School of Computing & Information Technology

CSCI862 System Security
Spring 2016

Assignment 1 (12 marks, worth 12%)

Due 11:59pm Saturday 20^{th} August 2016.

Part One: Short answer questions: 4 Marks

1. A phonetic segment generator works as follows. Each segment has 3 English letters. The form of each segment is $\Delta\Phi\Delta$ (consonant, vowel, consonant), where Φ is an element in {a, e, i, o, u} and Δ is an English letter which is not in {a, e, i, o, u}. Determine the entropy associated with the following method of generating a password. **1 Mark**

   > Choose, and place in this order, one phonetic segment cosisting of lower case letters, followed by three digits and then two symbols drawn from the set {+,*,@,#,$}. Finally, apply `Whirlpool` to give an output string in hex which will be used as a password.

   You should assume the random choices are made with equal likelihood of each symbol from the space being chosen. So for a random digit there are 10 possibilities each of which is chosen with probability 1/10.

2. Consider the following statements and answer the subsequent questions:

   ```
   Alice can climb walls and jump fences.
   Bob can push walls and push doors.
   Chris can push Alice, push fences and jump walls.
   Dan can open doors and jump Alice.
   ```

   (a) What are the subjects, objects and actions for this scenario? **0.25 Mark**

   (b) Draw an access control matrix representing this scenario. **0.25 Mark**

3. Consider the lattices in the file `A1-Q3.pdf` and answer the questions below. A line going down from level X to level Y indicates that level X dominates level Y. Assume this diagram is with reference to BLP. For the questions other than the first you can work with whichever of the diagrams you feel most comfortable with. Justify your answers.

   (a) How are the diagrams A and B related as lattices? **0.25 Mark**

(b) I have one object at level 1010 and another object at 0001. What is the most appropriate level for a subject to be assigned so both objects can be read by the subject? **0.25 Mark**

(c) I have one subject at level 0110 and another subject at 0111. What is the most appropriate level for an object to be assigned so both subjects can append to the object? **0.25 Mark**

(d) I have three objects, one at 1000, one at 0100, and one at 0010. What is the most appropriate level for a subject to be assigned so all three objects can be read by the subject? **0.25 Mark**

(e) Explain how the digits can be interpreted in a multilateral and multilevel sense. **0.5 Mark**

4. Assume that Alice has registered with the server Bob to use Lamport's one–time password scheme. Alice's password is `Alice1234567`, where you should replace the `1234567` with your own student number. If $n = 10$ initially, what are the first two and the last one–time passwords transmitted by Alice? Use MD5 as the hash function. **0.5 Mark**

5. Consider the BLP level relationship diagram in 862-A1-Q5.pdf, and the associated explanation of the notation, and answer the subsequent questions.

   (a) Does the diagram define a lattice? Justify your answer. **0.25 Mark**

   (b) Assume that if the diagram didn't define a lattice you have fixed it so it does, without changing the relationships between the existing levels. Some of the domination relationships shown in the diagram are redundant. Identify two such lines and explain why they are unnecessary. **0.25 Mark**

Part Two: Implementing a rainbow table 8 Marks

You are to write a program, in C/C++ or Java, that compiles on Banshee with an instruction you provide. It should run on Banshee 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 `/usr/dict/words` 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.

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.

2. For each previously unused word W, first mark it as used and then carry out the following process:

   (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.

4. Output the list of words and corresponding "final current hashes" to a text file `Rainbow.txt`.

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.

2. If the hash value isn't in the rainbow table you hash and reduce until you get a hash value that is in the rainbow table, or until you have done the hash and reduce 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.

3. Once you have identifed 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.

4. Output the releative reduced word, that is your pre–image.

A reduction function is designed to take a valid hash value and return a valid word, in this case a valid word from `Passwords.txt`. The reduction function to be used is based on the number of words in the `Passwords.txt` file. You should take an appropriate number of bits from the front of the hash function, convert that to an integer that identifies one of the passwords.

You should use MD5 as the hash function.

1. Submission is via Moodle.

2. Your program will be compiled and tested on Banshee.

3. Include the compilation instructions with your submission (i.e., provide a readme.txt file).

4. Late submissions will be marked with a 25% deduction for each day, including days over the weekend.

5. Submissions more than three days late will not be marked, unless an extension has been granted.

6. If you need an extension apply through SOLS, if possible **before** the assignment deadline.

7. Plagiarism is treated seriously. Students involved will receive zero.