Andy Qin

COEN146 L

12 March 2022

Lab 9 Report

Introduction

* This report is a documentary of how I personally approached the Lab and using the OpenSSL library.
* My goal is to demonstrate the steps I performed to completer the lab, and make other students understand the process of this lab.
* Lab 9 deals with network security, with some experiencing of different encryption methods like monoalphabetic encryption, public/private key, hashing, and certification.

Procedures

Get the OpenSSL Software

* If using ECC Linux/ MAC Windows/ system
  + OpenSSL is already installed
  + Check the integrity of software by typing “openssl version -a” in the command line.
* If using Personal Linux/MAC/ Windows
  + OpenSSL is installed by default as well, nut modifications of settings are needed to run some of the openSSL features correctly (the public/private key pairs).
  + More information from https://www.openssl.org
* MY CHOICE AND SUGGESTION
  + **USE ECC COMPUTER SYSTEM!!! IT IS SAFE AND SET-UP FOR YOU!!!**
  + IT CAN SAVE YOU A LOT OF TIME FROM FINDING HOW TO CHANGE YOUR PC SETTINGS TO USE OPENSSL
  + OPENSSL MODIFIES YOUR SECURITY SETTINGS, YOU MIGHT NOT BE ABLE TO REVERT CHANGES!!!

Use OpenSSL on .txt file

1. First: **create a random .txt file without any encryption**

Type ***echo "This is a secret file that has important information which we do not want to reveal" > plaintext*** in the command line

NOTE: what this command does is **write the sentence in the quote to a .txt file called “plaintext”**; so, **the sentence in the quote “ ” and the name of the .txt file can be changed to any other name** (you name it!)

1. Second: **try simple encryption techniques like Caesar Shift or Monoalphabetic encryption (symmetric key encryption)**

**NOTE**: A **Caesar Shift** is a encryption technique of plain texts where **each distinct letter is replaced by another letter** (for example; I represent actual A as B, B as C, D as E; so what actually is “BAD” in normal syntax would be “CBE” after encryption), and the replacement is one-to-one (which **means every letter is matched to a different unique letter**)

Assume we map **“a-z”** to “***qgvmftzyceolhsuwbjaxdnikpr”:***

Type ***tr 'a-z' 'qgvmftzyceolhsuwbjaxdnikpr' < plaintext > ciphertext*** to create a ciphertext of the normal plaintext using the above mapping

Parameters of **“tr”**: (letters to map, the patterns wanted to map to, the file to apply letter maps, the file to write results text)

HERE IS MY EXAMPLE

Graphical user interface, text, application

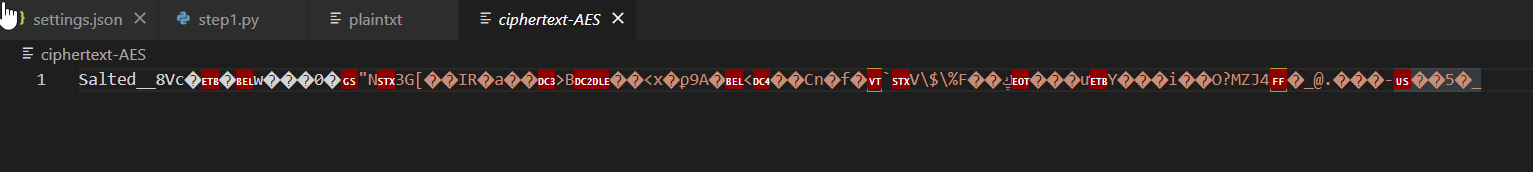
Description automatically generatedLeft-Hand-Side: Plain Text File

Right-Hand-Side: Encrypted Text File with the above letter replacement**, IT SHOULD NOT MAKE ANY SENSES!!!**

**NOTE:** Try **reverse the ciphertext** to **the plaintext** with the above **letter replacement** and see if it works!

Then type **tr 'qgvmftzyceolhsuwbjaxdnikpr' 'a-z' < ciphertext > plaintext to** decipher the ciphertext to a plaintext and see if it gives you the original file.

1. Third: **Play Around with Block Ciphers (symmetric key encryption)**
   1. Data Encryption Standard (DES)
      1. This is a symmetric key pair encryption
      2. uses 56-bit keys although a 64-bit key is fed into the algorithm.
      3. Encrypts a block of data
   2. Advanced Encryption Standard (AES)
      1. This is a symmetric key pair encryption
      2. Uses Triple-Key DES
      3. uses three different key sizes: 128, 192, and 256 bits
      4. more secured than DES
   3. To play around with AES
      1. ***use openssl enc -aes-128-ecb -e -in plaintext -out ciphertext -k 00112233445566778899AABBCCDDEEFF*** to get a binary cipher of plain text **USING 128-BIT KEYS**. **SHOULD NOT MAKE ANY SENSE!!!**



* + 1. Use ***% openssl enc -aes-128-ecb -d -in ciphertext -out plaintxt -k 00112233445566778899AABBCCDDEEFF*** to decipher the text to see if it matches the original file

1. Last: Play around with Public Key Encryption (asymmetric key encryption)
   1. Quick Introduction
      1. Asymmetric key encryption requires a public key, a private key, and a certification.
      2. Public key 🡪 known by all people, used to check for sender authority and encryption; 1024 bits
      3. Private key 🡪 Known only by the specific receiver, used to decrypt message; 1024 bits
      4. Certificate/Signature 🡪 used to verify sender identity with confidentiality
   2. Play with private/public key (RSA)
      1. Type ***openssl genrsa -aes128 -out privatekey 1024*** to create a private/public 1024-bit key pair
      2. Extract the public key from the pair using ***openssl rsa -in privatekey -pubout > publickey***
      3. **The private key should be created in a file called “privatekey”**

***Graphical user interface, text

Description automatically generated***

***Text

Description automatically generated***

***The Public Key should be extracted into a file called “Public key”***

***Text

Description automatically generated***

To see how asymmetric encryption looks like on a plaintext:

***Text

Description automatically generated***Try **openssl *rsautl -encrypt -inkey publickey -pubin -in plaintext -out ciphertext*** and ***openssl rsautl -decrypt -inkey privatekey -in ciphertext***Text

Description automatically generated

***NOTE: YOU WOULD PROBABLY NEED TO VERIFY YOU IDENTITY USING THE PRIVATE KEY TO ENCRYPT THE MESSAGE***

1. **Last: Play with Hashing**
   1. Requires no keys, outputs a message digest; only authentication needed is a signature from user.
   2. To generate hush message:
      1. Use **openssl sha256 -binary plaintext > plaintext.sha256**
      2. Then sign the hush using **xxd plaintext.sha256**
      3. Verify the signature using ***openssl rsautl -sign -inkey privatekey -in plaintext.sha256 -out plaintext.sig***
         1. ***NOTE:* hushing only works when you already have a private key.**
2. **Conclusion**
   1. **Hope you learned something on how to use the OpenSSL software. It is really interesting how authentication and network security works to secure data transfer.**
   2. **This will probably be your final lab for computer network but it is just as important as the network fundamentals.**