Networks and Systems Security II - Winter 2022

Sambuddho Chakravarty

March 7, 2022

Assignment 3 (total points: 45)

Due date: March 28, 2022.@ 23:59 Hrs. (hard deadline)

This exercise requires you to write a secure file copy program, which works similar to `scp`. The program must use `netcat` as a client and a server, where the payload is fed to (or from) a separate program that encrypts (or decrypts) the data being transmitted. The program emulates the `scp` program, *i.e.* the program to copy files encrypted and encapsulated inside a `ssh` connection/tunnel.

The parent program must fork two child processes or threads. In case of the client program, one of these child process/threads should read the contents of a file and encrypt it and send it (using any known IPC mechanism) to the other sibling process/thread. The other thread's job is to encapsulate the encrypted data into TCP packet payload and send it to the peer (*i.e.* the server).

Correspondingly, the server program must also create two child processes or threads. One of these listens on a particular IP/Port number (to which the client connects). This is also to be implemented using the `netcat` program. This thread/process reads the data sent by the client and sends it to the other process/thread (of the server), using some IPC mechanism. This other process/thread decrypts it and saves it in a local file.

The encrypted ciphertext that the client shares with the server, must also sent together with an IV/nonce and a HMAC signature, derived from the key used to encrypt the data. The server must not only decrypt the data, but also validate the HMAC, *before* saving it to a local file.

Important assumptions:

1. The two programs use a pre-shared key to encrypt (or decrypt) the data before (or after) sending (or receiving) it. One way to derive the key is by reading the requisite bytes from the file `/dev/urandom`.
2. Both programs could run on the same machine, whereby both use the local host prefixes and subprefixes (127.0.0.0/8) and ephemeral ports. Alternatively, the could be on separate VMs as well.
3. The file encryption/decryption and generation/validation of the HMAC signatures MUST involve the use of `openssl libcrypto` library. You could use the envelope functions (starting names like `EVP_`) for doing the same.
4. Every file is accompanied with a single HMAC, which is sent along with the packets bearing the encrypted bytes corresponding to the file being transmitted. This must be validated before the decrypted data is stored in a local file.
5. The ciphertext is also accompanied with the IV/nonce used to encrypt the data. Both the IV/nonce and the key can be derived from the reading bytes from the pseudo-device file `/dev/urandom`.

The following figure can help you get a better picture of how this is to be done.

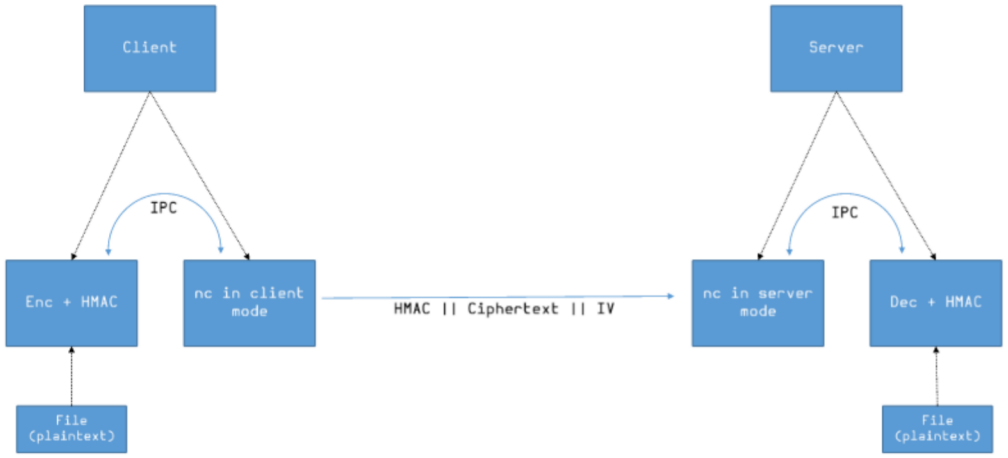

Figure 1: The secure file copy client and server programs.

Some useful links with examples on how to use envelope functions:
`https://www.openssl.org/docs/man1.1.1/man7/evp.html` `https://wiki.openssl.org/index.php/EVP_Symmetric_Encryption_and_Decryption`
`https://home.uncg.edu/cmp/faculty/srtate/580.s13/digest_ex.php`

What you submit/Rubric:

1. Correctly compiled programs (both client and server) (via a Makefile) – 10 points.
2. Correct functioning of all the commands of the program, designed using `sockets` API, `libcryto` – 20 points.
3. HMAC correct validation: To test HMAC validation, you could you `netfilter` kernel module wherein you flip certain bits of the encrypted payload of packets (generated using `netcat`). The HMAC validation should fail in such cases. The file MUST not be written to the server's disk/filesystem – 10 points.
4. Documentation describing the system design and the assumptions made – 5 points.