Homework to be submitted by 21 Sep 18:59 hour

Group 5

Wong Ann Yi (1004000)
Liu Bowen (1004028)
Tan Chin Leong Leonard (1004041)

Exercise 1

Is it possible to design and implement a system in which no assumptions about trust are made? Why or why not?

Answer:

A system which is designed and implemented with no assumptions about trust will face a risk of under provisioning which leads to an ineffective defense or over provisioning which leads to high cost overrun. This is because building a security system rests on assumptions about the environment and the level and types of security required. For example, withdrawing cash from ATM requires a user keying his PIN. The assumption is that the physical ATM, its location, the user, his ATM card and his PIN are secure. This assumption is treated as an axiom and is made because almost every user would use an ATM card and his PIN to withdraw cash from an ATM. However, an attacker can steal a user's wallet which contains his ATM card and guess his PIN (if he uses his birth date which is found in his ID card as the PIN) and is able withdraw his money from the ATM. Therefore, in an untrustworthy environment, the assumption is wrong and the consequence invalid. The design of the system may need to include a camera or biometric sensor to detect the correct facial or fingerprint before money can be dispensed.

Secondly, in general, designers of policies and system always make two assumptions. One is that the policy correctly and unambiguously partitions the set of system states into "secure" and "non-secure" states[1]. Like mentioned in course notes, let P represents all status in a system, which definitely is consist several partitions, such as Q represents secure status, R includes restricted rules and the reset is insecure. The other necessary assumption is that the security mechanisms prevent the system from entering a "non-secure" state. In fact, these two assumptions differ from each other. The first one asserts the policy is a correct description of what constitutes a "secure" system while the second one emphasizes the significance of security mechanisms, as mechanisms with good security performance can definitely enhance the level of policy.

In summary, an effective security system need to be planned and implemented by considering the assumptions on trust. These assumptions include: each mechanism is designed to implement one or more parts of the security policy, which not only guarantees the security requirements of whole system but also enforces the security level of system. The union of the mechanisms implements

[1] Computer Security: Art and Science (Matt Bishop) – Chapter 1.4. Assumptions and Trust

all aspects of the security policy. During the implementation process, the mechanisms are implemented, installed and administered correctly as well.

Exercise 2

A respected computer scientist has said that no computer can ever be made perfectly secure. Why might she have said this?

Answer:

Given enough resources (and time), an attacker can evade the security procedures and mechanisms being enacted in an organization and its computing assets. If there is ever a perfectly secure method, there will not be a need for detection and recovery mechanisms which much effort has gone into building them. Also a perfectly secure system would imply that the system specification, design and implementation which determine the "trust" of the security system are flawlessly carried out which is not possible in the real world. While organizations invest resources in policies and mechanisms which support the three aspects of security in confidentiality, integrity and availability, a serious attacker can deploy resources to crack or seek weaknesses in the entire connecting components from the layers in the networking, OS and application protocols to physical computing assets and human negligence. While researchers have studied the broad categories of attacking threats such as disclosure, deception, disruption and usurpation, and try to develop effective defensive mechanisms, the techniques used by the attacker is always evolving and in reality, most organizations do not have the speed to implement up-to-date counter measures readily. In addition, there are many components in a computer system which are not within one's control. For example, two vulnerabilities named Meltdown and Spectre were discovered in all Intel chipsets, and they could allow an attacker to access system memory and potentially obtain passwords and other secure information[2] in all computers (as most are using Intel processors). Many companies were not prepared to deploy patches rapidly to all the computing systems and many were trying to figure out which of the computing inventory were being affected. Therefore, it is not realistic to assume a perfectly safe computer system when there are many components in the system which are developed and made by many external companies which one does not have control over. Other than Intel, the popular Microsoft Windows and Google Android operating systems are also known have security vulnerabilities.

Human vulnerability is another factor which can help an attacker hack into a secured system. Using techniques such as social engineering, an attacker can obtain password or security badge to gain access to the computer system to obtain valuable information. In addition, unhappy employees or "insiders" can bypass security controls to attack the system from the "inside" and untrained personnel could misconfigure the system to unwittingly open system ports for an outsider who is "eavesdropping" to enter into the system.

[2] "Meltdown and Spectre are wake-up calls for the tech industry" - https://www.engadget.com/2018/01/05/meltdown-and-spectre-are-wakeup-calls-for-the-tech-industry/

A secured system is only good for the point in time and a robust operation and maintenance process must be in place to test its defense effectiveness, find new threats, uncover vulnerabilities and implement patches. Therefore, there is no perfectly secure computer. The process and tenets to build a "trusted" security system in the areas of policy, specification, design and implementation are not foolproof and perfect. The external components and services (for examples the chips and software manufacturers are prone to security lapses) may contain security risks which one cannot control. The human factors are key contributing factors to security risk and the attackers' techniques and tactics will continuously evolve.

Exercise 3

Give examples for situations in which:

a) A compromise of confidentiality leads to a compromise in integrity and vice versa.
Answer:
Confidentiality leads to a compromise in integrity
A user compromises his cryptographic key which is used to encrypt the data. The attacker uses the key to access the data and make unauthorized changes to it.

Integrity leads to a compromise in confidentiality
An attacker gains access to the bank system and changes the money in certain accounts and in process, he also gained access to owners' personal and private information.

b) A compromise of confidentiality leads to a compromise in availability and vice versa.
Answer:
Compromise of confidentiality leads to a compromise in availability
An attacker obtains the username and password to gain access to a company's database server and intentionally disrupt its function during trading hours leading to the company losing large amount of money.

Compromise of availability leads to a compromise in confidentiality
The IT security system of the company went down during the business hours. The IT administrator made a bypass to ensure business operation was resumed. One of the employees seized the opportunity to gain access to the financial payroll server and viewed the confidential salary information of each employee.

c) A compromise of integrity leads to a compromise in availability and vice versa.
Answer:
Compromise of integrity leads to a compromise in availability
A departing IT administrator changes the passwords to the computing resources in the company. The new IT administrator cannot login and therefore unable to access the computing resources. The company has to schedule downtime in order to recover the passwords which

leads to the disruption of its operation and services and subsequent loss of customers' confidence.

Compromise of availability leads to a compromise in integrity
As the customer withdrew cash from the ATM, the bank database system crashed at the precise moment when the transaction took place. This caused the customer's account value to be un-deducted despite the withdrawal. As a result, the bank lost money for this transaction.

Exercise 4

The query-set-overlap mechanism requires a history of all queries to the database. Discuss the feasibility of this control. In particular:

a) How will the size of the history affect the response of the mechanism?

Answer:

The history tracker keeps the attributes answered in the past queries. If the history is small, then the mechanism will have less possible queries and the query-set-overlap mechanism will work. However, if the history is big there is high processing overhead as every new query will need to be compared with all previous ones causing the system to slow down. In addition, the profile of all the users are required to be stored and this increases the use of storage space.

b) How practical is a system that enforces this mechanism?

Answer:

It is not practical for a system to enforce this especially in large datasets with many users as the history tracker becomes very big. The processing overhead will be high causing the system to slow down and large storage space is required. In addition, this mechanism is not effective against several users who may collude to make independent queries and later collate them to form useful information.

However, we read an interesting journal article which a particular team of researchers have developed a model to mitigate these shortcomings. On improving performance, one possible solution is to specify policies such as discard stored information whenever the database is updated. On preventing a colluded attack, one possible solution is to assume all queries from all users as events of a single infinite-length execution[3]. As we have limited time to research sufficient materials, we would conclude (based on current information) that the query-set-overlap mechanism has limitations and will not be practical for large datasets and organizational use.

[3] Query Monitoring and Analysis for Database Privacy - A Security Automata Model Approach - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4795904/

Exercise 5

Let c be a copy flag and let a computer system have the same rights as in Exercise 5 of the Classwork. The set of rights {read, write, execute, append, list, modify, own}.

a) Using the syntax in Section 2.3 of Bishop's, write a command copy_all_rights(p,q,s) that copies all rights that p has over s to q.

Answer:

```
command create_file(p,s)
    create object s;
    enter c into a[p,s];
    enter own into a[p,s];
end
command copy_all_rights(p,q,s)
      enter r into a[q,s];
      enter w into a[q,s];
      enter e into a[q,s];
      enter a into a[q,s];
      enter l into a[q,s];
      enter m into a[q,s];
      enter own into a[q,s];
end
```

You need to check if the right is available in p before u copy over to q.

b) Modify your command so that only those rights with an associated copy flag are copied. The new copy should not have the copy flag.
Answer:

```
command copy_all_rights(p,q,s)
    if c in a[p,s]
    then
      enter r into a[q,s];
      enter w into a[q,s];
      enter e into a[q,s];
      enter a into a[q,s];
      enter l into a[q,s];
      enter m into a[q,s];
end
```

You need to check if the right is available in p before u copy over to q.

The own right is not copied to the q and therefore process q does not have the rights to copy flag.

c) In part b), what conceptually would be the effect of copying the copy flag along with the right?
Answer:

If the copy flag is copied with the right, the process q is able to assign all the rights over object s to other processes.

Exercise 6

This exercise asks you to consider the consequences of not applying the principle of attenuation of privilege to a computer system.

a) What are the consequences of not applying the principle at all? In particular, what is the maximal set of rights that subjects within the system can acquire (possibly with the cooperation of other subjects)?

Subjects will be able to grant other subjects rights even if it itself does not possess those rights

Answer:

There will be security risks and undesirable consequences for not applying the principle of attenuation of privilege. In a system without such principle, any subject can grant any rights to another subject(s) and object(s) without rules. This will lead to harmful effects to the confidentiality, integrity and availability of resources and result in security risks.

For example, if the Linux operating system does not apply the principle of attenuation of privilege, a user or subject A can grant any rights (write, delete and others) to user B. This may lead to a non-secure state and exposing objects owned by A to unauthorized use by B. Another example would be giving rights to others to read data tables and making changes to the data which breaches the confidentiality and integrity of the data (for example salary data of employees)

Assuming that there is a set of rights *R* in system with the following functions: {read, write, execute, append, list, modify, own}. Subjects within a system can acquire the maximal set of rights, subjects are able to acquire all the rights in system over an object with the help of other subjects. For example, a subject (without being granted any rights) may simply copy an object from the system and acquire all the rights of the object, which allows it to write, delete, edit it and even to transfer it to other subjects.

b) Suppose attenuation of privilege applied only to access rights such as read and write, but not to rights such as own and grant rights. Would this ameliorate the situation discussed in part a)? Why or why not?

Answer:

This would not ameliorate the situation as discussed in part a). if the principle attenuation of privilege is not applied to rights such as “own” and “grant_rights”, a subject A, can grant own rights to another subject B. As a result, B can add privileges such as “read” and “write” to itself.

c) Consider a restricted form of attenuation, which works as follows. A subject q is attenuated by the maximal set of rights that q, or any of its ancestors, has. So, for example, if any ancestor of q has r permission over a file f, q can also r f. How does this affect the spread of rights throughout the access control matrix of the system? Develop an example matrix that includes the ancestor right, and illustrate your answer.

Answer:

This form of attenuation will result in subject *q* inheriting the maximal set of rights from any of its ancestors in the system. For the security concerns, it can affect the security situation in certain system. Considering *Ancestor1*, the ancestor of *q*, has a new right *r* which isn't willing to grant to *q*.

For example, an ancestor process may have read, write and delete rights over a data table but it only wishes to give read only rights to a child process (whose purpose is to provide SQL query on the dataset for analysis purpose but not for any modification). However, in this form of attenuation, the child process would inherit the full rights to modify and may cause the dataset integrity to be breached.

Therefore, this spread of rights will allow unauthorized users to be granted access and weaken the control power of the access control matrix.

Assume that maximal set of rights are as follow *{read, write, modify, own}*

Scenario 1:

	File *f*	File *m*
q	*read*	*write*
Ancestor1	*read*	
Ancestor2		*write*

In scenario 1, *q* has *read* permission over *f* as its *Ancestor1* has. Concurrently, *q* holds the *write* rights over *m* as well.

Scenario 2:

	File *f*	File *m*
q	*read, modify*	*write, read*
Ancestor1	*read*	*read*
Ancestor2	*modify*	*write*

In scenario 2, *Ancestor2* updates *modify* rights over file *f* and therefore *q* inherits the same modify rights. *q* also acquires *read* permission over file *m* which is inherited from *Ancestor1*.

3a) Example for Integrity -> Confidentiality is not correct

5a) You need to check if the right is available in p before u copy over to q.

5b) You need to check if the right is available in p before u copy over to q.

6a) Subjects will be able to grant other subjects rights even if it itself does not possess those rights.