Security Requirements

CEN429 Secure Programming — Week 13

Asst. Prof. Dr. Uğur CORUH · 11.12.2026

Today's Plan (3 Hours)

Hour	Section	Topic
1	0–3	Basic concepts · a good requirement · traceability · requirement block · deferred requirements
2	4–5	Common Criteria (TOE, ST, PP, SFR/SAR, EAL) · FIPS 140-3
3	6–9	ETSI, GSMA, EMVCo, PCI, MASVS · the course's requirement families · compliance matrix · project

A Brief History — How Did Security Requirements Become Standardised?

- **1985** — **TCSEC** formalises requirement levels
- **1994** — **FIPS 140** cryptographic module requirements (today **140-3**)
- **1999** — **Common Criteria**: **PP/ST**, **SFR/SAR**, **EAL**
- **2010s** — **OWASP MASVS** (mobile), **ETSI EN 303 645** (IoT), **EMVCo/PCI** (payment)

> The unchanging principle: a requirement must be **measurable** and **traceable**; a "met" with no evidence is invalid.

Where Does This Week Fit In?

- **Week 12:** how a product is **evaluated/tested**.
- **This week (13):** where the **requirements** that evaluation measures come from, how they're written, how they're traced.
- Output: your project's **compliance matrix** (S17) and **deferred requirements** (S14).

Learning Outcome

This week is about **LO.7**.

By the end, you will be able to:

- Tell a good security requirement apart from a bad one
- Build a **traceability/compliance matrix**
- Compare what CC, FIPS, ETSI, EMVCo, PCI, and MASVS ask for

Main Idea

> Security is not a "feeling"; it is managed with **written, measurable, traceable** requirements: requirement → control → verification → evidence.

Today we will learn to build this chain.

0. Basic Concepts (From Scratch)

What Is a Requirement?

- **Requirement:** a condition the system **must meet**.
- A security requirement: a security condition.
- A good requirement is **verifiable** (testable).

Three Types of Requirement

- **Functional:** which security function will exist? (e.g., data is protected with AEAD)
- **Assurance:** how will we trust it was done correctly? (e.g., a test report)
- **Process:** how must the organisation operate? (e.g., every change is reviewed)

Good vs. Bad Requirement

- **Bad:** "The application must be secure." (unverifiable)
- **Good:** "The release build must be produced with a stack canary, PIE, and full RELRO." (measurable)

Traceability

- **Traceability:** linking each requirement to a **control**, a **test**, and **evidence**.
- "Where was this requirement met, how was it verified?"

Compliance Matrix

- **Compliance matrix:** a requirement → status → section → verification → evidence table.
- The project's **S17** section.
- The first thing an evaluator looks at.

Requirement Statuses

- **Met:** the product satisfies this requirement (with evidence).
- **Deferred:** another party satisfies it (to whom, why, how).
- **Not met:** not yet satisfied.

Deferred Requirement

- If a component **cannot meet** a requirement, it **defers** it to the parent application/OS.
- The guide states **to whom**, **why**, and **how** it will be met.
- The project's **S14** section.

Common Criteria (CC)

- **Common Criteria (ISO/IEC 15408):** the standard for product security evaluation.
- Concepts: TOE, ST, PP, SFR, SAR, EAL (shortly).

CC · Basic Terms

- **TOE:** the product being evaluated.
- **ST (Security Target):** this product's security-target document.
- **PP (Protection Profile):** a common requirement set for a product **class**.

CC · SFR, SAR, EAL

- **SFR:** security **functional** requirements.
- **SAR:** security **assurance** requirements.
- **EAL:** the **depth** level of the evaluation (EAL1–EAL7).

FIPS 140-3

- **FIPS 140-3:** the standard for validating cryptographic **modules**.
- Security levels (1–4).
- It covers only the module, not the whole application.

Sector Standards

- **ETSI EN 303 645:** baseline IoT security.
- **GSMA, EMVCo, PCI:** mobile/payment.
- **OWASP MASVS:** mobile application requirements.

Requirement vs. Control · "Not Met" vs. "Not Applicable"

- **Common mistake:** "Requirement: AES-256-GCM must be used." → this is a **control**, not a requirement. Correct: "...must be protected with AEAD" (requirement) + "AES-256-GCM, S7.2" (control).
- **Not met:** the requirement applies but is not yet satisfied — a **gap**, written into residual risk.
- **Not applicable:** the requirement does not apply to the product at all (e.g., "data in transit" requirements if it doesn't use the network) — must always be written **with a rationale**.

Status and Decision Words — Summary

Word	When to use it	What must accompany it
Met	The product satisfies the requirement itself	Control + verification + evidence
Deferred	Another party satisfies it	To whom + why + how
Not met	Applies but not yet satisfied	Residual risk + planned fix
Not applicable	Does not apply to the product at all	Rationale
must (MUST)	Mandatory	A direct finding if not met
should (SHOULD)	Strong recommendation	A written rationale if not met

Now We're Ready

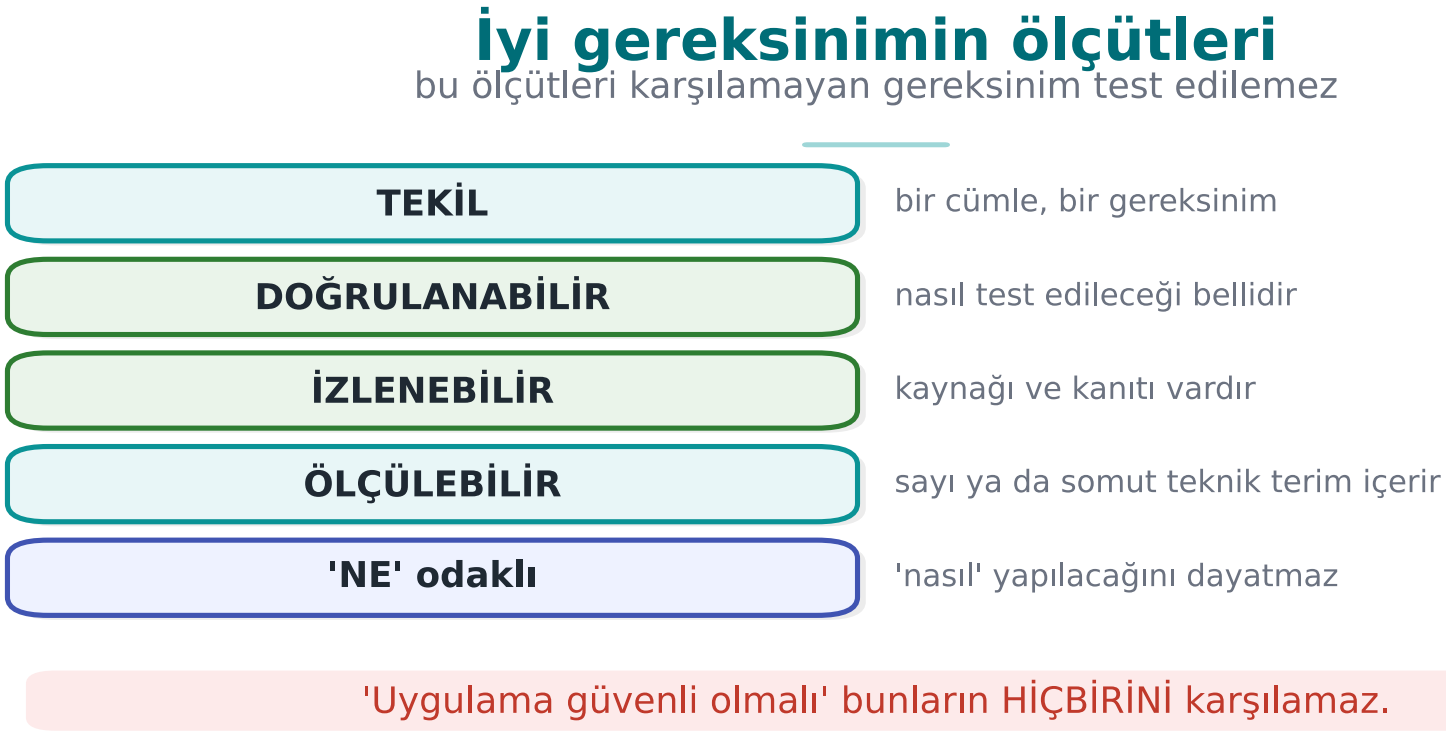
Terms:

requirement (functional/assurance/process) · traceability · compliance matrix · status (met/deferred/not met) · deferred · CC (TOE/ST/PP/SFR/SAR/EAL) · FIPS 140-3 · ETSI/GSMA/EMVCo/PCI/MASVS

Now: how do you write a good requirement?

1. How Do You Write a Good Requirement?

Criteria of a Good Requirement — Diagram



'Uygulama güvenli olmalı' bunların HİÇBİRİNİ karşılamaz.

Three Types (Recap)

Type	Question	Example
Functional	Which security function?	Sensitive data is protected with AEAD
Assurance	Was it done correctly?	Static analysis + test report
Process	How does the organisation work?	Every change is reviewed

Criteria of a Good Requirement

- **Verifiable:** can it be tested?
- **Singular:** does it state a single thing?
- **Measurable:** is it concrete?
- **Feasible:** is it achievable?

Bad → Good (1)

- **Bad:** "The application must be secure."
- **Good:** "The release build must be produced with a stack canary, PIE, and full RELRO."

Why is it good? **Testable** (checksec).

Bad → Good Requirement — Diagram

Kötü gereksinim → iyi gereksinim

fark: ölçülebilir dayanak ve tekil ifade

KÖTÜ
'Uygulama güvenli olmalı'
'Anahtarlar iyi yönetilmeli'
'Veriler uygun saklanmalı'

→ doğrulanamaz

İYİ
'C sınıfı her varlık beklemede
en az 128 bit AES-GCM ile
şifrelenmelidir'

→ test edilebilir

Zayıf gereksinim test edilemez; test edilemeyen gereksinim karşılandığı kanıtlanamaz.

Bad → Good (2)

- **Bad:** "Data must be encrypted."
- **Good:** "Every class-C asset must be encrypted at rest with AEAD at a level of ≥128 bits."

Why is it good? Which data, which level, which method is clear.

Bad → Good (3)

- **Bad:** "Keys must be well managed."
- **Good:** "(1) Every key's purpose, cryptoperiod, and destruction are documented. (2) It is erased from memory once its job is done."

Singular, verifiable items.

Worked Example · Improving a Bad Requirement

Starting sentence (from a real draft):

> "The application must protect user data."

We will turn this sentence into a good requirement in six steps; at every step, **before/after** and **why it changed**.

Step 1 — Flag the Vague Words

- **Before:** "The application must protect user data."
- Vague words: "user data" (which data?), "must protect" (against what, at what level?).
- **Why it changed:** when an evaluator reads this sentence, they find nothing to test.

Step 2 — Link It to the Asset Table

- **After:** "**Class-C** fields in the local database must be protected."
- **Why it changed:** "user data" is not a single thing; it is separate rows in the asset table (Week 1) — session token, profile, payment token... each in a different class.
- "Must be protected" still isn't measurable → next step.

Step 3 — Clarify the Protection Goal

- **After:** "...the **confidentiality and integrity** of class-C fields must be protected."
- **Why it changed:** the threat model shows both "reading the file" (confidentiality) and "modifying the file" (integrity) as risks; both are needed at the same time.

Step 4 — Add a Measurable Technical Criterion

- **After:** "...must be protected with authenticated encryption (**AEAD**)."
- **Why it changed:** Week 9's rule — if confidentiality and integrity are both wanted **at the same time**, the right tool class is AEAD; still no specific library name is given.

Step 5 — Add the State and the Obligation Keyword

- **After:** "Sensitive class-C data **at rest** must be protected with authenticated encryption (AEAD)." (**MUST**)
- **Why it changed:** the data here is at rest (on disk); "must" was chosen because this is an indispensable condition against the threat — if it were "may," it would be treated as optional.

Step 6 — Give It an Id and Link It to the Threat

- **After (final form):** `CEN429-DR-01` — "Sensitive class-C data at rest must be protected with authenticated encryption (AEAD)." *(Threat: T-03, "an attacker who compromises the device reads/modifies the database file".)*
- **Why it changed:** a requirement with no id never enters the traceability chain.

Six Steps · Which Ambiguity Did Each Remove?

Step	Ambiguity removed
1	Flagging the vague word
2	Which data (asset table)
3	Which goal (confidentiality/integrity)
4	Which tool class (AEAD)
5	Which state, which obligation
6	Id and threat link

If any one is missing, the requirement stays open to debate; the evaluator sends it back.

Second and Third Examples

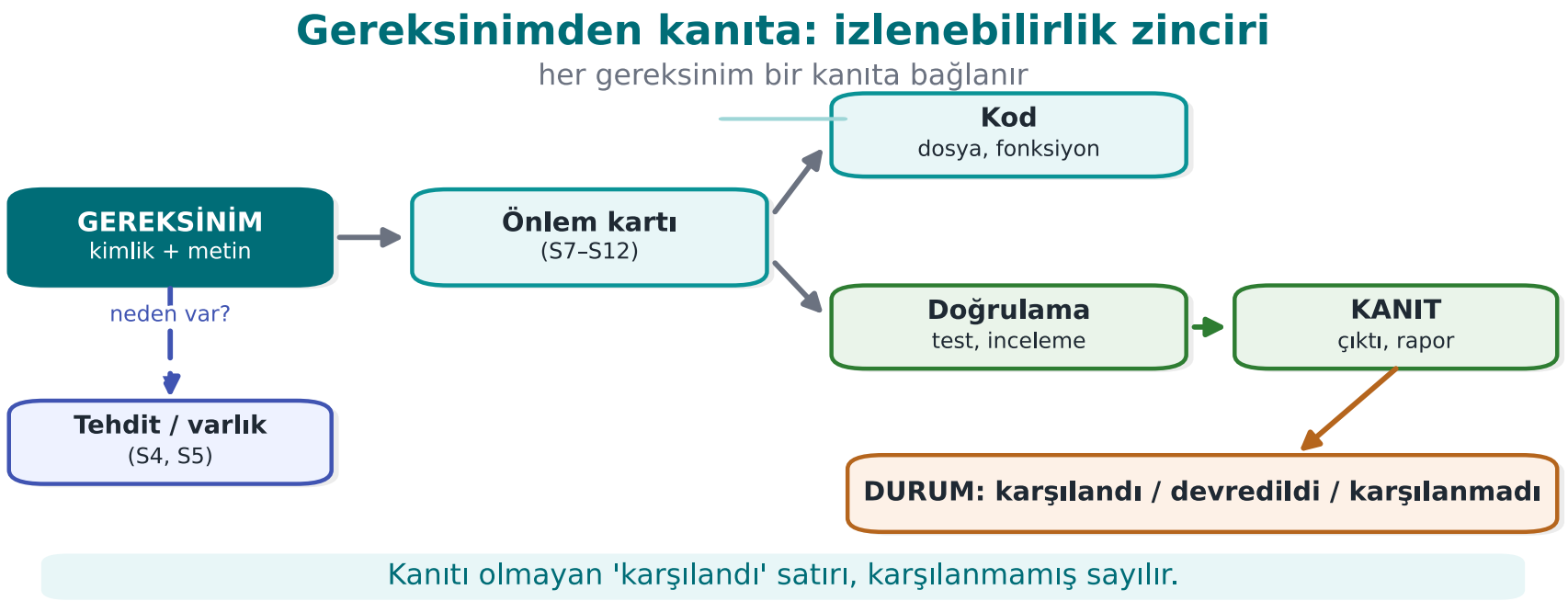
- **Second example (speed test):** "Must be fast and secure" → two separate requirements in one sentence. Run through the same six steps: `CEN429-ID-04` — "Every login attempt must be validated server-side; client-side checking alone must not be considered sufficient."
- **Third example (process):** "Code review must be performed." → Which change? Who reviews it? When? Good version: "Every change must be reviewed by at least one person other than the developer who wrote it, before being merged into the main branch, and the approval must be recorded in the version control system."

Section 1's Rule and the Obligation Keywords

- Write **what** is being asked for in the requirement column; write **how** you meet it in a separate control sentence.
- Write the **tool class** (AEAD, CSPRNG, TLS 1.2+) in the requirement text; put a specific library/version name in the control.
- **must (MUST):** mandatory, a direct finding if not met. **should (SHOULD):** strong recommendation, needs a rationale. **may (MAY):** optional, not a finding.

2. From Requirement to Evidence: Traceability

The Chain



Every requirement must be linked to **evidence** through this chain.

Why Is Traceability Important?

- The evaluator asks "where is this requirement?"
- Without traceability: searching everything from scratch.
- With it: found directly from the matrix.

A Traceability Example

Requirement	Control	Verification	Evidence
Data with AEAD	AES-GCM	Unit test	Test output
Release protections	Flags	checksec	Protection table

Two Directions

- **Forward:** requirement → where was it met?
- **Backward:** this code/test → which requirement does it meet?
- A good matrix is traceable in **both directions**.

⚠️ A "Met" With No Evidence

- If the matrix says "met" but there is no evidence...
- In the evaluator's eyes it counts as **not met**.
- It becomes one of the first findings.

Worked Example · An End-to-End Chain

Let's fill in how the chain is built, start to finish, through a single asset: the user's **session token**.

Six links: asset → threat → requirement → control → verification → evidence.

1. Asset

Field	Value
Asset	The user's session token
Location	Client memory; carried in the HTTP header on every request
Class	**C** (confidentiality)
Lifecycle	Generated at login → used on every request → invalidated at logout/timeout

2. Threat

- An attacker positioned on the network (e.g., on the same public Wi-Fi) can listen on an unencrypted connection and capture the token.
- Using the captured token in their own requests, they can **act as the user**.
- STRIDE: **Spoofing** + **Information Disclosure** (Week 1 terminology).

3. Requirement

The asset → threat → goal (confidentiality+identity) → tool class → state → obligation chain is briefly reapplied:

> `CEN429-DT-01` — "The session token must be carried over a channel using TLS 1.2 or higher, with the server certificate validated."

4. Control

The guide's **S10.3 "Transport security"** section:

> "The client performs a TLS 1.2+ handshake on every connection to the server; the certificate chain and hostname are validated (see Week 10). No request is sent before the channel is established."

5. Verification

The security testing team tries to intercept the connection with a MITM (man-in-the-middle) tool:

- Verifies the connection is **rejected** when an invalid/self-signed certificate is presented.
- Verifies a plaintext (non-TLS) connection attempt is **rejected**.

6. Evidence

- The test tool's log output: `mitm_test_2026-11-03.log`
- CI run record: **#617**
- TLS handshake packet capture: `handshake.pcapng`
- Stored in the `evidence/week13/` folder, referenced by these file names in the matrix.

The Chain's Filled-In Row

Id	Requirement	Status	Control	Verification	Evidence
CEN429-DT-01	Session token must be carried over TLS 1.2+ with certificate validated	Met	S10.3	MITM test: invalid certificate/plaintext rejected	`mitm_test_2026-11-03.log`, CI #617, `handshake.pcapng`

If one link were missing: without the asset, "which data" stays unclear; without the threat, the requirement looks arbitrary; without the control, the claim is unsupported; without verification, "how was it tested" goes unanswered; without evidence, the row becomes the **first finding**.

The Full Chain of the "Not Met" Status

Link	Met (DT-01)	Not met (DT-04)
Requirement	Session token carried over TLS	Server certificate must be pinned
Status	Met	**Not met**
Control	S10.3	— (not yet)
Verification	MITM test passed	—
Evidence	Test log	**Residual risk:** written into S16.4

In "not met," the control/verification can stay blank, but the **evidence** (where the residual risk is documented) cannot.

Section 2's Rule

- Every "met" row's evidence column must have a **findable, named** reference: a file name, a CI number, a test report section.
- "Exists," "tested" is not evidence, it is a **promise** of evidence.
- The evaluator's most common finding: **a "met" with no evidence**.

Sections 1–2 — Quick Check

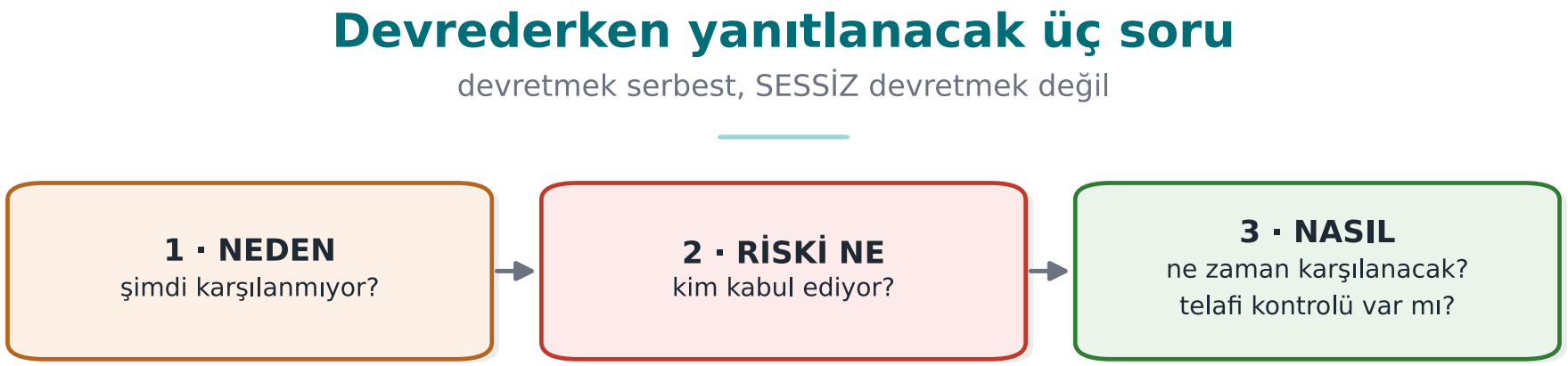
1. What are the four criteria of a good requirement?
2. Why is "the application must be secure" bad?
3. What is the traceability chain?

Sections 1–2 — Answers

1. **Specific · measurable/verifiable · singular · traceable** (feasible, atomic).
2. **Vague and unmeasurable**; "secure" has no definition → cannot be tested, everyone understands it differently.
3. **Threat/standard → requirement → design/code → test/evidence.** Every requirement is linked **both ways** to a source and to evidence.

3. The Requirement Block and Deferred Requirements

Three Deferral Questions — Diagram



Kayıt ve onay olmadan atlanan gereksinim = gizli açık, sahipsiz risk.

The Requirement-Block Pattern

Every requirement is written as a **block**:

- **Id:** a unique number (e.g., CEN429-DU-01)
- **Text:** what is asked for (singular, measurable)
- **Status:** met / deferred / not met
- **Compliance:** how it was met
- **Verification + evidence**

Example Block

```
CEN429-DU-01
Metin: Yerel DB'deki C sınıfı veri AEAD ile şifrelenir.
Durum: Karşılandı
Karşılama: AES-256-GCM, anahtar TEE'de
Doğrulama: T-05 birim testi
Kanıt: test çıktısı, S16
```

Why a Block?

- Every requirement has the same structure → **comparable**.
- The evaluator reads it fast.
- The compliance matrix is produced from these blocks.

What Is a Deferred Requirement?

- A component **cannot meet** a requirement **itself**.
- It **defers** the responsibility to the parent application/OS/hardware.
- This is not an escape, it is an **explicit contract**.

Three Questions When Deferring

Write, for every deferred requirement:

- **To whom?** (parent application / OS / hardware)
- **Why?** (why this component cannot meet it)
- **How?** (how the other party will meet it)

Deferred · Example

```
CEN429-AP-03: Güvenli kurulum ve güncelleme
Durum: Devredildi
Kime: Üst mobil uygulama (MPA)
Neden: SDK dağıtım kanalına sahip değil
Nasıl: MPA imzalı güncelleme + sürüm denetimi sağlar
```

⚠️ No Silent Deferring

- If you don't meet a requirement and **don't write it down** → a gap.
- Deferring must be **documented**; otherwise it counts as "not met."
- The S14 section exists exactly for this.

Met, or Deferred?

Status	Meaning
Met	The product provides it (with evidence)
Deferred	Another party provides it (to whom/why/how)
Not met	Not yet there (residual risk)

Worked Example · Filling In the Block (Met)

This is the **written-into-the-guide** form of the `CEN429-DR-01` chain from Section 2:

> [CEN429-DR] CEN429-DR-01 — MET
> Requirement: Sensitive class-C data at rest must be protected with authenticated encryption (AEAD).
> Compliance: The local vault file is encrypted with AES-256-GCM (S7.2); the key is derived from the device's secure storage unit. Verification: a unit test checks that decryption is rejected when a single byte changes. Evidence: CI #482, `evidence/week13/test_butunluk_ci482.log` .

The Same Pattern · Deferred

> [CEN429-AP] CEN429-AP-07 — DEFERRED (to the parent application)
> Requirement: The application must be installed and updated securely.
> Compliance: The library has no distribution/update mechanism of its own. To whom: the parent application developer. Why: the library does not access installation at the network/file-system level. How: store signature validation, or update packages signed with the signing scheme from Week 6.

A deferred block also has a **Compliance** field — but it answers "how must the other party do it, why aren't we doing it" instead.

The Same Pattern · Not Met

> [CEN429-DT] CEN429-DT-04 — NOT MET
> Requirement: The server certificate must be pinned (certificate pinning).
> Status: Currently only standard TLS chain validation exists, no pinning. Residual risk: MITM may be possible if a forged certificate is obtained from a trusted root (S16.4). Planned fix: v1.1 (task #217).

The "not met" block fills in the **Status** and **Residual risk** fields instead of **Compliance**.

The Role of the Block's Fields

- **Header line** ([Family] Id — Status): tells the evaluator's eye quickly where to go.
- **Requirement** line: repeats the "what" from the standard/threat, not reinterpreted.
- **Compliance:** describes the "how" with concrete file/section names.
- **Verification + Evidence:** the last two links of the traceability chain.

Deferring to Hardware and Section 3's Rule

- **If** a secure element/TEE exists: the requirement can be deferred to hardware.
- **If not**: it cannot be deferred — there is no party to defer to; the product must take a software-based control or write "not met" + residual risk. Writing "deferred to hardware" here is an **invalid deferral**.
- **Rule:** the compliance text must always end with a concrete reference; before deferring, verify the other party can **actually** meet it.

Section 3 — Quick Check

1. What are the requirement block's fields?
2. Which three questions are answered when deferring?
3. Why is silent deferring a problem?

Section 3 — Answers

1. **Id · measurable statement · source/rationale · related asset/threat · verification method (evidence) · status/priority.**
2. (1) **Why** isn't it met now? (2) **What's the risk, who accepts it** (approval)? (3) **When/how** will it be met (plan/mitigation)?
3. Without a record/approval, the requirement gets skipped → a **hidden gap**, no one owns it, it blows up in the audit. Defer openly, with a rationale, and approved.

4. Common Criteria (ISO/IEC 15408)

Common Criteria — Diagram

Kapsam (ST/TOE) darsa yüksek EAL az şey kapsar. Kapsamı bilmeden EAL karşılaştırılmaz.

What Is CC?

- **Common Criteria:** the international standard for evaluating product security.
- Provides a common **language** and **method**.
- Countries recognise each other's evaluations.

TOE · Target of Evaluation

- **TOE (Target of Evaluation):** the complete product being evaluated.
- Uniquely identified (Week 12): version + binary + hash.
- Out-of-scope components are written down.

ST · Security Target

- **ST (Security Target): this** product's security-target document.
- Contains: threats, objectives, requirements (SFR/SAR).
- Specific to the product.

PP · Protection Profile

- **PP (Protection Profile):** a common requirement set for a product **class**.
- Example: "mobile device PP," "smart card PP."
- STs generally build on a PP.

ST vs. PP

	ST	PP
Scope	Single product	Product class
Written by	Developer	Community/authority
Role	This product's target	Common baseline

SFR · Functional Requirements

- **SFR (Security Functional Requirements):** what the product **will do**.
- Example: encryption, access control, logging.
- Chosen from a standard catalogue.

SAR · Assurance Requirements

- **SAR (Security Assurance Requirements): how we will trust** it was done correctly.
- Example: source review, test depth, vulnerability analysis.
- EAL determines these.

EAL · Evaluation Depth

- **EAL (Evaluation Assurance Level):** depth from 1 to 7.
- A higher EAL = a deeper review.
- "+" shows additional assurance components (EAL4+).

⚠️ A Common Misunderstanding of EAL

- EAL is the **depth** of security review, not the **amount** of security.
- Saying "EAL4+ > EAL2 is more secure" is **wrong**.
- Security depends on the **threats and objectives in the ST**.

Worked Example · ST Skeleton — TOE

Let's fill this in for a mobile payment component in three steps.

Step 1 — Define the TOE:

> TOE: the "CEN429-Pay" library, version 1.0, Android ARM64 build only.
> **Out of scope:** the parent application's interface, server-side components, the operating system kernel.

Writing what's out of scope matters as much as writing the scope.

Threat → Objective → SFR Mapping

Threat	Security objective	SFR family	Course requirement
T.EAVESDROP	Transmitted data must stay confidential/intact	FCS, FTP	CEN429-DT-01
T.TAMPER	The application must check its integrity	FPT	CEN429-AP-04
—	Identity must be verified	FIA	CEN429-ID-02

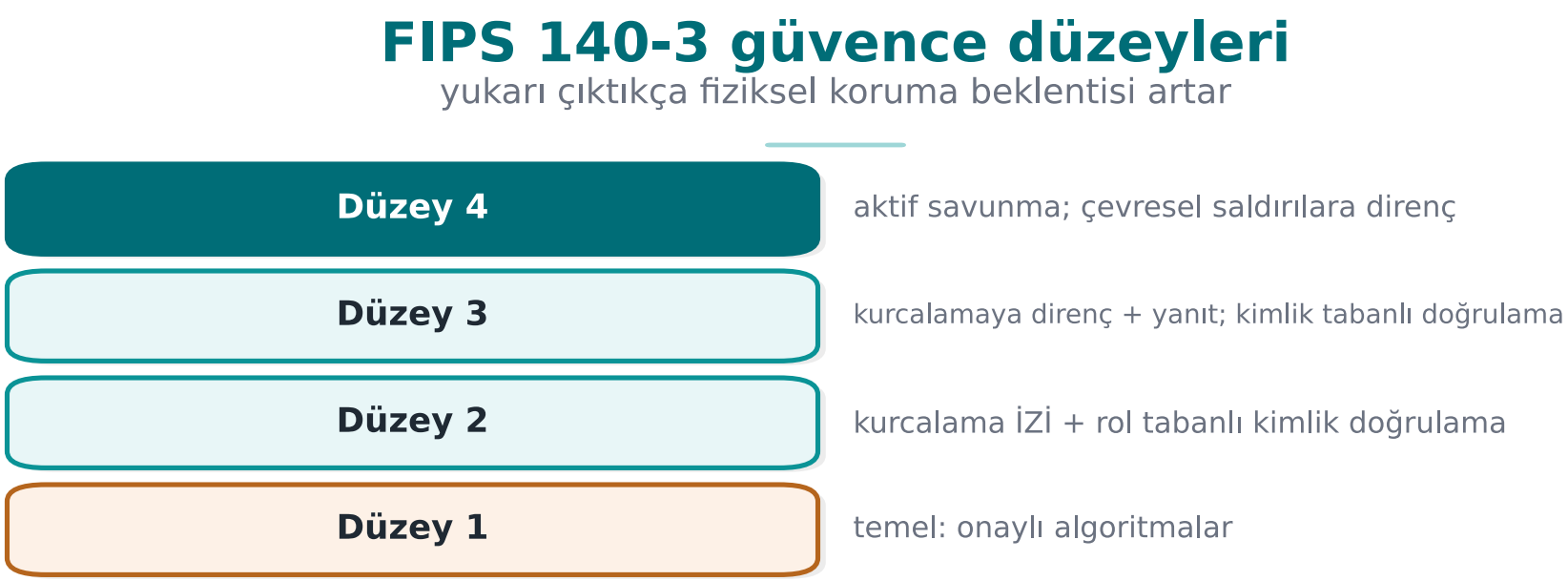
This table is the heart of the ST: every row is a bridge between "why this requirement exists" and "what evidence is requested."

Section 4's Rule

- A TOE definition always lists **both what it includes and what it leaves out**.
- SFRs are chosen from the catalogue, **not invented**; if there's no counterpart, justify it as an "extended component."
- The evaluator checks that every threat links to an objective, and every objective to an SFR.

5. FIPS 140-3

FIPS 140-3 Levels — Diagram



'FIPS doğrulanmış kütüphane kullanıyoruz' demek yetmez: doğru kip ve yapılandırma şart.

What Is FIPS 140-3?

- The standard for validating **cryptographic modules**.
- The module's algorithms, self-test, key management.
- Covers only the **module**.

Security Levels (1–4)

Level	Roughly
1	Basic; approved algorithms
2	Tamper **evidence**
3	Tamper **resistance** and response
4	The highest physical protection

⚠️ The FIPS Compliance Trap

- Using a FIPS-**validated** library does **not** automatically make the application FIPS-compliant.
- The application must use the module in **approved mode** and **correctly**.
- It must also manage keys correctly.

CC and FIPS Together

- **CC:** evaluates the whole product.
- **FIPS:** validates the crypto module.
- A product can use a FIPS-validated module inside a CC evaluation.

Worked Example · A FIPS Level 1 Audit

The project's crypto layer: OpenSSL's `EVP` interface, AES-256-GCM, keys generated from `getrandom()`, and it **silently falls back to a default key on error**.

Let's check off the Level 1 expectations one by one.

Level 1 · Status Table

Level 1 expectation	Status	Why
Approved algorithm, in approved mode	Partial	Can't be claimed without a CAVP certificate
Power-up/conditional self-tests	Not met	The project doesn't run its own self-tests
Roles and services defined	Not met	The code does no role separation
Zeroization of parameters	Not met	Falls back to a default key on error
Security policy document	Not met	No such document exists

The Most Critical Gap and Section 5's Rule

- Falling back to "a default key" on error is the **exact opposite of the zeroization principle**.
- Week 3's rule: "if the random generator fails, **stop**, don't continue" — the opposite is done here.
- **Rule:** a FIPS claim must always have a **CMVP certificate number and scope** behind it; if not, honestly narrow it to "we use FIPS-approved algorithms (not a validated module)."

Sections 4–5 — Quick Check

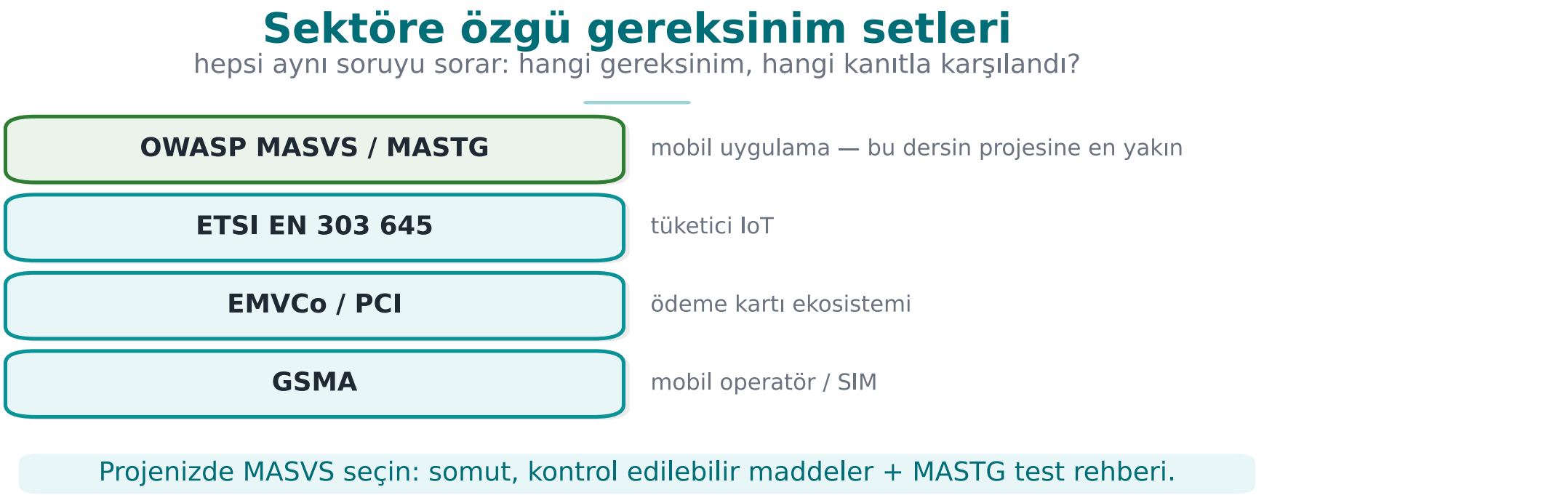
1. What is the difference between ST and PP?
2. What are SFR and SAR?
3. Why is "EAL4+ is always more secure" wrong?
4. Does a FIPS-validated library make the application FIPS-compliant?

Sections 4–5 — Answers

1. **PP** is a standard requirement set for a product **class**; **ST** is the document that says what a specific **product (TOE)** meets.
2. **SFR** is what the product **will do** (functional); **SAR** is **how much it will be trusted** (assurance; EAL comes from here).
3. EAL measures **assurance depth**, not the amount of security; if the **scope (ST/TOE)** is narrow, a high EAL covers little.
4. **No.** The module must be used in the **correct mode/configuration, with approved algorithms**. Module certificate ≠ application compliance.

6. Sector-Specific Requirement Sets

Sector Standards — Diagram



Why Sector Standards?

- CC/FIPS are general; sectors want **additional** requirements suited to their own risks.
- IoT, mobile, payment are different threat environments.
- Follow whichever your project is closest to.

ETSI EN 303 645 (IoT)

- **Baseline** security requirements for consumer IoT.
- Example: no default passwords, secure updates, store sensitive parameters securely.
- A broad, applicable baseline.

ETSI · Example Provisions

- Unique passwords (no common default).
- Manage security updates.
- **Securely store sensitive security parameters** (→ project S5/S7/S8).

GSMA

- Security guidelines for the mobile operator ecosystem.
- Device, network, service security.
- SIM/eSIM, identity.

EMVCo

- Card payment ecosystem standards (cards, terminals, mobile payment).
- Functional conformance + security evaluation.
- Attack potential scoring (Week 12).

PCI

- **PCI DSS:** security for systems that process card data.
- **PCI MPoC/CPoC:** software-based payment acceptance.
- Strict; laboratory evaluation.

OWASP MASVS

- **Mobile Application Security Verification Standard.**
- Mobile application security **requirements**.
- Levels: baseline (L1), defence-in-depth (L2), resilience (R).

MASVS · For Your Project

- The **most applicable** requirement set.
- Tested with MASTG (Week 12).
- Crypto, storage, communication, code quality, resilience.

Standards · Comparison

Standard	Domain	Focus
CC	General product	Evaluation depth
FIPS 140-3	Crypto module	Module validation
ETSI 303 645	IoT	Baseline requirement
EMVCo/PCI	Payment	Functional + security
MASVS	Mobile	Application requirement

Comparison · The Lesson

- All share the same root: a **written, verifiable** requirement.
- The difference: scope and strictness.
- **MASVS** is the most practical starting point for your project.

The Same Requirement, Four Standards (1) — Data at Rest

Standard	Counterpart
Common Criteria	FCS + FDP components
FIPS 140-3	Level 1: approved algorithm, self-test, zeroization
ETSI EN 303 645	Provision 5.4 "Securely store sensitive parameters"
OWASP MASVS	MASVS-STORAGE
The course's family	`CEN429-DR-01`

The Same Requirement, Four Standards (2) — Weak Authentication

Standard	Counterpart
Common Criteria	FIA components
FIPS 140-3	Level 2: role-based authentication
ETSI EN 303 645	Provision 5.1 "No universal default passwords"
OWASP MASVS	MASVS-AUTH
The course's family	`CEN429-ID-01` / `CEN429-ID-02`

Overlap and Section 6's Rule

- Compliance evidence for one standard can be **partly reused** for another — but that doesn't mean "automatically met"; every standard has **its own additional criteria**.
- "We use a FIPS-validated module, so we also meet MASVS-CRYPTO" is a **dangerous** sentence.
- **Rule:** use overlap to speed up gathering evidence, but check off every standard **separately**.

Section 6 — Quick Check

1. Who is ETSI EN 303 645 for?
2. Why is MASVS the most suitable for your project?
3. What is the difference between CC and MASVS?

Section 6 — Answers

1. Baseline security for consumer **IoT** devices (no default passwords, updates, secure communication…).
2. The project is application-focused; **MASVS** has concrete, checkable items + the **MASTG** test guide → directly applicable.
3. **CC** is heavy, accredited lab/certificate (EAL); **MASVS** is a light, open, developer-friendly **self-assessment** standard.

7. Carrying Requirements Into the Project

Requirement Carry-Over Decision — Diagram

Devredilen gereksinim de yazılır: kime, neden, karşı taraf nasıl karşılayacak.

Requirement → Asset → Control

Gereksinim bloğunun alanları

her gereksinim bu altı alanla yazılır

Kimlik (ID)	CEN429-DR-01 gibi benzersiz
Ölçülebilir ifade	ne yapılacak
Kaynak / gerekçe	hangi tehdit ya da standart
İlgili varlık	neyi koruyor (S5)
Doğrulama yöntemi	hangi test kanıtlayacak (S16)
Durum	karşılandı / devredildi / karşılanmadı

Alanlardan biri boşsa gereksinim izlenebilir değildir.

Every requirement is linked to an **asset** and to a **control**.

The Link to Asset Management

- Requirements are linked to the S5 **asset list**.
- Every asset: C/I/I+ label, protection control.
- The requirement makes that protection **mandatory**.

Carrying It Into the Plan

- Functional requirement → design/code (S6–S11).
- Assurance requirement → test (S16).
- Process requirement → development process (S13).

Worked Example · Applying the Six Steps to CR-03

`CEN429-CR-03` ("keys must be erased once their job is done"):

- **1. Applicability:** there's a session key and a vault-file key → it applies.
- **2. Responsibility:** the code is the project's own → we meet it ourselves.
- **3. Linking to assets:** the vault-file key's "erasure" column is empty — **the gap is noticed here**.

The Same Example · From Threat to Release Plan

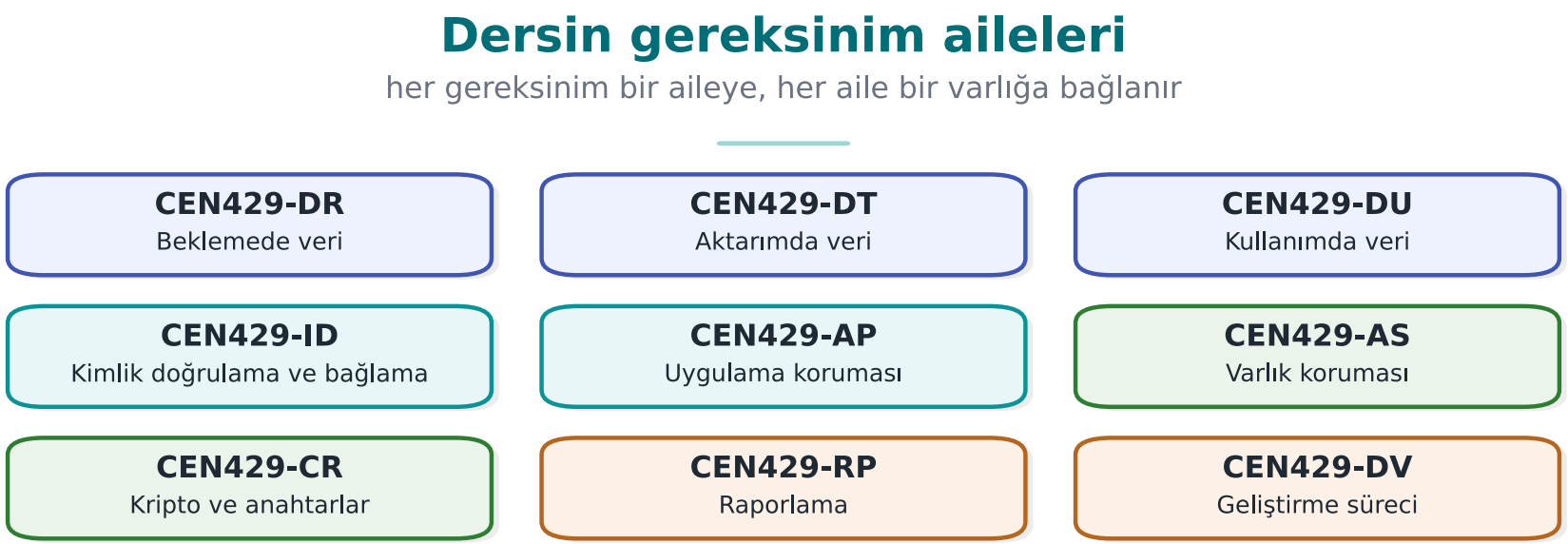
- **4. Linking to threats:** links to the threat "an attacker with physical access takes a memory dump."
- **5. Control/verification:** the `sifreleme_bellek_sil()` call; verification: showing the key's bytes are **not present** in a memory dump.
- **6. Release plan:** the session key was met in v0.9; the vault-file key is planned for v1.0, and until then written as **not met** + residual risk.

Section 7's Rule

- Every "not applicable" decision must be written together with **evidence** supporting its rationale (a code scan, an architecture diagram).
- A "not applicable" with no rationale is just as untrustworthy as a "met" with no evidence.
- Skipping steps 3 and 4 (linking to assets/threats) is the most common mistake.

8. The Course's Requirement Families

Requirement Families — Diagram



Kimlik numarası olmayan gereksinim izlenemez; izlenemeyen gereksinim denetlenemez.

The Course's Own Ids

This course uses its own requirement families, adapted from open standards:

CEN429-**AP** (application protection), **ID** (identity), **AS** (asset), **DR/DU/DT** (data: at rest/in use/in transit)…

Families (1)

Family	Topic
AP	Application protection (obfuscation, RASP)
ID	Identity and binding
AS	Asset management
DR	Data at rest

Families (2)

Family	Topic
DU	Data in use
DT	Data in transit
RP	Reporting/logging
CR	Cryptography
DV	Development/process

Every Family → Links to Weeks

- AP → 4, 5, 6, 9, 11, 14
- CR/DR/DU/DT → 3, 10
- AS → 1, 3
- DV/RP → 12, 13

Your project chooses a **subset** of these families.

Compliance Matrix Activity

Activity · Step 1

- Choose **five requirements** from your own project (from different families).
- Write a **requirement block** for each one.

Activity · Step 2

Fill in a matrix row for every requirement:

Requirement	Status	Section	Verification	Evidence
…	…	…	…	…

Activity · Step 3

- Write at least one as **deferred** (to whom/why/how).
- Show at least one's **evidence** concretely.
- If there's a "not met," write it into the **residual risk**.

Activity · Evaluation

Check through an evaluator's eyes:

- Does every row have evidence?
- Are the deferred ones explicit?
- Are the requirements **verifiable**?

Worked Example · A First Draft (Inadequate)

A team's matrix row for `CEN429-AP-04`, in the **first draft**:

Id	Requirement	Status	Control	Verification	Evidence
CEN429-AP-04	The application must check its integrity at runtime	Met	There's an integrity check	Tested	—

Three problems: (1) the control is a **repetition** of the requirement. (2) "Tested" is **not** a verification method. (3) The evidence is **empty**.

Fixing the Draft · Step by Step

1. Make the control concrete: a hash of the code section is computed at startup and compared with the embedded expected hash at build time; execution stops on mismatch (S12.3).
2. Clarify the verification method: a byte of the binary is changed, and it's verified the application **refuses to start**.
3. Add evidence: the CI job `ci-integrity-check`, run **#391**, `integrity_test.log`.

The Corrected Row

Id	Status	Control	Verification	Evidence
CEN429-AP-04	Met	S12.3: a startup hash is computed, compared with the embedded hash	Tested: startup is refused when a byte changes	CI `ci-integrity-check` #391, `integrity_test.log`

Where Is the Difference?

- The difference isn't in the row's **length**, it's in the **concreteness** of every cell.
- The first draft also looked "filled in," but no cell could be verified **independently**.
- What matters to the evaluator: being able to answer "how do we know this?" for every cell.

Section 8's Rule

- Every row's evidence must be **specific to that row**: a specific test name, a specific CI number, a specific log file.
- Copying a single general report into every row makes the **whole matrix** look suspicious.
- If referencing a general report, state **which section/page** of it too.

Sections 7–8 — Quick Check

1. Which two things is a requirement linked to?
2. Name three of the course's requirement families.
3. What must be in every row of a compliance matrix?

Sections 7–8 — Answers

1. Upward to a **source** (threat/standard/asset) and downward to **evidence** (test/code) — two directions.
2. (Any three of the nine families) e.g. **data security · code hardening · RASP · crypto/certificates · memory protection · interface protection**.
3. The requirement's **id+statement · status** (met/deferred) **· evidence** (test/file/S-section). **A row with no evidence counts as not met.**

9. Project and Closing

Project · S14 and S17

- [ ] **S17 compliance matrix:** status, section, verification, evidence for every applicable requirement.
- [ ] **S14 deferred requirements:** to whom, why, how.
- [ ] If there's at least one "not met," write it into **residual risk**.

Through an Evaluator's Eyes

- A "met" with no evidence = not met.
- Are the deferred ones explicit and justified?
- Are the requirements verifiable?

How to Prepare in 45 Minutes + the Evidence Folder

1. Mark the requirements that apply from the family table; leave the ones that don't as "not applicable, rationale: ..."
2. Fill in the six columns (id, requirement, status, control, verification, evidence); don't write "met" with no evidence.
3. Move the "deferred" rows to S14, with to-whom/why/how.
4. Link your asset table (S5) to every requirement; write which standards you're basing it on into S1.

```
evidence/week13/  <- test/log/pcap dosyaları
  README.md       <- her dosya hangi gereksinimin kanıtı, bir satırla
```

If Time Is Short · Priority Order

1. First, an S17 skeleton with **at least 15 requirements** (id+requirement+status) — far better than an empty S17.
2. Complete the "met" rows you actually **have evidence** for; honestly turn the ones without into "not met."
3. Move the deferred rows to S14.
4. Finally, fill in the remaining control/verification columns.

Rule: a "small but honest" matrix always scores better than a "large but evidence-free" one.

Solved Self-Check

Question 1

What are the four criteria of a good security requirement?

Answer: Verifiable, singular, measurable, feasible.

Question 2

Why is "the application must be secure" bad? How is it fixed?

Answer: It's unverifiable. Write it measurably: e.g., "the release build is produced with a stack canary, PIE, and full RELRO."

Question 3

What is the traceability chain?

Answer: Requirement → control → verification → evidence. Every requirement is linked to evidence.

Question 4

What does a "met" with no evidence mean in a compliance matrix?

Answer: It counts as not met for the evaluator; it's one of the first findings.

Question 5

Which three questions are answered when deferring a requirement?

Answer: To whom, why, how. Silent deferring counts as "not met."

Question 6

What is the difference between ST and PP?

Answer: ST is a single product's security target; PP is a common requirement set for a product class. ST generally builds on a PP.

Question 7

Is "EAL4+ is always more secure than EAL2" correct?

Answer: No. EAL is the depth of the evaluation; security depends on the threats and objectives in the ST. "+" is an additional assurance component.

Question 8

Is an application that uses a FIPS-validated library FIPS-compliant?

Answer: Not on its own. Validation covers the module; the application must use it in approved mode, correctly, and manage keys correctly.

Question 9

Where does ETSI EN 303 645's "securely store sensitive parameters" provision map to in the project?

Answer: S5 asset list, S7 data security/shell matrix, S8 key lifecycle.

Question 10

If a library cannot meet the "secure update" requirement?

Answer: It defers to the parent application and writes to-whom/why/how it will be met in the guide.

Glossary

Term	Meaning
Functional/assurance/process	The three requirement types
Traceability	Requirement→evidence chain
Deferred	A requirement carried over to another party
TOE/ST/PP	CC's basic concepts
SFR/SAR/EAL	Functional/assurance/depth

Summary: This Week in One Sentence

> Security is managed with **written, verifiable, traceable** requirements; every requirement is linked to a
> control, a test, and **evidence**; what can't be met is explicitly **deferred**.

Next Week

Week 14 — Tigress and Diversification

The automated, diversified application of the obfuscation rules; measurement and the build pipeline.

Appendix A · A Worked Compliance Matrix

Context · NotKasa

A synthetic application: keeps local encrypted notes, talks to a server, gets updated.

Let's pick a few requirements and write matrix rows.

Where a Requirement Comes From · The Threat

```
Tehdit: cihaz çalınırsa yerel notlar okunur
   ↓
Amaç: beklemede gizlilik
   ↓
Gereksinim: C sınıfı veri AEAD ile şifrelenir
```

A requirement doesn't come out of thin air; it derives from a **threat**.

Block · DR-01 (Data at Rest)

```
CEN429-DR-01
Metin: Yerel DB'deki C sınıfı veri AES-256-GCM ile şifrelenir.
Durum: Karşılandı
Karşılama: AES-256-GCM, anahtar TEE'de
Doğrulama: T-05
Kanıt: test çıktısı (S16)
```

Block · CR-02 (Crypto)

```
CEN429-CR-02
Metin: Anahtarların amacı, kripto-periyodu, imhası belgelenir.
Durum: Karşılandı
Karşılama: S8 anahtar tablosu
Doğrulama: belge incelemesi
Kanıt: S8
```

Block · DT-03 (Data in Transit)

```
CEN429-DT-03
Metin: Sunucu iletişimi TLS 1.3 ve sertifika zinciri doğrulaması kullanır.
Durum: Karşılandı
Karşılama: TLS 1.3 + SAN denetimi + SPKI pin
Doğrulama: T-11
Kanıt: test + S11
```

Block · AP-04 (Application Protection) — Deferred

```
CEN429-AP-04
Metin: Güvenli kurulum ve güncelleme sağlanır.
Durum: Devredildi
Kime: Üst uygulama (MPA)
Neden: SDK dağıtım kanalına sahip değil
Nasıl: MPA imzalı güncelleme + sürüm denetimi
```

Block · AS-05 — Not Met

```
CEN429-AS-05
Metin: Tüm hassas varlıklar için bellek izleme tespiti.
Durum: Karşılanmadı
Kalan risk: köklü cihazda canlı bellek analizi
Azaltma: kısa ömürlü anahtar + sunucu denetimi
```

Matrix · Combined View

Id	Status	Section	Evidence
DR-01	Met	S8	T-05
CR-02	Met	S8	S8
DT-03	Met	S11	T-11
AP-04	Deferred	S14	—
AS-05	Not met	S12	residual risk

What the Matrix Tells Us

- Three met (with evidence), one deferred, one residual risk.
- Every row is traceable.
- The evaluator starts from this table.

Appendix B · Standard Mapping

The Same Requirement, Multiple Standards

A single requirement can link to more than one standard:

Uyum matrisi: her satır bir kanıt ister

S17'nin iskeleti

Kimlik	Gereksinim	Durum	Önlem	Kanıt
DR-01	Beklemede AEAD	Karşılandı	S7.2	test çıktısı
AP-07	Güvenli güncelleme	Devredildi	S14.2	gerekçe
DT-04	Sertifika sabitleme	Karşılanmadı	—	kalan risk

Kanıt sütunu boş olan 'Karşılandı' satırı, KARŞILANMAMIŞ sayılır.

Why Is Mapping Useful?

- You meet it once and satisfy several standards **at the same time**.
- Add a standard column to the compliance matrix.
- The work isn't repeated even if certification changes.

A Mapping Example

Requirement	ETSI	MASVS	CC
Encrypted storage	✓	STORAGE	SFR
Secure communication	✓	NETWORK	SFR
Secure update	✓	—	SAR

Appendices A–B · Summary

- A requirement derives from a threat, is written as a block, and enters the matrix.
- One requirement can meet many standards.
- No row is "met" without evidence.

Appendix C · Requirement Families in Detail

AP · Application Protection

- **Example:** "Sensitive functions are obfuscated and protected with an integrity check."
- Link: Weeks 4, 6, 9, 11, 14.
- Verification: obfuscation measurement + RASP test.

ID · Identity and Binding

- **Example:** "The device and version are bound to key usage."
- Link: Weeks 6, 11.
- Verification: device-binding test.

AS · Asset Management

- **Example:** "Every asset is labelled C/I/I+ and its lifecycle is documented."
- Link: Weeks 1, 3.
- Verification: S5 asset list.

DR/DU/DT · Data

- **DR (at rest):** "Local data is encrypted with AEAD."
- **DU (in use):** "The key is erased after use."
- **DT (in transit):** "TLS 1.3 + chain validation."

CR · Cryptography

- **Example:** "Approved algorithm, mode, and key length; key hierarchy documented."
- Link: Weeks 3, 10.
- Verification: algorithm inventory.

RP · Reporting

- **Example:** "Sensitive data is never logged in plain text in the release."
- Link: Weeks 4, 6.
- Verification: `strings`, log audit.

DV · Development/Process

- **Example:** "Every change is reviewed before being merged; the SBOM is updated."
- Link: Weeks 5, 12, 13.
- Verification: process records, SBOM.

From Family to Requirement · The Rule

- Your project chooses **at least one** requirement from every family.
- Every requirement derives from a threat, written as a block.
- It enters the matrix, linked to evidence.

Appendix D · Common Mistakes

Mistake · An Unverifiable Requirement

- An unmeasurable phrase like "must be secure."
- Correct: a concrete, testable item.

Mistake · A "Met" With No Evidence

- The evidence column is empty in the matrix.
- The evaluator counts it as not met.

Mistake · Silent Deferring

- Not writing down a requirement that isn't met.
- Correct: to whom/why/how (S14).

Mistake · Mistaking EAL for the Amount of Security

- "EAL4 > EAL2 is more secure" is wrong.
- EAL is depth; security depends on the ST.

Mistake · A FIPS-Validated Library = FIPS Compliance

- Even if the module is validated, the application must use it correctly.
- Approved mode + correct key management.

Mistake · Leaving Residual Risk Blank

- No product has zero residual risk.
- An empty "residual risk" = an incomplete analysis.

Checklist

- [ ] Every requirement is verifiable + singular
- [ ] Every row has status/section/verification/evidence
- [ ] Deferred ones: to whom/why/how
- [ ] Not-met ones → residual risk
- [ ] Standard mapping (ETSI/MASVS/CC)
- [ ] At least one requirement per family

Final Word (Week 13)

> A requirement is born from a threat; it is linked to a control, a test, and **evidence**; if it cannot be met, it
> is explicitly **deferred**. The compliance matrix is the map of this chain.