Mini-Exam 2, Most Repeated Errors

September 15, 2021

General Advice

- Answer the question. If the question asks about principles, use the principles. If the question asks for an attack and mitigation; provide both. Wrong statements in other aspects may result in point reductions.
- Read the question carefully. If you are uncertain about details, spell this out clearly in your answer. For example, if you are not sure what are the assumptions about the threat model in the question, state in your answer how you interpreted the question.

Question: Leaking information and Covert Channels

Error 1: Provide a mitigation that does not prevent the leakage. Many provided a mitigation that instead of focusing on the leakage, focused on eliminating the impact of the leakage. For instance if the TAs would leak the correct answer (a, b, c, or d) in a multi-choice question, then shuffle the answers. While this solves the cheating problem, does not solve the leaking problem which is what was asked in the question.

Error 2: Providing a mitigation that renders the system useless. Many provided a mitigation consisting on either forbidding TAs from reading the exam questions or forbidding TAs from answering questions on Moodle. While indeed this would stop leakage, it also destroys the system. These tasks are part of the job of the TAs, and they must be able to perform them. This would be equivalent to saying that to avoid memory leaks in the computer, don't switch it on.

Question: BIBA model

Error 1: Breaking the functionality of the system. Some answers assigned cashiers and their daily earnings to the lowest level; and accountants/managers and monthly balance to a higher level; but did not provide a means for information from the cashiers to flow to the balance. While from a security perspective the isolation would do the job; this is not a suitable solution. Any answer that

would explain that information needs to flow but needs to be sanitized was accepted.

Error 2: Forget some object. Some answers forgot to assign levels to the daily earnings, and explain how they would be treated. As above, without this information the system cannot operate.

Question: BLP model

Note: The easiest way to solve this question is to draw a lattice diagram with the relationships you require in the system. This helps set the security levels for each entity.

Error 1: Expressing security levels and dominance relationships without using categories. Many answers used only the label (e.g., secret and public) and not both label and category (e.g., engineering and marketing) to set security levels. Some answers provided the categories but did not use them to determine security levels. A security level requires both pieces of information.

Error 2: For allowing both read/write, security levels of the subject and object should be the same. In BLP, if a subject needs to have both read and write access to an object, it needs to be on the same security level as the object. For example, engineers need to read and write source code. The same logic applies to the campaign document, where nobody should modify it (write access). This implies that it should be at the lowest security level (and other entities in the system should dominate it). If other parties are set at the same level, they will be able to modify it, unless the ds-property has been used.

Error 3: Confusing objects and categories. Some answers used the objects in the system as categories for the subjects. For example, an engineer's security level is (secret, source code), and there is no security level for source code. This does not provide information on what an engineer/other parties can do to the source code. It works only if source code is given its own category and assigned a security level.

Question: Diffie-Hellman exchange

Error 1: Assume Mallory can compute a discrete logarithm. The assumption that discrete logarithm is hard is central to many cryptographic protocols, including the Diffie-Hellman (DH) protocol. Unless a threat model explicitly includes computing the discrete logarithm, computing it should not be considered possible in your analysis. In other words, an attack that proposes to compute a discrete logarithm (in a group where this problem is hard) in general cannot be considered feasible.

Recall that to derive a shared secret g^{xy}, Bob computes $(g^x)^y$, using Alice's public key g^x. Assuming that the discrete logarithm is hard, an eavesdropping attacker cannot derive the shared secret when only observing g^x and g^y.

Question: Password updates

Error 1: Not justifying using the principles. Please remember to follow the instructions in the questions. If the question asks to "justify using the security principles", you must use the security principles to obtain the maximum grade.

Error 2: Unmatching principle and justification. Many responses seem to have picked a (sometimes unrelated) security principle first and then tried to fit the principle to an argument about a security property. This often resulted in a mismatch between the argument and the cited security principle. Please make sure that your justifications that use a security principle are coherent, and your argument is directly connected to the cited principle, if any.

Error 3: Incorrect use of 'privilege'. Many have used separation of privilege (or, related mistake, least privilege) as the principle supporting the password update policy: it separates the password cracking and the necessity to crack it within 3 months. A privilege is an unrelated concept referring to rights given to principals in a system. Technical terms such as "privilege" in computer security cannot be stretched out to mean arbitrary abilities.

Error 4: Lack of justification for Least Common Mechanism. Although least common mechanism can indeed be applied to this example, it was almost always unjustified ("common mechanism increases complexity" is not a concrete enough justification). An example of a better justification: an adversary can leverage the knowledge of the regular password-update policy to target the password update mechanism at a certain time (at the end of a 3-month cycle) to cause company-wide disruption.

Question: Unix Permissions

Error 1: Giving execute rights for txt and csv files The requirements of the system didn't specify the owner or employees should have execute rights over the txt and csv files. When not specified, it's better to follow the least privilege principle. Giving execute rights on a txt file makes it possible to run the content of the file as a script on the terminal, which in the context of the exercise is not needed.

Error 2: Putting the sticky bit at an incorrect position. The sticky bit should always be placed at the end of the line for a directory as follow : "drwxrwx–t owner employees dir". The 't' should replace the 'x' for others.

Error 3: Misuse of the setuid bit. When using the setuid bit, the 's' should replace the execute right of the owner, not the group : "-rws–x— owner employees program". Some students sometimes put the 's' in the position of the execute right of the group, which corresponds to the setgid bit. Some students also put the 's' on the execute right of the owner but for the csv inventory file instead of the program.

Question: Adversarial Thinking

Error 1: Unique daily song per TA Assigning a unique song to each TA per day is sufficient to prevent replay attacks while a specific TA is in the coffee lounge. However, as the question states, TAs might come back frequently so that as soon as a TA leaves the lounge the others can expect her to come back. This would allow a replay attack by an eavesdropping student at a later point. To avoid replay attacks, there would have to be a **fresh bit of randomness** every time someone enters the room. The problem with solutions that explicitly suggested to allow each song to be used only once per day is that this hampers the **utility of the system** as we were searching for a mechanism that allows TAs to come back frequently. This is another important aspect to keep in mind when designing security mechanisms.

Question: Guess my number

Error 1: Only proving no-cheating to Bob. The question asks "allows Bob to *publicly show* that Alice has cheated". Just knowing that Alice has cheated is not enough and Bob needs to show this in public (i.e., to 3rd parties). Therefore, using MAC between Alice and Bob does not solve the problem as MAC does not provide non-repudiation.

Error 2: Publishing hash/MAC(x) in the begging. Alice's number x is limited between 0 and 100. Even if the hash (or a MAC with a known key) is pre-image resistant, Bob can compute the hash for all 99 possibilities and compare them with the published hash, and learn x before guessing.

Question: Poem competition

Error 1: Listing properties instead of analyzing the problem. We asked for the minimum properties required for this specific scenario. Listing all hash properties does not solve this problem.

Error 2: You need to know a poem p where $h = Hash(p)$ to send h. If an adversary wants to send the same hash h as a good poet's submission and claims the poet's writing as his submission, then the adversary does not need to write a poem p' with the same hash as this p' will never be revealed.

Error 3: Accidental collision. The probability of two *random* message m and m' having the same hash $h = Hash(m) = Hash(m')$ without any adversarial tampering is negligible.