# IMT4123 - Assignment 1

—

January 8, 2022

## Task 1

For each of the following statements, give an example of a situation in which the statement is true.

1. Prevention is more important than detection and recovery.

   **Example:** An simple example can be when we want to prevent unauthorized access to a system. Implementing a mechanism like a login password for the user is a simple way to handle this. In this case, we care more about stopping unauthorized users from entering our system and getting information that is not meant for them, rather than detecting the user.

2. Detection is more important than prevention and recovery.

   **Example:** Continuing with the idea of a system with password protection, let us say there is a detection mechanism checking for the number of tries a user types his password. In this case, we might not want to prevent the user from logging in, but we would like a log of the number of entries and perhaps an error message so that the admins can investigate the incident further if need be. Detection is good for cases where we want to monitor attacks that cannot be prevented.

3. Recovery is more important than prevention and detection.

   **Example:** Recovery would be highly important if we have a system with a lot of stored data. We cannot assume that we will not be attacked or that we can detect every attack, thus we need to use recovery. If we are attacked by ransomware a recovery mechanism would be important to recover the data from a backup as soon at is assessed safe to do so, in order to keep the business operational. Recovery also involves identifying the attack and attacker and fixing the vulnerabilities in the system to prevent further attacks of the same type.

## Task 2

Identify mechanisms for implementing the following. State what policy or policies they might be enforcing.

1. A password changing program will reject passwords that are less than five characters long or that are found in the dictionary.

   **Answer:** The mechanism is the password changing program which rejects the password that does not meet the password policy's requirements. The policy states that passwords that are less than five characters long or are found in the dictionary shall be rejected.

2. Only students in a computer science class will be given accounts on the department's computer system.

   **Answer:** The mechanism enforcing the policy is the procedure of deciding who gets accounts on the computer system. The policy enforced is that only students in a computer science class can use the departments' computer system.

3. The login program will disallow logins of any students who enter their passwords incorrectly three times.

**Answer:** The mechanism enforcing the policy is that the accounts of students will be disabled if they incorrectly enter their password three times. The policy is enforcing that only authorized students can log in to the computer system. It does however also lock out users who forget their passwords.

4. The permissions of the file containing Carol's homework will prevent Robert from cheating and copying it.

   **Answer:** The mechanism is the access control that restricts access to the system user's files according to the policy. The policy is so that a student cannot read another student's file, in this case, homework.

5. When World Wide Web traffic climbs to more than 80% of the network's capacity, systems will disallow any further communications to or from Web servers.

   **Answer:** The mechanism blocks the traffic to or from the WWW servers when the given threshold is reached. The policy enforces a threshold of 80% of the network's capacity so that the WWW traffic does not affect the other traffic of the system.

6. Annie, a systems analyst, will be able to detect a student using a program to scan her system for vulnerabilities.

   **Answer:** The mechanism is the system, program, etc. which Annie uses to detect the scanning. The policy is that systems cannot be scanned for vulnerabilities by students.

7. A program used to submit homework will turn itself off just after the due date.

   **Answer:** The mechanism is that the program used for submitting homework will disable itself after the due date. The enforced policy is that homework is not accepted after the due date.

## Task 3

A Policy restricts the use of electronic mail on a particular system to faculty and staff. Students cannot send or receive electronic mail on that host. Classify the following mechanisms as secure, precise, or broad.

1. The electronic mail sending and receiving programs are disabled.

   **Answer:** Precise. Given the definition of precise mechanism saying that a mechanism is precise if **R** is equal to **Q**. We see that no mail can be sent or received, thus **R** is equal to **Q**.

2. As each letter is sent or received, the system looks up the sender (or recipient) in a database. If that party is listed as faculty or staff, the mail is processed. Otherwise, it is rejected. (Assume that the database entries are correct.)

   **Answer:** Secure. Given the definition of secure mechanism saying that a security mechanism is secure if **R** (some) is included in **Q**. Seeing as that some of the users are allowed to send mail given they are in the database, we can see that this is a secure system.

3. The electronic mail sending programs ask the user if he or she is a student. If so, the mail is refused. The electronic mail receiving programs are disabled.

   **Answer:** Broad. Given the definition of broad mechanism saying that a mechanism is broad if the system can go to non-secure and secure states. We can see that which state the system takes depends solely on the user's input. There is no factual check to see if the users are who they claim to be, thus a student can claim to be staff and enter a non-secure state. Or vise versa.

## Task 4

Consider a very high-assurance system developed for the military. The system has a set of specifications, and both the design and implementation have been proven to satisfy the specifications. What questions should school administrators ask when deciding whether to purchase such a system for their school's use?

**Answer:**

- Does the system primarily focus on confidentiality?
- If so can they handle the possible loss of integrity or availability?
- At what level do they trust their users?

Given that they are a school, looking into a system that satisfies the specification of a commercial security policy instead as the integrity of the students and faculty's documents and information might be more important to prevent tampering of data. The level of trust is the most important question for the admins as it dictates the level of integrity and confidentiality policies that are needed for the system.

## Task 5

Consider a computer system with three users: Alice, Bob, and Cyndy. Alice owns the file alicerc, and Bob and Cyndy can read it. Cyndy can read and write Bob's file bobrc, but Alice can only read it. Only Cyndy can read and write her file cyndyrc. Assume that the owner of each of these files can execute it.

**Assumption:** The task text is very specific in its wording of what rights the users have on the files. I assume that the users are not allowed to read or write to their own files as it is never specified that the owners are allowed to read or write their own files, but the task text is very specific in that the owners can execute their own file.

1. Create the corresponding access control matrix.

| User \file | alicerc | bobrc | cyndyrc |
|---|---|---|---|
| Alice | ox | r | - |
| Bob | r | ox | - |
| Cyndy | r | rw | orwx |

2. Cyndy gives Alice permission to read cyndyrc, and Alice removes Bob's ability to read alicerc. Show the new access control matrix.

| User \file | alicerc | bobrc | cyndyrc |
|---|---|---|---|
| Alice | ox | r | r |
| Bob | - | ox | - |
| Cyndy | r | rw | orwx |

## Task 6

Consider the set of rights {read, write, execute, append, list, modify, own}

1. Using the syntax discussed in the lecture, write a command delete_all_rights (p, q, s). This command causes p to delete all rights the subject q has over an object s.

   **Assumption:** As sub-task 2 and 3 both state the if-clauses in the task description, I presume that we are not to take into account the general rule that only the owner of the object can modify it; hence I have not included such an if-statement for this command.

```
Command delete_all_rights(p, q, s)
    delete r from A[q, s];
    delete w from A[q, s];
    delete x from A[q, s];
    delete a from A[q, s];
    delete l from A[q, s];
    delete m from A[q, s];
    delete o from A[q, s];
end
```

2. Modify your command so that the deletion can occur only if p has modify rights over s.

```
Command delete_all_rights(p, q, s)
if m ∈ A[p, s]
then
    delete r from A[q, s];
    delete w from A[q, s];
    delete x from A[q, s];
    delete a from A[q, s];
    delete l from A[q, s];
    delete m from A[q, s];
    delete o from A[q, s];
end
```

3. Modify your command so that the deletion can occur only if p has modify rights over s and q does not have own rights over s.

   **Answer:** As it is not possible to test for the absence of rights, we have to be a bit clever. A workaround (based on the coursebook: Computer Security: Art and Science) is to create a temporary object **z** so that A[q, z] will contain the right **o** if **q** does not have **o** rights over **s**, and will contain the right **m** if **q** has **o** rights over **s**. To make this work we need to create some additional commands. I will do this in three steps:

   (a) Make a create command for the object **z**.

   (b) Make a modify command for the object **z**.

   (c) Make a delete command for the object **z**.

   (d) Gather the commands in delete_all_of_rights

   Step a: Making the create command for the object **z** and grant **q** ownership.

```
command create_sub_object(p, q, z)
    create object z;
    enter o in A[q,z];
end
```

   Step b: Making the modify command for the object **z** that deletes **q**'s ownership of **z** and grants **q** modify right of **z**.

```
command modify_sub_object(p, q, s, z)
if o ∈ A[q, s]
then
    delete o into A[q, z];
    enter m into A[q,z];
end
```

   Step c: We will also need a command to delete the rights from **z** if **p** has **m** rights over **s** and **q** does not have **o** rights over **s**.

```
command delete_rights(p, q, s, z)
if m ∈ A[p, s] and o ∈ A[q, z]
then
    delete r into A[q, s];
    delete w into A[q, s];
    delete x into A[q, s];
    delete a into A[q, s];
    delete l into A[q, s];
```

```
        delete m into A[q, s];
        delete o into A[q, s];
    end
```

Step d: At last we can create a "common" command where we sum up all the needed commands to circumvent the fact that we cannot test for the absence of rights:

```
    command delete_all_of_rights(p, q, s, z)
        create_sub_object(p, q, z);
        modify_sub_object(p, q, s, z);
        delete_rights(p, q, s, z);
        destroy object z;
    end
```

## Task 7

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

1. 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)?

   **Answer:** As the principle of attenuation of privilege states that; *"A subject may not increase its rights, nor grant rights it does not possess to another subject"* (Computer Security Art and Science, M. Bishop 2019, p.43), not applying the principle does mean that a subject in the system can grant whatever right it wants to whichever subject it want. This gives that the maximal set of rights that subjects within the system can acquire is all the rights in the system over all the subjects. Some rights in the system might only be granted given specific conditions. If this is the case, then the maximal set of rights depends on these conditions.

2. 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:** No. In this case, the attenuation of privilege applied would only control the transfer of rights in the system. Thus there is no control mechanism in place to make sure that owners cannot grant rights they do not possess e.g. they can grant whichever right they want.

3. 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 means that **q**'s access is limited by the access his ancestors have for their files. This implies that **q** will have more access, on a global scale in the system, than any of his ancestors as he will have access to the files of **q1**...**qn**, while for example **q1** only has access to his files, and not necessarily access to the same files as **q2**. Simply put, it is a "access-up" system where there is less access the higher in the system you are, thus limiting a user like **q1** to e.g. read files that might be read by **q3**.

| User/File | f1 | f2 | f3 |
|---|---|---|---|
| q1 | r | - | - |
| q2 | r | rw | - |
| q3 | r | rw | rwx |
| q4 | r | w | rwx |

In the table we see that **q1** (first ancestor) has access to **f1**, but not **f2**, and **f3**. The rights **q1** has for **f1** are transferable to **q2-qn**. To illustrate that a user **q** might choose to not have all the rights to a file I have given **q4 w** access for **f2**. As seen by the matrix it is evident that **qn** (newest user) will/can have access to more files than his ancestors, but his rights over these files are restricted to his ancestor's rights over set files e.g. **q3** cannot suddenly get **w** access for **f1**.

## Task 8

Let c be a copy flag and let a computer system have the same rights as in task 6.

1. Using the syntax discussed in the lecture, write a command copy_all_rights(p,q,s) that copies all rights that p has over s to q.

   **Answer:**

   ```
   command copy_all_rights(p, q, s)
   if r ∈ A[p, s]
   then
       enter r into A[q, s];
   if w ∈ A[p, s]
   then
       enter w into A[q, s];
   if x ∈ A[p, s]
   then
       enter x into A[q, s];
   if a ∈ A[p, s]
   then
       enter a into A[q, s];
   if l ∈ A[p, s]
   then
       enter l into A[q, s];
   if m ∈ A[p, s]
   then
       enter m into A[q, s];
   if o ∈ A[p, s]
   then
       enter o into A[q, s];
   end
   ```

2. 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 r ∈ A[p, s] and c ∈ A[p, s]
   then
       enter r into A[q, s];
   if w ∈ A[p, s] and c ∈ A[p, s]
   then
       enter w into A[q, s];
   if x ∈ A[p, s] and c ∈ A[p, s]
   then
       enter x into A[q, s];
   if a ∈ A[p, s] and c ∈ A[p, s]
   then
       enter a into A[q, s];
   ```

```
if l ∈ A[p, s] and c ∈ A[p, s]
then
    enter l into A[q, s];
if m ∈ A[p, s] and c ∈ A[p, s]
then
    enter m into A[q, s];
if o ∈ A[p, s] and c ∈ A[p, s]
then
    enter o into A[q, s];
end
```

3. In part (b), what conceptually would be the effect of copying the copy flag along with the right?

   **Answer:** Copying the copy flag would conceptually mean that the ability to give rights is transferred as the copy flag is implemented to stop indiscriminately giving away rights.

## Task 9

Prove the following lemma

**Proof:** Using the theory of sharing rights and tg-paths we can prove that if **x** and **z** are objects, while **y** is a subject, the lemma is true if:

1. **x** creates (tg to new vertex) **v**.
2. **x** grants (g to **v**) to **z**.
3. **z** grants ($\alpha$ to **y**) to **v**.
4. **x** takes ($\alpha$ to **y**) from **v**.

This sequence of rule adds an edge labeled $\alpha$ from **x** to **y** as seen in figure 1.

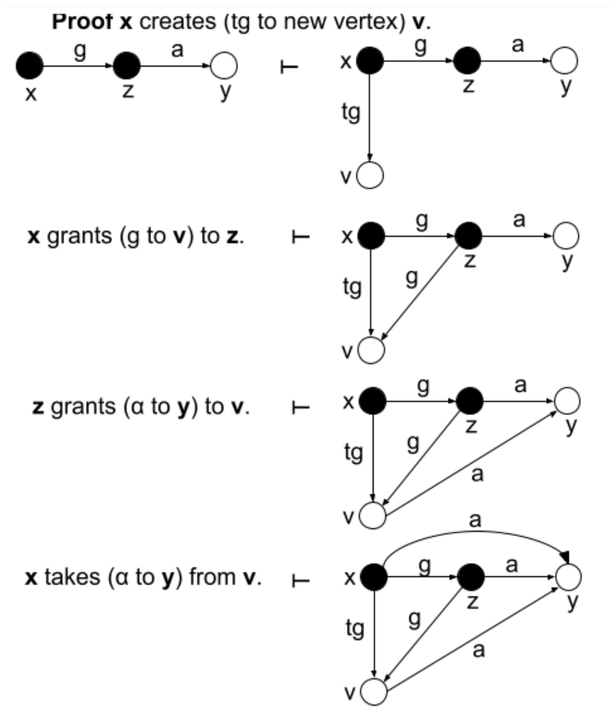

Figure 1: Lemma proof

## Task 10

Prove or disprove the claim of the lemma of task 9 holds when x is an object.

**Proof:** Given that only subjects can give and grant rights, or create objects, and given the directions of the edge from x to **z** and **z** to **y**, we see that it is not possible for **z** to transfer ($\alpha$ to **y**) to **x**. We can try to use a tg-path to a new object **v**:

1. **z** creates (tg to new vertex) **v**.
2. ... and nothing.

As there are no other subjects to take or grant rights to or from **v**, we have no way to transfer the rights as seen in figure 2.

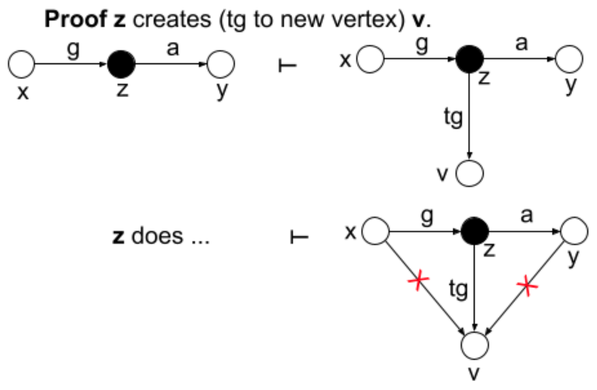

Figure 2: Lemma proof

## Task 11

The proof of Theorem 3.1 states that we can omit the delete and destroy commands as they do not affect the ability of a right to leak when no command can test for the absence of rights. Justify this statement. If such tests were allowed, would delete and destroy commands affect the ability of a right to leak?

**Answer:** The analysis mentioned in the proof is based on the fact that they can check for the changes that occurred during the sequence. As we are not able to check for the absence of rights it means that we cannot evaluate these two commands. Furthermore, the primitive commands delete and destroy cannot leak rights as they have no function to transfer them, only remove as given by their syntax, given that the leakage of right is defined as the transfer of rights. Hence, testing for the absence of rights should not lead to leakage.

```
Primitive Command: delete r into A[s, o]
Primitive Command: destroy subject s
Primitive Command: destroy object o
```