File encryption using OpenSSL and GnuPG (GPG/PGP)

Systems and Network Security

Contents

1 Notes

- Commands preceded with "$" imply that you should execute the command as a general user - not as root.
- Commands preceded with "#" imply that you should be working as root.
- Commands with more specific command lines (e.g. "RTR-GW>" or "mysql>") imply that you are executing commands on remote equipment, or within another program.

2 Exercises

2.1 Using OpenSSL and symmetric encryption to encrypt files

OpenSSL, which is present by default in the base system on Ubuntu (and most other Linux distributions, if not all), is a powerful toolkit that includes

many useful tools to generate checksums, manage certificates, and perform encryption/decryption.

In this lab, we will use the OpenSSL command to encrypt files using the AES 256 encryption algorithm, using a symmetric key.

2.1.1 Encrypting files with OpenSSL

First, let's find file we want to encrypt. We can create one. If you want to, you can write whatever you want it in, or use the example below, replacing "myname" with your name (no spaces - for example JohnDoe).

```
$ cd
$ cat > my-secrets-myname.txt <<EOF

My name is "My Name"

My credit card number is 1234-5678-9012-3456

The password for my phone is 42

EOF
```

This will create a file "my-secrets-myname.txt" containing the above text.

Now, let's ask OpenSSL to encrypt this file using a passphrase of our choosing:

```
$ openssl aes-256-cbc -e -in my-secrets-myname.txt -out my-secrets-myname.enc
```

Explanation:

aes-256-cbc The algorithm we use to encrypt. Advanced Encryption Standard -e Encrypt -in The input file -out The output file

You will be prompted to enter the passphrase - choose one that is not too short to be easily guessed, but not too long that you forget it :)

```
enter aes-256-cbc encryption password:   <-- type your passphrase here
Verifying - enter aes-256-cbc encryption password:   <-- again
```

Now check that you have both the original file and the encrypted file:

```
$ ls -l my-secrets-myname.*

-rw-rw-r-- 1 sysadm sysadm 112 Jul 30 11:43 my-secrets-myname.enc
-rw-rw-r-- 1 sysadm sysadm  80 Jul 30 11:42 my-secrets-myname.txt
```

Note the difference in the size between the original (.txt) and the encrypted copy (.enc).

What happens if you try and view the contents of the .enc file ?

```
$ more my-secrets-myname.enc

Salted__<lots of unprintable characters>
```

Notice that you cannot read the contents - it is a pure binary stream. What if you wanted to send this file in an email to one of your friends, or a family member for safekeeping ?

We can ask OpenSSL to "base64" encode the encrypted file: this will convert the binary file into a slightly larger file, using only ASCII characters which are email-friendly (or even ready to paste in a chat window!) We use the '-a' parameter of the openssl command:

Example:

```
openssl aes-256-cbc -e -a -in my-secrets-myname.txt -out my-secrets-myname.asc
```

Notice how we introduce the '-a' parameter, and we now create an encrypted file ending in .asc.

You will, once again, be asked to enter a passphrase. You can use the same as before, or a different one - just don't forget it!

Check the file sizes once again:

```
$ ls -l my-secrets-myname.*

-rw-rw-r-- 1 sysadm sysadm 155 Jul 30 12:08 my-secrets-myname.asc
-rw-rw-r-- 1 sysadm sysadm 112 Jul 30 11:43 my-secrets-myname.enc
-rw-rw-r-- 1 sysadm sysadm  80 Jul 30 11:42 my-secrets-myname.txt
```

Notice how the .asc file is the largest of the three.

What happens if you try and view the contents of the .asc file ?

```
$ more my-secrets-myname.asc

U2FsdGVkX1+QfdOIoM3qsIFRP48NUgbeHekpKPezReKTgTnrl8c/QShOuhym+NMY
[...]
1tZe8GQ/zTAimamMWJhO0A==
```

You will notice that only printable characters are used.

You could, for exmaple, now paste this information into an email, a chat session, and only reveal the passphrase to the recipient over the phone - or maybe only after a certain time.

You can now delete your unencrypted file.

```
$ rm my-secrets-myname.txt
```

2.1.2 Decrypting files with OpenSSL

Now, let's have some fun. We're going to do the folloging:

1. Copy your encrypted file to someone else's server in the class. You get do decide which server - just pick someone! ** Make sure that you have replace 'myname' with your name! ** If you ALL have called your file "myname" then they will overwrite each other's!

```
$ scp my-secrets-myname.asc sysadm@ip.of.other.group:
```

2. Once the file is copied, ask a member of the other group to decrypt your file. For this, you will need to communicate your passphrase to them. It's probably easiest to write it down on a piece of paper.
3. Someone is probably going to copy a file to YOUR server, and will ask you to do the same (decrypt the file once they have given you the passphrase). Remember to use 'ls -l' to verify that they have copied their file to your server!

To decrypt the file, use the following command:

```
$ openssl aes-256-cbc -d -a -in my-secrets-theirname.asc -out my-secrets-theirname.txt
```

You will be prompted for a passphrase.

- Has this worked ?
- If not, what happened ? Did you remember '-a' ? Otherwise OpenSSL cannot know that the file is encoded in ASCII!
- What does openssl say if you use the wrong passphrase ?

If it has worked, OpenSSL will not complain, and you will be able to read the decrypted file:

```
$ ls -l my-secrets-theirname.txt

-rw-rw-r-- 1 sysadm sysadm 102 Jul 30 12:45 my-secrets-theirname.txt
```

- Can you view the contents ?
- What did you observe ?
- What are the advantages of using symmetric keys ?
- What are the inconvenients ?