Assignment 1: Guessword

Software Security

Teacher Assistant: Elia Geretto

Q1 2021/22

Goals

- Easy assignment to warm you up.
- Test whether you can program in C (required to complete the other assignments).
- Familiarize you with simple crypto primitives.

Task description

You are given:

- an `/etc/passwd` file.
- an `/etc/shadow` file.
- a limited amount of time.

You should write a program to recover as many user passwords as possible using dictionaries of words commonly used in passwords.

/etc/{passwd,shadow}

Files used in Unix systems to store user passwords.

/etc/passwd line example

`jde100:x:1001:1001:John Doe,,,:/home/jde100:/bin/bash`
Username: `jde100`, UID: `1001`, Real name: `John Doe`

/etc/shadow line example

`jde100:$1$M9$80ZJhwG6Ngh1lLoKg5MdC1:14486::::::`

- Hashed with MD5 (algorithm: `1`)
- Salt used: `M9`
- Hashed password (base64): `80ZJhwG6Ngh1lLoKg5MdC1`

The password used is `iloveyou` (one of the most common passwords).

Program behavior

Your program will be invoked as follows:

```
$ ./guessword passwd_path shadow_path
```

It should write the passwords recovered to `stdout` as follows:

```
jde100:iloveyou
juy556:jose1985
hpn427:password1
```

Omit the users for which you cannot recover a password.

Requirements

Your program must:

- be written entirely in C.
- contain only code written by yourself.
- not use external libraries other than GNU libc (included by default), libcrypt (`-lcrypt`) and libpthread (`-lpthread`).
- not invoke external programs or connect to the network.
- compile without errors/warnings and run on a standard installation of Ubuntu 20.04 LTS.

Grading (1)

The program will be tested on the following system:

- Ubuntu 20.04 LTS (in Docker)
- AMD EPYC 7662 CPU (3.3 GHz, 4 cores maximum)
- 4 GB RAM maximum
- About 7323 MD5 hashes/s (with `crypt`, single thread)

Your program will be run with **different** `passwd`/`shadow` files and will be killed after **15 mins**. The output will be matched against the correct passwords.

Grading (2)

Your grade will be calculated as follows:

$$\text{grade} = \text{base} + \text{bonus} - \text{malus}$$

$$\text{base} = 1 + 9 \times \frac{\text{correct passwds} - \text{incorrect passwds}}{\text{total users}}$$

$$\text{bonus} = \frac{\text{max rank} - \text{rank}}{\text{max rank}}$$

rank : index in list sorted by rounded grade and submission time

malus : number of days past the deadline

Hints (1)

- Find patterns in the passwords people choose using the example files you get.
  - Grading files have different passwords than example files but follow the same patterns.
  - Use only the dictionary files we provide (gutenberg and top250).
- All the passwords are hashed with MD5.
- The passwords in the same `shadow` file use the same salt, but different files have different salts.

Hints (2)

- Linux uses `crypt` to generate `/etc/shadow`. You will need GNU libc extensions, so do not use Cygwin.
- Use `fflush(stdout)` after every line of the output. You do not want passwords to remain in the output buffer when the program is killed.

Testing

C is difficult to get right! Make sure your program does not contain undefined behavior:

- Use `-Wall -Wextra` and fix all warnings.
- Use sanitizers or `valgrind`.
- Verify the correctness of the output.

If your program crashes during the evaluation or the output format is incorrect, you will be penalized and possibly not pass the assignment!

Submission (1)

Deadlines and submission and handout will be handled through Canvas.

In the handout package, you will find a `sanity_check.py` script. Use it to **test** your ZIP archive before submission. If the archive does not pass the tests, your assignment will not be evaluated!

If your program does not compile, your assignment will not be evaluated!

Do **not** include binaries in your archive!

Pick a hacker handle between 3 and 16 alphanumeric characters which is also a valid UNIX username. It will be used for the entire course.

Submission (2)

The archive should contain:

- `README`: Plain ASCII file containing on **separate lines**, in this sequence, with no further text:
  - Hacker handle, name, e-mail, VUnet ID, student number
- `guessword.c`: All your code. It must compile with:

```
$ gcc -O3 -Wall -Wextra -pthread -o guessword \
    guessword.c -lcrypt -lpthread
```

- `*.txt` files that are needed by your program.

The files should **not** be in a subdirectory.

The size should be at most 5 MB compressed and 50 MB uncompressed.

Contact

You can ask questions through the "Discussions" page on Canvas, so that everyone can view the answer. Please, put "Assignment 1" in the discussion title.

If you think that your question contains sensitive information, you can ask it via email to `e.geretto@vu.nl` putting `softsec@vusec.net` in CC. Please, put the tag `[SoftSec]` in the subject.