Assignment – 5- Sharams Kunwar – SSL & TLS

- SSL and TLS
  - SSL stands for Secure Socket Layer and TLS stands for Transport Layer Security
  - They work by encrypting data in transit between two applications, like a web browser and a web server.
  - They also provide authentication, which allows applications involved in communication to verify each other's identities preventing eavesdropping and man in the middle attack.
- Working of SSL and TLS
  - Client and Server establish a connection and negotiate the SSL/TLS version and encryption algorithm to be used.
  - Then, the server sends the client its SSL/TLS certificate containing its identity, like domain name and public key, which is then validated by the client with a trusted certificate authority.
  - Then, they generate a shared secret key which encrypts, and decrypts data sent between two applications and finally exchange of data takes place.

Hands-on:

A self-signed SSL certificate was generated which would be valid for 365 days and uses a 2048-bit RSA key.

To generate the certificate, following command was run:

Command: sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/selfsigned.key -out /etc/nginx/ssl/selfsigned.crt

On running the command, I was asked to enter Distinguished Name or DN, but I left it empty for the instance which would flag the certificate as untrusted by the browser as the identity of the certificate owner wouldn't be validated, when client connects to server.

Screenshot:

```
sharumss@sharumss-VirtualBox:/$ sudo mkdir -p /etc/nginx/ssl/
sharumss@sharumss-VirtualBox:/$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/selfsigned.key -out /etc/nginx/ssl/selfsig
ned.crt
.....+...+..+.++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*.............+...+...+.....+......+...+...+...+......+.+............+..+.
........+......++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*........+.....+...+...+.+......+..............+..+.......+.....+......+.
..+.+.........+..+..........+...+.........+.....+.....+.........+..+....+...+......+.....+......+...+.........+..+...+.............+..............+..+
...+...........+......+.+...+......+...........+.........................+...+...+........+.+......+...........+.....+.+......+...+..+...............+.
.+.+..+...+..............+..........+.........+...+...+...+.+..........+...+......+..............+..........+........+...+.........+...+.......++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
.+...+........+....+...+..+.....................+...+..+......+..........+...+..+...+......+...++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++*.++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*...+...+..............+...+....+.....+...+.........+...............+...+.+.
.....+............................++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-----
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]: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 []:
sharumss@sharumss-VirtualBox:/$
```

Then, HTML pages were created inside /var/www/shrooms.com, which includes a landing page, i.e., index.html, and error pages to be displayed on encountering 403, 404, 502 errors.

Screenshots:

```
 GNU nano 6.2              /var/www/shrooms.com/index.html
<!DOCTYPE html>
<html>
<head>
<title>Sairam</title>
</head>
<body>
    <h1>Sairam learns DevOps </h1>
</body>
</html>
```

Figure 1 Index.html

```
<!DOCTYPE html>
<html>
<head>
<title>Oh No</title>
</head>
<body>
    <h1> Error  </h1>
</body>
</html>
```

Figure 2 404.html

```
<!DOCTYPE html>
<html>
<head>
<title>Oh NO</title>
</head>
<body>
    <h1> Oh no, error!  </h1>
</body>
</html>
```

Figure 3 403.html

```
 GNU nano 6.2                  /var/www/shrooms.com/502.html
<!DOCTYPE html>
<html>
<head>
<title>Oh NO</title>
</head>
<body>
    <h1> sorry!!  </h1>
</body>
</html>
```

Figure 4 502.html

Then, the shrooms.com, was mapped to Ip address 127.0.1.1 in /etc/hosts/ to not surf the internet upon requesting shrooms.com and direct the request to the local server.

Following configuration was made in the /etc/hosts directory:

Screenshot:

```
127.0.0.1       shrooms.com www.shrooms.com
127.0.1.1       sharumss-VirtualBox
```

After overriding DNS resolution, a conf file was setup as follows:

```
 GNU nano 6.2                                    /etc/nginx/sites-enabled/shrooms.com
server{
        listen 80;
        listen [::]:80;

        server_name shrooms.com www.shrooms.com;

        return 301 https://$host$request_uri;
}

server{
        listen 443 ssl;
        listen [::]:443 ssl;

        server_name shrooms.com www.shrooms.com;

        ssl_certificate /etc/nginx/ssl/selfsigned.crt;
        ssl_certificate_key /etc/nginx/ssl/selfsigned.key;



        error_page 403 /403.html;
        error_page 404 /404.html;
        error_page 502 /502.html;

        location / {
                root /var/www/shrooms.com;
                index index.html;
        }

        location /forbidden {
                deny all;
        }
}
```

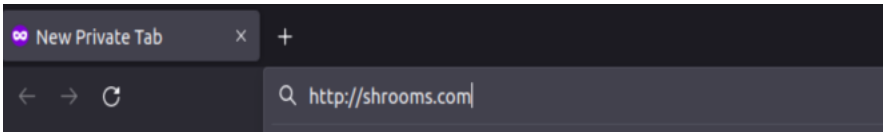
Here's what the configuration does:

- The first server block defines configuration for an HTTP server.
  - listen 80; and listen [::]:80 specifies the server should listen on port 80 for incoming ipv4 and ipv6 requests.
  - 'return 301 https://$host$request_uri;' tells server to respond with a 301, which is a permanent movement of a page to the HTTPS version of shrooms.com using $host and $request_uri to construct the redirect URL dynamically.
- The second server block defines configuration for an HTTPS server.
  - listen 443; and listen [::]:443 specifies the server should listen on port 443 for incoming ipv4 and ipv6 requests.
  - server_name defines the domain name which the server block responds to.
  - ssl_certificate and ssl_certificate_key specifies path to ssl certificate and private key files which enables HTTPS on the server.
  - error_page outlines the location of custom error pages for error codes like 403,404,502.
  - location block defines the root directory of the server and the path of landing page.

Here’s what the configured server looks like:

- Trying to navigate to server using ‘http://’

Screenshot:



- Despite requesting ‘http://’, we were navigated to ‘https://’

Screenshot:

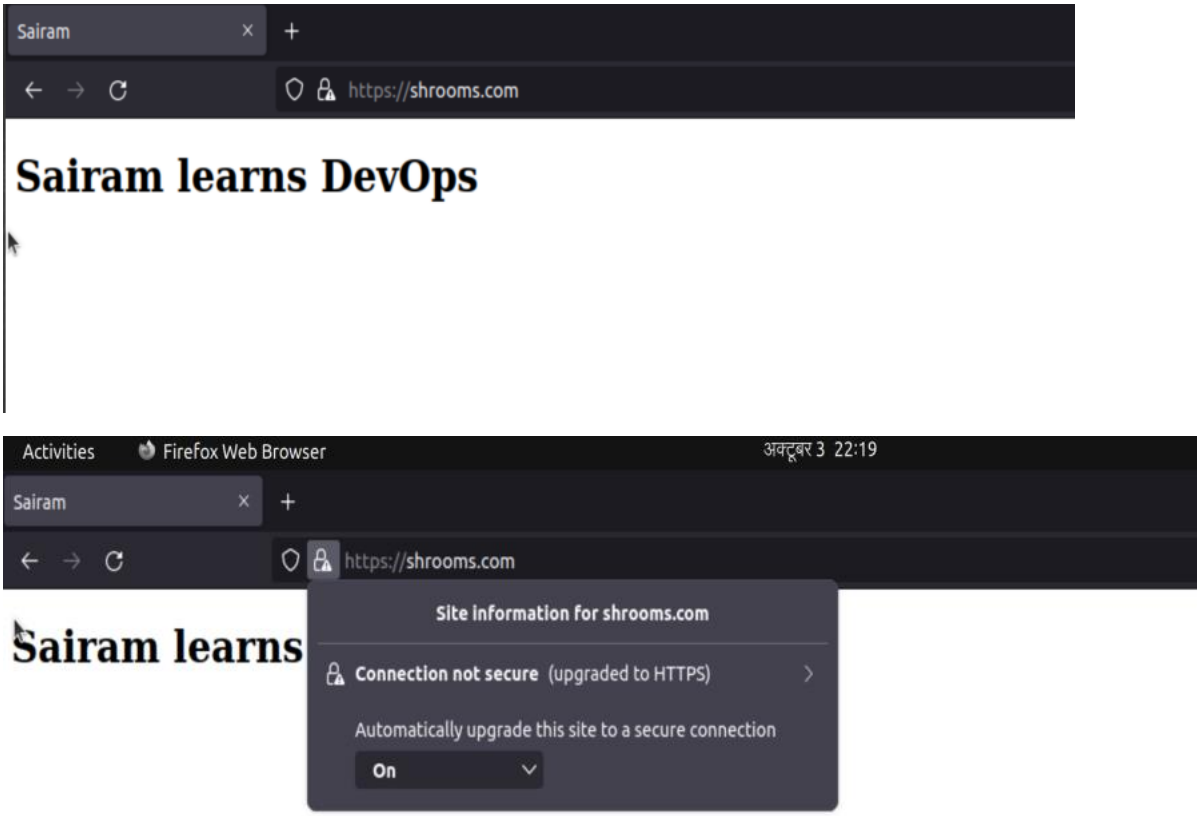
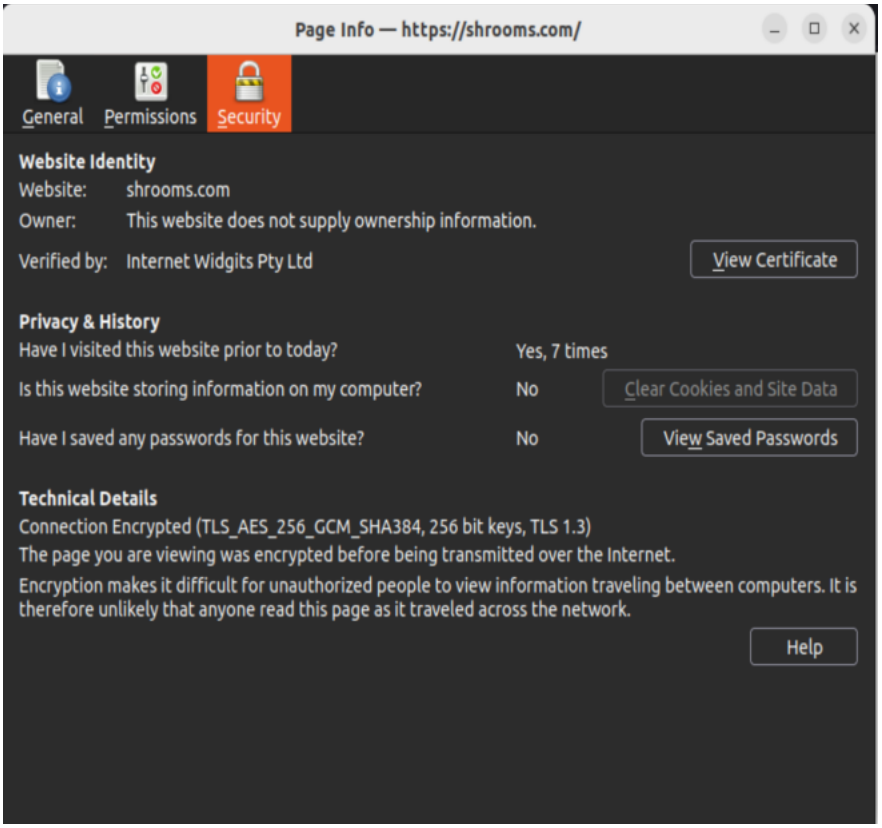


Figure 5 Site information

Reviewing the certificate

Screenshot:



As we can see, the ownership information is not supplied here, which occurred due to not filling up the DN information properly, earlier.

Then, the redirection pages were checked to see if they were working:

- Error 404:
  - 404 error occurs when the server is reachable, but the requested page can't be retrieved.
  - To verify this, we tried accessing 'hello' directory, which doesn't exist, and we were served with the error page.

Screenshot:

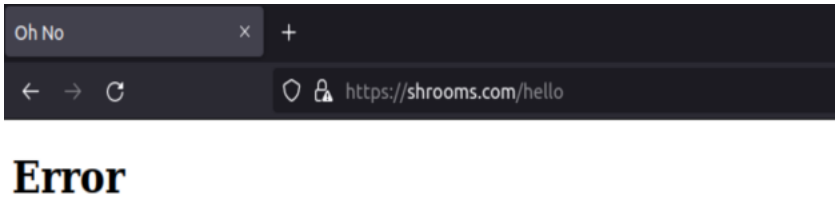


Figure 6 404 error redirection page

- Error 403:
  - 403 error occurs when the server understands the request but can't retrieve the page because the client isn't allowed to access the page.
  - To verify this, we tried accessing 'forbidden' directory, where all access is restricted, and we were served with the error page.

  Screenshot:

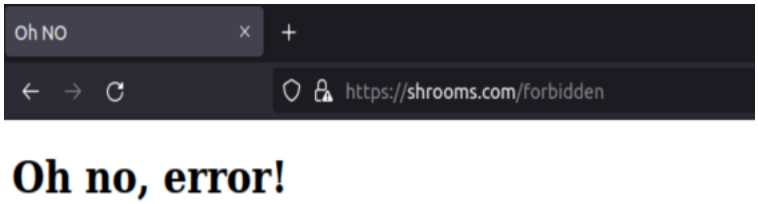


Figure 7 403 error redirection page

- Error 502:
  - 502 error occurs when the server encountered a temporary error and could not complete the request.
  - To verify this, we tried accessing the domain, after stopping nginx service, and we were served with the error page.

Figure 8 Stopping the nginx service

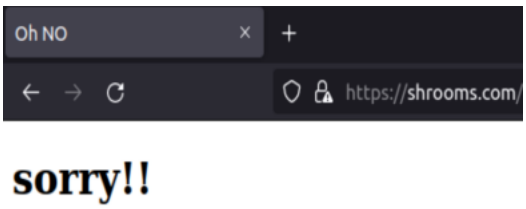


Figure 9 502 error redirection page

- SSL chaining

SSL chaining is the process of verifying the authenticity of an SSL certificate by tracing its chain of trust back to a trusted root certificate authority (CA).

- When a client connects to a server, the server sends the client its SSL certificate. This certificate contains information about the server's identity, such as its domain name and public key.
- The client verifies the server's certificate by checking the signature of the issuing CA. If the signature is valid, the client knows that the server is who it claims to be.
- The issuing CA's certificate may also be signed by another CA, and so on. This chain of certificates is called the certificate chain.
- The client follows the certificate chain until it reaches a trusted root CA. A trusted root CA is a CA that is pre-installed in the client's operating system or web browser.
- If the client reaches a trusted root CA, it knows that the server's certificate is valid and can establish a secure connection.

Here is an example of an SSL certificate chain:

- The server certificate is signed by the intermediate CA certificate, which is signed by the root CA certificate.
- When a client connects to the server, it receives the server certificate and the intermediate CA certificate. The client verifies the signature of the intermediate CA certificate using the root CA certificate, which is pre-installed in the client's operating system or web browser.
- If the signature is valid, the client knows that the server certificate is valid and can establish a secure connection.

If the attacker had a valid SSL certificate, the client would not be able to tell that it was being impersonated. However, if the attacker's certificate is not signed by a trusted root CA, the client will not be able to verify the certificate and will not establish a secure connection.

SSL chaining is important for building trust with users. When users see that a website is using an SSL certificate that is signed by a trusted root CA, they know that the website is authentic and that their data is safe.