SSD & WS
Homework : rainbow attack

R. Absil

2022 - 2023

The objective of this group (3-5 students) homework is to implement an attack on password tables with a rainbow table. The deadline is set on October 9 at 23h59.

Minimal objectives

From a table of passwords stored as pairs "(login,hash)" with the help of some cryptographic function H, you must implement a rainbow attack.

For academic reasons (mainly simplicity),

- passwords are *not* salted,
- passwords are stored after a single pass through the hash function,
- passwords are alphanumeric with length[1] at least 6 and at most 10,
- the hash function H is SHA-256.

The choice of language is left to your discretion (but that choice is your responsibility). Should you use custom libraries, their code *must* be open-source.

Please note that you must at least submit two scripts and one text file :

- a preprocessing script allowing to generate a "sufficiently large"[2] rainbow table RT,
- an attack script allowing to exploit RT in order to find passwords from their hashes.

[1] You are allowed to build a rainbow table per password length, for simplicity reasons.
[2] The user can decide what is "sufficiently large".

In *no way* you are allowed to submit the rainbow table RT, which can be quite large.

For the sake of uniformity, your attack script must allow to input hashes stored in a text file, one hash per line, written as a base-16 string of length 64.

Should you find it useful, two scripts are provided for you:

- `gen-passwd`, generating passwords accepted by the policy, storing them in one text file, and their hashes in another file,
- `check-passwd`, checking whether passwords stored in one file match hashes stored in another file.

You can compile them using the commands

```
g++ -o gen-passwd -std=XXX random.hpp sha256.cpp gen-passwd.cpp passwd-utils.hpp
g++ -o check-passwd -std=XXX random.hpp sha256.cpp check-passwd.cpp passwd-utils.hpp
```

where `XXX` is assumed to refer at least `c++17`. Running these programs without command line arguments will provide further information about how to use them. You will also find an implementation of a thread pool should you find it useful[3].

You will also find an open source `C++` implementation of SHA-256. A `main` file also shows how to use this implementation.

Submission modalities

Projects have to be implemented in groups of 3 to 5 students, and submitted with the help of a gitlab[4] repository[5]. For that purpose, send me an email on October 2 at 23h59 at the latest with the ssh URL[6] to your repository[7], and the name and matricule of your group members.

You have to submit your work on October 9 at 23h59 at the latest. The minimal requirements for submitted projects are as follows:

- projects have to be submitted on time,
- projects have to provide a `README` file
  - mentioning the name and matricule of your group members,
  - explaining how to build[8] your project on a ubuntu 22.04 distribution (we recommend here to either provide a makefile, or a shell script to install missing dependencies, compile the project and run relevant scripts),

[3] It is unlikely that you will meet the efficiency requirements detailed later without multithreading.
[4] That is, not a github repository.
[5] Create the repository yourself, add me (`rabsil`) as maintainer.
[6] A gitlab ssh URL looks like `git@gitlab.com:username/projectname.git` .
[7] The automatic email notification is *not* enough.
[8] It is expected that your script installs missing dependencies in addition to compiling your code.

– explaining how to use your project (for example, “to launch the attack, type the following command in a shell”).

Projects failing to meet these requirements will not be graded (that is, they will get 0/20). In particular, projects that do not compile according to your *exact* instructions will not be graded. Furthermore, note that we shall in *no way* build or run your projects in an IDE.

Note that these above conditions are necessary but clearly not sufficient to get 10/20. To increase your chances of successfully complete this homework, I would strongly advise

- to be able to generate a sufficiently large rainbow table under one night of user time on a laptop[9],
- not to generate a rainbow table bigger than 12 GB,
- to be able to successfully crack 50% of a set of hashes ($\simeq$ 100) provided as a text file[10] under 45min of CPU time on a laptop[9].

It is forbidden to cry and forbidden to laugh.

[9] That is, you cannot reasonably assume I have a computing cluster at my disposal, nor that my machine will behave fairly if you load computations on the GPU.

[10] Recall that passwords are alphanumeric (lower and upper case) with length at least 6 and at most 10, are stored unsalted after a single pass to the SHA-256 hash function.