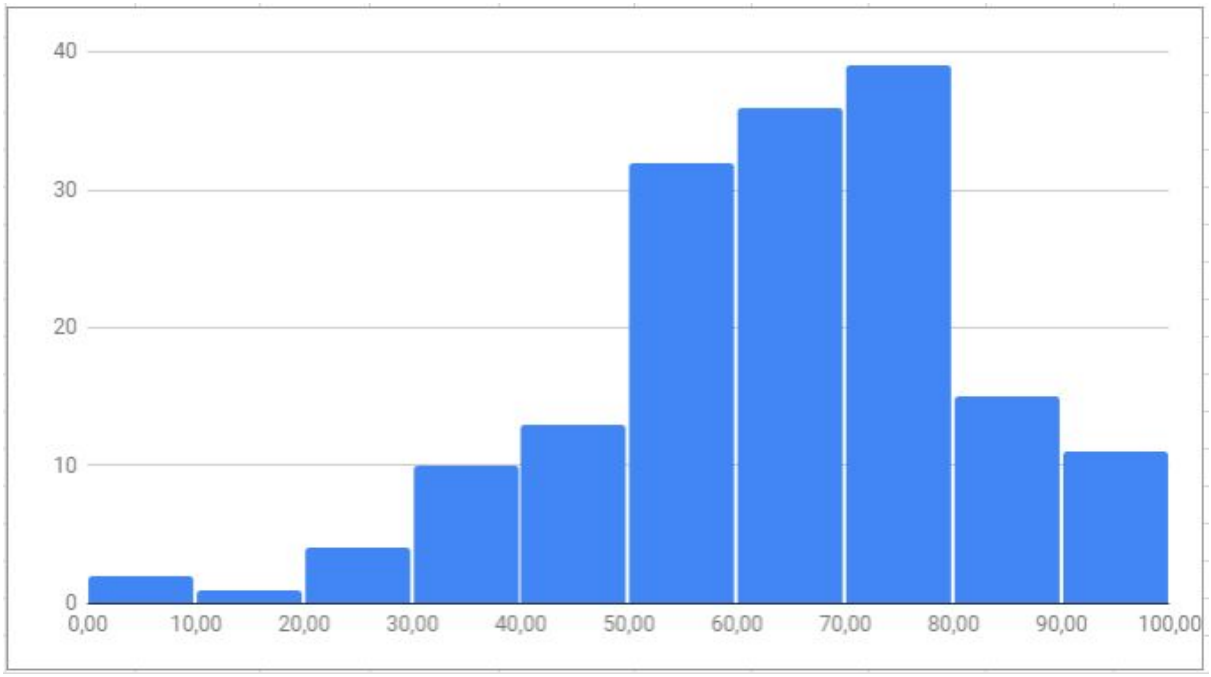
# COM-301 : Computer Security
# MIDTERM - 25th October

## Grades distribution in Midterm




**(Note that there was only one 100 and one 98 above 95)**

- **Between 0-50: You really need to improve**. If you find concepts hard, come to the exercise sessions, come to office hours, and/or ask in the forum.
- **Between 50-65: You need to push a bit more.** Think about coming to office hours or asking in the course forum to fix the concepts that are not fully clear.
- **Between 65-80: You are keeping up,** but you need to keep working to maintain the level.
- **Between 80-100: You will probably do ok**, but do not lower your guard.

# MREs

## (Most Repeated Errors)

## General Advice

- Writing more than what is asked is not a good strategy. If it is right, we will not give more points. In fact, it is running a risk. We did subtract point for non asked inputs that were fundamentally wrong.
- When you are asked to choose a property, an approach, identify one technology, do not list all possible ones. The goal of the question is understanding the properties and to choosing the suitable one rather than just memorizing every property.

### Q1 Security principles

**MRE1:** Mixing security principles and security properties**.** Many answers to this questions referred to violation of Confidentiality, Integrity, or Availability. These are not principles, these are properties!!!
The properties are part of the security policy. What property we want to achieve for each asset.
The principles are guiding propositions that serve as the foundation for a security-oriented chain of reasoning. They help making decisions on how to build secure systems so that the security policy (i.e., the properties), are fulfilled.

### Q2 Access Control Matrix

**MRE1:** When writing the ACL, respectively the Capabilities, keep a space for void permissions.Examples:

ACL
    xxx.txt : {Alice:{read}, Bob:{}, Charlie:{read,execute}}
Capabilities
    Alice = {xxx:{read}, yyy:{write}, zzz:{}}

Keeping the void permissions defeats one of the purposes of using an ACL instead of the matrix, namely reduce space to make it efficient. (https://moodle.epfl.ch/pluginfile.php/2309922/mod_resource/content/2/COM-301%20-%20Access%20control%20-%20handouts.pdf, slide 27).

Note: for those of you that instead of void permission set a negative permission, keep that one in the ACL/Capabilities was considered correct. In this case one cannot save space by not having the negative permission, since it needs to be used when checking access control.

**MRE2:** When writing the ACL, respectively the Capabilities, not adding the principals, respectively the objects. Examples

ACL
    xxx.txt : {{read}, {write}, {read,execute}}
Capabilities
    Alice = {{read}, {write}, {read,execute}}

If the principals/objects are not in the access control mechanism, how is the system supposed to know to which element do permissions refer to? You cannot assume that the system knows and/or has this in memory. First, this would require that the void permissions are kept in order to match the list of principals/objects. This implies that, as in the previous error, you do not save space and are equally inefficient and error-prone as in the full matrix.

## Q3 Permissions

**MRE1:** Opening a file does not necessarily require a read permission. It depends on the mode in which it is opened. If a file is opened in append mode, you have to look at the 'w' permission, not the 'r'.

**MRE2:** Write and execute permissions mean different things. Write permissions allow you to modify a file. Execute permissions allow you to run a file. Having execute permissions does not imply that malicious code can be written in the script and run (you would also need write permission for that).

**MRE3:** If the question asks about configuration issues when a particular user (for example Charlie or Bob) executes a script, frame your answer from the point of view of that user. For example, in this scenario, if we consider a user Charlie belonging to 'others':

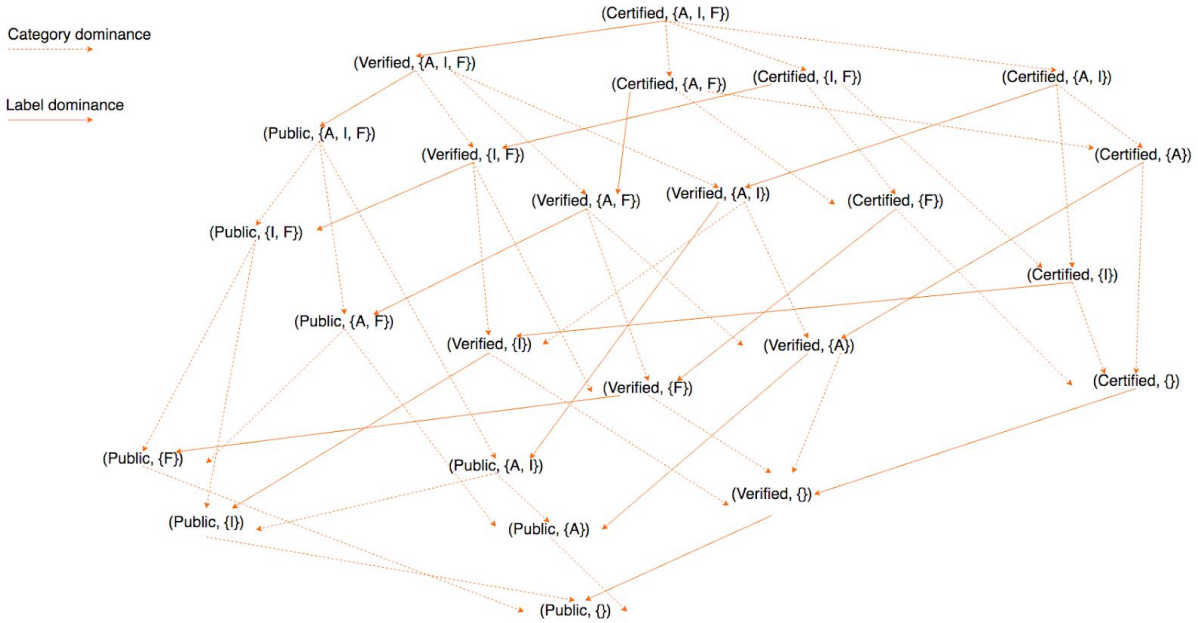
```
-rwx--x--x   Alice Alice+Bob msg
-rwxrw---x   Alice Alice+Bob msgfile
```

It is indeed correct that a member of Alice+Bob can modify msgfile directly. However, it is incorrect to state that a member of Alice+Bob can overwrite Charlie's messages in msgfile. The configuration does not have write permission for Charlie for msgfile, which means that Charlie cannot even write a message to msgfile in the first place (which is the main problem with this configuration if we ask about Charlie).

**MRE4:** Members that come under the category of 'others' are those that are neither in the 'user' nor the 'group' category. That means that permissions to others do not apply to the members of the group. The OS checks 'others' permissions only if the principal is not the owner or a member of the group.

## Q4 BIBA and BLP

### Building the lattice:




**MRE1:** When we ask for subject classification, it means both the classification label and category, not just the classification label. Example: (certified, {administration, finance, investments}), not just certified alone.

**MRE2:** Same point as above for when we ask whether a principal can read/write a file. Procedure to answer such questions: 1. Determine if there is a dominance relationship between the principal and the file using both classification label and category (using the rules learned in class) and establish what is dominant. 2. Look at the rules of reading and writing based on whether it is BIBA or BLP. 3. Apply the rule to the scenario.

**MRE3**: In BIBA, the classification that gets the highest privilege for reading is (public, {}), not (public, {administration, finance, investments}). A classification of (public, {administration, finance, investments}) cannot read (public, {administration}) for example (No read down in BIBA). Similarly, in BLP, the classification that gets the highest privilege for writing is (public, {}), not (public, {administration, finance, patient}). A classification of (public, {administration, finance, patient}) cannot write (public, {administration}) for example (No write down in BLP).

## Q5 Hash functions

**MRE1:** Many answers just stated all the hash properties. That was not the goal of the exercise. The idea was to think about which of the properties are necessary for each particular case and only provide those. (we did not subtract points for other desirable functionality/properties that were correct such as: use a slow hash for passwords).

**MRE2**: “Hashes for signatures need pre-image resistance”. This is incorrect. The hash usually goes together with the message, so there is no need to hide the message itself. In fact, to check the correctness of the signature the receiver needs to produce the hash, i.e., the message is known.

## Q6 PGP

**MRE1:** Using OTP for encryption. Instead of using a secure encryption Enc to encrypt the OTP key which is as long as the message and sending both the OTP key and the OTP encrypted message to the receiver, the system can directly use Enc to encrypt the message. In general, using an OTP is never the best option and it should not be used in reality.

**MRE2:** Specifically using a stream cipher instead of symmetric encryption. There was no penalty for writing stream cipher as an example of symmetric encryption. The picture only shows symmetric encryption and it doesn’t specify stream or block cipher, but PGP uses block ciphers for symmetric encryption. In practice, block ciphers or more popular than stream ciphers because they have better diffusion.

**MRE3: “**PGP is asymmetric encryption”. This is incorrect. PGP is hybrid encryption which uses both symmetric and asymmetric encryption.

**MRE4:** Hash, CRC, or parity provides integrity. This is incorrect. These functions are deterministic and they don’t have any secret input. They can protect the message against noise, but an adversary can easily recompute the hash after changing the message, so they don’t provide integrity.

**MRE5:** Using MAC to get integrity in PGP. MAC requires a pre-shared key to work. In PGP the sender chooses a random key and encrypt it with the receiver’s public key, so the key is not pre-shared. An adversary can intercept the message and replace it with a new message with a new mac key. It’s important to note that the adversary cannot read the message, but he can replace it with a new one. The receiver, gets an email which uses a random MAC key. There is no way to link this random key to the identity of the sender. **Note**: You cannot assume a pre-shared MAC key between the sender and the receiver. The main reason behind the PGP is that there are two people who want to communicate without having any shared key.

**MRE6:** Building your own MAC! The idea behind HMAC is hashing a message with a secret key, but the actual primitive is not as simple as H(key|message). **Writing that you could use H(key|message) as a HMAC is actually designing your own cryptographic primitive!!!** MAC is not a suitable answer for this question and gets zero point either way, but even if it was the correct answer you would have faced a penalty.