# CMPSC 443: Computer Security

Fall 2025
Project 4: SSH Protocol and MITM Attack
Due: 11:59 pm (eastern time), Dec 12, 2025
Total Points: 150, Bonus Points: 100

December 1, 2025

## 1 Introduction

This project has two main parts. In the first part, you will develop a client-server system that provides secure file transfer. You will implement part of the SSH protocol to construct a secure channel between the client and server over which to perform secure file transfer. You will use the OpenSSL library to implement the SSH protocol. The SSH protocol produces a symmetric key shared between the client and server, and you will use the OpenSSL library once again to use that key to transfer the file. In the second part, you will develop a man-in-the-middle (MITM) attack against the SSH protocol that you implemented in part 1. You will create an MITM that pretends to be the target server, but instead opens a secure connection to the client that it can use to read client communications forwarded to the target server.

The system you produce must run on the Westgate Linux lab machines. These machines are named e5-cse-135-XX.cse.psu.edu, where XX is a number between 01 and about 40. All these machines should be identical and already have the OpenSSL library installed. You should SSH into those machines to verify that your code works. We developed and tested the project code on those machines, so should work fine, but it is up to you to make sure. **You will need to speak to the CSE IT folks (`helpdesk@cse.psu.edu`) if you do not have access to those machines.**

## 2 Overview of Tasks

### 2.1 Part 1

The main task in part 1 is to implement the SSH protocol as described in the paper (`https://syed-rafiul-hussain.github.io/index.php/teaching/cmpsc443-f25/docs/ssh.pdf`). There are two functions `client_authenticate` and `server_protocol` to be implemented.

### 2.2 Part 2

The original SSH protocol that you will implement in part 1 is prone to a kind of MITM attack called **Server Spoofing**. In this attack, a malicious entity may intercept client communications intended for a target server to create a MITM connection, where the malicious entity pretends it is the server to the client and as a client to the target server. Since the SSH protocol does not authenticate that the public key provided by the server is really associated with the target server to which the client is intending to connect, a malicious entity can take advantage of this. The SSH systems take some measures to prevent this attack outside the crypto protocol. You will extend the server of part1 to produce a MITM server that performs this server spoofing MITM attack against the (unmodified) client and (mostly unmodified) server implemented in part 1.

Use the client and server you implemented in part1. For this attack, you will continue to run these components unmodified except for a small modification to one function `(receive_file)` used by the server.

## 3 Implementation

### 3.1 Create public and private keys

You can create the public and private keys using the OpenSSL system using the following commands:

```
# generate key pair - mykey.pem holds private key
openssl genrsa -out mykey.pem 2048

# extract public key in basic format - pubkey.pem is in PKCS#8 format
openssl rsa -in mykey.pem -pubout -out pubkey.pem

# convert public key to RSA format - rsapub.pem holds public key
openssl rsa -pubin -in pubkey.pem -RSAPublicKey_out > rsapub.pem
```

### 3.2 Running the Client and Server

The server program will be started by the following command line:

```
cmpsc443-p4-server <private-key-file> <public-key-file> <server-port> <real-server-ip:port>
<MIMT-option>
```

where (1) the `<private-key-file>` is the name of the file that stores the private key for the server and (2) `<public-key-file>` is the name of the file that stores the corresponding rsa format public key for the server.

`MIMT-option` can be either 0 or 1. If it is set to 1, it is a malicious MIMT server. Then `<server-port>` means the port used by the malicious server, and the `<real-server-ip:port>` is the real server ip and port. If it is set to 0, it should be a normal server, `<server-port>` should be the real port. And `<real-server-ip:port>` will be ignored.

The client program will be started by the following command line:

```
cmpsc443-p4 <file-to-transfer> <server-ip-address:server-port>
```

where (1) the `<file-to-transfer>` is the file path name of the file to transfer from the client to the server and (2) `<server-ip-address>` is the IP address of the server host.

Start the normal server first, as it will wait for connection requests from clients. When a connection request is received from a client the sequence of steps will be performed.

1. Perform the SSH protocol: The client will initiate the SSH protocol to produce a symmetric key to be shared by the client and server.
2. Transfer the file: The `<file-to-transfer>` will be sent encrypted and integrity protected from the client to the server. The server will store the file in a directory called "shared" under the directory from which the server is run.

3. Server awaits next request: The client will terminate and the server will await the next request from the next client.

## 3.3 Part 2: Bonus

You will run the server with MITM with option “1”. This will invoke `server_secure_transfer` from part 1. You need to modify that function to perform the MITM proxy attack.

### 3.3.1 Server Spoofing Attack

In this attack, you will modify the code in server secure transfer to perform the SSH protocol as the server to a part 1 client, but also initiate a new SSH connection to a part 1 server as the client. This attack emulates the case where a MITM may obtain an IP address for another server or sit between the client and server on the network. The attack must:

1. construct a SSH connection with the client sufficient for the client to transfer the file;
2. replay the client messages to create a second SSH connection to another normal server
3. modify the client file by appending ‘malicious’ at the end of it.
4. forward the client’s messages to transfer the modified file to the normal server.

The easiest way to do that is to start with the server protocol and add the steps necessary to replay client messages to the part 1 server sufficient to transfer the file. This should not take a lot of code given what you have for part 1.

To test this, you can do this:

1. Run a normal server at port X, say 9156.
2. Run a malicious server at port Y, say 9999, and forwarding to X.
3. Run a client connecting to port Y.

Now, the normal server should obtain a modified message.

## 4 Deliverable

Please submit a zip containing only two files : cmpsc443-proto.c and a PDF report.
In the report, it should include

1. The screenshot of a successful normal server/client file transfer, include the commmands, and the message received
2. The screenshot of a successful MITM attack, include the commmands, and the modified message received;
3. Explain two ways to mitigate or prevent this attack, each in one sentence.

## 5 Grading (Total 150 points, Bonus 100 Points)

1. We can build and run what you have submitted without any compilation or runtime errors (20 points).
2. SSH protocol (100 points).

3. Server spoofing attack works (Bonus 100 points).
4. Report (30 points)