**Laboratory Exercise 6 – Cyber Basics – Introduction to Cryptography Lab**

**1. Overview**

This laboratory exercise will provide some hands-on experience with symmetric and asymmetric encryption using command-line tools in Linux.

**2. Resources required**

This exercise requires “Kali Linux with Metasploitable (2020.09)” in the Cyber Range.

**3. Initial Setup**

From you Cyber Range course, select the **Kali Linux with Metasploitable (2020.09)** environment. Click “start” to start your environment and “join” to get to your Linux desktop login. If necessary, enter login credentials:

Username: **student**

Password: **student**

**4. Tasks**

**Task 1: Symmetric Encryption with ccrypt**

Ccrypt is a symmetric file and stream encryption utility for Linux and Unix that replaces the weaker **bcrypt** utility. Ccrypt uses the Rijndael cipher, which is the algorithm on which the Advanced Encryption Standard (AES) is based.

2. Ccrypt is not installed by default on your Ubuntu Linux virtual machine. Open a terminal and use the Linux package manager to install this software at the command line as follows.

**$ sudo apt-get update**

[you may be prompted for your ‘sudo password’. On you Virginia

Cyber Range Ubuntu Linux VM it is your student account password.

Probably ‘student’.]

**$ sudo apt-get install ccrypt**

To see all of the command-line options available to ccrypt, use the following command:

**$ ccrypt --help**

3. Next we need a file to encrypt. Download a text file from the Cyber Range using the command below.

**$ wget artifacts.virginiacyberrange.net/gencyber/textfile1.txt**

You can examine the contents of the file using the Linux ‘cat’ utility as follows.

**$ cat textfile1.txt**

4. Use **ccrypt** to encrypt your textfile. Ccrypt will ask for an encryption key – you can simply type a passphrase at the command line (you will use the same passphrase to decrypt the file). Be sure that you are in the directory location as your text file and encrypt it as follows. The ‘-e’ option is used to specify encryption rather than decryption.

**$ ccrypt -e textfile1.txt**

If you list your directory you should see **textfile1.txt.cpt** – the encrypted version of the file replaced the plaintext version. Use the linux ‘cat’ utility to view the file. It should be unintelligible.

**$ cat textfile1.txt.cpt**

You could now send this file to someone else and as long as they have the passphrase, they can decrypt and read it. Note that the file textfile1.txt is gone. Don’t forget your encryption key if you want this file back!

4. Use **ccrypt** with the **–d** switch to decrypt your file. Be sure to use the same passphrase as in step 3, above.

**$ ccrypt –d textfile1.txt.cpt**

Your unencrypted file should be restored to **textfile1.txt** (use ‘**cat’** to be sure).

**Question 1 (50 pts):** submit one or multiple screenshots, which demonstrate the following procedures: 1. display the content of the file, 2. encryption the file, 3. display the content of the encrypted file, 4. decryption the file, and 5. display the content of the decrypted file.

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

**Task 2: Asymmetric Encryption using Gnu Privacy Guard (gpg)**

Asymmetric encryption using Gnu Privacy Guard (gpg), an open-source implementation of Pretty-Good Privacy (pgp). Gpg is included in your Kali Linux VM so we don’t need to install anything. Below we will take basic steps to create a public/private key pair, then encrypt a file using our own public key and decrypt it using our own private key. There are lots more features and options, however. Review the man page for the gpg utility for more details.

1. First we have to create an encryption key

**$ gpg –-gen-key**

Depending on your system, you

* You may be prompted for a key type (this is used to select the encryption algorithm for the keys). The default is RSA, simply press enter to accept the default.
* Next you may be prompted for a key length. 2048 bits is the default and is generally accepted to be sufficiently long (although if you need your data to stay secret well into the future you can select 4096). Press enter to accept the default key length of 2048.
* Next you may be prompted for your key expiration. Press enter to accept the default (key never expires). When asked ‘are you sure’, enter ‘y’ and press enter.
* You will be prompted for a real name. Use any name you want, but remember it (must be at least 5 characters).
* Enter an email address, it can be a non-existent one since we are not really going to use it in the real world, but do remember what you entered.
* You can enter a passphrase for your key if you would like, or you can leave the passphrase blank and just press enter. If you enter a passphrase, be sure to remember it for later.

Once complete, you should get output listing a public key fingerprint and some other data.

2. Download the textfile.

**$ wget artifacts.virginiacyberrange.net/gencyber/textfile2.txt**

3. Now we’ll encrypt the file using our public key. Be sure you are in the same directory as your new text file and encrypt it as follows.

**$ gpg –e –r *your-email-address* textfile2.txt**

A new file will be added with a .gpg extension. Use ‘**cat’** to examine the file. It should be unreadable.

4. Use gpg to decrypt the file using your private key (delete the old file first).

**$ rm textfile2.txt**

**$ gpg -d textfile2.txt.gpg**

Your unencrypted file should be restored!

5. Now that you know that your key works for encryption and decryption, you can share your public key with others so that they can encrypt files to be decrypted with your private key. Use the following syntax to export your key to a text file.

**$ gpg --export -a *your-email-address* > public.key**

Examine the key using ‘**cat’**. The ‘**-a**’ flag has the key encoded in ascii (text). Some people append a text version of their public key to their email signatures, making is easy for others to use to encrypt files and send to them.

**$ cat public.key**

From here, you could share your public key with others at a key-signing party, upload it to a key server, or otherwise make it available for others to use to encrypt documents that only you can decrypt.

**Question 2 (50 pts)**: submit one or multiple screenshots, which demonstrate the following procedure: 1. The successful generation of the pgp key. 2. The content of the downloaded file. 3. Encryption of the file. 4. The content of the encrypted file. 5. Decryption of the file. 6. The content of the decrypted file. 7. Your complete pgp public key.

A computer screen capture

Description automatically generated with medium confidenceA computer screen capture

Description automatically generated with medium confidenceA computer screen capture

Description automatically generated with medium confidence