ISE - System security

Practical session

The goal of this practical session is to familiarize with the notion we saw in class, especially on *authentication* and *access control*. The following exercises should be done in-order, since completing each step will enable you to reach the next!

You are expected to provide a report on moodle, containing :

- ☐ a PDF document with answers with a proper justification (2 to 5 pages expected)
- ☐ a test file (extension .sh) containing the commands you used to solve the problems (you can choose to make multiple files, or a single file, but use comments to identify which commands correspond to which exercise).

You are free to write the report in English or French, as you prefer. **You can work in pairs (and send one report for two students), but everyone should do the exercise.**

Prerequisites

Before starting the exercises, make sure you prepared your work environment with all the resources you need:

- ☐ create the directory `~/Documents/ISE/TP_system/`,
- ☐ get the archive available on Moodle, and uncompress it. It should contain:
  - a file `rockyou.txt`, which is a dictionary of passwords that where found in various data breaches,
  - a file `hash.txt`, which contains 50'000 hashes from passwords leaked in previous data breaches*,
  - a file `passwords_dump.txt` that contains the hashed passwords of users from a server.
- ☐ download hashcat, uncompress the archive and rename the folder "hashcat". This archive contains a standalone version of the tool `hashcat`, which can be used to "crack" password from their hash, using multiple approaches.
- ☐ Open a terminal and navigate to `~/Documents/ISE/TP_system/`.

In this session, we assume that you extracted all resources in the folder `~/Documents/ISE/TP_system/`.

Cheat Sheet

Here are a few useful commands and shortcuts that should prove useful for these exercises.

- `man <command>` Gives you the manual page of a command. This should be your first reflex when encountering a new command, or when you don't know how to use it.
- `chown` Change the ownership of a file or a folder. Use `-R` to make it recursive (so the change affect all sub-directory and files).
  For instance: `$ chown -R userA:groupA directory`

*This database is actually extracted from a real-world database leakage (see https://haveibeenpwned.com/Passwords)

- `chmod` Change the access right on a file or directory.
  For instance: `$ chmod 755 file` will give all right to the user, and read/execute rights to both group and others
- `cat` displays the content of a file in the terminal.
- `nano` Open an simple editor in the terminal.
  There are many tools that can be used. Here, I recommend using a simple and intuitive editor, `nano`. You can simply navigate using the arrow keys, edit on the fly, save using `Ctrl+O`, and exit using `Ctrl+X`. Other editors, such as `vim` or `emacs` are much more powerful, but require some time to learn the shortcuts.
- Shortcut: `Ctrl+D` logs you out. It's very convenient to get back to a previous shell after changing user with `su`
- Shortcut: `Ctrl+L` clears your shell
- Shortcut: `Ctrl+R` search in your command line history. This is very useful when you only remember part of the command line, or don't want to scroll the history for a command you typed a long time ago.

1 Recovering your password

In this exercise, you will familiarize with the tool `hashcat`, and learn some methods to recover weak passwords from their hash value (also called *digest*).

To get started, your mission is simple: recover as many passwords as possible from a sample of leaked hash. The file can be found on Moodle, it is named `hash.txt`. All passwords were stored hashed using SHA-1, without salt. This is commonly names ”raw SHA-1”.

▶ **Question 1.** First, you will familiarize with the tool `hashcat`. This tool is made to be generic, and supports various way of recovering passwords from their hash. Start by taking a look at the documentation of `hashcat`. Which option should you use to indicate that you are attacking SHA-1 hashes?

▶ **Question 2.** Before trying to recover your password to access the server, let's try a few approaches available on `hashcat`. For these questions, the goal is to recover as many passwords as possible from the hash of the file `hash.txt`

▶ **2.a.** The most straightforward approach, known as *bruteforce*, is to try to hash every possible letters/numbers/symbol combinations, and see if the output matches one the hash in the list. You can run it by simply giving the file containing the digests to the tool, with the option specifying the hash format (see previous question).
Let it run for 2 minutes before stopping it (`Ctrl+C`) even if you didn't find any password. How many password(s) did you find? How can you explain this result?

▶ **2.b.** Now, let's try a more efficient approach: using a dictionary as input. The idea is, instead of naively bruteforce the passwords, to get a list of potential passwords, and see if any of the passwords on that list matches one of our digest once processed with SHA-1. You can do this using `hashcat` by giving the path to the dictionary file(s) after the file containing the hash. Here, you should use the `rockyou.txt` dictionary.
How long did it take? How many password(s) did you find?
When hashcat finds a password, it remembers it, and stores it in hashcat/hashcat.potfile. You can display the cracked passwords using the `--show` option.

▶ **2.c.** The dictionary approach is still limited. `hashcat` enable users to create elaborate rules to try *variations* and/or *combination* of the dictionary's entries. While the creation of such rule is out of the scope of this class, some rules are shipped with the tool, in `hashcat/rules/`. Try to use some of them to recover even more passwords in the `hash.txt` file using the rockyou dictionary.
You can select the rule to use with the option ”`-r <path to rule file>`”. Try to run the rules `combinator.rule` and `rockyou-3000.rule` (stop it after a few minutes to avoid wasting too much time). Compare the number of password you find with these two rules, and the previous approaches.

▶ **2.d.** Compare these approaches. Which one is the more powerful?

▶ **2.e.** Do you think you can recover any password using these approaches? Why?

▶ **2.f.** Based on how you cracked the passwords, what recommendations can you make on how to choose a secure password?

Each student has an account on a distant server, reachable at the URL `ise.istic.univ-rennes1.fr`, but you can only connect to it using SSH, a secure protocol. You can connect to this server using the following command line in a terminal:

```
ssh <USERNAME>@ise.istic.univ-rennes1.fr
```

Only one problem: you don't know the password! You only know that the file `/etc/shadow`, containing all usernames, and corresponding hashed passwords, has been leaked. It is available on moodle under the name `password_dump.txt`.

> **Note:** The file `/etc/shadow`, on UNIX systems, contains very sensitive information, including username and hashed passwords. Here, the passwords are stored insecurely on purpose for the exercise. Modern system use secure Key Derivation Functions (`scrypt`) with a different random salt for each password.

▶ **Question 3.** To continue to the next exercise, you must recover your password and connect to the server using the command line above. There is one good news, and one bad news for you:

- All password where hashed for storage, so you can't simply read them
- From the format of the stored password, you know that they are not hashed securely... They are hashed using MD5 (an old and insecure function), and with no salt: this is called "raw MD5".

Using `hashcat`, recover your password. In the report, explain what you tried to recover the password, and the password you found for your account.
Hint: you may need to extract the hash of your password from the file, and use some rules to recover the password.

Congratulations, you should have remote access to your account on the server! First things first, let's secure your account!

> Before proceeding to the next exercise, make sure you can indeed connect to the server. Contact the professor if you can't.

2 Secure your account credentials

As stated in the previous exercise, your password were leaked! You have to change it, and it would be a good idea to add a layer of security... maybe a second authentication factor?

▶ **Question 1.** What are the benefits of using a second authentication factor? What kind of factor can you add?

▶ **Question 2.** In this exercise, you will set up a secure authentication with 2 different factors: a password and a temporary code generated by an application (Google Authenticator). For each of them, answer the following question: is it based on something you know, something you are, or something you own?

> From here, everything should be done on the remote server, using the SSH connection you established at the end of the last exercise.

▶ **Question 3.** Using the `passwd` command, change your password for a secure one. Describe your though process to choose a secure password.

> **Note:** This is out of scope of this exercise, but we usually prefer to disable the password authentication, and prefer to rely on a solution based on *asymmetric cryptography.* This implies to generate some key, deploy them properly, and configure the SSH service to prefer this method.

▶ **Question 4.** Now, let's add a second authentication factor. Here, we will be using `Google Authenticator`, which is based on Temporary One Time Password (TOTP).

▶ **4.a.** Install the google-authenticator package using the following command:
```
sudo apk add google-authenticator
```

▶ **4.b.** Configure the server to use `google-authenticator` to provide a second authentication factor.

- enter the command `google-authenticator`, and follow the instructions.
- it will generate a secret code and/or a QR code: this must be used to set up the client (application that generate the codes): a smartphone with the authenticator app, or a browser extension (Authenticator).
- on your computer (not on the server), open Firefox, install the extension, and add the code. You will get a code that changes every 30 seconds. You must copy/paste it to the server command line to continue the installation

You can choose to answer the question asked by the installer however you want, but in your report, you must justify the choices you made.

▶ **4.c.** Now, you need to tell the server that it must also use google-authenticator, and not only the password to authenticate users. To do so, modify the file `/etc/pam.d/sshd` (you must use `sudo`) to add the following line:

```
auth required pam_google_authenticator.so
```

after the line

```
auth include base-auth
```

This tells the system that it must authenticate users using the "base" process (password), **and** then use the google authenticator as a second factor.

▶ **4.d.** Finally, since we are using ssh, we have to modify the SSH configuration to make sure it can interact with google authenticator.
To do so, open the file `/etc/ssh/sshd_config`, and add the following lines at the end:

```
ChallengeResponseAuthentication yes
PasswordAuthentication yes
AuthenticationMethods keyboard-interactive:pam
UsePAM yes
```

▶ **4.e.** You can now restart the SSH service on the remote server, so it reloads the new configuration. To do so, use the following command:

```
sudo kill -HUP $(pidof sshd.pam)
```

You can check that it worked by typing the command

```
pidof sshd.pam
```

It should return a number. If it is not the case, you likely made a mistake in the `/etc/ssh/sshd_config` file.

> **Note:** This is an unusual way to restart a service. It is due to the particular setup used for this practical session.

> Don't close this terminal/session before you confirmed that everything is working as inted, or you may not be able to log back in! If something went wrong and you can't log back in, carefully check that you don't have any typo or missing letter in the two previously edited file !

▶ **Question 5.** On a different terminal, reconnect to the server using SSH. It should ask for your password **and** a temporary code.

3 Access Control

Great, know that you secured your connection, let's look at what you have on this server! You are working in a multi-user Linux environment and your home directory contains different files and folders, each with specific access permissions. Some files belong to shared groups. Your task is to analyze the current permissions, understand potential security risks, and modify access control to secure your data.

▶ **Question 1.** Start by taking a look at the files and folders in your directory using the command `ls -l`
Pay specific attention to:

- `files/` (regular files)
- `sensitive/` (admin-controlled files)
- `highly_sensitive/` (very restricted access, only you should access it)

For each file and directory, give a description of the access rights. Identify users and groups that can access it, and what they can do.

▶ **Question 2.** Let's take a closer look at the `highly_sensitive` folder. You should be the only one able to access it.

▶ **2.a.** List all users who can read the file `secret_plans.txt`. Does it seem reasonable to you?
Hint: you may need to look at the file `/etc/group`.

▶ **2.b.** Suggest two ways to make the access to this file more restricted, using `chmod` and `chown` respectively.

▶ **2.c.** Can you read the content of the file `flag` with `cat` (without using `sudo`)? Why?

▶ **2.d.** Find a way to recover the content of the file without using `sudo`. You must explain how you did it, and put the content of the file in your report.