the chmod command will change the protection of the file
chmod: Modifies the permissions on a directory or file.
Important: The chmod command works differently for directories and files. On a file there are read, write, and execute permissions (rwx). On a directory the read bit allows list capability,

the write allows changing of files, and the execute bit allows execution of programs.
Example: chmod 755 bin conf logs
The value 755 is the octal number representation. A value of 4 is list, a value of 2 is change, and a value of 1 is search. In this case the value 7 means that the owner has list, search, and change capability, all others have list and search.

Main Configuration Files – General Security
There are three main configuration files for the Apache HTTP Server and they are typically located in the /usr/local/apache/conf directory. They are httpd.conf, srm.conf, and access.conf. These files are the control center for Apache. Understanding how the configuration files work is required, since most of the Apache configuration changes will be made in those files. The httpd.conf file is the primary configuration file. This is where most of the directives are utilized. The srm.conf files allow you to add resources to the web site and the access.conf file is where you set access permissions for your files. These configuration files use directives and parameters to control webserver behavior. These configuration parameters can be found http://httpd.apache.org/docs/mod/core.html.

Server Access Control - General Security
The access.conf file should contain the directives that control who is allowed access to the Apache directory structure. It should be set to *deny from all* initially and then modified to include the *allow from* directive as you determine who should be accessing your site. You may allow access from a domain, IP address, or range of IP addresses. This has similar functionality to TCP Wrappers.

Example of the access.conf file:

```
<directory /usr/local/http/docs/private>
        <limit>
            order deny,allow
            deny from all
            allow from sans.org
         </limit>
</directory>
```

Website Password Protection - General Security
The .htaccess file is where access controls to a particular site directory should be added. This .htaccess file should be copied to the directory for the site to be protected. There also needs to be an entry in the httpd.conf or srm.conf file utilizing the AccessFileName directive.

Example of the .htaccess file:

```
AuthName PrivateFiles
AuthType Basic
AuthUserFile /path/to/httpd/users
require valid-user
```

The following command structure will add a user to the password file. It needs to be executed every time a user is to be added to the protected site.

htpasswd -c /path/to/httpd/users username

Apache Log Files - General Security
The administrator has control over the information that will be added to the log file by using log format directives. The directive statement *LogFormat "%a %l"* will allow the logging of the IP address and hostname of the browser making the request. For security purposes you should plan to keep track of web site users that failed authentication. You may do this by including *LogFormat "%401u"* in the httpd.conf file. There are many additional parameters allowed with this directive. The point is to review the log file directive settings to be sure you have the information that you need for your web hosting purposes as well as your security requirements. Another important server log file is error_log. This file contains records of administrative events including server starts, stops and failed CGI execution attempts.

Security Related Directives - Buffer Overflow and HTTP DoS
Covered in this section are several of the server security related directives and their purpose. There are many directives available with Apache. They can be found on the Apache HTTP Server Project Web Site http://httpd.apache.org/docs/mod/directives.html.

Using following directives will help reduce the risk of DoS and Buffer Overflow attacks.

Denial of Service attack (DoS).
LimitRequestbody: Numeric parameter controlling maximum HTTP request body size.
LimitRequestFields: Numeric parameter that limits allowable number of request headers.
KeepAlive: Setting this parameter to off will disable a constant connection.
KeepAliveTimeout: Limits the time Apache will wait for additional requests.

Buffer Overflow attack.
LimitRequestFieldSize: Limits the size of each request header.
LimitRequestLine: Limits the length of each request line.

Common Gateway Interface Security Risks (CGI) - Root Exploit
CGI applications create vulnerabilities in several areas. Informational, execution of system commands through applications, and system resource usage. If a CGI program allocates memory statically then there is an opportunity for a buffer overflow attack on the system running that CGI script. To reduce this possibility the programmer should modify their CGI code to dynamically allocate memory. An additional measure of security is to utilize CGI wrappers such as suEXEC or CGI Wrap. These applications ensure that CGI applications are run under an individual users ID only so the only effect of a bad CGI program will be on that users directory files.

Perl, a powerful and flexible language designed primarily for text manipulation, allows programmers to utilize system calls. If these calls are not made carefully they will open up the server to entry from hackers. Perl scripts need to be checked carefully for these types of security holes. Adding routines to check for proper input strings will help to be sure that users are inputting valid data. Make sure the Apache Server is not running as root and restrict Perl scripts to a specific directory. Files should be protected from write access to that directory.

Server-Side Includes (SSI) – General and Root Exploit
SSI allows the programmer to create frequently used routines and include those routines into other code when needed. SSI also allows real-time content and conditional execution of external programs. This opens up the server to execution of malicious programs. Including the directive IncludesNoEXEC in the access.conf file will disable executable SSI files. As a result of using this directive, an include statement inside any code segment will not execute CGI scripts.

Additional Security Tools
Using TCP Wrappers and Tripwire will provide additional protection to the server. TCP Wrappers will allow you to control who has Telnet or FTP access to your server. Tripwire is an extensive product which allows you to set security policies to control file access. It has the ability to reports on files that were accessed or attempted to be accessed. You may configure Tripwire to monitor your web server configuration, website content, and CGI bins.

Summary
Apache is a solid server product that provides excellent stability. Careful installation and configuration of a UNIX system and Apache Server will provide an excellent platform for web sites to reside on. Following is a summary of securing the Apache Server.

- Install and upgrade to the most secure Apache versions.
- Check for proper directory and file permissions.
- Review httpd.conf, srm.conf, and access.conf for proper controls.
- Configure server log files for most complete information reporting.
- Add password protection to secure sites (.htaccess)
- Add CGI Wrappers (to restrict CGI activity to a users directory structure)
- Check Perl Scripts (for execution of system calls and form input checking)
- Check SSI directives for the allowing of external code execution.
- Add TCP Wrappers and Tripwire to the system Apache is running on

Print References

Arnold,Mark, Almeida, Jeff D, Miller, Clint, Administering Apache, McGraw-Hill, 2000 169, 241-245

Kabir,Mohamed J, Apache Server Administrator's Handbook, IDG Books Worldwide, 1999: 379-400

Web Site References

Marshall, Brad "An Introduction to Securing Apache", Wed, 11 Aug 1999 http://www.linux.com/security/newsitem.phtml?sid=12&aid=3549 (2 Apr. 2001)

Coar, Ken, Securing Your Web Pages With Apache", 2000 http://apache-server.com/tutorials/LPauth1.html (2 Apr. 2001)

Coddington, Dale, "Installing the Apache Webserver with SSL", Tue. 27th March 2000, http://www.securityfocus.com/focus/sun/articles/apache-inst.html (2 Apr. 2001)

Securing your Web server", Stop the wrong people from accessing your site! Make sure your site is secure from prying eyes and malicious intent. http://www.sun.com/sunworldonline/swol-06-1996/swol-06-webmaster.html (2 Apr. 2001)

Hontañón, Ramón, "Emerging Technology: Maximizine Apache Server Integrity", 17th Jul. 2000, http://www.planetit.com/techcenters/docs/security-hostile_content/technology_feature/PIT20000717S0011?printDoc=1 (2 Apr. 2001)

For Further Information

Apache Web Sites
www.apache.org
www.apachetoday.com

Other Books
Garfinkel, Simson / Spafford, Gene. Practical UNIX & Internet Security, Second Edition, , Incorporated, April 1996