

# 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
```

---

Revision #20

Created 24 October 2020 02:56:01 by Admin

Updated 24 July 2023 20:30:33 by Admin