Samuel Petit
20683298
saapetit@connect.ust.hk

1. (9pt) Static and dynamic security analysis.
(a) (3pt) Explain the difference of static security analysis and dynamic security analysis.

Static security analysis doesn't need to execute the program to analyse it. It can analyse the whole program however because of this it may lead to false positives.

Dynamic security analysis must run the software in order to analyse it. Thus it can only analyse the parts of the programs that have been executed which may lead to false positives.

(b) (3pt) Can taint analysis be implemented as a "dynamic analysis"? Explain your answer.
Yes. It is possible and in fact very popular to use dynamic taint analysis in security testing. The way it would be implemented is by running a program and observing which computations are predefined by taint sources. This can lead to many different applications such as detecting code injection for example, by monitoring if the user input is executed.

(c) (3pt) Can fuzz testing be implemented as a "static analysis"? Explain your answer.
Noit is not possible to implement fuzz testing as a static analysis because at its core fuzzing will execute a program with different inputs and then monitor the execution for any errors.

2. (9pt) Information flow analysis and taint analysis.
(a) (4pt) We talked about two kinds of taint source/sink configurations: 1) take untrusted user inputs as the taint source, and critical software statements, e.g., the index of array access as the taint sink point, and 2) take sensitive program value (e.g., RSA private key) as the taint source, and use certain network message sending APIs as the taint sink points. Explain the why we would use these two configurations.

Configuration 1 would be used because using user input as an input to sensible parts of a code base can be very risky if not checked properly. By using this configuration we can identify such places in a code base and make sure that user input is properly used. (i.e. by checking if the input does not create an out of bounds exception or cause code injection and other similar applications).
Configuration 2 would be used because identifying parts of a code base which uses secret / critical information would enable a programmer to identify potential errors and security issues when dealing with sensitive information and thus making sure they do no expose any kind of vulnerability by doing so.

(b) (5pt) Explain the difference between information flow and data flow. Which one is preferred in security analysis, and why?
Information flow is more general than data flow. Specifically information flow shows the movement of information between variables in the context of a computer system.
On the other hand, data flow gathers information about the possible set of values calculated at various points in a computer program.

3. (6pt) This problem deals with storing passwords in a file.
(a) (2pt) Why is it a good idea to hash passwords that are stored in a file?

It is a good idea because it increases the security of the system. If the passwords were not hashed, all it would take for an attacker to access all of those passwords is to gain access to the system. On the other hand, with hashed passwords the attackers need to crack the hashing system on top of gaining access

(b) (2pt) Why is it a much better idea to hash passwords stored in a file than to encrypt the password file?

It is a better idea because encrypting a file only requires to hack a single encryption, whereas encrypting every password separately requires the hacker to decrypt all of the passwords one by one, so it would be much harder.

(c) (2pt) What is a salt and why should a salt be used whenever passwords are hashed?
A salt is a randomly generated piece of text, it is very useful when hashing passwords because by appending the salt to the password and then hashing that value, we can then store both the salt and the hashed password. This makes it possible to generate unique hash values for passwords that maybe the same, thus making an attack that more difficult to crack.

4. (7pt) Suppose that on a particular system, all passwords are 10 characters, there are 64 choices for each character, and the system has a password file containing 512 hashed passwords. Furthermore, the attacker has a dictionary of 2^20 common passwords. Provide pseudo-code for an efficient attack on the password file in the following cases.
(a) (4pt) The password hashes are not salted.

Assuming hash() is the function used by the system to hash passwords

```
for(String password in 220 commonPasswordsList) {
    String hashed = hash(password);
    if(hashed is an entry in the password file) {
        print(password)
    }
}
```

(b) (3pt) The password hashes are salted.

Assuming hash() is the function used by the system to hash passwords with a given salt assuming that we know the length of the salt used by the system
Let's assume that length is 2 for simplicity

```
for(String password in 220 commonPasswordsList) {
   String salt;
   for(char c1  in all possible 64 chars) {
      for(char c2  in all possible 64 chars) {
         salt = c1 + c2;
          String hashed = hash(password, salt);
            if(hashed is an entry in the password file) {
               print(password)
            }
      }
   }
}
```

Note : in the case that we do not know the length of the salt then it becomes even more complicated as we have to try every combination of 64 chars for every possible length.

5. (12pt) Suppose you are a merchant and you decide to use a biometric fingerprint device to authenticate people who make credit card purchases at your store. You can choose between two different systems: System A has a fraud rate of 1% and an insult rate of 5%, while System B has a fraud rate of 5% and an insult rate of 1%.
(a) (2pt) Define fraud rate.

The fraud rate is the false accept rate : it represents the percentage of times and invalid user was accepted to the biometric system.

(b) (2pt) Define insult rate.

The insult rate is the percentage of times a valid user is rejected by the system.

(c) (4pt) What is the equal error rate, how is it determined, and why is it useful?

The equal error rate is the rate at which both insult and fraud rate are equal. It is useful for determining the overall accuracy of a biometric system.

(d) (2pt) For the merchant scenario described above, which system is more secure and why?

For a system to be the more secure we would take the higher insult rate in this case as we care more about valid users entering the system than making the system user friendly. Thus in this scenario, we would take system A.

(e) (2pt) For the merchant scenario described above, which system is more user friendly and why?

The more user friendly system is system B because it contains a lower insult rate, meaning the user will be less likely to try multiple times to gain access to the system.