Lesson 2 Labs

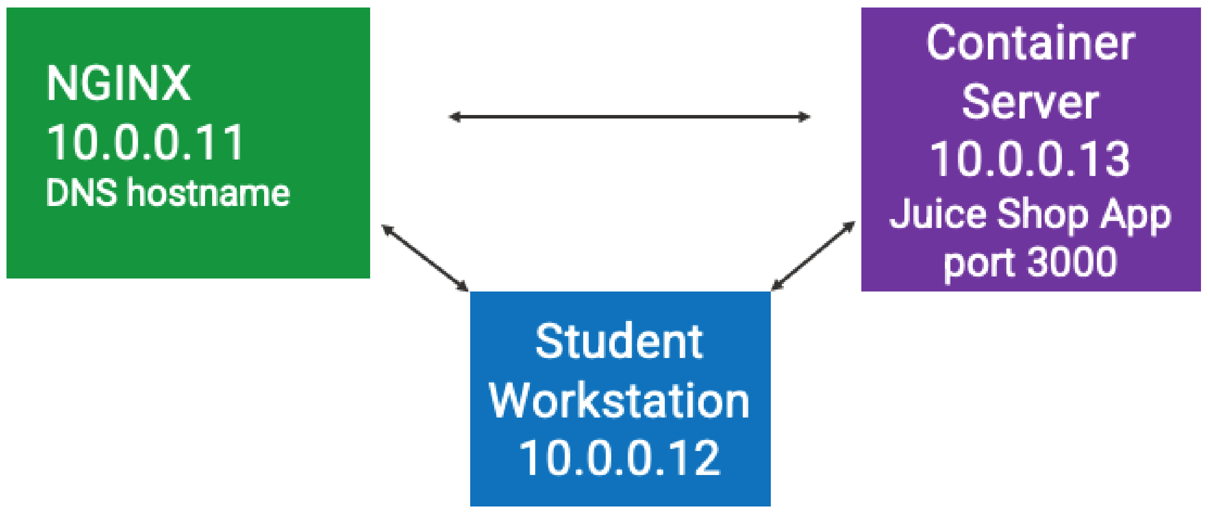
Lab Contents

Estimated time to complete labs: **[XX] minutes**

Lab Credentials and IP Address List

LAB SYSTEM	**IP ADDRESS**
NGINX	10.0.0.11
Student Workstation	10.0.0.12
Container Server	10.0.0.13

Lab Network Topology



There are 3 lab systems: NGINX, Student Workstation and Container Server. The NGINX lab system has NGINX Plus R29 installed. The Container Server system has the OWASP Juice Shop application running on port 3000 using Kubernetes. The Student Workstation lab system is an Ubuntu Linux system that is a graphical system used to test access to the NGINX lab system.

Lab 1: Setting up an HTTPS Server

Estimated time for completion: **XX minutes**

Requirements

The following tasks must be completed before beginning this lab:

- Confirm access to the NGINX and Student Workstation labs systems.
- The NGINX lab system must have a **`/etc/nginx/conf.d/juice.conf`** file specifying **`proxy_pass`** and **`upstream`** to where the Juice Shop app is running.
- The OWASP Juice Shop web application is running on the Server Container lab system.

Overview

In this lab begins with generation of certificates for the NGINX system in order to configure HTTPS for accessing the web application Juice Shop. Then a server context to support HTTPS is added. The lab continues with editing the NGINX configuration file for the Juice Shop application to support specific SSL ciphers. These SSL ciphers require generation of a new Diffie-Hellman Key-Exchange to use this specific SSL cipher algorithm for Forward Secrecy. The lab concludes with testing the NGINX configuration to confirm it now uses HTTPS Forward Secrecy with the Diffie-Hellman SSL cipher algorithm.

Objectives

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

- Create a private CA Root Authority certificate
- Create a public and private certificate for the NGINX lab system
- Set permissions for cert files
- Add the NGINX directives for certificates
- Configure a server context listening on port 443 for HTTPS
- Set header values to be passed to the upstreams
- Specify which SSL cipher algorithms for NGINX to use
- Configure Forward Secrecy in NGINX for Diffie-Hellman
- Test HTTPS configuration

Exercise 1: Generating Certs and Keys

Overview

In this exercise, you generate certificates for the NGINX system in order to configure HTTPS for accessing the web application Juice Shop.

1. Create a private key for the CA Root Authority, generate the X509 certificate for the CA Root Authority, and create the public and private certs for your NGINX system. The **`create_certs.sh`** script will create the CA Root Authority certs, and certs for a DNS name that represents your NGINX lab system **www.nginxtraining.com**.

```
$ cd ~/ssl
$ less create_certs.sh
$ ./create_certs.sh www.nginxtraining.com
$ ls -ltr
```

> NOTE
> Using self-signed certificates or a certificate signed by a private CA is for training or testing purposes. In a production environment you would have certs signed by a legitimate CA Root Authority.

2. List the permissions, owner and group of the cert files created. Change the group to nginx for the newly created certificate files and change the permissions as follows.

```
$ sudo cp *crt www*key /etc/nginx/ssl
$ cd /etc/nginx/ssl
$ sudo chgrp nginx *
$ sudo chmod 640 *
$ ls -ltr
$ sudo file *
```

```
student@nginx:/etc/nginx/ssl$ ls -ltr
total 12
-rw-r----- 1 root nginx 1675 Jun 16 10:39 www.nginxtraining.com.key
-rw-r----- 1 root nginx 1310 Jun 16 10:39 www.nginxtraining.com.crt
-rw-r----- 1 root nginx 1375 Jun 16 10:39 ca-cert.crt
student@nginx:/etc/nginx/ssl$ sudo file *
ca-cert.crt:                 PEM certificate
www.nginxtraining.com.crt: PEM certificate
www.nginxtraining.com.key: PEM RSA private key
student@nginx:/etc/nginx/ssl$
```

NOTE
The **`ca-cert.crt`** file is in PEM format. When using Dynamic certificate loading read access is required.

3. Before making changes, backup the original NGINX SSL configuration file /etc/nginx/ssl-configs/ssl-params.conf

```
$ sudo cp /etc/nginx/ssl-configs/ssl-params.{conf,orig}
```

4. Edit the ssl-params.conf file to add the **ssl_certificate** directives.

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

```
...
#ssl_session_tickets off;
#ssl_stapling on;
#ssl_stapling_verify on;

ssl_trusted_certificate /etc/nginx/ssl/ca-cert.crt;
ssl_certificate /etc/nginx/ssl/www.nginxtraining.com.crt;
ssl_certificate_key /etc/nginx/ssl/www.nginxtraining.com.key;

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

NOTE
The **`ssl_trusted_certificate`** directive needs to be before the **`ssl_certificate`** and **`ssl_certificate_key`** directives.

5. Save the file and exit the editor.

   ```
   :wq!
   ```

6. Test that the configuration syntax has no errors.

   ```
   $ sudo nginx -t
   ```

Expected Results

You should now have the appropriate certificates set up for NGINX to access the Juice Shop application when HTTPS is configured.

Exercise 2: Configuring an HTTPS Server

Overview

In this exercise, you establish a server context to support HTTPS.

1. Verify **NGINX** has support for HTTPS.

   ```
   $ sudo nginx -V
   ```

```
student@nginx:~$ sudo nginx -V
nginx version: nginx/1.23.4 (nginx-plus-r29)
built by gcc 9.3.0 (Ubuntu 9.3.0-10ubuntu2)
built with OpenSSL 1.1.1f  31 Mar 2020
TLS SNI support enabled
configure arguments: --prefix=/etc/nginx --sbin-
path=/usr/sbin/nginx --modules-path=/usr/lib/nginx/modules --conf-
path=/etc/nginx/nginx.conf --error-log-
path=/var/log/nginx/error.log --http-log-
path=/var/log/nginx/access.log --pid-path=/var/run/nginx.pid --
lock-path=/var/run/nginx.lock --http-client-body-temp-
path=/var/cache/nginx/client_temp --http-proxy-temp-
path=/var/cache/nginx/proxy_temp --http-fastcgi-temp-
path=/var/cache/nginx/fastcgi_temp --http-uwsgi-temp-
path=/var/cache/nginx/uwsgi_temp --http-scgi-temp-
path=/var/cache/nginx/scgi_temp --user=nginx --group=nginx --with-
compat --with-file-aio --with-threads --with-http_addition_module -
-with-http_auth_request_module --with-http_dav_module --with-
http_flv_module --with-http_gunzip_module --with-
http_gzip_static_module --with-http_mp4_module --with-
http_random_index_module --with-http_realip_module --with-
http_secure_link_module --with-http_slice_module --with-
http_ssl_module --with-http_stub_status_module --with-
http_sub_module --with-http_v2_module --with-mail --with-
mail_ssl_module --with-stream --with-stream_realip_module --with-
stream_ssl_module --with-stream_ssl_preread_module --build=nginx-
plus-r29 --with-http_auth_jwt_module --with-http_f4f_module --with-
http_hls_module --with-http_proxy_protocol_vendor_module --with-
http_session_log_module --with-stream_mqtt_filter_module --with-
stream_mqtt_preread_module --with-
stream_proxy_protocol_vendor_module --with-cc-opt='-g -O2 -fdebug-
prefix-map=/data/builder/debuild/nginx-plus-1.23.4/debian/debuild-
base/nginx-plus-1.23.4=. -fstack-protector-strong -Wformat -
Werror=format-security -Wp,-D_FORTIFY_SOURCE=2 -fPIC' --with-ld-
opt='-Wl,-Bsymbolic-functions -Wl,-z,relro -Wl,-z,now -Wl,--as-
needed -pie'
student@nginx:~$
```

2. Before making any changes, backup the **`juice.conf`** configuration file.

```
$ sudo cp /etc/nginx/conf.d/juice.{conf,orig}
```

3. **Edit** the **`juice.conf`** Juice Shop configuration file.

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

   a. After the **`listen`** directive, add a **`return`** directive that forces all http traffic to **`https`** and close this **`server`** context by adding a right brace **`}`**. Do not delete any other lines in the file as we are not finished editing.
   b. After the close of the **`return`** directive, create a new **`server`** context listening on port **`443`**, with the **`ssl`** parameter enabled and make this the **`default`** server.
   c. After the **`listen`** directive, add the **`server_name`** directive with the DNS name for your NGINX system, **`www.nginxtraining.com`**.
   d. After the **`root`** directive, add an **`include`** directive for accessing your custom security settings file **`/etc/nginx/ssl-configs/ssl-params.conf`**.
   e. After the **`access_log`** directive, add the following **`proxy_set_header`** directives.

Your **`juice.conf`** file should now look like this (partial) with changes highlighted:

```
...
upstream juice_server {
 server 10.0.0.13:3000;
    zone juiceshop 64k;
}

server {
    listen 80;
    return 301 https://$host$request_uri;
}
server {
    listen 443 ssl default_server;
    server_name www.nginxtraining.com;
    root   /usr/share/nginx/html;
    include /etc/nginx/ssl-configs/ssl-params.conf;
    access_log  /var/log/nginx/access.log  combined;
    access_log /var/log/nginx/juice.access.log proxy_log;
    error_log /var/log/nginx/juice.error.log info;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $host;

    location / {
       proxy_pass http://juice_server;
    }
...
```

4. Save the and exit the **`juice.conf`** configuration file.

5. Test your NGINX updated configuration files for any syntax errors.

```
$ sudo nginx -t
```

Expected Results

There are now 2 **`server`** blocks in the NGINX **`juice.conf`** file, one that listens on port 80 for HTTP traffic and one that listens on port 443 for HTTPS traffic. All HTTP traffic will be redirected to the **`server`** block for HTTPS.

Exercise 3: Set SSL Ciphers

Overview

In this exercise, you will edit the NGINX SSL parameters configuration file associated with the Juice Shop application to configure specific SSL ciphers.

1. Use openssl to list the ciphers that match our request with a complete description of the protocol version, key exchange, authentication, encryption and mac algorithms used along with any key size restrictions.

   ```
   $ openssl ciphers -v 'AES256+EECDH:AES256+EDH:!aNULL'
   ```

   ```
   $ openssl ciphers -v | grep TLSv1.3
   ```

```
student@nginx:/etc/nginx/ssl$ openssl ciphers -v
'AES256+EECDH:AES256+EDH:!aNULL'
TLS_AES_256_GCM_SHA384  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(256)
Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256 TLSv1.3
Kx=any      Au=any  Enc=CHACHA20/POLY1305(256) Mac=AEAD
TLS_AES_128_GCM_SHA256  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(128)
Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH     Au=ECDSA
Enc=AESGCM(256) Mac=AEAD
ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2
Kx=ECDH     Au=RSA  Enc=AESGCM(256) Mac=AEAD
ECDHE-ECDSA-AES256-CCM8 TLSv1.2 Kx=ECDH     Au=ECDSA
Enc=AESCCM8(256) Mac=AEAD
ECDHE-ECDSA-AES256-CCM  TLSv1.2 Kx=ECDH     Au=ECDSA
Enc=AESCCM(256) Mac=AEAD
ECDHE-ECDSA-AES256-SHA384 TLSv1.2 Kx=ECDH     Au=ECDSA
Enc=AES(256)  Mac=SHA384
ECDHE-RSA-AES256-SHA384 TLSv1.2
Kx=ECDH     Au=RSA  Enc=AES(256)  Mac=SHA384
ECDHE-ECDSA-AES256-SHA  TLSv1 Kx=ECDH     Au=ECDSA
Enc=AES(256)  Mac=SHA1
ECDHE-RSA-AES256-SHA    TLSv1
Kx=ECDH     Au=RSA  Enc=AES(256)  Mac=SHA1
DHE-DSS-AES256-GCM-SHA384 TLSv1.2
Kx=DH       Au=DSS  Enc=AESGCM(256) Mac=AEAD
DHE-RSA-AES256-GCM-SHA384 TLSv1.2
Kx=DH       Au=RSA  Enc=AESGCM(256) Mac=AEAD
DHE-RSA-AES256-CCM8     TLSv1.2
Kx=DH       Au=RSA  Enc=AESCCM8(256) Mac=AEAD
DHE-RSA-AES256-CCM      TLSv1.2 Kx=DH       Au=RSA  Enc=AESCCM(256)
Mac=AEAD
DHE-RSA-AES256-SHA256   TLSv1.2
Kx=DH       Au=RSA  Enc=AES(256)  Mac=SHA256
DHE-DSS-AES256-SHA256   TLSv1.2
Kx=DH       Au=DSS  Enc=AES(256)  Mac=SHA256
DHE-RSA-AES256-SHA      SSLv3
Kx=DH       Au=RSA  Enc=AES(256)  Mac=SHA1
DHE-DSS-AES256-SHA      SSLv3
Kx=DH       Au=DSS  Enc=AES(256)  Mac=SHA1
student@nginx:/etc/nginx/ssl$ openssl ciphers -v | grep TLSv1.3
TLS_AES_256_GCM_SHA384  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(256)
Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256 TLSv1.3
Kx=any      Au=any  Enc=CHACHA20/POLY1305(256) Mac=AEAD
TLS_AES_128_GCM_SHA256  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(128)
Mac=AEAD
student@nginx:/etc/nginx/ssl$
```

NOTE
Remember to copy the command from the SNIPPETS.txt file or type in the quotes manually.

An "!" means that the cipher is deleted from the list. In this lab we are removing **`!aNULL`** which is the anonymous Diffie-Hellman algorithm which is vulnerable to MITM (Man In The Middle) attacks.

2. Edit the configuration file **`ssl-params.conf`** to add your allowed/preferred ciphers for both TLSv1.2 and TLSv1.3.

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

```
ssl_ciphers AES256+EECDH:AES256+EDH:!aNULL;
ssl_conf_command Options PrioritizeChaCha;
ssl_conf_command Ciphersuites
TLS_CHACHA20_POLY1305_SHA256;
```

```
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers AES256+EECDH:AES256+EDH:!aNULL;
ssl_conf_command Options PrioritizeChaCha;
ssl_conf_command Ciphersuites TLS_CHACHA20_POLY1305_SHA256;
ssl_prefer_server_ciphers on;
```

3. Save the and exit the **`ssl-params.conf`** configuration file.

4. Test your NGINX updated configuration files for any syntax errors.

```
$ sudo nginx -t
```

Expected Results

The Juice Shop application is now configured to use specific SSL ciphers.

Exercise 4: Configure Forward Secrecy

Overview

In this exercise, a new Diffie-Hellman Key-Exchange will be generate and the NGINX Juice Shop application will be configured to use this specific SSL cipher algorithm for Forward Secrecy.

1. Generate a new Diffie-Hellman Key-Exchange. Use the **`-dsaparam`** flag in order to avoid testing for primality in all 4 million numbers.

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

2. **Edit** the configuration file **`ssl-params.conf`** to add the **`ssl_dhparam`** directive with the **`dhparam.pem`** file you created in the previous step.

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

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

```
...
#ssl_stapling_verify on;

ssl_trusted_certificate /etc/nginx/ssl/ca-cert.crt;
ssl_certificate /etc/nginx/ssl/www.nginxtraining.com.crt;
ssl_certificate_key /etc/nginx/ssl/www.nginxtraining.com.key;
ssl_dhparam /etc/nginx/dhparam.pem;

add_header Strict-Transport-Security "max-age=63072000;
includeSubdomains";
...
```

3. Save the and exit the file.

4. Reload NGINX to use your updated configuration files.

```
$ sudo nginx -t && sudo nginx -s reload
```

5. Edit the file **`/etc/hosts`** to add the DNS name www.nginxtraining.com to be associated with your NGINX lab system 10.0.0.11.

```
$ sudo vim /etc/hosts
```

```
10.0.0.11 www.nginxtraining.com
```

NOTE
Typically, if you are testing you would not edit system files but would instead use command options to handle your test case. For the curl command you can add the option **--resolve www.nginxtraining.com:443:10.0.0.11** but there may be limited information returned so for our training purposed adding the NGINX DNS name and IP address to /etc/hosts is preferable.

Expected Results

Your NGINX configuration now uses a Diffie-Hellman SSL cipher algorithm for Forward Secrecy.

Exercise 5: Testing HTTPS Configuration

Overview

In this exercise, test the NGINX configuration to confirm it now uses HTTPS with a specific SSL cipher algorithm and is set up to use Forward Secrecy with Diffie-Hellman.

1. Perform a curl test using HTTP. The output should have a 301 Moved Permanently message.

```
$ curl -I http://www.nginxtraining.com
```

```
student@nginx:/etc/nginx/ssl$ curl -I http://www.nginxtraining.com
HTTP/1.1 301 Moved Permanently
Server: nginx/1.23.4
Date: Fri, 16 Jun 2023 18:38:04 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: https://www.nginxtraining.com/

student@nginx:/etc/nginx/ssl$
```

NOTE
This works for **`curl`** so you don't have to use the **`-k`** option, but in the browser it still gives ~~https~~ which means it's not secure/trusted. Depending on which browser you are using you can add your certificate to the browser trust but since in production you will not be using self-signed certs you can ignore these warnings in our training lab environment.

2. Perform another curl test using HTTP with the **`-L`** option to follow to the new location where the requested page has moved. These are two ways to not have curl complain about not being able to verify the certificate from using either a self-signed cert or like in our lab using a cert that was created from a private CA. The results of both commands are the same.

```
$ cd ~/ssl
$ curl -IL --cacert ca-cert.crt http://www.nginxtraining.com
$ curl -ILk http://www.nginxtraining.com
```

```
student@nginx:~/ssl$ curl -IL --cacert ca-cert.crt
http://www.nginxtraining.com
HTTP/1.1 301 Moved Permanently
Server: nginx/1.23.4
Date: Fri, 16 Jun 2023 18:41:20 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: https://www.nginxtraining.com/

HTTP/1.1 200 OK
Server: nginx/1.23.4
Date: Fri, 16 Jun 2023 18:41:20 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 1924
Connection: keep-alive
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Fri, 16 Jun 2023 16:14:08 GMT
ETag: W/"784-188c4fc49ee"
Vary: Accept-Encoding
Strict-Transport-Security: max-age=63072000; includeSubdomains
X-Frame-Options: DENY
X-Content-Type-Options: nosniff

student@nginx:~/ssl$
```

NOTE
Your output should have a 301 Moved Permanently but will also follow to the new location (200 OK) and that requires the certificate to be verified which is why we used the option **`--cacert`** to specify the CA Root Authority cert that our cert & key files were referenced with or the **`-k`** option to skip verification. Another way to avoid the verification issue is to put the CA cert you created into the system certificates store; for an Ubuntu system that would include copying the file **`ca-cert.crt`** to **`/usr/local/share/ca-certificates`** and then running **`sudo update-ca-certificates`**.

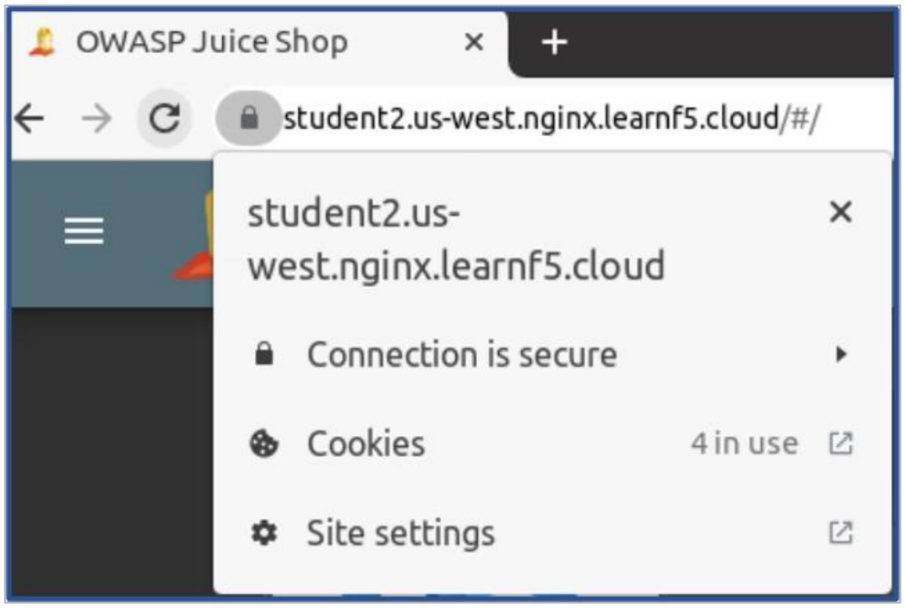
3. Perform a curl test using **HTTPS**. Your output should have a **200 OK** response.

```
$ curl -I https://www.nginxtraining.com
```

TROUBLESHOOTING
If you receive a 301 Moved Permanently response then make sure that the curl command is using **https** not http.

4. In your browser use **http://www.nginxtraining.com** to verify if the webpage now uses **https**. Because our certificate is signed by a private CA the browser can't verify which means the ~~https~~ will be crossed out and you need to click on the Advanced box at the bottom left and then click on the Proceed to www.nginxtraining.com (unsafe) link, which will take you to the Juice Shop web app via your NGINX lab system.

NOTE
If you were using a public (legitimate) CA certificate, then you would get a lock symbol or https.


OWASP Juice Shop
student2.us-west.nginx.learnf5.cloud/#/
student2.us-west.nginx.learnf5.cloud
Connection is secure
Cookies
4 in use
Site settings

Expected Results

Your NGINX configuration now uses HTTPS with a specific SSL cipher algorithm and is set up to use Forward Secrecy with Diffie-Hellman.

Expected Results

Lab 2: Secure Upstream Traffic

Estimated time for completion: **XX minutes**

Requirements

The following tasks must be completed before beginning this lab:

- You must have a valid certificate and private key for your NGINX lab system.
- You must have a server block defined for Juice Shop to handle HTTPS requests.

Objectives

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

- Secure upstream traffic with ssl parameters
- Configure the NGINX API Dashboard to gather and view metrics
- Configure NGINX access and error logs
- Access the NGINX Plus API Dashboard
- Customize the format of the NGINX access log file
- Access the NGINX access and error log files

Exercise 1: Secure Upstream Traffic

1. Backup the configuration file **`api_server.conf`**

```
$ sudo cp /etc/nginx/conf.d/api_server.{conf,orig}
```

2. Edit the configuration file **`api_server.conf`**

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

3. Enable HTTPS on port **8080** using the parameter **`ssl`.**

```
server {
    listen 8080 ssl;
    root /usr/share/nginx/html;
    ...
}
```

4. In the **`server`** block with **`ssl`**, create a **`location`** with the following **`proxy`** settings.

   ```
   location / {
   proxy_pass https://api_server;
   proxy_ssl_trusted_certificate /etc/nginx/ssl/ca-cert.crt;
     proxy_ssl_verify off;
   }
   ```

5. Save the **`api_server.conf`** file and then reload NGINX to use your updated configuration files.

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

6. Test your configuration with curl. Your output should get a **502 Bad Gateway** response because you are trying to access upstreams that are not configured.

   ```
   $ curl -k https://www.nginxtraining.com:8080/
   ```

7. View the contents of the **proxy-ssl-params.conf** file which defines the SSL protocols and HTTP version.

   ```
   $ cat /etc/nginx/ssl-configs/proxy-ssl-params.conf
   ```

8. Edit the **`api_server.conf`** file to add the **`proxy-ssl-params.conf`** file in the **`location /`** context.

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

   location / {
   include /etc/nginx/ssl-configs/proxy-ssl-params.conf;
     ...
   }
   ```

9. Continue editing the **`api_server.conf`** file and for both **`server`** contexts, enable **`ssl`** and include the file **`ssl-params.conf`**

NOTE
This is **not the same pathname** as the previous step. There is no **proxy** in the name so be sure to edit the filename if you copy and paste the previous command or just copy from the Snippets file!

```
listen 8081 ssl;
include /etc/nginx/ssl-configs/ssl-params.conf;
…
listen 8082 ssl;
include /etc/nginx/ssl-configs/ssl-params.conf;
```

The **`api_server.conf`** file should look like this with changed highlighted:

```
upstream api_server {
  server 127.0.0.1:8081;
    server 127.0.0.1:8082;
}

server {
    listen 8080 ssl;
    root /usr/share/nginx/html;
    error_log /var/log/nginx/api_server.error.log info;
    access_log /var/log/nginx/api_server.access.log combined;
    include /etc/nginx/ssl-configs/ssl-params.conf;

    location / {
        include /etc/nginx/ssl-configs/proxy-ssl-params.conf;
        proxy_pass https://api_server;
        proxy_ssl_trusted_certificate /etc/nginx/ssl/ca-cert.crt;
        proxy_ssl_verify off;
    }

}

server {
    listen 8081 ssl;
    include /etc/nginx/ssl-configs/ssl-params.conf;
    root /home/student/public_html/application1;
}
server {
    listen 8082 ssl ;
    include /etc/nginx/ssl-configs/ssl-params.conf;
    root /home/student/public_html/application2;
}
```

10. Save the configuration file and reload NGINX.

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

11. Test the upstream backend systems using the **curl** command.

```
$ cd ~/ssl
$ curl --cacert ca-cert.crt https://www.nginxtraining.com:8080
$ curl --cacert ca-cert.crt https://www.nginxtraining.com:8080
$ curl --cacert ca-cert.crt https://www.nginxtraining.com:8080
```

NOTE
Your output should be like “**this is app1**”, then “**this is app2**” and then “**this is app1**” again. It does not matter which upstream server responds first since the default load balancing method is round robin so the backend servers should alternate serving the response. You might need to do more than three curl statements to see the Load Balancing start.

TROUBLESHOOTING:
If you are still getting a 502 Bad Gateway response message then make sure that you added "ssl" to both listen directives for ports 8081 and 8082 in the /etc/nginx/conf.d/api_server.conf configuration file and that you reload to use the updated configuration file.

Expected Results

The default round robin load balancing is set up for your application when using port 8080.

Exercise 2: Setup the NGINX Plus API Dashboard

1. Edit the file **`/etc/nginx/conf.d/api_server.conf`**. In the **`context`** for **`8080`**, create two new prefix locations: **`api`** and **`dashboard`**.

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

location /api {
    api;
    access_log off;
}

location /dashboard {
    root /usr/share/nginx/html;
    try_files $uri $uri.html /dashboard.html;
}
```

NOTE
For our lab purposes we are turning off the access log since it generates a fair amount of information whereas in production you may not want to turn off.

2. Continue editing the **api_server.conf** file, to gather status metrics add a **zone** directive in the **upstream** which is for the backends.

```
upstream api_server {
  zone api_server_upstream 64k;
    server 127.0.0.1:8081;
    server 127.0.0.1:8082;
```

3. Continue editing the **api_server.conf** file, to gather status metrics by adding a **status_zone** for the **server** listening on port **8080**, which is for the front end, virtual servers. Save the file.

```
server {
  listen 8080 ssl;
    status_zone api_gateway;
    root /usr/share/nginx/html;
...
```

The **api_server.conf** file should look like this with updated highlighted:

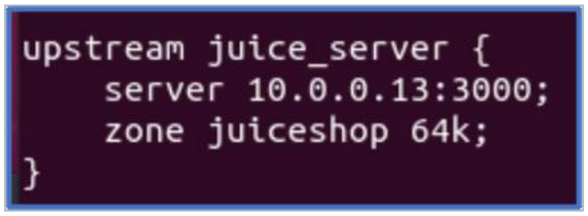
```
upstream api_server {
    zone api_server_upstream 64k;
    server 127.0.0.1:8081;
    server 127.0.0.1:8082;
}

server {
    listen 8080 ssl;
    status_zone api_gateway;
    root /usr/share/nginx/html;
    error_log /var/log/nginx/api_server.error.log info;
    access_log /var/log/nginx/api_server.access.log combined;
    include /etc/nginx/ssl-configs/ssl-params.conf;

    location / {
        include /etc/nginx/ssl-configs/proxy-ssl-params.conf;
        proxy_pass https://api_server;
        proxy_ssl_trusted_certificate /etc/nginx/ssl/ca-cert.crt;
        proxy_ssl_verify off;
    }

    location /api {
        api;
        access_log off;
    }

    location /dashboard {
        root /usr/share/nginx/html;
        try_files $uri $uri.html /dashboard.html;
    }

}

server {
    listen 8081 ssl;
    include /etc/nginx/ssl-configs/ssl-params.conf;
    root /home/student/public_html/application1;
}
server {
    listen 8082 ssl ;
    include /etc/nginx/ssl-configs/ssl-params.conf;
    root /home/student/public_html/application2;
}
```

4. Save the configuration file and reload NGINX.

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

5. To gather more status metrics, edit the file **`/etc/nginx/conf.d/juice.conf`**. Add a **zone** directive in the **`upstream juice_server`**.

   ```
   upstream juice_server {
       server 10.0.0.13:3000;
       zone juiceshop 64k;
   }
   ```



```
upstream juice_server {
    server 10.0.0.13:3000;
    zone juiceshop 64k;
}
```

6. Continue editing the `juice.conf` file by adding a `status_zone` for the `server` listening on port `443`.

   ```
   server {
     listen 443 ssl default_server;
       status_zone proxy;
       server_name www.nginxtraining.com;
   ```

7. Save and exit the file.

8. Test the syntax of your configuration files and when there are no errors reload the NGINX configuration.

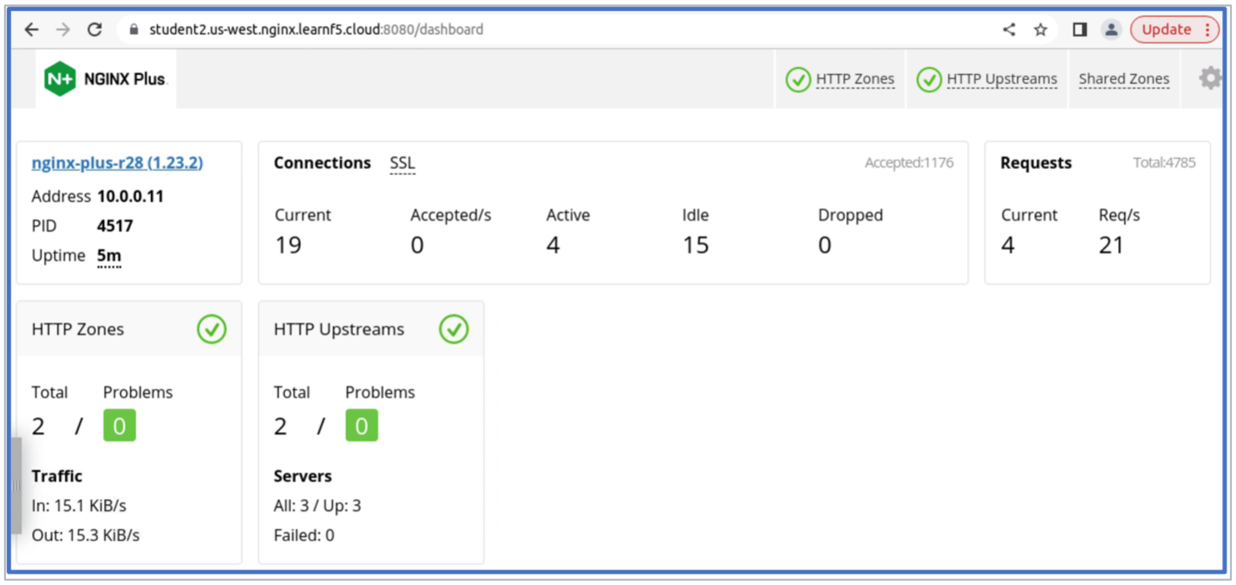
   ```
   $ sudo nginx -t && sudo nginx -s reload
   ```

9. Send a few `curl` commands so that you have some newer data in your NGINX dashboard. **Note**: Change to the `/ssl` directory.

   ```
   $ cd ~/ssl
   $ curl --cacert ca-cert.crt https://www.nginxtraining.com:8080
   $ curl --cacert ca-cert.crt https://www.nginxtraining.com:8080
   ```

10. Enter **https://www.nginxtraining.com** in a browser. Note that you will still get a note from the browser about not trusted and this is okay since we have not added our private Root CA to the browsers trust.

11. Access the NGINX dashboard from a browser using **https://www.nginxtraining.com:8080/dashboard**



nginx-plus-r28 (1.23.2)
Address 10.0.0.11
PID 4517
Uptime 5m

Connections	SSL			Accepted:1176
Current	Accepted/s	Active	Idle	Dropped
19	0	4	15	0

Requests	Total:4785
Current	Req/s
4	21

HTTP Zones
Total / Problems
2 / 0
Traffic
In: 15.1 KiB/s
Out: 15.3 KiB/s

HTTP Upstreams
Total / Problems
2 / 0
Servers
All: 3 / Up: 3
Failed: 0

NOTE
If the HTTP Upstreams is yellow, it should eventually turn to green. You could send more requests by doing several **`curl -k https://www.nginxtraining.com:8080`**

12. In the NGINX dashboard click on the **HTTP Upstreams** tab to see both upstreams **`api_server`** and **`juice_server`**.

13. In the dashboard click on the **HTTP Zones** tab to see the **`zones`** you added to the **`api_server.conf`** and **`juice.conf`** files (**`api_gateway`** and **`proxy`**).

14. In your dashboard click on the **Shared Zones** tab to see the **api_server_upstream** zone you added to the **`api_server.conf`** file and the **juiceshop** zone in the **`juice.conf`** file.

15. To get back to the original dashboard view, click the **NGINX Plus logo** (top left of the dashboard).

Expected Results

The NGINX Plus Dashboard should be accessible to view upstreams, HTTP zones and shared zones.

Exercise 3: Configure NGINX Logging

1. Edit the configuration file **`juice.conf`**.

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

2. Add a custom log format in the http context (at the very top of the file, outside of any server context).

```
log_format proxy_log "
Request: $request
    Status: $status
    Client: $remote_addr
    Upstream: $upstream_addr
    Forwarded-For: $proxy_add_x_forwarded_for
    ssl_server_name: $ssl_server_name
";

upstream juice_server {
    server 10.0.0.13:3000;
    zone juiceshop 64k;
}
```

3. Add the **`access_log`** and **`error_log`** directives in the **`ssl server`** context and **delete** the original **`access_log`** directive using the combined format. Then save the file.

   ```
   access_log /var/log/nginx/juice.access.log proxy_log;
   error_log /var/log/nginx/juice.error.log info;
   ```

4. Test the syntax of your configuration files and when there are no errors reload the NGINX configuration.

   ```
   $ sudo nginx -t && sudo nginx -s reload
   ```

5. Open another terminal to view the NGINX access log file.

   ```
   $ sudo tail -f /var/log/nginx/juice.access.log
   ```

6. Test the new log format using curl commands and focus on the values output in the variable **`ssl_server_name`** which comes from the SNI TLS

handshake. View the **`curl_script.sh`** file to view the four curl commands and then run this script.

```
$ cat ~/curl_script.sh
$ ~/curl_script.sh
```

```
student@nginx:~$ sudo tail -f /var/log/nginx/juice.access.log

Request: GET / HTTP/1.1
    Status: 200
    Client: 127.0.0.1
    Upstream: 10.0.0.13:3000
    Forwarded-For: 127.0.0.1
    ssl_server_name: -


  Request: GET / HTTP/1.1
    Status: 200
    Client: 127.0.0.1
    Upstream: 10.0.0.13:3000
    Forwarded-For: 127.0.0.1
    ssl_server_name: localhost


  Request: GET / HTTP/1.1
    Status: 200
    Client: 10.0.0.11
    Upstream: 10.0.0.13:3000
    Forwarded-For: 10.0.0.11
    ssl_server_name: -


  Request: GET / HTTP/1.1
    Status: 200
    Client: 10.0.0.11
    Upstream: 10.0.0.13:3000
    Forwarded-For: 10.0.0.11
    ssl_server_name: www.nginxtraining.com
```

NOTE
When using these values the variable **`ssl_server_name`** is only set when passing localhost and www.nginxtraining.com.

7. Use **Ctrl-C** to stop the tail command.

Expected Results

You will now have a custom log format for the NGINX access log.

Expected Results