RUTGERS UNIVERSITY

Department of Computer Science

Computer Security

Exam 2

November 3, 2025

Review – Answers Discussions

Prepared with tender loving care by Paul Krzyzanowski

100 POINTS – 25 QUESTIONS – 4 POINTS EACH – For each statement, select the *most* appropriate answer.

1. What happens if 51% of the computers (miners) in the Bitcoin network suddenly go offline?
   a. The remaining miners can immediately rewrite old blocks because the network has lost majority consensus.
   b. New blocks are created more slowly until the system readjusts to the lower mining power.
   c. The Bitcoin protocol pauses until at least an additional 2% of the original miners return.
   d. A double-spending attack becomes possible even without dishonest miners.

Correct answer: b – New blocks are created more slowly until the system readjusts to the lower mining power

Mining continues but at a slower pace until the network uses its Difficulty Adjustment Algorithm to restore its target block rate.

Incorrect answers:

a. Honest miners cannot alter past blocks without majority hash power.

c. The network doesn't stop; it just slows down.

d. Double spending requires control of most active miners, not fewer participants.

2. What is the main purpose of *Proof of Work* in Bitcoin?
   a. To prove that every transaction in a block is valid.
   b. To reward miners for solving math problems.
   c. To vote on which transactions are valid.
   d. To determine which miner earns the right to publish the next block.

Correct answer: d – To determine which miner earns the right to publish the next block

Proof of Work is a competition that makes miners expend computational effort; the first valid hash grants the right to append a block and claim the reward.

Incorrect answers:

a. Transaction validity is checked separately by all nodes, not proved by mining.

b. Miners are rewarded for securing the network, not for solving math problems; and finding a hash isn't really solving a math problem!

c. Miners don't vote on validity; consensus follows the chain with the most accumulated work.

3. Each *transaction* in Bitcoin is:
   a. Encrypted for the recipient and signed by the sender.
   b. Encrypted for the network and verified by miners.
   c. Unencrypted and unsigned but validated through Proof of Work.
   d. Unencrypted and signed by the sender.

Correct answer: d – Unencrypted and signed by the owner

Bitcoin transactions are public; ownership is proven through digital signatures using the sender's private key.

Incorrect answers:

a. Transactions are not encrypted — anyone can view them.

b. Proof of Work secures blocks, not individual transactions.

c. Every valid transaction must include a digital signature to prove ownership.

4. A major weakness of *Discretionary Access Control* (DAC) systems is that:
   a. They cannot enforce global security policies across all users.
   b. They prevent users from easily sharing files.
   c. They are too restrictive for multiuser systems.
   d. They rely on encryption to protect against unauthorized access to files.

Correct answer: a – They cannot enforce global security policies across all users

DAC allows owners to override or weaken security.

Incorrect answers:

b. DAC actually makes sharing easy.

c. DAC is flexible, not restrictive.

d. Encryption is unrelated to DAC; access control manages permissions, not cryptographic protection..

5. What does *multilateral security* add to the Bell-LaPadula (BLP) model?
   a. Encryption of all files.
   b. Availability capabilities via replication.
   c. Compartments that restrict access to the same classification level.
   d. Discretionary Access Control (DAC) support.

Correct answer: c

Multilateral security divides each classification level into compartments representing separate projects or domains. Users need both appropriate clearance and compartment authorization, enforcing need-to-know.

Incorrect answers:

a. Multilateral security is about access boundaries, not encryption.

b. It's about access control, not backups.

d. It's an access control model, not a network security mechanism.

6. In *Role-Based Access Control* (RBAC), access permissions are assigned to:
   a. Users directly based on their identity.
   b. Groups representing departments.
   c. Attributes describing user properties.
   d. Job functions.

Correct answer: d – Job functions.

In RBAC, permissions are grouped by job function (or "role"). Users are assigned to these functions and automatically inherit the associated permissions. This model simplifies administration and enforces least privilege through role design.

Incorrect answers:

a. This describes Discretionary Access Control (DAC), where permissions are granted directly to users.

b. Department-based grouping is a simple form of group access control, not the formal RBAC model.

c. This describes Attribute-Based Access Control (ABAC), which makes decisions based on user or environmental attributes rather than predefined job roles.

7. The *setuid* bit in Linux:
   a. Allows a program to run with the privileges of the file's owner.
   b. Allows a program to run only with the privileges of the user who launched it.
   c. Always runs a program with root privileges.
   d. Lets a user program run with normal privileges but temporarily access restricted system calls.

Correct answer: a – Allows a program to run with the privileges of the file's owner

When a file with the setuid bit is executed, the process's effective user ID becomes that of the file owner, not the user who ran it.

Incorrect answers:

b. That's the default behavior when setuid is *not* set.

c. Only files owned by root gain root privileges; most setuid programs are owned by non-root users.

d. Access to restricted system calls depends on privilege level, not setuid itself.

8. In an *Access Control List* (ACL)-based system, permissions are associated with:
   a. Each object (file, directory, or device).
   b. Each subject (user or process).
   c. The user's privilege level.
   d. Entries in the system access database.

Correct answer: a – Each object (file, directory, or device)

ACLs list which users or groups have specific permissions for a particular object.

Incorrect answers:

b. That describes a capability-based system.

c. Privilege levels are separate from per-object access rights.

d. Access databases store credentials, not object permissions.

9. Microsoft applies a simplified *Biba model* to Windows primarily to:
   a. Prevent browsers from accessing the user's personal files.
   b. Prevent users from accessing files belonging to other users on the system.
   c. Allow users to assign custom labels to files based on sensitivity.
   d. Prevent low-privilege programs, such as browsers, from modifying system files.

Correct answer: d – Prevent low-privilege programs like browsers from modifying system files.

Windows uses a simplified form of the Biba integrity model (Mandatory Integrity Control) to preserve system integrity. Low-integrity processes, such as browsers or downloaded code, cannot write to high-integrity system files or registry keys.

Incorrect answers:

a. This addresses confidentiality, not integrity.

b. Restricting users from each other's files is part of Discretionary Access Control, not the Biba model.

c. Assigning sensitivity labels to files is a Bell–LaPadula confidentiality mechanism, not an integrity control..

10. The *Chinese Wall* model enforces confidentiality by:
    a. Assigning users to specific roles, with per-role access permissions.
    b. Comparing classification labels between users and objects before granting access.
    c. Preventing users from accessing data for competing clients after prior access to other data.
    d. Requiring approval for every access.

Correct answer: c – Preventing users from accessing data for competing clients after prior access

It blocks cross-client access within a conflict class to avoid information leakage.

Incorrect answers:

a. That's RBAC.

b. That's Bell–LaPadula.

d. That describes workflow or separation-of-duty constraints, not Chines

11. What ended the effectiveness of traditional text-based CAPTCHAs?
    a. Users found them annoying.
    b. Machine learning and AI could solve them reliably.
    c. They became too computationally expensive on the server.
    d. Accessibility laws banned them.

Correct answer: b

By the mid-2010s, machine learning, OCR, and image recognition could solve traditional CAPTCHAs reliably, with AI eventually outperforming humans at reading distorted text.

Incorrect answers:

a. User behavior didn't end their effectiveness—AI capabilities did.

c. Cost wasn't the issue that ended their effectiveness.

d. No laws banned CAPTCHAs; they simply became technically obsolete.

12. What technical mechanism does a system use when it asks users to check a box labeled "*I'm not a robot*"?
    a. It verifies that automated bots cannot convincingly simulate natural human behavior.
    b. It encrypts the checkbox click using a private key unique to each browser.
    c. It analyzes subtle behavioral cues such as mouse movements and interaction timing.
    d. It relies on the user's IP address to determine if they are trustworthy.

Correct answer: c – It analyzes subtle behavioral cues such as mouse movements and interaction timing.

reCAPTCHA v2 uses behavioral and contextual data collected during the checkbox interaction—such as cursor trajectory, click timing, and focus changes—to estimate the likelihood that the user is human.

Incorrect answers:

a. Too vague; the system doesn't "verify" bots can't mimic behavior—it measures behavior to decide probabilistically.

b. No encryption-based verification is involved; the click event is not authenticated cryptographically.

d. IP reputation may influence the risk score but is not the technical mechanism for human detection..

13. What happens when the same memory is freed twice (a *double-free*)?
    a. It does no harm and helps prevent memory leaks.
    b. It ensures that the memory region is securely wiped.
    c. It can corrupt the memory allocator's internal data structures.
    d. It may slow performance by triggering garbage collection.

Correct answer: c – Freeing memory twice corrupts the allocator's internal state.

A double-free places the same memory block back into the allocator's free list more than once. This leads to heap corruption, crashes, or exploitable vulnerabilities since the allocator may later hand out overlapping memory regions.

Incorrect answers:

a. Freeing memory twice is not harmless and does not prevent leaks; it causes undefined behavior.

b. The free operation does not erase or clear memory contents.

d. Languages like C and C++ do not use garbage collection, so performance is unaffected in this way.

14. Which property of *integer overflows* makes them dangerous?
    a. They modify adjacent memory locations.
    b. They produce segmentation faults that stop execution.
    c. They make arithmetic operations unpredictable.
    d. They can cause incorrect allocation sizes that lead to buffer overflows.

Correct answer: d – They can cause incorrect allocation sizes that lead to buffer overflows.

When arithmetic wraps around, an expression used to compute a buffer size or index may produce a smaller-than-intended value. The program may then allocate too little memory and later write past the end of the buffer, causing corruption.

Incorrect answers:

a. Integer overflows corrupt computed values, not adjacent memory directly.

b. Most integer overflows do not trigger hardware faults or segmentation violations.

c. Overflowed arithmetic is deterministic within fixed-width integer limits; the danger is logic error, not unpredictability.

15. How does *Address Space Layout Randomization* (ASLR) strengthen system security?
    a. By making memory accesses atomic to prevent race conditions.
    b. By changing the base addresses of code and data regions across process instances.
    c. By randomly scrambling the order in which stack, heap, and text regions appear in a process's address space.
    d. By inserting random values at the ends of stack frames and heap allocations.

Correct answer: b – ASLR randomizes base addresses of memory regions to make addresses unpredictable.

ASLR varies where the stack, heap, executable, and libraries are mapped each run so an attacker cannot rely on fixed addresses for pointers, return-to-libc calls, or ROP gadgets without an information leak.

Incorrect answers:

a. Memory atomicity addresses concurrency and race conditions, not address predictability.

c. Randomizing the order of regions sounds similar but is misleading; ASLR changes base addresses and offsets, not necessarily the relative ordering of region types — the effective protection is unpredictability of addresses, not a simple permutation of region sequence.

d. Inserting random values at frame or allocation ends describes canaries or guard data; that defends against overflow detection, not address-guessing, so it is a different mitigation.

16. What protection mechanism was *Return-Oriented Programming* (ROP) specifically created to bypass?
    a. Address space layout randomization (ASLR)
    b. Non-executable memory (NX)
    c. Stack canaries
    d. Heap integrity checks

Correct answer: b – Non-executable memory (NX).

NX (or DEP) prevents injected code from executing on the stack or heap. ROP was created to bypass this by reusing existing executable code ("*gadgets*") already loaded in memory.

Incorrect answers:

a ASLR randomizes addresses and makes ROP harder, not easier.

c Stack canaries detect stack corruption but don't stop code execution.

d Heap integrity checks protect allocator metadata, not code execution.

17. What is the goal of *fuzzing* in memory vulnerability detection?
    a. To insert special test data that overwrites memory near buffers to check for overflow resilience.
    b. To verify the correctness of compiler optimizations.
    c. To prevent buffer overflows through runtime checks.
    d. To automatically generate inputs that may trigger crashes or unsafe behavior.

Correct answer: d – To automatically generate inputs that trigger crashes or unsafe behavior.

Fuzzing is a dynamic testing method that feeds random or mutated inputs to a program to uncover crashes or unsafe memory operations so developers can identify and fix underlying vulnerabilities.

Incorrect answers:

a This can describe some forms of instrumentation or stress testing, not fuzzing; fuzzers do not modify program memory directly but rely on input-driven execution to expose faults.

b Compiler optimization validation is unrelated to fuzzing.

c Fuzzing detects vulnerabilities; it does not prevent them or add runtime protections.

18. What is a *return-to-libc* attack?
    a. Injecting modified code into library functions.
    b. Forcing the program to load a different shared library.
    c. Overwriting a return address so execution jumps to an existing library function.
    d. Causing a program to exit prematurely.

Correct answer: c – Overwriting a return address so execution jumps to an existing library function.

A return-to-libc attack reuses existing code by redirecting control flow to a function already present in a shared library (commonly `system()` in libc). This bypasses non-executable memory protections because no new code is injected—only the return address and arguments are manipulated.

Incorrect answers:

a Injecting or modifying code is classic code injection, not return-to-libc, which uses existing instructions.

b Forcing a different library to load (e.g., via LD_PRELOAD) alters linking behavior, not control flow at runtime.

d Causing early program termination is a crash, not an exploit technique.

19. Which of the following operations is at risk of a TOCTTOU (*Time Of Check To Time Of Use*) attack?
    a. Test that a user-supplied filename does not have a "`..`" sequence and then create it.
    b. Set a *umask* to ensure only the owner gets read-write access and then create the file.
    c. Delete the file `myfile.txt` if it is smaller than 20 megabytes.
    d. Delete all files under `/var/log` that were created more than 365 days ago.

Correct answer: c – Delete the file myfile.txt if it is smaller than 20 megabytes.

This sequence checks a property (file size) and then acts on the same file. Between the check and the action, an attacker could replace the file with another one, exploiting a race condition.

Incorrect answers:

a. Checking the filename string before creation does not involve any subsequent operation on the same file object; the check and use apply to different entities.

b. Setting a umask affects future file permissions globally and does not introduce a race between checking and using a specific file.

d. Deleting files based on age uses metadata but not in that which could be invalidated between steps.

20. What makes *parameterized queries* effective against SQL injection?
    a. They separate user input from the query.
    b. They ensure that user input does not contain special characters that will change the query.
    c. They split query construction into multiple steps to make attacks harder.
    d. They automatically escape or sanitize user input.

Correct answer: a - Parameterized queries separate user input from the SQL statement.

Prepared statements send the SQL template and the parameter values separately to the database engine; the engine treats parameters strictly as data, never as executable SQL, so attackers cannot change the query structure by supplying malicious input.

Incorrect answers:

b. Ensuring inputs lack special characters is brittle and impractical; attackers can use encodings or alternate syntax to bypass such checks.

c. Splitting construction into steps is an imprecise description; the key is that parameter binding happens at execution time so inputs are not parsed as SQL, not merely that there are multiple steps.

d. Automatic escaping sounds plausible but is not what parameterized queries rely on; escaping can be error-prone and database-specific, whereas parameterization guarantees data is not interpreted as code.

21. What gap does *AppArmor* address that seccomp-BPF cannot?
    a. Digital signature verification for application integrity.
    b. Pathname-based access control to files and directories.
    c. Control over process memory usage and allocation.
    d. User identity verification and authentication.

Correct answer: b – Pathname-based access control to files and directories.

AppArmor enforces mandatory access control rules using file pathnames to restrict what files or directories a process can access. seccomp-BPF, in contrast, only filters which system calls a process may invoke.

Incorrect answers:

a. AppArmor does not use digital signatures or verify code integrity at runtime.

c. Memory allocation and management are handled by the kernel, not by AppArmor or seccomp-BPF.

d. Authentication is managed by other subsystems such as PAM, not by AppArmor..

22. How is *path equivalence* used in attacks?
    a. Replace a validated file via a race condition so that a different file is used.
    b. Supply a different-looking path that resolves to a protected file.
    c. Open many equivalent pathnames to exhaust file descriptors.
    d. Create visually similar filenames to trick users.

Correct answer: b - An attacker supplies a different-looking path that resolves to a protected file.

By providing a pathname that appears benign but, after canonicalization (resolving `..`, symlinks, or encodings), points to a protected resource, an attacker can bypass naive string-based checks and access files outside the intended directory.

Incorrect answers:

a. This describes a TOCTTOU (time-of-check to time-of-use) race, not path equivalence.

c. Resource exhaustion attacks are unrelated to the resolution of distinct path strings to the same file.

d. Typosquatting or visually similar filenames is a social-engineering technique, not path equivalence based on filesystem resolution.

23. How do Linux *capabilities* improve on traditional UNIX root privileges?
    a. They break root privileges into smaller, independently assignable pieces.
    b. They enforce strong authentication before administrative operations.
    c. They allow an administrator to manage multiple machines with one login.
    d. They add access control lists (ACLs) to all system files.

Correct answer: a - They break root privileges into smaller, independently assignable pieces.

Capabilities divide the all-powerful root privilege into specific, limited privileges (for example, `CAP_NET_ADMIN`, `CAP_SYS_TIME`), allowing processes or users to perform certain administrative actions without full root access.

Incorrect answers:

b. Capabilities manage privileges after authentication, not user login or authentication strength.

c. Managing multiple machines with one login involves centralized authentication systems (like LDAP or Kerberos), not Linux capabilities.

d. ACLs govern file access permissions, whereas capabilities control what privileged operations a process can perform.

24. Which Linux mechanism provides configurable per-process *system call filtering*?
    a. chroot.
    b. Control groups (cgroups).
    c. seccomp-BPF.
    d. Namespaces.

Correct answer: c - seccomp-BPF.

Seccomp with BPF (Secure Computing with Berkeley Packet Filter) allows a process to define which system calls it can make. The kernel enforces these filters on a per-process basis, providing fine-grained control over what untrusted code can do.

Incorrect answers:

a. chroot changes a process's root directory to restrict filesystem access but does not control which system calls can be invoked.

b. Cgroups limit resources such as CPU, memory, and I/O bandwidth but do not restrict system calls.

d. Namespaces isolate global resources (like process IDs, mounts, or networking) between processes, but they do not filter or block system calls.

25. What is the key architectural difference between *containers* and *virtual machines*?
    a. Containers share the host operating system kernel, while each virtual machine includes its own kernel.
    b. Containers rely on hardware virtualization support, while VMs can operate without it.
    c. Containers provide stronger isolation because they emulate hardware.
    d. Containers provide hardware-level isolation, while VMs provide only process-level isolation.

Correct answer: a – Containers share the host operating system kernel, while each virtual machine includes its own kernel.

Containers virtualize at the operating system level, isolating processes within the same kernel. Virtual machines emulate hardware and run full guest operating systems with separate kernels.

Incorrect answers:

b. Containers run directly on the host kernel and do not require hardware virtualization; hypervisors may use hardware support for VMs.

c. Virtual machines, not containers, emulate hardware and thus provide stronger isolation.

d. Virtual machines provide hardware-level isolation, while containers provide process-level isolation within the same OS.

The end.