+1/1/60+

EPFL

1

Teachers : Prof. Payer
COM-402 Exam - MA
January 31st 2024
Duration : 180 minutes

Answer Key

SCIPER : **999000**

Do not turn the page before the start of the exam. This document is double-sided, has 24 pages, the last ones possibly blank. Do not unstaple.

Instructions

- This is a closed-book exam. Books, notes, and electronic devices are not allowed, except for up to two A4 pages of notes (this amounts to one double-sided A4 sheet or equivalent).
- No questions will be answered during the exam, except questions related to the understanding of the English language.
- Once finished, leave this document on the table.
- You can use the blank pages at the end as scratch paper. It will not be accounted for in grading.

For the **multiple choice** questions:

- There are 33 multiple-choice questions, each of which counts for 1 or 2 points.
- The number of points per question is shown next to the question. The number of points is not indicative of the number of right/wrong statements.
- *At least* one answer per question is correct.
- Each question offers 4 answers. The questions are graded according to the "all-or-nothing" principle, i.e., you only get points if you check *all* the correct and *only* the correct boxes.
- Make a mark **inside** the box corresponding to your answer. If you ticked the wrong answer, use white-out fluid or tape to erase the box completely. **Do not try to re-draw the box.**
- Use a black or blue pen to mark your final answers. **Pencils are not allowed.**

For the **open text** questions:

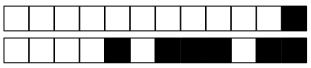
- There are 10 open-text questions, counting for 6-8 points each.
- Do **not** tick the numbered boxes at the top of the open-ended questions.
- Write your answers in the corresponding text boxes. We will ignore text outside of the boxes.

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

For your examination, preferably print documents compiled from auto-multiple-choice.



First part: Multiple choice questions

For each question, mark the box corresponding to the correct answer(s). Every question has **at least one** correct answer.

Question 1 (1 point)

Which of the following statements about Malware are correct?

- ☐ Worm = requires human help to spread from one system to another
- ☐ Malware = Malleable hardware
- ■ Trojan = hides in useful software
- ■ Ransomware = encrypts files and requests payment for decryption

Question 2 (1 point)

Hacktivism...

- ■ May involve publishing confidential data
- ☐ Always leaves an identifiable trace of responsibility
- ■ Is often politically motivated
- ☐ Requires a software exploit

Question 3 (1 point)

Extortion scams...

- ■ Convince the user they were "hacked"
- ☐ Are always targeted against handpicked individuals
- ■ May rely on leaked information (such as password lists)
- ☐ Require a software exploit

Question 4 (2 points)

Let $h : \{0,1\}^* \to \{0,1\}^{256}$ be a collision resistant hash function targeting 128 bits of security. Which of the following properties must the hash function satisfy?

- ■ First preimage resistance.
- ☐ Third preimage resistance.
- ■ Second preimage resistance.
- ■ Finding x, x' with $x \neq x'$ with $h(x) = h(x')$ must take at least 2^{128} bit operations.

Question 5 (1 point)

Tick all those that apply:

- ■ AES is a symmetric key cryptography algorithm.
- ■ RSA is an asymmetric key cryptography algorithm.
- ■ Asymmetric key cryptography includes encryption and signatures.
- ■ Symmetric key cryptography requires parties to share a secret key.

Question 6 (1 point)

Consider a Diffie-Hellman (DH) key exchange. Which of the following apply:

- [x] Security of DH relies on the discrete logarithm assumption i.e. that given g, p and $g^a \bmod p$ it is hard to find a.
- [x] DH key exchange with authentication and nonce is secure against MITM.
- [ ] Security of DH relies on the discrete logarithm assumption i.e. that it is hard to factor large composite numbers.
- [ ] A protocol that starts with DH cannot use symmetric cryptography afterwards.

Question 7 (1 point)

Apart from a key, what other inputs do common OTP generation algorithms (HOTP and TOTP) require?

- [x] An independent counter
- [x] A timestamp
- [ ] The ID of the device used for its generation
- [ ] The location of the device calculating it

Question 8 (1 point)

In Kerberos, what does the Ticket Granting System provide along with the encrypted session key?

- [ ] The nonce used for encryption
- [ ] The ticket granting ticket
- [x] The service ticket
- [ ] Nothing

Question 9 (1 point)

In the context of delegated authentication, which statement is accurate?

- [x] It allows one system to authenticate users on behalf of another system
- [ ] It is a concept applicable only in cloud-based authentication systems
- [ ] It is available only for UNIX/POSIX systems
- [ ] It is synonymous with biometric authentication and is widely used in mobile devices

Question 10 (1 point)

Which of the following statements about database security is/are correct?

- [x] It is better to have the database encrypt data before writing to files than the operating system encrypt the content of the files, due to the principle of least privilege.
- [ ] Discretionary and role-based access control are mutually exclusive. Using them together leads to redundancy and conflicting access permissions.
- [x] Access control may be implemented at both the database and application layers to ensure defense in depth.
- [ ] Role-based access control grants privileges directly to users instead of objects.

Question 11 (1 point)

Which of the following statements about passwords is/are correct?

- [x] The use of salt makes dictionary attacks less effective.
- [x] Salt should be added to the original user password before hashing.
- [ ] Using memory hard functions makes it impossible to crack a password.
- [ ] PAKE is designed for authenticating local parties over insecure channels.

Question 12 (1 point)

Which of the following statements about the rainbow table is/are correct?

- [x] It is hard to perform rainbow table attacks if different salts are used for different users.
- [x] Memory hard functions slow rainbow table attacks down.
- [ ] The rainbow table is proposed to mitigate the collision problem of hash functions in Hellman's original solution.
- [ ] A rainbow table with more columns requires more storage space for storing it.

Question 13 (1 point)

Which of the following statements are true regarding type systems of programming languages?

- [x] Type inference allows programmers to write fewer type annotations/declarations.
- [ ] A strict type system trusts programmer-supplied type conversions.
- [ ] A programming language with a dynamic type system implies it also has a relaxed type system (e.g., as is true for Python).
- [x] Programs written in statically-typed languages are oftentimes easier to debug than ones written in dynamically-typed languages because type errors are reported earlier.

Question 14 (1 point)

Which of the following is true regarding thread safety?

- [x] Thread-safety bugs in programs can raise security concerns.
- [ ] In Go, programmers must send data by reference in channels to avoid overhead and ensure thread-safety.
- [ ] In Rust, data races are prevented through ownership. An object can only be borrowed by one writer and multiple readers.
- [ ] In Java, each `synchronized` method of an object holds a different lock, allowing different methods to execute concurrently, while serializing execution when multiple threads execute the same method.

Question 15 (1 point)

Which of the following is **false** regarding compartmentalization and sandboxing?

- [ ] Compartmentalization applies the principle of least privilege, i.e., faulty components should not influence the whole system.
- [ ] Sandboxing a program often requires support from the operating system.
- [x] In Chromium, the sandboxed rendering engine runs as a separate process and interacts with the environment directly through arbitrary syscalls.
- [ ] A key challenge to utilizing compartmentalization effectively is defining components and policies clearly.

Question 16 (1 point)

A canary intends to protect against which of the following vulnerabilities?

- [ ] Double free
- [ ] OS command injection
- [ ] Memory Leakage
- [x] Stack overflow

Question 17 (1 point)

For XSS (cross-site scripting) vulnerabilities, in which context is the attacker's injected code executed?

- [ ] LDAP
- [ ] OS commands
- [ ] SQL
- [x] HTML/JavaScript

Question 18 (1 point)

You're exploiting a buffer overflow in a x86 64-bit function, whose C code and assembly are shown below.

```
1 void vuln() {
2     char buf[200];
3     gets(buf);
4     return;
5 }
```

```
1 push rbp;
2 mov rbp, rsp;
3 sub rsp,200;
4 lea rdi, [rbp-200];
5 call gets;
6 mov rsp,rbp;
7 pop rbp;
8 ret;
```

What is a possible structure of the exploitation payload that overwrites the return address of the vulnerable function?

- [ ] 204 bytes filler + 4 bytes return address
- [ ] 200 bytes filler + 4 bytes return address
- [ ] 200 bytes filler + 8 bytes return address
- [x] 208 bytes filler + 8 bytes return address

Question 19 (1 point)

Source code reviews are an essential part of testing because:

- [ ] They find all vulnerabilities.
- [x] They allow a deep inspection of the code.
- [ ] Can be fully automated for large code bases.
- [ ] Consider interactions with other software installed on any target system.

Question 20 (1 point)

Fuzzing is an established bug finding technique. Which of the following statements are true?

- [x] Greybox fuzzing requires lightweight instrumentation of the target program to collect coverage as feedback.
- [ ] To bypass hard checks such as cryptographic hashes, fuzzers call into a SAT solver.
- [ ] Fuzzing can find all bugs in an application.
- [x] Mutational fuzzing will automatically modify the input to trigger different parts of the program.

Question 21 (1 point)

Sanitization provides an oracle to identify bugs. AddressSanitizer in particular:

- [ ] Discovers unaligned memory accesses.
- [x] Incurs overhead of 1-3x, slowing down fuzzing.
- [x] Finds memory safety violations.
- [ ] Flags data races at particular addresses.

Question 22 (1 point)

Select all the attacks a VPN does **not** protect you against:

- [x] XSS.
- [x] Phishing.
- [x] SQL Injection.
- [x] Buffer Overflows.

Question 23 (2 points)

What are possible reasons why companies should avoid the use of WAF (Web Application Firewalls)?

- [x] WAFs require deep packet inspection with state tracking, considerably slowing down users' traffic.
- [x] The deep packet inspection WAFs perform increases attack surface, potentially enabling the possibility of memory corruption or other vulnerabilities while parsing packets.
- [x] WAFs which rely on machine learning models to detect attacks are susceptible to false positives. They are often outdated and will not detect the latest attacks.
- [x] When a single WAF protects multiple webservers, the WAF exposes a single point of failure (and attack surface).

Question 24 (1 point)

Select all the answers that mark one of the reasons why logging is useful in network security:

- [ ] Logs make the user able to recover data even after the event of a ransomware attack.
- [x] Automated tools can be setup to automatically scan logs to detect malicious behavior.
- [ ] Logs are always small in size, and they do not require additional computation resources, so there is no reason why not to have them.
- [x] Logs are useful to audits and helping incident response efforts.

Question 25 (1 point)

What primitives does Android use to prevent apps from accessing other apps' data?

- [ ] Containerization
- [x] Per-app user ids
- [ ] Disjoint file systems
- [x] SELinux

Question 26 (1 point)

Which of the following holds true for Android applications?

- [ ] Applications can directly communicate with other apps via shared memory
- [x] Applications are required to send requests to system services via the Binder IPC mechanism
- [ ] The application code is always easy to reverse engineer due to the Java bytecode
- [ ] Applications by default run as root, and the user has to manually restrict an app's permissions

Question 27 (1 point)

Which services run inside a Trusted Execution Environment (TEE) in Android devices?

- ☐ Bank transfer validation
- ■ Secure key storage
- ■ Biometric access validation
- ■ DRM mechanisms

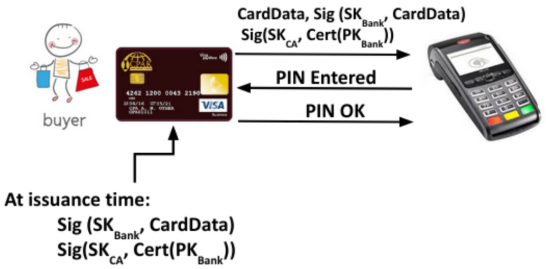
Question 28 (1 point)

Why does a Trusted Platform Module (TPM) require non-volatile storage?

- ☐ To store the values of Platform Configuration Registers (PCRs) used to measure trusted software.
- ☐ To store intermediate values while computing cryptographic operations.
- ■ To persistently store artifacts, such as keys or passwords.
- ☐ To store memory values read illegally during speculative execution.

Question 29 (2 points)

The graphic to the right illustrates the case of authorizing offline payments with a card using *Static Data Authentication*. Each option below comprises a problem, and a proposed solution. Select the options with valid problems for the scenario to the right, along with their correct solution (i.e., both statements must be true/match).



- ☐ **Problem**: "Yes Card", i.e. the fake card always replies OK to every entered PIN. **Solution**: Verify the card with Dynamic Data Authentication. A valid card can be trusted to later verify the PIN correctly.
- ■ **Problem**: Replay attacks. A cloned card can repeat the same info without knowing the secret keys (SK_{CA} or SK_{Bank}). **Solution**: Ask the card to sign terminal-generated nonces with its secret key with every round of interaction.
- ☐ **Problem**: Replay attacks. A cloned card can repeat the same info without knowing the secret keys (SK_{CA} or SK_{Bank}). **Solution**: Ask the card to sign a card-generated nonce with its secret key.
- ☐ **Problem**: "Yes Card", i.e. the fake card always replies ok to every PIN. **Solution**: Move the number pad for the PIN from the terminal to the card. The card reads the PIN, verifies it, and sends the result to the terminal.

Question 30 (1 point)

Below, we show the standard Spectre gadget which leaks data when executed speculatively with x >= 256, where the size of array1 is 256 elements.

```
if (x < 256) {
    secret = array1[x];          // load secret
    y = array2[secret * 4096];   // leak secret
}
```

A software mitigation for Spectre involves using a bitmask on the index, as highlighted in the code below.

```
if (x < 256) {
    secret = array1[x & 0xff];   // load secret
    y = array2[secret * 4096];   // leak secret
}
```

- [ ] The bitwise AND causes reordering of the two loads, so that the second load executes first without leaking any secret.
- [ ] KPTI is used to protect against Meltdown, not Spectre. The secret load should not be able to address kernel addresses, irrespective of KPTI.
- [x] The bitwise AND limits the range of speculative addresses within the array bounds, preventing access to secrets outside the array.
- [ ] The additional bitwise AND operation prevents speculation at the branch, preventing any speculative load (load secret step).

Question 31 (1 point)

Which of the following best describes a "Grey-box attack" in the context of machine learning security?

- [x] An attack where the model architecture is known, but the parameters are unknown, allowing limited interaction with the model.
- [ ] An attack where both the model architecture and parameters are known, and the attacker can replicate the model.
- [ ] An attack where neither the model architecture nor the parameters are known, leading to blind interaction with the model.
- [ ] An attack focused on compromising the confidentiality and privacy of the model's training or test data.

Question 32 (1 point)

In the context of Machine Learning as a Service, what is/are the primary goal of an adversary in a "Retraining attack"?

- [ ] To compromise the confidentiality and privacy of the user's query data.
- [ ] To identify the type of linear model being used by making a limited number of queries.
- [x] To "steal" the expensive model by observing its outputs and reconstructing its parameters.
- [ ] To disrupt the availability of the ML service by overwhelming it with queries.

Question 33 (1 point)

What is/are significant risk(s) associated with overfitted machine learning models?

- [x] Overfitted models are more susceptible to membership inference attacks, leading to potential leakage of training data.
- [x] Overfitting poses a risk to the generalization of the learned models, leading to low accuracy on the test dataset.
- [ ] They are fast to converge, leading to shorter training time.
- [ ] Overfitting in models results in increased inference-time computational costs, making them less efficient.

Second part: Open questions

Answer each question in the box below it. Your answer should be carefully justified. All steps of your argument should be discussed in detail. Leave the checkboxes empty, as they are used for grading.
Please write carefully and clearly. If we cannot read or understand your answer, we cannot grade it.
If you do not know a word in English, you may write it in another language, in quotes, with the language in parenthesis if it is not obvious, e.g., "Ordinateur" (FR).

Question 34: Cryptography (7 points)

This question is about key exchange and authentication. Let:

- $\mathbb{G}$ be a cryptographically secure group with $\mathbb{G} = \langle g \rangle$;
- $\mathsf{KDF} : \{0,1\}^* \to \mathbb{G}$ be a key-derivation function; and
- $\mathsf{MAC} = (\mathsf{MAC.Gen}, \mathsf{MAC.Verify})$ a message authentication code.

Let A, B be two parties each with, resp., keypairs $(\mathsf{pk}_A, \mathsf{sk}_A), (\mathsf{pk}_B, \mathsf{sk}_B)$. ($\mathsf{pk}$ = public key, sk = secret key)

Consider the following key-exchange protocol.

(A) A samples $x' \leftarrow \{0,1\}^{128}$, sets $x = \mathsf{KDF}(x', \mathsf{pk}_A, \mathsf{pk}_B, \mathsf{sk}_A)$, and sends $X = g^x$ to B.

(B) B samples $y' \leftarrow \{0,1\}^{128}$, sets $y = \mathsf{KDF}(y', \mathsf{pk}_A, \mathsf{pk}_B, \mathsf{sk}_B)$, and sends $Y = g^y$ to A.

(C) A computes $\mathsf{k} = \mathsf{KDF}(Y^x)$ and sends $\tau = \mathsf{MAC.Gen}(\mathsf{k}, \mathsf{pk}_A)$ to B.

(D) B computes $\mathsf{k} = \mathsf{KDF}(X^y)$ and accepts if $\mathsf{MAC.Verify}(\mathsf{k}, \tau, \mathsf{pk}_A)$ succeeds.

(E) A and B now share a common key k and B is convinced that it is talking to A.

This protocol is insecure. Explain an attack and discuss how the protocol can mitigate this threat.

0 .5 1 .5 2 .5 3 .5 4 .5 5 .5 6 .5 7

Answer. Solution: 1. At no point we are signing, so any party can play the role of A. 2. Even if fixing this, no nonce is ever sent, so a party can interact with A, and replay the interaction.

Grading. 4 points for attack, 3 points for fix (signatures + nonce).

Question 35: Access Control and Authentication (6 points)

A company's flexible internal network has an access control mechanism, the network is divided into subnetworks, each restricted to certain employees. The employees cannot grant their network permissions to other users. Inside a subnetwork, accessed only by the developer employees, its computer systems form a distributed environment which has a filesystem. This filesystem allows the developers to create and own files, but only the owner of the file can write to it. A file needs to be written by several developers. Each developer knows exactly who the next developer writing to the file is.

In this context, consider the following two scenarios:

- Scenario 1: the network access policy and the subnetworks (subnetworks are objects, users are subjects)
- Scenario 2: the distributed environment filesystem (files are objects and users are subjects)

a) Assign an access control policy (choose between MAC, DAC, and RBAC) to the system for each of the two scenarios below. (2 points)

b) For each scenario, justify your chosen policy through one pro and one con. (4 points)

0 .5 1 .5 2 .5 3 .5 4 .5 5 .5 6

Answer. Scenario 1: RBAC (MAC gives half the points, as the network is not flexible), Scenario 2: DAC. Pros and Cons:

- RBAC: Easy to grasp the idea of roles, easy to manage, easy to tell through roles which permissions a subject has and why. Difficult to decide on the granularity of roles. Role meaning is fuzzy. Unclear if roles can be shared across different departments
- DAC: Flexible. Easy to manage (owners get to set the permissions themselves). Intuitive. Depends on the owner's judgment. Only works if programs are not malicious and users make no mistakes. Vulnerable to the "Trojan"/declassification problem.
- MAC: Addresses the limitations of DAC. Easy to scale. Can be too restrictive, prevent legitimate tasks. Not flexible.

Grading. 1 point per scenario for the right policy. 1 point for each correct pro and con. Wrong pros/cons do subtract each 0.5.

Question 36: Data Security and PAKE (7 points)

Consider a hash function for storing passwords:

$$h(p[0:63]) = h1(p[0:31]) * h2(p[32:63])$$

Where,

- h is the hash function, taking a 64-bit password as input and generating a 60-bit hash value as output.
- p is the password to hash. All passwords have 64 bits. $p[0:31]$ takes the first 32 bits of the password.
- $h1$ and $h2$ are two hash functions, taking the first and the second 32-bit of the password as input separately and generating a 30-bit hash value as output each.
- The hashing result is given by the product of $h1()$ and $h2()$. There is no salt added nor multiple iterations of hashing when storing the passwords with h.

Now you plan to build a look-up table to crack passwords stored with h. Thinking of the time-memory trade-off, you would like to reduce the size of the table, even though the time needed to crack passwords is increased. Please consider how to build the table fully utilizing the property of the hash function h and answer the following questions. (*Hint: Building a rainbow table to achieve the trade-off does not fully utilize the property of* h.)

a) Describe the required look-up table, i.e., how would you construct this table? (2 points)

b) In total, how many hash operations (invocations of $h1$ or $h2$) are required to construct the table? (Just state the power of 2, not the specific value. Same for question c).) (1 point)

c) In total, how many hash-password pairs are stored in the look-up table? Explain. (2 points)

d) What is the procedure for cracking the password of a given hash value (h_0) with your look-up table? (2 points)

Answer.

a) Build two sub-tables of $h1$ and $h2$, by enumerating every possible 32-bit value and calculating its hash value. Store only 1 password pre-image for each hash value (because by construction, there will be collisions).

b) 2^{33}.

c) 2^{31}. Two sub-tables of 2^{30} rows each, indexed by the hash value of a corresponding password.

d) Factorize h_0, yield a pair of $h1_0$ and $h2_0$, repeat until find a hash match in the two sub-tables, then concat the two parts of the password to get the final result.

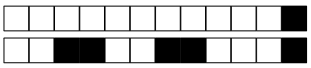
Grading.

a) Calculation: 1 point, storing: 1 point.

b) 1 point.

c) Number of pairs: 1 point, explanation: 1 point.

d) 2 points.



Question 37: Programming Language Security (8 points)

Answer the following questions regarding automatic memory reclamation techniques.

a) What kind of errors do automatic memory reclamation techniques (e.g., as used by Python or the JVM) seek to prevent? (1 point)

b) What safety do they achieve? (1 point)

c) Enumerate all three automatic storage reclamation techniques introduced in class, and name one language implementing the technique. (3 points)

d) Discuss the pros and cons of each technique. (3 points)

0 .5 1 .5 2 .5 3 .5 4
.5 5 .5 6 .5 7 .5 8

Answer.

a) E.g., use-after-free, double-free, memory-leak (1 point)

b) Temporal Memory Safety (1 point)

c) Reference counting: Python. Garbage collection: Java. Ownership: Rust. (3 points, 0.5 per technique and 0.5 per correct language)

d) Reference counting. Merit: easy to implement. Demerit: circular references cause leak and significant overhead. (1 point) Garbage collection. Merit: Less expensive than RC and collects unreachable circular structures. Demerit: pause program execution (stutter in user experience), and increased memory requirement (memory is not immediately freed). (1 point) Ownership. Merit: almost no overhead and prevents temporal memory errors entirely when writing safe Rust. Demerit: certain data structures like linked-list, trees, and circular references are either awkward or impossible to implement with just borrowing. In such cases, the programmer still needs to resort to reference counting (Rc, Arc in Rust) or manually manage memory (raw pointers and unsafe). (1 point)

Grading. In general, incomplete or partially correct answers get 0.5 points.
Mentioning memory safety without temporal or incomplete enumeration gets 0.5 points.
Correct mentions besides Python, Java, and Rust are also granted 1 point.

Question 38: OWASP and Buffer Overflows (5 points)

You have found a website with an SQL injection vulnerability. After playing around a bit you determine that the website has the following two SQL tables:

```
TABLE courses(id INTEGER, coursename VARCHAR, teacher VARCHAR)
TABLE users(id INTEGER, username VARCHAR, password VARCHAR)
```

You have also figured out that the vulnerable SQL query string is built like this:

```
query = ‘‘select teacher from courses where coursename = ‘ ‘‘ + userinput + ‘‘ ’;’’
```

You have full control over the contents of the `userinput` variable.

a) Please provide an SQL injection payload for `userinput` that selects the password for the user `gannimo` from the users table. (3 points)

b) After exploiting the vulnerability and disclosing it to the developers, the developers suggest implementing a complex regular expression to test if the `userinput` variable contains an SQL injection payload. What other alternative do the developers have to mitigate the vulnerability and why is this alternative potentially more secure? (2 points)

0 .5 1 .5 2 .5 3 .5 4 .5 5

Answer. one possibility: userinput = “asdf’ union select password from users where username=’gannimo’; – “ buggy regex, use prep. stmnts or escape all sql metacharacters.

Grading. 3 pt: correct SQLi payload. 2 pt: reasonable payload with union. 1 pt: injecting/union payload. 1 pt for either prep. statements or escaping. 1 pt for justification.

Question 39: Automated Testing (6 points)

Imagine you are writing a fuzzer for a library that exposes a broad API. In this scenario, analysts generally write so-called fuzz drivers that are similar to unit tests. Fuzz drivers call API functions with fuzzer provided input. Discuss two challenges when writing effective fuzz drivers, propose a solution for each and explain how it solves the challenge.

0 .5 1 .5 2 .5 3 .5 4 .5 5 .5 6

Answer. How the API is being called Fuzz drivers in general, how API is structured Mutation and data flow between API calls. 1 point per challenge, 1 point per solution, 1 point per explanation

Grading. Reasonable solution: 2 points Justification on solving the challenge: 3 points

Question 40: Network Security (6 points)

You recently rewrote your personal blog into a static website, and now you are in the process of deploying it. Given that there is no dynamic functionality (i.e., no input and data is ever sent or stored other than the page content), you are considering whether HTTPS is necessary.

Discuss whether to deploy HTTPS or stick with HTTP by providing three pros/cons in total (for example: two pros and one con). Justify.

0 .5 1 .5 2 .5 3 .5 4 .5 5 .5 6

Answer. Pros of using HTTPS:

(A) Authentication (an attacker might provide different page content)

(B) Privacy for the clients (the request is not public)

Cons of using HTTPS:

(A) Heavier request load

(B) More work for the sysadmin managing certificates

(C) Delay in key exchange and handshakes

(D) More attack surface (a static webserver can be implemented in 100 lines of C, HTTPS requires much more complexity, often using an outdated openssl library)

Grading. 6 points total. 2 points for each good pro/con. If the justification is incorrect, the pro/con awards only 1 point.
A pro/con that is not true awards -2 points.

Question 41: Mobile Security (6 points)

You're a member of the Android security team and are thinking of ways to improve (compared to the status quo) security of Android systems for all users. Describe and justify two policies/mechanisms to protect applications against exploitation (e.g., classic memory corruptions) or privilege escalation/misuse (e.g., accessing other applications' data).

0 .5 1 .5 2 .5 3 .5 4 .5 5 .5 6

Answer. Possible answers:

- Classic mechanisms like ASLR (which is currently kinda broken in Android) to reduce exploitability of memory corruption bugs
- Native code only written in memory safe languages (e.g., native libraries for Android apps)
- Syscall filters to restrict applications' capabilities
- Containerization/leveraging kernel namespaces

Grading. 3 points per policy/mechanism. 1 point for mentioning something that makes sense, 2 points for a valid justification.

Question 42: TEEs and Side Channels (7 points)

The naive RSA "square and multiply" algorithm implementation (shown below) leaves a power side-channel trace. The parameter `w` is the width of the key, with `d[0]` being the most-significant bit. Note: the star (`*`) denotes multiplication and the caret (`^`) denotes exponentiation.

```
 ========== RSA ==========
decrypt(c = ciphertext, d = decryption key, n = modulo):
    s_0 = 1
    for k = 0 to w-1:
        if d[k] == 1 then:
            R_k = (s_k * c) mod n
        else:
            R_k = s_k
        endif
        s_(k+1) = (R_k)^2 mod n
    end for
    return R_(w-1)
============================
```

a) Why does the above code cause a power side-channel leakage? (1 point)

b) How is the power trace correlated to the secret decryption key (`d`)? (1 point)

c) Write the pseudo-code for a leakage-free RSA decryption implementation. Your code must have only minor modifications to the above implementation. (3 points)

d) Explain how your code fixes the leakage of the naive implementation. (2 point)

Answer.

a) The power usage trace of the CPU differs based on each bit of the key. Both the multiply and exponentiation functions consume visible power traces, but the sequence of multiplication and exponentiation depends on the bits of the key (secret).

b) While the exponentiation function is called for each iteration, the multiplication function only runs when the key bit is 1.

c) Shown below

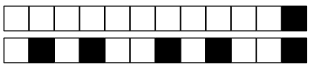
d) My code fixes the side-channel leakage since both multiplication and exponentiation are run irrespective of the key bit. Instead, the key bit determines the multiplicands to `t_0` and `t_1`, leading to the correct value in `R_k`. Control dependency has become a data dependency.

```
========== Algorithm ========
decrypt(c = ciphertext, d = decryption key, n = modulo):
    s_0 = 1
    for k = 0 to w-1:
        t_0 = s_k
        t_1 = (s_k . c) mod n
        R_k = ((t_0 * (1 - d[k])) + (t_1 * d[k])) mod n
or
        R_k = (s_k + (s_k * (c - 1) * d[k])) mod n

        s_(k+1) = (R_k)^2 mod n
    end for
    return R_(w-1)
============================
```

1 pt for working RSA (semantics still valid) 2 pts for leakage-free computation, possibly 1 point for each correct case

Grading. Code: 1 point for correct, 1 point for minimal changes.



Question 43: ML Security (6 points)

Describe the concept of "Adversarial Examples" in the context of machine learning along the following axes. For each axis, describe in at most three sentences.

a) Goal of adversarial examples. (1 point)

b) Method of creation. (1 point)

c) Challenge in defense. (2 points)

d) One example of how to defend adversarial examples. (2 points)

0 .5 1 .5 2 .5 3 .5 4 .5 5 .5 6

Answer.
(A) Goal of adversarial examples: Adversarial examples are inputs intentionally designed to cause a ML model to make mistakes. They differ from standard inputs in that they do not conform to the training distribution and are not drawn independently. The attacker often manipulates these inputs, or samples from a single input repeatedly.
(B) Method of creation: These examples are typically created using gradient descent or similar methods to find a perturbation that maximizes the model's loss. The process involves taking steps in the direction of maximum loss while adhering to a constraint like a perturbation norm. This process can include small affine transformations and is often repeated several times from a random starting point.
(C) Challenge in defense: Defending against adversarial examples is difficult because it usually involves preparing for specific threat models (e.g., perturbations up to a certain norm) without general guarantees. Or, we need to improve the worst-case prediction.
(D) Example: include adversarial training, where the model is trained on simulated adversarial examples, and certified defenses that ensure no example can exist within a certain radius. Detect suspicious queries. Note: "Address over-fitting by using regularization" could also be accepted as correct answer.

Grading.
(A) and (B) worth 1 points each and (C) and (D) worth 2 points each. 6 points in total.
(A) try to steal the model: not ok.
(B) only answering black-box or white box gives no points. Should mention using gradient descent (for a white box), or find to adversarial patch that is universal. there is too many possible adversarial examples, is ok.
(C) answering adversarial examples are transferable, is ok. answeing privacy is not ok. the noise added is small and hard to detect, is ok. ML models are not interpretable, is ok.
(D) answer detecting malicious input, is ok; keep the model secret is ok