Global Information Assurance Certification Paper

General Security Guidelines for an Apache Web Server on Solaris

Scott Tieman

Jun 24, 2001

Abstract

This document is based on security considerations for running an Apache web server on a Solaris 8 (Intel) platform. Although the operating system and web server are specific, the platform hardening concepts can be applied to support other Unix based platforms and applications.

Introduction:

Majority of web server vulnerabilities on a Unix platform are stemmed from open default services specific to any Unix host. The primary goal will be to limit network exposure of these services and provide knowledge and application of some valuable open-source security tools. Addressed will be Network-Based Security, Host-Based Security, Apache security elements, and position of the web server in relation to your network infrastructure.

Network-Based Security:

Network services are primarily generated from the Internet daemon and the run control scripts as the system goes through its run levels during boot-up. Many of these services have inherent vulnerabilities associated with them and are normally not required for the operation of a web server.

- **Internet daemon services or Inetd:** Located in the /etc/inetd.conf file. When a service is requested from another host, the Internet daemon intercepts the request and hands it to the appropriate server. The inet.conf file determines which services are available by whether there is a comment or not in front of the entry.

  - Example of an offered telnet service:

  telnet stream tcp nowait root /usr/sbin/in.telnetd in.telnetd

  - Example of telnet NOT being offered:

  #telnet stream tcp nowait root /usr/sbin/in.telnetd in.telnetd

Securing these services are simply accomplished by editing this file and "commenting out" the entries. Note: After commenting

them out, a kill hup signal must be sent to the Inetd process for it to take effect.

- **Services started in Run Control scripts:** Located in the /etc/rc*.d directories. The asterisk refers to the number of the corresponding run level. These files, contained within the directories, beginning with **"K"** are ran to terminate (Kill) a process. Files beginning with **"S"** are ran to (start) a system process. Scripts are ran in numerical order. The number following the first letter designates the order in which the scripts are ran.

  - Example: **ls /etc/rc2.d**

    S71rpc
    S75cron
    S88sendmail

One way to secure services not required, even when the system is rebooted, is to edit the corresponding services within the rc*.d directories. Change directories to the appropriate rc*.d and rename the "S" to lowercase "s" for the particular service you want to stop when system is initialized at boot-up.

- Example: **mv S88sendmail s88sendmail**
  ls /etc/rc2.d

  S71rpc
  S75cron
  s88sendmail

- **Verifying running services:** There are several ways to verify what services are running on your platform. They range from simple commands to open source ports scanners launched from another host. Remember the primary goal: "limit network exposure" from the outside world.

Commands:

netstat –a Displays all TCP/UDP connections you are running. TCP connections that are offered will be denoted with "LISTEN" while UDP connection are denoted as "IDLE".
rpcinfo –p Displays all the Remote Procedure Calls (RPC) programs and what port they are running on.
Port Scanners:

NMAP is an extremely versatile port scanner from http://www.insecure.org/ with many options. At a minimum you should run the following commands.

```
./nmap -sT <IP>  #This option will scan TCP ports.
./nmap -sU <IP>  #This option will scan UDP ports.
./nmap -sR <IP>  #This option will scan for rpc ports.
```

Host-Based Security:

There are many aspects to host-based protection. I will focus on one that is packaged with Solaris and other open source tools, which are similar in functionality that look for specific host vulnerabilities.

- **ASET or Automated Security Enhancement Tool:** ASET is Sun's packaged host-based security tool. It was originally packaged with Solaris 2.5 and continued its support to Solaris 8. It contains three preset levels of security features: Low, Medium, and High. At each level, the degree of security is implemented, ranging from file permissions of key files to certain network services not being offered. More information can be found in the man pages or at http://www.sun.com/smcc/solaris-migration/docs/transition-guide_2.5/security.html

- **COPS or Computer Oracle Password System:** This security tool was written by Dan Farmer and although similar to ASET, is has been tested on a variety of Unix flavors. It was written in C and individual shell scripts designed to look at a multiple host-based issues. Note, COPS does NOT actively set policies to your system, they are only suggestions. What may be a security problem to Dan Farmer, could be necessary for your systems operation.

  - Example COPS Report:

  ATTENTION:
  Security Report from WED 15:15:28 CDT 2001 from host <Host Name>

  Warning! /etc/security is _world_ Readable!

  COPS can be downloaded from Dan Farmer's actual site at http://www.fish.com/cops/

- **Tiger:** Another host-based security tool developed by Texas A & M University (TAMU). Was designed to allow access to certain hosts from off the compass. The host would have to be secured through the security suggestions from Tiger. Designed for SunOS 4.x and 5.x, with partial checks with other Unix flavors. Care must be taken in implementing these suggestions, keeping in mind the purpose of this security tool.

  - Example Tiger report:

  --warn—[permw] /etc/security should not be world readable.

  Tiger can be downloaded from TAMU at http://www.net.tamu.edu/ftp/security/TAMU/

- **Tripwire:** This tool can be classified as a file integrity security tool. It was designed by Gene Kim and Gene Spaford from Purdue University, but the management was commercialized by Tripwire Inc. They are still required to maintain a free version of this product as agreed by Purdue. The free version follows the acronym ASR for Academic Source Release. Basically, this tool compares a predetermined set of files and directories against a previously stored database. It uses multiple message digest and signature algorithms to support integrity of potentially manipulated files. Note, this tool does not prevent system intrusions, but merely reports that something has changed.

  - Example of Tripwire report:

Observed (What it is)	**Expected (What it should be)**
/etc/hosts.allow	
sf_size: 162	**170**
st_mtime: Sun Jun 10 17:34	**Fri Jun 9 16:04**
MD5(sig1) Ki8G54!O9XMlws2K5kL4SQ	**7iGG5N3OQX9l7s2KhzLlY4**

  This indicates that the file size and time stamp was altered on the /etc/hosts.allow file. Also the Message Digest 5 is indicating a different signature. Tripwire can be downloaded from http://www.tripwire.com/downloads/

  Additionally, Tripwire supports a trial "Web page" version specifically for the Apache web server which alerts when the web pages are being altered.

Apache Security Issues:

Apache is comes in two distributions, source code and binary. The latest version is freely available at http://httpd.apache.org/dist/httpd/ Although there is many security aspects to Apache, this section will focus on Server configuration within the httpd.conf file, Host-based access, Host-based authentication, and logging.

- **Httpd.conf:** Apache is primarily configured through this file and manipulated/controlled with multiple directives. Completed documentation can be found at http://httpd.apache.org/docs/

- **Host-Based Access:** Location for host-based access/authentication can be accomplished in two locations: the .htaccess file which resides within the particular directory you are limiting access or the httpd.conf file. Each have pro's and con's.

  .htaccess

  Pro: - Server does not have to be re-started after updating the file.
  - If you need to relocate directories within your server, your .htaccess file containing your access controls will move with them.

  Con: - No centralized location. Might have to access multiple directories.

  Httpd.conf

  Pro: - All access controls reside in centralized location.

  Con: - Each time the file is updated, the server must be re-started before it takes effect.

  Note: Regardless of the location, the syntax/content remains the same. Host-Based access controls are normally performed through container directives. The following are the most popular ones utilized which control access to for specific HTTP methods, directory and file access that resides within your HTML pages.

<Limit> Restricts directives that are contained within a particular HTTP method such as "get" or "put". These methods are contained within the HTTP request header from the client (browser) to the server (Web Server). Get, informs the server that the client is requesting to retrieve information, such as a document or a HTML page. Put, informs the server to store something, such as a file.

- Example of limit Directive:

```
<Limit GET PUT>
order deny, allow
deny from all
allow from 199.211.200.131>
</Limit>
```

<Directory> This directive is applied to directories and sub-directories.

- Example of Directory Directive:

```
<Directory /local/etc/html/trusted>
order deny, allow
deny from all
allow from <199.211.200.131>
</Directory>
```

<File> This directive is applied to files and utilizes the same syntax as the <Directory> directive.

Note: Two container directives used together to support Host-Based access:

```
<Directory /local/etc/html/trusted>
<Limit GET PUT>
order deny, allow
deny from all
allow from 199.211.200.131>
</Limit>
</Directory>
```

- **Host-Based Authentication:** Authentication refers to login name and password entries to gain access to a specific location on the server. Apache utilizes the <AuthType>, <AuthName>, and <AuthUserFile> directives.

<AuthType> Defines the authentication mechanism the server will use.

- Basic = Utilized the most. However, information is passed in the clear and is susceptible to sniffing unless utilized with SSL.
- Digest = Uses a Message Digest (MD5) for authentication.

<AuthName> Defines the label that is passed to the Authentication Directive. Also appears on the login prompt as well.

<AuthUserFile> Defines the absulute path whare the authorized users and their passwords are located.

-Example of Authentication Directives:

```
<Directory /local/etc/html/trusted>
AuthType Basic
AuthName Users
AuthUserFile /usr/local/priv_users

<Limit GET PUT>
Require valid-user
</Limit>
</Directory>
```

This example would require a valid user name and password to access the directory trusted. The File Directive could have easily been used instead of the Directory Directive.

Creating valid users. Apache uses the htpasswd utility to create valid accounts that the Authentication Directives require.

- Example of the htpasswd command:

```
htpasswd -c /usr/local/priv_users bob
```

This would create a user called bob and you would be prompted next to enter a password for the user. The "-c" will automatically create the file name priv_users.

- **Logging**: Monitoring logs that Apache generates is a valuable asset for tracking server access and potential server problems. By default, Apache generates two log files:

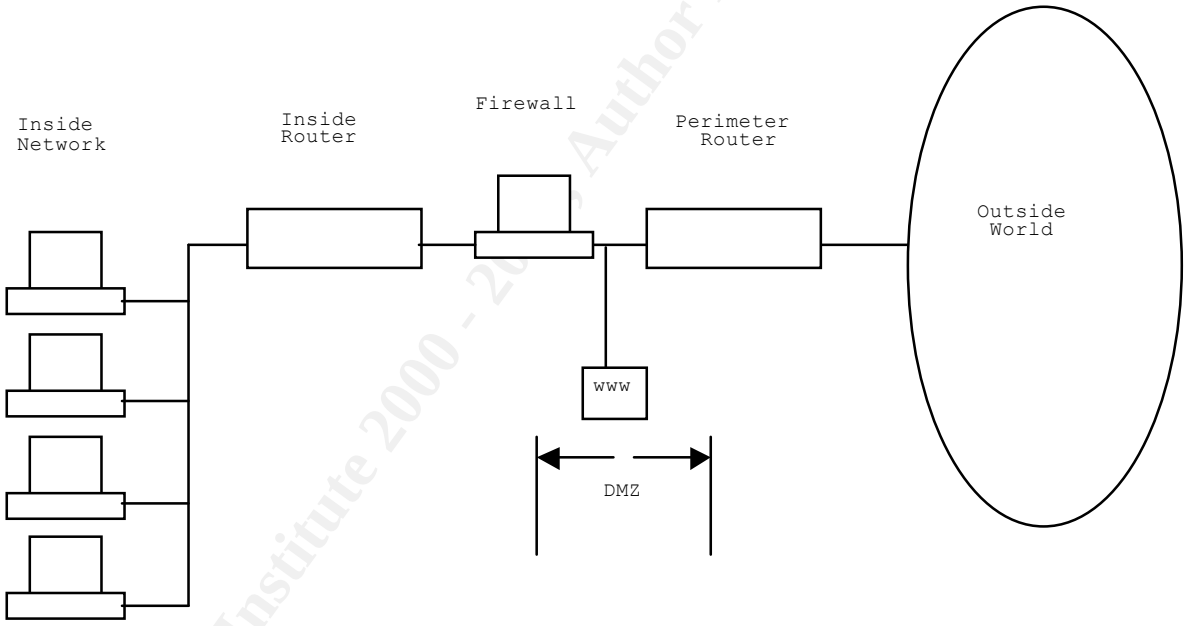
- Access_log = Tracks all HTTP connections by IP and provides a time stamp.

- Error_log – Tracks server related events such as re-starts or general server problems.

Note: Additional security considerations can be found at http://httpd.apache.org/docs/misc/security_tips.html

Web Server Location:

Assuming you are working with a publicly accessed server, it is imperative of the placement in relation to your firewall. The following diagram illustrates a typical "Dual-Honed" firewall topology.



- **Topology Features:** Provides defense in depth (3 layers). Perimeter and inside routers would utilize Access Control Lists (ACL's) with the firewall provides 3 layers of protection.

- **DMZ:** De-Militarization Zone. Area between the perimeter router and the firewall. If the public web server is compromised, protection exists with the firewall itself.

Conclusion:

To reduce your percentage of vulnerabilities associated with your platform and web server, you must proactively keep ahead of

vendor patches and monitor various security alerts. The following URL's will assist you in keeping abreast of security related issues:

Security related information for Sun products and vendor patches:
http://sunsolve.sun.com/

Patches related to Apache:
http://httpd.apache.org/dist/httpd/patches/

References:

1. Gregory, Peter. "Solaris Security". Sun Microsystem Press/Prentice Hall, Inc. 2000. Page 124-128.

2. Fydor, "NMAP Man Page". URL:
http://www.insecure.org/nmap/nmap_manpage.html

3. Calkins, Bill. "Solaris 7 Administrator Certification". New Riders, 2001. Page 174-176.

4. Farmer, DAN. "COPS Overview". URL:
http://www.fish.com/cops/overview.html 18 May 1993.

5. Texas A & M University, "Tiger Readme Page". URL:
http://www.net.tamu.edu/ftp/security/TAMU/tiger.README 19 Jul 1999.

6. Beale, Jay. "Tripwire - The Only Way To Really Know". URL:
http://www.securityportal.com/topnews/tripwire20000711.html

7. Wainwright, Peter. "Professional Apache". Wrox Press Ltd. 1999. Page 345-352.

8. Oppliger, Rolf. "Internet and Intranet Security". Artect House, Inc. 1998. Page139-140.

9. Goncalves, Marcus. "Firewall Complete". McGraw-Hill. 1998. Page 46.