

## Activity V : Public Key Infrastructure

Created by : Krerk Piromsopa, Ph.D

### Overview

In this activity, you will learn the fundamentals of Public Key Infrastructure. We will need the following tools:

- A. OpenSSL. On Linux and Mac OS X, the OpenSSL is installed by default. For Windows, you may download it from <https://wiki.openssl.org/index.php/Binaries> .
- B. Python with PyOpenSSL and pem to do our exercise. If you python does not come with PyOpenSSL and pem, you may install it with pip  
\$ pip install pyopenssl  
\$ pip install pem
- C. You also need ca-certificates.crt from your OS (e.g. /etc/cacerts/ca-certificates.crt in Linux) or take it from the course web.

### Exercise

Issuing the following command. **openssl s\_client -connect twitter.com:443**

Once connected, you may try  
GET / HTTP/1.0  
[Enter twice]

(Note that the server may return HTTP 404. This is completely normal since we did not send a request for a valid resource.)

Repeat the same step again, now with

This command basically connects to port 443 (HTTPS) with the TLS/SSL. This is like a standard telnet command, but with openssl performing the encryption for you.

1. From the two given openssl commands, what is the difference?

Note: If your operating system does not show any error in the first command, try **openssl s\_client -connect twitter.com:443 -CApath /dev/null** . If the results are still the same, your system is not reliable. You may ignore this exercise. (Modern

versions of Mac OS X will always read CA from keychains. There is no intuitive way to turn it off.)

Same output

```
flap@flap-VirtualBox:~$ openssl s_client -connect twitter.com:443 -CAfile /etc/ssl/certs/ca-certificates.crt
CONNECTED(00000005)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert High Assurance EV Root CA
verify return:1

flap@flap-VirtualBox:~$ openssl s_client --connect twitter.com:443
CONNECTED(00000005)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert High Assurance EV Root CA
verify return:1
depth=1 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 Assurance Server CA
```

2. What does the error (verify error) in the first command mean? Please explain.

I get no error

3. Copy the server certificate (beginning with -----BEGIN CERTIFICATE----- and ending with -----END CERTIFICATE-----) and store it as twitter\_com.cert. Use the command **openssl x509 -in twitter\_com.cert -text** to show a text representation of the certificate content. Briefly explain what is stored in an X.509 certificate (i.e. data in each field).

```
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      07:be:ed:4d:99:6a:eb:0f:34:c0:a3:7e:ed:d6:01:fe
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2
High Assurance Server CA
    Validity
      Not Before: Feb  6 00:00:00 2020 GMT
      Not After : Feb  5 12:00:00 2021 GMT
    Subject: C = US, ST = California, L = San Francisco, O = "Twitter, Inc.", O
= sinl, CN = twitter.com
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (2048 bit)
      Modulus:
        00:d3:34:cf:aa:53:a1:95:dc:8f:85:70:72:92:79:
        3a:43:cf:ec:33:86:81:d8:5d:cb:6f:e3:1e:15:5d:
        06:ad:2f:89:d7:55:9e:28:99:19:8a:58:02:13:ab:
        65:99:a6:6a:44:c8:9d:bb:40:a1:88:83:bc:24:fa:
        cd:4f:b0:e8:2c:8b:63:c0:7a:7c:1c:e2:fa:5e:22:
        b2:67:90:bf:15:d5:0a:04:95:b8:43:6c:90:31:0e:
        ce:74:6f:7a:7f:f0:3c:21:58:04:2e:c2:c4:b1:e7:
        c2:57:7e:77:04:ed:67:bl:7d:0b:68:45:1d:e4:1a:
        9e:e1:81:07:2f:10:a9:7c:86:fd:47:ba:56:94:02:
        f0:fa:19:30:d4:fc:ad:26:d1:32:c5:a3:4b:ea:a8:
        0c:a9:18:0d:71:47:cc:9c:1d:80:17:79:00:82:c3:
        d8:eb:3f:f5:c8:6d:93:da:93:23:72:7b:2e:23:6c:
        59:9f:ae:26:8a:92:3c:d9:00:23:c5:cd:b4:6f:d9:
        75:b0:0a:a6:3c:98:0a:60:61:e7:44:94:d6:c8:23:
        4c:c4:2c:c7:2b:70:5b:37:1a:04:06:cd:5c:42:48:
        75:b5:6a:a8:3c:38:8a:66:6f:87:44:54:66:cb:23:58:
        4c:c4:2c:c7:2b:70:5b:37:1a:04:06:cd:5c:42:48:
        35:28:13:8a:08:71:a1:f5:4b:b9:fd:6a:ac:2a:58:
        b4:f7:03:a1:ac:b7:dc:c6:0e:99:07:2a:fd:04:1f:
        d8:0d
      Exponent: 65537 (0x10001)
    X509v3 extensions:
      X509v3 Authority Key Identifier:
        keyid:51:68:FF:90:AF:02:07:75:3C:CC:D9:65:64:62:A2:12:B8:59:72:3B

      X509v3 Subject Key Identifier:
        DC:DD:FA:BD:72:88:5B:0D:1F:B5:80:2C:64:26:58:74:F9:06:DB:AE
      X509v3 Subject Alternative Name:
        DNS:twitter.com, DNS:www.twitter.com
      X509v3 Key Usage: critical
        Digital Signature, Key Encipherment
      X509v3 Extended Key Usage:
        TLS Web Server Authentication, TLS Web Client Authentication
      X509v3 CRL Distribution Points:

        Full Name:
          URI:http://crl3.digicert.com/sha2-ha-server-g6.crl

        Full Name:
          URI:http://crl4.digicert.com/sha2-ha-server-g6.crl

      X509v3 Certificate Policies:
        Policy: 2.16.840.1.114412.1.1
          CPS: https://www.digicert.com/CPS
        Policy: 2.23.140.1.2.2

      Authority Information Access:
        OCSP - URI:http://ocsp.digicert.com
        CA Issuers - URI:http://cacerts.digicert.com/DigiCertSHA2HighAssura
```

Version number: certificates version

Serial number: unique number identify cert

Signature algorithm: Hash algorithm

Issuer information

Company file for issue information

Date range that the certificate valid

Pub key details

4. From the information in exercise 3, is there an intermediate certificate? If yes, what purpose does it serve?

Hint: Look for an issuer and download the intermediate certificate. You may use the command **openssl x509 -inform der -in intermediate.cert -text** to show the details of the intermediate certificate. (Note that the -inform der is for reading the DER file. The default file format for x509 is the PEM file.)

```
Root CA
    Issuer: C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert High Assurance EV
    Validity
        Not Before: Oct 22 12:00:00 2013 GMT
        Not After : Oct 22 12:00:00 2028 GMT
    Subject: C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 High Assurance Server CA
```

Yes the purpose are for providing an added level of security and compromise root PK

5. Is there an intermediate CA, i.e. is there more than one organization involved in the certification? Say why you think so.

Yes, in the picture above Common Name is difference

6. What is the role of ca-certificates.crt?

Store trusted certificate

7. Explore the ca-certificates.crt. How many certificates are in there? Give the command/method you have used to count.

127

```
flap@flap-VirtualBox:~$ grep -o -i BEGIN /etc/ssl/certs/ca-certificates.crt | wc -l
```

8. Extract a root certificate from ca-certificates.crt. Use the openssl command to explore the details. Do you see any Issuer information? Please compare it to the details of twitter's certificate and the details of the intermediate certificate.

There are issuer information but it's value is in gibberish compare to twitter

```
flap@flap-VirtualBox:~$ openssl x509 -in /etc/ssl/certs/ca-certificates.crt -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 6828503384748696800 (0x5ec3b7a6437fa4e0)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: CN = ACCVRAIZ1, OU = PKIACCV, O = ACCV, C = ES
```

9. If the intermediate certificate is not in a PEM format (text readable), use the command to convert a DER file (.cert .cer .der) to PEM file. **openssl x509 -inform der -in certificate.cer -out certificate.pem**.  
(You need the pem file for exercise 10.)

10. From the given python code,<sup>1</sup> implement the certificate validation.

```
from OpenSSL import crypto
import pem

def verify():
    with open('./target.cert', 'r') as cert_file:
        cert = cert_file.read()

    with open('./intermediate.cert', 'r') as int_cert_file:
        int_cert = int_cert_file.read()

    pems=pem.parse_file('./ca-certificates.cert');
    trusted_certs = []
    for mypem in pems:
        trusted_certs.append(str(mypem));

    trusted_certs.append(int_cert);

    verified = verify_chain_of_trust(cert, trusted_certs)

    if verified:
        print('Certificate verified')

def verify_chain_of_trust(cert_pem, trusted_cert_pems):

    certificate = crypto.load_certificate(crypto.FILETYPE_PEM, cert_pem)

    # Create and fill a X509Store with trusted certs
    store = crypto.X509Store()
    for trusted_cert_pem in trusted_cert_pems:
```

---

<sup>1</sup> Code taken from <http://www.yothenberg.com/validate-x509-certificate-in-python/> . It has been modified for this exercise.

```
        trusted_cert = crypto.load_certificate(crypto.FILETYPE_PEM,
trusted_cert_pem)
        store.add_cert(trusted_cert)

# Create a X509StoreContext with the cert and trusted certs
# and verify the the chain of trust
store_ctx = crypto.X509StoreContext(store, certificate)
# Returns None if certificate can be validated
result = store_ctx.verify_certificate()

if result is None:
    return True
else:
    return False
```

Use your program to verify the certificates of:

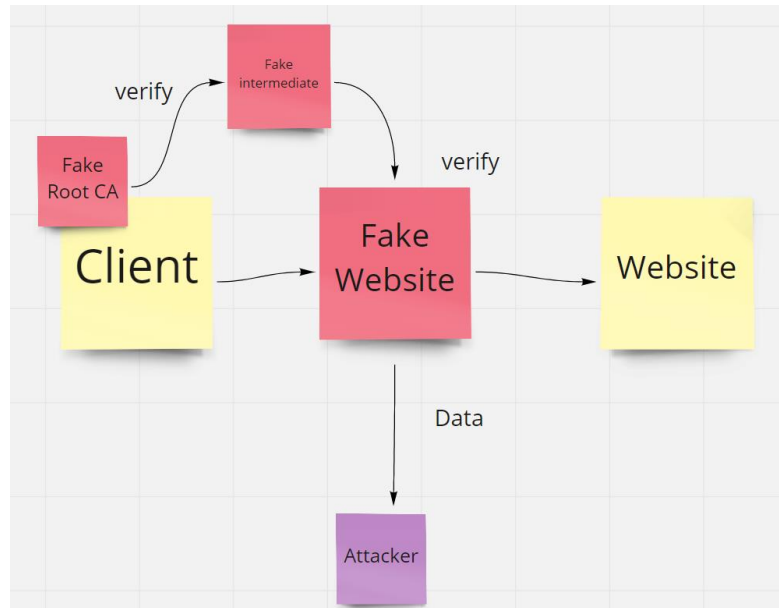
Twitter, google, [www.chula.ac.th](http://www.chula.ac.th), [classdeedee.cloud.cp.eng.chula.ac.th](http://classdeedee.cloud.cp.eng.chula.ac.th)

They are all verified. Especially consider [www.chula.ac.th](http://www.chula.ac.th) which have 3 intermediate which are also all verified.

```
flap9@Avalon MINGW64 ~  
$ python cert-verify.py google_com.cert  
google_com  
Certificate verified  
  
flap9@Avalon MINGW64 ~  
$ python cert-verify.py classdd_com.cert  
lassdd_com  
Certificate verified  
  
flap9@Avalon MINGW64 ~  
$ python cert-verify.py twitter_com.cert  
witter_com  
Certificate verified  
  
flap9@Avalon MINGW64 ~  
$ python cert-verify.py chula_int1_com.cert  
hula_int1_com  
Certificate verified  
  
flap9@Avalon MINGW64 ~  
$ python cert-verify.py chula_int2_com.cert  
hula_int2_com  
Certificate verified  
  
flap9@Avalon MINGW64 ~  
$ python cert-verify.py chula_int3_com.cert  
hula_int3_com  
Certificate verified
```

11. Nowadays, there are root certificates for class 1 and class 3. What uses would a class 1 signed certificate have that a class 3 doesn't, and vice versa?  
**Class3 is specialization of class1 which contain only high-security certificates which check organization and geolocation of server. On the other hand class1 only check domain name.**
12. Assuming that a Root CA in your root store is hacked and under the control of an attacker, and this is not noticed by anyone for months.
  - a. What further attacks can the attacker stage? Draw a possible attack setup.  
**Attacker can do man-in-the-middle spoofing by let client connect to fake**

website that verify by fake root CA that attacker inject to root store.



- b. In the attack you have described above, can we rely on CRLs or OCSP for protection? Please explain

CRLs can't help in this case since fake-root is newly created and not in CRLs.  
OCSP can't help either since it's rely on root which got faked.