International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 8, October - 2012

Web Server Security

[1] K.Rajiv [2] A.Prashanthi [3] Ch.Bharadwaja

ABSTRACT

Web Server security comes to being from confidentiality, in-tegrity, availability of appropriate information and authentication. A leaky server can cause a vital harm to an organization. So, security is the most complex topic that the modern world is concerned about. A security breach in-curs a cost for the organization in terms of money as well as goodwill. Databases store confidential and sensitive in-formation. Hence, it is the most important task of an organization to safeguard crucial information from being stolen and misused. The general security issues of a Web Server and how security features are maintained in Apache are the points of focus in this paper.

1 INTRODUCTION

A Web Server is a software application responsible for fetching and serving web pages requested by a client. Web pages may be stored in the host computer or another computer's hard drive.

The general goal of network security is to keep strangers out. Yet the point of a Web site is to provide the world with controlled access to the network [1]. Drawing the line can be difficult. A poorly configured Web server can punch a hole in the most carefully designed firewall system. A poorly configured firewall can make a Web site impossible to use. Things get particularly complicated in an intranet environment, where the Web server must typically be configured to recognize and authenticate various groups of users, each with distinct access privileges.

Apache is the most widely used web server. 40 mil-lion sites now running on the Apache server, which powers nearly 70 percent of web sites. Understanding the approach of Apache towards security can help us make other applications secure.

This paper is organized as follows: in section 2, we have presented the general security issues of Web Servers. In section 3 we have gone into depth and discussed how these security features are implemented in Apache. Finally, we match the security features of Apache with various security patterns.

2 SECURITY ISSUES

Web Servers are very attractive targets for attackers and that's why security is an essential topic for administrators of both internet-connected and intranet servers. In this section we discuss the general security issues of a web server.

2.1. Integrity and Privacy

A web server is a gateway through which the entire internet population can peek into. To prevent common attacks of content corruption and data theft, integrity and privacy is the primary issue.

2.2. Common Gateway Interface (CGI) script

CGI scripts are programs that run on the Web Servers in real time. They handle different inputs from the users and therefore get inputs from web browsers, access database and can return information to the client browser. CGI scripts are like miniature servers. So, a buggy script can be a potential attack target. CGI scripts can present security holes in two ways:

- They may show such information from the host that may help attackers break through the server.
- The user inputs may be tricky enough to be executable commands and do unwanted harm to the host machine.

2.3. Access Control

Access Control is, who is allowed to view what on a Web Server and more specifically to execute what in case of CGI scripts.

2.4. Data transmission through TCP/IP

The TCP/IP protocol was not designed with security in mind. Hence it is vulnerable to network eavesdropping. When confidential documents are transmitted from the Web server to the browser, or when the end-user sends private information back to the server inside a fill-out form, someone may be listening in.

2.5. Denial-of-Service (DoS)

Attacks in which the intruder's goal is to shut down the tar-get rather than steal data are called DoS attacks[2]. In these network-based attacks authorized users are denied the use of network services. DoS attacks come in variety of forms and aim at a variety of services such as the consumption of scarce, limited, or non-renewable resources, destruction or alteration of configuration information and physical de-struction or alteration of network components.

2.6. Ability of Web Server

A web server with limited features is better. Simple servers that do little more than make static files available for re-quests are probably safer than complex servers that offer such features as on-the-fly directory listings, CGI script execution, server-side include processing and scripted error handling.

2.7. File Permission

There are two file systems roots working in a server, namely, the document root which has all the HTMLs and the server root having all the logs and configuration files. It is important to get the permissions right for the server side root because this keeps all the sensitive information and also the CGI scripts.

3. APACHE SECURITY ISSUES

While discussing about security features of Web Servers, Apache is the best option to study because of its easily being customized by design. Apache is the number one choice for a Web server for both Linux and FreeBSD. It is the most widely used Web Server on the Internet because of its standards compliance, scalability, dynamic shared objects, customizability, programmability[3] and more. Here we will discuss what the security issues in Apache are and how they are implemented.

3.1. Securing against CGI

Three ways to secure against CGI[4] are discussed below:

3.1.1. The ScriptAlias approach

The first step in securely configuring CGI under Apache is to create a central directory to store CGI applications in. This directory should always be separate from the DocumentRoot tree. It also ensures only Web administrators can access the files that reside there. Then Apache is informed which directory contains CGI programs using the ScriptAl-ias directive.

```
ScriptAlias /cgi-bin/ /www/mysite/cgi-bin/
```

ScriptAlias designated directories are not able to be browsed by default for security reasons. Ideally, no one but the lead CGI developer and the system administrator should have full access on the files contained by the directory referenced by ScriptAlias.

3.1.2. The Alias/AddHandler approach

The AddHandler directive is used to run CGI scripts from arbitrary directories. To allow CGI program execution for any file ending in .cgi in users' directories, the following configuration can be used:

```
<Directory /home/*/public_html> Options
+ExecCGI
AddHandler cgi-script .cgi
</Directory>
```

3.1.3. Reducing CGI risks with wrappers

A wrapper allows CGI applications to be run under the user ID of the site owner, i.e. the owner of the directories and documents that comprise a Web site. Wrapping CGI applications restricts the damage a user can do to the user's files alone. Most CGI wrappers perform additional security checks before they allow a requested application to execute. Two popular CGI wrappers are:

- suEXEC: When a request is made for a CGI or SSI file not owned by the Apache user, the request is passed to suEXEC with the program name and the owner's user/group ID. suEXEC then runs a series of checks to ensure the request is valid. If it is, the script is ex-ecuted. Failure in any of the checks causes the script not to run and an error is logged.

- CGIWrap: CGIWrap is similar to the suEXEC pro-gram in that it permits user access to CGI programs without the risk of compromising server security. It does this by running any program defined as a CGI application as the file owner rather than the Apache user. CGIWrap also performs several security checks on the CGI application; the application will not be executed if any of the checks fail.

3.2. Securing against Server Side Includes

SSI are directives that are placed in HTML pages, and evaluated on the server while the pages are being served. They let adding dynamically generated content to an existing HT -ML page, without having to serve the entire page via a CGI program, or other dynamic technology. SSI presents a server administrator with potential risks[4]:

- First of all the risk is of ‘Increased Load on Server’. All SSI enabled files have to be parsed by the server whether there are not any SSI directives included within the files.
- Secondly, using the ‘exec cmd’ element, SSI-enabled files can execute any CGI script or program under the permissions of the user and group Apache runs as, as configured in httpd.conf. For example:

```
<pre>
<!--#exec cmd="ls" --> </pre>
```

  Or on Windows

```
<pre>
<!--#exec cmd="dir" --> </pre>
```

  This feature is exceedingly dangerous, as it will execute whatever code happens to be embedded in the exec tag.

There are actually three ways to enhance security and also getting the advantages of SSI:

- To isolate the damage a wayward SSI file can cause, a server administrator can enable suEXEC.
- The extension of SSI enabled files should be different, such as conventional .shtml. This helps keep server load at a minimum and allows for easier management of risk. One disadvantage to this approach is that if SSI directives are to be added to an existing page, the name of that page, and all links to that page have to be changed, in order to give it a .shtml extension, so that those directives would be executed. The other method is to use the XBitHack directive:

```
XBitHack on
```

  XBitHack tells Apache to parse files for SSI direc-tives if they have the execute bit set. So, to add SSI directives to an existing page, rather than having to change the file name, the file has to be made executable using chmod.

```
chmod +x pagename.html
```

- Another solution is to disable the ability to run scripts and programs from SSI pages. To do this ‘Includes’ is replaced with ‘IncludesNOEXEC’ in the ‘Options’ directive. Users may still use:

```
<--#include virtual="..." -->
```

  to execute CGI scripts if these scripts are in directo-ries designated by a ScriptAlias directive.

3.3. Memory and Resource Management: Apache Pool

Apache provides an own memory and resource management known as pools. Pools are means to keep track of all re-sources ever used by Apache itself or any add-on module used with Apache. A pool can manage memory, sockets and processes, which are all valuable resources for a server system.

With the pool concept, a developer registers any memory, socket or process with a pool that has a predetermined lifetime. Once the pool is destroyed, all resources managed by that pool are released automatically. Only a few routines that have been tested thoroughly will then take care of freeing all resources registered for this pool. That way only a few routines have to make sure that all resources are freed. Mistakes within those are a lot easier to find and this technology takes a burden of all other developers.

3.4. Intrusion Detection: Apache Log

In order to effectively manage a web server, it is necessary to get feedback about the activity and performance of the server as well as any problems that may be occurring. The Apache HTTP Server provides very comprehensive and flexible logging capabilities. Log files show what actually is going on against the server. Though the log files show what has already happened, they give a view of understand-ing of what attacks is thrown against the server and allows checking if the necessary level of security is present[6].

- The access log stores the IP address of the client accessing the system as well as the file retrieved.
- The agent log details the client programs that were used to access the server.
- The error log lists server errors. Keeping track of these logs can be helpful in identifying attempted intrusions. Penetration techniques can also be gleamed from the logs.
- The refer log indicates the URL that the browser pre-viously visited and that of the URL that the browser is currently viewing.

Apache provides a number of modules to support logging. Some of them are:
mod_log_forensic: This module provides for forensic logging of client requests. Logging is done before and after processing a request, so the forensic log contains two log lines for each request.
mod_logio: This module provides the logging of input and output number of bytes received/sent per request.
mod_log_config: This module provides for flexible logging

of client requests. Logs are written in a customizable for-mat, and may be written directly to a file, or to an external program.

3.5. Access Control

Apache has the module mod access to provide access control. mod access as its name clearly implies, provides access control for documents. It allows one to restrict or allow access to resources based on the client's host name, IP ad-dress, or network address. The IP based control is known as Mandatory Access Control (MAC) and password based control is known as Discretionary Access Control (DAC)[5]. The directives provided by mod access are:

- Allow Directive: Controls which hosts can access an area of the server. Examples are:
  A partial domain name: Allow from apache.org
  A full IP address: Allow from 10.1.2.3
  A partial IP address: Allow from 10.1
- Deny Directive: Controls which hosts are denied access to the server. The syntax is:

  Deny from all
- Order Directive: Controls the default access state and the order in which Allow and Deny are evaluated. Ordering is of: Allow, Deny /Deny, Allow/ Mutual-failure. In the following example, all hosts in the apache.org domain are allowed access; all other hosts are denied access.

  ```
  Order Deny, Allow
  Deny from all
  Allow from apache.org
  ```

3.6. Authentication

Apache provides a number of modules to support authentication. They are:

mod auth: User Authentication by User Name/Password. Any module that is called during the authentication phase must verify the identity of the requester, based on credentials presented by the user. The authentication module deals with the HTTP Basic Authentication facility, which is simply a user name/password pair submitted by the client together with the request for a document. This module allows one to check this information against a flat file database similar to UNIX's /etc/passwd and /etc/group files and to deny access when the given user name/password doesn't match the database information. Special variants of this module exist that offer the same functionality but use a database from other than a flat file (for performance reasons).

mod auth anon: User Authentication by Anonymous Na-me/E-Mail Address.

mod auth dbm: User Authentication by User Name/ Pass-word (UNIX NDBM).

mod auth db: User Authentication by User Name/ Pass-word (Berkeley-DB).

mod digest: Message digest-based authentication mechanism. In addition to the classical HTTP/1.0 Basic Authentication mechanism, a message digest-based[2] HTTP authentication mechanism exists as defined in RFC 2617. Instead of transferring a clear-text user name/password pair with the HTTP request (which can be easily monitored), a message digest is calculated (via the MD5 algorithm) and transferred together with the user name. This module then performs the same message digest calculation for the pass-word stored in the server's authentication database. When the two digests are equal, access is allowed. This approach offers an obvious advantage relative to basic authentication, as the password is not sent over the network.

4. CONCLUSION

In this paper, we have presented the security issues of web server and how these features have been implemented in the Apache HTTP Server. A closer look at the implementation of the security features reveals that they closely resemble with various security patterns [7]. mod access deploys the pattern Single Access Point to provide password based access control mechanism. The IP based access control mechanism is based on the Roles pattern. Moreover, the authenticated users can see their files only, i.e. the security pattern Limited View is used here. The practices that Apache uses to be secured against CGI, SSI, intrusion and memory leaks can be repeated to make other programs secure and efficient.

5. REFERENCES

[1] L. D. Stein, "Web Security: A Step-by-Step Reference Guide," Addison Wesley Professional, 1998.

[2] A. S. Tanenbaum, "Computer Networks, Fourth Edition," Pearson Education Inc., 2003.

[3] C. Aulds, "Linux Apache Web Server Administration, Second Edition," Sybex Inc., 2001.

[4] J. P. Sousa and D. Garlan, "Apache HTTP Server Version 2.0 Documentation," The Apache Software Foundation, 1995– 2005.

[5] J. Pieprzyk, T. Hardjono and J. Seberry, "Fundamentals of Computer Security," Springer-Verlag, 2003.

[6] S. Chanson, "Internet Security Handbook, Edition Two" The Hong Kong University of Science and Technology, June 2001.

ISSN: 2278-0181
Vol. 1 Issue 8, October - 2012

[7] J. Yoder and J. Barcalow, "Architectural Patterns for Enabling Application Security" Pattern Languages of Programs, 1997.

AUTHORS

K.Rajiv Working as an Assistant Professor in **Nalla Narsimha Reddy Educational Society of Institutions.** Done his M.Tech in 2010 and B.Tech in 2008.

A.Prashanthi Working as an Assistant Professor in **Nalla Narsimha Reddy Educational Society of Institutions**. Done her M.Tech in 2010 and B.Tech in 2007.

Ch.Bharadwaja Working as an Assistant Professor in **Nalla Narsimha Reddy Educational Society of Institutions**. Done his B.Tech in 2010.