# CIT 285 - Lab #5: Cryptography Basics

## Lab Preparations

You will do all parts of this lab on your Kali VM. We will use the OpenSSL package to experiment with encryption and decryption using both symmetric and asymmetric (public key) encryption algorithms. We will also use OpenSSL to compute message authentication codes.

The command man openssl will provide short descriptions of how the command works. See the OpenSSL command HOWTO at <http://www.madboa.com/geek/openssl/> to obtain more details.

**We will use some fairly long command lines, so you may be tempted to use copy and paste. Don’t.** Using copy and paste between a word processor document and the command line will cause some very difficult to debug problems.

* For example, many word processors use “smart quotes” instead of actual double quotes. Smart quotes are angled double quotes that look nice, but neither the left nor the right smart quote is the same character as the ASCII double quote character used on the command line and in programming languages. These copy and paste errors can be quite difficult to debug, as the error messages will never state that copy and paste was the problem, and a wide variety of punctuation characters used in programming and command shells are affected.

Remembering the basics of command line editing will help make issuing commands faster and more accurate.

* The up and down arrows navigate through history, while the command history will show a numbered list of all past commands from this session.
* The ! command can be used to execute any command from history by number, e.g. !14 will execute the 14th command in the history list, and !! will execute the previous command.
* While editing a command line, you can use emacs style editing commands, such as
  + ctrl+a to move to the beginning of the line
  + ctrl+e to the end of line
  + ctrl+k will delete from the cursor position to the end of the line.

## 1: Encryption and Decryption

In Kali, copy the password file **from your Metasploitable 2 VM** using the commands below **to your Kali VM**. Accept the authenticity of the host (if necessary) and add the Meta IP permanently to the Kali known host file if required.

Basic file navigation commands in sftp are the same as in the UNIX shell, but you also need the get and put commands to download and upload files. The help command will display a list of available commands.

**$ sftp msfadmin@Meta\_IP**

**sftp> cd /etc**

**sftp> get passwd**

**sftp> quit**

**$ ls -l**

1.1: Encrypt the new passwd file with Advanced Encryption Standard (AES) 128-bit version using the openssl command. Use whatever password you want when prompted.

* Compare the sizes and types of the resulting files.
* Describe how they differ in your own words.

**$ openssl aes-128-cbc -salt -in passwd -out passwd.aes**

**$ ls -l passwd\***

**$ file passwd\***

**Graphical user interface, text

Description automatically generated**

**So as we can see from the screenshot we see that the size of the files differ and we can also see that when we use the file command we are not prompted with the normal ACII text but are asked to openssl with the salted password to even see it.**

## 2: Message Authentication Codes

In this part of the lab, we will use openssl to generate message authentication codes, which are also known as message digests, to check when a file has been modified.

2.1: Compute the MD5, SHA-1, and SHA-256 MACs for the passwd file. Provide the three resulting MACs in the box below and answer the following questions.

* What are the similarities between the three different MACs?
* What are the differences?

**$ openssl dgst -md5 passwd**

**$ openssl dgst -sha1 passwd**

**$ openssl dgst -sha256 passwd**

**Text

Description automatically generated**

The similarities between the three are that they are all encrypted with all these random numbers and letters, while the differences that I see is the length of them.

2.2: Copy the passwd file, then compare the MACs of the original and copied passwd files.

* Do the MACs differ in any way?

**$ cp passwd passwd.1**

**$ openssl dgst -md5 passwd.1**

**$ openssl dgst -sha1 passwd.1**

**$ openssl dgst -sha256 passwd.1**

Graphical user interface, text

Description automatically generated

**As far as I can see everything is the same. No differences.**

2.3: Edit passwd.1 and change the first character, which is an **r**, to **S**, then compute the checksums again. Provide the checksums in the box below and answer the following questions.

* What are the differences between the MACs for passwd and passwd.1?
* Are the majority of the encoded digits different?
* How are the differences between the MACs of the two files important for using MACs to secure the integrity of files?

**$ nano passwd.1**

**$ openssl dgst -md5 passwd.1**

**$ openssl dgst -sha1 passwd.1**

**$ openssl dgst -sha256 passwd.1**

**Graphical user interface, text

Description automatically generated**

**The MACs for passwd.1 are all very different now since we changed that one letter. Majority of the encoded digits and numbers are very different than the original passwd. This is a perfect example of showing how important MACs can be due to how the integrity of the files changed drastically just with one letter. The more letters, words, strings, etc changed the more random the number and letters will be.**

* What are the differences between the MACs for passwd and passwd.1?
* Are the majority of the encoded digits different?
* How are the differences between the MACs of the two files important for using MACs to secure the integrity of files?

## Submission

Upload a completed copy of this document to Canvas by the due date