Lab 6A Mininet TLS (SSL)1

*CST 311-30, Introduction to Computer Networks, On-line*

**READ INSTRUCTIONS CAREFULLY BEFORE YOU START THE LAB.**

Lab must be submitted electronically to Canvasby 11:59 p.m. on the due date. Late lab assignments will not be accepted.

Background2,3,4,5,6,7,8

In Lab 3, you ran an emulated network that included a simple web server (http.server). In this lab, you will use the Transport Layer Security (TLS) protocol to make this web server more secure. TLS is a modified and more secure version of the Secure Socket Layer (SSL) version 3. For production systems, you would normally request a Public Key infrastructure (PKI) certificate from a well-known Certificate Authority (CA), which acts as a trusted third-party to verify that an entity is who it says it is.

Since getting a certificate can cost time and money, many organizations use self-signed certificates for development systems, which is what you will do in this lab. First, you will establish your Mininet VM as a root CA. Typically, the root CA is created, which is used to sign a certificate for an intermediate CA. Then for security reasons the root CA is taken off-line, and the intermediate CA signs and issues certificates for devices (e.g. servers) and people as required. Since this is a short-lived lab with little security risk, the intermediate CA step can be skipped.

Part 1: Create a Root CA

Acting as a certificate authority (CA) means dealing with cryptographic pairs of private keys and public certificates, which include the public keys. In this part, you will first create a root CA using the OpenSSL module that is already installed on your Mininet-VM. The very first cryptographic pair you will create is the root pair. This consists of the root key (cakey.pem) and root certificate (cacert.pem).

You will then create the certificate request for a Mininet emulated SimpleHTTPServer, again using the OpenSSL module. This certificate request will be used by the root CA to generate this web server’s certificate. Finally, you will run a Python script to bind this certificate to this web server. This will enable a client in the emulated network to securely connect to the SimpleHTTPServer using SSL.

**Prepare to Create Certificate Authority**

1. Power on the VM, and login with your ssh terminal program. Start X-Windows (unless your terminal program starts this by default).
2. PKI keys and certs are stored in the /etc/ssl directory in Ubuntu. Certificates for well-recognized Certificate Authorities are stored in the /etc/ssl/certs subdirectory. You will be using the default settings that are listed in the Open SSL configuration file. Open this file and look for the [ CA\_default ] section. Examine this section.

mininet@mininet-vm:~$ cat /etc/ssl/openssl.cnf

*Note that the variable dir is stated as a relative path and that a new demoCA directory and three subdirectories will have to be created. Also note the filenames for the CA certificate (cacert.pem) and the CA private key (cakey.pem). The extension .pem means that these files will be in the Privacy Enhanced Mail format.*

1. Edit the the openssl.cnf file.

mininet@mininet-vm:~$ sudo nano /etc/ssl/openssl.cnf

Change the variable *dir* at the top of the [ CA\_default ] section to its absolute path: /etc/ssl/demoCA

1. Create the subdirectories listed in the lines below the *dir* line.

mininet@mininet-vm:~$ sudo mkdir /etc/ssl/demoCA

mininet@mininet-vm:~$ sudo mkdir /etc/ssl/demoCA/certs

mininet@mininet-vm:~$ sudo mkdir /etc/ssl/demoCA/newcerts

mininet@mininet-vm:~$ sudo mkdir /etc/ssl/demoCA/private

*You may notice that there is also a directory listed as $dir/crl in the openssl.cnf file. This is for certificate revocation lists (CRLs). You will not be working with CRLs, so there is no need to create this directory.*

1. Two additional files are needed for the CA: one to keep track of issued certificates, and another to keep track of the last serial number used for an issued certificate. You will need to initialize the starting serial number, and you must be at root prompt to enter this number.

mininet@mininet-vm:~$ sudo touch /etc/ssl/demoCA/index.txt

mininet@mininet-vm:~$ sudo su

root@mininet-vm:/home/mininet# sudo echo ‘1000’ > /etc/ssl/demoCA/serial

*This sets the initial serial number at 1000*.

1. It is a good idea to exit the root prompt now

root@mininet-vm:/home/mininet# exit

1. Change directory to etc/ssl/demoCA

mininet@mininet-VM:~$ cd /etc/ssl/demoCA

1. Generate the CA’s RSA private key of size 2048 bits that will be encrypted by the Advanced Encryption Standard (AES) using a 256-bit key. .

mininet@mininet-VM:/etc/ssl/demoCA$ sudo openssl genrsa -aes256 -out cakey.pem 2048

1. You will get the following output, which includes being prompted for a passphrase. The passphrase will prevent anyone who gets your private key from generating a root certificate of their own.

Generating RSA private key, 2048 bit long modulus

............................+++

.....................................+++

e is 65537 (0x10001)

Enter pass phrase for cakey.pem:

Verifying - Enter pass phrase for cakey.pem

1. Enter and verify the passphrase for cakey.pem. (You can use spaces in the passphrase, e.g., Nifty birds shoot acorns.)

**Generate Root CA Signing Certificate**

1. Now create the root CA certificate. (Input command all in one line):

mininet@mininet-VM:/etc/ssl/demoCA$ sudo openssl req -x509 -new -nodes -key cakey.pem -sha256 -days 1825 -out cacert.pem

*This command will create a certificate that is valid for 5 years and follows the X.509 standard. It uses an SHA256 hash. The -x509 flag indicates that this will be a self-signed certificate instead of a certificate request.*

1. Enter the passphrase you entered in Step 7 when prompted. Then answer the series of questions that follow. Your answers to these questions will be embedded in the certificate

| **Country Name (2 letter code)** | The two-letter country code where your company is legally located.  (US is the code for the United States) |
| --- | --- |
| **State or Province Name** | The state/province where your company is legally located. *(Any state name works.)* |
| **Locality Name (e.g., city)** | The city where your company is legally located (A*ny city works.*). |
| **Organization Name (e.g., company)** | Your company's legally registered name. (*Use SCD for this lab*.). |
| **Organizational Unit Name (e.g., section)** | The name of your department within the organization. *(Use CST311 for this lab.)* |
| **Common Name (e.g., server FQDN)** | For a server, the Common Name (CN) is the fully-qualified domain name (FQDN)  (*Use* ca.csumb.test). |
| **Email Address** | Your email address. (*You can leave this option blank; simply press****Enter***.) |

1. Look at this root certificate in default form with the following command:

mininet@mininet-VM:/etc/ssl/demoCA$ cat cacert.pem

*Get a screen shot of the root certificate you created.*

1. Get details about the root certificate, by decrypting it into a readable format.

mininet@mininet-VM:/etc/ssl/demoCA$ sudo openssl x509 -text -noout -in /etc/ssl/demoCA/cacert.pem

Note that this certificate is valid for 5 years*.*

*Get a screen shot of this decrypted root certificate.*

1. Install Your Root Certificate Key in Mininet VM Private Store

Normally, you would need to add the root CA certificate to any laptops, desktops, tablets, and phones that will be accessing your HTTPS sites. For this lab, you will only need to install this certificate in the Mininet VM, because the emulated client and web server will both be running in this VM. While still in the directory /etc/ssl/demoCA, move the root key to the private directory.

mininet@mininet-VM:/etc/ssl/demoCA$ sudo mv ./cakey.pem ./private

1. Now you need to copy the root CA certificate into the /etc/ssl/certs directory, where all the other trusted CA certificates are stored in a chain. A simple copy does not work. You will use the *ca-certificates* application to do this.

*Install this application package if it is not already installed*.

mininet@mininet-VM:/etc/ssl/demoCA$ sudo apt-get update

mininet@mininet-VM:/etc/ssl/demoCA$ sudo apt-get install -y ca-certificates

1. Next copy the root CA certificate to the directory that the ca-certificates application uses, changing to a crt extension (all in one line).

mininet@mininet-VM:/etc/ssl/demoCA$ sudo cp cacert.pem /usr/local/share/ca-certificates/cacert.crt

1. Now run the ca-certificates application.

mininet@mininet-VM:/etc/ssl/demoCA$ sudo update-ca-certificates

*Look in the /etc/ssl/certs directory to make sure that the root certificate has been copied correctly.*

Part 2: Generate the Server Certificate

1. Remain in the /etc/ssl/demoCA subdirectory.
2. Generate a new 2048-bit RSA private key for your server using the command below (all in one line). Use the fake domain for your web server (cst311.test).

mininet@mininet-VM:/etc/ssl/demoCA$ sudo openssl genrsa -out cst311.test-key.pem 2048

1. Generate a certificate signing request to “send” to the root CA, using the private key you generated above. To do this, enter the following command (all in one line):

mininet@mininet-VM:/etc/ssl/demoCA$ sudo openssl req -nodes -new -config /etc/ssl/openssl.cnf -key cst311.test-key.pem -out cst311.test.csr

*The -nodes flag means “do not encrypt the private key”. If you encrypt the private key with a passphrase, you will have to input the passphrase every time you start the web server, which can be a nuisance if you are automatically starting the web server by a script. Since this is a lab, you can skip the key encryption. If you are in a production environment, you will need to take extra security steps to protect the private key.*

1. Answer the series of questions that follow using the template below. Your answers to these questions will be embedded in the certificate. *(Note: You can put the answers to these questions in the command line instead of entering them interactively. You may find this useful in Programming Assignment #4. See Note 7 at end of this document; in this example State = CA and Locality=Seaside.8)*

| **Country Name (2 letter code)** | The two-letter country code where your company is legally located.  (*US is the code for the United States*.) |
| --- | --- |
| **State or Province Name** | The state/province where your company is legally located. (*Enter any state*.) |
| **Locality Name (e.g., city)** | The city where your company is legally located. (*Enter any city.*) |
| **Organization Name (e.g., company)** | Your company's legally registered name. (*Use CST311 for this lab*) |
| **Organizational Unit Name (e.g., section)** | The name of your department within the organization. (*Use Networking for this lab*) |
| **Common Name (e.g., server FQDN)** | For a server, the Common Name (CN) is the fully qualified domain name (FQDN) 9  (*Use* www.cst311.test). |
| **Email Address** | Your email address. (*You can leave this option blank; simply press****Enter****.)* |
| **A challenge password** | Leave this option blank (s*imply press****Enter***). (*This "challenge password" is not the same thing as a passphrase used to encrypt the secret key. The "challenge password" is basically a shared-secret nonce between the requestor and the Certification Authority, embedded in the CSR, which the CA may use to authenticate the requestor should that ever be needed. This password was previously needed for certificate revocation but is no longer needed*.) |
| **An optional company name** | Leave this option blank (s*imply press****Enter***). |

1. Now use the Root CA to create the X.509 server certificate that is valid for 365 days. Sign the certificate with the CA certificate (cacert.pem). To do this, enter the following command (all in one line):

mininet@mininet-VM:/etc/ssl/demoCA$ sudo openssl x509 -req -days 365 -in cst311.test.csr -CA cacert.pem -CAkey ./private/cakey.pem -CAcreateserial -out cst311.test-cert.pem

1. You will be asked to enter the passphrase for cacert.pem. Use the passphrase you chose in Step 10.

*You have created a server certificate, that is valid for one year, and signed it with your CA’s certificate*.

1. Examine the server certificate you just created, decrypted into a readable format.

mininet@mininet-VM:/etc/ssl/demoCA:$ sudo openssl x509 -text -noout -in cst311.test-cert.pem

*Get a screen shot of this decrypted server certificate.*

1. Copy the server certificate and its key into the newcerts and private subdirectories, respectively, that you created in Step 3. These directories must match the variables where you define this location in the Python script that is provided in Appendix A.

mininet@mininet-vm:/etc/ssl/demoCA:$ sudo mv cst311.test-cert.pem newcerts

mininet@mininet-vm:/etc/ssl/demoCA:$ sudo mv cst311.test-key.pem private

1. For security purposes, you should protect the server’s private key. This is especially important since you did not encrypt the key. One way to provide some level of protection is to change the permissions on the /etc/ssl/demoCA/private directory and all files in this directory to read and write only for root. Again, you will have to be at the root prompt.

mininet@mininet-vm:/etc/ssl/demoCA:$ sudo su

root@mininet-vm:/etc/ssl/demoCA# chmod -R 600 private

1. Exit the root prompt.

root@mininet-vm:/etc/ssl/demoCA# exit

1. Modify the Mininet VM’s host file to include Common Name (CN)10 of the CA server (ca.csumb.test), using the local host IP address of 127.0.0.1 and the SSL web server ([www.cst311.test](http://www.cst311.test)) that you input in the certificate request in Part 2. For this web server IP address, use the default address given to the host h2 in the basic mininet topology (10.0.0.2). Place these entries directly below the existing entries for ‘localhost’ and ‘mininet-vm’.

mininet@mininet-vm:/etc/ssl/demoCA$ sudo nano /etc/hosts

1. Verify that the entry for ca.csumb.test in the /etc/hosts file works.

mininet@mininet-vm:/etc/ssl/demoCA$ ping ca.csumb.test

*Pinging* [*www.cst311.test*](http://www.cst311.test) *will not work until you start the Mininet emulation. You will do this in Part 3.*

1. Change to the mininet home directory.

mininet@mininet-vm:etc/ssl/demoCA$ cd $HOME

1. Create the Python script provided in Appendix A. and store in the $HOME/CST311 directory you created in Lab 0.

Part 3: Run the HTTPS Web Server

You will run this part of the lab using the basic mininet topology of two-hosts and one switch.

1. Start the mininet emulation for the basic topology. This time use the -x argument to open xterm windows for all the emulated devices.

mininet@mininet-vm:~$ sudo -E mn -x

*Opening the xterm windows put you at the root prompt for all the emulated devices. This is required for the host running the web server software, because you have made the private keys readable only by root (in step 27).*

1. Verify that the name resolution for the CN of the SSL web server works.

mininet> h1 ping [www.cst311.test](http://www.cst311.test.com)

You should get a successful ping.

1. Start Wireshark on host s1 and set the capture to either s1-eth1 or s1-eth2.

“**switch: s1**” root@mininet-vm:~# wireshark &

1. Run the Python script provided in Appendix A on host h2.

“**host: h2**” root@mininet-vm:~# python3 CST311/tlswebserver.py

*The message “listening on port 4443” appears.*

1. Fetch the index page of the web server. You will need to use the FQDN of the webserver in your request, as well as specifying the test port. Note that the test port is 4443.10

“**host: h1**” root@mininet-vm:~# wget <https://www.cst311.test:4443>

1. If you are having problems, you can use the following command to troubleshoot the client handshake with the TLS server.

“**host: h1**” root@mininet-vm:~# openssl s\_client www.cst311.test:4443

*A successful result will show the server’s certificate, as well information about the ciphers used.*

1. Examine the Wireshark capture11, particularly, the Client Hello, Server Hello, Client Key Exchange, and New Session Ticket packets. In the Client Hello and Server Hello Packets, note that there is a field in each called Random. These values in these fields are nonces used to compute the Master Secret and defend against the “connection replay attack”. 12

*Get a screenshot of the Wireshark capture, select the Server Hello packet. In the packet details,* *expand* *the TLS Record Layer Server Key Exchange after expanding the Transport Layer Security. (Note Info field of the Server Hello packet will show Server Hello, Certificate, Server Key Exchange, Server Hello Done.)*

1. Exit Mininet and shut down the Mininet VM cleanly.

### What to Turn in via Canvas:

1. Answer the questions on the Quiz.11
2. A screen shot of the root certificate you created as required in Step 13.
3. A screen shot of this decrypted root certificate as required in Step 14.
4. A screen of the decrypted server certificate as required in Step 25.
5. A screenshot of the Wireshark capture as required in Step 39.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Notes and References:

1. James F. Kurose and Keith W. Ross (2021)*, Introduction to Computer Networking: A Top-down Approach*, 8th edition, Hoboken NJ:Pearson, Section 8.6, pp 644-650.
2. <https://docs.python.org/2/library/simplehttpserver.html>
3. [SimpleHTTPServer with SSL | The Road to Elysium (fbarr.net)](https://jorge.fbarr.net/2017/06/11/simplehttpserver-with-ssl/)
4. [Create the root pair — OpenSSL Certificate Authority — Jamie Nguyen (jamielinux.com)](https://jamielinux.com/docs/openssl-certificate-authority/create-the-root-pair.html)
5. [How to Create Your Own SSL Certificate Authority for Local HTTPS Development (deliciousbrains.com)](https://deliciousbrains.com/ssl-certificate-authority-for-local-https-development/)
6. [OpenSSL Certificate Authority (CA) on Ubuntu Server (networklessons.com)](https://networklessons.com/uncategorized/openssl-certification-authority-ca-ubuntu-server)
7. [ssl — TLS/SSL wrapper for socket objects — Python 3.11.0 documentation](https://docs.python.org/3/library/ssl.html)
8. sudo openssl req -new -config /etc/ssl/openssl.cnf -key cst311.test-key.pem -out cst311.test.csr -subj “/C=US/ST=CA/L=Seaside/O=CST311/OU=Networking/CN=www.cst311.test”
9. The Common Name (CN) is the value of the bind-to-address variable, e.g. [www.example.com](http://www.example.com).
10. Port 4443 is typically used for test SSL web servers in lieu of the standard port 443, because you typically need root privileges to work with ports below 1024.
11. Wireshark questions adapted from j.F. Kurose and K.W Ross, op. cit, *SSL Wireshark Lab version 8.1.*

**APPENDIX A: Python Script to Enable SSL in SimpleHTTPServer**

**(tlswebserver.py)**

#!/usr/bin/env python3

import http.server

import ssl

# Variables, including location of server certificate and private key file

server\_address = "www.cst311.test"

server\_port = 4443

ssl\_key\_file = "/etc/ssl/demoCA/private/cst311.test-key.pem"

ssl\_certificate\_file = "/etc/ssl/demoCA/newcerts/cst311.test-cert.pem"

#Context is the TLS Server with its certificate file and key file location

context = ssl.SSLContext(ssl.PROTOCOL\_TLS\_SERVER)

context.load\_cert\_chain(ssl\_certificate\_file, ssl\_key\_file)

## Don't modify anything below

httpd = http.server.HTTPServer((server\_address, server\_port), http.server.SimpleHTTPRequestHandler)

httpd.socket = context.wrap\_socket(httpd.socket,

server\_side=True)

print("Listening on port", server\_port)

httpd.serve\_forever()