Web Authentication Security

Donna Selman

Web Authentication Security
Donna Selman (Version 1.4b)
06 June 2003

Abstract

This document will cover four web authentication security techniques that are used by web server administrators to provide web browser clients access to the file systems on their host computers. These four authentication techniques are Basic Authentication, Digest Authentication, Database Authentication, Anonymous Authentication, and N-Tier Authentication. Because these four techniques are basic to security, they can, and often are, utilized on almost any web server. However, the web server referenced in this document will be Apache 1.3 HTTP Web Server. First, there is a brief introduction to web servers in general. Next, there is an introduction to the Apache Web Server in particular. Finally, all four of the web authentication techniques will be discussed. It is hoped that this document will provide enough detail to serve as a supplemental security how-to guide on using web authentication security.

Introduction

During my research, I discovered, although wasn't surprised, that authentication is one of the most import security concerns in web server administration. Although I have worked in the computer field for over seventeen years, I am new to the field of web server administration. As such, I needed to use a number of different resources in order to finish this paper. A comprehensive list of those resources can be found on the last page. The one resource that was the most useful in providing step-by-step procedures is an article titled "Authentication, Authorization, and Access Control".[1] It can be found on the main Apache Web Site.

What is a web server?

According to Brad Bell, "the definition and purpose of a web server is a software package that serves either static content to a Web browser at a basic level, or dynamic content that require end-user interaction." [2] This specialized software resides on a computer. The computer itself is usually, although not always, located in-between the web browser clients and the back-end database server. When a user types in a request for a web page, the web server software maps the requested URL to a file on the host server. Then the web server loads the file from the host server's disk to the user's web browser. The protocol used to facilitate this form of 'communication' between a user's browser and the web server is called Hypertext Transfer Protocol (HTTP).

Originally the web server's primary function was to serve static content pages written in HTML, and later XML, along with image files to a web browser client. But as the Internet became more popular, and more easily accessible, this main function of the

[1] http://httpd.apache.org/docs/howto/auth.html
[2] http://www.sans.org/rr/web/popular.php

web server has been changing to include the serving of dynamic web page content as well. This dynamic content is often in the form of web-based applications. There are several different web servers available on the Internet. Some of the more popular ones are Apache HTTP Web Server, Microsoft's Internet Information Server (IIS), and Netscape's Enterprise Server.

What is Apache?

The Apache HTTP Server Project is a project of the Apache Software Foundation. "It 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." [3] The Apache Software Foundation provides support for open-source software projects. Apache is an open-source HTTP server. "Because it is a 'freeware' product, it is more widely used than all other available web servers combined." [4]

Apache is one of the most popular and powerful web servers on the Internet today. Apache offers the latest protocols, including HTTP/1.1. The Apache Web Server utilizes both the Common Gateway Interface (CGI) and JavaScript to retrieve page content from a host computer or to run an application program. CGI is "the most commonly used standard to serve customized, dynamic content pages…which define how the web server should run programs locally and transmit the output to the requesting web browser client." [5] The Apache Web Server software is extremely customizable and highly configurable because it utilizes both 'core' modules and several modifiable third-party modules. It allows for further customization by providing for the addition of user modules that can be created using Apache Application Program Interface (API).

The Apache Web Server software has been successfully installed on almost every major computer hardware platform in existence. It is compatible with almost all operating systems including Windows NT/98/95, NetWare, OS/2, UnixWare, and most flavors of Unix.

What is authentication? What is authorization? What is web server access control?

Authentication is when a web client makes a browser request for a particular resource, it is absolutely essential for the web server to be able to verify that the user is in-fact who or what they claim to be. This authentication must take place before allowing them access to any of the files on the host server. The most common method of authentication being utilized today is a combination of username and password. It is assumed that knowledge of the correct password will validate that the user or process is authentic. The main weakness in this method is that passwords can be easily guessed, compromised or forgotten.

[3] http://httpd.apache.org/ABOUT_APACHE.html
[4] Theriault and Newman, p.413.
[5] Theriault and Newman, p. 415.

Authorization is after the web client has been authenticated, then the next step is to check and see what resources the web browser client can be given during that session. Before authorization can take place, the web server administrator must have set up permissions on the server. The initial setting up of web server permissions, and the actual checking of them each time a resource is requested by a web browser client is called authorization.

Although not discussed in this paper, another equally important web security technique is web server access control? Unlike authentication and authorization, access control is less user specific and more system based. It is established during the configuration of the server itself. "Access control is the application of some other, and usually unrelated, criteria to control access. This can be your network address, the time of day, or the phase of the moon." [6] Access control may or may not have anything to do with the actual web client making the request for the resource.

What is Basic Authentication? How is it configured? How secure is it?

Basic Authentication uses the mod_auth module. Basic Authentication is the simplest and up until recently the most common security method utilized by web server administrators. Due to client password issues and concerns, other methods are now becoming more common. When a web browser client makes a request that requires authentication, a '401 – Authentication Required Message' including the 'realm' that the client requested is sent back to the client's browser. The client is then prompted for a valid username and password. If both are correct, then the web resource is returned.

HTTP is 'stateless'. So unfortunately each time a web browser client makes a resource request, they would again be prompted for the same information. However most browsers (including latest versions of Internet Explorer and Netscape) will make use of 'caching' where they can temporarily store the web browser client's username and password. This information is only available during the current session. The cached information is passed to the HTTP server during future requests.

Basic Authentication is configured by first creating a password file, then letting the web server know that a password file exists, and when necessary, by creating a group file in which to add web browser clients.

To create the password file, you would use the htpasswd utility that comes with Apache and is located in the Apache bin directory. The password file will contain a listing of valid usernames and passwords. The passwords will be stored encrypted. However, this file should NOT be stored in the same document directory where the web browser clients' resources exist.

[6] http://www.serverwatch.com/tutorials/print.php/2202671

If there is 'not' an existing password file, then use the following to create a 'NEW' password file and add user1 to it. If there is already a password file, then using the –c flag will delete the existing file and create a new file with only one user in it.

Here are the four steps to create a new password file by the name of passwds, which will be located in the directory named security that we will also create:

cd $ORACLE_HOME/Apache/Apache
mkdir security
htpasswd –c $ORACLE_HOME/Apache/Apache/security/passwds user1
Type in password for User1, twice, when prompted to do so.

You should now have a file by the name of passwds in the security directory with one user in it named 'User1'.

To add users to an 'EXISTING' password file named passwds in our security directory, do the following:

cd $ORACLE_HOME/Apache/Apache/security
cp passwds passwds.backup.mmddyyyy
htpasswd $ORACLE_HOME/Apache/Apache/security/passwds user2
Type in password, twice, for user2 when prompted to do so.

You should now have a file named passwds with a new user named 'User2' added to the bottom of the file.

Note: If you should accidentally wipe out the password file by using the –c flag, you should be able to recover if you created a backup file using the command in step 2. It would be a good idea to replace mmddyyyy with current month, day and year.

Even though the passwords are encrypted, you should make the file as secure as possible by changing the permissions. The owner of the file (i.e., root or oracle) should have write permissions. And the web server process should only have read permission.

To change the permissions on the password file with oracle as the owner, and apache as the group, use the following three commands:

chown oracle $ORACLE_HOME/Apache/Apache/security/passwds
chgrp apache $ORACLE_HOME/Apache/Apache/security/passwds
chmod 640 $ORACLE_HOME/Apache/Apache/security/passwds

To make the configuration changes to tell Apache to use the password file, you will need to use the following five directives:

AuthType should be set to the type of authentication
AuthName should be set to the authentication realm or name

AuthUserFile should be set to the location of the password file
AuthGroupFile should be set to the location of the group file (optional)
Require should be set to the requirements that must be satisfied

An example of a directive would be:

AuthType Basic
AuthName “Club Members Only”
AuthUserFile /u01/app/oracle/product/8.1.7/Apache/Apache/security/passwds
Require user user1 user2
-or-
Require valid-user (if you want to grant access to any valid user)

In our example above, the phrase “Club Members Only” will be displayed in the password pop-up box displayed when web browser client is prompted for their username and password. Please note that that the Require Directive uses valid-user (not valid_user) as a parameter.

The five directives listed above may be put into either the .htaccess file or into the Apache configuration file. If they are put in the .htaccess file, then the “server does not have to be re-started after updating the file, (and) if you need to relocate directories in your server, your .htaccess file containing your access controls will move with them. If the directives are put into the main Apache configuration file httpd.conf, then all access controls reside in centralized location, but the web server will need to be restarted for the changes to take effect.” [7]

If you want a particular group of web browser clients to have the same access to a resource, then you can create a group file. This group file should be placed in the same location as the password file. By using a group file, you can add and remove names and not have to restart Apache each time a change is made. This group is called an authentication group. The format of the group file is simple. The name of the group appears first, and then a list of the members in the group separated by spaces.

Type the following commands to create a group called coffee group under our grps directory, which is under our directory called security:

mkdir $ORACLE_HOME/Apache/Apache/security/grps

cd $ORACLE_HOME/Apache/Apache/security/grps

vi coffeegroup

insert the following text and then save the file

coffeegroup: donna shirley neoma

[7] http://www.sans.org/rr/unix/apache.php

After creating this group file, you would use the AuthGroupFile directive.

An example of this type of directive would be:

```
AuthType Basic
AuthName "Club Members Only"
AuthUserFile /u01/app/oracle/product/8.1.7/Apache/Apache/security/passwds
AuthGroupFile /u01/app/oracle/product/8.1.7/Apache/Apache/security/grps
Require group coffeegroup
AuthAuthoritative on
```

When a web browser client makes a request for a resource, the group file is checked first. If the name is in the group file, then the username and password is checked in the password file. If the name is not in the group file, or if the username and password are not the same as in the password file, then access to the resource will be denied.

Notice that we have set AuthAuthoritative to 'on' in our example above. By doing so, this has made these authentication directives 'authoritative'. This is not a problem when both the group file and password is available. But should something happens that either the group and/or password file is not available, then the web browser client will be denied access. If we set AuthAuthoritative to 'off', then the client could use another directive in the main Apache configuration file httpd.conf file to see if they can be given authorization to the requested resource.

Overall, Basic Authentication is 'not' very secure. Even if the password is stored in a protected directory on the server, it is passed from the client to the server in plain text across the network. The passing of both username and password will occur every time the web browser client makes a request. "Anyone listening with any variety of packet sniffer will be able to read the username and password in the clear as it goes across."[8]

What is Digest Authentication? How is it configured? Are there security concerns?

Digest Authentication is implemented by using the mod_auth_digest module. The older mod_digest module has become obsolete in that it is not compatible with most of the newer web browsers. Digest Authentication uses the htdigest utility instead of the htpasswd utility used by Basic Authentication.

The two utilities are similar except that a realm is also required when using htdigest. Other than that, the two utilities functions are very similar. Another difference is a security enhancement in which the passwords are MD5 encrypted. So when the web browser client connects to the web server, the password is not displayed in clear text.

The three steps for configuring your web server with Digest Authentication is:

[8] http://httpd.apache.org/docs/howto/auth.html

Create the password file
Configure the web server to use this password file
If needed, create the group file

To create the password file, you would follow the same steps that have been outlined in the Basic Authentication method except that you would replace htpasswd with htdigest. Remember to only use the –c parameter when creating a new password file.

For example, to create a new digest password file:

htdigest –c $ORACLE_HOME/Apache/Apache/security/digestpwds realm user1

As with Basic Authentication, Digest Authentication is done by using a directive.

Here is an example directive to use for Digest Authentication:

AuthType Digest
AuthName “Elite Members Only”
AuthDigestFile /u01/app/oracle/product/8.1.7/Apache/Apache/security/digestpwds
Require user user1
AuthAuthoritative on

The phrase “Elite Members Only” will be displayed in the pop-up menu. As in Basic Authentication, the use of a group file is optional. So in our example above we omitted the AuthDigestGroupFile. If you wanted to use a group, you would create it the same way as was done in the Basic Authentication. These digest directives can be placed in the .htaccess file or in main Apache configuration file httpd.conf . If they are placed in the web server’s configuration file, then you will need to restart Apache for the changes to take place. If they are placed in the web server’s .htaccess file, then they will take effect immediately.

“A limitation is that the digest files are keyed on the username; htdigest will not create multiple entries for user1 in different realms. The workaround is, that you have to store the realms in different password files, it you want the same user to belong to different realms”.[9]

Digest Authentication has great advantages over Basic Authentication. But it is also important to know that Digest Authentication is ‘not’ supported by all the major web browsers. So you must be sure that you are in an environment that will let you dictate the type of browser that your web browser clients will use. The web browsers that can be used with Digest Authentication are “Opera 4.0 or later, Microsoft Internet Explorer 5.0 or later, Mozilla 1.0.1 and Netscape 7 or later.” [10]

[9] Ball, Hundt, and Rasmussen, pg 9-18.
[10] http://httpd.apache.org/docs/howto/auth.html

Although Digest Authentication is more secure than Basic Authentication, there are at least two major security considerations. The first is that the only item that is encrypted is the web browser client's password. The data is still sent across the wire in plain text. The other security consideration is although only a digest version of your password is sent across the wire, the password can still be intercepted by someone using a sniffer. If this person is good with decoding HTTP, they can use the decoded password to gain access to your web site.

What is Database Authentication? How is it configured? Are there security concerns?

Both Basic Authentication and Digest Authentication store their information in text files. These text files are read every time a web browser client uses HTTP to make a request. Aside from not being very secure, the reading of text files can cause the web server to slow down. In fact, if the text files get big enough, even if a valid username and password combination is used it can be rejected due to Apache timing out.

To address this issue, HTTP Server now comes with a PERL script utility that is used to create and maintain DBM databases. Other web servers use DB databases. The name of the utility is dbmmanage for both types of databases. The two modules used by Database Authentication are mod_auth_db and mod_auth_dbm. The DB and DBM Database files created are the same. Your platform will dictate which one is available. In the following examples, DBM is used. But you can substitute DB for DBM in any of these commands.

Neither the mod_auth_dbm or mod_auth_db module is compiled by default. To use Database Authentication with Apache, you will need to execute one of these two configuration commands. It should be executed in your Apache 'source' directory:

./configure - -enable-module=auth_dbm
./configure - -enable-module=auth_db

After configuring the module, the main configuration file httpd.conf in the web server must be configured for use with Database Authentication.

The two steps to configure a dbm file are as follows:

Create the user file
Configure the web server to use this file for authentication

To create the user file, you would use the dbmmanage utility.

For example, to create a new user file named dbmpwds and add the first user:

dbmmanage dbmpwds.dat adduser user3

As with Basic Authentication and Digest Authentication, Database Authentication is done by using a directive. The passwords are stored in an encrypted format.

Here is an example directive to use for Database Authentication:

AuthName “Database Users Only”
AuthType Basic
AuthDBMUserFile /u01/app/oracle/product/8.1.7/Apache/Apache/security/dbmpwd.dat
Require user user3

The basic syntax of the dbmmanage utility is as follows:

dbmmanage database command username password

The password parameter is only used by the dbmmanage ‘add’ command. There are seven commands that are supported by the dbmmanage utility.
The seven dbmmanage utility commands are:

add	import
adduser	update
check	view
delete	

An example of using dbmmanage to create a user:

dbmmanage passwd.dbm adduser user4

The user will be prompted for a new password twice. Then user1 is created in the database named passwd.dbm. The password is encrypted.

An example of using dbmmanage to delete the user you just created:

dbmmanage passwd.dbm delete user4

An example of using dbmmanage to update the user you just created:

dbmmanage passwd.dbm update user4

An example of using the import command to change an existing password file named password.file into a DBM database named passwd.dbm:

dbmmanage passwd.dbm import < password.file

The main security concern with Database Authentication is that it is more difficult to administer. If there is not a trained administrator, then human error can compromise the authentication process. The dbmmanage utility is more complicated to use than either

the htdigest or htaccess utilities. Also, it can be more time consuming. The more time that interaction is needed, the more likely an error will occur.

What is Anonymous Authentication? How is it configured? Are there security concerns?

Anonymous Authentication uses the mod_auth_anon module. This type of authentication relies on the username being supplied by web client's browser.

Here are the directives that can be used with Anonymous Authentication:

AuthName
AuthType
require
Anonymous_Authoritative
Anonymous <list of users> (No Default)
Anonymous_MustGiveEmail on / off (On is the Default)
Anonymous_VerifyEmail on/off (Off is the Default)
Anonymous_LogEmail on/off (Off is the Default)
Anonymous_NoUserID on/off (Off is the Defautl)

AuthName, AuthType, require, and Anonymous_Authoritative are used the same way as with either the Basic Authentication or the Digest Authentication. There are no user files, no password files, and no group files in Anonymous Authentication. The Anonymous Directive is a list of usernames that can access the web server. This requirement can be overridden by setting Anonymous_NoUserID Directive to 'off'.

The Anonymous_MustGiveEmail directive expects the password entered to be an email address. The Anonymous_VerifyEmail directive will verify that it is in an email format. It does not verify that it is a valid email address. The Anonymous_LogEmail directive will send a copy of any successful attempts. "The information on the given user Ids is kept in the server access log file, typically access_log." [11]

Here is an example of Anonymous Authentication that will allow the web browser clients guest1 and guest2 to access the web server using email addresses as a password:

AuthName "For Invited Guests"
AuthType Basic
Anonymous guest1 guest2 guest3
require valid-user
Anonymous_Authoritative on
Anonymous_MustGiveEmail on
Anonymous_VerifyEmail on
Anonymous_LogEmail on
Anonymous_NoUserID off

[11] Ball, Hundt, and Rasmussen, p. 9-19.

The main security consideration is that the email address is not checked to see if it is a 'valid' email address. Only the successful attempts are logged. Because of the amount of time required to list all of the 'guest' users, more than likely the need for entering any authentication information will be turned 'off'. This in itself is a security concern.

Conclusion

There are many different types of authentication available in today's world. This paper has addressed just the four basic ones that ship with the Apache HTTP Server. There are other ones that are vendor specific. For example: Oracle uses two methods of N-Tier Authentication called Proxy Authentication and Client_Identifier. [12] There is also Secure Socket Layer (SSL) protocol and Digital ID authentication. But these types will need to be addressed in another paper. It is truly hoped that the information provided was in enough detail to be useful.

[12] O'Rourke, p.69

LIST OF REFERENCES

Books:

Ball, David and Hundt, Heike and Rasmussen, Hanne Rue. Administering Oracle9i Application Server Student Guide. Redwood Shores: Michelle Cheung, 2001. 9.1-9.21.

Brown, Bradley D. Oracle9i Web Development. New York: The McGraw-Hill Companies, Inc, 2001. 76-80.

Burleson, Donald K. Oracle9i:Unix Administration Handbook. New York: The McGraw-Hill Companies, Inc, 2002. 335-339.

Theriault, Marlene and Newman, Aaron. Oracle Security Handbook. New York: The McGraw-Hill Companies, Inc, 2001. 412-433.

Magazine Articles:

O'Rourke, Cameron. "N-Tier Authentication." Oracle Magazine May/June 2003. Volume XVII / Number 3 (2003): 69.

Internet Sources (URLs):

Apache HTTP Server Documentation Project. "Authentication, Authorization, and Access Control". URL: http://httpd.apache.org/docs/howto/auth.html (31 May 2003)

Apache HTTP Server Project. "What IS the Apache HTTP Server Project?" URL: http://httpd.apache.org/ABOUT_APACHE.html (31 May 2003)

Bell, Brad. "Security Strengths and Weaknesses of Two Popular Web Servers." 19 Aug 2001 URL: http://www.sans.org/rr/web/popular.php (31 May 2003)

Bowen, Rich. "Safer Apache Driving with AAA." 07 May 2003 URL: http://www.serverwatch.com/tutorials/print.php/2202671 (31 May 2003)

Tieman, Scott. "General Guidelines for an Apache Web Server on Solaris." 24 Jun 2001 URL: http://www.sans.org/rr/unix/apache.php (31 May 2003)

Last Updated: November 15th, 2019

Upcoming SANS Training

Click here to view a list of all SANS Courses

European Security Awareness Summit 2019	London, GB	Nov 18, 2019 - Nov 21, 2019	Live Event
SANS Austin 2019	Austin, TXUS	Nov 18, 2019 - Nov 23, 2019	Live Event
SANS Munich November 2019	Munich, DE	Nov 18, 2019 - Nov 23, 2019	Live Event
SANS SEC401 Madrid November 2019 (in Spanish)	Madrid, ES	Nov 18, 2019 - Nov 23, 2019	Live Event
SANS Atlanta Fall 2019	Atlanta, GAUS	Nov 18, 2019 - Nov 23, 2019	Live Event
Pen Test HackFest Summit & Training 2019	Bethesda, MDUS	Nov 18, 2019 - Nov 25, 2019	Live Event
SANS November Singapore 2019	Singapore, SG	Nov 18, 2019 - Nov 23, 2019	Live Event
SANS Tokyo November 2019	Tokyo, JP	Nov 25, 2019 - Nov 30, 2019	Live Event
SANS Cyber Threat Summit 2019	London, GB	Nov 25, 2019 - Nov 26, 2019	Live Event
SANS Bangalore 2019	Bangalore, IN	Nov 25, 2019 - Nov 30, 2019	Live Event
SANS Paris December 2019	Paris, FR	Dec 02, 2019 - Dec 07, 2019	Live Event
SANS Security Operations London 2019	London, GB	Dec 02, 2019 - Dec 07, 2019	Live Event
SANS San Francisco Winter 2019	San Francisco, CAUS	Dec 02, 2019 - Dec 07, 2019	Live Event
SANS Nashville 2019	Nashville, TNUS	Dec 02, 2019 - Dec 07, 2019	Live Event
SANS Frankfurt December 2019	Frankfurt, DE	Dec 09, 2019 - Dec 14, 2019	Live Event
SANS Cyber Defense Initiative 2019	Washington, DCUS	Dec 10, 2019 - Dec 17, 2019	Live Event
SANS Austin Winter 2020	Austin, TXUS	Jan 06, 2020 - Jan 11, 2020	Live Event
SANS Threat Hunting & IR Europe Summit & Training 2020	London, GB	Jan 13, 2020 - Jan 19, 2020	Live Event
SANS Miami 2020	Miami, FLUS	Jan 13, 2020 - Jan 18, 2020	Live Event
Cyber Threat Intelligence Summit & Training 2020	Arlington, VAUS	Jan 20, 2020 - Jan 27, 2020	Live Event
SANS Amsterdam January 2020	Amsterdam, NL	Jan 20, 2020 - Jan 25, 2020	Live Event
SANS Tokyo January 2020	Tokyo, JP	Jan 20, 2020 - Jan 25, 2020	Live Event
SANS Anaheim 2020	Anaheim, CAUS	Jan 20, 2020 - Jan 25, 2020	Live Event
MGT521 Beta Two 2020	San Diego, CAUS	Jan 22, 2020 - Jan 23, 2020	Live Event
SANS Las Vegas 2020	Las Vegas, NVUS	Jan 27, 2020 - Feb 01, 2020	Live Event
SANS San Francisco East Bay 2020	Emeryville, CAUS	Jan 27, 2020 - Feb 01, 2020	Live Event
SANS Vienna January 2020	Vienna, AT	Jan 27, 2020 - Feb 01, 2020	Live Event
SANS Security East 2020	New Orleans, LAUS	Feb 01, 2020 - Feb 08, 2020	Live Event
SANS London February 2020	London, GB	Feb 10, 2020 - Feb 15, 2020	Live Event
SANS Northern VA - Fairfax 2020	Fairfax, VAUS	Feb 10, 2020 - Feb 15, 2020	Live Event
SANS New York City Winter 2020	New York City, NYUS	Feb 10, 2020 - Feb 15, 2020	Live Event
SANS Gulf Region 2019	OnlineAE	Nov 16, 2019 - Nov 28, 2019	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced