Networks and Systems Security II - Winter 2025

Sambuddho Chakravarty

February 20, 2025

Exercise 2 (total points: 50(+25 bonus))

Due date: March 5. Time: 23:59 Hrs.

There are three parts to this exercise. Each of these would involve the use of the VMs created in the previous exercises. You need to attempt all three parts.

1 Directory Encryption using `eCryptfs` (total points: 25)

This assignment aims to familiarize you with using `eCryptfs` directory encryption scheme. You are expected to perform the following tasks:

1. Install `eCryptfs` on one of your VMs that you used in the previous exercise.
2. Create a subdirectory which would be encrypted directory.
3. Mount it as an encrypted directory using `eCryptfs`. Use either your name as the passphrase.
4. Once mounted every file subsequently added to this directory would be encrypted and stored.
5. Create a temporary file – `temp1` and add some plaintext to it, while the subdirectory is still mounted using `eCryptfs`.
6. Unmount the directory and validate that the contents of the file `temp1` are encrypted. Print the contents on the screen and take a snapshot of the same (to be submitted).
7. Mount the subdirectory again and decrypt the file using the passphrase and print the contents on the screen.

What you need to submit:

You need to submit a report showing the following:

1. Screenshot for each of the steps required for the setup, showing all the commands executed and their outcomes (10 points).

2. Screenshot showing you display the encrypted file and the unencrypted one (10 points).
3. Description of the commands used, in terms of the arguments supplied and why it was run (*i.e.* the objective of running the command) (5 points).

2 Using `OpenSSL` Toolkit to build an AES-enabled echo server and client using `netcat` (total points: 25)

The objective of this exercise is to familiarize you with using the `OpenSSL` toolkit to do basic AES encryption, besides getting hands-on experience using `netcat`. In this exercise, you need to build a *Hello Alice!* server. When a client connects to the server and sends a proper noun, *e.g.* "Bob", the server responds with "Hello Bob!". However, the communication between the client and server must be encrypted with AES256-GCM cipher (available in the `OpenSSL` toolkit).

The server listens on a particular port number for incoming connections. Upon accepting connections, it receives the bytes and decrypts them (considering they are encrypted) using AES256-GCM, available in the `OpenSSL` framework/toolkit. Thereafter, the server must respond back with a string that prepends "Hello" in front of the decrypted string. The response string must also be encrypted the same was, as done by the client.

You can assume that the client and server use a known symmetric key. You may also assume that the IV is known apriori. One easy way is to rely on a passphrase and let OpenSSL derive the IV and Key pair through KDF functions internally.

You must also capture the network traffic between the client and server using `Wireshark/tcpdump` to confirm that both parties are using encrypted traffic.

You are supposed to implement the above mere with appropriate shell commands (*e.g.* `bash`) and no C/Python/Java/C++/whatever other language program should be there. You may also rely on additional known shell commands like `cat`, `echo`, `sed`, *etc.*. No uncommon/unknown command (built-in or external) should be used.

What you need to submit:

You need to submit a report showing the following:

1. Screenshot for each of the steps required for the setup, showing all the commands executed and their outcomes (10 points).
2. Screenshot of `wireshark/tshark/tcpdump` showing directional encrypted traffic (10 points).
3. Description of the commands used, in terms of the arguments supplied and why it was run (*i.e.* the objective of running the command) (5 points).

Using `OpenSSL` Toolkit and `netcat` to build an AES-enabled `scp` (Bonus points: 25)

Similar to the task above, you are required to build an AES-enabled `scp` (secure copy, that usually relies on SSH). Your program would rely on the symmetric cipher (instead of SSH). To test it, you can send a text file through the AES encrypted channel and show that the file was correctly received. Here, instead of AES-GC,M you are required to generate an HMAC and send that along side the file. Not only should the file(s) be correctly transmitted and received, but the HMACs must also be validated.

What you need to submit:

You need to submit a report showing the following:

1. Screenshot for each of the steps required for the setup, showing all the commands executed and their outcomes (10 points).
2. Screenshot of `wireshark`/`tshark`/`tcpdump` showing directional encrypted traffic (10 points).
3. Description of the commands used, in terms of the arguments supplied and why it was run (*i.e.* the objective of running the command) (5 points).