# CS 577 Cybersecurity Lab

# Lab 1 – due 9/11/14 6:16pm


# Georgios Portokalidis – Stevens Institute of Technology


**This lab will be done on the linux-lab.**

You will need ssh, PuTTY, and a terminal to connect.

If you need to open multiple terminals to the same host for the exercise make sure you are indeed connected to the same host. linux-lab.cs.stevens.edu is just an alias that connects you to one out of many hosts. For instance, if after connecting to linux-lab you are in host1, make sure to connect to host1.srcit.stevens-tech.edu from the second terminal.

NOTE: Make sure nobody else is using the port numbers used in the examples below

**Exercise 1. Man-in-the-Middle Attack (60%)**

A client C is communicating with a server S through a secure channel established using SSL. C has the certificate of the server, however, it is not correctly verifying it. Create a socket proxy to perform a man-in-the-middle attack and eavesdrop on the data sent by the client. The proxy will write all the data received by the client and server into files client.log and server.log.

**a) Generate an RSA public/private-key pair**

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

What is the e value printed out?

Try creating bigger keys. Can you notice any difference when generating it?

**b) Generate a self-signed certificate**

```
openssl req -new -x509 -key private.pem -out cacert.pem -days 1095
```

Provide all the information required for your server, do not use the default values!

What purpose do the information you enter for the certificate serve?

**c) Safe communication between client and server**

Client has the server certificate and verifies it.

```
openssl s_server -cert cacert.pem -key private.pem -accept 8900

openssl s_client -connect 127.0.0.1:8900 -state -verify 1 -CAfile
cacert.pem -verify_return_error
```

**d) Unsafe communication between client and server**

Client has the server certificate but does not verify it correctly.

```
openssl.exe s_client -connect 127.0.0.1:8900 -state -verify 1 -CAfile
cacert.pem
```

**e) Perform a MiTM attack.**

Write a proxy and setup your own "malicious" certificate. To emulate you interposing on the network have your proxy listen on another port (e.g., 9000) and have the client connect to that port.

./bad_proxy –listen 127.0.0.1:9000 –connect 127.0.0.1:8900 –cert bad_cert.pem –key bad_private.pem

Have the proxy log all traffic between client and server to client.log and server.log.

**f) Confirm if adding back -verify_return_error would prevent the MiTM attack.**

Try to perform the MiTM attack after the client uses the above option.

To help you:

Here is a neat python proxy sample: http://voorloopnul.com/blog/a-python-proxy-in-less-than-100-lines-of-code/ (However, no restriction about which programming language is used. C is recommended!)

**Exercise 2) Create you implementation of HMAC for generating and verifying MAC codes using SHA-1 (40%)**

Create a function

void hmac_generate(unsigned char *msg, unsigned long msglen, unsigned long key, unsigned char *digest)

To help you:

OpenSSL has a SHA1 function which you can use.