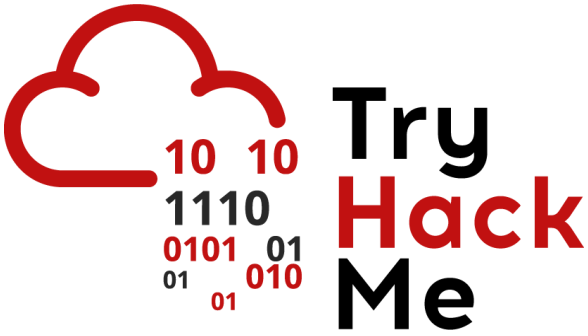

Encryption

Introduction

Encryption is a method of scrambling data into a format that only authorized individuals can understand. Unlike hashing, encrypted data can be reversed to reveal the original message, whereas hashing cannot be reversed (decrypted) to show the original data.

Before continuing, please read the following CloudFlare blog post on Encryption: https://www.cloudflare.com/learning/ssl/what-is-encryption/

Now you understand the general concept of encryption, let's attempt encrypting and decrypting some data using Asymmetric and Symmetric encryption.

Symmetric Encryption

Remember, Symmetric encryption is where you use the same key to encrypt and decrypt data.

Let's use the pre-installed program gpg to encrypt a file. Execute the command: **gpg -c data.txt** it will prompt you for a password, once complete it will create data.txt.gpg. If you attempt to view the .gpg file, you will see all of the data has been scrambled. GPG uses the AES algorithm to encrypt your files - with encryption, the algorithm used has a big part to play in the speed of encrypting and decrypting, and how secure the algorithm is (certain algorithms can be cracked easily). Advanced Encryption Standard (AES) was established by the U.S. National Institute of Standards and Technology (https://en.wikipedia.org/wiki/Advanced_Encryption_Standard).

Once you've sent your encrypted data to your friend, they're going to need to decrypt it to reveal the plaintext data. They should also know the key to decrypt the file. If they execute gpg -d **data.txt.gpg** and enter the correct key, it will decrypt the file and they will be able to view your secret message.

Data integrity is also important here, we can take a hash of the encrypted file and share this with both the sender and receiver of the encrypted message to ensure the data has not been tampered with in transit. In Linux, if you do **md5sum data.txt**, if will give you a hash value of that file. Try updating the files content and running the command again on the file, it will be different. You will notice on many downloading platforms, when you download a file, it will often include a hash; this is so you are able to check when you've downloaded the file, it has not been modified and you are in fact downloading the original file.

Asymmetric Encryption

Remember, Asymmetric encryption uses a public and private key. If you encrypt data with someone else's **public** key, it can only be decrypted using that persons **private key**. Remember, anyone can have access to your public key, but you want to keep your private key safe. If someone gets a hold of your private key, they will be able to decrypt anything that has ever been encrypted using your public key.

SSH keys use public and private keys, you generate a private key, and with that you have a public key generated (which is generated from the private key). Then you place your public key onto the server, then when you want to SSH into a machine, you use your private key to authenticate yourself as if the server can successfully decrypt your message with your public key, it knows the user has the corresponding private key.

So, if you use your **public** key to encrypt a message, it can only be decrypted with your **private** key. If you use your **private** key to encrypt a message, it can only be decrypted with your **public** key.

Let's generate some public/private keys and encrypt/decrypt a message!

To generate a private key we use the following command (8912 creates the key 8912 bits long):

openssl genrsa -aes256 -out private.key 8912

To generate a public key we use our previously generated private key:

openssl rsa -in private.key -pubout -out public.key

Lets now encrypt a file (plaintext.txt) using our public key:

openssl rsautl -encrypt -pubin -inkey public.key -in plaintext.txt -out encrypted.txt

Now, if we use our private key, we can decrypt the file and get the original message:

openssl rsautl -decrypt -inkey private.key -in encrypted.txt -out plaintext.txt