Midterm Exam Week: Quiz-1

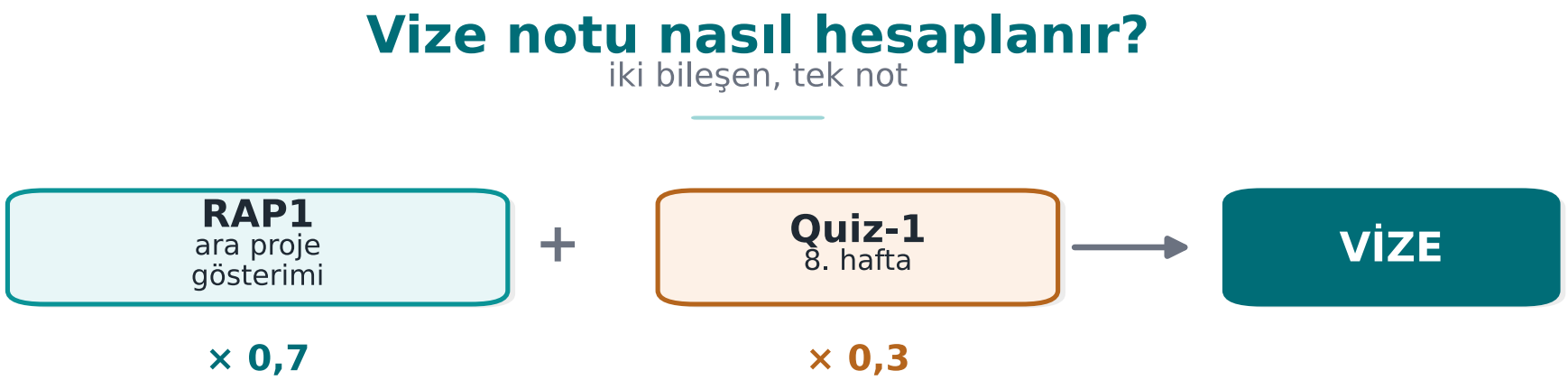
CEN429 Secure Programming — Week 8

Asst. Prof. Dr. Uğur CORUH · 31.10–08.11.2026

Quiz-1: What, When?

- Scope: **weeks 1–6** · **40%** of the midterm
- Date, time, place, duration, question format: announced **in class**
- What's measured: not memorization, but the **why** question
- Best preparation: each week's **Self-check** questions + rerunning the demos

Midterm Grade Calculation — Diagram

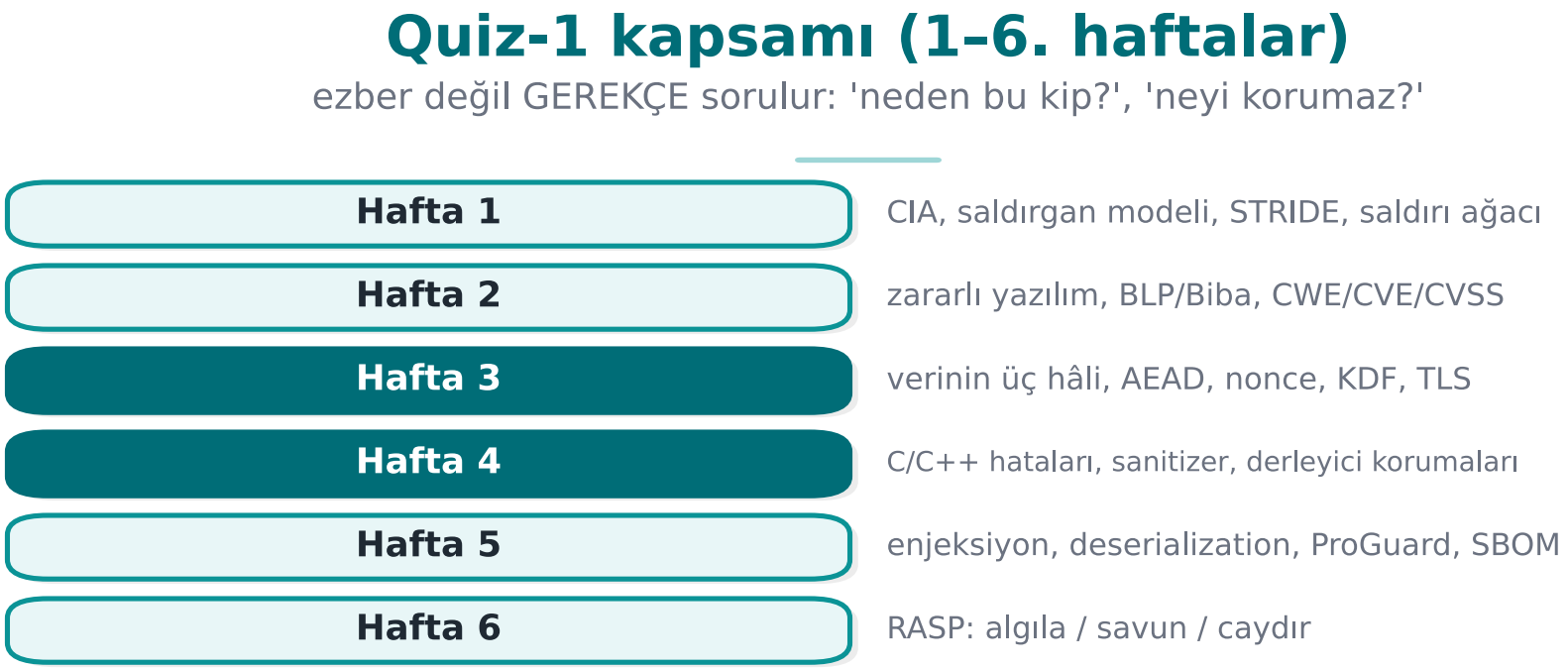


Başarı notu = 0,4 · Vize + 0,6 · Final. Final de aynı biçimde RAP2 ve Quiz-2'den gelir.

Topic Map (1)

Week	Core concepts
1	CIA · asset–threat–vulnerability–risk · white-box · Saltzer–Schroeder · 7 layers · 7-step protection plan · interface/asset table · STRIDE, DFD · attack tree · secure startup · memory errors · secure erasure
2	Virus/worm/trojan · outbreak · rule-based detection · integrity monitoring · BLP, Biba, Clark–Wilson · Unix/Windows ACL · RBAC · CWE, OWASP, CVE, CVSS · log injection
3	AEAD · digest/MAC/signature · CSPRNG · nonce/IV/salt · ECB · Argon2id · HKDF · key lifecycle · TLS 1.3 · pinning · fail-open · masking · shells

Quiz-1 Scope — Diagram



Her haftanın 'Kendini sınama' bölümünü cevaba bakmadan çözün.

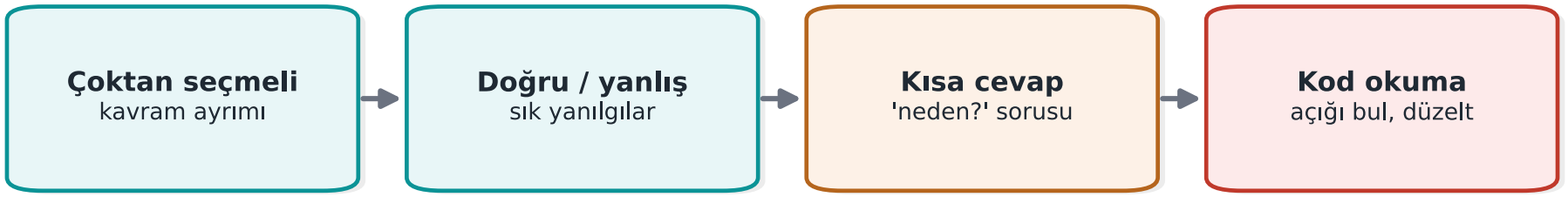
Topic Map (2)

Week	Core concepts
4	SEI CERT · input validation · format string · UAF · integer/UB · error handling, signals · static analysis · sanitizers · fuzzing · canary/FORTIFY/ASLR/NX/RELRO/CFI · introduction to obfuscation
5	Managed language · CERT Java · injection root cause · SQL/command/path · deserialisation · XXE/XSS · Python/JS, ReDoS · bytecode · ProGuard/R8 · SBOM, VEX
6	Detection–defence–deterrence · MATE · RASP architecture · integrity · debugger/environment/hook · memory protection · root/signature · flow counter · response, decoy, device binding

Question Types — Diagram

Quiz-1 soru tipleri

dördü de 'anladın mı' diye sorar, 'ezberledin mi' diye değil



Kod okuma sorusunda önce açığın SINIFINI (CWE) söyleyin, sonra düzeltmeyi.

Frequently Confused (1)

A	B	Difference
Encoding	Encryption	Encoding is keyless, no confidentiality
Digest	MAC	MAC requires a secret key
MAC	Signature	Signature is asymmetric → non-repudiation
Nonce	Salt	Nonce never repeats · salt is per password
PBKDF2/Argon2id	HKDF	Password (slow) · high-entropy secret (fast)
Chain validation	Hostname verification	"Trusted CA?" · "For this server?"
Fail-open	Fail-closed	On error "pass" · on error "reject"

Frequently Confused (2)

A	B	Difference
Canary	ASLR	Overflow detection on return · address randomisation
ASan	UBSan	Memory access · undefined behaviour
Unsigned wraparound	Signed overflow	Defined · **undefined**
Parameterised query	Escaping	The real fix · a second line of defence
ProGuard	String obfuscation	Names · strings
Obfuscation	Secure coding	Delays · fixes
Masking	Tokenization	Hide on display · token bound to a vaulted value
Pseudonym	Anonymous	Still personal data · cannot be linked to a person

A One-Week Plan

Day	What to do
1	Week 1: STRIDE, attack tree, protection plan · Demo 1–4
2	Week 2: BLP, Biba, Clark–Wilson · score a CVSS vector yourself
3	Week 3 (1): AEAD, nonce, KDF, random numbers · Demo 1–5
4	Week 3 (2): key management, TLS, pinning, shells · Demo 6–9
5	Week 4: CERT pairs, format string, UAF, UB, protections
6	Weeks 5–6: injection, deserialisation, ProGuard, SBOM, RASP
7	Work through sample questions against the clock, review mistakes

Study Plan — Diagram

Bir kuralı 'kendi cümlelerinizle' anlatabiliyorsanız öğrenmişsinizdir.

Example: Find the Bug (1)

```
void kaydet(const char *ad) {
    char tampon[32];
    sprintf(tampon, "kullanici=%s", ad);
    syslog(LOG_INFO, tampon);
}
```

```
char parola[64];  oku(parola, sizeof parola);  dogrula(parola);
memset(parola, 0, sizeof parola);  return;
```

Example: Find the Bug (2)

```
String sql = "SELECT * FROM notlar WHERE ogrenci = '" + no + "'";
```

```
EVP_DecryptUpdate(ctx, acik, &n, sifreli, sifreli_n);
kullan(acik, n);
EVP_DecryptFinal_ex(ctx, acik + n, &m);
```

```
try { pinDenetle(zincir); } catch (KeyStoreException e) { e.printStackTrace(); }
```

Example: Scenarios

1. The same **embedded AES key** on every user's device: risks and recommendation?
2. A shipped package is a **debug build**: which findings?
3. "I did not make this transaction": which STRIDE letter, which evidence?
4. A function that extracts an archive: which bugs do you guard against?
5. What was the real problem in Log4Shell, what would have prevented it?
6. Why shouldn't RASP checks be evaluated with a single `if`?

Next Week

Week 9 — Advanced Code Obfuscation and Diversification

- Obfuscation taxonomy, opaque predicates, data encoding
- Virtualisation and compiler-based obfuscation (concept)
- Measuring obfuscation: potency, resilience, cost

Appendix · Week-by-Week Topic Map (1–6)

Week 1 · Key Points

- CIA triad; a bug ≠ a vulnerability.
- MATE (white-box) attacker.
- STRIDE, attack tree, DFD.
- Layered defence; design principles.

Week 2 · Key Points

- Threat modelling steps.
- Trust boundary; least privilege.
- Asset list and C/I/I+ labels.

Week 3 · Key Points

- The three states of data.
- AEAD; nonce reuse is catastrophic.
- CSPRNG; KDF (password → key).
- Key hierarchy; forward secrecy.

Week 4 · Key Points

- Three layers: code → compiler/OS → obfuscation.
- CERT/CWE; UAF; integer/UB.
- Sanitizers (ASan/UBSan), fuzzing.
- Compiler protections (canary, ASLR, NX).

Week 5 · Key Points

- A managed language solves memory errors, not injection.
- SQL/command/path injection → parameterization/argument array.
- Deserialisation; ProGuard/R8; SBOM.

Week 6 · Key Points

- RASP: detect, defend, deter.
- Integrity (self-hashing), anti-debug, hook detection.
- Cross-checking; response policy; device binding.

Frequently Confused

A	B	Difference
Bug	Vulnerability	An exploitable bug
Black-box	White-box (MATE)	Has the program
Symmetric	Asymmetric	Single / paired key
AEAD	Encryption only	+ integrity
Obfuscation	Secure code	Does not fix the bug

Appendix · Solved Practice Questions

Question 1

Why does a MATE attacker make cryptography alone insufficient?

Answer: The key sits in memory; the attacker can run the program and read it. Cryptography provides confidentiality, but in a white-box setting the key is not protected; RASP/obfuscation/white-box cryptography is needed.

Question 2

What happens if the same nonce is used twice in AES-GCM?

Answer: Confidentiality and integrity collapse; the keystream and data can leak. The nonce must always be unique.

Question 3

What definitively prevents SQL injection?

Answer: A parameterised query (prepared statement); data is never interpreted as a command. No string concatenation.

Question 4

Which overflow can ASan not see?

Answer: A field-to-field overflow within the same structure (there is no poisoned region in between).

Question 5

What does a canary stop, and what does it not stop?

Answer: Stops: a stack overflow that reaches the return address. Does not stop: heap overflow, variables before the canary, a leaked canary.

Question 6

Why is RASP self-hashing not enough on its own?

Answer: The attacker can find the digest function and bypass it. Obfuscation + cross-checking + overlapping checks are needed.

Question 7

Why a KDF and a salt when deriving a key from a password?

Answer: A password has low entropy; the KDF slows down guessing, and the salt stops identical passwords from producing the same key (rainbow table).

Question 8

An example of the least-privilege principle?

Answer: The application's DB user should only be able to `SELECT`; it should not have `DROP` privilege. Excess privilege = greater damage.

Question 9

What is the "E" in STRIDE, and one control for it?

Answer: Elevation of privilege. Control: least privilege, authorization checks, secure defaults.

Question 10

Why is deserialisation dangerous, and what is the fix?

Answer: Code can execute during extraction (a gadget chain). Fix: a data format + schema; an allowlist filter if unavoidable.

Example · Multiple Choice

Which of the following is an AEAD mode?

A) ECB B) CBC C) **GCM** ✓ D) Raw RSA

Example · True/False

- Obfuscation fixes the bug. **(F)**
- The nonce must be secret. **(F — must be unique, not secret)**
- A parameterised query prevents SQL injection. **(T)**

Study Plan

- Do each week's "Self-check" without looking at the answers.
- Learn frequently confused pairs by their **reason**, not by rote.
- Review the Week 3 and Week 4 demos once more.

Final Word (Quiz-1 Prep)

> Not memorization — what's measured is the **why**: "why this mode?", "why this check?", "what doesn't it protect?"

Good luck.