UNIVERSITÉ TOULOUSE III PAUL SABATIER

2018/2019

Practicals°4: hash, steganography, cryptography …

1. introduction

During this lab, we will work with the Debian operating system under Raspberry Pi (you can also use your linux or Bash system under Windows 10) and we will discuss some elements and tools of the security: data integrity, attack by brute force, encryption and steganography.

2. MD5 / SHA-1 footprint

In cryptography, MD5 (Message Digest algorithm 5 defined by Ronald Rivest in 1991 at MIT) and SHA-1 (Secure Hash Algorithm 1 defined by the NSA in 1993) are hash functions frequently used to check the integrity of files.

Became an internet standard (http://www.ietf.org/rfc/rfc1321.txt), MD5 is used in many security applications and can quickly ensure the integrity of files (useful when downloading). So, by comparing the md5 fingerprint of the file you just picked up to the fingerprint that the download site gives you, you can make sure you have the same file as the original.

Preamble

`md5sum` is a fingerprinting tool already present on linux systems.

Then install **`7zip`** on your machine by typing **`sudo apt-get install p7zip`**

Exercises

1. Download `canal.jpg` and `pigeonnier.jpg` image files at `https://github.com/truillet/ups/tree/master/m2issd/ressources` (`wget command https://github.com/truillet/ups/blob/master/m2issd/ressources/canal.jpg` for example)
2. Compare the received files with the MD5 fingerprints given on the same site (file `md5sum.txt`)
3. What do you conclude?
4. What would you do?

3. GPG

GnuPG (GNU Privacy Guard) is the GNU's Not Unix (GNU) implementation of the OpenPGP standard (public / private key cryptography format) standardized by RFC 4880 and available in Linux distributions. GPG allows you to sign and encrypt any digital document.

We will start by generating the public / private key pair using the command `gpg --gen-key`

Choose successively the following values: (your name) / (your address) / (if applicable)) / (your passphrase) / (your passphrase again) / (do things to generate entropy) then wait ...

Check that the keys have been created with the command `gpg -list-keys.` The `gpg --fingerprint` command will allow us to find the fingerprint of the public key

It will now be necessary to store on a server our public key to be able to receive eg encrypted e-mails.

Let's first create a revocation certificate (just in case) with the command `gpg --gen-revoke (your e-mail address)> revoc_ (your e-mail address) .txt`

Answer the asked questions. Then send your public key to the server (eg pgp.mit.edu) using:

```
gpg --keyserver pgp.mit.edu --send-keys
```

We will now encrypt our first file. Open a document (for example using `nano mondoc.txt`), write text and save.

To encrypt, nothing simpler than use the command: `gpg -er (your email address) mondoc.txt` (**-e** means you will encrypt the file and **-r** you will use the public key specified after)

Normally, you should have a `mondoc.txt.gpg` file that appeared

To decipher a text, you must have the secret key which is good because you have it! To decipher, just type `gpg mondoc.txt.gpg`

- Encrypt a text and send it to somebody.
- What to do to make it work and the person can decipher it?
- To decipher a text, you must have the secret key which is good because you have it! To decipher, just type `gpg mondoc.txt.gpg`
- What is the size of the key used par RSA?

4. steganography

Steganography is "*the art of dissimulation*". This consists of hiding a message / file in a trivial object. There are many examples of use of this technique since antiquity (tattoo on a skull, invisible ink, microdot, ...)

Computer science makes it possible to refine this technique by adding either files of innocuous appearance to another file (use for example of the LSB technique - **L**east **S**ignificant **B**it)

Exercise

Let's start by installing steghide by typing: **`sudo apt-get install steghide`**

Download the `canal_midi.jpg` image files from
`https://github.com/truillet/ups/tree/master/m2issd/ressources`

You have also learned that the passphrase used is "***channel***". Check that the photo contains hidden content using `steghide info canal_midi.jpg`

Then extract the contents by typing: `steghide extract -sf canal_midi.jpg`

5. attacking passwords by brute force

We will now try to crack a password by brute force (try all the possible combinations). There is a lot of software available to test all combinations, combining the use of processor power like that of the graphics card (GPU).

Download `perdu.zip` file at

`https://github.com/truillet/ups/tree/master/m2issd/ressources`.

- By "luck", you also learned that the password is at most composed of 5 elements and only composed of characters in lowercase ... Find it! 😊 ... Finally, check the content!
- Estimate the maximum number of attempts to crack the password. Estimate this number of tests if using capital letters. What do you conclude?
- To try to decrypt the password, let's install the `fcrackzip` software (decryption tool): `sudo apt-get install fcrackzip`
- Then decode the file using: `fcrackzip -u -c a -p aaaaa file.zip`

-u allows to try the password found with unzip

-c a indicates that you will use all lowercase characters to try to decrypt the password

-p aaaaa indicates that you want to decode up to 5 characters

Note: If you want to protect your zip file, use the command: `zip -e file.zip files_to_zip`

You will definitely need to install zip using: `sudo apt-get install zip`

7. Global exercice

Knowing that to encode a file, use the command `steghide embed -cf file.jpg -ef file_to_hide`,

- Create a zip file containing anything protected by a 5-letter password.
- Create a second zip file protected by another password (you decide which one with the size of your choice) contains your public key.
- Hide your zip file in a jpeg image using the first password
- Share two jpeg images, one of which contains the hidden message and the first zip file
- Get work from one of your colleagues and try to find the shared public key!
- Code a secret message with the public key found and send it to your colleague
- Get the coded message and decode it ...

5. codage RSA (Rivest, Shamir, Adleman, 1977)

RSA (named after its inventors and so-called asymmetric cryptography coding (public key and private key) RSA ensures confidentiality (only the owner of the private key can decrypt the received message) and non-alteration and non-repudiation Only the owner of the private key can sign a message A decrypted signature with the public key can prove the authenticity of the message. The algorithm is public, its difficulty lies in the difficulty to factorize a large number.

A RSA key of 2048 bits could be cracked in 2016 but the ANSSI always recommends the use of RSA keys of 2048 bits (`https://www.ssi.gouv.fr/guide/cryptographie-les-regles-du-rgs`).

Operation

Bob has a confidential message that he wants to send to Alice. Alice builds two keys:

- A **public** encryption key that it passes to Bob.
- a **private** decryption key which it keeps carefully.

Bob uses the public key to encrypt the message, and passes it to Alice. Alice uses the private key to decrypt the received message.

Key generation

Define two large prime numbers "*randomly*" chosen: **p** and **q**.

Let **n = p.q** and **phi (n) = (p-1). (q-1)**

Define a large integer "*randomly*" chosen, prime with **phi (n)** and **e** the inverse of **d modulo phi (n).**
(i.e. **e*d = 1 mod phi (n)**)
The public encryption key is the pair (**n, e**), the private decryption key, the pair (**n, d**).

Exercise

René uses RSA and publishes his public key (**n = 187, e = 3**)

1. Encode the following message (ASCII): BOB with René's public key (B has ASCII code 66)
   Reminder: encrypt character(x) → $C = (x)^e \bmod n$
2. Using the fact that the Euler totient function **phi (n) = (p-1). (q-1)** is equal to **160**, find the prime factors **p** and **q** (allowing the calculus of the secret key of René)
3. What do you conclude about the algorithm in general?

8. Going further

- Kali Linux 2019.1, « *The most advanced penetration testing distribution* » (**https://www.kali.org/downloads)**
- Mimikatz, **http://blog.gentilkiwi.com/mimikatz**

Mimikatz was developed as a proof of concept to show Microsoft that their authentication protocols were vulnerable to attack. Mimikatz is an open-source application that allows users to view and save authentication credentials like Kerberos tickets. Attackers commonly use Mimikatz to steal credentials and escalate privileges: in most cases, endpoint protection software and anti-virus systems will detect and delete it. Conversely, pentesters use Mimikatz to detect and exploit vulnerabilities in your networks so you can fix them.
When you run Mimikatz with the executable, you get a Mimikatz console in interactive mode where you can run commands in real time. Run Mimikatz as Administrator: Mimikatz needs to be "Run as Admin" to function completely, even if you are using an Administrator account.
First, run the command:

```
mimikatz # privilege::debug
```

The output will show if you have appropriate permissions to continue.

And finally, output all of the clear text passwords stored on this computer.

```
mimikatz # sekurlsa::logonpasswords
```

Mimikatz is also composed of several modules that you can use (crypto, Kerberos, misc, …).