

Lab 11: Encrypting Web Traffic

Estimated time for completion: **20 minutes**

Requirements

The following tasks must be completed before beginning this lab:

- Getting Started with NGINX, (the Getting Started Guide in LearnF5)
- Log into Hosted Environment, your lab initialization instructions are located in the LearnF5 course

Scenario

In these exercises, you will be able to set up a directory and run the **openssl** command and set up the ssl parameters according to NGINX best practices and then force all traffic to https. Then you'll test the site on SSL Labs.com

Objectives

At the end of this lab you will be able to:

- Configure HTTPS access
- Set up encryption levels on the proxy server
- Test the security of your server against Qualys

Lab Contents

Exercise 1: Set up a directory for certificates and keys

Exercise 2: Configure the HTTPS access and test it



IMPORTANT

You can copy and paste the commands and text from the examples to your terminal or editor, (just make sure you don't copy and paste the \$ prompt!)

Exercise 1: Generate Certificate and Key

Learning Objective:

Set up the directory and run the `openssl` command to generate a self-signed certificate and key

Scenario

In this exercise, you set up a directory and run the `openssl` command.

1. Use the following command to create a directory for certificates and keys:

```
$ sudo mkdir -p /etc/nginx/ssl
```

2. Change directories to the ssl directory:

```
$ cd /etc/nginx/ssl
```

To create cert and key files, run the `openssl` command and press return through all of the `openssl` prompts:

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa_4096  
-keyout nginx.key -out nginx.crt
```

```

NGINX$ sudo mkdir -p /etc/nginx/ssl
NGINX$ cd /etc/nginx/ssl
/ssl/nginx/ssl/nginx.key -out /etc/nginx/ssl/nginx.crt -
keyout
Can't load /home/student1/.rnd into RNG
139977674801600:error:2406F079:random number
generator:RAND_load_file:Cannot open
file:../crypto/rand/randfile.c:88:Filename=/home/student1/.rnd
Generating a RSA private key
///++++
.....
.....
.....++++
writing new private key to '/etc/nginx/ssl/nginx.key'
-----
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 [AU]):
State or Province Name (full name) [Some-State]:
Locality name (eg, city) []:
Organization name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
NGINX$

```

3. List the cert and key files you created:

```
$ ls -l
```

There should be a main (master) NGINX process and at least one worker process displayed, with their corresponding Process ID's (highlight added below).

```
NGINX$ ls -l
total 8
-rw-r--r-- 1 root root 1939 Oct 18 20:38 nginx.crt
-rw----- 1 root root 3268 Oct 18 20:38 nginx.key
NGINX$
```

Exercise 2: Configure and Test SSL Parameters

Learning Objective:

Set up the ssl parameters according to NGINX best practices & Test the configuration of the site against sslabs.com

Scenario

In this exercise, you set up the ssl parameters according to NGINX best practices and then force all traffic to https. Then you'll test the site on SSL Labs.com

1. Create a file called `ssl-params.conf`:

```
$ sudo vim /etc/nginx/conf.d/ssl-params.ocnf
```

2. Paste the following configurations into the file:

```
ssl_certificate /etc/nginx/ssl/nginx.crt;
ssl_certificate_key /etc/nginx/ssl/nginx.key;
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers "AES256+EECDH:AES256+EDH:!aNULL";
ssl_prefer_server_ciphers on;
ssl_conf_command Ciphersuites TLS_AES_256_GCM_SHA384;
ssl_session_cache shared:SSL:10m;
ssl_session_tickets off;
add_header Strict-Transport-Security "max-age=63072000;
includeSubdomains";
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;
```

3. Save and exit the file. (esc and `:wq`).
4. Open the `myserver.conf` file and change the `listen` directive (port 80) to port 443 and add the `ssl` parameter:

```
$ sudo vim /etc/nginx/conf.d/myserver.conf
```

```
server {
listen 443 ssl;
    root /usr/share/nginx/html;
    . . .
```

5. Add a new server block above this one, forcing http traffic to https:

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

This shows the changed part of the `myservers.conf` file:

```
server {  
listen 80;  
    return 301 https://$host$request_uri;  
}  
  
server {  
    listen 443 ssl;  
    root /usr/share/nginx/html;  
    error_log /var/log/nginx/upstream.error.log info;  
    access_log /var/log/nginx/upstream.access.log combined;  
  
    proxy_cache_key $scheme$host$request_uri;  
    add_header X-Proxy-cache $upstream_cache_status;  
  
    location / {  
        proxy_pass http://myServers;  
        health_check match=health_conditions fails=2  
uri=/health/test.html;  
        proxy_cache upstream_cache;  
        proxy_cache_valid 200 5m;  
    }  
}
```

6. Save the file and reload NGINX. (**esc** and **:wq**).

```
$ sudo nginx -s reload
```

7. Test the access using the following **curl** command (the **-i** parameter shows body content, the **I** parameter shows headers, the **L** parameter tells NGINX to follow any redirect, and the **k** parameter tells NGINX to ignore any SSL errors such as a self-signed certificate). You get both the return and the redirect:

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

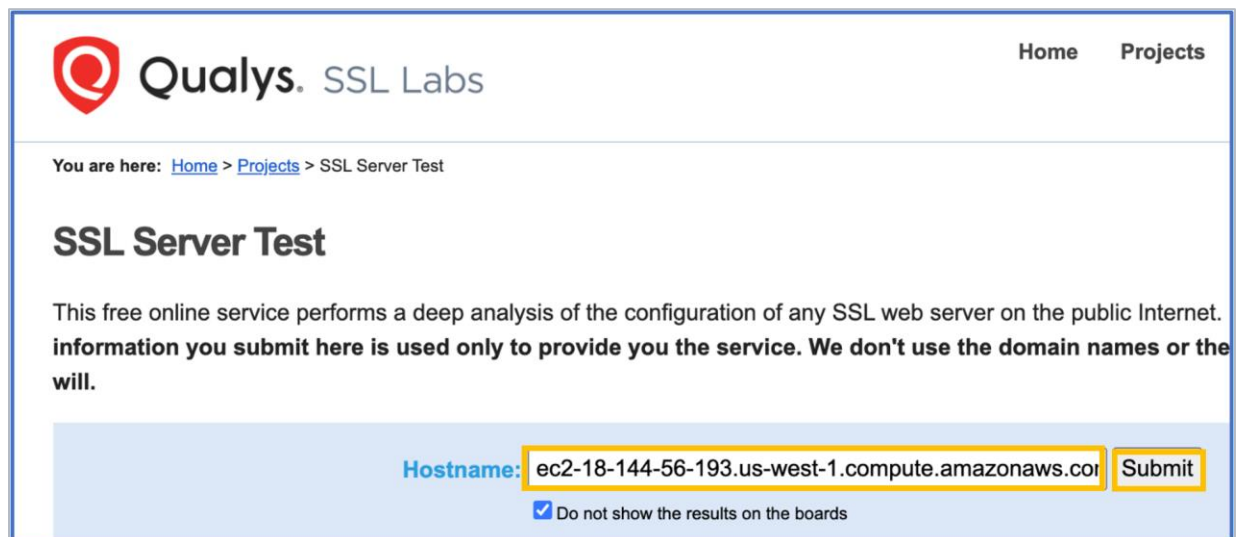
```
$ curl iILk http://localhost/
HTTP/1 1 301 Moved Permanently
Server: nginx/1.15.10
Date: Thu, 28 Nov 2019 14:16:34 GMT
Content-Type: text/html
Content-Length: 170
Connection: keep-alive
Location: https://localhost/
Strict-Transport-Security: max-age=63072000; includeSubdomains
X-frame-Options: DENY
X-Content-Type-Options: nosniff

HTTP/1 1 200 OK
Server: nginx/1.15.10
Date: Thu, 28 Nov 2022 14:16:34 GMT
Content-Type: text/html
Content-Length: 26
Connection: keep-alive
Last-Modified: Mon, 25 Nov 2022 11:10:25 GMT
ETag: "5caba10f-1a"
Strict-Transport-Security: max-age=63072000; includeSubdomains
X-frame-Options: DENYX
X-Content-Type-Options: nosniff
X-Proxy-Cache: HIT
Accept-Ranges: bytes
```

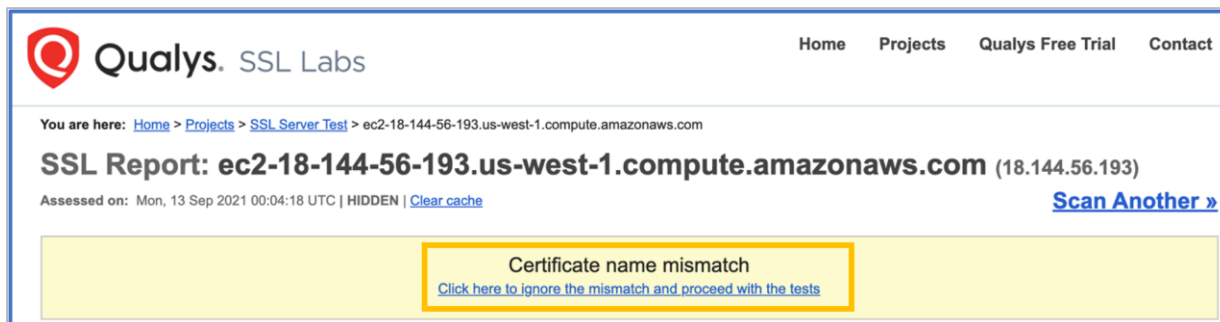
8. In a browser, test your NGINX lab system on ssllabs.com.
 - a. Click on "Test your server".



- b. Enter your lab systems FQDN in the **Hostname** box and click **Submit**.

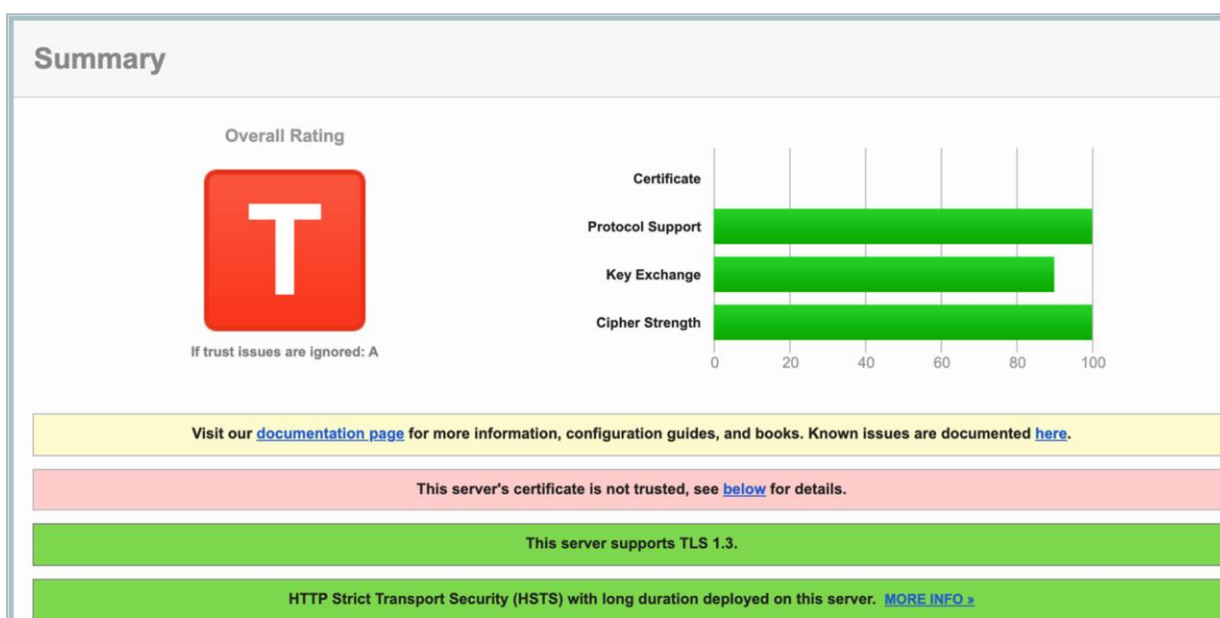


c. When the certificate mismatch message appears, click the link to ignore it.



The screenshot shows the top section of a Qualys SSL Labs report. The header includes the Qualys logo and navigation links (Home, Projects, Qualys Free Trial, Contact). Below the header, the report title is "SSL Report: ec2-18-144-56-193.us-west-1.compute.amazonaws.com (18.144.56.193)". The assessment date is "Mon, 13 Sep 2021 00:04:18 UTC | HIDDEN | Clear cache". A yellow box highlights a "Certificate name mismatch" message with a link to "Click here to ignore the mismatch and proceed with the tests". A "Scan Another »" link is also present.

d. Explore the Web page. What were the results?



If there were no trust issues, then our NGINX configuration would get an A rating. The trust issue is because our certificate chain does not contain all the necessary certificates to connect the web server certificate to one of the root certificates in the trust store.

Expected Results

In these exercises, you were able to set up a directory and run the **openssl** command, set up the ssl parameters according to NGINX best practices & Test the configuration of the sight against ssllabs.com

