## 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 <a href="https://wiki.openssl.org/index.php/Binaries">https://wiki.openssl.org/index.php/Binaries</a>.
- 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

openssl s\_client -connect twitter.com:443 -CAfile ca-certificates.crt

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 opensel 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.)

- 2. What does the error (verify error) in the first command mean? Please explain.
- 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).
- 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.)
- 5. Is there an intermediate CA, i.e. is there more than one organization involved in the certification? Say why you think so.
- 6. What is the role of ca-certificates.crt?
- 7. Explore the ca-certificates.crt. How many certificates are in there? Give the command/method you have used to count.
- 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.
- 9. If the intermediate certificate is not in a PEM format (text readable), use the command to convert a DER file (.crt .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, 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:
       trusted cert = crypto.load certificate(crypto.FILETYPE PEM,
trusted cert pem)
       store.add cert(trusted cert)
    # Create a X590StoreContext 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
```

<sup>&</sup>lt;sup>1</sup> Code taken from <a href="http://www.yothenberg.com/validate-x509-certificate-in-python/">http://www.yothenberg.com/validate-x509-certificate-in-python/</a>. It has been modified for this exercise.

Use your program to verify the certificates of: Twitter, google, www.chula.ac.th, classdeedee.cloud.cp.eng.chula.ac.th

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