School of Computing & Information Technology

CSCI262 System Security
SIM-2021-S4

Assignment 1 (14 marks, worth 14%)
Due date: 23rd October, 2021, 20:55 (SG Time).

Part One: Short answer questions: 4 Marks

1. Determine the entropy associated with the following method of generating a password. **1 Mark**

   > Choose, and place in this order, one lower case letter, following by one upper case letter, followed by two digits, followed by @, followed by two letters, each upper or lower case, and then followed by three symbols drawn from the set {$,9,5,v,w,J}. Finally, apply the hash function `Tiger` to give an output string in hex which will be used as a password.

   Assume random choices are made with equal likelihood of each symbol from the space being chosen from. So for a random digit there are 10 possibilities, each chosen with probability 1/10.

2. Generate a BLP lattice–structured system where the objects and subjects are appropriately levelled to give access consistent with the access control matrix below. You need to describe the process by which you obtain your lattice. R and W correspond, respectively, to read and append. You are to use only the mandatory BLP rules, and a default allow in place of the discretionary rule. Be sure to add a level as necssary to ensure this is a lattice. **2 Mark**

	O_1	O_2	O_3	O_4	O_5	O_6
S_1	R	R	R	R	R	RW
S_2	R			W	W	W
S_3	R	RW	RW		W	W
S_4	R			R		W
S_5	R					
S_6	R	R	R		R	

3. For the following collection of statements, describe the sets of actions, objects, and subjects; and draw an access control matrix to represent the scenario. **1 Mark**

```
Alice can climb trees and eat apples.

Bob can climb fences, eat apples, and wave flags.

Trees can hurt apples.

Carol can jump waves and wave flags.
```

Part Two: Implementing a rainbow table — 10 Marks

You are to write a program, in C/C++, Java or Python, that should run using the following instruction:

```
$ ./Rainbow Passwords.txt
```

where the file `Passwords.txt` contains a list of possible passwords. The password file contains a password per line, as in the provided `words` file and consists of strings of printable characters. Any password used must be taken from this file, so the only stored hash information needs to relate to those entries in the file.

The program is used to find pre–images for given hash values. Rainbow tables can be used to solve pre–image problems for hash functions. At the simplest level they can simply be a list of hash values and the corresponding pre-images, often from some dictionary. This can be expensive in terms of storage space however, and a more efficient way of identifying pre-images involves the use of the hash function and reduction functions.

First step (5 marks): Your program will do some initial computations to generate the rainbow table. The process is as follows:

1. Read in the list of possible passwords. Report on the number of words read in. (**0.5 mark**)
2. For each previously unused word W, first mark it as used and then carry out the following process (**3 marks**):
   (a) Apply the hash function H to the word W to produce a hash value $H(W)$, which we refer to as the current hash.
   (b) Apply the reduction function R to the current hash, which will give a different possible password which should be marked as used and then hashed. The resulting hash value is recorded as the current hash.
   (c) Repeat the previous step four times. You can deal with collisions if you like but are not required to.
   (d) Store the original word W and the final current hash as an entry in your rainbow table.
3. To assist with the later identification of the pre–images you should sort the rainbow table based on the hash values (**0.5 mark**).

4. Output the list of words and corresponding "final current hashes" to a text file `Rainbow.txt`. Report to standard out the number of lines in your rainbow table (**1 mark**).

Second Step (5 marks): You are now ready to carry out the second part of the exercise, finding pre–images. Request a hash value from the user. There should be appropriate error checking as to the length of the input string etc. The process of identifying the pre–image for the provided hash value is sketched as follows:

1. Check if the hash value is in the rainbow table (**1 mark**).
2. If the hash value isn't in the rainbow table you reduce and hash until you get a hash value that is in the rainbow table, or until you have done the reduce and hash enough times that it's clear from the way the rainbow table is generated that something is wrong. Display an appropriate message if this happens (**2 mark**).
3. Once you have identified the relevant hash value in the table you take the corresponding password and hash it. If that is your hash value you have your pre–image. If it isn't, then reduce and hash again, until the reduced word hashes to the hash value being searched for (**1 mark**).
4. Output the releative reduced word, that is your pre–image (**1 mark**).

A reduction function is designed to take a valid hash value and return a valid password, in this case a valid word from `Passwords.txt`. The reduction function to be used should be based on the number of words in the `Passwords.txt` file, and since that isn't a fixed length file your method needs to be dynamic. You should find a way to convert a hash value to an integer that identifies one of the passwords. You can define yourself a reduction function R or find some examples online. You should describe how your reduction function works in `readme.txt`.

You should use MD5 as the hash function. You don't need to implement the MD5 hash function by yourself. You can find such a file online. Provide a reference if you use any open source implementation.

1. Submission is via Moodle.
2. Include the compilation instructions with your submission in a file `readme.txt`. That file should also contain a description of how you do the reduction. If no such file exists in your submission then you will get **zero** for this Part two.
3. Late submissions will be marked with a 25% deduction for each day, including days over the weekend.
4. Submissions more than three days late will not be marked, unless an extension has been granted.
5. If you need an extension apply through SOLS, if possible **before** the assignment deadline.
6. Plagiarism is treated seriously. Students involved will receive zero.