Homework : rainbow attack
Secure software development and web security

R. Absil

2019 - 2020

The objective of this homework is to implement an attack on password tables with a rainbow table. The deadline is set on 19 April at 23h55.

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 at least 6 and at most 8,
- 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" [1] rainbow table RT,
- an attack script allowing to exploit RT in order to find passwords from their hashes.

In *no way* you have 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.

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

1. The user can decide what is "sufficiently large".

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

Grade

To get a grade of 10/20, you must at least
- submit your project *on time* as a `.zip`, `.tar`, `.gz` or `.7z` file format,
- at least cover the minimal objectives,
- provide a `makefile` or a shell script in order to build your project[2],
- provide a `readme` file explaining how to use your project (that is, the set of commands necessary to build the rainbow table and run the attack script).

Note that these above conditions are necessary but not sufficient. 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[3],
- not to generate a rainbow table bigger than 12 Gb,
- to be able to successfully crack 75% of a set of hashes ($\simeq$ 100) provided as a text file[4] under 45min of CPU time on a laptop[3].

This project is a *group project*, a single submission per group of 3 students is enough. The deadline is set on 19 April at 23h55.

2. This includes compiling your scripts as well as installing third party missing libraries
3. 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.
4. Recall that passwords are alphanumeric (lower and upper case) with length at least 6 and at most 8, are stored unsalted after a single pass to the SHA-256 hash function.