

# SSL ? CA

SSL ???? Secure Sockets

Layer??  
???????????????????????????????? (????????????)????????????  
(????????????????????????????)??

????? - Certificate Authority (CA)????????????????SSL????????? Server ? Root ?????

????????????????????????? Verisign

??

CA ???? Linux ? Windows ????????? Linux ????Windows ?????? Windows ???Certificate  
Service?

- [Resources](#)
- [SSL ?????](#)
- [SSL/TLS Web Server](#)

# Resources

## Let's Encrypt

- [How To Secure Apache with Let's Encrypt on Ubuntu 16.04](#)
- [SSL For Free ?? SSL ??????? Let's Encrypt ???????](#)
- [How to Use Let's Encrypt to Install Free SSL Certificates on Your Linux VPS](#)
- [Apache with Let's Encrypt Certificates on CentOS 8](#)
- [How to manage Let's Encrypt SSL/TLS certificates with certbot](#)
- [certbot](#)
- [How to issue Let's Encrypt wildcard certificate with acme.sh and Cloudflare DNS](#)
- [How to forcefully renew Let's Encrypt certificate](#)
- [How to Check Let's Encrypt SSL Certificate Expiration Date](#)
- [Nginx] [Create a Web Server with NGINX and Secure it Using Certbot](#)

[Certbun](#) - Certbot alternative

- [More On Installing And Configuring Certbun For Use With Apache](#)

## Test SSL

????????? SSL ???????

- <https://www.tecmint.com/testssl-sh-test-tls-ssl-encryption-in-linux-commandline/>
- <https://github.com/drwetter/testssl.sh>
- <https://testssl.sh/>
- <https://www.ssllabs.com/ssltest/>
- <https://github.com/Matty9191/ssl-cert-check>

## Monitoring SSL

- <https://certificatemonitor.org>  
Source code: <https://github.com/RaymiiOrg/certificate-expiry-monitor>

- <https://alerts.httpscop.com>

# SSL Certificates

- [mkcert](#) - A simple tool for making locally-trusted development certificates. It requires no configuration.
  - [How to create locally trusted SSL Certificates on Linux and macOS with mkcert](#)
  - [How to Create Locally Trusted SSL Certificates with mkcert on Ubuntu 20.04](#)
- [SSL Certificates Cheat-Sheet](#)

## ACME (Automated Certificate Management Environment)

### [ACME](#)

????????????????????IETF??  
??ACME????????????????????  
????????????????????????????????????

- [Step Certificates](#)
- [Self-Host ACME Server](#)
- [Run your own private CA & ACME server using step-ca](#)

## Certimate - ????????

Certimate ?????????????????? SSL ???????????

??

- ???
- ?????????????????????90??

Certimate ?????????????????????????????????

- ???
- ???
- ?????????????????????? SSL ???

URLs:

- <https://docs.certimate.me/>
- GitHub: <https://github.com/usual2970/certimate>



# SSL ?????

## Check TLS/SSL certificate expiration date

from a website)

```
# NOTE: openssl requires at least 1.1.1
SITE_URL="www.cloudcoin.global"
SITE_SSL_PORT="443"

## note echo added ##
echo | openssl s_client -servername ${SITE_URL} -connect ${SITE_URL}:${SITE_SSL_PORT} \
| openssl x509 -noout -dates

# Alternatively

openssl s_client -connect ${SITE_URL}:${SITE_SSL_PORT} -servername ${SITE_URL} 2> /dev/null | openssl
x509 -noout -dates
```

### output

```
depth=2 C = IE, O = Baltimore, OU = CyberTrust, CN = Baltimore CyberTrust Root
verify return:1
depth=1 C = US, O = "Cloudflare, Inc.", CN = Cloudflare Inc ECC CA-3
verify return:1
depth=0 C = US, ST = CA, L = San Francisco, O = "Cloudflare, Inc.", CN = cloudcoin.global
verify return:1
DONE
notBefore=Jun  5 00:00:00 2020 GMT
notAfter=Jun  5 12:00:00 2021 GMT
```

- **s\_client** : The s\_client command implements a generic SSL/TLS client which connects to a remote host using SSL/TLS.
- **-servername \$DOM** : Set the TLS SNI (Server Name Indication) extension in the ClientHello message to the given value. *NOTE: openssl requires at lease 1.1.1*
- **-connect \$DOM:\$PORT** : This specifies the host (\$DOM) and optional port (\$PORT) to connect to.
- **x509** : Run certificate display and signing utility.
- **-noout** : Prevents output of the encoded version of the certificate.
- **-dates** : Prints out the start and expiry dates of a TLS or SSL certificate.

from a PEM encoded certificate file)

```
openssl x509 -enddate -noout -in /etc/nginx/ssl/www.cyberciti.biz.fullchain.cer
```

```
# find out if the TLS/SSL certificate expires within next 7 days (604800 seconds):  
openssl x509 -enddate -noout -in my.pem -checkend 604800
```

## output

```
notAfter=Dec 29 23:48:42 2020 GMT
```

## Shell script to alert sysadmin

```
#!/bin/bash  
# Purpose: Alert sysadmin/developer about the TLS/SSL cert expiry date in advance  
# Author: Vivek Gite {https://www.cyberciti.biz/} under GPL v2.x+  
# -----  
PEM="/etc/nginx/ssl/letsencrypt/cyberciti.biz/cyberciti.biz.fullchain.cer"  
  
# 7 days in seconds  
DAYS="604800"  
  
# Email settings  
_sub="$PEM will expire within $DAYS (7 days)."  
_from="system-account@your-dommain"  
_to="sysadmin@your-domain"  
_openssl="/usr/bin/openssl"  
$_openssl x509 -enddate -noout -in "$PEM" -checkend "$DAYS" | grep -q 'Certificate will expire'  
  
# Send email and push message to my mobile  
if [ $? -eq 0 ]  
then  
    echo "${_sub}"  
    mail -s "$_sub" -r "$_from" "$_to" <<< "Warning: The TLS/SSL certificate ($PEM) will expire soon on  
$HOSTNAME [$(date)]"  
    # See https://www.cyberciti.biz/mobile-devices/android/how-to-push-send-message-to-ios-and-android-  
from-linux-cli/ #  
    source ~/bin/cli_app.sh  
    push_to_mobile "$0" "$_sub. See $_to email for detailed log. -- $HOSTNAME " >/dev/null  
fi
```

## With Python

- [How to prevent TLS certificates from expiring with Python \(activestate.com\)](#)

## Get the Server certificate from the website

NOTE: openssl requires at least 1.1.1

```
echo -n | openssl s_client -showcerts -servername some.web.site -connect some.web.site:443 | sed -ne '/-BEGIN
CERTIFICATE-/,/END CERTIFICATE-/p' > some-web-site.pem
```

“ ??? openssl ????? Firefox ? Chrome ??? Server ???

????? \*.crt, \*.pem ?????

```
??????????? -----BEGIN CERTIFICATE----- ? -----END CERTIFICATE-----
????????????????????????????? openssl ???
```

```
openssl x509 -text -noout -in my_CA.pem
```

## Check the TLS version

```
nmap --script ssl-enum-ciphers -p 443 www.google.com
```

```
# -tls1 for TLSv1
```

```
# -tls1_1 for TLSv1.1
```

```
# -tls1_2 for TLSv1.2
```

```
openssl s_client -connect www.google.com:443 -tls1_3
```

## Generate a CSR (Certificate Signing Request)

```
sudo apt install openssl #[On Debian/Ubuntu]
```

```
sudo yum install openssl #[On CentOS/RHEL]
```

```
sudo dnf install openssl  #[On Fedora]
```

```
openssl req -new -newkey rsa:2048 -nodes -keyout example.com.key -out example.com.csr
```

## Show the Information of the certificate

```
openssl x509 -noout -text -in your.cer
```

```
# For the cert file that is generated by Windows
```

```
openssl x509 -noout -text -inform der -in win.cer
```

## More commands

```
# Generating a Private Key
```

```
openssl genpkey -algorithm RSA -out private.key
```

```
# Generating a Certificate Signing Request (CSR)
```

```
openssl req -new -key private.key -out csr.csr
```

```
# Generating a Self-Signed Certificate
```

```
openssl req -new -x509 -key private.key -out certificate.crt -days 365
```

```
# Encrypting Files
```

```
openssl enc -aes256 -in sensitive.txt -out sensitive.enc
```

```
# Decrypting Files
```

```
openssl enc -aes256 -d -in sensitive.enc -out sensitive.txt
```

```
# Converting Certificate Formats
```

```
openssl x509 -in certificate.crt -out certificate.pem
```

```
# Creating a Certificate Chain
```

```
cat intermediate.crt root.crt > chain.crt
```

```
# Signing a CSR with a CA
```

```
openssl x509 -req -in csr.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out certificate.crt -days 365
```

```
# Generating a Random Number
```

```
openssl rand -hex 16
```

```
# Checking CSR Details
```

```
openssl req -in csr.csr -noout -text
```

```
# Viewing Certificate Expiry
```

```
openssl x509 -enddate -noout -in certificate.crt
```



# Converting PFX to PEM

```
openssl pkcs12 -in certificate.pfx -clcerts -nokeys -out certificate.pem
```

# Creating a Password-Protected Private Key

```
openssl genpkey -algorithm RSA -aes256 -out private.key
```

# Testing SSLv2/v3 Protocol Support

```
openssl s_client -connect example[dot]com:443 -ssl2/-ssl3
```

# Extracting Public Key from Private Key

```
openssl rsa -in private.key -pubout -out public.key
```

# Encrypting and Decrypting Files with a Passphrase

# Encrypting File:

```
openssl enc -aes256 -salt -in sensitive.txt -out sensitive.enc
```

# Decrypting File:

```
openssl enc -aes256 -d -in sensitive.enc -out sensitive_decrypted.txt
```

# SSL/TLS Web Server

## Generate Certificates

Method 1: ??? CA ???

```
mkdir /etc/apache2/certs
cd /etc/apache2/certs
openssl genrsa -out myhomepbx.key 2048
openssl req -new -key myhomepbx.key -out myhomepbx.csr
openssl x509 -req -days 3650 -in myhomepbx.csr -signkey myhomepbx.key -out myhomepbx.crt
```

Method 2: ?? CA ???

```
# generate CA
# organizationName = HomePBX
# commName = HomePBX CA
# Enter PEM pass phrase: set new password that is used to sign the certicicate.
cd /etc/ssl/homepbxCA
openssl req -new -x509 -extensions v3_ca -keyout ca.key -out ca.crt -days 3650

# prerequisites
# Edit the openssl.homepbx.cnf as required
cp /etc/ssl/openssl.conf ./openssl.homepbx.cnf
touch index.txt
echo '01' > serial
mkdir newcerts

# generate Server certificates
# organizationName = HomePBX (it must be the same as CA, otherwise it cannot be signed by the CA)
# commName = FQDN of website or *
# Enter PEM pass phrase: It's not required, enter to skip it if wanted.
openssl req -config openssl.homepbx.cnf -new -nodes -keyout server.key -out server.csr
openssl ca -config openssl.homepbx.cnf -days 3650 -in server.csr -out server.crt

# generate PKCS12 for client authentication
# NOTE: you can create PKCS12 file by either server certificate or CA certificate.
```

```
# Enter Export Password: set new password that is used for importing the PKCS12
openssl pkcs12 -export -clcerts -in ca.crt -inkey ca.key -out homepbx_2021y.p12

# Alternatively
openssl pkcs12 -export -clcerts -in server.crt -inkey server.key -out homepbx_2021y.p12
```

## openssl.homepbx.cnf

```
...
[ CA_default ]

dir          = .           # Where everything is kept <== Here
certs        = $dir/certs   # Where the issued certs are kept
crl_dir       = $dir/crl     # Where the issued crl are kept
database      = $dir/index.txt # database index file.
#unique_subject = no         # Set to 'no' to allow creation of
                             # several certs with same subject.
new_certs_dir = $dir/newcerts # default place for new certs.

certificate   = $dir/ca.crt  # The CA certificate <== Here
serial        = $dir/serial   # The current serial number
crlnumber     = $dir/crlnumber # the current crl number
                             # must be commented out to leave a V1 CRL
crl           = $dir/crl.pem  # The current CRL
private_key   = $dir/ca.key# The private key <== Here

x509_extensions = usr_cert    # The extensions to add to the cert
...
```