GEEKFLARE

21 OpenSSL Examples to Help You in Real-World

Create, Manage & Convert SSL Certificates with OpenSSL

One of the most popular commands in SSL to **create**, **convert**, **manage** the SSL Certificates is OpenSSL. There will be many situations where you have to deal with OpenSSL in various ways and here I have listed them for you as a handy cheat sheet.

In this article, I will talk about frequently used OpenSSL commands to help you in the real world.

Some of the abbreviations related to certificates.

- SSL – Secure Socket Layer
- CSR – Certificate Signing Request
- TLS – Transport Layer Security
- PEM – Privacy Enhanced Mail
- DER – Distinguished Encoding Rules
- SHA – Secure Hash Algorithm

- PKCS – Public-Key Cryptography Standards

1. Create new Private Key and Certificate Signing Request

```
openssl req -out geekflare.csr -newkey rsa:2048 -nodes -keyout geekflare.key
```

Above command will generate CSR and 2048-bit RSA key file. If you intend to use this certificate in Apache or Nginx then you need to send this CSR file to certificate issuer authority and they will give you signed certificate mostly in der or pem format which you need to configure in Apache or Nginx web server.

2. Create Self-Signed Certificate

```
openssl req -x509 -sha256 -nodes -newkey rsa:2048 -keyout gfselfsigned.key -out
 gfcert.pem
```

Above command will generate a self-signed certificate and key file with 2048-bit RSA. I have also included sha256 as it's considered most secure at the moment.

Tip: by default, it will generate self-signed certificate valid for only one month so you may consider defining –days parameter to extend the validity.

Ex: to have self-signed valid for 2 years.

```
openssl req -x509 -sha256 -nodes -days 730 -newkey rsa:2048 -keyout gfselfsigne
d.key -out gfcert.pem
```

3. Verify CSR file

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

Verification is important to ensure you are sending CSR to issuer authority with required details.

4. Create RSA Private Key

```
openssl genrsa -out private.key 2048
```

If you just need to generate RSA private key, you can use above command. I have included 2048 for stronger encryption.

5. Remove Passphrase from Key

```
openssl rsa -in certkey.key -out nopassphrase.key
```

If you are using passphrase in key file and using Apache then every time you start, you have to enter the password. If you are annoyed with entering password then you can use above openssl rsa -in geekflare.key -check to remove the passphrase key from existing key.

6. Verify Private Key

```
openssl rsa -in certkey.key –check
```

If you doubt on your key file, you can use above command to check.

7. Verify Certificate File

```
openssl x509 -in certfile.pem -text –noout
```

If you would like to validate certificate data like CN, OU, etc then you can use above command which will give you certificate details.

8. Verify the Certificate Signer Authority

```
openssl x509 -in certfile.pem -noout -issuer -issuer_hash
```

Certificate issuer authority signs every certificate and in case you need to check them, you can use above command.

9. Check Hash Value of A Certificate

```
openssl x509 -noout -hash -in bestflare.pem
```

10. Convert DER to PEM format

```
openssl x509 –inform der –in sslcert.der –out sslcert.pem
```

Usually, certificate authority will give you SSL cert in .der format and if you need to use them in apache or .pem format, you can use above command to convert them.

11. Convert PEM to DER format

```
openssl x509 –outform der –in sslcert.pem –out sslcert.der
```

In case you need to change .pem format to .der

12. Convert Certificate and Private Key to PKCS#12 format

```
openssl pkcs12 -export -out sslcert.pfx -inkey key.pem -in sslcert.pem
```

If you need to use a cert with the java application or with any other who accept only PKCS#12 format, you can use above command, which will generate single pfx containing certificate & key file.

Tip: you can also include chain certificate by passing –chain as below.

```
openssl pkcs12 -export -out sslcert.pfx -inkey key.pem -in sslcert.pem -chain c
acert.pem
```

13. Create CSR using existing private key

```
openssl req -out certificate.csr -key existing.key -new
```

If you don't want to create a new private key instead using existing one, you can with above command.

14. Check contents of PKCS12 format cert

```
openssl pkcs12 -info -nodes -in cert.p12
```

PKCS12 is binary format so you won't be able to view the content in notepad or another editor. So you got to use above command to view the contents of PKCS12 format file.

15. Convert PKCS12 format to PEM certificate

```
openssl pkcs12 -in cert.p12 -out cert.pem
```

If you wish to use existing pkcs12 format with Apache or just in pem format, this will be useful.

16. Test SSL certificate of particular URL

```
openssl s_client -connect yoururl.com:443 -showcerts
```

I use this quite often to validate the SSL certificate of particular URL from the server. This is very handy to validate the protocol, cipher, and cert details.

17. Find out OpenSSL version

```
openssl version
```

If you are responsible for ensuring OpenSSL is secure then probably one of the first things you got to do is to verify the version.

18. **Check PEM File Certificate Expiration Date**

```
openssl x509 -noout -in certificate.pem -dates
```

Useful if you are planning to put some kind of monitoring to check the validity. It will show you date in notBefore and notAfter syntax. notAfter is one you will have to verify to confirm if a certificate is expired or still valid.

Ex:

```
[root@Chandan opt]# openssl x509 -noout -in bestflare.pem -dates
notBefore=Jul 4 14:02:45 2015 GMT
notAfter=Aug 4 09:46:42 2015 GMT
[root@Chandan opt]#
```

19. Check Certificate Expiration Date of SSL URL

```
openssl s_client -connect secureurl.com:443 2>/dev/null | openssl x509 -noout -enddate
```

Another useful if you are planning to monitor SSL cert expiration date remotely or particular URL.

Ex:

```
[root@Chandan opt]# openssl s_client -connect google.com:443 2>/dev/null | open
ssl x509 -noout -enddate

notAfter=Dec 8 00:00:00 2015 GMT
```

20. Check if SSL V2 or V3 is accepted on URL

To check SSL V2

```
openssl s_client -connect secureurl.com:443 -ssl2
```

To Check SSL V3

```
openssl s_client -connect secureurl.com:443 -ssl3
```

To Check TLS 1.0

```
openssl s_client -connect secureurl.com:443 -tls1
```

To Check TLS 1.1

```
openssl s_client -connect secureurl.com:443 -tls1_1
```

To Check TLS 1.2

```
openssl s_client -connect secureurl.com:443 -tls1_2
```

If you are securing web server and need to validate if SSL V2/V3 is enabled or not, you can use above command. If enabled, you will get "**CONNECTED**" else "**handshake failure**"

21. Verify if particular cipher is accepted on URL

```
openssl s_client -cipher 'ECDHE-ECDSA-AES256-SHA' -connect secureurl:443
```

If you are working on security findings and pen test results show some of the weak ciphers is accepted then to validate, you can use above command. Off course, you will have to change the cipher and URL, which you want to test against.

If mentioned cipher is accepted then you will get "**CONNECTED**" else "**handshake failure**".

I hope above commands help you to know more about OpenSSL to manage SSL certificates for your website.