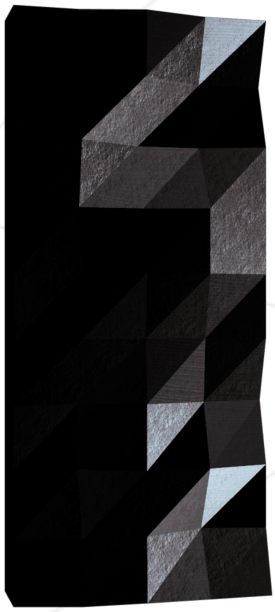

Cybersecurity Bootcamp | 42 Madrid

ft_otp

Summary: Nothing ever lasts forever...

Version: 1

Contents

Chapter I

Introduction

Passwords are `one of the biggest headaches in computer security` Users forget them, share them, reuse them and choose them horribly bad. Furthermore, passwords are sooner or later leaked in security breaches. One way to avoid this is to use one-time passwords, based on timestamps, which expire after a few minutes and then become invalid. Whether you already use this system, or if you have never heard of it, it is quite likely that one of your passwords has been compromised at some point in your life.

In this project, the aim is `to implement a TOTP` (Time-based One-Time Password) system, which will be capable of generating ephemeral passwords from a master key.
It will be based on the `RFC`: https://datatracker.ietf.org/doc/html/rfc6238, so you could use it in your day to day.

Chapter II

Prologue

The Silk Road stretched across the entire Asian continent, connecting China with Mongolia, Persia, India, the Middle East, Turkey, Europe, and Africa. Despite the name, it was not the valuable cloth that was primarily traded. Glass, leather, weapons or war machines traveled the world, expanding industrial discoveries, printing techniques, gunpowder or the compass.

Password, please?

Chapter III

Mandatory Part

In the language of your choice, you must implement a program that allows you to register an initial password, and is capable of generating a new password each time it is requested. You can use any library that facilitates the implementation of the algorithm, as long as they do not do the dirty work, that is, it is strictly forbidden to use any TOTP library. Of course, you can and should make use of some library or function that allows you to access system time.

An example of the use of the program would be:

- The program should be called `ft_otp`
- With the `-g` option, the program will receive as an argument a hexadecimal key of at least 64 characters. The program will safely store this key in a file called `ft_otp.key`, which will be encrypted.
- With the `-k` option, the program will generate a new temporary password and print it to standard output.

```
$ echo -n "NEVER GONNA GIVE YOU UP" > key.txt
$ ./ft_otp -g key.txt
./ft_otp: error: key must be 64 hexadecimal characters.
$ xxd -p key.txt > key.hex
$ cat key.hex | wc -c
64
$ ./ft_otp -g key.hex
Key was successfully saved in ft_otp.key.
$ ./ft_otp -k ft_otp.key
836492
$ sleep 60
$ ./ft_otp -k ft_otp.key
123518
```

You can check if your program is working properly by comparing generated passwords with `Oathtool` or any tool of your choice.

```
oathtool -totp $(cat key.hex)
```

Chapter IV

Bonus Part

The evaluation of the bonuses will be done `IF AND ONLY IF` the mandatory part is `PERFECT`. Otherwise, the bonuses will be totally `IGNORED`.

You can enhance your project with the following features:

- Allow to choose the encryption password of the master key ft_otp.key and request it every time a new password is generated.
- Develop a client that generates the master password and validates the results with a graphical interface.
- Any other features you think are genuinely useful. Your peers will judge if they are so.

Chapter V

Peer evaluation

This project will be corrected by other students. Turn in the files inside the Git repository and make sure everything works as expected.