Decrypting and Sending a File

Assignment

1h30m - 2h

Status | Incomplete

Introduction

In this activity, you will use the GPG command to encrypt/decrypt a file using a Linux terminal and, as an optional step, develop a BASH script with a single line to help with the message.

This exercise has four parts: 1. Encrypt using AES526 2. Export the Key 3. Signing the Message (optional) 4. Decrypt the Message

Creating the File

First, create your test text file with the command `echo`.

As you work in the Linux system with the Terminal, be careful with the casing of the commands; commands in Linux are case-sensitive.

You can enter whatever you'd like here within the quotation marks. You can also name the file whatever name you'd like. For this exercise it does not matter.

```
$ echo "LightHouse Labs Cyber Security BootCamp" > file.txt
```

Exercise 1: Encrypt Using AES256

Try to encrypt the file using AES256 and verify the `encrypted.txt` file. Then decrypt the file.

Remember, in this exercise and the following exercises, 'file.txt' is the unencrypted file you create, and 'encrypted.txt' is the encrypted version of your file.

In the first command, make sure you put the file you created above at the end of this command. You named your file file.txt, so put file.txt at the end. This command will take file.txt, encrypt it and output that to encrypted.txt. Once done, try to read the file. It should look encrypted.

```
$ gpg --symmetric --batch --cipher-algo AES256 --output encrypted.txt file.txt

$ cat encrypted.txt
```

Next you want to ***decrypt*** the file. This command will decrypt the encrypted.txt file and output it to data.txt

```
$ gpg --decrypt --batch --output data.txt encrypted.txt
```

You now want to read the newly created decrypted file data.txt; this file should now be readable.

```
$ cat data.txt
```

> **TIP:** how does GPG know which cipher is needed to decrypt the file (in this case, AES256 instead of the default CAST5)? The OpenPGP symmetric key encrypted session key packet says which algorithm the GPG has to use to decrypt the file.

GPG Part 2

Following NIST's Recommendation of File Encryption/Decryption

Share the file with a fellow student, but remember to send your public key as a file to the recipient. For that, you need to export the key, and only this way the other student can decrypt your file.

Exercise 2: Export the Key

Remember the email you used to create your GPG keypair? It is most likely student@email.com. To export the key, use the command below.

```
$ gpg --armor --output mypubkey.gpg --export student@email.com
```

For your friend to be able to open the file, they need to import your key. It can be done by running the following command. You'll have to first send them your public key, we recommend sending it via Discord or email.

```
$ gpg --import mypubkey.gpg
```

If you are unsure this key belongs to your friend, verify with them. Get them to run the following command:

```
$ gpg --list-keys --keyid-format LONG --fingerprint
```

You will now encrypt the message using the sender's public key. Assuming the sender's email that is associated with the public key is your.friend@yourfriendsdomain.com, and the file you want to encrypt is called myfile.txt, run the following command:

```
$ gpg --output myfile.txt.gpg --encrypt --recipient your.friend@yourfriendsdomain.com  myfile.txt
```

Exercise 3: Signing the Message

This step is optional. The reason you, as a sender, may want to sign the message, is for the recipient to verify that you indeed sent the message and not someone else. This is a form of anti-tampering.

Instead of signing the message (which is also a form of encrypting), you shall generate a checksum of the message and sign that instead.

Generate a SHA256 sum of the unencrypted file (assuming it is named `myfile.txt`) and sign that using your private key:

```
$ shasum -a 256 myfile.txt | awk '{print $1}' >myfile.txt.sha256sum

$ gpg --output myfile.txt.sha256sum.sig --sign myfile.txt.sha256sum
```

Exercise 4: Decrypt the Message

Suppose the encrypted message sent by the sender is called `myfile.txt.gpg` and was encrypted using your public key.

To decrypt this message using your private key, run the following command:

```
$ gpg --output myfile.txt --decrypt myfile.txt.gpg
```

Suppose the signature is named `myfile.txt.sha256sum.sig`. To verify that the sender indeed sends the signature, run the following command:

```
$ gpg --verify myfile.txt.sha256sum.sig
```

Check that the public key ID and fingerprint match the sender's public key ID in your keyring.

```
$ gpg --list-keys --keyid-format LONG --fingerprint
```

This will list the public keys in your GPG keyring alongside their fingerprint.

To get the actual content from the signature, run the following command:

```
$ gpg --output myfile.txt.sha256sum  --decrypt myfile.txt.sha256sum.sig
```

Conclusion

Encryption is a crucial component of cyber security, and being able to use encryption tools such as GPG and develop BASH scripts to simplify the process of encryption and decryption is a valuable skill for any cyber security professional.

✓ Mark Completed

← Previous
The Minimum Security Strength of Cryptography

Next →
Symmetric, Asymmetric Key and Hash Quiz

How well did this activity help you to understand the content?

Let us know how we're doing

☆ ☆ ☆ ☆ ☆

W07D1

Mon Aug 5

Lectures (1)

Work (4)

3 hrs

- The Minimum Security Strength of Cryptography
- Decrypting and Sending a File
- Symmetric, Asymmetric Key and Hash Quiz
- Using TLS and SSL

W07D1 Schedule »