COMPX518-23A
Assignment 3

Total Marks: 20:
Due: 12th June, 5 PM, 2023

Part 1: MD5 Collisions (10 marks)

A cryptographic hash function requires that both the pre-image resistance and collision free properties are satisfied. The cryptographic hash function MD5 is now broken and algorithms have been invented that find collisions in MD5 easily. In this part of the assignment you will use a collision generation tool to find collisions and then generate two C binary files that have slightly different content but generate the same MD5 hash. This part of the assignment needs to be done on Linux. The `md5collgen` tool to be used for this part has been tested on Ubuntu 20.04.

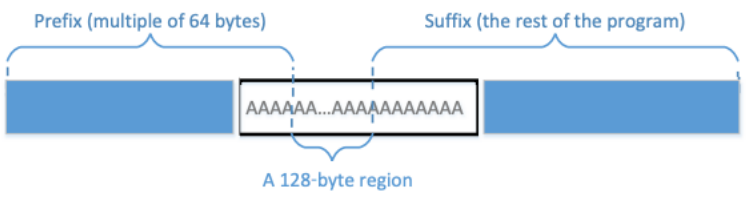
1. Use the `md5collgen` tool from moodle to generate two files with the same md5 hash but different content. The `md5collgen` tool uses a prefix file containing arbitrary data and creates two files with that prefix and additional content that produces the same md5 hash. Use the tool as shown below.

   ```
   md5collgen -p prefix.txt  -o out1 out2
   ```

   1.1 In your report, show the prefix file and the two output files **[1 mark]**
   1.2 In your report, answer the following questions **[2 marks]**
      1.2.1 How many new bytes does the tool generate?
      1.2.2 What happens when the length of your prefix is 64 bytes and when it is not?

2. In the previous task the two files generated contained random data. In this task you will generate two executables that differ slightly but have the same MD5 hash. Use the C source code on moodle. You will generate two versions of the executable from this program such that the contents of the xyz array are different but the MD5 hash is same for both the versions.

   Start by compiling the source code to an executable/binary. Now open the binary in a hex editor and locate the position where the xyz array starts. Inside the array, we need to find two locations, from where we can split the executable file into three parts: a prefix, a 128-byte region, and a suffix. The length of the prefix needs to be a multiple of 64 bytes as shown in the following diagram.


Prefix (multiple of 64 bytes)
Suffix (the rest of the program)
AAAAAA...AAAAAAAAAAA
A 128-byte region

Now run `md5collgen` on the prefix to generate two outputs that have the same MD5 hash value. Let us use P and Q to represent the new 128 bytes generated by the tool for the two outputs. Therefore, we have the following:

```
MD5 (prefix || P) = MD5 (prefix || Q)
```

Based on the Merkle Damgard construction of MD5, we know that if we append the same suffix to the above two outputs, the resultant data will also have the same hash value. Basically, the following is true for any suffix:

```
MD5 (prefix || P || suffix) = MD5 (prefix || Q || suffix)
```

Therefore, we just need to use P and Q to replace 128 bytes of the array (between the two dividing points), and we will be able to create two executables that have the same hash value. Their outcomes are different, because they each print out their own arrays, which have different contents.

Tools: You will need the `md5collgen` tool and a hex editor such as `bless` or `ghex` for this. To divide a file from a particular location, you can use the head and tail utilities on Linux. Some hex editors provide this feature as a GUI.

```
$ head -c 3200 a.out > prefix
$ tail -c 100 a.out > suffix
$ tail -c +3300 a.out > suffix
```

The first command above saves the first 3200 bytes of a.out to prefix. The second command saves the last 100 bytes of a.out to suffix. The third command saves the data from the 3300th byte to the end of the file a.out to suffix. With these two commands, we can divide a binary file into pieces from any location. If we need to glue some pieces together, we can use the cat command.

In the report you will describe in detail the entire process with screenshots. Make sure that you show the output of execution of the two final binaries and their md5 hash. You will also paste (not screenshot) the binary content in hexadecimal of the xyz arrays in the two executables. **[7 marks]**

Part 2: Threat Modelling (10 marks)

In this part you will create a threat model for a simple web based quiz application such as the one we use for quizzes in moodle.

The application has two main external entities, the administrators and the users.

The databases used are, Credentials, Question DB, Current quiz, Scores, User responses.

1. The administrators land on the admin page, where they login using their credentials. The users land on the users page, where they login using their credentials. The credentials are stored in the Credentials database.
2. The administrators can perform the below 5 activities
    a. Change password: where the credentials are stored in the credentials database
    b. Add a question: This adds a new question to the Question DB
    c. Create Quiz: This creates a new quiz using questions in Question DB and populates entries in Current quiz database
    d. View scores: This allows administrators access to view all the scores in the Scores database
    e. Edit scores: This allows the administrators full access to all the scores in the Scores database
3. The users can perform the below 3 activities
    a. Change password: where the credentials are stored in the credentials database
    b. Take quiz: The user takes the current quiz. The quiz questions comes from the Current quiz database. The answers are stored in the User response database and the score is stored in the Scores database.
    c. View scores: This allows users access to see all their scores from the Scores database

You will first create a Data Flow Diagram for this application. The diagram should show, external entities, processes, data stores, data flow and trust boundaries. **[6 marks]**

You will then perform a threat modelling exercise for the data flows in your diagram. Use the STRIDE model and list all possible threats to all the processes. If you find more than five threats for a process list the top five. **[4 marks]**

You will find that you do not have sufficient information for many of the processes described above. This is usually the case in real life and will require the developer to discuss them with the client. In the assignment when faced with lack of information assume the worst and proceed accordingly.

Assignment submission

Submit the assignment as a single pdf file.

Extensions

No extensions will be given unless approved by the Department of Computer Science (https://www.cs.waikato.ac.nz/student-resources/application-for-an-extension-of-deadline)
You can submit late. However, late submissions will be deducted **1 mark/ day.**

Plagiarism

Credit all sources you refer to. Students found plagiarising will be reported to the disciplinary committee. You are expected to follow the University's guidelines here: https://www.waikato.ac.nz/students/academic-integrity/student-information/plagiarism. Assignments will be checked against anti-plagiarism checkers.