Bachelor's Degree in Techniques for Software Development

Networks and Internet Applications
Practical exercise 4. How can I avoid intruders or attacks?

This practice has four parts:

- Part 1: Integrity with SHA.
- Part 2: Symmetric encryption with GPG.
- Part 3: Asymmetric encryption with GPG.
- Part 4: Apache secure configuration, client connection and SSL explanation.

Qualification

The qualification depends on how many parts are delivered:

- Part 1: allows you to opt for a maximum grade in this practice of C-.
- Parts 1 and 2: allows you to opt for a maximum grade in this practice of C+.
- Parts 1, 2, and 3: allows you to opt for a maximum grade in this practice of B.
- Parts 1, 2, 3 and 4: allows you to opt for a maximum grade in this practice of A.

Introduction

In this practice, you will work with **integrity, confidentiality, authenticity and non-repudiation** concepts, together with symmetric and asymmetric encryption, basic for computer security. We are going to use different tools included with Ubuntu, which means that you will develop the practice over a Linux server (you can choose the one you prefer) and a Linux or Windows client. You can use a virtual machine or an installation in a real machine.

For all exercises where the execution of some command is requested or send an e-mail, **copy a screenshot** where one can see the command and the response.

For the screenshots requested, **change the system prompt in order it shows the current date and your username**.

During the practice, the execution of some commands is requested. **It is required to include in the response the command used, together with the meaning of each parameter used**.

Send only your responses, including the corresponding screenshots, to the continuous evaluation registry. **You do not need to include any part of this document nor the questions.**

First Part (maximum qualification C-): integrity

According to Wikipedia, SHA (*Secure Hash Algorithm*) is a family of encryption *hash* functions. SHA is extensively used in software programming to provide the security that a file downloaded from the Internet has not been modified. Comparing the published *hash* with the checksum of the downloaded file, a user may be confident enough that the file is the same as the one published by developers. In this way, it is a tool that allows us to check the downloads' integrity, although we can also use it with files we may have in our own computer.

Exercise 1

a) Download a file from the Internet, which has its SHA published. Using the proper Ubuntu command, check that the SHA from the download is the same as the one published. Include a screenshot that shows that they are the same.

b) Search information related to the different functions that make up the SHA family. Which length have the generated checksums?

c) Which problem has the usage of SHA-0? And SHA-1?

d) Create a text file and calculate its hash with a function of the SHA-2 family. Modify the file and check that it is not the same (screenshot).

e) Is it possible to obtain the original text from its *hash*?

Second Part (maximum qualification C+): symmetric encryption

Pretty Good Privacy (PGP) is a software developed by Phil Zimmermann with the objective of protecting information shared through the Internet and facilitating authentication. Thanks to its success, the IETF used PGP as a basis for defining OpenPGP standard (defined in IETF RFC4880), facilitating the proper encrypted communication between different programs that implement the same protocol. GNU Privacy Guard (GnuPG) is a free and complete implementation of OpenPGP. This tool allows encryption, as well as signing information and communications, together with the management of the keys needed to do so.

In this second part of the practice, we will use the tool GnuPG to encrypt and decrypt messages (plain text files) using symmetric encryption. GnuPG is usually installed in most GNU/Linux distributions. To check it, use the command:

```
gpg --version
```

In case the command does not exist, install it using:

```
sudo apt-get install gnupg
```

Exercise 2

a) In symmetric cryptography all security relays in the key, so it has to be very difficult to break. Describe two features that make a key difficult to break.

b) Mention three currently used symmetric encryption algorithms and search which is the key length.

c) Use `gpg` command to find out which symmetric encryption algorithms supports.

d) Use `man gpg` to know which is the option for symmetric encryption. Create a text file

with some content and encrypt it. Try to see it with `cat` or `more`.

e) Which option is used to decrypt a file? Do it.

f) Which algorithm is used by GPG by default? Use the proper option to encrypt a text file with algorithm CAMELLIA256. Include a screenshot showing both the command used and all the options that appear on the screen until the key is generated, and explains both the parameters used in the call and all the options that appear.

Third Part (maximum qualification B): asymmetric encryption

In this third part of the practice, you have to create a pair of public and private keys using GnuPG. Then, you have to use them to encrypt and decrypt a file asymmetrically.

Exercise 3. Generate keys

a) Check using `gpg` command if you have any private/public key in your system.

b) Generate a new key pair with the `gpg` command.

c) Check that they have been created and briefly mention its characteristics.

Exercise 4. Send public key

Now, you can send the public key to another classmate through e-mail. The most common option is to upload the public key to a key server (https://www.rediris.es/keyserver/index.html.es) but we will send it by e-mail to simplify: Contact with some other classmate through the forum.

a) Export your public key to a file.

b) Send your public key to your classmate by e-mail.

Exercise 5. Import public keys

a) Import the public key you have received from your classmate.

b) Check that the key has been imported into your key ring.

c) Create a file called `<UOC username>_exer5_archive.txt` and encrypt it with the key you have just imported.

d) Send the file by e-mail to your classmate, encrypted with its public key.

Exercise 6. Decrypt the file you have received

a) Check that you cannot see the contents of the file you have received.

b) Decrypt the file.

c) Check that you can see the contents of the file you have received that was encrypted with your public key.

Exercise7. Other GPG options

a) Show the command to delete a key from your key ring.

b) Show the command to sign a message, without encrypting it.

c) Does it make sense to sign a message that is not encrypted? Justify your answer.

Fourth Part (maximum qualification A): Apache secure configuration, client connection and SSL/TLS explanation

Apache, available both for Linux and Windows, is the second HTTP server most used nowadays (https://w3techs.com/). It is designed in a modular way, which means that, apart from the server, there are different modules that allow adding several features.

In this part, we are going to enable Apache SSL module that offers support for SSL v3 and TLS v1.x, and we are going to configure HTTPS to have a secure web server. Afterwards, we will connect to the server with several clients. First of all, we will use the tool `openssl` in client mode, and then we will use a browser.

Enable `mod_ssl` module

Apache2 `mod_ssl` module provides the possibility of encrypting data using SSL (*Secure Socket Layer*). When we want to connect to a website with these features, we have to use the prefix `https://` in the URL, in the browser address bar. SSL is based on Public Key Cryptography (PKI).

`mod_ssl` module is installed by default in Apache2. To enable it, use the command `a2enmod` (*Apache2 Enable Module*). The following command should enable `mod_ssl` module:

```
sudo a2enmod ssl
```

To apply changes, you have to reload Apache2 configuration or restart with command:

```
sudo service apache2 restart
```

Note - To install any Apache2 module you have to use the `apt-get` tool.

Exercise 8. HTTPS configuration

Apache includes a self-signed certificate that we may use for our websites, as well as a configuration file for SSL called `default-ssl`.

a) Install Apache using `apt` and check that it works for HTTP.

b) Change the default page and make it show the message "<Your name and surname> server works!"

c) Locate in file `/etc/apache2/sites-available/default-ssl` the directives related to SSL and explain their meaning.

d) Enable HTTPS by writing:

```
sudo a2ensite default-ssl
```

e) Check with command `netstat -ntl` that HTTPS is enabled.

Exercise 9

Connect to your server using the following command:

```
openssl s_client -connect 127.0.0.1:443
```

Explain **in detail** the meaning of the following fields inside `SSL-Session`:

- Protocol
- Cipher
- Session-ID
- TLS session ticket

Exercise 10

Make an SSL connection to your server with your browser while you capture network traffic with Wireshark (use VirtualBox bridge mode):

a) Copy a screenshot where it is shown that your browser warns you when connecting to a server with a self-signed certificate.

b) Explain which algorithms have been used in the connection.

c) Identify and explain the three steps of the *SSL handshake* in the capture.

d) What guarantees does a self-signed certificate offer (and what does it not)?

References

- https://www.openpgp.org/
- https://www.gnupg.org/
- https://www.apache.org/