NGINX Core - Module M10 Labs

**** The Labs Begin on Page 2 ****

PLEASE READ THESE SYSTEM USE INSTRUCTIONS:

The lab systems default to text mode, if you want to take advantage of copy-and-paste functionality, or need to use multiple **Terminal** applications, you'll need to start the GUI Desktop environment using the following instructions.

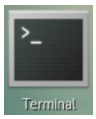
Steps to Prepare for a GUI Lab Environment:

1. Sign on to the system with the user credential **root** and password **training**.

2. Start the GUI desktop with the command:

   ```
   $ startx
   ```

3. When the Graphical Desktop appears, click on the **Terminal** app icon on the left-side of the desktop to open the Terminal.



   Note: You can resize the **Terminal** app to suit your preferences or to show more text output on the screen.

To Copy and Paste to a Lab System in GUI Mode:

1. Select the desired text from this lab guide and use **Cmd+c** (macOS) or **Ctrl-c** (Windows and Linux) to copy the text to the system clipboard.

2. Then click into the **Terminal** app in the lab system, and press Ctrl-Shift-V or select the Terminal application's **Edit** -> **Paste** menu item.

3. You may need to press it twice to insert the copied text.

   Note: When pasting into **Vim**, you MUST be in **Insert** mode first, or the keystrokes will get mis-interpreted. If you get a strange `^M` or other character first, just delete it and re-paste again.

NOTE: The **commands** you should be typing, (or copying & pasting for convenience) are displayed like the below text; (don't type or copy the **$** prompt, just the `green` text):

```
$ command
```

The **output** from commands, or the file contents shown in commands like **vim** are displayed like this:

```
Line of output
Another line of output
```

Labs for Module 10 - Encrypting Web Traffic

Labs Included

- Lab 10.1 - Create the ssl_test.conf File
- Lab 10.2 - Generate a self-signed certificate and key
- Lab 10.3 - Set protocols and ciphers
- Lab 10.4 - Configure the add_header directive
- Lab 10.5 - Configure the Diffie-Hellman key exchange
- Lab 10.6 - Force HTTP traffic to HTTPS

PLEASE READ!

You have been allocated **1 hour** for the labs in this section. Please leave the Virtual Lab Machine browser window/tab open after completing the first lab, then continue the section content. You will then be asked to complete the rest of the labs throughout the rest of the section.

If you let too much time lapse and the Virtual Lab Machine times out, simply restart it by clicking on the Launch button for the Lab and start again with the first lab.

Note: We have included configuration file samples in their entirety to make catching up with a particular lab easier. For example, if your lab times out and you were in a later lab, simply copy the configuration file contents into the appropriate file and continue with the next lab.

Lab 10.1: Create the ssl_test.conf configuration file

Learning Objectives

By the end of the lab you will be able to:

- Create a new configuration file to test mapping
- Test for results

Overview

In this exercise, you will create a new configuration file in preparation for testing the map directive results of your requests.

Steps

1. Rename the **`default.conf`** configuration file:

   ```
   $ mv /etc/nginx/conf.d/default.{conf,bak}
   ```

2. Create a new configuration file named **`ssl_test.conf`**:

```
$ vim /etc/nginx/conf.d/ssl_test.conf
```

3. Create a server block in the **ssl_test.conf** file:

   Note: When in Vim, press **i** to enter insert mode, and unless you are typing the server block manually, refer to the copy-and-paste instructions on the first page of the lab.

   New configuration is shown in **bold** text.

```
server {
  listen 80 default_server;
  root /usr/share/nginx/html;
  server_name localhost;
  access_log /var/log/nginx/https.access.log combined;
  error_log /var/log/nginx/https.error.log error;
    location / {
      return 200 "it works!\n";
    }
}
```

   Note: Closely inspect the file contents if you used the copy-and-paste function. You should indent the lines as shown in the example, additionally there may be other formatting issues to fix.

4. When done adding the content to the file, quit the editor by first pressing the **ESC** key and then typing:

```
:wq
```

5. Reload your NGINX configuration with

```
$ nginx -s reload
```

6. Test the your configuration results with:

```
$ curl http://localhost/
```

```
it works!
```

End of Lab

Lab 10.2: Generate a self-signed certificate and key

Learning Objectives

By the end of the lab you will be able to:

- Use the **`openssl`** command
- Supply the information need to generate a self-signed certificate and key
- Alter the **`listen`** directive on your system to support SSL
- Configure the use of the certificate and key on your system
- Test for results

Overview

In this exercise, you will use the openssl command to generate a self-signed certificate key and then alter the listen directive to support SSL. You'll then configure your system to use the SSL certificate and key to answer requests made to the system.

Steps

1. Generate an SSL certificate and key with the openssl command:

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:4096 -keyout
/etc/ssl/nginx/nginx.key -out /etc/ssl/nginx/nginx.crt
```

You will see the openssl command displaying it's creating a 4096 bit RSA private key, like below:

```
Generating a 4096 bit RSA private key
.....................................................................
.............++
.....................................................................
.....................................................................
.....................++
writing new private key to '/etc/ssl/nginx/nginx.key'
-----
```

Note: Next you'll be answering several questions, please use the example answers shown in the step below

2. Enter the information in **bold** as answers to the openssl command's questions:

```
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name
or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
```

```
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:MT
Locality Name (eg, city) [Default City]:Electric
Organization Name (eg, company) [Default Company Ltd]:Example
Electric
Organizational Unit Name (eg, section) []:IT Department
Common Name (eg, your name or your server's hostname)
[]:www.example.com
Email Address []:root@example.com
```

Note: Closely inspect your answers to make sure they match those above.

3. Edit the **ssl_test.conf** file:

```
$ vim /etc/nginx/conf.d/ssl_test.conf
```

4. Change the listen directive to listen on port 443 for ssl:

Note: When in Vim, press **i** to enter insert mode, and unless you are typing the server block manually, refer to the copy-and-paste instructions on the first page of the lab.

Replaced configuration is shown in **bold** text.

```
server {
  listen 443 ssl default_server;
  root /usr/share/nginx/html;
  server_name localhost;
  access_log /var/log/nginx/https.access.log combined;
  error_log /var/log/nginx/https.error.log error;
  ssl_certificate /etc/ssl/nginx/nginx.crt;
  ssl_certificate_key /etc/ssl/nginx/nginx.key;

    location / {
      return 200 "it works!\n";
    }
}
```

Note: Closely inspect the file contents if you used the copy-and-paste function. You should indent the lines as shown in the example, additionally there may be other formatting issues to fix.

5. When done adding the content to the file, quit the editor by first pressing the **ESC** key and then typing:

```
:wq
```

6. Reload your NGINX configuration with

```
$ nginx -s reload
```

7. Test the your configuration results with:

```
$ curl -kv http://localhost/
```

```
* About to connect() to localhost port 80 (#0)
*   Trying ::1...
* Connection refused
*   Trying 127.0.0.1...
* Connection refused
* Failed connect to localhost:80; Connection refused
* Closing connection 0
curl: (7) Failed connect to localhost:80; Connection refused
```

You should see the connection is refused, you are trying to connect to the HTTP default port **80**, and the server is only accepting HTTPS connections on port **443**.

8. Test the your configuration results with:

```
$ curl -kv https://localhost/
```

```
---
* About to connect() to localhost port 443 (#0)
*   Trying ::1...
* Connection refused
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* SSL connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate:
*     subject: E=root@example.com,CN=localhost,OU=IT Department,O=Example
Electric,L=Electric,ST=MT,C=IQ
*     start date: Apr 15 21:23:24 2022 GMT
*     expire date: Apr 15 21:23:24 2023 GMT
*     common name: localhost
*     issuer: E=root@example.com,CN=localhost,OU=IT Department,O=Example
Electric,L=Electric,ST=MT,C=IQ
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: localhost
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.17.6
< Date: Fri, 15 Apr 2022 21:41:40 GMT
< Content-Type: application/octet-stream
< Content-Length: 10
< Connection: keep-alive
<
it works!
* Connection #0 to host localhost left intact---
```

You should see the connection is now accepted, you are trying to connect to the HTTPS default port **443**, and the server is properly configured to accept HTTPS connections on port **443**.

Note: We will be configuring the system to accept and move connections to HTTP over to the HTTPS configuration in a later activity.

End of Lab

Lab 10.3: Set protocols and ciphers

Learning Objectives

By the end of the lab you will be able to:

- Add an ssl_protocols directive and define protocols
- Add an ssl_ciphers directive and define ciphers
- Specify that preferred servers are on
- Test for results

Overview

In this exercise, you will be adding SSL-related directives and configuration information to properly support SSL on your system and testing to see that everything is working.

Steps

1. Edit the **`ssl_test.conf`** file:

   ```
   $ vim /etc/nginx/conf.d/ssl_test.conf
   ```

2. Add the **`ssl_protocols`** directive, the **`ssl_ciphers`** directive and the **`ssl_prefer_server_ciphers`** directive to configure the server to respond to your requests:

   Note: When in Vim, press **`i`** to enter insert mode, and unless you are typing the server block manually, refer to the copy-and-paste instructions on the first page of the lab.

   New configuration is shown in **bold** text.

```
server {
  listen 443 ssl default_server;
  root /usr/share/nginx/html;
  server_name localhost;
  access_log /var/log/nginx/https.access.log combined;
```

```
  error_log /var/log/nginx/https.error.log error;
  ssl_certificate /etc/ssl/nginx/nginx.crt;
  ssl_certificate_key /etc/ssl/nginx/nginx.key;
  ssl_protocols TLSv1.2 TLSv1.3;
  ssl_ciphers AES256+EECDH:AES256+EDH:!aNULL;
  ssl_prefer_server_ciphers on;

    location / {
      return 200 "it works!\n";
    }
}
```

Note: Closely inspect the file contents if you used the copy-and-paste function. You should indent the lines as shown in the example, additionally there may be other formatting issues to fix.

3. When done adding the content to the file, quit the editor by first pressing the **ESC** key and then typing:

   ```
   :wq
   ```

4. Reload your NGINX configuration with

   ```
   $ nginx -s reload
   ```

5. Test the your configuration results with:

   ```
   $ curl -kv https://localhost
   ```

```
* About to connect() to localhost port 443 (#0)
*   Trying ::1...
* Connection refused
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* SSL connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate:
*     subject: E=root@example.com,CN=www.example.com,OU=IT
Deparment,O=Example Electric,L=Electric,ST=MT,C=US
*     start date: Apr 18 21:47:51 2022 GMT
*     expire date: Apr 18 21:47:51 2023 GMT
*     common name: www.example.com
*     issuer: E=root@example.com,CN=www.example.com,OU=IT
Deparment,O=Example Electric,L=Electric,ST=MT,C=US
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: localhost
> Accept: */*
>
< HTTP/1.1 200 OK
```

```
< Server: nginx/1.17.6
< Date: Mon, 18 Apr 2022 21:50:31 GMT
< Content-Type: application/octet-stream
< Content-Length: 10
< Connection: keep-alive
<
it works!
* Connection #0 to host localhost left
```

The test shows the configuration is correct.

End of Lab

Lab 10.4: Configure the add_header directive

Learning Objectives

By the end of the lab you will be able to:

- Configure the **`add_header`** directive
- Test for results

Overview

In this exercise, you will add an if directive to your configuration to further modify the results. You'll test with the -i option to curl for more information on each request.

Steps

1. Edit the **`ssl_test.conf`** file:

   ```
   $ vim /etc/nginx/conf.d/ssl_test.conf
   ```

2. Update the map directive in the **`ssl_test.conf`** file:

   Note: When in Vim, press **`i`** to enter insert mode, and unless you are typing the server block manually, refer to the copy-and-paste instructions on the first page of the lab.

   New configuration is shown in **bold** text.

```
server {
  listen 443 ssl default_server;
  root /usr/share/nginx/html;
  server_name localhost;
  access_log /var/log/nginx/https.access.log combined;
```

```
  error_log /var/log/nginx/https.error.log error;
  ssl_certificate /etc/ssl/nginx/nginx.crt;
  ssl_certificate_key /etc/ssl/nginx/nginx.key;
  ssl_protocols TLSv1.2 TLSv1.3;
  ssl_ciphers AES256+EECDH:AES256+EDH:!aNULL;
  ssl_prefer_server_ciphers on;
  add_header Strict-Transport-Security "max-age=63072000;
  includeSubdomains; ";
  add_header X-Content-Type-Options nosniff;
  add_header X-Frame-Options DENY;

    location / {
      return 200 "it works!\n";
    }
}
```

Note: Closely inspect the file contents if you used the copy-and-paste function. You should indent the lines as shown in the example, additionally there may be other formatting issues to fix.

3. When done adding the content to the file, quit the editor by first pressing the **ESC** key and then typing:

   ```
   :wq
   ```

4. Reload your NGINX configuration with

   ```
   $ nginx -s reload
   ```

5. Test the your configuration results with:

   ```
   $ curl -kv https://localhost/
   ```

```
* About to connect() to localhost port 443 (#0)
*   Trying ::1...
* Connection refused
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* SSL connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate:
*     subject: E=root@example.com,CN=www.example.com,OU=IT
Department,O=Example Electric,L=Electric,ST=MT,C=US
*     start date: Apr 18 15:53:27 2022 GMT
*     expire date: Apr 18 15:53:27 2023 GMT
*     common name: www.example.com
*     issuer: E=root@example.com,CN=www.example.com,OU=IT
Department,O=Example Electric,L=Electric,ST=MT,C=US
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
```

```
> Host: localhost
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.17.6
< Date: Mon, 18 Apr 2022 16:05:27 GMT
< Content-Type: application/octet-stream
< Content-Length: 10
< Connection: keep-alive
< Strict-Transport-Security: max-age=63072000;
<   includeSubdomains;
< X-Content-Type-Options: nosniff
< X-Frame-Options: DENY
<
it works!
* Connection #0 to host localhost left intact
```

The test shows the configuration is correct.

End of Lab

Lab 10.5: Configure the Diffie-Hellman key exchange

Learning Objectives

By the end of the lab you will be able to:

- Generate a new dhparam.pem file using openssl
- Add a ssl_dhparam directive to the ssl_test.conf file
- Test the results

Overview

In this exercise, you will add an if directive to your configuration to further modify the results. You'll test with the -i option to curl for more information on each request.

Steps

1. Generate a new dhparam.pem file with openssl:

   ```
   $ openssl dhparam -dsaparam -out /etc/nginx/dhparam.pem 4096
   ```

   Note: The output shown below is greatly truncated, there will be a number of lines of output until the command is finished.

   ```
   Generating DSA parameters, 4096 bit long prime
   ```

```
.....................+...+.........+...............+....++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++*
...........+.....................................+.........+..............+....
.+...+........................+..+.....+...
++++++++++++++++++++++++++++++++++++++++++++++++++*
```

2. Edit the **`ssl_test.conf`** file:

   `$ vim /etc/nginx/conf.d/ssl_test.conf`

3. Add the file just generated to your configuration using an **`ssl_dhparam`** directive in the server context:

   Note: When in Vim, press **`i`** to enter insert mode, and unless you are typing the server block manually, refer to the copy-and-paste instructions on the first page of the lab.

   New configuration is shown in **bold** text.

```
server {
  listen 443 ssl default_server;
  root /usr/share/nginx/html;
  server_name localhost;
  access_log /var/log/nginx/https.access.log combined;
  error_log /var/log/nginx/https.error.log error;
  ssl_certificate /etc/ssl/nginx/nginx.crt;
  ssl_certificate_key /etc/ssl/nginx/nginx.key;
  ssl_protocols TLSv1.2 TLSv1.3;
  ssl_ciphers AES256+EECDH:AES256+EDH:!aNULL;
  ssl_prefer_server_ciphers on;
  add_header Strict-Transport-Security "max-age=63072000;
  includeSubdomains; ";
  add_header X-Content-Type-Options nosniff;
  add_header X-Frame-Options DENY;
  ssl_dhparam /etc/nginx/dhparam.pem;

    location / {
      return 200 "it works!\n";
    }
}
```

   Note: Closely inspect the file contents if you used the copy-and-paste function. You should indent the lines as shown in the example, additionally there may be other formatting issues to fix.

4. When done adding the content to the file, quit the editor by first pressing the **ESC** key and then typing:

   `:wq`

5. Reload your NGINX configuration with

```
$ nginx -s reload
```

6. Test the your configuration results with:

```
$ curl -kv https://localhost/
```

```
* About to connect() to localhost port 443 (#0)
*   Trying ::1...
* Connection refused
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* SSL connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate:
*     subject: E=root@example.com,CN=www.example.com,OU=IT
Department,O=Example Electric,L=Electric,ST=MT,C=US
*     start date: Apr 18 15:53:27 2022 GMT
*     expire date: Apr 18 15:53:27 2023 GMT
*     common name: www.example.com
*     issuer: E=root@example.com,CN=www.example.com,OU=IT
Department,O=Example Electric,L=Electric,ST=MT,C=US
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: localhost
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.17.6
< Date: Mon, 18 Apr 2022 16:16:39 GMT
< Content-Type: application/octet-stream
< Content-Length: 10
< Connection: keep-alive
< Strict-Transport-Security: max-age=63072000;
<   includeSubdomains;
< X-Content-Type-Options: nosniff
< X-Frame-Options: DENY
<
it works!
* Connection #0 to host localhost left intact
```

The test shows the configuration is correct.

End of Lab

Lab 10.6: Force HTTP Traffic to HTTPS

Learning Objectives

By the end of the lab you will be able to:

- Add another server that listens on port 80
- Configure redirects to 443 and SSL
- Test the results

Overview

In this exercise, you will add another server context that listens on port 80 and then ensure it redirects requests to port 443 using SSL, and of course, test your results.

Steps

1. Edit the **`ssl_test.conf`** file:

   ```
   $ vim /etc/nginx/conf.d/ssl_test.conf
   ```

2. Add a server directive that listens on port 80 and redirects to port 443 using SSL:

   Note: When in Vim, press **`i`** to enter insert mode, and unless you are typing the server block manually, refer to the copy-and-paste instructions on the first page of the lab.

   New configuration is shown in **bold** text.

```
server {
listen 80 default_server;
return 301 https://$host$request_uri;
}

server {
  listen 443 ssl default_server;
  root /usr/share/nginx/html;
  server_name localhost;
  access_log /var/log/nginx/https.access.log combined;
  error_log /var/log/nginx/https.error.log error;
  ssl_certificate /etc/ssl/nginx/nginx.crt;
  ssl_certificate_key /etc/ssl/nginx/nginx.key;
  ssl_protocols TLSv1.2 TLSv1.3;
  ssl_ciphers AES256+EECDH:AES256+EDH:!aNULL;
  ssl_prefer_server_ciphers on;
  add_header Strict-Transport-Security "max-age=63072000;
  includeSubdomains; ";
  add_header X-Content-Type-Options nosniff;
  add_header X-Frame-Options DENY;
```

```
  ssl_dhparam /etc/nginx/dhparam.pem;

    location / {
      return 200 "it works!\n";
    }
}
```

Note: Closely inspect the file contents if you used the copy-and-paste function. You should indent the lines as shown in the example, additionally there may be other formatting issues to fix.

3. When done adding the content to the file, quit the editor by first pressing the **ESC** key and then typing:

   ```
   :wq
   ```

4. Reload your NGINX configuration with

   ```
   $ nginx -s reload
   ```

5. Test the your configuration results with:

   ```
   $ curl -kv https://localhost
   ```

```
* About to connect() to localhost port 80 (#0)
*   Trying ::1...
* Connection refused
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> HEAD / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: localhost
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
HTTP/1.1 301 Moved Permanently
< Server: nginx/1.17.6
Server: nginx/1.17.6
< Date: Mon, 18 Apr 2022 17:23:14 GMT
Date: Mon, 18 Apr 2022 17:23:14 GMT
< Content-Type: text/html
Content-Type: text/html
< Content-Length: 169
Content-Length: 169
< Connection: keep-alive
Connection: keep-alive
< Location: https://localhost/
Location: https://localhost/

<
* Connection #0 to host localhost left intact
```

The test shows the configuration is correct.

End of Lab