

Study Guide and Self-Check

Learning objectives, key terms and self-check questions for all fifteen chapters. The questions here are separate from the book's exercises, and answered.

Accompanies *Computer Security: The Foundations*, Krerk Piromsopa, Ph.D., 2026.
<https://www.cp.eng.chula.ac.th/~krerk/books/ComputerSecurity/>

How to use this guide

For each chapter: read the objectives first, so you know what you are reading for. Read the chapter. Then cover the answers and work the self-check questions — these test recall and understanding, not the analysis the book's exercises ask for. If a term in the *Key terms* list is unfamiliar after reading, that is exactly the term to return to.

The self-check questions are not the graded exercises. They are shorter, they have answers, and they exist so you can tell whether you have understood a chapter before you attempt the exercises that have not.

1 Introduction

After this chapter you can

- state a threat model as assets, adversaries, capabilities and goals;
- distinguish security from privacy, and say why each needs the other;
- name the CIA triad and give a control that trades one leg for another;
- name the four pillars and express risk as likelihood $\times$ impact.

Key terms

security · privacy · CIA triad · confidentiality · integrity · availability · authentication · authorization · auditing · risk · threat · vulnerability

1. Why is “is this system secure?” not a well-formed question?

Answer: Security is relative to a threat model. Without an asset, an adversary and that adversary's capability, there is nothing to be secure against.

2. A system faithfully protects a database it should never have collected. Which property is satisfied and which is violated?

Answer: Security may be satisfied; privacy is violated. Security is necessary for privacy but not sufficient.

3. Give the risk equation, and say why a vulnerability with no threat actor carries no risk.

Answer: Risk = likelihood $\times$ impact. With no actor, the likelihood of exploitation is zero, so the product is zero.

4. Name the four responses to a risk.

Answer: Mitigate, transfer, avoid, accept.

2 Authentication

After this chapter you can

- name the authentication factors and say when two are truly independent;
- explain salting and why a slow hash is used to store passwords;
- explain why biometrics are identifiers, not secrets;
- say what makes passkeys/FIDO2 phishing-resistant.

Key terms

authentication · biometrics · multi-factor authentication · passkey · salt · pepper · rainbow table · credential stuffing · social engineering

1. Why is a per-user salt stored in plaintext next to the hash, and what does it defeat?

Answer: The salt is not a secret; it makes every stored hash unique, so a precomputed (rainbow) table is useless and each password must be attacked separately.

2. Why is a fast hash such as SHA-256 the wrong choice for storing passwords?

Answer: It is fast: commodity GPUs compute billions per second, so a leaked database is cracked cheaply. Use a slow, memory-hard KDF (Argon2, bcrypt, scrypt).

3. Why can a stolen fingerprint not be revoked, and what does that imply for remote authentication?

Answer: A biometric is a fixed physical identifier, not a changeable secret; you cannot reissue it. It is suited to a local trusted device, not to a remote shared secret.

4. A user on `chula-login.example` is phished. Why does a passkey for `chula.ac.th` not sign in?

Answer: The authenticator binds the signature to the requesting origin. The browser will not produce a `chula.ac.th` signature for a request from a different origin, so there is nothing to steal or replay.

3 Authorization

After this chapter you can

- separate policy from mechanism;
- distinguish MAC, DAC, RBAC and ABAC by who decides;
- contrast ACLs and capabilities, especially on revocation;
- state the Bell–LaPadula and Biba rules and what each protects.

Key terms

access control · MAC · DAC · RBAC · Bell–LaPadula model · Biba model · least privilege

1. Who assigns the label in MAC, and who in DAC?

Answer: In MAC the system (policy) assigns it and users cannot change it; in DAC the resource owner does, and may delegate.

2. You want to answer “what can Alice reach?” quickly. ACL or capability?

Answer: Capabilities store rights by subject, so enumerating one subject's rights is direct; an ACL stores by object and answers "who can reach this?" instead.

3. Why is revoking a bearer token before it expires hard?

Answer: A token is a capability: the right travels with the holder and is not re-checked against a list, so revocation needs extra machinery (short lifetimes, a deny-list, a checked refresh token).

4. State the two Bell–LaPadula rules in the "no read up / no write down" form, and say what they protect.

Answer: No read up, no write down. They protect confidentiality: information flows only upward in classification.

4 Auditing

After this chapter you can

- say what a usable log record must contain;
- explain why logs are shipped off-host and chained;
- contrast signature and anomaly detection;
- explain the base-rate fallacy for IDS alerts.

Key terms

auditing · IDS · SIEM · DMZ · firewall · threat hunting · chain of custody

1. Name two failure/success outcomes a log must record, and why logging only failures is dangerous.

Answer: Both successes and failures. A successful compromise produces successful operations; if only failures are logged, the intrusion is invisible.

2. Why send logs to a separate host immediately?

Answer: On a compromised machine the local logs are attacker-controlled and are edited or truncated first; an off-host, append-only copy survives.

3. A detector is 99% accurate; genuine attacks are one event in a million. Why are most alerts still false?

Answer: The base rate of attacks is tiny, so the many false positives from the vast benign population swamp the few true positives. Precision, not accuracy, is the figure that matters.

4. Which DMZ traffic direction is denied by default, and why is that the rule that matters?

Answer: DMZ → internal. It ensures a compromised public server does not become a route into the internal network.

5 Integrity

After this chapter you can

- distinguish data integrity from system integrity;
- state the three hash properties and which falls first;
- say why a bare checksum does not prove tampering;
- choose between a MAC and a signature by non-repudiation.

Key terms

integrity · hash function · HMAC · digital signature · code signing

1. Name the three hash properties, and which is easiest to break.

Answer: Pre-image, second pre-image and collision resistance. Collision resistance falls first, at the birthday bound $2^{n/2}$.

2. Why does a checksum posted next to a download not prove the file is untampered?

Answer: An attacker who can change the file can change the checksum. Detecting tampering needs an independent path — a signature over a trusted key, or a separate channel.

3. You need a third party to be convinced who signed a message. MAC or signature?

Answer: A digital signature: it gives non-repudiation. A MAC uses a shared secret, so either holder could have produced it.

4. A signed but malicious software update is distributed. What did the signature prove, and what did it not?

Answer: It proved origin and that the artefact was unmodified since signing. It did not prove the code is safe.

6 Cryptography**After this chapter you can**

- state Kerckhoffs's principle;
- say why ECB is unsafe and why AEAD is the default;
- explain the key-distribution problem public key solves, and the binding problem it creates;
- explain “sign the hash”, forward secrecy, and the PQC threat.

Key terms

cryptography · symmetric encryption · asymmetric encryption · digital certificate · PKI · TLS

1. State Kerckhoffs's principle.

Answer: A system must stay secure when everything about it except the key is public.

2. Why must ECB mode never be used?

Answer: Identical plaintext blocks map to identical ciphertext blocks, so structure in the data survives encryption (the “ECB penguin”).

3. What does authenticated encryption (AEAD) add over plain encryption, and why does it matter?

Answer: Integrity and authenticity of the ciphertext. Without it, an attacker who flips ciphertext bits can flip plaintext bits undetected.

4. Public-key cryptography removes the shared-secret problem. What new problem does it create, and what solves it?

Answer: The binding problem — whose key is this? — solved by certificates and a PKI chain of trust.

5. Why does “harvest now, decrypt later” mean PQC migration is urgent for some data?

Answer: Traffic recorded today can be decrypted once a quantum computer exists, so anything with a long confidentiality lifetime is already at risk.

7 Software Security

After this chapter you can

- threat-model with STRIDE against a data-flow diagram;
- recall the Saltzer–Schroeder principles;
- explain attack-surface reduction and defence in depth;
- say why client-side checks are never a security control.

Key terms

least privilege · attack surface · defence in depth · secure defaults · supply chain

1. What do the six letters of STRIDE stand for?

Answer: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege.

2. Why is deleting a feature one of the highest-yield security actions?

Answer: It reduces the attack surface: code that is not there has no vulnerabilities.

3. Three WAF rules are not defence in depth. Why not?

Answer: They are one layer. Depth requires independent layers that fail separately (e.g. a WAF, parameterised queries, and least-privilege DB credentials).

4. Why is a price validated only in client-side JavaScript not a control?

Answer: Anything the client can send, an attacker can send; the check must be re-made server-side on data the server can vouch for.

8 Network Security

After this chapter you can

- name an attack by the OSI layer it lives on;
- explain why the core protocols do not authenticate the sender;
- explain the SYN flood asymmetry and how SYN cookies remove it;
- explain amplification and why egress filtering (BCP 38) helps.

Key terms

firewall · DMZ · Zero Trust · TLS

1. ARP spoofing lets an attacker do what, and at which OSI layer?

Answer: Become a man-in-the-middle on the local segment, at layer 2, by claiming another host's IP with its own MAC.

2. What is the asymmetry a SYN flood exploits, and how do SYN cookies remove it?

Answer: One SYN costs the attacker nothing but ties up server state; SYN cookies encode the state in the sequence number and keep none until the ACK returns.

3. Why does IP spoofing work at all?

Answer: Nothing in the IP header authenticates the source address, and most networks do not filter outbound packets by source (BCP 38).

4. Which single control most reduces the value of a man-in-the-middle position?

Answer: End-to-end encryption: the attacker can relay traffic but not read or alter it.

9 Digital Forensics

After this chapter you can

- order the forensic phases and the order of volatility;
- explain chain of custody and why a hash proves the copy;
- say why deleted data persists, and how SSDs change that;
- explain why timelines are normalised to UTC.

Key terms

chain of custody · hash function · anti-forensics

1. In what order is evidence collected, and give the most volatile source.

Answer: Most volatile first: CPU registers/cache, then RAM, then network and process state, then disk, then archival media.

2. Why hash a disk image at acquisition?

Answer: A matching hash at every handover proves the working copy is identical to the original — the basis of chain of custody.

3. Why does a deleted file often remain recoverable on a hard disk but not on an SSD?

Answer: On an HDD, deletion only frees the clusters; the data stays until overwritten. On an SSD, TRIM and wear levelling can erase blocks without the OS asking.

4. Why must a timeline be normalised to UTC before analysis?

Answer: Mixed timezones and unsynchronised clocks otherwise put events in the wrong order, which is usually the whole argument.

10 AI and Machine Learning Security

After this chapter you can

- name the four attack boundaries of an ML system;
- explain adversarial examples and backdoor poisoning;
- explain why prompt injection has no parameterisation fix yet;
- name the dangerous combination for agentic LLMs.

Key terms

adversarial examples · data poisoning · backdoor attack (ML) · prompt injection · large language model · differential privacy

1. Name the four boundaries at which an ML system can be attacked.

Answer: Training data, the training process, the model artefact, and the inference API.

2. Why can a backdoored model pass every accuracy test?

Answer: It behaves normally on all inputs except those carrying the attacker's trigger, which the test set does not contain.

3. Why is prompt injection structurally like SQL injection, and why is it harder to fix?

Answer: Both are data interpreted as control. SQL has parameterised queries to separate the channels; natural-language prompts have no equivalent separation yet.

4. What three capabilities together make an LLM agent dangerous?

Answer: Access to private data, exposure to untrusted content, and the ability to act or communicate externally — the “lethal trifecta”.

11 Buffer Overflow

After this chapter you can

- explain why the return address is next to the buffer;
- distinguish overwriting data from overwriting a code pointer;
- say what canaries, NX and ASLR each stop and how each is answered;
- explain how memory-safe languages remove the class.

Key terms

buffer overflow · stack canary · ASLR · DEP/NX

1. Why does writing past a local buffer reach the return address?

Answer: The buffer fills toward higher addresses while the saved frame pointer and return address sit just above it on the same stack, with no bounds check in C/C++.

2. NX makes the stack non-executable. What technique answers it, and how?

Answer: Return-oriented programming: chain existing code fragments ending in `ret`. No new code is injected, so NX is satisfied.

3. One leaked pointer can defeat ASLR. Why?

Answer: Segment bases are randomised as a whole, so a single leaked address reveals the offset for everything in that segment.

4. What does a stack canary *not* protect against?

Answer: Overwrites that do not cross the canary — function-pointer, vtable and heap-metadata corruption.

12 Input Validation

After this chapter you can

- state the one sentence common to every injection bug;
- validate on input (allowlist) and encode on output by context;
- explain why parameterised queries beat escaping;
- explain why XSS defeats CSRF defences.

Key terms

command injection · SQL injection · path traversal · XSS · CSP · allowlist · denylist

1. State the single sentence that describes every injection vulnerability.

Answer: Attacker-supplied data reaches an interpreter, which parses part of it as instructions rather than as values.

2. Why is a parameterised query safe regardless of the input's content?

Answer: The statement is parsed before the data is bound, so the parameter can never change the parse tree — no escaping is involved to get wrong.

3. Why is an allowlist preferred over a denylist?

Answer: You can enumerate what you need (finite, known); you cannot enumerate every attack. An allowlist fails closed, a denylist fails open.

4. Why does an XSS bug defeat every CSRF defence?

Answer: The attacker's script runs in your origin and can read the anti-CSRF token, so the token no longer protects anything.

13 Physical Security**After this chapter you can**

- explain why physical access defeats software controls;
- recall deter/delay/detect/respond and the delay condition;
- name the attacks needing only minutes of access;
- explain secure disposal and encrypt-then-destroy-the-key.

Key terms

insider threat · least privilege · tailgating

1. Why is “deter, delay, detect, respond” incomplete without a measurement?

Answer: A delay is only useful if it exceeds your response time; both must be measured, or the delay buys nothing.

2. What does a cold-boot attack exploit?

Answer: DRAM retains its contents for seconds after power loss (longer when chilled), so keys can be read out after a quick reboot.

3. Why encrypt a disk from day one even if you never lose it?

Answer: Disposal then becomes destroying the key — fast, verifiable, and possible even on a drive that has failed and cannot be overwritten.

4. Why are insiders unaffected by perimeter controls?

Answer: They already hold credentials and know what is valuable and what is monitored; the controls that work against them are procedural (least privilege, separation of duties, recertification).

14 Malware**After this chapter you can**

- classify malware by behaviour;
- explain why polymorphism ended signature-only detection;

- recall the infection lifecycle and where persistence shows;
- name the control that decides a ransomware outcome.

Key terms

malware · ransomware · rootkit · botnet · APT

1. What distinguishes a worm from a virus?

Answer: A worm self-propagates over a network without user action; a virus needs a host file to be run.

2. Why did detection move away from signatures?

Answer: Polymorphic and metamorphic code changes its bytes every infection, so there is no constant signature to match.

3. Where is stealthy malware most likely to be visible, and why?

Answer: In its persistence mechanism: to survive a reboot it must leave something behind (a run key, a service, a scheduled task).

4. Which single control most decides a ransomware outcome?

Answer: Tested, offline or immutable backups that are not reachable from the compromised credentials.

15 Cloud Security

After this chapter you can

- apply the shared responsibility model;
- explain why IAM is the perimeter;
- explain the SSRF-to-instance-metadata escalation;
- say why container isolation is not VM isolation.

Key terms

cloud computing · shared responsibility model · VPC · container · CSPM · Zero Trust

1. Under the shared responsibility model, what is always the customer's, at every service level?

Answer: The data, the identities and access policies, and the configuration.

2. Why is IAM described as the new perimeter?

Answer: There is no network edge; a validly-credentialled API call from anywhere is indistinguishable from a legitimate one, so identity is the boundary.

3. Describe the SSRF-plus-metadata escalation.

Answer: An SSRF bug makes the server fetch the instance metadata endpoint (169.254.169.254), which returns the instance role's credentials — the Capital One pattern.

4. Why is a shared-kernel container weaker isolation than a VM?

Answer: Containers share the host kernel, so a kernel vulnerability is a container escape; a VM has its own kernel behind the hypervisor.

A note on the case study

Appendix B, *Investigating an IMSI Catcher*, ties Chapters 5, 9, 13 and 14 together in one investigation. Read it last, and ask yourself at each step: at which point could the examiner's conclusion have been challenged, and what preserved it?