Most Repeated Errors in COM-301 Midterm 2024

General Feedback

1. Be careful about what is asked and answer only this, e.g., do not answer a fix if the question is asking “what is the vulnerability”.

Access Control & Security Principles

Q1

- **Applying setuid to the document (hybris_plans.dwg) rather than the program.** Setuid applies to executables not to (non-executable) data files.
- **Incorrectly writing the setuid permission.** The correct notation is -rwsr-xr--. Other notations were only given partial points.
- **Giving *write* access to everyone to the document (hybris_plans.dwg)** and consequently removing any notion of access control on that file which makes the configuration insecure. This violates numerous security principles.

Q2

- **Claiming that least privilege is fulfilled.** The principle states that principals should be given only the rights necessary to perform their tasks. However, managers (other than the CEO, and unrelated to the designer team) can also execute the program, a violation of least privilege.
- **Giving a wrong justification to why it’s not fulfilled.**
    - Some students confused the least privilege principle with other principles and thus gave a **mismatched justification.**
    - **Arguing about the security** of the configuration rather than the principle, e.g., world-writable sensitive documents is insecure hence violates the least privilege.
    - **Arguing that Dave should not have some permission as the owner.** But Dave is also part of the designer group, with legitimate access to the design files. While the idea is correct only partial points were given to this justification.

Q3

- **Justifying outside the emergency deletion procedure**. E.g., some argue that fail-safe default is not followed since the plan will leak in the case where they fail to detect a spy. This justification of the fail-safe default principle is not about the mechanism or procedure. Imagine the emergency deletion procedure as an algorithm, it needs a one-bit input signal: a detected spy sets the input to 1, otherwise 0. A failure of *spy detection* means the algorithm receives the input signal as 0, and the algorithm does not delete anything. From the algorithm perspective, no deletion is the expected output. In other words, a failure of spy detection is NOT a failure of the emergency deletion procedure.
- **Justifying the principle narrowly from the specific example given in the lecture.** E.g., some argue the deletion is based on the detection of a spy, which is blocklist based, instead of allowlist based. This justification only receives partial points, because it is unclear how this specific scenario can use an allowlist based method: does it really make sense to say that the design document should only be created under certain conditions, otherwise it shall not be created as a file? How can a company operate in this case?
- **Not addressing the specific scenario of this question**. Some people argue that fail-safe default is not followed because the system does not by default go back to a "safe state" without further explanation. What is the "safe state" in this emergency deletion scenario? What is a "failure" when it is about fail-safe default? Paraphrasing the lecture materials without addressing the scenario of this question receives no point.
- **Confusing complete mediation with the best-possible or finest-granular authorization check.** To quote from the lecture, complete mediation indicates that *every* access to *every* object must be checked for authority. The emphasis is on the completeness of checking; however, it says nothing about whether the check itself is "good" or "fine grained" enough to catch every possible problem. The fact that the file system in this question uses a UNIX-like permission system with a role for each user means that the access is indeed mediated. If the execution permission is given to the management group and Carla tries to run the program, the file system verifies that Carla, as a logged in user, is part of the group that has the execution permission.
- **Giving a mismatched justification for the chosen principle.** This mistake exists under every given principle. For example, some people argue that it is unnecessary to give permission to the whole management team given that Carla, the CEO, should be the only one who can delete. This argument itself is not wrong; however,

this falls into the least privilege principle (NOT separation of privilege), which is not among the list of given principles.

Mandatory Access Control

Q4

- **Not applying the theory to the question**: Most of the point deductions were because the answer only explained BLP and the No Read Up property, but didn't apply it to the problem at hand. It's critical for the answer to clearly show that one of the classifications (TS) dominates the other (U), and that the objects and groups concerned are associated to these classifications. You need to show that you can apply the theory to a given example, and not just recite the content of the class.

Q5

- **Proposing to use declassification as covert channel:** Most wrong answers refer to the declassification process as the covert channel used to transfer info from the Casting Directors to the Casting members. There is two main issues with such answers: (1) declassification process is set up by the MAC and by definition will hide sensitive info (it is not the user who decides how it's done), so, when declassified, the list of casting members will be empty, since the names are meant to be kept hidden from the unclassified level. (2) declassification by itself does not hide information in a valid message, which contradicts the definition of a covert channel. Answers that include covert channels like hiding messages in white space of other documents were given partial or full points depending on their argumentation.
- **Violating the No Write Down policy:** Many incorrect answers violate the no-write-down policy by allowing the Casting Directors from writing in the equipment list or other unclassified files. Answers mentioning a declassification through which the directors get files into the casting member's level, were given partial or full points according to whether the declassification holds in their case.
- Using a channel that is not covert:
    - **Using external communication channels without including covert codes in messages or with ambiguous modifications:** Many answers suggest communication channels outside the studio like (messaging, meeting in a coffee shop, …). While such channels would get the message from the directors to the members, it is not a covert channels (which we ask for explicitly) as the message is in the clear and not hidden in an allowed message by the MAC.

    - **Other notable mentions:** Using encryption as covert channel (check slides), subjects temporarily lowering their level to unclassified to write down (categorically disallowed in BLP), and implicit assumptions about the existence of a read timestamp log.
- **Incomplete descriptions of the covert channel:** Some answers lost partial points because they did not fully describe the covert channels. Examples of missing information: how to map the message to the list of names, and how to make the message not raise alarms by the MAC or other participants.

Authentication

Q6

- **Improper justifications**: Many answers referred to the "salt being in the clear" as being the sole reason of making the dictionary attack possible, salt do not need to be encrypted nor hashed. If a system does not use salts, then one can precompute hashes and attack the whole database; if a system uses salts, the attack should now be made per-salt (and hence, per-user) and is therefore more expensive to conduct for a whole database.
- **Not answering the question**: Many answers forgot to answer both questions: "how to do the attack" and "why is the attack feasible", answering only one of them. Some referred explicitly to a brute-force attack instead of a dictionary-based one.

Q7

- **Not properly including the salt**: many answers miss to include the salt in the entry. Doing so actually makes the hash unusable. To authenticate, one needs the salt to compute the hash to compare them. If the salt is not present, it's impossible to compute H(provided_password, salt).
- **Complex answers**: There is no need to have something more complicated than "H(pwd, salt), salt". Anything extra is unnecessary.

Q8

- **Saying the attacker can "reverse the hash":** It is incorrect because hashing is a one-way process. Unlike encryption, which can be decrypted with a key, hashing cannot be undone to retrieve the original input. In our case, it is then impossible to retrieve passwords like that.
- **Proposing to bruteforce or find the salt**: Many students argued about the fact that an adversary could find the salt and/or bruteforce it because it takes only 4 values. However, the goal in here what either to precompute it, or to use the salt directly

stored in the line of the password to crack. Hence, it was clearly not the goal of the attack and furthermore does not help the attacker.

- **Not being precise or incomplete description**: Answers only mentioning a dictionary attack were not given points since dictionary attacks are not feasible for any salt. To get full points, the answer must contain more details, for example the number of tables to pre-compute.

Cryptography

Q9

- **Answering "no" because the server public key is public:** while everyone can encrypt arbitrary values using the server's public key, only the sensors know the MAC key, and only the sensors are able to compute a valid MAC.
- **Forgetting to mention the key:** the MAC guarantees plaintext integrity because the key k is not known to the adversary.
- **Answering "no" because of replay attacks:** while replay attacks are possible, they do not influence plaintext integrity, and the replayed value is still a value measured by a sensor.
- **Brute-forcing the MAC:** some answers propose to record many MACs for different values, and (assuming that the number of possible values is small) want to brute-force the secret MAC key k. Even if the message space of the MAC is small and even given the fact that the MAC is deterministic, the key comes from a cryptographically large set, and a brute-force is infeasible (by construction!).

Q10

- **Claiming that we can use a public-key to decrypt a public-key emessage**. This is obviously wrong as we need the private key to decrypt such a cipher.
- **Incomplete justifications.** If argued that confidentiality holds, the justification had to include arguments for cipher encryption, MAC and hash (need to argue that each of them is secure wrt confidentiality).

Q11

- **Claiming that non-repudiation is the issue:** Having non-repudiation will not stop replay-attacks. Non-repudiation will ensure that the author of the message will not be able to deny writing it, but will not prevent replay attacks.
- **Claiming that the lack of acknowledgment from the server is the issue:** This is orthogonal to the problem described in the question. Having the server acknowledge the message from the sensors will not prevent replay attacks.

Q12

- **Incomplete answers:** When introducing a new component to the scheme, such as a unique id for each sensor, or a challenge, it needs to be clearly defined (Ex: *"…, where sensor_id is a unique id assigned to each sensor"*). Secondly, make sure to write the full scheme when proposing a solution. If only Enc(…) is explicit, we cannot assume what format you chose for the rest of the scheme (MAC, DS, Hash).
- **Using MAC with an asymmetric key:** MAC is a symmetric primitive. When using an asymmetric scheme, you have to use digital signatures for integrity.
- **Suggesting a nonce of counter without stating it has to be unique for each sensor:** When suggesting a challenge-response protocol for freshness, a challenge is always assumed to be unique for each message exchange with the server. However, when suggesting the use of a nonce or counter for freshness, you have to explicit the fact that it is unique for each sensor, or it will not thwart replay attacks (two messages from two different sensors could still look the same)
- **Suggesting a more granular time format to solve the issue:** Although the low granularity of the time field in the message makes the risk of collision between two different messages from two different sensors lower, having a very granular time field in the message will **NOT thwart** replay attacks. In this situation, if the server receives two identical messages, it will still not be able to tell if it is the same messages replayed or two identical messages from different sensors (even if it will happen with lower probability).

Q13 (No MRE :))

Q14

- **Wrong definition of second pre-image resistance:** Second pre-image resistance implies that given **input x**, it is hard to find another input x' having the same hash value. Many answers mentioned in the definition that adversary has not the input but only the hash value **h(x)** and can generate x' such that h(x) = h(x'). Note that, if adversary can do that it is a violation of first pre-image resistance, since violating first pre-image resistance *does not require* to reverse the hash function to its original value, any value in the input domain of the hash function should suffice. Such answers were granted only with partial points.
- **Putting the absence of collisions as the reason to have second pre-image resistance**: In some exam works, it was stated that second pre-image resistance is necessary to avoid collisions. This argument contains two errors. First, having second preimage resistance does not imply the absence of collisions, it only implies that it is computationally hard to find them for *a given input x*. The absence

of collisions is a much stronger property than the second preimage resistance, which normally does not hold for the hash functions since their input space is infinite, while output space is limited (i.e. a map from 2^1024 values to 2^512 values always have at least two input values mapped to the same output). Second, the presence of random or deliberate collisions is not a security issue within the given threat model (sending spam).

- Some students proposed an alternative threat model (impersonating other users, poisoning the database) where deliberately crafted collisions can bring benefits to some adversaries. Such answers were graded with full points.