School of Science
and Engineering
University of Dundee

AC31012/AC51042: Information Security

Assignment 1: Applied Cryptography

Deadline for Submission: **23:59 (11.59pm) on Monday 31st October 2022**

Hand in Method: One team member to submit a single **ZIP** file named after the team to the assignment document on My Dundee.

Date Feedback will be Received by: This will be received within the University's 3-week policy which for this assignment is Monday the 21st November 2022.

Penalty for Late Submission: One grade point per day late (meaning if a submission is one day late and marked as a C2 it will receive a C3 grade). A day is defined as each 24-hour period following the submission deadline, including weekends and holidays. Assignments submitted more than 5 days after the agreed deadline will receive a zero mark (AB).

Percentage of Module: This assignment is worth 25% of this module for AC31012 students, and 20% of this module for AC51042 students.

Overview of Assignment

In this applied cryptography assignment, you will work as a team to **implement two password login (authentication) procedures in C++**. One of the login procedures should be **secure**, but the other should have a **covert backdoor** that allows you to login as root or any other user on the system without knowing their passwords.

Later in the module, a part of your mark for Assignment 2 will be determined by the (in-)ability of the other teams to find the backdoor in the subverted procedure that you are developing here.

This means that your team will have to consider operational security: **ensure that no one outside of your team has access to your code and your ideas**. As part of Assignment 2, only the source code of all subverted login procedures will be distributed. The other teams will not have access to your secure login procedure or its source code, so you do not need to worry about code similarity between your secure and your subverted login procedures.

This assignment can be solved in **Ubuntu on the QMB Lab PCs**. You may use any other computer and operating system, but it is then your responsibility to ensure that your code also compiles on an Ubuntu configuration as found on the QMB Lab PCs.

Password File Format

Your login procedures must be able to read files that contain zero or more lines containing a **username** and a **SHA256**-hashed password in the format:

```
username:SHA256-hashed-password
```

For example, a password file might look like this:

```
root:5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8
alice:5ef2c394d5b63e4175cd331c74c8453c3e36eb8f47f6d648397ff6c1314fd705
```

The SHA256-hashed password for user `root` in the password file above is “password”. You can verify this by typing the following in the command line:

```
echo -n "password" | openssl sha256
```

In the same way, you can verify that the password for user `alice` is “mushroom”.

The **openssl** development library (pre-installed in Ubuntu on the QMB Lab PCs) provides the necessary functions to compute **SHA256** hashes (you just need to add the library to your code with #include "openssl/sha.h").

As a starting point for developing your login procedure, you can read the following Stack Overflow post:

- Generate sha256 with OpenSSL and C++

Functionality of the Login Procedures

Your **secure and subverted** login procedures **must** both satisfy the following requirements:

- **R1:** The login procedure must work with the password file format described in the previous section.
- **R2:** It must include the authlib.h header file and use the two functions defined therein.
- **R3:** It must call the function void authenticated(std::string u), where u is the username, whenever a user enters a correct username and password pair.

In addition, your **secure** login procedure **must** satisfy the following requirements:

- **R4:** It must call the function rejected(std::string u) if an invalid username and password pair was entered.
- **R5:** It must not call authenticated(std::string u) unless a correct username and password pair for username u was entered.

It is up to you to decide whether your secure and subverted login procedures offer the user one or more attempts to log in before rejecting and exiting. Your secure and subverted login procedures may offer additional functionalities, which may help disguise backdoors, but **shorter backdoored login submission will receive more marks** (see the marking scheme for details on this).

You **must not** modify `authlib.h` or `authlib.cpp`. You **cannot** assume that the functions in `authlib.cpp` will be implemented in the same manner when your code is tested, so do not rely on this as part of the design of your login procedures. Likewise, you can modify the passwords file as you work on your login procedures, but you **cannot** change the filename or assume anything about the contents of this file when your code is tested, other than they will follow the format above.

Submission

This assignment must be handed in by **one student in each group**. Submit your assignment as a single ZIP file **named after your group** by the deadline.

The ZIP file must contain:

1) A file **login.cpp**. This is the secure password login procedure. Your login.cpp program **must:**
    a) satisfy requirements **R1–R5** above,
    b) compile without warnings when the flags -Wall -pedantic -Wextra are used,
    c) hash the submitted passwords with **openssl**'s sha256 hash function,
    d) contain fully commented source code.
2) A file **login-subverted.cpp**. This is the password login procedure with a backdoor. Your backdoor **must:**
    a) allow you to login as root or any other user on the system without knowing their passwords,
    b) satisfy requirements **R1–R3** above,
    c) compile without warnings when the flags -Wall -pedantic -Wextra are used,
    d) hash the submitted passwords with **openssl**'s sha256 hash function,
    e) contain fully commented source code (but comments may be misleading ;).
3) A file **report.pdf**. This PDF file documents the vulnerability in your backdoored login procedure. Your report **must** :
    a) be no more than **1 page** and list the team name and team members,
    b) describe the steps to trigger the vulnerability, i.e. how can an attacker login without knowing a user's password,
    c) show where the vulnerabilities are in the code,
    d) explain why you think that your vulnerabilities are difficult to detect,
4) A **Makefile**. This file compiles both your secure and your subverted login procedures.

Marking Scheme

A grade	B grade	C grade	D grade	Fail	Mark
The login.cpp source code is secure, correct, works with hashed password database and is thoroughly commented. *10 to 7 marks*	The login procedure works correctly, i.e., satisfies 1a, 1b, and 1c. *6 marks*	login.cpp compiles but only satisfies two of the three conditions 1a, 1b, 1c. *5 marks*	login.cpp compiles but only satisfies one of the three conditions 1a, 1b, 1c. *4 marks*	login.cpp *not submitted, or it does not compile or does not satisfy any of the conditions 1a, 1b, 1c above.* *3 to 0 marks*	
Comments:					
The login-subverted.cpp source code is works with hashed password database and is thoroughly commented. (Comments are allowed to be misleading!) *10 to 7 marks*	The login-subverted procedure works correctly with the hashed password database and does not produce compiler warnings (i.e. satisfies 2a, 2b, 2c, and 2d.) *6 marks*	The login-subverted procedure compiles but only satisfies two of the three conditions 2a, 2b, 2c, 2d. *5 marks*	Login-subverted.cpp compiles but only satisfies one of the three conditions 2a, 2b, 2c, 2d. *4 marks*	Login-subverted.cpp *not submitted, or it does not compile or does not allow the attacker to authenticate as another user.* *3 to 0 marks*	
Comments:					
The vulnerability is well explained its covertness is well-justified. It is clear that research has been taken into this area and alternatives considered. *10 to 7 marks*	All three sections are adequately described. *6 marks*	1 out of the three mandatory sections (see 3a, 3b, 3c, 3d above) is inadequately covered. *5 marks*	2 out of the 3 mandatory sections (see 3a, 3b, 3c, 3d above) are inadequately covered. *4 marks*	Inadequate report or no report.pdf submitted. *3 to 0 marks*	
Comments:					
All files (Makefiles, report, cpp sources) are submitted in the required file format and structure. The report is not longer than 1 page. *10 to 7 marks*	Minor issues in one of the submitted files or with the submission (e.g., naming, missing team members, etc.). *6 marks*	One or more files inadequately submitted or minor issues with two or more files. *5 marks*	One file missing. *4 marks*	Two or more files missing. *3 to 0 marks*	
Comments:					
Marks awarded by ranking submissions from shortest to longest in number of ";" and "," used in source code. *10 to 7 marks*	Code is obfuscated as, e.g., HEX strings in order to minimize number of ";" and ",". *6 marks*	Sum of ";" and "," characters in source code of subverted login is greater than 65. *5 marks*	Sum of ";" and "," characters in source code of subverted login is greater than 100. *4 marks*	Inadequate subverted login procedure submitted (e.g., implemented w/o hash function.) *3 to 0 marks*	
Comments: