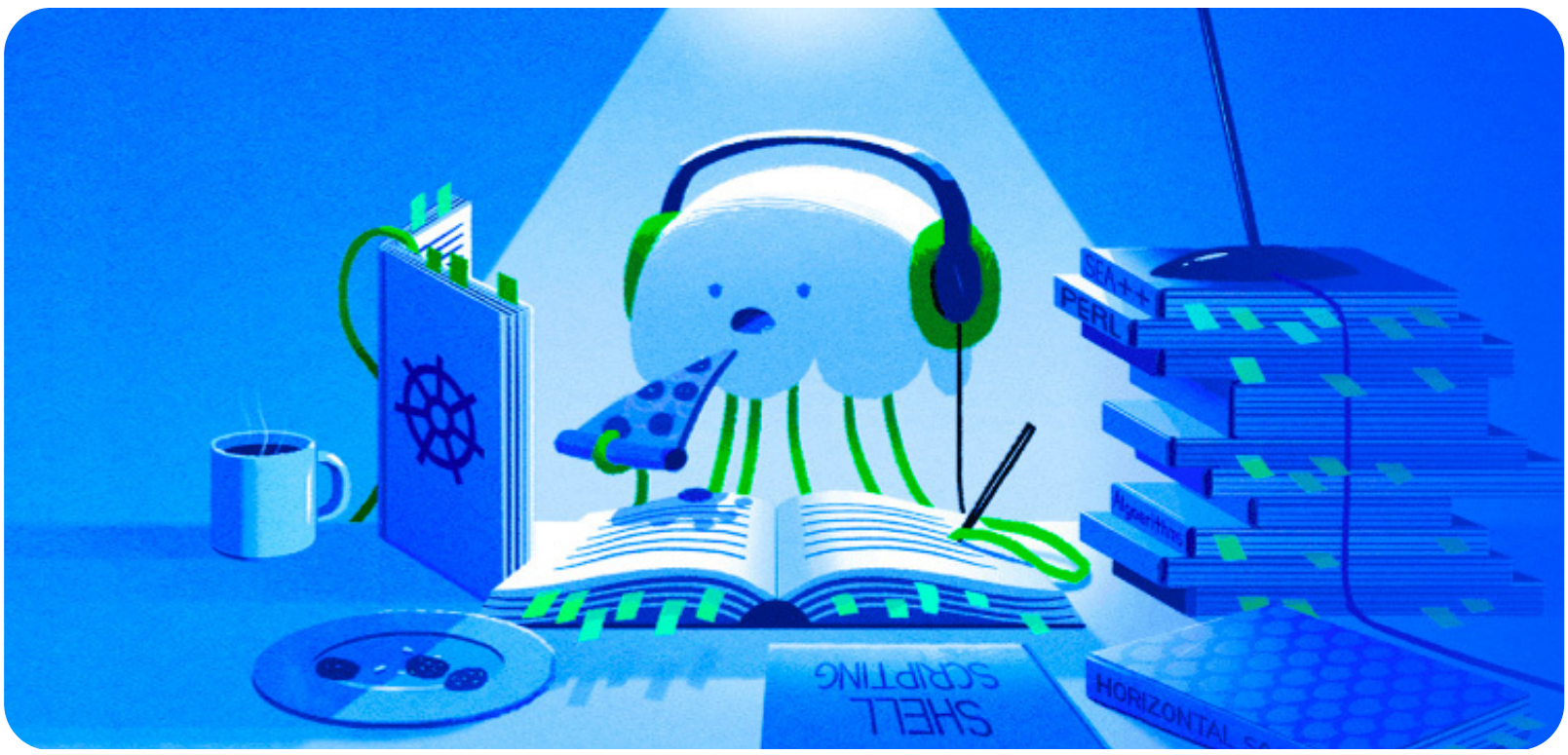
// Tutorial //

How To Set Up Password Authentication with Apache on Ubuntu 14.04

Published on August 10, 2015

Apache Security Ubuntu

By Justin Ellingwood



Not using Ubuntu 14.04?
Choose a different version or distribution.

Ubuntu 14.04

Introduction

When setting up a web server, there are often sections of the site that you wish to restrict access to. Web applications often provide their own authentication and authorization methods, but the web server itself can be used to restrict access if these are inadequate or unavailable.

In this guide, we'll demonstrate how to password protect assets on an Apache web server running on Ubuntu 14.04.

Prerequisites

To get started, you will need access to an Ubuntu 14.04 server environment. You will need a non-root user with `sudo` privileges in order to perform administrative tasks. To learn how to create such a user, follow our Ubuntu 14.04 initial server setup guide.

Install the Apache Utilities Package

In order to create the file that will store the passwords needed to access our restricted content, we will use a utility called `htpasswd`. This is found in the `apache2-utils` package within the Ubuntu repositories.

Update the local package cache and install the package by typing this command. We will take this opportunity to also grab the Apache2 server in case it is not yet installed on the server:

```
$ sudo apt-get update
$ sudo apt-get install apache2 apache2-utils
```
Copy

Create the Password File

We now have access to the `htpasswd` command. We can use this to create a password file that Apache can use to authenticate users. We will create a hidden file for this purpose called `.htpasswd` within our `/etc/apache2` configuration directory.

The first time we use this utility, we need to add the `-c` option to create the specified file. We specify a username (`sammy` in this example) at the end of the command to create a new entry within the file:

```
$ sudo htpasswd -c /etc/apache2/.htpasswd sammy
```
Copy

You will be asked to supply and confirm a password for the user.

Leave out the `-c` argument for any additional users you wish to add:

```
$ sudo htpasswd /etc/apache2/.htpasswd another_user
```
Copy

If we view the contents of the file, we can see the username and the encrypted password for each record:

```
$ cat /etc/apache2/.htpasswd
```
Copy

```
Output
sammy:$apr1$lzxsIfXG$tmCvCfb49vpPFwKGVsuYz.
another_user:$apr1$p1E9MeAf$kiAhneUwr.MhAE2kKGYHK.
```

Configure Apache Password Authentication

Now that we have a file with our users and passwords in a format that Apache can read, we need to configure Apache to check this file before serving our protected content. We can do this in two different ways.

The first option is to edit the Apache configuration and add our password protection to the virtual host file. This will generally give better performance because it avoids the expense of reading distributed configuration files. If you have this option, this method is recommended.

If you do not have the ability to modify the virtual host file (or if you are already using `.htaccess files for other purposes), you can restrict access using an` .htaccess `file. Apache uses` .htaccess` files in order to allow certain configuration items to be set within a file in a content directory. The disadvantage is that Apache has to re-read these files on every request that involves the directory, which can impact performance.

Choose the option that best suits your needs below.

Configuring Access Control within the Virtual Host Definition

Begin by opening up the virtual host file that you wish to add a restriction to. For our example, we'll be using the `000-default.conf` file that holds the default virtual host installed through Ubuntu's apache package:

```
$ sudo nano /etc/apache2/sites-enabled/000-default.conf
```
Copy

Inside, with the comments stripped, the file should look similar to this:

/etc/apache2/sites-enabled/000-default.conf

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Authentication is done on a per-directory basis. To set up authentication, you will need to target the directory you wish to restrict with a `<Directory ___>` block. In our example, we'll restrict the entire document root, but you can modify this listing to only target a specific directory within the web space:

/etc/apache2/sites-enabled/000-default.conf

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    <Directory "/var/www/html">
    </Directory>
</VirtualHost>
```

Within this directory block, specify that we wish to set up `Basic` authentication. For the `AuthName`, choose a realm name that will be displayed to the user when prompting for credentials. Use the `AuthUserFile` directive to point Apache to the password file we created. Finally, we will require a `valid-user` to access this resource, which means anyone who can verify their identity with a password will be allowed in:

/etc/apache2/sites-enabled/000-default.conf

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    <Directory "/var/www/html">
        AuthType Basic
        AuthName "Restricted Content"
        AuthUserFile /etc/apache2/.htpasswd
        Require valid-user
    </Directory>
</VirtualHost>
```

Save and close the file when you are finished. Restart Apache to implement your password policy:

```
$ sudo service apache2 restart
```
Copy

The directory you specified should now be password protected.

Configuring Access Control with .htaccess Files

If you wish to set up password protection using `.htaccess` files instead, you should begin by editing the main Apache configuration file to allow `.htaccess` files:

```
$ sudo nano /etc/apache2/apache2.conf
```
Copy

Find the `<Directory>` block for the `/var/www` directory that holds the document root. Turn on `.htaccess` processing by changing the `AllowOverride` directive within that block from "None" to "All":

/etc/apache2/apache2.conf

```
. . .

<Directory /var/www/>
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>

. . .
```

Save and close the file when you are finished.

Next, we need to add an `.htaccess` file to the directory we wish to restrict. In our demonstration, we'll restrict the entire document root (the entire website) which is based at `/var/www/html`, but you can place this file in any directory you wish to restrict access to:

```
$ sudo nano /var/www/html/.htaccess
```

Copy

Within this file, specify that we wish to set up `Basic` authentication. For the `AuthName`, choose a realm name that will be displayed to the user when prompting for credentials. Use the `AuthUserFile` directive to point Apache to the password file we created. Finally, we will require a `valid-user` to access this resource, which means anyone who can verify their identity with a password will be allowed in:

/var/www/html/.htaccess

```
AuthType Basic
AuthName "Restricted Content"
AuthUserFile /etc/apache2/.htpasswd
Require valid-user
```

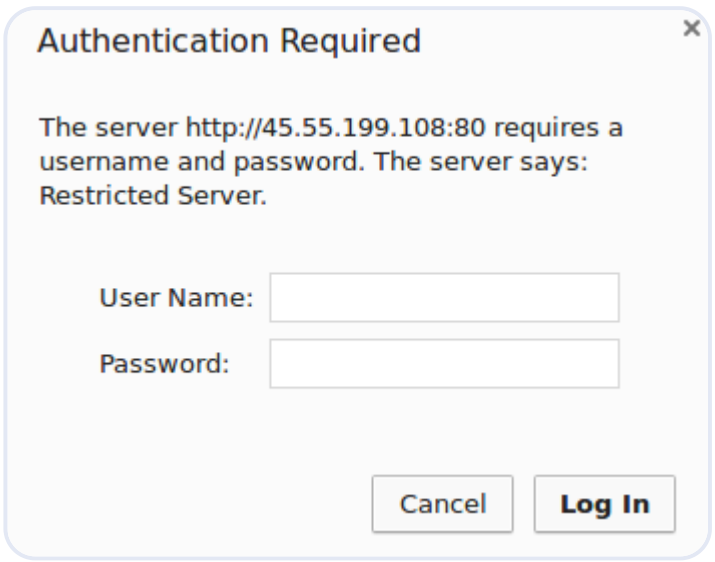
Save and close the file. Restart the web server to password protect all content in or below the directory with the `.htaccess` file:

```
$ sudo service apache2 restart
```

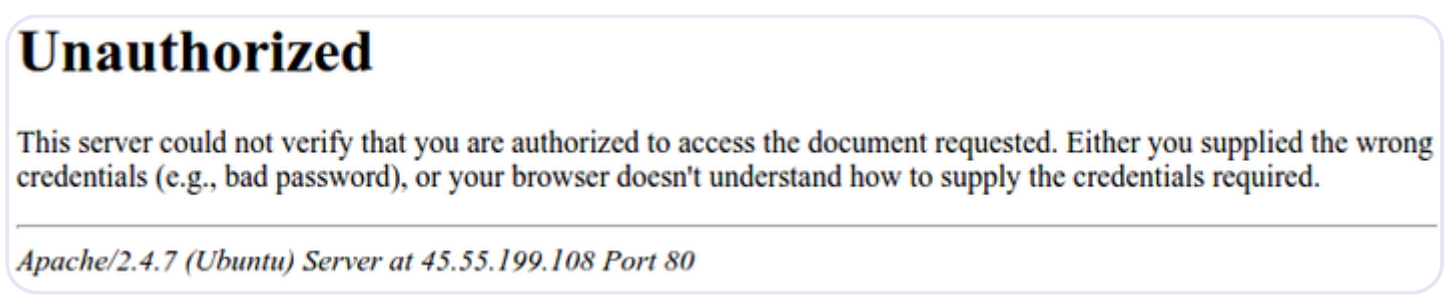
Copy

Confirm the Password Authentication

To confirm that your content is protected, try to access your restricted content in a web browser. You should be presented with a username and password prompt that looks like this:



If you enter the correct credentials, you will be allowed to access the content. If you enter the wrong credentials or hit "Cancel", you will see the "Unauthorized" error page:



Conclusion

You should now have everything you need to set up basic authentication for your site. Keep in mind that password protection should be combined with SSL encryption so that your credentials are not sent to the server in plain text. To learn how to create a self-signed SSL certificate to use with Apache, follow this guide. To learn how to install a commercial certificate, follow this guide.

About the authors

Justin Ellingwood Author

Comments

10 Comments

You can type `!ref` in this text area to quickly search our full set of tutorials, documentation & marketplace offerings and insert the link!

Sign In or Sign Up to Comment

esukf • February 29, 2016

I could not get things to work when configuring Basic authentication using Virtual Host Definition following this guide.

The auth directives no longer works within a <Directory> directive in Apache 2.4. They must be defined within a <Location> directive as per the example from Apache docs. E.g.

```
<Location />
    AuthType Basic
    AuthName "Restricted Content"
    AuthUserFile /etc/apache2/.htpasswd
    Require valid-user
</Location>
```

Show replies Reply

skylarkrieger • July 10, 2019

So I accidently password protected my entire site and it worked. Then I changed the directory to protect to the directory that actually needed protections, and it now is no longer working. I got rid of the .htaccess file in the root directory and changed the paths in the files and put a new .htaccess file in the directory i want to be password protected, but I can still get to files in that directory without the password.

Reply

skylarkrieger • July 10, 2019

This comment has been deleted

Reply

sakshi • October 16, 2017

How do I restrict access to just one file instead of a directory?

If I use the same code with Files directive instead of Directory directive, it doesn't work.

(Apache 2.2.3)

Show replies Reply

fredinger • August 6, 2017

Thank you for the great tutorial. I have just 1 Problem. When I set up the password authentication like in the Tutorial, it doesn't work for https. So what I did is, I added a new VirtualHost to the 000-default.conf File, created an SSL certificate and basically added the same content as in the VirtualHost for http. However, I can still connect to this Directory without