Mosaic User Authentication Tutorial

Introduction

This tutorial surveys the current methods in NCSA Mosaic[1] and NCSA HTTPd[2] for restricting access to documents. The tutorial also walks through setup and use of these methods.

Mosaic 2.0 and NCSA HTTPd allow access restriction based on several criteria:

- Username/password–level access authorization.
- Rejection or acceptance of connections based on Internet address of client.
- A combination of the above two methods.

This tutorial is based heavily on work done by Ari Luotonen at CERN and Rob McCool at NCSA. In particular, Ari wrote the client–side code currently in Mosaic 2.0, and Rob wrote NCSA HTTPd 1.3.

Tutorial Contents

- Introduction[3]
- Getting Started[4]
- General Information[5]
- How Secure is it?[6]
- Basic By–Password Authentication: Step By Step[7]
- Multiple Usernames/Passwords[8]
- More Examples[9]
- For More Information[10]

Getting Started

Before you can explore access authorization, you need to install[11] NCSA HTTPd 1.0a5 or later on a Unix machine under your control, or get write access to one or more directories in a filespace already being served by NCSA HTTPd. Other HTTP Servers also support access authentication, and some of the information presented here will help you understand how it works, but this tutorial is designed specifically for NCSA HTTPd.

NCSA HTTPd 1.5 has support for HTTP Basic Authentication (Basic), as well as the proposed Message Digest Authentication (MD5). Most, if not all, current browsers should support HTTP Basic Authentication, but not all browsers support MD5. Some

1 http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html
2 http://hoohoo.ncsa.uiuc.edu/docs/Overview.html
3 #Introduction
4 #Start
5 #General
6 #Secure
7 #Basic
8 #Multiple
9 #Examples
10 #More
11 ../setup/Install.html

browsers that do include NCSA Mosaic/X 2.7[12] and Spyglass Mosaic[13]

General Information

There are two levels at which authentication information can be passed to the server: the global access configuration file and the per−directory configuration files. This tutorial primarily covers per−directory configuration. See the NCSA HTTPd documentation[14] for information on global configuration.

Per−directory configuration means that users with write access to part of the filesystem that is being served (the Document Tree) can control access to their files as they wish. They need not have root access on the system or write access to the server's primary configuration files. Also, the per−directory configuration files are read and parsed by the server on each access, allowing run−time re−configuration. The global configuration files are only parsed on start−up or restart, which usually requires root authority. There is a speed penalty associated with using the per−directory configuration files, but that's the trade−off you have to take.

Access control for a given directory is controlled by a specific file in the directory with a filename as specified by the AccessFileName[15] directive. The default filename is `.htaccess`

How Secure Is It?

In Basic HTTP Authentication, the password is passed over the network *not encrypted* but *not as plain text* −− it is "uuencoded." Anyone watching packet traffic on the network will not see the password in the clear, but the password will be easily decoded by anyone who happens to catch the right network packet.

So basically this method of authentication is roughly as safe as `telnet`−style username and password security −− if you trust your machine to be on the Internet, open to attempts to `telnet` in by anyone who wants to try, then you have no reason not to trust this method also.

In MD5 Message Digest Authentication, the password is not passed over the network at all. Instead, a series of numbers is generated based on the password and other information about the request, and these numbers are then hashed using MD5. The resulting "digest" is then sent over the network, and it is combined with other items on the server to test against the saved digest on the server. This method is more secure over the network, but it has a penalty. The comparison digest on the server must be stored in a fashion that it is retrievable. Basic Authentication stores the password using the one way `crypt()` function. When the password comes across, the server uudecodes it and then crypts it to check against the stored value. There is no way to get the password from the crypted value. In MD5, you need the information that is stored, so you can't use a one way hashing function to store it. This means that MD5 requires more rigorous security on the server machine. It is possible, but non−trivial, to implement this type of security under the Unix™ security model.

12 http://www.ncsa.uiuc.edu/SDG/Software/XMosaic/index.html
13 http://www.spyglass.com/products/browser.html
14 http://hoohoo.ncsa.uiuc.edu/docs/Overview.html
15 ../setup/srm/AccessFileName.html

Basic ByPassword Authentication: Step By Step

This should help you set up protection on a directory via the Basic HTTP Authentication method. This method also uses the standard plaintext password file. If you have a large user base, NCSA HTTPd supports a DBM based password file for faster access.

So let's suppose you want to restrict files in a directory called `turkey` to username `pumpkin` and password `pie`. Here's what to do:

Create a file called `.htaccess` *in directory* `turkey` *that looks like this:*

```
AuthUserFile /otherdir/.htpasswd
AuthGroupFile /dev/null
AuthName ByPassword
AuthType Basic

<Limit GET>
require user pumpkin
</Limit>
```

Note that the password file will be in another directory (`/otherdir`).

Also note that in this case there is no group file, so we specify `/dev/null` (the standard Unix way to say "this file doesn't exist").

`AuthName` can be anything you want. The AuthName field gives the Realm name for which the protection is provided. This name is usually given when a browser prompts for a password, and is also usually used by a browser in correlation with the URL to save the password information you enter so that it can authenticate automatically on the next challenge. *Note: You should set this to something, otherwise it will default to ByPassword, which is both non−descriptive and too common.*

`AuthType` should be set to `Basic`, since we are using Basic HTTP Authentication. Other possibilities for NCSA HTTPd 1.5 are PEM, PGP, KerberosV4, KerberosV5, or Digest. These other types of authentication will be discussed later.

In this example, only the method GET is restricted using the `LIMIT` directive. To limit other methods (particularly in CGI directories), you can specify them separated by spaces in the `LIMIT` directive. For example:

```
<LIMIT GET POST PUT>
require user pumpkin
</LIMIT>
```

If you only use `GET` protection for a CGI script, you may be finding that the `REMOTE_USER` environment variable is not getting set when using `METHOD="POST"`, obviously because the directory isn't protected against `POST`.

Create the password file `/otherdir/.htpasswd`

The easiest way to do this is to use the `htpasswd` program distributed with NCSA HTTPd. Do this:

```
htpasswd -c /otherdir/.htpasswd pumpkin
```

Type the password –– `pie` –– twice as instructed.

Check the resulting file to get a warm feeling of self–satisfaction; it should look like this:

```
pumpkin:y1ia3tjWkhCK2
```

That's all. Now try to access a file in directory `turkey` –– your browser should demand a username and password, and not give you access to the file if you don't enter `pumpkin` and `pie`. If you are using a browser that doesn't handle authentication, you will not be able to access the document at all.

Multiple Usernames/Passwords

If you want to give access to a directory to more than one username/password pair, follow the same steps as for a single username/password with the following additions:

Add additional users to the directory's `.htpasswd` *file.*

Use the `htpasswd` command without the `-c` flag to add additional users; e.g.:

```
htpasswd /otherdir/.htpasswd peanuts
htpasswd /otherdir/.htpasswd almonds
htpasswd /otherdir/.htpasswd walnuts
```

Create a group file.

Call it `/otherdir/.htgroup` and have it look something like this:

```
my-users: pumpkin peanuts almonds walnuts
```

`...` where `pumpkin`, `peanuts`, `almonds`, and `walnuts` are the usernames.

Then modify the `.htaccess` *file in the directory to look like this:*

```
AuthUserFile /otherdir/.htpasswd
AuthGroupFile /otherdir/.htgroup
AuthName ByPassword
AuthType Basic

<Limit GET>
require group my-users
</Limit>
```

Note that `AuthGroupFile` now points to your group file and that group `my-users` (rather than individual user `pumpkin`) is now required for access.

That's it. Now any user in group `my-users` can use his/her individual username and password to gain access to directory `turkey`.

Prepared Examples

Following are several examples of the range of access authorization capabilities available through Mosaic and NCSA HTTPd. The examples are served from a system at NCSA.

Simple protection by password.

This document[16] is accessible only to user `fido` with password `bones`.

Important Note: There is *no* correspondence between usernames and passwords on specific Unix systems (e.g. in an `/etc/passwd` file) and usernames and passwords in the authentication schemes we're discussing for use in the Web. As illustrated in the examples, Web–based authentication uses similar but *wholly distinct* password files; a user need never have an actual account on a given Unix system in order to be validated for access to files being served from that system and protected with HTTP–based authentication.

Protection by password; multiple users allowed.

This document[17] is accessible to user `rover` with password `bacon` and user `jumpy` with password `kibbles`.

Protection by network domain.

This document[18] is only accessible to clients running on machines inside domain `ncsa.uiuc.edu`.

Note for non–NCSA readers: The `.htaccess` file used in this case is as follows:

[16] /examples/auth/by–password/access–me.html
[17] /examples/auth/by–password–multiple–users/access–me.html
[18] /examples/auth/by–domain/access–me.html

```
AuthUserFile /dev/null
AuthGroupFile /dev/null
AuthName ExampleAllowFromNCSA
AuthType Basic

<Limit GET>
order deny,allow
deny from all
allow from .ncsa.uiuc.edu
</Limit>
```

Protection by network domain -- exclusion.

This document[19] is accessible to clients running on machines anywhere *but* inside domain `ncsa.uiuc.edu`.

Note for NCSA readers: The `.htaccess` file used in this case is as follows:

```
AuthUserFile /dev/null
AuthGroupFile /dev/null
AuthName ExampleDenyFromNCSA
AuthType Basic

<Limit GET>
order allow,deny
allow from all
deny from .ncsa.uiuc.edu
</Limit>
```

For More Information

- NCSA HTTPd Authentication Directives AccessFileName[20] – Access Configuration[21]
- W3C on HTTP–based Authentication[22]
- RFC 1321 on MD5 Digest Algorithm[23]
- Draft IETF specification for MD5 Digest Authentication[24]
- Spyglass' Efforts on WWW Security[25]

Return to the Tutorial Overview[26]

19 /examples/auth/by–domain–not–ncsa/access–me.html
20 ../setup/srm/AccessFileName.html
21 ../setup/access/Overview.html
22 http://www.w3.org/hypertext/WWW/AccessAuthorization/Overview.html
23 http://ds.internic.net/rfc/rfc1321.txt
24 http://ds.internic.net/internet–drafts/draft–ietf–http–digest–aa–0 1.txt
25 http://www.spyglass.com/six/developers_tech_doc2.html
26 index.html

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[27] / 9−27−95

[27] ../FeedBack.html