Comprehensive Cybersecurity Midterm 1 Study Guide (Open Notes) — Expanded

Everything included + deeper explanations + extra likely exam questions • Generated 2026-02-09

How to use this document (open-notes strategy)

- This document is intentionally long and detailed. Use headings to jump to what the question is asking (CIA, crypto primitives, ECB vs CBC, hashes/SHA-1, OS hardening, setuid, sandboxing, etc.).
- For a "define" question: copy the Definition line and then include 1–2 supporting sentences from the "Why it matters" section.
- For a "compare" question: use the side-by-side comparison bullets and include a short scenario/example to show you understand the difference.
- For scenario questions: (1) identify the asset, (2) identify the vulnerability/threat/attack, (3) map to CIA impact, (4) propose prevention/detection/recovery controls.
- For True/False: write "True/False" plus one short justification sentence using a key phrase from the relevant section.

1) Cybersecurity fundamentals

1.1 What is cybersecurity?

Cybersecurity is the practice of protecting systems, networks, programs, and data from digital attacks, unauthorized access, misuse, disruption, or damage. In practical terms, cybersecurity is about reducing risk: preventing successful attacks when possible, detecting attacks that occur, and recovering quickly to limit harm. Cybersecurity includes technical controls (like encryption and access control), operational processes (like patch management and incident response), and people-focused measures (like training and policies).

1.2 The CIA triad (security goals)

Most security questions can be answered by mapping the situation to one or more CIA goals (Confidentiality, Integrity, Availability). Then you state what is being violated and which controls address it.

Confidentiality, Integrity, and Availability are not mutually exclusive—real incidents often affect multiple goals. For example, ransomware primarily affects availability, but it can also affect integrity (data altered/encrypted) and confidentiality (data exfiltration).

Confidentiality (secrecy/privacy)

Definition: Confidentiality is the avoidance of unauthorized disclosure of information. It ensures that information cannot be accessed or viewed by unauthorized users.

Why it matters: Confidentiality protects sensitive data such as personal identifiers, financial records, medical data, and proprietary business information. Loss of confidentiality can cause identity theft, financial loss, legal violations (e.g., HIPAA), and reputational damage.

Typical controls: Encryption (data at rest and in transit), access control policies, least privilege, authentication (including MFA), secure key management, and auditing/monitoring of access.

Integrity (authenticity / trustworthiness)

Definition: Integrity ensures information can be modified only by authorized users and is not altered unexpectedly. In communication, integrity is connected to authenticity: the receiver can trust that the message has not been tampered with.

Why it matters: Integrity is essential when decisions depend on accurate data (banking transactions, grades, medical orders, system configurations). Integrity loss can lead to incorrect decisions, fraud, safety risks, or system compromise.

Typical controls: Hashes/checksums for tamper detection; MAC/HMAC for keyed integrity; digital signatures for public verifiability; access controls; auditing; backups/version control for recovery.

Availability

Definition: Availability ensures that information and services are accessible to authorized users in an acceptable time frame when requested.

Why it matters: Even if confidentiality and integrity are perfect, a system that is frequently down is not useful. Availability matters for critical services (healthcare systems, banking, emergency services, enterprise operations).

Typical threats: DoS/DDoS attacks, ransomware, hardware failures, misconfiguration, power/network outages.

Typical controls: Redundancy and failover, load balancing, DDoS mitigation, monitoring and alerting, backups and disaster recovery planning, capacity planning.

1.3 CIA examples (attack + control) — ready for short answers

- Confidentiality example: Attacker steals a customer database (data breach) → Control: encrypt database + strict access control + MFA for admins + audit logs.
- Integrity example: Attacker modifies a software update or payment record → Control: digital signatures on updates + integrity verification (hashes) + secure change control + backups for rollback.
- Availability example: DDoS overwhelms web service → Control: rate limiting + DDoS scrubbing/CDN + redundant infrastructure + monitoring/incident response.

Exam tip: If you're asked "Which CIA property is affected?", name the property and then explicitly connect it to the scenario: "This is a confidentiality issue because unauthorized parties gained access to data," etc.

1.4 Core security vocabulary (must memorize)

These terms are foundational. Many exam questions implicitly test whether you can identify each one in a scenario.

Asset

Definition: An asset is anything valuable that must be protected. Assets can include data, systems, services, hardware, personnel, intellectual property, and organizational reputation.

How to identify in scenarios: Ask “What are we trying to protect?” In a bank scenario, the asset might be account balances and customer identities; in OS security, the asset might be system integrity and sensitive files.

Vulnerability

Definition: A vulnerability is a weakness in design, implementation, configuration, or operation that could be exploited. Vulnerabilities can be technical (buffer overflow), configuration-related (open admin ports), or human/process-related (poor patching practices).

How to identify: Ask “What weakness makes the attack possible?”

Threat

Definition: A threat is a circumstance, event, or actor with the potential to cause harm by exploiting vulnerabilities. A threat can be a person (attacker), a group (criminal org), or an event (power outage).

How to identify: Ask “Who/what can cause harm here?”

Attack

Definition: An attack is an attempt to collect, disrupt, deny, degrade, or destroy system resources or information. Attacks can be active (modifying, disrupting) or passive (eavesdropping). Attacks can also be insider or outsider.

How to identify: Ask “What action is being taken to exploit the vulnerability?”

Countermeasure (control)

Definition: A countermeasure is a technique/device/process used to prevent, detect, or recover from attacks. Good answers often include all three: prevention, detection, recovery.

Note: Controls can introduce complexity and new vulnerabilities. Even with controls, some residual risk remains.

Risk

Definition: Risk is the measure of how threatened an asset is, often modeled as likelihood × impact. Risk management is about prioritizing what to fix first based on the most dangerous combinations.

Exam phrasing: “Risk increases when vulnerabilities are severe, threats are likely, and impacts are high.”

1.5 Authentication vs Authorization (common exam trap)

Authentication

Definition: Authentication determines identity or role (who you are). Authentication answers the question: “Are you really the user you claim to be?”

Factors: Something you know (password/PIN), something you have (phone/token/smart card), something you are (biometrics). Multi-factor authentication (MFA) combines two or more categories to reduce credential theft risk.

Authorization

Definition: Authorization determines what an authenticated identity is allowed to do (permissions), based on an access control policy. Authorization answers: “Now that we know who you are, what are you allowed to access/do?”

Example: Logging in is authentication. Being allowed to view an admin dashboard is authorization.

Q: One-sentence distinction (copy/paste).

A: Authentication verifies identity/role. Authorization determines allowed access/actions under a policy.

1.6 Security trade-offs (essay-ready, with detail)

Confidentiality vs Usability

Security measures that improve confidentiality can reduce usability. For example, MFA, complex password requirements, timed logouts, and strict access controls reduce unauthorized access but can slow down legitimate work and increase frustration.

In healthcare, staff may need fast access to patient records; too much friction can delay care. In banking, stronger login security reduces fraud but can drive users away if the process is too hard. The security goal is to balance friction with risk, and to design controls that are secure and usable (e.g., MFA that is quick but strong).

Integrity vs Performance

Integrity measures like hashing and digital signatures add computation and sometimes extra steps. For example, verifying a digital signature requires cryptographic operations and may add latency.

However, integrity controls prevent severe outcomes like malicious updates, tampered transactions, and corrupted files. In many contexts, the performance cost is worth it because the cost of tampering is much larger.

Availability vs Cost

High availability often requires redundancy (extra servers, multiple data centers), monitoring, incident response capability, backups, and disaster recovery planning.

These measures cost money (hardware, cloud spend, staffing) and may add complexity. The organization chooses a target availability level (SLA/SLO) and funds the infrastructure to meet it.

2) Cryptography concepts (Crypto + Authentication)

Cryptography is used to protect data and communications. Exam questions often test whether you can identify which cryptographic tool is appropriate for a goal (confidentiality vs integrity vs authentication vs non-repudiation).

2.1 Map primitive → security goal (memorize)

- Encryption → confidentiality (hide content).
- Hash function → integrity fingerprint / tamper evidence (no secret key).
- MAC/HMAC → integrity + origin authentication with a shared secret key.
- Digital signature → integrity + public-key authentication + non-repudiation (publicly verifiable).
- Certificate/CA → bind identity to a public key using a trusted signature.

2.2 Symmetric cryptography (secret-key)

Symmetric cryptography uses a single secret key shared between parties for both encryption and decryption. It is typically very fast and is the standard approach for encrypting large amounts of data (bulk encryption).

- Pros: fast, efficient for large data.
- Cons: key distribution problem (both sides must securely share the secret key).
- Example: AES.

2.3 Asymmetric cryptography (public-key)

Asymmetric cryptography uses a public key and a private key. The public key can be shared widely, while the private key is kept secret. Asymmetric crypto supports use-cases like secure key exchange and digital signatures.

- Pros: solves key distribution for many use-cases; supports digital signatures.
- Cons: slower than symmetric crypto; typically not used for bulk data encryption.

Common pattern: use public-key crypto to establish a shared secret (or exchange a symmetric key), then use symmetric crypto (AES) for the actual data.

2.4 Brute-force vs cryptanalytic attacks (detailed)

A brute-force attack is an exhaustive search: the attacker tries all possible keys until the correct one is found. Brute-force success depends heavily on key length and computational resources.

A cryptanalytic attack exploits weaknesses in the algorithm, implementation, protocol, or plaintext structure to reduce the work below brute force. For example, mode-of-operation mistakes (ECB patterns) or weak randomness can break confidentiality even with long keys.

- Exam phrasing: “Trying all possible private keys” = brute-force attack.
- Key length helps against brute force, but does not guarantee security if the system design is flawed.

2.5 AES essentials (expanded)

AES (Advanced Encryption Standard) is a symmetric block cipher standardized by NIST. It operates on 128-bit blocks and supports key sizes of 128, 192, or 256 bits. AES is widely used because it is secure and efficient (often hardware-accelerated).

AES can be used in different modes of operation to encrypt data longer than one block; the mode matters for security properties (pattern leakage, parallelism, error propagation).

2.6 Modes of operation: ECB vs CBC (expanded)

ECB (Electronic Codebook)

ECB encrypts each plaintext block independently with the same key. If plaintext blocks repeat, ciphertext blocks repeat. This leaks patterns and structure, especially in structured data (images, formatted documents, repeated headers).

- Main weakness: pattern leakage (identical plaintext blocks → identical ciphertext blocks).
- When you might see it: legacy systems, incorrect implementations, toy examples.
- Why it fails conceptually: there is no randomness or chaining between blocks.

Exam-ready line: “ECB is not semantically secure for structured/repeating plaintext because it leaks patterns.”

CBC (Cipher Block Chaining)

CBC uses an initialization vector (IV) and chaining to hide patterns. Each plaintext block is XORed with the previous ciphertext block before encryption. This ensures that even identical plaintext blocks can encrypt to different ciphertext blocks (as long as chaining differs).

- Strength: hides repeating patterns better than ECB.
- Requirement: must use a fresh/unpredictable IV for each encryption to avoid revealing relationships.
- Tradeoff: encryption is sequential (each block depends on previous ciphertext), which impacts parallelization and robustness in lossy channels.

Exam-ready line: “CBC prevents ECB-style pattern leakage by chaining blocks with XOR and an IV, but it requires sequential processing.”

2.7 Hash functions (expanded, exam-ready)

A hash function maps arbitrary-length input to a fixed-length output (digest). Cryptographic hashes aim to behave like random functions with specific hardness properties. They are used widely for integrity and as building blocks in authentication and signatures.

Key security properties

- Preimage resistance: given a hash output h, it should be infeasible to find any message M such that H(M)=h.
- Second-preimage resistance: given an input M1, it should be infeasible to find a different M2 such that H(M2)=H(M1).
- Collision resistance: it should be infeasible to find any two distinct inputs M1≠M2 with the same hash output.

Important intuition: collision resistance is usually the hardest/most important property for many applications, and collisions become feasible sooner than people expect because of the birthday paradox.

Birthday attack intuition (expanded)

The birthday paradox shows that collisions appear after roughly √N samples where N is the number of possible hash outputs. For an n-bit hash, N=2^n, so collisions appear after about 2^(n/2) attempts. This is why collision security is about half the digest size.

Practical implication: 160-bit hashes (like SHA-1) provide about 80-bit collision security in the ideal case, which is below modern recommendations for collision resistance in many uses.

2.8 Why SHA-1 is not secure (expanded answer)

SHA-1 is deprecated primarily because it does not provide acceptable collision resistance. When collisions can be found, hashes are no longer reliable unique fingerprints.

This matters because many systems rely on hashing as part of signature and certificate workflows: if an attacker can create two different messages with the same hash, they can potentially get a signature on one message and transfer it to the other message (conceptually undermining integrity/authentication).

- Core exam phrase: “SHA-1 is not considered secure due to inadequate collision resistance.”
- Recommended alternative: SHA-256 (SHA-2) or SHA-3.

Q: Is SHA-1 secure?

A: No. SHA-1 is deprecated because its collision resistance is not acceptable for modern security needs; use SHA-256/SHA-3 instead.

2.9 MAC and HMAC (expanded)

A Message Authentication Code (MAC) provides integrity and origin authentication when the sender and receiver share a secret key. Unlike a plain hash, a MAC uses a secret key, so an attacker who does not know the key cannot forge a valid tag.

HMAC is a standardized MAC construction based on hash functions. It is designed to be secure even when the underlying hash is iterative, and it avoids pitfalls in naive constructions.

- Hash alone: integrity fingerprint but anyone can compute it (no authentication).
- HMAC: integrity + authentication because only key holders can compute valid tags.

2.10 Digital signatures (expanded)

Digital signatures provide integrity, authentication, and non-repudiation. The signer uses a private key to sign; anyone with the public key can verify the signature.

Non-repudiation means the signer cannot plausibly deny having signed the message (assuming the private key was protected and signature scheme is secure). This property is important in legal and audit contexts.

- Use-case: signed software updates; signed documents; certificate issuance.
- Contrast with MAC: MAC does not provide non-repudiation because any shared-key holder could have created the tag.

2.11 Certificates and CA (expanded)

Certificates solve the ‘who owns this public key?’ problem. A certificate authority (CA) signs a statement binding an identity (domain/person/org) to a public key. Verifiers trust the CA (and sometimes a chain of intermediate CAs) to validate identities properly.

In web security, this underpins TLS/HTTPS: your browser trusts a set of root CAs and uses certificates to authenticate servers.

2.12 RSA (expanded, exam-ready)

RSA is an asymmetric cryptosystem used for encryption and signatures. RSA is not a symmetric block cipher like AES.

RSA security intuition: multiplying large primes is easy; factoring their product $N=pq$ is hard. Many RSA operations occur modulo N.

- Key generation outline: choose large primes p,q; compute $N=pq$; compute $\phi(N)=(p-1)(q-1)$; choose e with $\gcd(e,\phi(N))=1$; compute d such that $e \cdot d \equiv 1 \pmod{\phi(N)}$.
- Public key: (e,N). Private key: d (and often p,q).

Q: Is RSA a block cipher?

A: No. RSA is an asymmetric/public-key scheme; AES is a symmetric block cipher.

2.13 Passwords and password storage (expanded)

Passwords are a common authentication factor (“something you know”). Security problems occur when users choose weak passwords, reuse passwords, or when systems store passwords insecurely.

Passwords should not be stored in plaintext. Systems store a password verifier, typically by hashing the password with a salt and using a slow password hashing function (KDF) to make offline cracking expensive.

- Salt: random value stored with the hash to prevent precomputed attacks and ensure identical passwords do not have identical hashes.
- Slow hashing/KDF: makes brute-force guessing slower (bcrypt, scrypt, Argon2).

2.14 OpenSSL commands (expanded + correct)

These are correct commands you can paste into answers when asked about generating keys. Common mistake: writing 'genpkay' instead of 'genpkey'.

Generate a 2048-bit RSA private key:

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -out private.pem
```

Extract the public key from the private key:

```
openssl pkey -in private.pem -pubout -out public.pem
```

3) Operating system security (expanded)

OS security assumes that bugs and misconfigurations are inevitable. The OS provides isolation, access control, and security mechanisms to limit damage when something goes wrong.

3.1 Why OS security matters (expanded)

Software is complex and constantly changing. Even with careful development, vulnerabilities occur (buffer overflows, use-after-free, logic bugs, misconfigurations). Attackers exploit these weaknesses to gain unauthorized access, run code, or steal data.

Modern systems therefore rely on layered defenses: secure coding, OS-level isolation, least privilege, patching, and monitoring.

3.2 Defense in depth (expanded)

Defense in depth means using multiple independent controls at different layers. If one layer fails, other layers can still prevent compromise or reduce impact.

- Application layer: sandboxing, input validation, safe libraries.
- OS layer: process isolation, access control, permissions, kernel protections.
- Hardware/firmware layer: secure boot, memory protections.
- Network layer: firewalls, segmentation, monitoring.

Exam example: If a browser tab renderer is compromised, sandboxing limits file access; OS permissions limit access to other processes; network monitoring may detect suspicious traffic.

3.3 Principle of least privilege (expanded)

Least privilege reduces the blast radius of compromise. If an attacker takes over a low-privilege account, they should not automatically gain access to sensitive admin functions or system-wide resources.

Least privilege applies to both users and processes: services should run with only the permissions they need. For example, a web server process should not run as root unless absolutely necessary.

3.4 UNIX/Linux security model (expanded)

UNIX systems use users, groups, and permissions to control access to objects (files and many resources treated as files).

- Users have UIDs; UID 0 is root.
- Groups have GIDs; users can belong to groups to share resource access.
- Each file has an owner (user), a group, and permission bits for user/group/other: read (r), write (w), execute (x).
- Processes run with a user identity and inherit permissions based on effective IDs.

Q: Fill-in: most powerful Linux user?

A: root.

3.5 setuid (purpose + risk) — expanded

The setuid bit is a special permission on executables. When set, running the program sets the process's effective user ID (EUID) to the executable's owner rather than the user who launched it.

Purpose: allow ordinary users to perform a specific privileged task through a controlled program. Classic example: changing a password requires writing to protected files; a carefully written setuid program can do that safely.

Risk: if a setuid-root program has a bug, an attacker can exploit it to gain root privileges. Because of this, setuid programs are security-critical and must be minimal, carefully audited, and hardened.

Q: Short answer: purpose of setuid?

A: The setuid bit allows a program to run with the effective UID of the file owner (often root) rather than the calling user, enabling controlled privilege escalation for tasks requiring elevated permissions.

3.6 Linux capabilities (expanded)

Linux capabilities split root's all-powerful privileges into distinct capabilities. A process can be granted only the capability it needs (e.g., binding to low ports) instead of full root power.

This reduces risk: if the process is compromised, the attacker gains only limited privileges rather than complete system control.

3.7 Windows security context + UAC (expanded)

Windows uses security tokens for processes and security descriptors for objects. Tokens represent the security context (user, groups, privileges). Access checks compare the token to object permissions.

User Account Control (UAC) prompts appear when an action requires administrator privileges. UAC helps prevent silent privilege escalation by requiring user approval for admin-level operations.

Q: Fill-in: UAC prompt appears because action requires _____ privileges.

A: administrator

3.8 Sandboxing and isolation examples (expanded)

Sandboxing restricts what code can access, even if that code is running under a user account. The goal is to limit damage if code is malicious or exploited.

- Mac App Sandbox: mediates access to hardware, network, app data, and user files; if access is not requested/allowed, it is rejected at runtime.
- Android: apps run with separate UIDs and permissions; a reference monitor enforces permissions on IPC and resources.
- Chrome: separates browser and renderer processes; renderer is sandboxed; broker mediates privileged actions and reduces the renderer's privileges.

4) OS hardening, operations, and continuous security

4.1 What hardening means (expanded)

Hardening means reducing the attack surface and configuring a system securely. Default configurations prioritize usability and broad compatibility; hardening intentionally disables unnecessary components and tightens privileges to reduce vulnerabilities.

4.2 Hardening checklist (expanded)

- Patch management: keep OS and critical software updated; prioritize security patches; test patches before wide deployment; maintain an inventory of systems.
- Remove unnecessary services/apps/protocols: if you don’t need it, disable it; fewer running services means fewer exposed vulnerabilities.
- Account hygiene: disable default accounts, remove unused accounts, enforce strong authentication, use MFA for admins, restrict privilege escalation.
- Permissions and least privilege: ensure services run with minimal permissions; restrict file and network access.
- Host security controls: firewall rules, anti-malware/EDR, IDS/IPS where appropriate, application allowlisting.
- Configuration baselines and scanning: use checklists/benchmarks; scan for misconfigurations and vulnerabilities regularly.

4.3 Security maintenance is continuous (expanded)

Security is continuous because new vulnerabilities are discovered, attackers change tactics, systems are updated, and configurations drift over time. Even a secure system today can become insecure tomorrow if it is not maintained.

- Continuous tasks: monitor logs, patch/update, scan for vulnerabilities, review access, back up data, test recovery, run security assessments, update policies and configurations.

Q: Is security maintenance one-time?

A: No. Maintaining security after deployment is continuous.

4.4 Logging and monitoring (expanded)

Logs are critical for detecting suspicious activity and reconstructing incidents after the fact. Without logs, it is difficult to know what happened, what data was accessed, or what changes were made.

Effective logging includes: choosing relevant events to log, protecting logs from tampering, centralized log collection, alerting on suspicious patterns, and regular review (often automated).

4.5 Backups vs archives (expanded)

Backups and archives support integrity and availability. Backups enable recovery after accidental deletion, corruption, or ransomware. Archives retain data over long periods for legal and operational requirements.

- Backup: focused on restoration and recovery; typically rolling windows (daily/weekly).

- Archive: long-term retention; often for compliance and historical record.

5) Ethical hacking & incidents (expanded)

5.1 Ethical hacking definition (expanded)

Ethical hacking (penetration testing) is authorized security testing that uses attacker techniques to identify vulnerabilities. Ethical hacking must be legal and approved by the system owner, and results should be responsibly reported.

Ethical hackers do not steal information or damage systems; the goal is to improve security, not to cause harm.

5.2 Hacker types (expanded)

- White hat: authorized defensive testing and security improvement.
- Black hat: malicious activity for gain or disruption.
- Grey hat: mixed behavior; may violate rules/laws even if motivation is not purely malicious.

5.3 Hacking process (expanded)

A common high-level hacking workflow includes:

1. Footprinting (reconnaissance): gather information about targets (domains, IPs, employees, tech stack).
2. Scanning: probe systems for open ports, services, versions, and vulnerabilities.
3. Gaining access: exploit vulnerabilities or credentials to obtain access.
4. Maintaining access: persist, escalate privileges, and continue access (in real attacks; in ethical hacking, persistence may be simulated and carefully controlled).

5.4 Tools (expanded)

- Kali Linux: security-focused Linux distribution with many tools.
- Nmap: port scanning and service enumeration (identify attack surface).
- Metasploit Framework: exploitation framework for testing known vulnerabilities.

5.5 Incident case study: Equifax (expanded framing)

Equifax (2017) is a commonly referenced breach. A useful exam framing is to identify CIA impact, likely root cause themes, and realistic mitigations.

- CIA impact: primarily confidentiality (exposure of sensitive personal data).
- Root cause themes: delayed patching of known vulnerabilities, inadequate vulnerability management, insufficient monitoring and asset visibility.
- Mitigations: faster patching, strong asset inventory, continuous vulnerability scanning, monitoring/log analysis, segmentation, and governance.

6) Your provided sample questions — answers + explanations

Q: T/F: “SHA-1 is considered to be very secure.”

A: False. SHA-1 is deprecated because its collision resistance is not acceptable for modern security needs.

Explanation: Collision resistance is essential when hashes serve as unique fingerprints. If collisions are feasible, an attacker can potentially undermine integrity checks and signature workflows by producing different messages with the same hash.

Q: T/F: "RSA is a block cipher in which plaintext/ciphertext are integers between 0 and n–1."

A: False as stated. RSA is an asymmetric/public-key cryptosystem, not a symmetric block cipher like AES.

Explanation: RSA does operate over integers modulo n, but classification matters: AES is a block cipher (fixed-size blocks), while RSA is public-key crypto used for key exchange, small-message encryption with padding, and signatures.

Q: T/F: "Hash functions generally execute slower than AES."

A: Often true in practice because AES is highly optimized and frequently hardware-accelerated; exact performance depends on implementation and platform.

Explanation: This question is usually testing that symmetric encryption like AES is designed to be efficient for bulk data and often has hardware support.

Q: T/F: "Authentication determines who is trusted for a given purpose."

A: False. Authentication verifies identity/role; authorization determines what actions/access are permitted (closer to "trusted for a purpose").

Q: T/F: "Modern OS are immune to buffer overflows due to built-in features."

A: False. Mitigations reduce risk, but vulnerabilities still exist and can be exploited; software bugs are inevitable.

Q: T/F: "A malicious driver can bypass many controls to install malware."

A: True. Drivers often run with high privilege (kernel-level), so a malicious driver can subvert many user-space controls.

Q: MCQ: After deployment, maintaining security is ____.

A: Continuous. Security requires ongoing patching, monitoring, scanning, backups, and configuration review.

Q: MCQ: Resources for security planning include ____.

A: All of the above (hardening guides, documentation, online references, checklists, tools).

Q: MCQ: Trying all possible private keys is a ____ attack.

A: Brute-force attack.

Q: Fill blank: Most powerful Linux user is ____.

A: root.

Q: Fill blank: UAC prompt appears because action requires ____ privileges.

A: administrator.

Q: Short answer: What is a hash function and its role in cryptography?

A: A hash function deterministically maps arbitrary-sized input to a fixed-size digest. Hashes support integrity checks, message authentication (via MAC/HMAC), digital signatures, and password storage (as hashed verifiers).

Extra: Hash alone does not provide confidentiality. For authentication, you need a key (HMAC) or a private key (signature).

Q: Short answer: Purpose of setuid bit?

A: setuid allows an executable to run with the effective UID of the file owner (often root) rather than the calling user, enabling controlled privilege escalation for specific tasks.

Risk: vulnerabilities in setuid-root programs can lead to full system compromise, so setuid programs must be minimal and carefully audited.

7) Last-minute checklist — fully answered with examples

7.1 Define CIA + give one attack/control for each (expanded)

Confidentiality: prevent unauthorized disclosure.

- Attack example: attacker steals database records (breach).
- Control example: encrypt data at rest/in transit; restrict access with least privilege and MFA.

Integrity: prevent unauthorized modification.

- Attack example: attacker alters a transaction record or software update.
- Control example: digital signatures or HMAC; integrity checks with hashes; backups/versioning for recovery.

Availability: ensure timely access to services/data.

- Attack example: DDoS overwhelms service or ransomware encrypts critical files.
- Control example: redundancy/failover, rate limiting/DDoS mitigation, backups and disaster recovery.

7.2 Authentication vs authorization (expanded)

- Authentication = identity verification (login proves who you are).
- Authorization = permission checking (what you can access after login).

7.3 encryption/hash/MAC/signature mapping (expanded)

- Encryption: confidentiality; reversible with a key.
- Hash: integrity fingerprint; one-way; no authentication by itself.
- HMAC/MAC: integrity + shared-secret authentication.
- Digital signature: integrity + public-key authentication + non-repudiation.

7.4 ECB vs CBC (expanded)

ECB leaks patterns because repeated plaintext blocks produce repeated ciphertext blocks. CBC hides patterns by chaining: each plaintext block is combined with the previous ciphertext and an IV before encryption.

7.5 SHA-1 insecurity (expanded)

SHA-1 is insecure because collision resistance is not sufficient. If collisions can be found, hashes can no longer serve as unique fingerprints for integrity/signature uses. Modern systems use SHA-256/SHA-3.

7.6 setuid purpose + risk (expanded)

setuid enables controlled privilege escalation by running a program with the file owner's effective privileges. It is risky because a vulnerability in a setuid-root program can be exploited to gain root access.

7.7 OS hardening + continuous maintenance (expanded)

Hardening reduces attack surface and tightens configuration. Maintenance is continuous because new vulnerabilities appear and systems change over time.

- Hardening steps: patch, disable unnecessary services, remove default accounts, enforce least privilege, firewall/EDR, scanning, logging/monitoring.
- Continuous tasks: patching, scanning, monitoring logs, backups and recovery testing.

7.8 Defense in depth (expanded)

Defense in depth layers controls. Example: sandboxed browser renderer limits file access; OS isolation limits process-to-process access; network monitoring detects suspicious exfiltration; backups enable recovery.

8) Additional likely exam questions (large bank with detailed answers)

These are common question styles for Intro/Crypto/OS security midterms. Use them for practice or as an answer bank during the open-notes exam.

Q: Define and contrast: confidentiality, integrity, availability.

A: Confidentiality prevents unauthorized disclosure (privacy). Integrity prevents unauthorized modification (trustworthiness). Availability ensures authorized users can access resources when needed in acceptable time. Real incidents often affect multiple properties; for example, ransomware primarily harms availability (data inaccessible) but can also harm integrity (data altered) and confidentiality (data exfiltration).

Q: In a scenario, how do you identify asset vs threat vs vulnerability vs attack?

A: Asset is what you value and protect (data/service). Vulnerability is the weakness that makes harm possible. Threat is the actor/event with potential to exploit the weakness. Attack is the specific action taken (exploitation, stealing, DoS).

Q: What does a hash guarantee and what does it not guarantee?

A: A cryptographic hash provides a fixed-length digest useful for tamper detection and integrity checking. It does not provide confidentiality (anyone can compute hashes) and does not authenticate a sender unless combined with a secret key (HMAC) or signatures.

Q: Why are collisions important and what is the birthday bound?

A: Collisions matter because many systems treat hash outputs as unique fingerprints. The birthday bound shows collisions appear around 2^(n/2) operations for an n-bit hash, meaning collision security is about half the digest size.

Q: Why is SHA-1 deprecated and what should replace it?

A: SHA-1 is deprecated due to inadequate collision resistance. Replace with SHA-256 (SHA-2) or SHA-3.

Q: Compare MAC vs digital signature in terms of who can verify and non-repudiation.

A: MAC uses a shared secret; only parties with the secret can create/verify, and it does not provide non-repudiation because either party could have produced the tag. Digital signatures are publicly verifiable with a public key and support non-repudiation (assuming key control).

Q: Why do systems use public-key crypto + symmetric crypto together?

A: Public-key crypto is slower but solves key distribution and supports authentication. Symmetric crypto is fast for bulk data. Common pattern: use public-key methods to establish a shared key, then use AES for data.

Q: Explain why ECB is insecure for images/documents.

A: ECB encrypts blocks independently; repeated plaintext blocks yield repeated ciphertext blocks, revealing patterns and structure, which leaks information even though the data is encrypted.

Q: Explain what an IV is and why it matters in CBC.

A: An IV (initialization vector) is an initial value used in CBC chaining so that the first block encryption incorporates randomness/uniqueness. A fresh/unpredictable IV prevents revealing relationships across encryptions of similar messages.

Q: What is least privilege and give an OS example.

A: Least privilege grants only needed permissions. Example: run a web server as a dedicated non-root user that can read web content but cannot access system files.

Q: What is attack surface and how does hardening reduce it?

A: Attack surface is the set of reachable entry points and vulnerable components. Hardening reduces it by disabling unnecessary services, closing ports, removing unused software, and tightening permissions.

Q: Why are drivers a big security risk?

A: Drivers run with high privilege (kernel-level). A malicious or compromised driver can bypass many controls, hide malware, and fully compromise the system.

Q: Explain setuid and why it exists; then explain why it is risky.

A: setuid exists to allow controlled privilege escalation for specific tasks. It is risky because vulnerabilities in setuid-root programs can be exploited for full privilege escalation.

Q: List and explain prevention vs detection vs recovery controls (with examples).

A: Prevention stops attacks (MFA, patching, firewall). Detection identifies attacks in progress or after occurrence (logging, IDS, monitoring). Recovery restores service and limits harm (backups, incident response, failover).

Q: Explain why security maintenance is continuous.

A: New vulnerabilities and threats emerge, systems change, and configurations drift. Continuous patching, scanning, monitoring, and testing are required to remain secure.

9) Quick lookup (one-liners)

- SHA-1 secure? No—deprecated (collision resistance).
- RSA block cipher? No—public-key; AES is symmetric block cipher.
- Brute force? Try all keys.
- Root? UID 0; most powerful user.
- UAC? Prompts for admin privileges.
- setuid? Runs with file-owner EUID; risky if vulnerable.
- ECB vs CBC? ECB leaks patterns; CBC chains with IV to hide patterns.
- Hash vs HMAC vs signature? Hash=integrity fingerprint; HMAC=integrity+shared-key auth; signature=integrity+public-key auth+non-repudiation.

- This comprehensive study guide combines the core concepts from your lecture slides, homework assignments, and sample exam questions to help you prepare for your Cybersecurity Midterm 1.
- **Cybersecurity Midterm 1: Comprehensive Study Guide**
- **1. Cybersecurity Foundations**
- Cybersecurity is the practice of protecting systems, networks, programs, and data from digital attacks, unauthorized access, or damage.
- **The C-I-A Triad**
- The foundational principles of cybersecurity encompass the entirety of information protection required to keep data secure and systems operational.
  - **Confidentiality (Secrecy/Privacy):** Ensures that information cannot be accessed or viewed by unauthorized users.
    - **Need-to-Know (NTK):** Access should be limited to specific resources required for a user's role.
    - **Trade-off:** Increasing confidentiality (e.g., via MFA) can reduce **usability** by creating slower workflows and user frustration.
  - **Integrity (Authenticity):** Ensures information can only be modified by authorized users and prohibits unauthorized changes.
    - **Trade-off:** Ensuring integrity through methods like hashing or digital signatures can impact **system performance** by increasing CPU time and latency.
  - **Availability:** Ensures that information is accessible to authorized users within an acceptable timeframe.
    - **Methods:** Physical protection (infrastructure) and computational redundancies (fallback devices).
    - **Trade-off:** High availability involves high **financial costs** for duplicate systems and 24/7 support.
- **Key Terminology**
  - **Asset:** Anything of value that needs protection, such as data, systems, or personnel.
  - **Vulnerability:** A flaw or weakness in a system's design or operation that could be exploited.
  - **Threat:** A circumstance or event with the potential to adversely impact assets.
  - **Risk:** An expectation of loss, expressed as the probability that a threat will exploit a vulnerability.
-

- **2. Cryptographic Concepts**
- Cryptography provides the mathematical tools to guarantee C-I-A goals.
  - **Symmetric Cryptography:** Uses a single secret key shared between parties for both encryption and decryption.
  - **Asymmetric (Public-Key) Cryptography:** Uses a public key for encryption and a private key for decryption (e.g., RSA).
  - **Hash Functions:** Deterministic algorithms that map arbitrary-sized data to a fixed-size digest used for integrity.
    - **Note:** SHA-1 is now considered **insecure** because its collision resistance is no longer acceptable.
  - **Digital Signatures:** Provide integrity, authentication, and non-repudiation.
  - **Attacks:**

    - **Brute-Force Attack:** Trying all possible private keys until the correct one is found.
    - **Cryptanalytic Attack:** Exploiting the nature of the algorithm or characteristics of the plaintext.
- **OpenSSL Practical Commands**
  - **Generate Private Key (RSA):** openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -out private.pem.
  - **Extract Public Key:** openssl pkey -in private.pem -pubout -out public.pem.
- ---
- **3. Operating System Security**
- Modern systems are complex and dynamic; therefore, software vulnerabilities are considered **inevitable**.
  - **Defense in Depth:** Building security protections at multiple layers (Hardware, Kernel, OS, and Application).
  - **Least Privilege:** Users and programs should only have the minimum privileges necessary to perform their tasks.
  - **Privilege Management:**
    - **Root:** The most powerful user in a Linux system.
    - **setuid bit:** Allows a program to run with the permissions of the file owner (often root) to enable controlled privilege escalation.
    - **UAC (User Account Control):** Windows prompt that appears when an action requires administrator privileges.
  - **System Hardening:** The process of securing a system by minimizing its attack surface (e.g., disabling unnecessary services).
- ---
- **4. Ethical Hacking & Incidents**
  - **Hacker Types:**
    - **White Hat:** Uses skills for defensive purposes and the common good.
    - **Black Hat:** Resorts to malicious activities for personal gain.
    - **Grey Hat:** Behavior is unpredictable, working both offensively and defensively.
  - **Hacking Process:** Footprinting $\rightarrow$ Scanning $\rightarrow$ Gaining Access $\rightarrow$ Maintaining Access.
  - **Real-World Case Study: Equifax Breach (2017):**
    - **Impact:** Compromised **Confidentiality** for millions due to unpatched vulnerabilities.
    - **Lesson:** Security maintenance must be a **continuous** process.
- ---
- **5. Sample Question Flashcards**
  - **Q: Is RSA a block cipher?**
    - A: **False.** RSA is an asymmetric/public-key scheme; AES is a symmetric block cipher.
  - **Q: Are modern OSs immune to buffer overflow attacks?**
    - A: **False.** Vulnerabilities are inevitable; bugs like heap overflows still occur.
  - **Q: What does the authentication function do?**
    - A: It determines the **identity** of a user (who they are), not who is trusted for a purpose.
  - **Q: When is the security process complete?**
    - A: Never; it is a **continuous** process.