**Name of Title:** Learning Nginx

**Video Name:** 03\_04 Configure Password Authentication

**Estimated Length:**

**Author Name:** Michael Jenkins

**Chapter\_Section\_Video:**

**Video Objective:**

At the end of this video the learner will be able to configure the site to have a password protected page.

**Introductory Statement:**

Install htpasswd and use it to generate a password; then configure /appointments so it requires a password before access is granted.

**Speaking Points:**

1. Point\_1
2. Point\_2
3. Point\_3
4. Point\_4
5. Point\_5

**Script:**

SLIDE: 03\_03 Configuring a password

Usernames and passwords are an important part of limiting access and keeping things secure online. Let’s use nginx’s HTTP Auth Basic module to configure a simple username and password prompt to protect parts of our website.

We’ve already configured our nginx server to only allow access to the appointments page from our network.

OPEN BROWSER TO <http://192.168.0.3/appointments/>

But even if the page is being accessed from someone on our network, we still might want to limit who can see this information. This is a good case for using basic authentication which will require anyone who accesses the page to enter a username and password.

Let’s go to the terminal and connect to our development environment.

BACK IN THE SHELL

Vagrant ssh

Sudo su -

We need elevated privileges so change to the root user.

Now we can install HT password which is a utility that lets us easily create a password file. Oddly enough, nginx doesn’t ship with a utility to create password files so we need to use the one that comes with Apache.

CTRL+L

Apt install -y apache2-utils

Which htpasswd

We can get it without having to completely install apache by just installing the apache2 utils package. This includes HT password and a few other tools that help with managing web sites.

Now we can create a password file. It's important to **not** store the password file in a location that can be accessed by anyone browsing the site. In some cases, developers will include the .htpasswd file in the root directory of the site and then configure the web server to not serve anything that starts with DOT H T. But we can avoid that all together by keeping the htpassword file \_outside\_ of the root directory.

Let’s create it in the /etc/nginx directory.

CTRL+L

We do this by running htpasswd and then dash C; the path to the file; and then the username that we want to create a password for. The dash C is needed to create the file.

htpasswd -c /etc/nginx/passwords admin

I’ll use admin for the user name

Running this command prompts us to enter a new password. And then to confirm the password by retyping it.

If we want to add other users to the file, we can run the command again without the dash C. Note that if you use the dash C option and the file exists, it will be overwritten. So be careful when you’re adding new users!

Let’s add two more users and their passwords:

CTRL+L

htpasswd /etc/nginx/passwords user1

htpasswd /etc/nginx/passwords user2

We can also change a user’s password by running the command again with the user’s name.

htpasswd /etc/nginx/passwords user2

We’ll be prompted to enter a new password.

And if we need to remove a user, we can use the dash Capital D switch with the file and the name of the user to delete.

htpasswd -D /etc/nginx/passwords user2

Also, the password file is just a plain text file so it can be viewed just like any other file

cat /etc/nginx/passwords

Let’s change the permissions so this file can only be read by the user account that nginx uses, www dash data, and of course, the root user.

ls -ltr /etc/nginx/passwords

chown www-data /etc/nginx/passwords

chmod 600 /etc/nginx/passwords

ls -ltr /etc/nginx/passwords

Now let’s configure nginx to use the password file.

To do that, we need to edit the vhost file for our demo site:

vim /etc/nginx/conf.d/wisdompetmed.local.conf

In the appointments location, we’ll add the auth\_basic directive and a message. Some browsers will display this message along with the password prompt:

auth\_basic "Authentication is required...";

We'll also add the auth\_basic\_user\_file directive and the path to our password file:

auth\_basic\_user\_file /etc/nginx/passwords;

Because the appointments location also serves PHP, we need to copy the PHP configuration into this code block.

location ~ \.php$ {

include snippets/fastcgi-php.conf;

fastcgi\_pass unix:/var/run/php/php7.2-fpm.sock;

fastcgi\_intercept\_errors on;

}

This is important because once nginx processes the appointment location, it won’t process the top level PHP location as well. We need to add it here as well so nginx knows how to handle any php files it encounters in this location.

Now let’s save the file and then test and apply the new configuration.

SAVE FILE

Nginx -t

Systemctl reload nginx

If we go back to the appointments page and reload, we should be prompted for a username and password:

OPEN BROWSER

REFRESH PAGE

SHOULD SEE PASSWORD PROMPT

Now let’s enter the username and password we set up.

ENTER admin admin

Great! We just confirmed that our appointments page is protected. That should help keep things just a little bit more secure.

For more information on configuring password authentication, you can refer to the documentation for the auth basic module on nginx.org.

EDITOR: FLY IN LINK TO <http://nginx.org/en/docs/http/ngx_http_auth_basic_module.html>

**Exercise Files:**

**Basement:**

### <http://nginx.org/en/docs/http/ngx_http_auth_basic_module.html>

### <https://www.cyberciti.biz/tips/linux-unix-bsd-nginx-webserver-security.html>

### Password Protect The Directory

First create the password file and add a user called vivek:

# mkdir /usr/local/nginx/conf/.htpasswd/

# htpasswd -c /usr/local/nginx/conf/.htpasswd/passwd vivek

Edit nginx.conf and protect the required directories as follows:

### Password Protect /personal-images/ and /delta/ directories ###  
location ~ /(personal-images/.\*|delta/.\*) {  
 auth\_basic "Restricted";  
 auth\_basic\_user\_file /usr/local/nginx/conf/.htpasswd/passwd;  
}

Once a password file has been generated, subsequent users can be added with the following command:

# htpasswd -s /usr/local/nginx/conf/.htpasswd/passwd userName

<https://support.hypernode.com/knowledgebase/protect-a-directory-with-a-password-in-nginx/>

<https://stackoverflow.com/questions/4697010/nginx-auth-basic-and-php>

With this configuration, nginx will only match **one** of the two blocks - the one with the highest precedence.

The solution is to combine the PHP block *into* the AUTH block. This is the approach recommended by the nginx author himself, Igor Sysoev.

location /admin/ {

auth\_basic "Admin-Section";

auth\_basic\_user\_file /myfolder/.htpasswd;

location ~ \.php$ {

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name; include fastcgi\_params;

}

}

If you’re following along with the exercise files, you can use the Vagrantfile in the folder for this lesson. It will boot the VM and install nginx and all supporting software and data for our demo site.

If you're not using the exercise files, you can follow along with a VM running Ubuntu 18.04 LTS. You’ll need root access and you’ll need to install nginx.

STOP HERE

There’s one more thing we can add to keep with the theme for our site. Let’s add an error\_page directive to handle 401 errors, the error anyone will see if they fail to enter a correct password. Some browsers, like Safari for example, might immediately show the error page instead of prompting for the password first. So we can add a page to help our site users find their way.

To make this a quick change, we can reuse the 403 error page and make changes for the “authorization required” error.

vim /etc/nginx/conf.d/wisdompetmed.local.conf

In the vhost file, we’ll find the error page line for 403 and add the 401 code as well.

error\_page 401 /401.html;

location = /401.html {

internal;

}

SAVE FILE

Nginx -t

Systemctl reload nginx

Now let’s create our 401 page:

cp /var/www/wisdompetmed.local/403.html /var/www/wisdompetmed.local/401.html

vim /var/www/wisdompetmed.local/401.html

<title>Authentication Required</title

<h1>401</h1>

<p>Authentication is required. :(</p>

Now if anyone gets a 401 error, they’ll see this page instead of the default 401 page from nginx. If you want to test the page, you can hitting it directly by browsing to the site and then 401 DOT HTML.

In the next lesson, we’ll add another layer of security by adding a self signed SSL certificate and then configuring nginx to serve the entire site over SSL.