



# **Windows Guide to Web Server Hardening**

## **Additional Reading**

[Visit our website](#)

# Introduction

In this guide, we will walk you through the installation and configuration of the Apache web server on Windows, followed by the steps to secure the Apache web server.

## Development environment

These instructions were developed using the following operating system and software.

**Operating system:** Microsoft Windows 11, Version 10.0.22621

**Web browser:** Microsoft Edge, Version 123.0.2420.53

## Setting up Apache on your client

The following instructions will show you how to download, configure, and start the Apache web server on your Windows client.

1. Visit the [Apache HTTP Server download page](#) and download the appropriate version for your system (32-bit or 64-bit). For instance, the ZIP file indicated in the screenshot below:

**Apache Lounge**  
Webmasters

### Apache 2.4 VS17 Windows Binaries and Modules

Apache Lounge has provided up-to-date Windows binaries and popular third-party modules for more than 15 years. We have hundreds of thousands of satisfied users: small and big companies as well as home users. Always build with up to date dependencies and latest compilers, and tested thorough. The binaries are referenced by the ASF, Microsoft, PHP etc. and more and more software is packaged with our binaries and modules.

The binaries, are build with the sources from ASF at [httpd.apache.org](http://httpd.apache.org), contains the latest patches and latest dependencies like zlib, openssl etc. which makes the downloads here mostly more actual then downloads from other places. The binaries **do not** run on XP and 2003. Runs on: 7 SP1, Vista SP2, 8/8.1, 10, 11 Server 2008 SP2 / R2 SP1, Server 2012 / R2, Server 2016/2019/2022.

Build with the latest Windows® Visual Studio C++ 2022 aka VS17. Has improvements, fixes and optimizations over VS16 in areas like Performance, MemoryManagement, New standard conformance features, Code generation and Stability. For example code quality tuning and improvements done across different code generation areas for "speed". And makes more use of latest processors and supported Windows editions (win7 and up) internal features.

**VS17 is backward compatible**, That means, a VS16/15/14 module can be used inside the VS17 binary.

Be sure you installed latest 14.38.33135 Visual C++ Redistributable Visual Studio 2015-2022 : [vc\\_redist\\_x64](#) or [vc\\_redist\\_x86](#) see [Redistributable](#)

#### Apache 2.4 binaries VS17

**Info & Changelog**

**Apache 2.4.59-240404 Win64**

● [httpd-2.4.59-240404-win64-VS17.zip](#)  
PGP Signature (Public [PGP\\_key](#)), SHA1-SHA512 [Checksums](#)

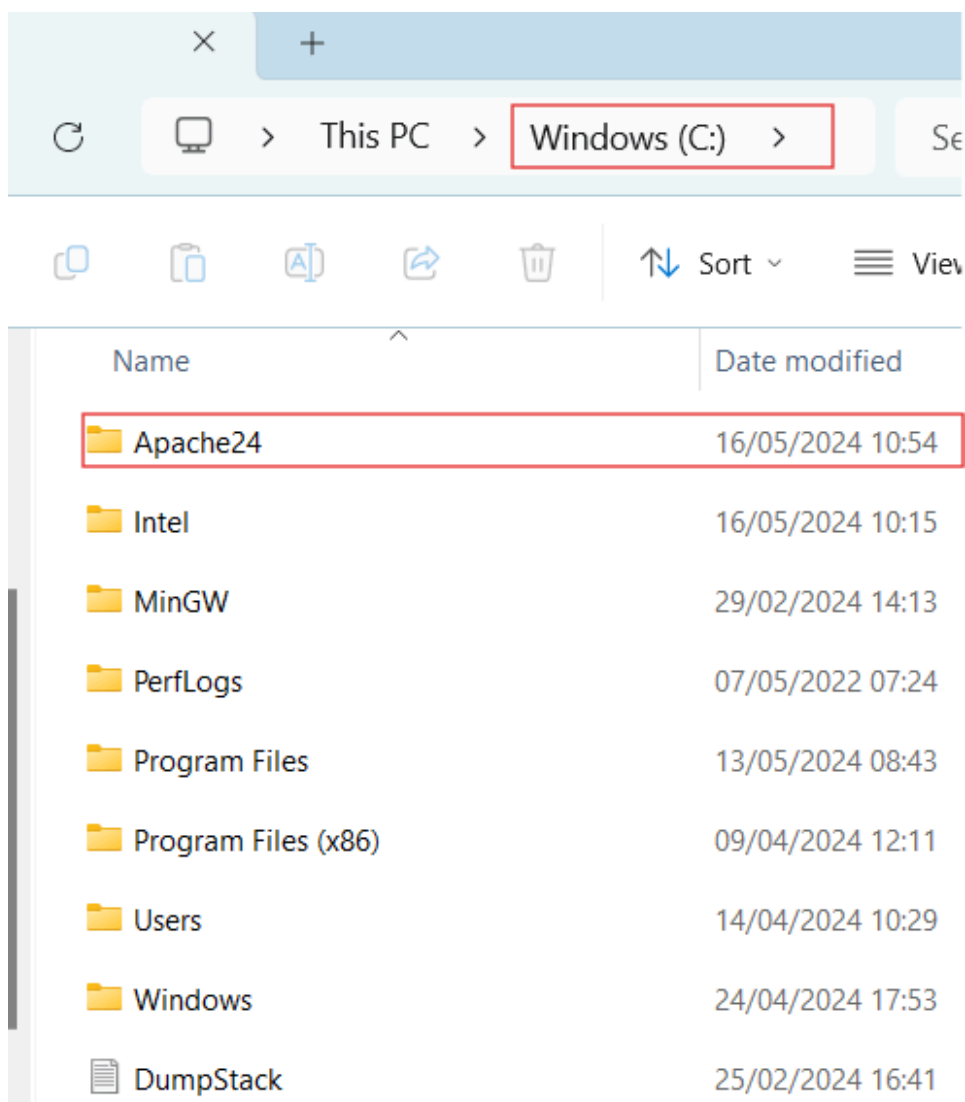
**Apache 2.4.59-240404 Win32**

04 Apr '24 11.431k

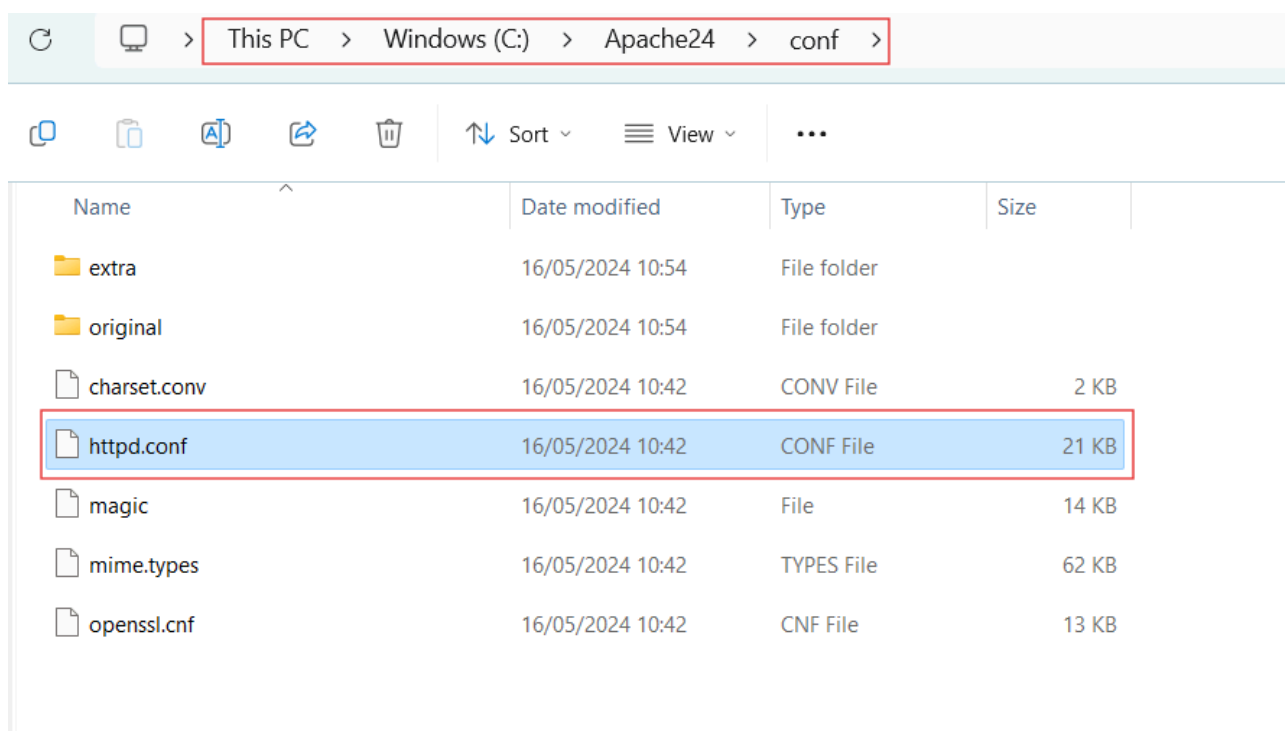
**Keep Server Online**

If you find the downloads useful, please express your satisfaction with a donation.  
[www.apachelounge.com](http://www.apachelounge.com)

2. Unzip the downloaded ZIP file to extract its contents. Within the unzipped folder, you will find another folder named **Apache24**. Copy this folder to the root directory of your C drive, so it resides at **C:\Apache24**.



3. Within the **Apache24** folder, navigate to a subfolder titled "**conf**". Here, locate a file named "**httpd.conf**".

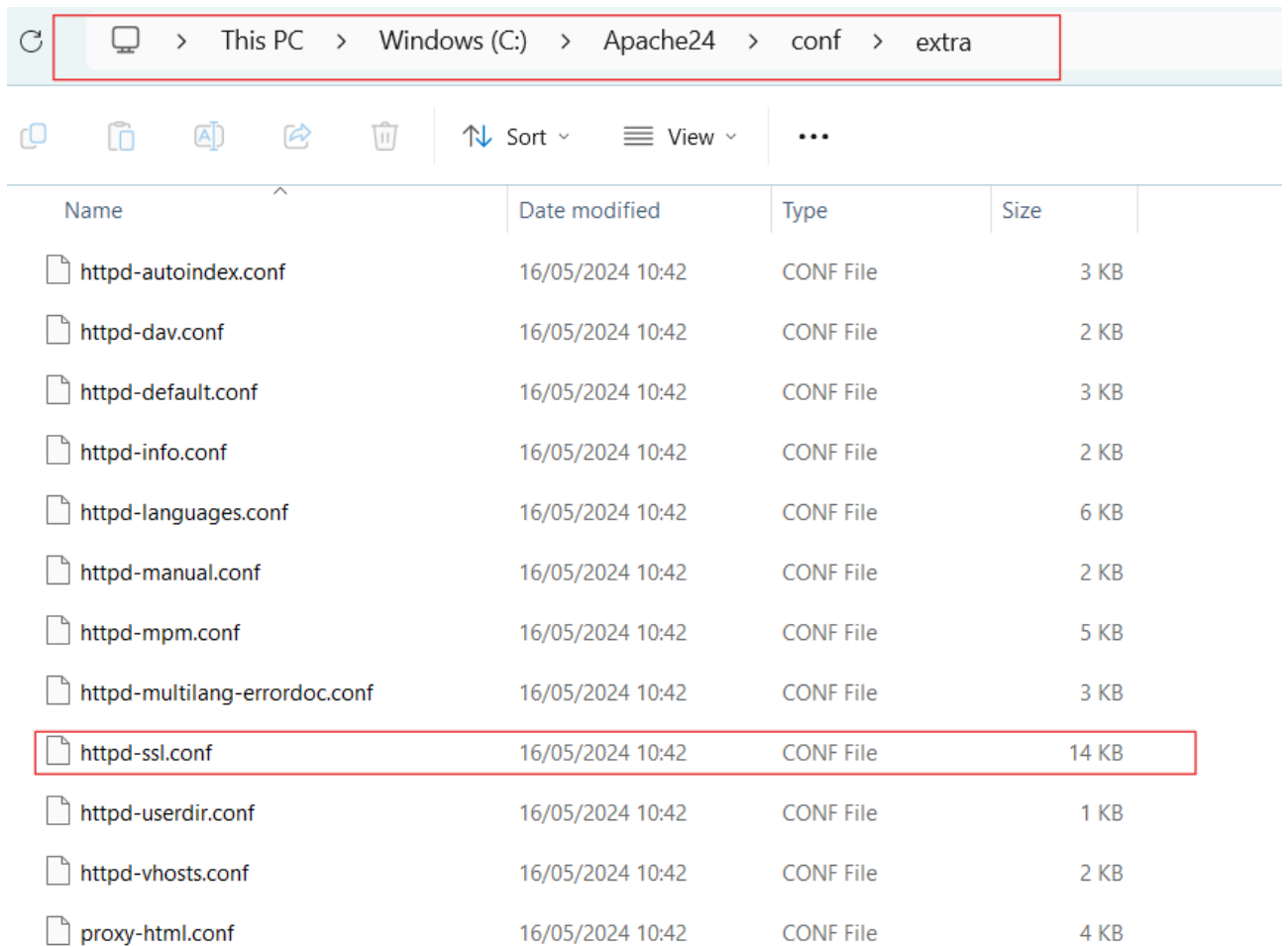


Open this file using a text editor such as Notepad or Visual Studio Code.

Once opened, locate the following settings and make the specified changes (if needed):

- Set "SRVROOT" to **C:/Apache24**.
- Set "ServerName" to **localhost:80**. This line is likely commented out. To activate it, simply remove the hash symbol (#) at the beginning to uncomment it.
- Set "DocumentRoot" to **C:\Apache\htdocs**. This line is likely commented out. To activate it, simply remove the hash symbol (#) at the beginning to uncomment it.

4. Navigate to the **C:/Apache24/conf/extra/** directory and locate the file named **"httpd-ssl.conf"**. Open this file using a text editor.



The screenshot shows a Windows File Explorer window with the address bar displaying the path: This PC > Windows (C:) > Apache24 > conf > extra. The file list below shows several .conf files. The file **httpd-ssl.conf** is highlighted with a red box.

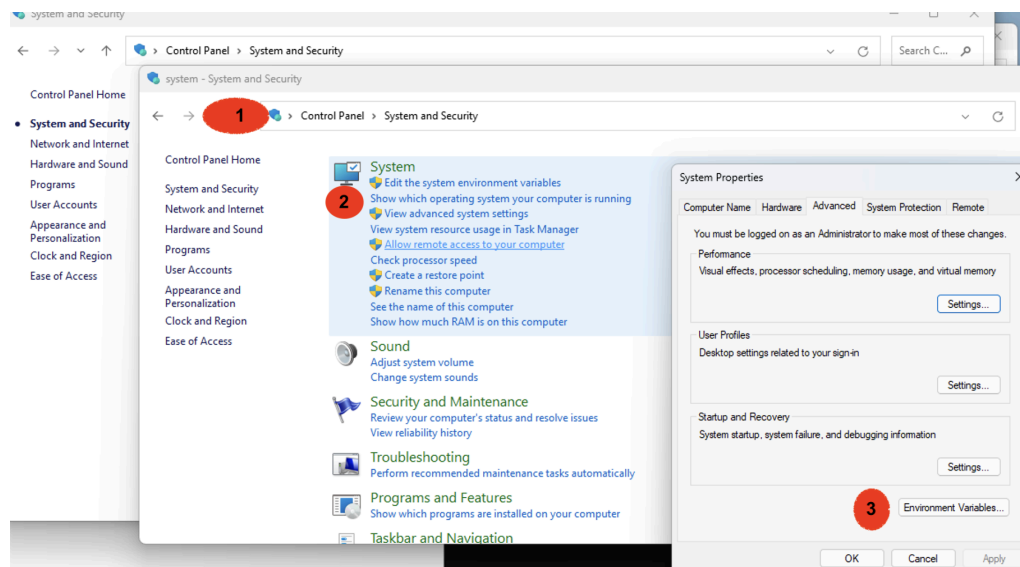
| Name                          | Date modified    | Type      | Size  |
|-------------------------------|------------------|-----------|-------|
| httpd-autoindex.conf          | 16/05/2024 10:42 | CONF File | 3 KB  |
| httpd-dav.conf                | 16/05/2024 10:42 | CONF File | 2 KB  |
| httpd-default.conf            | 16/05/2024 10:42 | CONF File | 3 KB  |
| httpd-info.conf               | 16/05/2024 10:42 | CONF File | 2 KB  |
| httpd-languages.conf          | 16/05/2024 10:42 | CONF File | 6 KB  |
| httpd-manual.conf             | 16/05/2024 10:42 | CONF File | 2 KB  |
| httpd-mpm.conf                | 16/05/2024 10:42 | CONF File | 5 KB  |
| httpd-multilang-errordoc.conf | 16/05/2024 10:42 | CONF File | 3 KB  |
| <b>httpd-ssl.conf</b>         | 16/05/2024 10:42 | CONF File | 14 KB |
| httpd-userdir.conf            | 16/05/2024 10:42 | CONF File | 1 KB  |
| httpd-vhosts.conf             | 16/05/2024 10:42 | CONF File | 2 KB  |
| proxy-html.conf               | 16/05/2024 10:42 | CONF File | 4 KB  |

Once opened, find the line containing the **SSLSessionCache directive** and comment it out: `SSLSessionCache "shmcb:${SRVR00T}/logs/ssl_scache(512000)".`

It is only for performance improvements, which are not necessary for this task.

5. Augment the Windows PATH environment variable with the path C:/Apache24/bin in the following way:

- You can find and edit the PATH variable by opening Windows' Control Panel, then navigating to "System", then "Edit the system environment variables" (marked (1) in the screenshot below).
- Click on "Environment Variables" (marked (2) in the screenshot below).
- Select "Path".
- Select "Edit" (marked (3) in the screenshot below) and then include the path.



6. Start the Apache server following these steps:

- Navigate to the bin directory using this command:
  - `cd C:/Apache24/bin/`
- Install the Apache 2.4 service with the following command:
  - `httpd -k install`
- Then start the Apache Server using this command:
  - `httpd.exe -k start`
- To stop the Apache Server use this command:
  - `httpd.exe -k stop`

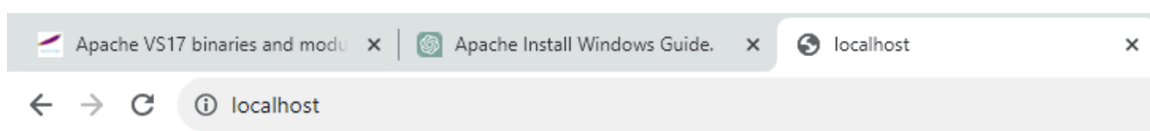


## Take note

If you encounter difficulties at this stage, review the error messages generated by the command. These messages usually pertain to directory structure errors. For example, ensure that the PATH command has been updated with the correct Apache directory structure by executing the following command in the Windows Terminal: `echo %PATH%`. You should receive output similar to the following:

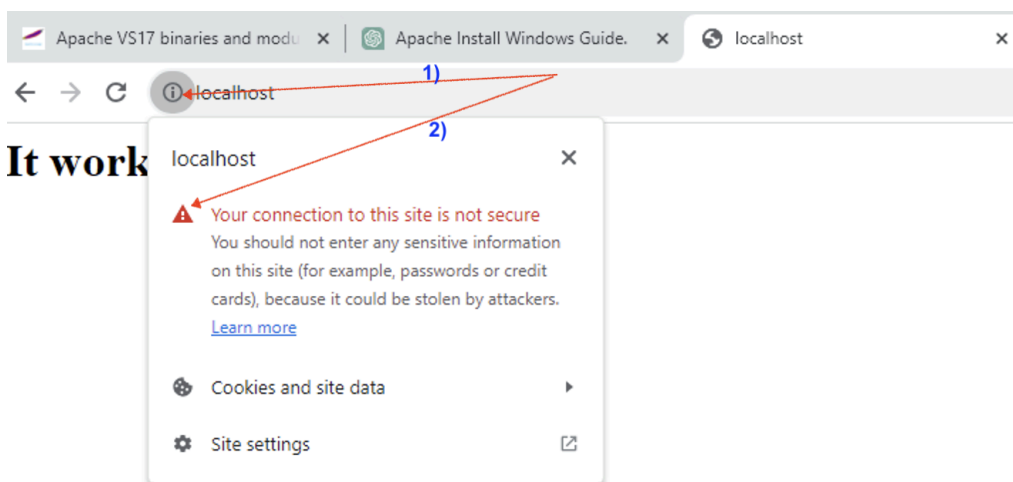
```
C:\Users\vcorbett>echo %PATH%
C:\Program Files\Parallels\Parallels Tools\Applications;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;"C:\Users\vcorbett\AppData\Local\Microsoft\WindowsApps;C:\Apache\bin";
```

7. Test the Apache web server by opening a web browser and navigating to **http://localhost**. You should encounter a web page resembling the following:



## It works!

Afterwards, click on the “i” icon located to the left of “localhost”. You should encounter a display similar to the following, indicating that the connection is not secure:



This occurs because you are utilising the HTTP protocol, which lacks the encryption, authentication, data integrity, and secure communication channels provided by the SSL protocol (HTTPS). Consequently, all your conversations can be eavesdropped on and tampered with.

## Create a certificate authority

We will first create a local certificate authority (CA) to prepare for the hardening of the Apache web server. A CA is a commercial, third-party, trusted authority, such as [GoDaddy](#) and [Verisign](#). CAs issue digital certificates, which are used to secure communication between two parties. You typically purchase the services of a CA to obtain such a certificate, ensuring secure communication, for example, between the Apache web server and its clients (i.e., web browsers).

We will establish a CA by creating a “self-signed” certificate. This certificate acts as a template for services like the Apache web server to generate their own certificate. Thus, Apache's web server certificate, signed by the CA, verifies the server's identity to web browsers requesting its service(s).

It is important to note that a self-signed CA certificate should only be used in lab environments. For production environments, it is necessary to use the services of a commercial CA to obtain a digital certificate.



# OpenSSL toolkit

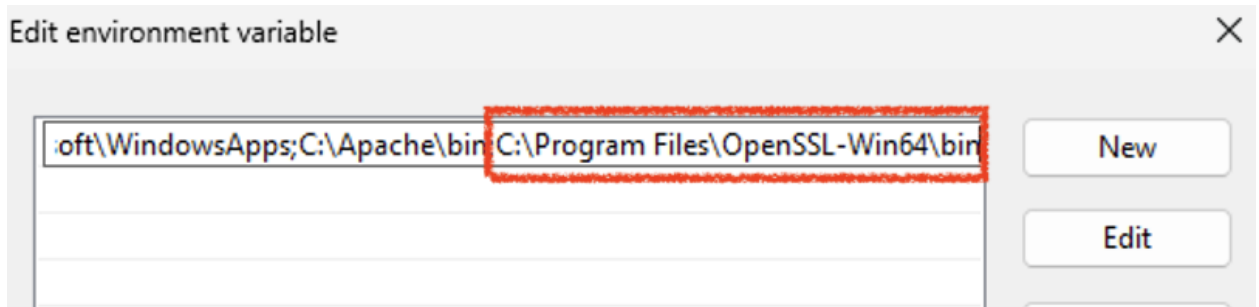
To “harden” the Apache web server, you will need to download the OpenSSL toolkit using the following instructions:

1. Visit the [OpenSSL download page](#).
2. Choose the appropriate version of OpenSSL for your Windows system (32-bit or 64-bit) and download the installer.

|   |                 |   |
|---|-----------------|---|
| Win64 OpenSSL v3.3.0<br><a href="#">EXE</a>   <a href="#">MSI</a> | 215MB Installer | Installs Win64 OpenSSL v3.3.0 (Recommended for software developers by the creators of <a href="#">OpenSSL</a> ). Only installs on 64-bit versions of Windows and targets Intel x64 chipsets. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation. |
|---|-----------------|---|

3. Double-click on the downloaded installer to start the installation process.
4. Follow the on-screen instructions provided by the installer. Typically, you'll need to agree to the license agreement and choose the installation directory (**C:/ProgramFiles/OpenSSL-Win64/**). Simply follow the installation prompts until the installation is complete.

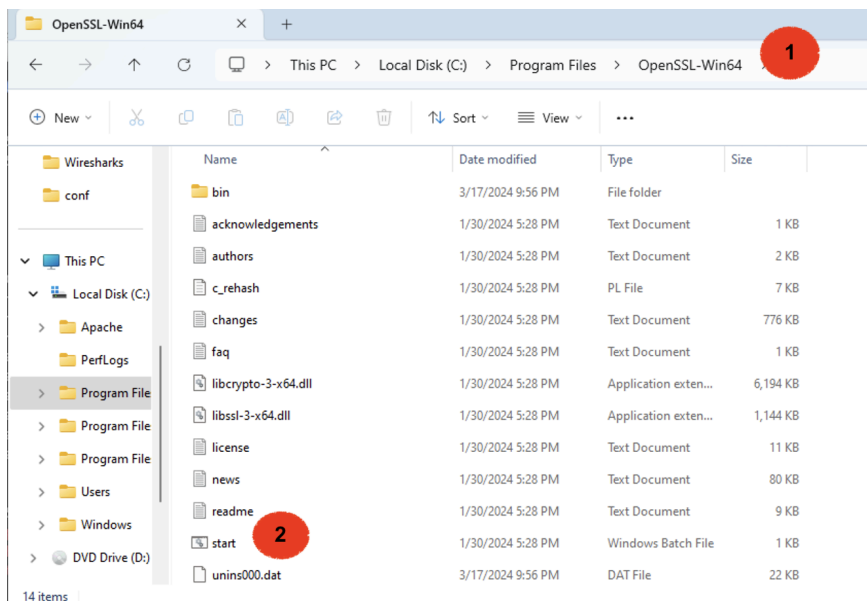
- Augment your Windows PATH environment variable with the OpenSSL **C:/ProgramFiles/OpenSSL-Win64/**, as shown here. **See above for instructions on navigating to the Windows environment variables editor.**



- Verify the installation by opening a command prompt and typing `openssl version`. You should see the version information for OpenSSL if the installation was successful.

## Certificate authority

- In the Windows File Explorer, navigate to the newly created SSL directory **C:\Program Files\OpenSSL-Win64\**, marked (1) in the screenshot below, and double-click on `start.bat`, marked (2) in the screenshot below. This will lead you to the command prompt.



- Navigate to the new directory **C:/Apache24/conf** via the following command: `cd C:/Apache24/conf`.

```
C:\Apache\conf>cd C:/Apache/conf
```

3. Create the file **/etc/apache2/san\_config** and add the code below. This file will be used in the subsequent `openssl` commands to create a CA certificate with a Subject Alternate Name (SAN) of "localhost". Please review the command parameters below for more information on these and all other characteristics of the digital certificate you will create.

```
[req]

default_bits = 2048

prompt = no

default_md = sha256

distinguished_name = dn

req_extensions = req_ext

[dn]

CN = localhost

[req_ext]

subjectAltName = @alt_names

basicConstraints = critical, CA:true

[alt_names]

DNS.1 = localhost
```

#### **Explanation of parameters for [req] section:**

- `default_bits = 2048` specifies the default key size in bits for the generated RSA private key. In this case, it's set to 2048 bits.
- `prompt = no` disables prompting for information during the certificate signing request (CSR) generation process.
- `default_md = sha256` sets the default message digest algorithm to SHA-256 for generating the CSR.
- `distinguished_name = dn` (see [dn] section below)
- `req_extensions = req_ext` specifies that the extensions to be added to the certificate request are defined in the `req_ext` section. It links the extensions defined in [req\_ext] to the certificate request.

### Explanation of parameters for [dn] section:

- **CN = localhost** specifies the common name associated with the certificate. In this case, it means that the certificate is intended for use with the hostname "localhost". The CA will use this information to issue a certificate with the specified common name.

### Explanation of parameters for [req\_ext] section:

- **subjectAltName = @alt\_names** specifies that the SAN extension should be included in the certificate request and takes values from the **alt\_names**.
- **basicConstraints = critical, CA:true** defines the basic constraints for the certificate. In this case, it indicates that the certificate to be issued is a CA certificate.

### Explanation of parameters for [alt\_names] section:

- **DNS.1 = localhost** specifies an alternative DNS name (SAN) for the certificate. In this case, it's set to "localhost", allowing the certificate to be valid for this name in addition to the CN specified in the [dn] section. Note that specifying the SAN is required for later versions of the Chrome and Chromium browsers.

4. The following command will generate a new private key (**server.key**), and certificate signing request (**server.csr**), explained below as parameters. "localhost" is defined as the common name (CN). The CN is a certificate field that identifies the entity for which the certificate is created. This will typically be your organisation's website domain name. As we are only creating a self-signed certificate for test purposes, we will define the CN as "localhost".

**Command:** `openssl req -newkey rsa:2048 -nodes -keyout server.key -out server.csr -subj "/CN=localhost" -config san_config`

Here is an explanation of each parameter:

- **-newkey rsa:2048** specifies that a new private key and CSR will be generated. The **rsa:2048** option indicates that an RSA key with a length of 2048 bits will be used. This part of the command combines the key generation and CSR creation into a single step.
- **-keyout server.key** specifies the filename (**server.key**) where the generated private key will be saved.
- **-out server.csr** specifies the filename (**server.csr**) where the generated CSR will be saved. The CSR contains information about the entity requesting the certificate, such as the CN and organisation.

- Sample output:

```
C:\Apache\conf>openssl req -newkey rsa:2048 -nodes -keyout server.key -out server.csr -subj "/CN=localhost"
```

Issue the following command to confirm that the CSR file has been created specifying a CA certificate with SAN value of "localhost". Here is the command and corresponding sample output:

```
"c:/program files/openssl-win64/bin/openssl" req -noout -text -in  
C:\Apache24\conf\server.csr
```

```
C:\Apache24\conf\etc\apache2">c:\program files\openssl-win64/bin/openssl" req -n
Certificate Request:
Data:
  Version: 1 (0x00)
  Subject: C=localhost
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    Public-Key: (2048 bit)
    Modulus:
      00:a4:b6:72:b4:df:5b:dd:5c:91:ef:d4:e0:44:be:
      5c:53:24:cb:2e:a3:48:6c:dd:ff:4b:6c:aa:20:8f:
      1d:1b:34:25:81:25:aa:08:50:3e:7f:75:be:e4:08:
      9d:a5:53:1a:c8:f5:64:cb:79:62:18:f6:af:03:be:
      2e:f7:4c:83:1e:0e:8e:14:41:4c:9c:ed:d4:b1:2e:
      3e:0f:f9:f1:29:f4:c2:73:59:46:b8:db:27:19:07:
      a5:77:3c:40:f5:1b:d2:36:a8:47:eb:5f:97:04:0c:
      95:26:8f:d0:4d:a4:d5:bb:34:ca:e1:19:8b:56:00:
      e4:d3:dd:f6:8e:66:71:bc:6c:53:98:d7:3f:8f:a1:
      0d:ce:34:c4:4a:7e:01:66:70:67:ef:f2:ce:e7:1c:
      49:82:bd:11:20:32:a6:0c:aad:22:79:ce:d5:58:3c:
      fb:c8:78:ff:eb:0e:59:a6:7e:6a:d0:19:ca:65:e9:
      6a:5f:64:85:d6:28:6c:17:94:6b:35:49:7f:cf:41:
      ce:be:d2:6f:69:6a:d8:bd:e9:cc:8b:51:61:ca:15:
      51:bb:e2:8d:48:2f:c7:72:00:b6:8a:6a:e8:5b:53:
      7f:92:36:f5:e5:91:9c:85:d0:75:c5:08:b0:1d:19:
      ca:24:b0:7d:83:22:91:45:6f:b4:0c:2e:5e:e4:84:
      2e:03
    Exponent: 65537 (0x10001)
  Attributes:
    X509v3 Subject Alternative Name:
      DNS=localhost
      X509v3 Basic Constraints: critical
      CA:TRUE
  Signature Algorithm: sha256WithRSAEncryption
  Signature Value:
    3d:36:78:a8:ee:9c:6d:e3:79:94:8c:59:0a:0e:1f:92:ad:2b:
    90:49:dc:b7:777:0b:7f:89:51:cb:28:62:47:3a:db:ae:7a:10:
    05:3b:57:2e:85:50:2c:ab:19:0a:a5:77:6a:cf:a7:50:4c:d5:
    00:79:cb:a7:7a:01:e4:78:0e:bb:ba:16:ff:85:20:42:53:b7:
    70:3f:bf:98:49:52:8c:30:bf:35:a6:84:c2:0e:0e:22:a7:41:
    cf:40:4e:8b:04:06:d1:12:4f:d5:4a:80:2f:86:da:72:39:4f:
    98:48:06:ab:96:25:66:d2:7b:22:d2:e0:a3:bfc:ca:7e:7d:1f:
    5a:36:5a:0b:b6:40:31:b8:59:f6:d6:2d:8a:7b:f5:de:fef:d4:
    f2:26:0c:30:1b:a2:2e:80:61:96:25:3f:64:07:f7:d5:b0:6a:
    5a:05:1c:2f:ff:9d:60:f6:7d:1f:66:42:23:17:1e:05:58:6d:
    d3:02:6d:b8:a8:0e:84:32:c1:c4:a4:07:24:ad:72:2d:4c:
```

5. The following command is used to create a digital certificate. We will use this certificate to enable the Apache web server to provide secure connections.

**Command:** `openssl x509 -req -in server.csr -signkey server.key -out server.crt -days 365 -sha256 -extensions req_ext -extfile san_config`

Here is an explanation of each parameter:

- `-req` specifies that the input file (**server.csr**) is a CSR rather than a certificate.
- `-days 365` specifies the validity period of the generated certificate in days. In this case, the certificate will be valid for 365 days from the time it is generated.
- `-in server.csr` specifies the input file containing the CSR (**server.csr**). The CSR contains the information required to generate the certificate, including the public key and details about the entity requesting the certificate.
- `-signkey server.key` specifies the private key (**server.key**) that will be used to sign the certificate. The private key is necessary for generating a self-signed certificate.
- `-out server.crt` specifies the output file where the generated certificate will be saved. In this case, the certificate will be saved as **server.crt**.

Command and sample output should be:

```
C:\Apache\conf>openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
Certificate request self-signature ok
subject=CN=localhost
```

Issue the following command to confirm that the digital certificate has been created correctly, specifying a CA certificate with SAN value of "localhost". Here is the command and corresponding output snippet:

```
"c:/program files/openssl-win64/bin/openssl" x509 -in server.crt  
-text -noout
```

```
C:\Apache24\conf\etc\apache2>"c:/program files/openssl-win64/bin/openssl" x509 -in server.crt -text -noout  
Certificate:  
  Data:  
    Version: 3 (0x2)  
    Serial Number:  
      7a:02:2e:70:65:a7:f3:38:af:1d:59:86:f8:cd:6f:9a:43:3d:ee:2e  
    Signature Algorithm: sha256WithRSAEncryption  
    Issuer: CN=localhost  
    Validity  
      Not Before: May 16 15:06:13 2024 GMT  
      Not After : May 16 15:06:13 2025 GMT  
    Subject: CN=localhost  
    Subject Public Key Info:  
      Public Key Algorithm: rsaEncryption  
      Public-Key: (2048 bit)  
      Modulus:  
        00:a4:b6:72:b4:df:5b:dd:5c:91:ef:d4:e0:44:be:  
        5c:53:24:cb:2e:a3:48:6c:dd:ff:4b:6c:aa:20:8f:  
        1d:1b:34:25:81:25:aa:08:50:3e:7f:75:be:e4:08:  
        94:da:53:1a:c8:f5:64:cb:79:62:18:f6:af:03:2e:  
        2e:f7:4c:83:1e:0e:8e:14:41:4c:9c:ed:da:1b:be:  
        3e:0f:f9:f1:29:f4:cd:73:59:46:b8:db:27:19:07:  
        a5:77:3c:40:f5:1b:cd:36:a8:47:eb:5f:97:04:9c:  
        95:26:8f:d0:4d:a4:d5:bb:34:ca:e1:19:8b:56:00:  
        e4:d3:dd:f6:8e:66:71:bc:6c:53:98:fd:3f:8f:a1:  
        0d:ce:34:c4:4a:7e:01:66:70:67:e9:f2:ec:e7:1c:  
        49:82:bd:11:20:32:a6:0c:ad:22:79:ce:d5:58:3c:  
        fb:c6:78:ff:eb:e0:59:a6:7e:6a:d0:19:ca:65:e9:  
        6a:5f:64:85:d6:28:6c:17:94:6b:35:49:7f:cf:41:  
        ce:be:d2:6f:69:6a:d8:bd:e9:cc:8b:51:61:ca:15:  
        51:bb:e2:8d:48:2f:c7:72:00:b6:8a:6a:e8:5b:53:  
        7f:92:36:f5:e5:91:c9:85:0d:75:c5:08:b0:1d:19:  
        ca:24:b0:7d:83:22:70:45:6f:bc:0a:ce:5e:e4:84:  
        2e:03  
      Exponent: 65537 (0x10001)  
    X509v3 extensions:  
      X509v3 Subject Alternative Name:  
        DNS:localhost  
      X509v3 Basic Constraints: critical  
        CA:TRUE  
      X509v3 Subject Key Identifier:  
        10:A8:CB:69:0D:B2:C4:29:AF:16:F8:C8:43:18:8A:CE:68:A6:22:05  
    Signature Algorithm: sha256WithRSAEncryption  
    Signature Value:  
      6e:dd:5c:15:2b:9a:83:db:81:84:25:6e:75:dd:fe:8c:99:49:  
      16:ba:da:47:e8:a7:62:50:77:a5:57:c7:2c:72:83:2b:c4:32:  
      1e:0d:72:c8:fe:b6:30:92:26:70:ba:d2:46:fa:ab:4e:e9:aa:
```

## Secure the Apache web server with the SSL/TLS using the CA certificate

Now, let's discuss the process of hardening the Apache web server, which involves implementing the SSL/TLS protocol. This protocol relies on the digital certificate created earlier. When a web browser connects to the website via **https://**, it utilises the HTTPS protocol, and the Apache web server presents its digital certificate, issued by the

local CA, to the browser. After the browser validates the certificate's validity, the SSL protocol establishes an encrypted and secure communication channel between the client and the server, protecting all data from eavesdropping and tampering.

In summary, the SSL protocol will provide the following critical security functions:

- Integrity: Data in transit remains intact and unaltered.
- Confidentiality: Data in transit is encrypted and, as such, it cannot be eavesdropped.
- Authentication: Digital certificates will verify the identity of the servers, reducing the risk of man-in-the-middle attacks.

For more information on SSL, visit this [web page](#).

To configure the Apache web server, follow these steps:

1. In the **C:/Apache/conf/extra/httpd-ssl.conf** file, confirm that the two lines shown in the screenshot below are uncommented. When uncommented, the Apache service will use the certification files indicated to start its SSL service. You can open this file using Notepad or VS Code.

```
SSLCertificateFile "${SRVROOT}/conf/server.crt"
#SSLCertificateFile "${SRVROOT}/conf/server-dsa.crt"
#SSLCertificateFile "${SRVROOT}/conf/server-ecc.crt"

# Server Private Key:
# If the key is not combined with the certificate, use this
# directive to point at the key file. Keep in mind that if
# you've both a RSA and a DSA private key you can configure
# both in parallel (to also allow the use of DSA ciphers, etc.)|
# ECC keys, when in use, can also be configured in parallel
SSLCertificateKeyFile "${SRVROOT}/conf/server.key"
```

While still in **C:/Apache/conf/extra/httpd-ssl.conf**, add the `Listen 8443` command and ensure it's not commented out (i.e., no leading #), to instruct Apache to listen for SSL connections on port 8443.

```
#
# When we also provide SSL we have to listen to the
# standard HTTP port (see above) and to the following HTTPS port
#
Listen 8443
|
```

Also in **C:/Apache/conf/extra/httpd-ssl.conf**, modify the `ServerName` variable as follows:



```
ServerName localhost:8443
```

2. In **C:/Apache/conf/httpd.conf**, uncomment the following two statements to enable and load the SSL module respectively:

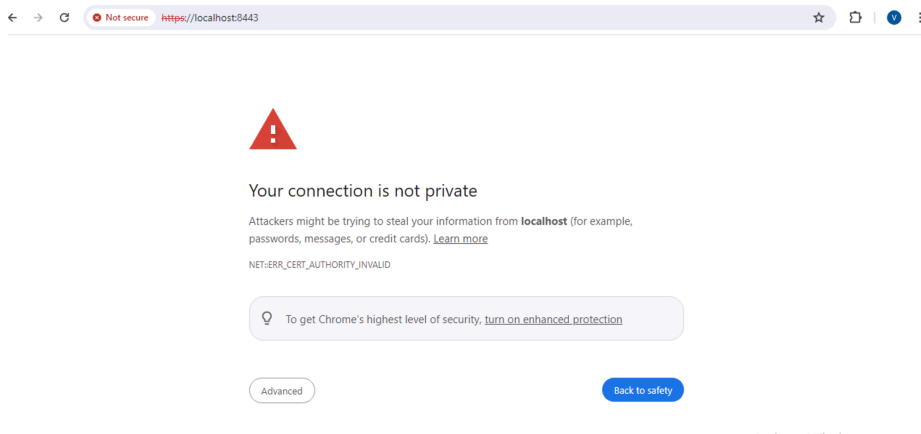
```
#Include conf/extra/httpd-ssl.conf
```

```
#LoadModule ssl_module modules/mod_ssl.so
```

```
LoadModule ssl_module modules/mod_ssl.so
```

## Test Apache's HTTPS

1. Install the [Chrome browser](#), if you don't have it installed.
2. If Apache is still running, stop it by issuing <ctrl c> or exiting the terminal.
3. Start the Apache server using: C:/Apache/bin/httpd.exe
4. Alternatively, you can start or stop Apache using the method referenced above.
5. In your browser, navigate to <https://localhost:443>, which should result in the following:



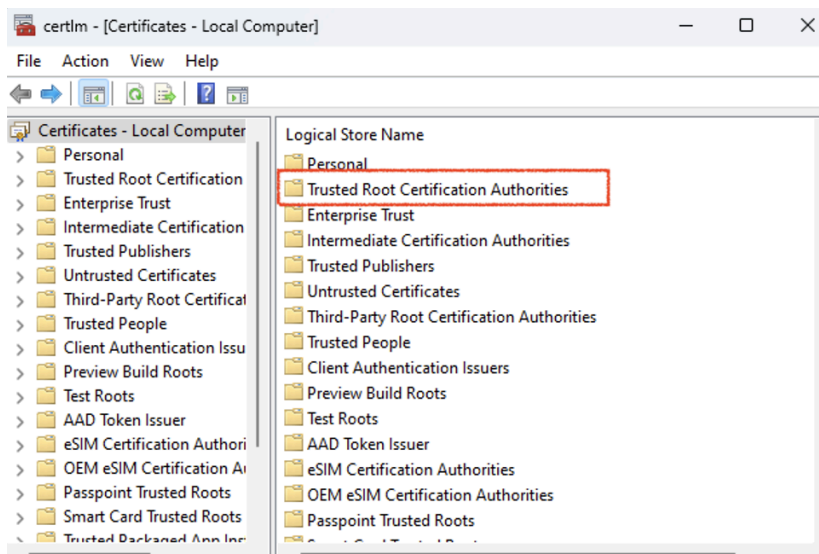


## Take note

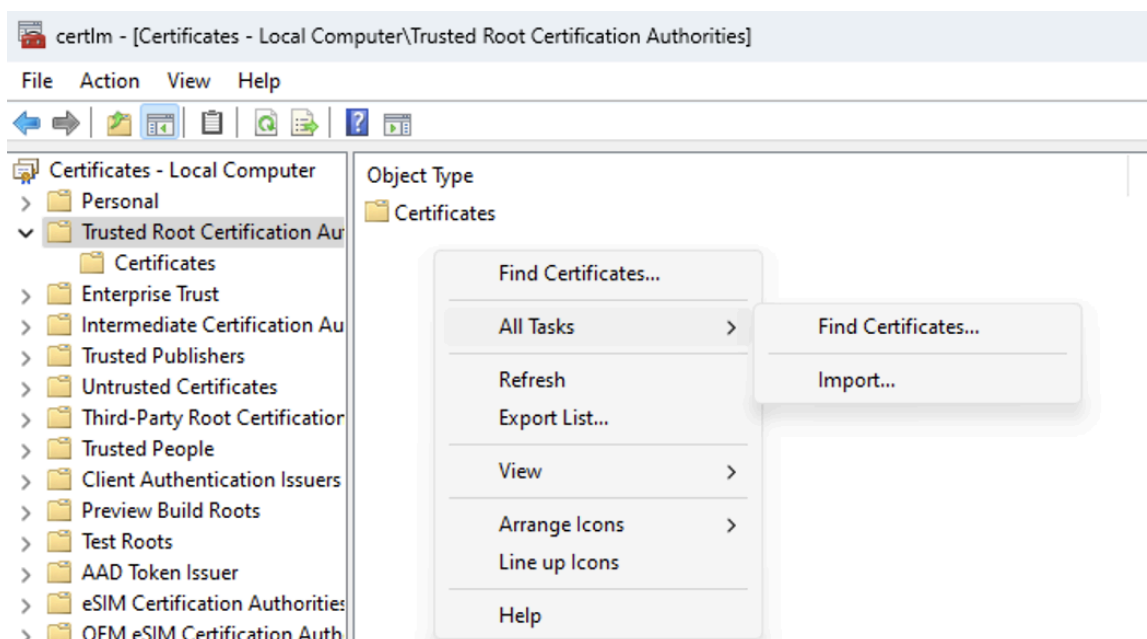
The warning is generated because your browser doesn't recognise the SSL certificate. It hasn't been issued by a trusted CA (e.g., Verisign). This behaviour is expected because we have created a self-signed certificate. Therefore, we can safely ignore the error.

---

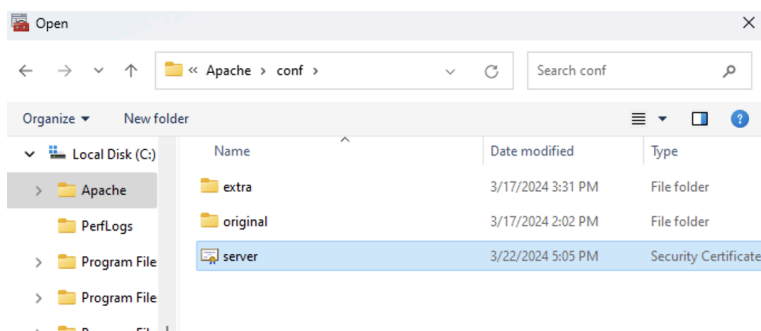
6. Add the self-signed certificate to the Microsoft certificate database. Navigate to `C:/Apache/conf` using this command: `cd /Apache24/conf`
7. Export the certificate from Apache via the following command:
8. `openssl x509 -outform der -in server.crt -out server.der`
9. In Windows, search for "Manage Certificates" and navigate to a folder named "Trusted Root Certification Authorities".



10. Right-click on an empty area in the right pane, select “All Tasks”, and then choose “Import” from the context menu.

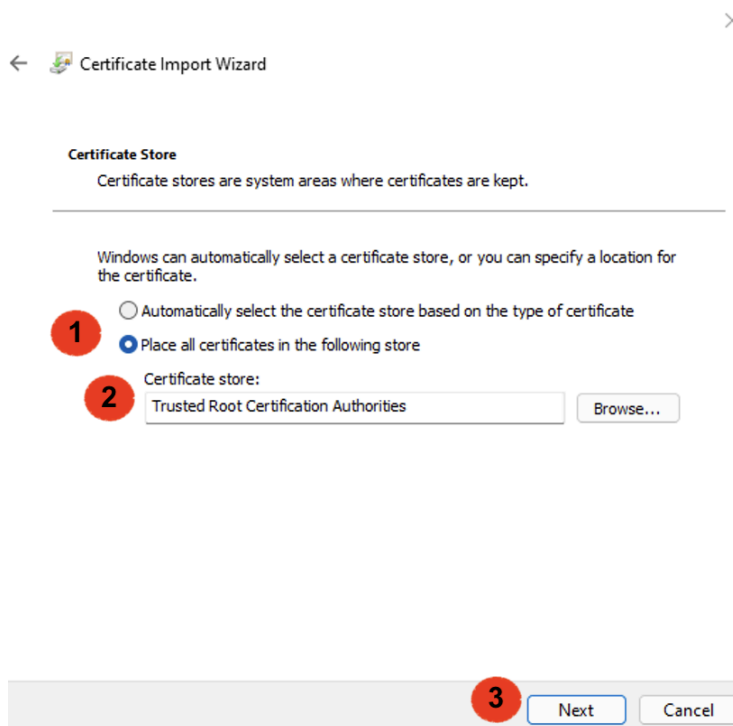


11. In the Certificate Import Wizard, click “Next”.
12. Continue to the import screen and select C:/Apache/conf/server.der.

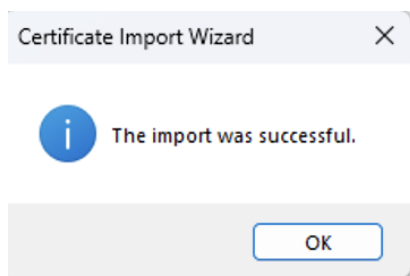


13. Click “Next”.

14. Choose the option “Place all certificates in the following store” (step (1)) and complete the window as shown below (step (2)). Then click “Next” (step (3)).

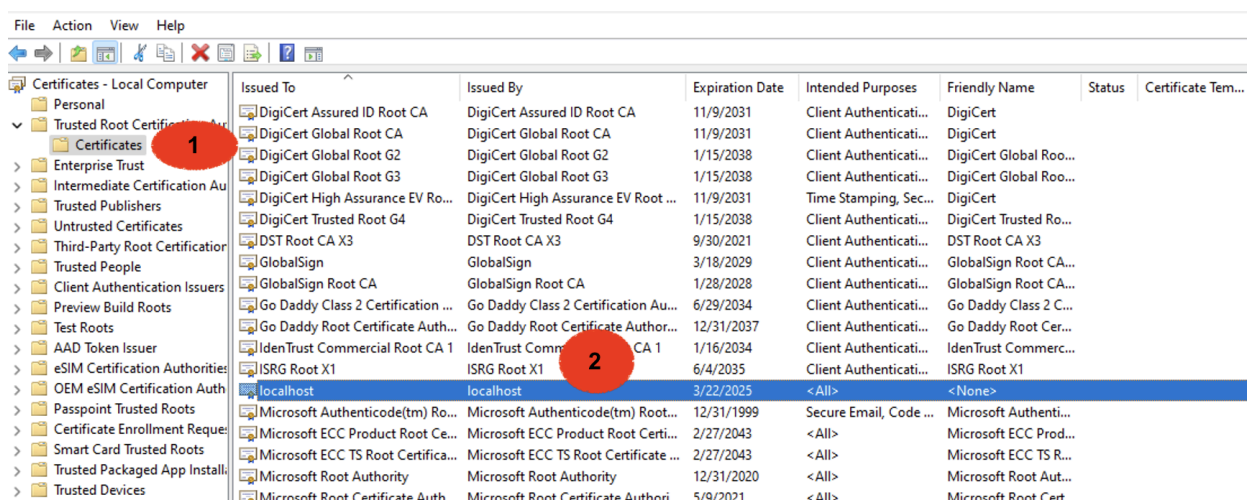


15. After confirming the import, you should receive a message indicating that the import was successful.

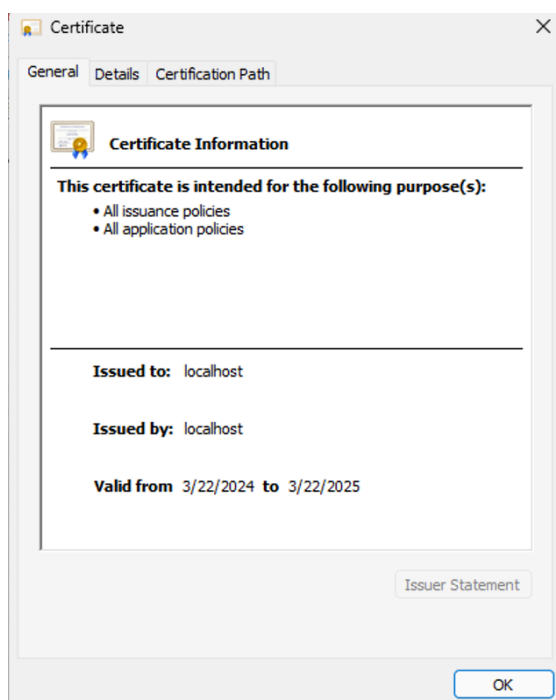


16. Now, start Apache again by entering: C:/Apache/bin/httpd.exe
17. Browse to <https://localhost:8443> using the Chrome browser again.

18. Confirm that your certificate is now listed in the “Trusted Root Certificate Authorities” folder in the certificate manager:



19. Now, let's take a look at the details by clicking on the certificate:



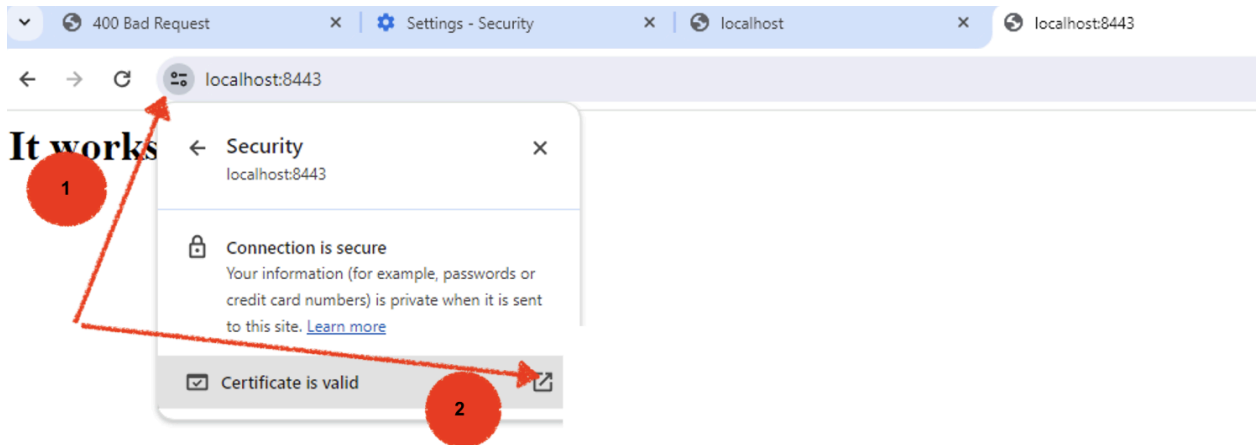
20. You'll notice that the “Issued to” and Issued by” are “localhost”.

21. In your Chrome browser, browse to <https://localhost:8443>. You should see a screen similar to the following:

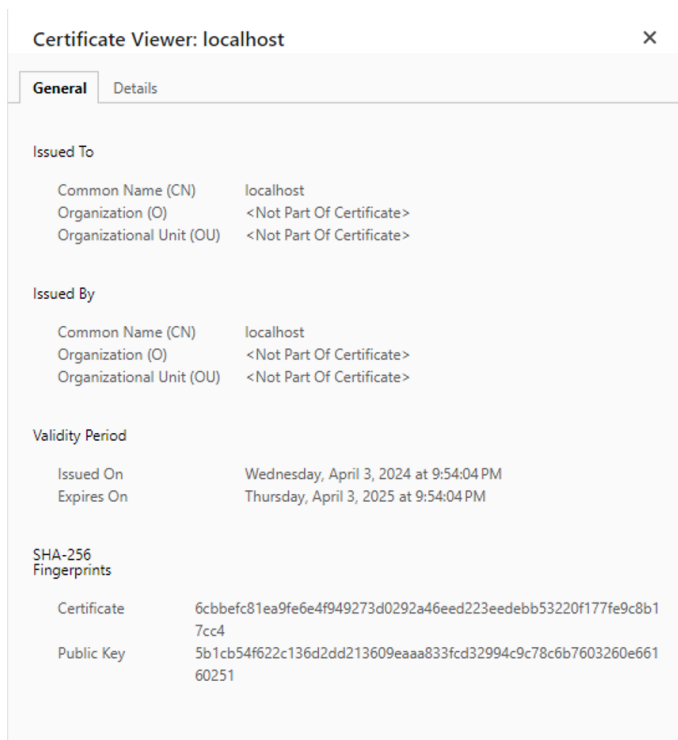


**It works!**

22. Open the certificate:



23. If you look at the certificate, you should see the same information as you saw above in the Microsoft CA.



24. In the "General" tab, the certificate issuer and validity period are displayed. You should verify whether it matches what you observed in the Microsoft CA.

# Summary

In this document, you have seen how using the SSL/TLS protocols, in conjunction with a digital certificate, ensures encrypted communication between a client (e.g., web browser) and server (e.g., web server), safeguarding against interception. The PKI certificate you created establishes trust by verifying the server's identity, thus mitigating risks associated with unauthorised access or man-in-the-middle attacks prevalent in HTTP-based connections.