CSE 543: Computer Security
Project 2: SSH Protocol with MITM Attack

Due: 11:59 pm (eastern time), October 12, 2020

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 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 a 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 cse-p204instXX.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

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/cse543-f20/docs/ssh.pdf`). Since any user is authorized to upload a file in this project, you only have to implement Steps 1-4 of the protocol, corresponding to messages 1-4 in the ProtoMessageType enumeration in `cse543-proto.h`. You are going to use the OpenSSL functions provided to implement the SSH protocol.

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.

3 Project Tasks

You should first download the tarball from `https://syed-rafiul-hussain.github.io/index.php/teaching/cse543-f20/projects/cse543-f20-p2.tar.gz`. The tarball has the initial code for part 1 and part 2. The project includes source code and a Makefile for building the tarball for your project for submission. We suggest you perform the project tasks listed below in the following order.

3.1 Part 1

The initial version of the program includes two functions `test_rsa` and `test_aes` that encrypt a message. These functions demonstrate how to perform symmetric and public key encryption with OpenSSL library, which should be a big help in the project.

3.1.1 Write the functions to build encrypted messages for sending and decrypted received messages

There are two pairs of functions for you to implement: `seal/unseal_symmetric_key` for public key crypto and `encrypt/decrypt_message`. These encryption functions must perform encryption and produce buffers containing the data necessary for the other party to decrypt. The decryption functions must extract the necessary information from a sent buffer and perform the decryption.

3.1.2 Write the function to generate pseudo random values

Develop the function `generate_pseudorandom_bytes` by using the OpenSSL functions for producing pseudo random values.

3.1.3 Develop the SSH Protocol

Implement the client and server portions of the SSH protocol, as described in the paper (`https://syed-rafiul-hussain.github.io/index.php/teaching/cse543-f20/docs/ssh.pdf`). There are two functions `client_authenticate` and `server_protocol` to be implemented.

3.1.4 Transfer the file securely

Implement the encryption and transfer functionality to send the file from the client to the server in the function `transfer_file`.

3.2 Part 2

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

3.2.1 Implement server spoofing attack

In the MITM attack, you will modify the code from the part 2 (see below) to perform the server spoofing attack.

4 Implementation

4.1 Part 1

4.1.1 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
```

4.1.2 Run the Client and Server

The server program will be started by the following command line:
`cse543-p1-server <private-key-file> <public-key-file>`
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 public key for the server.

The client program will be started by the following command line:
`cse543-p1 <file-to-transfer> <server-ip-address>`
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 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.

4.1.3 Perform the SSH protocol

The client will initiate the SSH protocol to produce a symmetric key to be shared by the client and server.

4.1.4 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.

4.1.5 Server awaits next request

The client will terminate and the server will await the next request from the next client.

4.2 Part 2

The code for part 2 consists of one new file `cse543-p2.c`, which replaces the `cse543-p1.c` file from part 1. The `cse543-p2.c` code only differs from the part 1 code only that the interface to start the server is slightly different (see MITM USAGE in the file). The first two arguments are the MITMs public and private keys as before. These keys will be different than the target server's public and private keys, of course. The third argument is still an IP address, but it is the IP address of the real (target) server. The fourth argument is the "MITM option" which you should set to 1 to perform the server spoofing attack. In summary, MITM should run as below:

```
cse543-p2-server <private_key_file> <public_key_file> <real-server-IP-address> 1
```

4.2.1 Use Your Part 1 Code

The client and target server will be largely unchanged from part 1. To run client, MITM, and target server on one machine, you need to take the following steps to have the target server communicate on a different port:

1. make a copy directory of part1 code and rename the new directory to part1_server. This is going to be the directory where you run the target server.
2. change line 36 in `cse543-network.h` from `#define PROTOCOL_PORT 9165` to `#define PROTOCOL_PORT 9166`. Now instead of port `1965`, the target server will listen on port `9166` .

MITM is set up to communicate with the client on port `9165` and with the target server on port `9166`. You will run 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, as described below.

4.2.2 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 (from part 1); (2) replay the client messages to create a second SSH connection to another part 1 server; and (3) forward the client's messages to transfer the same file to the part 1 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 for this, run the client and MITM server and the real server on the same machine** The real server will receive connections listening to port 9166. MITM will receive connections listening on port 9165 for the client, and 9166 for the real server.

5 Deliverable

Please submit a tarball of your code with the provided Makefile using make tar.

6 Grading

1. We can build and run what you have submitted without incident (10 points).
2. Encrypted communication works (10 points).
3. Generate pseudo random data (10 points).
4. SSH protocol implementation and specification (40 points)
5. Transfer file securely (10 points).
6. Server spoofing attack works (20 points).