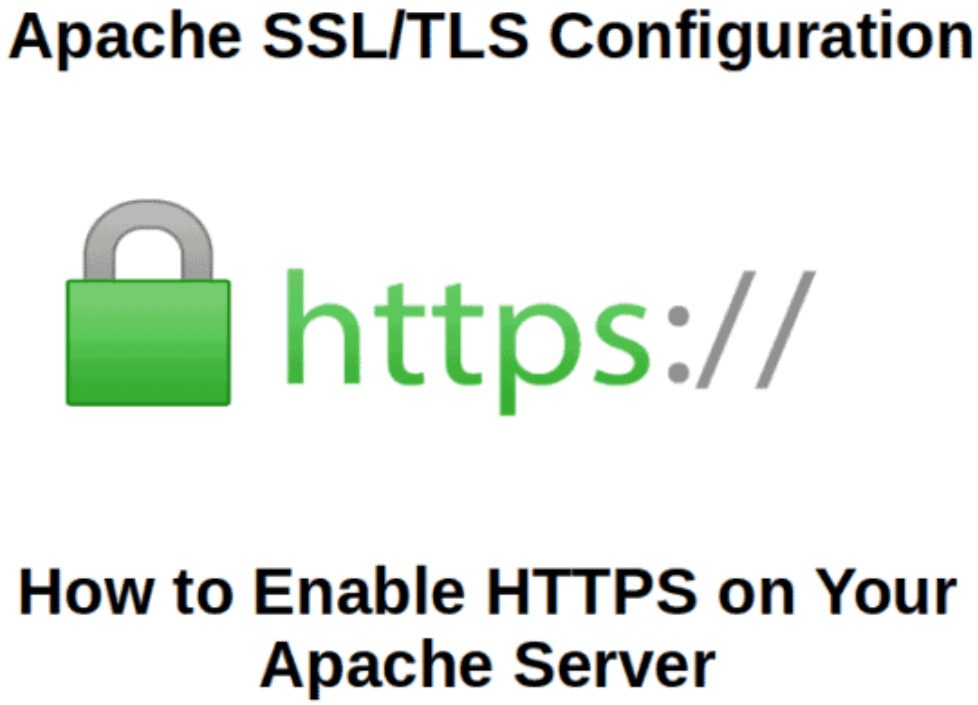
What is HTTPS and SSL?



Hypertext Transfer Protocol Secure abbreviates to HTTPS. It's a collection of rules governing how two parties (like users and websites) may safely share private information. This protocol facilitates using the transport layer security (TLS) protocol to establish an encrypted link between your client (browser) and the server it is communicating with. For this reason, it is also referred to as HTTP over TLS or HTTPS secure. HTTPS is the secured variant of the more common HTTP protocol.

SSL, or Secure Sockets Layer, is a protocol for creating encrypted and authorized connections between computers on a network, ensuring the safety of data and communications sent over the internet.

Moreover, SSL reveals details about the website, including the domain name and, if present, the website owner. Both server to server and server to client connections may be established using SSL.

Also Read
Designing a PKI Certification Authority Hierarchy – Best Practice

Why do you need HTTPS and SSL?

Since HTTPS provides the best protection for users' private data, it has replaced all other Web protocols as the standard.

Next concept is authentication, a process users go through to ensure they are connected to a simple website, not a fraud. A certificate authority (CA) is a third party organization that verifies the legitimacy of SSL/TLS certificates (i.e., website security certificates), which your browser reviews.

Secondly, HTTPS is not just important for sites that ask visitors to enter personal information. Attackers may get behavioural and identifying data through insecure connections in addition to information received directly from users.

In addition to increased data security, HTTP improves online functionality and user experience, two key concerns for site owners.

Users are more likely to trust an HTTPS website since they verify the site's identity using the SSL Certificate. Users then do not worry about their private information being stolen since the protocol encrypts all client server connections using SSL/TLS authentication.

Also Read
PKI Certificate Types Explained (TLS/SSL, Code Signing, Email, Client)

Apache SSL/TLS Configuration: Enable HTTPS on Your Apache Server

In this section, we show you how to enable HTTPS on an Apache server using a self signed certificate and Let's Encrypt certificate.

Prerequisites

- A server running Ubuntu Linux.
- A root user or a user with sudo privileges.

Also Read
How to Setup Apache Web Server + MySQL (LAMP Stack) on Linux in Azure/AWS/GCP

Install Apache Web Server

Before starting, the Apache web server package must be installed on your server. If not installed, install it via the APT command as shown below.

```
apt install apache2 openssl -y
```

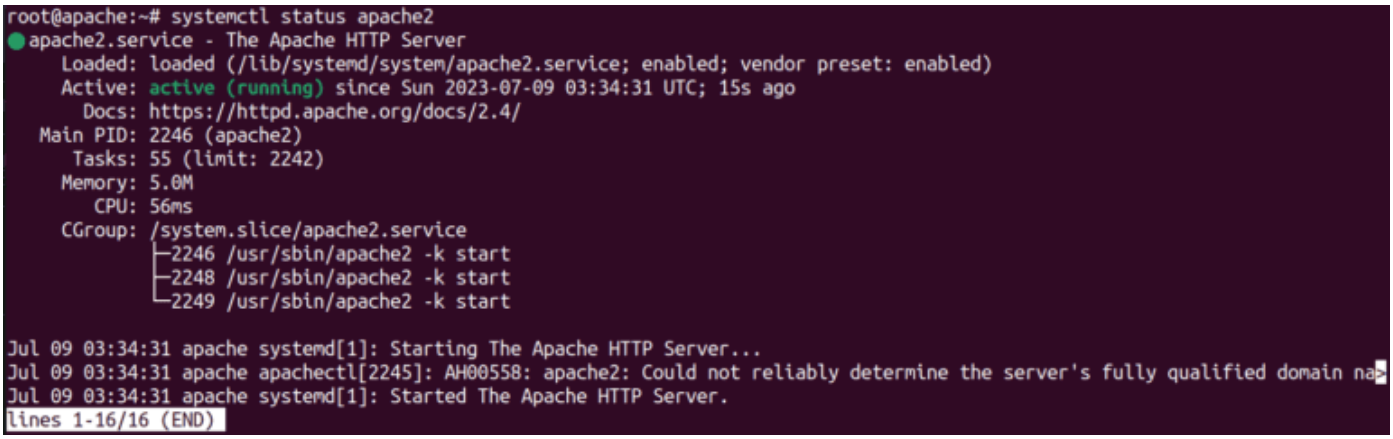
After installing the Apache web server, start and enable the Apache service.

```
systemctl start apache2
systemctl enable apache2
```

Now check the Apache running status using the following command.

```
systemctl status apache2
```

If everything is fine, you see the following screen.



Also Read
How to Install Apache Web Server on Ubuntu 20.04 Tutorial (Step by Step)

Configure UFW Firewall to Allow HTTP and HTTPS

If your server is configured with a UFW firewall then you need to allow HTTP and HTTPS service on UFW firewall.

First, check the UFW Apache app information using the following command.

```
ufw app info "Apache Full"
```

You see the following output.

```
Profile: Apache Full
Title: Web Server (HTTP,HTTPS)
Description: Apache v2 is the next generation of the omnipresent Apache web
server.

Ports:
80,443/tcp
```

Now, run the following command to allow both HTTP and HTTPS service.

```
ufw allow in "Apache Full"
```

Then, reload the UFW daemon to implement the changes.

```
ufw reload
```

Also Read
pfSense vs OPNsense – Which Firewall is Better? (Pros and Cons)

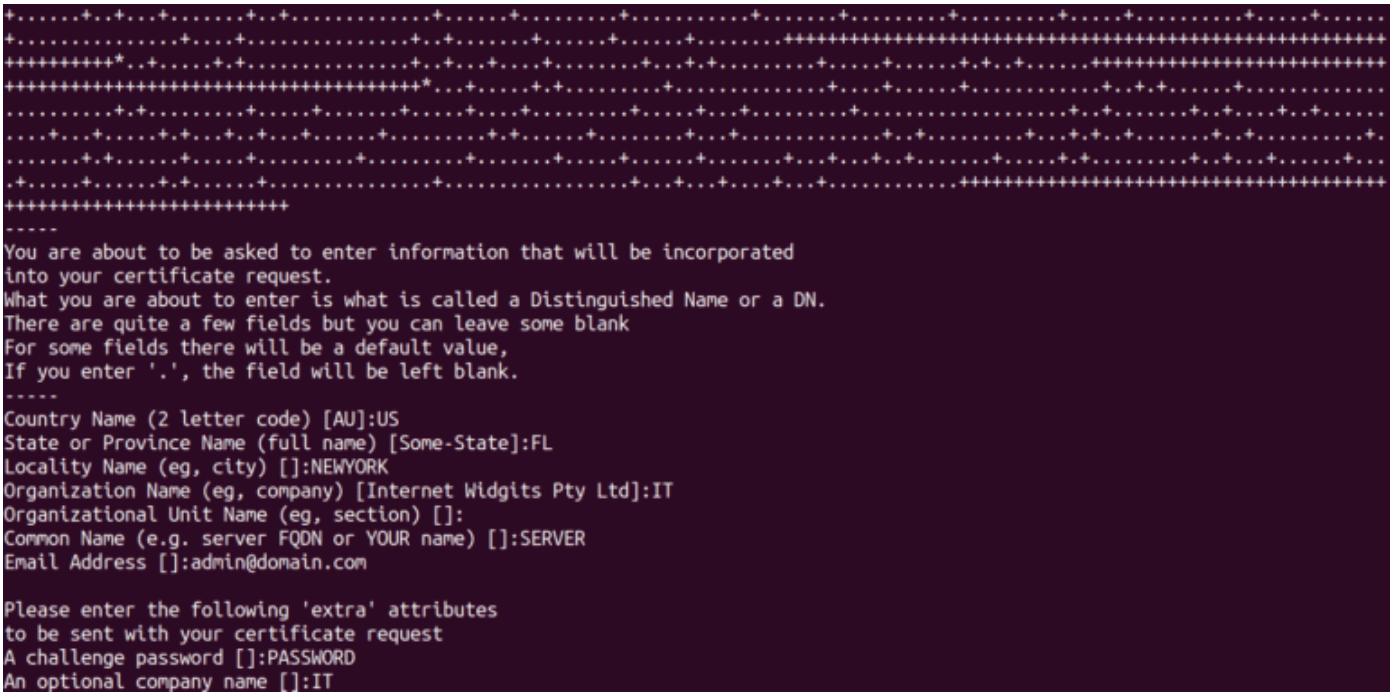
Enable HTTPS on Apache Using Self-signed SSL Certificate

A self-signed certificate is a certificate that is not signed by any certificate authority. It is signed by its own private key. Generally, it is used for testing environments or low-risk internal networks only.

First, generate a certificate signing request using the following command.

```
openssl req -nodes -newkey rsa:2048 -keyout /etc/ssl/private/private.key -out /etc/ssl/pri
```

You are then asked to provide some information as shown below.



After generating the certificate signing request, use this generated **.csr** file to generate a certificate and key file.

```
openssl x509 -in /etc/ssl/private/request.csr -out /etc/ssl/private/certificate.crt -req -
```

At this point, all required certificates are in your hand. Now, edit your default SSL configuration file.

```
nano /etc/apache2/sites-available/default-ssl.conf
```

Define your server IP and the path of your certificate as shown below.

```
<IfModule mod_ssl.c>
<VirtualHost _default_:443>
ServerAdmin admin@domain.com
ServerName your-server-ip
DocumentRoot /var/www/html
```

```
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

SSLEngine on
SSLCertificateFile /etc/ssl/private/certificate.crt
SSLCertificateKeyFile /etc/ssl/private/private.key

<FilesMatch "\.(cgi|shtml|phtml|php)$">
SSLOptions +StdEnvVars
</FilesMatch>
<Directory /usr/lib/cgi-bin>
SSLOptions +StdEnvVars
</Directory>

</VirtualHost>
</IfModule>
```

Save and close the file then activate the SSL virtual host file using the following command.

```
a2ensite default-ssl.conf
```

Next, edit the Apache default virtual host configuration file.

```
nano /etc/apache2/sites-available/000-default.conf
```

Define your server IP and website redirection as shown below.

```
<VirtualHost *:80>

ServerAdmin admin@domain.com
ServerName your-server-ip
DocumentRoot /var/www/html

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

Redirect "/" "https://your-server-ip/
</VirtualHost>
```

Save and close the file when you are done. Then, activate the Apache default virtual host file and other required modules using the following command.

```
a2ensite 000-default.conf
a2enmod ssl
a2enmod headers
a2enmod rewrite
```

Finally, restart the Apache service to apply the changes.

```
systemctl reload apache2
```

Now, open your web browser and access your Apache website securely using the URL **https://your-server-ip.** You see the warning page because you are using the self-signed certificate.