CE in a Connected World

Lab 3

In this Lab assignment you will get more hands-on work with crypto, and by way of doing so also familiarize yourself with two important tools: Docker containers, and the OpenSSL tool.

As part of this lab, you have 2 additional companion documents that you need to work with: The first is a Google Forms answer sheet where you can type your answers, and the second is a Google Sheets companion where you are paired up with an anonymous companion for this lab and you have to discover their identity by sending them secret messages!

Google Form answer sheet: https://forms.gle/JkSbQfQj7CtCXLuQA
Google Sheets companion information: https://bit.ly/3edSRrf

Part I: Docker

You can think of Docker as a VM platform, much like VirtualBox or VMWare, except that it is much lighter and faster than both of said applications. You will have seen in lecture how docker works, and you can find more information here: https://www.docker.com/resources/what-container

Let's set up a docker environment and experiment with it:

1. Download and install Docker: https://www.docker.com/products/docker-desktop
2. Make sure that it's installed properly, issue the following on a commandline: "docker run hello-world". You should see the output as per this page: https://hub.docker.com/_/hello-world
3. Now let's run an ubuntu instance: "docker run -it --name ec544 ubuntu /bin/bash". Note that you will be placed *inside* the container when you run this command!
4. Let's update the distro: "sudo apt-get update"
5. Let's install packages that we need: "apt-get install gcc git emacs build-essential -y"

Part II: Getting OpenSSL

OpenSSL is one of the most widely used toolkits for cryptographic and security primitives. It is an open source project. You will be downloading and compiling it within a Docker container.

Let's get OpenSSL downloaded & compiled:

1. Assuming that you are still in the docker container, let's move to your home folder: "cd ~"
2. Clone the OpenSSL git repository: "git clone https://github.com/openssl/openssl.git". This should create a **~/openssl** folder with all the code in it.
3. Cd into the openssl folder and build OpenSSL by following guidelines in the INSTALL documentation: https://github.com/openssl/openssl/blob/master/INSTALL.md
4. Once you have performed "make install", take note of the OpenSSL version through "openssl version" and make note of that in your assignment return.

Part III: Using OpenSSL to do crypto!

OpenSSL is both a stand-alone application and a library. In this lab, we are going to use it as a stand-alone application. There are many online resources available on openssl, and here is a simple book on crypto that you can use: https://en.wikibooks.org/wiki/Cryptography

Note: When copying and pasting into the Google Sheet, it is best to paste into the *formula bar* (not the excel sheet's cell). This is because many of the values you are pasting are multiple-lines of text, and pasting directly into the cell will overwrite the cells below it. By pasting into the formula bar, you effectively paste all of that information into a cell.

Also, be mindful of the "state" of your files and messages. It is easy to confuse base64 and non base64 versions of a file. Hint: the non base64 version will display garbage on the terminal, and the -a flag in openssl takes base64 inputs.

Additionally, while some openssl commands are given below, you will need to read the openssl documentation/manual to know how to conduct some of the tasks below:

Using the link provided from Project Gutenberg, take the text of Rubayat of Omar Khayyam, save it as a text file, and using openssl, create MD5, SHA128 and SHA256 hashes of the text. Here is a link to the text: http://www.gutenberg.org/cache/epub/246/pg246.txt. Make sure that when you save the text, you are not adding any extra characters to it.

1. Using openssl, create a 2048 bit RSA keypair for yourself. Post your public key on the google form excel sheet (next to the last 4 digits of your UID), post your private key on the Google Form (answer sheet). Include the "-----BEGIN/END xxx KEY-----" header and footer on both postings. (hint: use **openssl genpkey**, and **openssl rsa**)
2. Create a random 8 byte hexadecimal value that you will use as a password: **openssl rand -hex 8**. Put the password on Google Forms.
3. Using the aes-256-cbc mode, encrypt a short message to the person you are paired with on google sheet that identifies you. Use the random number as your password. Put your message in a file called plaintext so that you can easily pass it to openssl (e.g. **openssl aes-256-cbc -a -in plaintext -out ciphertext -pbkdf2 -pass pass:[from step 3 above]**). Put the ciphertext on google sheets.
4. Note that AES requires an initialization vector, but you are not providing one here, why is that?
5. Encrypt your password (shared key) from step 3 using your counterpart's public key (when they make their public key available). Put the results on the Google Sheet. (hint: use **openssl rsautl**). This generates a binary output, use the **base64** command to convert it into a base64 string (e.g. base64 encryptedkey)
6. If something has been posted for you on the "Encrypted shared secret" column, you will have to use one of the 3 keys you have already created in order to decrypt it. What did the message say, and who is your counterpart?

Some useful docker commands:

- docker run ubuntu --tag ec544 (to download ubuntu & start an instance named ec544)
- docker start ec544 (to restart it after you exit the container)
- docker exec -it ec544 /bin/bash (to re-enter it after you have restarted it)
- docker ps -a (see all containers and their states)
- docker images (see all operating system images that you have downloaded)