



The Gigaphone

May, 2000
Vol. 2, No. 3

The Shouting Ground Newsletter

105 S. Walnut St.
Ph: 217-351-7921
Fx: 217-351-7922

Champaign, IL 61820
email: admin@shout.net
www: www.shout.net

From the cubicle of the president ...

Web-Page Authentication

So you've got that perfect web-page up and running, and visitors are arriving by the herds! Sometimes it might be desirable to have a special section set up just for certain customers. Or perhaps you are a wholesaler, and you only want certain resellers to be able to visit a certain page outlining unique deals just for them. This is where web-page authentication can come in handy. In a nutshell, this technique allows the web-page designer to hide certain areas of your web-page from folks who just plain shouldn't be there.

Our web-servers run the Apache web-server software on machines running Linux, one of the many variants of UNIX. Apache provides mechanisms for setting up the username/password authentication on a "per-directory basis." This means that whatever it is you are protecting, must all reside within a particular directory that you plan to protect. Files not requiring protection should NOT reside in this directory.

The first step is to create a file called ".htaccess" within the directory you are protecting. A simple .htaccess file might look like this:

```
--
AuthType Basic
AuthName "Special Rates"
AuthUserFile
/home/users/bryan/my_password_files/.htpasswd
<Limit GET>
    require valid-user
</Limit>
--
```

Note: .htaccess files are text (ASCII) files. Therefore, if you are working on your .htaccess file on your home computer and plan to upload the file to the SGT web-server, make sure you use ASCII mode when transferring the file. (See the article in vol 1. no ? of the Gigaphone for why!)

Let's look at this file line by line. The first line tells the web-server what type of authentication we are doing. For now, we will always be using "Basic."

The next line uses the "AuthName" directive to indicate what is often referred to as the "Authorization Realm" for this particular directory. That information is passed to the web-

visitor so that they know which username and password to give.

The third line states that we are going to do password authentication, and that the location of the password file is "/home/users/bryan/my_password_files/.htpasswd"

The next three lines tell the web-server that when anyone issues a GET command (i.e., any time somebody tries to get to a file in that directory), that it requires a "valid-user", which it will obtain by querying for a username/password that exists within the password file.

The next step is to create the password file. This needs to be done from the UNIX prompt. Enter the directory where you are going to keep the password file, and create it using the "htpasswd" command.

```
duracef: /home/users/bryan> cd my_password_files
duracef: /home/users/bryan/my_password_files> htpasswd -c
.htpasswd bryan
New password:
Re-type new password:
Adding password for user bryan
duracef: /home/users/bryan/my_password_files>
```

First I changed directory into the directory where my password files are kept. Then I issued the "htpasswd" command. The -c flag means "create a new password file." The "bryan" afterwards means that I'm going to start my password file with a user "bryan". I am then prompted for the password for user "bryan", which I have to type in twice.

You can add subsequent username/passwords by issuing another htpasswd command -- this time without the -c.

```
duracef: bryan/my_password_files> htpasswd .htpasswd bryan2
New password:
Re-type new password:
Adding password for user bryan2
```

Note: Password files do not have to be named ".htpasswd". I can use any name I want, as long as I specify the full path to it in my ".htaccess" file. (Conversly, .htaccess files do have to be called .htaccess.)

If I peek at my password file, it looks like this:

```
duracef: bryan/my_password_files> cat .htpasswd
bryan:WxeudLD0U/6mM
bryan2:mHnMcU1x9m6P2
duracef: bryan/my_password_files>
```

Note that each username/password combination consists of a username, followed by a colon, followed by an encrypted version of the password I gave. This is pretty adequate security for most applications. However, keep in mind that it uses 56-bit DES encryption, which these days is not as hot as it used to be. If everything is setup right, the next time you visit that directory of your web-page, your web-browser will present you with a window requiring a valid username and

password. That's it! You can add usernames and passwords as needed. Unfortunately, the "htpasswd" command is not too clever, and you can't use it to delete usernames. To do that, we recommend using an online editor like pico(1) or vi(1). Alternatively, you can download it to your local computer and edit it there.

Neat!