CORRECTED

1

**Teacher: Prof. Dr. ETH Mathias Payer**
**COM-402 Information Security and Privacy – Quiz 02**
**4th November 2024**
**Duration: 15 minutes**

# Anon Ymous

SCIPER : **999999**

**Do not turn the page before the start of the quiz. This document is double-sided, has 4 pages, the last ones possibly blank. Do not unstaple.**

- **No other paper materials** are allowed to be used during the quiz.
- Using a **calculator** or any electronic device is not permitted during the quiz.
- For each question, mark the box(es) corresponding to the correct answer(s). Each question has **one or more** correct answers.
- For each question, we give:
  3 points by default,
  0 points if you give no answer,
  −1 point per incorrectly checked or missed answer.
  Each question has a minimum of 0 points, we do not award negative points.
- Use a **black or dark blue ballpen** and clearly erase with **correction fluid** if necessary.
- If a question is wrong, we may decide to nullify it.

Respectez les consignes suivantes | Observe this guidelines | Beachten Sie bitte die unten stehenden Richtlinien

| choisir une réponse \| select an answer Antwort auswählen | ne PAS choisir une réponse \| NOT select an answer NICHT Antwort auswählen | Corriger une réponse \| Correct an answer Antwort korrigieren |
|---|---|---|
| ☒ ☑ ■ | ☐ | ☐ ☐ |

ce qu'il ne faut **PAS** faire | what should **NOT** be done | was man **NICHT** tun sollte

▨ ☒ ◯ ⊡ ▨ ☐

**Question 1**
Which of the following threats does connecting to a database via TLS (with a pinned certificate) mitigate?

- ☐ A malicious database admin logging into the database and exfiltrating data.
- ☐ A malicious router observing the database queries from recorded network traffic.
- ☐ An attacker controlling a malicious CA and generating a certificate to man-in-the-middle decrypt the TLS traffic.
- ☐ A thief stealing hard disks from the database server to extract database contents.
- ☐ An attacker implanting a backdoor into the database software.

***Explanation:*** TLS only provides transport security, i.e., no encryption at rest. Therefore, anybody controlling the database software, the database server itself, or has access to the hardware can still extract data from the database (assuming no other protection mechanisms are in place).
MitM attacks are mitigated by certificate pinning. This way, not even a malicious CA can create certificates that would allow decrypting the TLS traffic.

**Question 2**
Which of the following statements about memory safety is/are correct?

- ☐ Spatial memory safety ensures the object that is accessed within correct bounds.
- ☐ Temporal memory safety ensures the privilege check (read/write) for a particular object.
- ☐ A violation of spatial memory always causes a use-after-free vulnerability.
- ☐ Replacing `free` with garbage collection is one way to enforce temporal safety.
- ☐ Overhead of explicit bound checking through, e.g., SoftBound in C is cheaper than Java due to its static nature.

***Explanation:*** Spatial memory safety: objects are only accessed in bounds, Temporal memory safety: only valid objects are accessed. Use after free is problem caused by a violation of temporal locality. C is way more expensive than Java in terms of bounds checking around %64.

**Question 3**
Which of the following is/are mechanism(s) for ensuring thread-safe concurrent access to shared variables in "modern" programming languages? Correct usage of ...

- ☐ Dreadlocks
- ☐ Mutexes
- ☐ Semaphores
- ☐ Ownership borrowing
- ☐ Deadlocks

***Explanation:*** Shared memory is not thread-safe on its own unless all access to the shared memory region is protected by other thread safety mechanisms.
Mutexes and semaphores are synchronization primitives available in the standard libraries for many programming languages, and ownership borrowing is a Rust concept that ensures thread-safe access to variables (either a single writer or multiple readers, never writers and readers at the same time).
Deadlocks are not a synchronization primitive or mechanism but a result of the lack of thread-safety.

**Question 4**
Which of the following is/are true for injection-style vulnerabilities?

- ☐ Injection-style vulnerabilities arise if user-controlled input is concatenated as-is with program code and evaluated.
- ☐ SQL and command injection vulnerabilities generally both lead to privileged (i.e., root user) remote code execution.
- ☐ LDAP injection vulnerabilities can be exploited to bypass user authentication.
- ☐ Command injection vulnerabilities only affect web services.

***Explanation:*** Injection-style vulnerabilities arise when user-provided input isn't sanitized, escaped, or explicitly treated as a parameter (e.g., via prepared SQL statements) but concatenated with other code and evaluated.
SQL and command injection vulnerabilities don't necessarily lead to privileged remote code execution. This depends on the user that is running the vulnerable service.
LDAP is commonly used to store user information, and LDAP injection vulnerabilities can then bypass user authentication.
Commend injection vulnerabilities can also affect other services, for example, imagine a serverless-computing/function-as-a-service platform that runs user-provided code and allows the user to escalate to the host machine.

**Question 5**
Which of the following vulnerabilities is/are initially caused by a violation of temporal memory safety?

- ☐ Data race
- ☐ Type confusion
- ☐ Double free
- ☐ Use after free
- ☐ Garbage collection

***Explanation:*** "Garbage collection" is not a vulnerability. "Type confusion" is not related to temporal security, i.e. it is not directly caused by a specific order of events to happen. Corrected

**Question 6**
Which of the following statement(s) about database security is/are true?

- ☐ We should enforce access control at all database layers to comply with the principle of defense in depth.
- ☐ All databases encrypt data at the database layer because the operating system cannot be trusted.
- ☐ Data encryption together with access control is part of the principle of defense in depth in databases.
- ☐ SQL databases use `GRANT` in both discretionary access control and role based access control.

***Explanation:*** Having the database encrypt data before writing to files is better than having the operating system encrypt data before writing to disks, because in the first case, the data can only be seen by database users that have sufficient privileges or by administrators of the server that can dump the memory of the database, but in the second case, users of the operating system that have the right to access the database files can also see the data.

**Question 7** Which of the following statement(s) about types in the C programming language is/are correct?

- ☐ C has a strict type system
- ☐ C does not trust programmer-supplied type conversions
- ☐ Type confusion bugs can occur in C
- ☐ The type system of C guarantees type safety

***Explanation:***

- C has a type system that is relaxed, not strict
- C trusts programmer-supplied type conversions
- type confusion bugs can happen in C: for example, there are no checks for consistent use of union types
- the two above points imply that C does not guarantee type safety

**Question 8** Which of the following statement(s) about buffer overflow defense mechanisms is/are correct?

- ☐ An attacker that can infer the value of a stack canary could use a stack-based buffer overflow to overwrite the RIP and hijack control flow.
- ☐ DEP mitigates code injection attacks by removing the ability to execute certain memory regions.
- ☐ Suppose the executed program is very modular, and there exist several functions which, when called in a specific order/with specific arguments, can execute an arbitrary command. DEP can protect against an attacker using these functions to obtain a shell.
- ☐ ASLR randomizes (some of) the process' blocks of memory with a different address at *every* run of the program.

***Explanation:***

- Stack canaries rely on the randomness of the canary value to provide integrity of the RIP. If an attacker knows the canary, he can simply overwrite RIP *and* the canary, making sure its payload contains correct value for the canary.
- DEP removes the ability to e**X**ecute code from the `data` and `stack` memory regions amongst others. It enforces that any writeable region is non-executable.
- If an attacker knows all the addresses of small, modular regions of code (= "gadgets") it would need to execute arbitrary code, then, since these code regions are in the executable code of the program itself (`text` memory region) DEP by itself does not provide any protection against an attacker using them. This is the key idea behind Return-oriented Programming.
- By definition, ASLR randomizes blocks of memory at every process start

**Question 9** Which of the following statement(s) is/are true about compartmentalization and sandboxing?

- ☐ JavaScript engines are sandboxed inside Chromium to restrict access to sensitive data on the client.
- ☐ Sandboxes implemented at the language level cannot be escaped.
- ☐ Chromium compartmentalizes rendering engines for each tab by putting them in different processes, and communicate with the outside world through the OS kernel.
- ☐ Both compartmentalization and sandboxing can enforce principle of least privilege.

***Explanation:*** Sandboxes themselves may contain bugs, and numerous vulnerabilities have been found in the JavaScript sandboxes of browsers.

**Question 10**
Which of the following is an/are example(s) of a OWASP A01 "broken access control" vulnerability?

- ☐ A web application that allows non-admin users to execute admin functionality.
- ☐ A stack buffer overflow that overwrites the return address.
- ☐ Passwords stored in plaintext.
- ☐ A malware that hides its malicious payload inside a real application

***Explanation:*** allowing non-admin users to access admin functionality is a text book exmaple of broken access control all other examples do not apply