**W2 TNE30024 Lab Notaions**

**4.1 Software**

This lab will rely upon the OpenSSL suite of programs.

OpenSSL should be installed in your RULE host. Check that it is installed

using:

which openssl

The response should be:

/usr/bin/openssl

OpenSSL commands are of the form:

openssl <command> <options>

**1. genpkey - Generate a Private Key**

This command is used to generate a private key.

**Usage:**

openssl genpkey -algorithm RSA -out private\_key.pem -pkeyopt rsa\_keygen\_bits:2048

* -algorithm RSA: Specifies the algorithm (RSA is common for SSL certificates).
* -out private\_key.pem: Specifies the output file for the key.
* -pkeyopt rsa\_keygen\_bits:2048: Sets the key length to 2048 bits, which is recommended for good security.

**2. req - Generate a Certificate Signing Request (CSR)**

A CSR is needed when you want a certificate authority (CA) to issue a certificate for your public key. It includes your public key and identifying information.

**Usage:**

openssl req -new -key private\_key.pem -out csr.pem -subj "/C=US/ST=State/L=City/O=Organization/OU=Department/CN=example.com"

* -new: Indicates that a new CSR is being generated.
* -key private\_key.pem: Specifies the private key file to use.
* -out csr.pem: Specifies the output file for the CSR.
* -subj: Allows you to specify the subject DN directly on the command line.

**3. x509 - Sign a Digital Certificate**

This command is used to create self-signed certificates or to sign certificates using your own CA.

**Usage for self-signing:**

openssl x509 -req -days 365 -in csr.pem -signkey private\_key.pem -out certificate.pem

* -req: Indicates that the input is a CSR.
* -days 365: Specifies the certificate's validity period (here, one year).
* -in csr.pem: Specifies the input CSR file.
* -signkey private\_key.pem: Specifies the private key to sign the CSR, making it a self-signed certificate.
* -out certificate.pem: Specifies the output file for the certificate.

**4. verify - Verify the Validity of a Certificate Chain**

This command checks whether a certificate is valid, i.e., signed by a trusted CA, and whether the certificate chain is complete.

**Usage:**

openssl verify -CAfile ca\_certificate.pem certificate.pem

* -CAfile ca\_certificate.pem: Specifies the CA certificate that should be trusted.
* certificate.pem: The certificate file that you want to verify.

**4.2 Construct Root Certificate for Certificate Authority**  
**1. Begin by removing any files from your workspace:**  
cd  
rm \*.\*  
**2. Generate a private key for the root certificate:**  
openssl genpkey -algorithm RSA -out root1.key  
**3. Examine the contents of the root key:**  
cat root1.key  
**4. Generate a self-signed (root) certificate called root.crt from the private key.** This will ask for several fields. Just accept the defaults  
openssl req -x509 -sha256 -new -key root1.key -days 365 -out root1.crt  
**5. Examine the contents of the certificate:**openssl x509 -in root1.crt -text -noout | more  
**6. You have now created a root certificate that can be used to sign other certificates.** You now have the basic capabilities of a Certificate Authority.

**4.3 Create a Simple Certificate signed using the Root Certificate**

**1. Create another key that will be signed using the root key. Call it server1.**

openssl genpkey -algorithm RSA -out server1.key

**2. Generate a signing request for the server.**

You will be asked for a challenge text. Use the text "challenge". For all other fields use the defaults:

openssl req -new -key server1.key -out req.pem

**3. Examine the contents of the request**

openssl req -in req.pem -text -noout | more

**4. Sign the request using your root certificate to generate a new certificate called server.crt:**

openssl x509 -req -in req.pem -set\_serial 10 -CA root1.crt -CAkey root1.key -passin pass:challenge -out server1.crt

**5. Examine the fields of the certificate:**

openssl x509 -in server1.crt -text -noout | more

**6. Take note of the validity duration of the certificate, certificate serial number, domain name and organisation.**

**7. Finally, verify the certificate:**

openssl verify -verbose -CAfile root1.crt server1.crt

**If Error occurs mean that your server and root mismatching identified to be similar, try to assign a common name for the server-side suggestively.**

**4.4 Create a webserver certificate for your RULE host**

Adapt the previous commands to do the following:

**1. Create a directory for storing private keys named keys and another directory for storing certificates called certs.**

**2. Move root1.key to keys and root1.crt to certs.**

**3. Create a new private key called root2.key and put it in the directory keys.**

**4. Use that key to generate another Self-signed Certificate called root2.crt.**

This certificate is to be valid for ten years. The serial number is to be 50.

Place it in the directory certs.

**5. Create a new private key called server2.key.** Put it in the directory keys.

**6. Generate another certificate called server2.crt signed using root2.**

It is to be placed in the certs directory. The organisation is to be TNE30024. The domain name is to be www.rule<nnn>.caia.swin.edu.au where <nnn> is your RULE host number. The validity is to be ten years. The serial number is to be your RULE host number. The owner is to be your name and the email is to be your email address. A challenge of “challenge” is to be included. Other fields are to be the default.

**7. Validate the new certificate by using certificate root2.** What do yousee? Save the text to show the lab supervisor.

**8. Attempt to validate the new certificate server using certificate root1.**

What do you see? Again, save the text to show the lab supervisor for demonstration.

### 4.4 Create a Webserver Certificate for Your RULE Host (WORKING)

#### Step 1: Create Directories for Keys and Certificates

mkdir keys certs

#### Step 2: Move Existing Key and Certificate

mv root1.key keys/

mv root1.crt certs/

#### Step 3: Create a New Private Key Called root2.key

openssl genpkey -algorithm RSA -out keys/root2.key

#### Step 4: Generate a New Self-Signed Certificate Called root2.crt

openssl req -x509 -sha256 -new -key keys/root2.key -days 365 -set\_serial 50 -out certs/root2.crt

#### Step 5: Create a New Private Key Called server2.key

openssl genpkey -algorithm RSA -out keys/server2.key

#### Step 6: Generate Another Certificate Called server2.crt Signed Using root2

openssl req -new -key keys/server2.key -out req2.pem

#### Step 7: Sign the Request Using Your root2 Certificate

openssl x509 -req -in req2.pem -set\_serial 103 -CA certs/root2.crt -CAkey keys/root2.key -passin pass:challenge -days 3650 -out certs/server2.crt

#### Step 8: Examine the Fields of the New Certificate

openssl x509 -in certs/server2.crt -text -noout | more

#### Step 9: Validate the New Certificate by Using Certificate root2

openssl verify -verbose -CAfile certs/root2.crt certs/server2.crt

#### Step 10: Attempt to Validate the New Certificate server Using Certificate root1

Should be failed

openssl verify -verbose -CAfile certs/root1.crt certs/server2.crt

Should be accepted

openssl verify -verbose -CAfile certs/root2.crt certs/server2.crt