NCS 490:
HTTP Lab

By: Tony Perez

November 21, 2014

Abstract

In this lab we looked into setting up an Apache web server, getting it to run using HTTPS, Hypertext Transfer Protocol Secure, setup userdir, and looked into the use of .htaccess and .htpasswd files. This was all done to setup a secure web server so we could see how it worked. We used our very own ssl, Secure Socket Layer, certifications to test and see if our site worked over HTTPS.

1 Introduction

For this lab we installed an Apache web server on one of our VMs. We then went on to making it support HTTPS, Hypertext Transfer Protocol Secure, using ssl. We also had to setup userdir which allows users to setup a site within their home directory or assigned directory within their home.

2 Installing Apache & Configuring Apache

We need the **httpsd**, **mod_ssl**. and **openssl** packages.

```
# yum install httpd mod_ssl openssl
```

We then make our directory that will contain our site's HTML files. We must put it in our **/var/www/** directory.

```
# mkdir -p /var/www/tp.com/public_html
```

We then grant permissions to the directory;
Give ownership to the user instead of root

```
# sudo chown -R tony:tony /var/www/tp.com/public_html
```

We also must allow everyone to read our files

```
# sudo chmod 755 /var/www
```

Now we will create the page within **/var/www/tp.com/public_html** and call it **index.html**.
To start we can put some basic information only.

```
<html>
  <head>
    <title>tp.com</title>
  </head>
  <body>
    <h1>Success!</h1>
  </body>
</html>
```

Now we will configure Virtual Hosting to make it point to our correct directory. The file we need to configure is **/etc/httpd/conf/httpd.conf**.

It needs to look like this:

```
NameVirtualHost *:80
#
# NOTE: NameVirtualHost cannot be used without a port specifier
# (e.g. :80) if mod_ssl is being used, due to the nature of the
# SSL protocol.
#

#
# VirtualHost example:
# Almost any Apache directive may go into a VirtualHost container.
# The first VirtualHost section is used for requests without a known
# server name.
#

<VirtualHost *:80>
    ServerAdmin tony@tp.com
    DocumentRoot /var/www/tp.com/new_one
    ServerName www.tp.com
    ServerAlias tp.com
    ErrorLog /var/www/tp.com/error.log
    CustomLog /var/www/tp.com/requests.log common
</VirtualHost>
```

Change the directory and users to match yours.
Now we need to add an entry to our DNS server to point to our server. For this example I will add www to use **www.tonyDNS.com** to point to our website. Remember to change your serial number.
Our zone file should look like this:

```
$ORIGIN tonyDNS.com.
$TTL 86400
@       IN      SOA     dns1.tonyDNS.com.       tony.tonyDNS.com.       (
                        2011071011      ;SERIAL
                        3600            ;Refresh
                        1800            ;Retry
                        604800          ;Expire
                        86400           ;Minimum TTL
)

@       IN      NS      dns1.tonyDNS.com.
@       IN      NS      dns2.tonyDNS.com.
dns1    IN      A       10.103.67.80
dns2    IN      A       10.103.67.81
ftp     IN      A       10.103.67.80
www     IN      A       10.103.67.80
```

We also have to add to our iptables port 80 for http:

```
# iptables -A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
# iptables-save
```

This is our result:

3 Configure SSL

Now we will setup SSL. I used **this site** to configure ssl on our VM and you can go to for more details but I will go through the basic commands.

First we will generate a private-key

```
# openssl genrsa -out ca.key 2048
```

Then generate an CSR

```
# openssl req -new -key ca.key -out ca.csr
```

Now we will copy the files to their correct location

```
# cp ca.crt /etc/pki/tls/certs
# cp ca.key /etc/pki/tls/private/ca.key
# cp ca.csr /etc/pki/tls/private/ca.csr
```

Now we will edit the **ssl.conf**.

```
# vi +/SSLCertificateFile /etc/httpd/conf.d/ssl.conf
```

We must change the paths that are currently their to the correct ones which we used when we copied them. Find **SSLCertificateFile** and **SSLCertificateKeyFile** and point them to the correct directory.

```
SSLCertificateFile /etc/pki/tls/certs/ca.crt
SSLCertificateKeyFile /etc/pki/tls/private/ca.key
```

We must also change the **DocumentRoot** to the directory where our site is located at. In my case I had to make it look like this:

```
DocumentRoot "/var/www/tp.com/public_html"
```

Once this is done we must again make sure https, port 443, is allowed in our iptables:

```
# iptables -A INPUT -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
# iptables-save
```

This is our result:

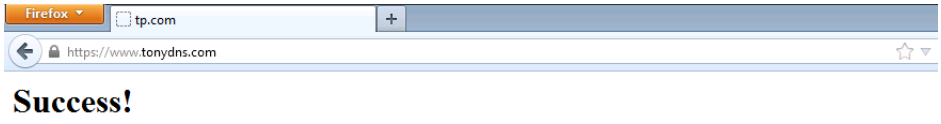


Figure 1: Notice the https:// in the address bar

4 Userdir setup

To get userdir all we have to do is change somethings in our **httpd.conf** file.
We must comment out the **UserDir disabled** line. Uncomment **UserDir public_html** and change public_html to the name you want, I used www. We must also uncomment a block like so:

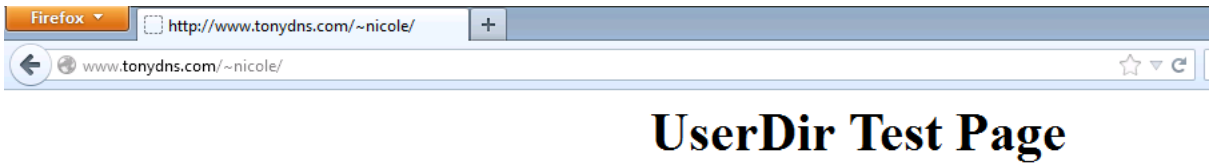
```
#    UserDir disabled

    #
    # To enable requests to /~user/ to serve the user's public_html
    # directory, remove the "UserDir disabled" line above, and uncomment
    # the following line instead:
    #
    UserDir www

</IfModule>

#
# Control access to UserDir directories.  The following is an example
# for a site where these directories are restricted to read-only.
#
<Directory /home/*/www>
    AllowOverride All
    Options None
    <Limit GET POST OPTIONS>
        Order allow,deny
        Allow from all
    </Limit>
    <LimitExcept GET POST OPTIONS>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Directory>
```

Figure 2: This is what it should look like

Once this is done make the folder within the home directory of that user.

```
# chmod 711
# mkdir www
# chmod 711 www
# touch www/index.html
```

Then edit the index.html and input some html to test.



UserDir Test Page

Figure 3: User nicole's web page

Now to use .htaccess and .htpasswd files to only allow the use of https and prompt the user for username and password.
To do this we must go into the directory where the page we want to lock down is. I will use the **www** directory of user **tony**. In our home directory we create the file **.htaccess**.

```
# touch .htaccess
```

Then edit the file and add:

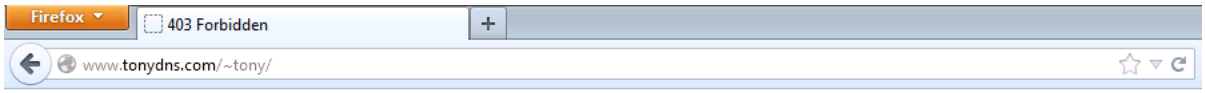
```
AuthUserFile /home/tony/www/.htpasswd
AuthType Basic
AuthName "My Files"
Require valid-user
SSLRequireSSL
```

To create the .htpasswd file all we need is a command:

```
# htpasswd -c /home/tony/www/.htpasswd tony
```

At the end you must specifiy a user then you will be prompted for a password.

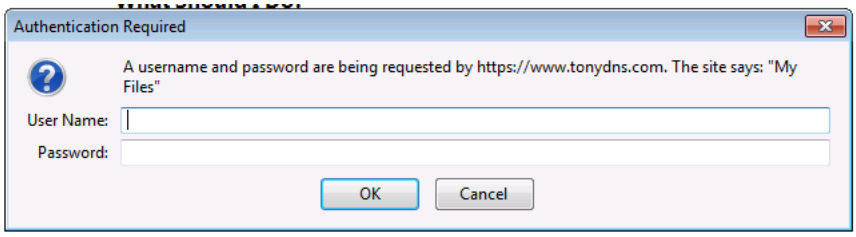
Now when you are done you can attemp to go the that users page like so without using https:



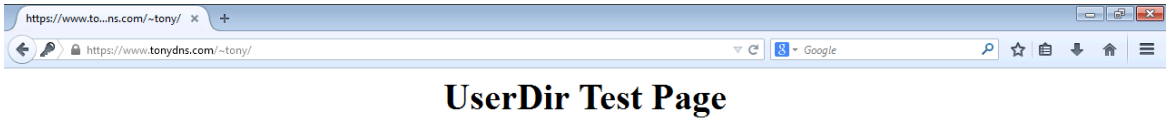
Forbidden

You don't have permission to access /~tony/ on this server.

But then when we do use https we get prompted with this:



And when we enter the correct credentials we see:



5 Conclusion

In this lab we saw how to make a web server and also make it secure by using ssl and allowing specific user access. This is a good way to setup a web page because it would make it hard for someone to break in and also enforce the use of a secure web site.