FPT Education
FPT UNIVERSITY

Session 5
Secure Design and Secure Design Review

Presenter: **Mr. Ngo Tung Son (Ph.D.)**

- Head of Information Assurance Department, FPT University, Hanoi, Vietnam
- Director of RISCS Laboratory, FPT University, Hanoi, Vietnam

FPT Education
FPT UNIVERSITY

Objectives

- Convert Threat Modeling outcomes, especially **STRIDE** findings, into concrete and testable architecture requirements and controls explicitly placed at the correct trust boundaries.
  Chuyển kết quả Threat Modeling, đặc biệt các phát hiện STRIDE, thành yêu cầu và kiểm soát kiến trúc có thể kiểm thử, đặt rõ ràng tại các ranh giới tin cậy phù hợp.
- Document a dedicated **Security** section in the Design Document using the **Claim–Argument–Evidence** structure with end-to-end traceability from requirement to mechanism to evidence.
  Ghi chép một mục Security riêng trong Tài liệu Thiết kế theo cấu trúc Claim–Argument–Evidence, bảo đảm truy vết từ yêu cầu → cơ chế → bằng chứng.
- Define measurable and, where practical, automated verification criteria integrated into the continuous integration and continuous delivery pipeline or enforced via pre-merge checklists.
  Xác định các tiêu chí kiểm chứng đo lường được và, khi khả thi, tự động hoá; tích hợp vào quy trình tích hợp/phân phối liên tục hoặc áp dụng qua checklist trước khi hợp nhất mã.
- Quantify residual risk and state explicit acceptance thresholds aligned with the Bug Bar.
  Lượng hoá rủi ro còn lại và nêu rõ ngưỡng chấp nhận phù hợp với Bug Bar.
- Assign a single accountable owner and a concrete timeline, and define how evidence will be collected, verified, and stored.
  Chỉ định một người chịu trách nhiệm và mốc thời gian cụ thể, đồng thời quy định cách thu thập, kiểm chứng và lưu trữ bằng chứng.
- Prepare verifiable evidence to pass the **Security Design Review**.
  Chuẩn bị bằng chứng có thể kiểm chứng để vượt qua Security Design Review.
- Specify rollback and kill-switch mechanisms and the necessary telemetry to achieve fail-secure behavior and operational observability.
  Xác định cơ chế rollback và kill-switch cùng telemetry cần thiết để đạt hành vi fail-secure và khả năng quan sát khi vận hành.
- Explicitly map every control to its architectural placement and enforcement point to avoid vague goals and ineffective implementations.
  Ánh xạ tường minh từng kiểm soát tới vị trí và điểm thực thi trong kiến trúc nhằm tránh mục tiêu mơ hồ và triển khai kém hiệu quả.
- Apply the above objectives end-to-end to a critical user story to produce a reusable reference pattern for other teams.
  Áp dụng toàn bộ mục tiêu trên theo chuỗi đầu-cuối cho một user-story then chốt để tạo mẫu tham chiếu có thể tái sử dụng cho các nhóm khác.

Secure Design and Secure Design Review

Part 1 – Integrating Security to Design

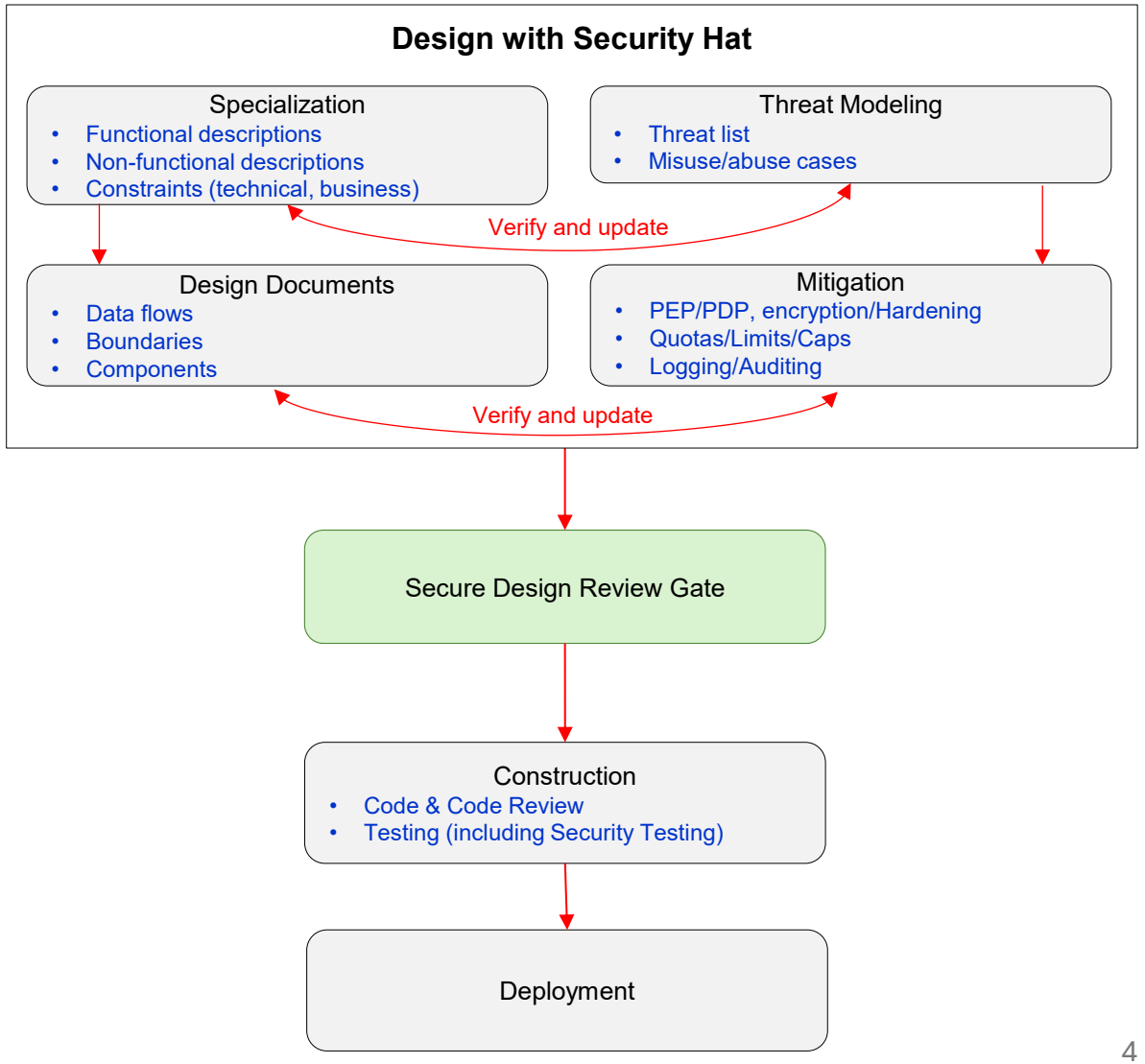
Secure Design in SDLC

KEY CONCEPTS

Linked as One Design System

- **Enforcement Hub:** Policy Enforcement Points (**PEP**) ↔ Policy Decision Points (**PDP**) ensure centralized, consistent enforcement.
  Trung tâm cưỡng chế: PEP ↔ PDP giúp thực thi tập trung, nhất quán.
- **Data-Aware Protections:** Encryption + hardening are selected by **data classification** and placed at the correct **trust boundaries**.
  Bảo vệ theo dữ liệu: mã hoá + gia cố dựa theo phân loại dữ liệu và đặt tại ranh giới tin cậy phù hợp.
- **Policy Controls:** Quotas, limits, and caps constrain abuse and limit blast radius; defined as **policy**, not scattered code checks.
  Kiểm soát chính sách: quota/limit/caps giới hạn lạm dụng và giảm phạm vi ảnh hưởng; được định nghĩa ở chính sách, không rải rác trong mã.
- **Observability:** Logging & auditing with explicit **schemas**, **retention**, and **alerting** make signals testable.
  Khả năng quan sát: logging & auditing có lược đồ, lưu trữ, cảnh báo rõ ràng để kiểm thử được.
- **Traceability Backbone:** Architecture Decision Records (**ADR**) link directly to **tests**, **configuration**, and **log schemas** → establishes end-to-end traceability from requirement → mechanism → evidence.
  Xương sống truy vết: ADR liên kết trực tiếp tới kiểm thử, cấu hình và lược đồ log → bảo đảm truy vết đầu-cuối từ yêu cầu → cơ chế → bằng chứng.

ILLUSTRATION



FPT Education
FPT UNIVERSITY

Making Design Assumptions Explicit

Unwritten assumptions undermine security design reviews. Designers should document them; reviewers should question anything unclear. A suitable place is a **background** section preceding the design body.
*Giả định ngầm làm suy yếu review an ninh. Nhà thiết kế cần ghi chép; người review nên hỏi mọi điểm chưa rõ. Vị trí phù hợp là mục **background** đặt trước phần thân thiết kế.*

KEY CONCEPTS

Why clarification of assumption matters

- Clarification of assumptions is important to security because misunderstandings are often the root cause of a weak interface design or mismatched interaction between components that attackers can exploit.
  Việc làm rõ các giả định rất quan trọng đối với bảo mật vì sự hiểu lầm thường là nguyên nhân gốc rễ của thiết kế giao diện yếu hoặc tương tác không khớp giữa các thành phần mà kẻ tấn công có thể khai thác.
- Explicit assumptions feed Secure Design decisions, bind to verification and fallback, and live in the Design Doc background and shared enterprise assumptions.
  Giả định rõ ràng nuôi dưỡng quyết định Secure Design, gắn với kiểm chứng và phương án hỏng, và được lưu ở phần nền tảng của Design Doc cùng danh mục giả định dùng chung của doanh nghiệp

GUIDE

Common assumptions to document

- Budget, resource, and time constraints that limit the design space.
  Giới hạn ngân sách, nguồn lực, thời gian thu hẹp không gian thiết kế.
- Whether the system is likely to be a target of attack.
  Khả năng hệ thống là mục tiêu tấn công.
- Non-negotiable requirements such as compatibility with legacy systems.
  Yêu cầu bắt buộc, ví dụ tương thích với hệ thống kế thừa.
- Expected security level the system must achieve.
  Mức an ninh kỳ vọng mà hệ thống phải đạt.
- Sensitivity of data and importance of protecting it securely.
  Độ nhạy dữ liệu và tầm quan trọng của bảo vệ an toàn.
- Anticipated needs for future changes to the system.
  Nhu cầu thay đổi trong tương lai.
- Specific performance or efficiency benchmarks to achieve.
  Chuẩn hiệu năng/hiệu quả cụ thể cần đạt.

EXAMPLE

Assumption	Verification	When Wrong / Fallback	Owner	Metric	Evidence
Only internal services call this API.	mTLS between services; firewall allowlist; identity-based policy at PEP/PDP.	Deny non-mTLS traffic; read-only mode; alert and quarantine source.	Platform Security	0% unauthenticated calls in logs; 100% endpoints enforce cert pinning	Policy as code; test reports; log samples; dashboard screenshot
Clients are trustworthy and untampered.	Device attestation; signed requests; server-side validation and throttling.	Restrict capabilities; require re-auth; block high-risk actions.	App Team	100% critical paths validated server-side; 0 critical actions from unverified devices	Negative test suite; WAF logs; auth server metrics
Dependencies meet availability SLOs.	Health checks; circuit breaker; retries with backoff; bulkhead isolation.	Shed load; degrade features; switch to fallback provider.	SRE	Error budget respected; p95 latency below X ms; failover test passes	SLO dashboards; chaos test reports
Endianness and data formats are consistent.	Compile-time checks; contract tests; runtime assertions on message headers.	Convert format on ingress; reject incompatible messages with clear errors.	Platform Core	0 schema/ format mismatches in CI; contract tests green	Contract test artifacts; CI logs
Legacy compatibility is non-negotiable.	Backward-compat suite; canary with legacy clients; feature flags.	Roll back via feature flag; route legacy traffic to compatibility layer.	Release Engineer	0 regressions in legacy canary; recovery < T minutes	Canary dashboards; feature flag history

Scope & Trust Boundaries

PRINCIPLES

Define scope clearly before reviewing security

- It is impossible to do a good security design review if scope is uncertain. Clarifying scope also answers **Four Questions #1: *What are we working on?*** For a billing system, include the new web-based API (an extension of the same design), but treat existing databases as out of scope if they are unchanged and already reviewed.

  Không thể review an ninh tốt nếu phạm vi mơ hồ. Làm rõ phạm vi cũng trả lời Câu hỏi #1: Chúng ta đang làm gì? Với hệ thống thanh toán, nên đưa API web mới vào phạm vi, còn CSDL hiện hữu có thể để ngoài phạm vi nếu không thay đổi và đã được xem xét an ninh trước.

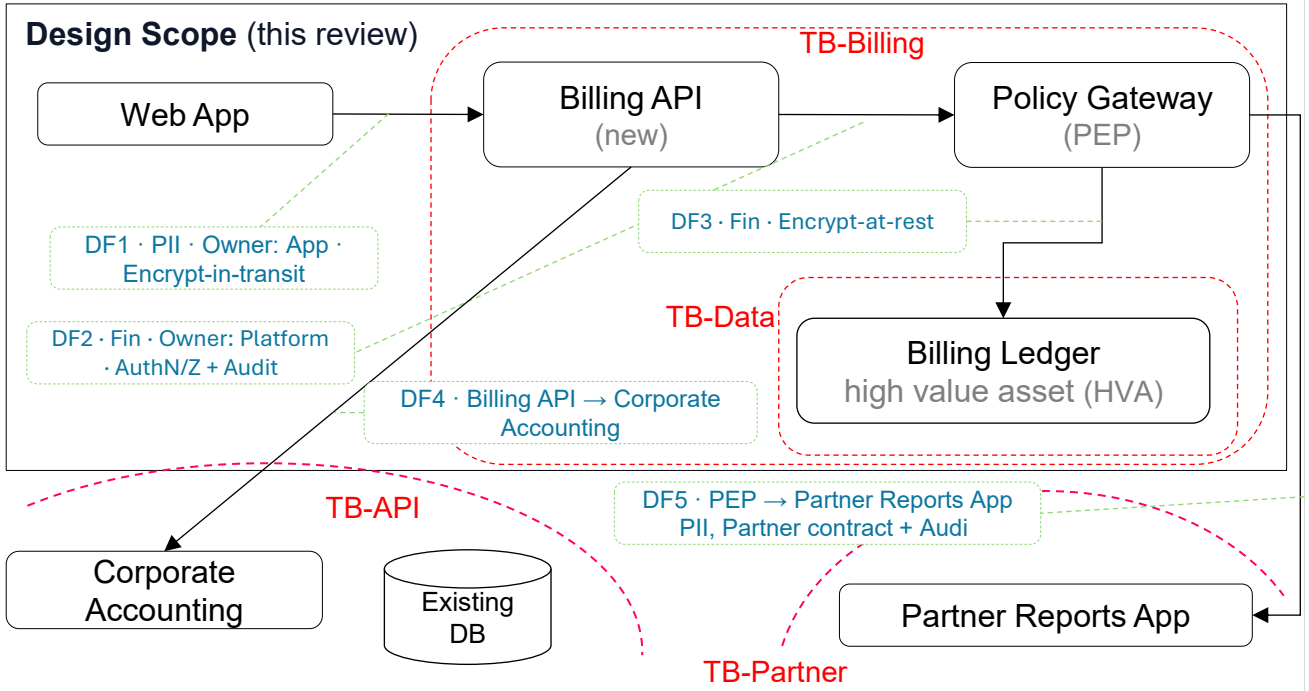
- **Avoid "falling through the cracks":** Vague scope leads to omissions such as assuming backups are handled elsewhere. State scope explicitly to prevent painful lessons after failures.

  Phạm vi mơ hồ dễ bỏ sót như mặc định ai đó lo backup. Hãy nêu rõ phạm vi để tránh bài học đau đớn khi sự cố xảy ra.

- **Iterate and adjust boundaries:** Define a narrow scope that matches redesign work (sprint/major revision). If redesign creeps, adjust the scope. When dependencies are involved, align security expectations; secure/insecure is a continuum—substantiate expectations with peer design review reports.

  Đặt phạm vi hẹp khớp phần tái thiết kế. Nếu phạm vi trượt, hãy điều chỉnh. Với phụ thuộc, cần đồng bộ kỳ vọng an ninh; mức độ an toàn là liên tục—hãy dựa vào báo cáo review của hệ thống tương tự để chứng minh kỳ vọng.

EXAMPLES



Clear in-scope vs out-of-scope areas, vertical trust boundaries (TB-API, TB-Partner), TB-Data around the High-Value Asset (HVA), and labeled dataflows with classification/owner/controls.

Rõ ràng vùng trong/ngoài phạm vi, các ranh giới tin cậy dọc (TB-API, TB-Partner), TB-Data quanh tài sản giá trị cao (HVA), và nhãn luồng dữ liệu kèm phân loại/chủ sở hữu/biện pháp.

Setting (Measurable) Security Requirements

Four Questions #2 + CIA triad
Security requirements largely derive from Four Questions #2: “What can go wrong?”. Use the C-I-A triad as a starting point: protect private data from unauthorized disclosure (confidentiality), secure and back up data (integrity), and define how robust and reliable the system must be (availability). Even when requirements seem obvious, write them down to convey priorities and your intended security stance.
Yêu cầu bảo mật chủ yếu xuất phát từ Câu hỏi #2: “Điều gì có thể sai?”. Bắt đầu với bộ ba C-I-A: ngăn lộ dữ liệu trái phép (bí mật), bảo vệ và sao lưu dữ liệu (toàn vẹn), và xác định mức độ bền vững/độ tin cậy hệ thống (sẵn sàng). Dù có vẻ hiển nhiên, hãy viết rõ để thể hiện ưu tiên và lập trường bảo mật.

WARNING SIGNS

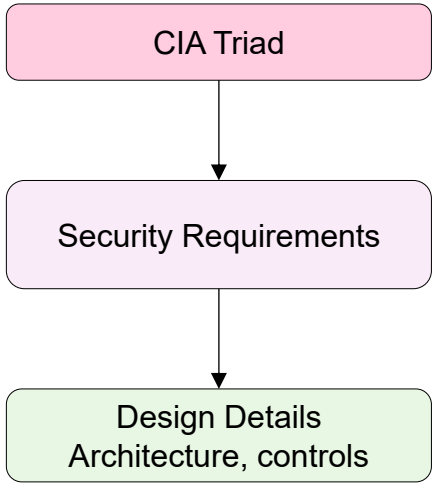
If security seems “low” — say it, bound it

- Prototypes with dummy data: you may skip a full review, but clearly mark the code so it isn’t repurposed with personal data.
  Prototype dùng dữ liệu giả: có thể bỏ qua review đầy đủ, nhưng đánh dấu rõ để tránh tái sử dụng với dữ liệu cá nhân.
- Benign datasets: e.g., shared weather readings — disclosure is harmless; still record this assumption and its limits.
  Bộ dữ liệu ít rủi ro: ví dụ số liệu thời tiết — công khai không gây hại; vẫn cần ghi nhận giả định và giới hạn.

CRITICAL

General Guidelines

- Express requirements as end goals without dictating “how to”.
  Diễn đạt yêu cầu như mục tiêu cuối, không áp đặt cách làm.
- Consider all stakeholder needs and balance conflicts.
  Cân bằng nhu cầu các bên liên quan khi có xung đột.
- Acknowledge acceptable costs and trade-offs for critical mitigations.
  Thừa nhận chi phí/đánh đổi chấp nhận được.
- Explain the motivation behind unusual goals.
  Giải thích động cơ của các yêu cầu khác thường.
- Set achievable goals, not mandates for perfection.
  Đặt mục tiêu khả thi, không đòi hỏi hoàn hảo tuyệt đối.



EXTREME EXAMPLES

Insider Resistant Operations

- privileged admins could impersonate officers and mass-exfiltrate; mitigate with least privilege, duty separation, immutable logs.
  admin đặc quyền → giả mạo sĩ quan, exfil hàng loạt; giảm: least-privilege, tách nhiệm vụ, log bất biến.

Bank Authentication Server

- Bank auth server: private key leak = total collapse; use HSM, key ceremonies, two-person rule.
  lộ private key = sụp đổ; dùng HSM + lễ tạo/luân chuyển khóa + 2-person rule.

One-shot scientific integrity

- one-shot data → instant multi-copy replication, dual networks to remote site, integrity checks.
  dữ liệu một lần → nhân bản tức thời đa-media, 2 mạng về site xa, kiểm tra toàn vẹn.

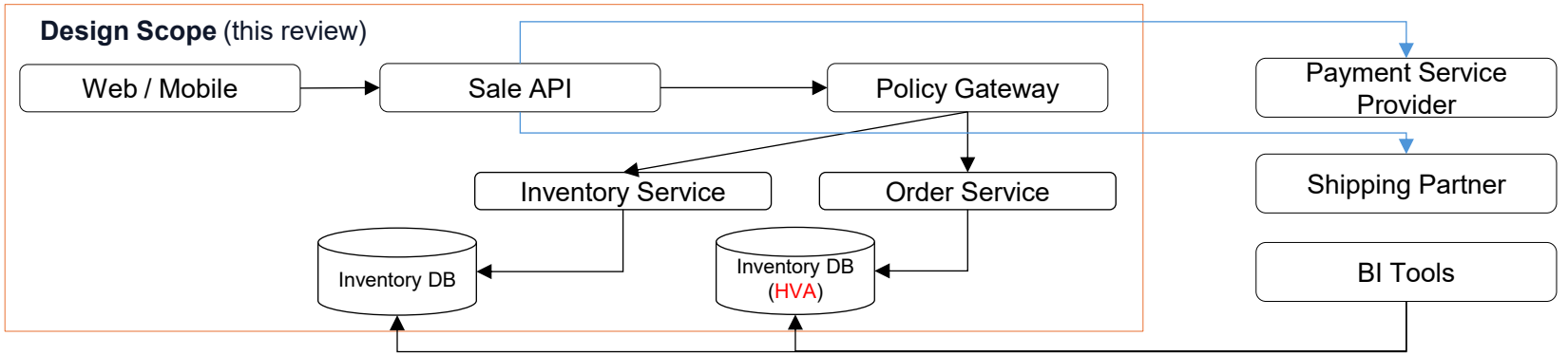
Examples of Security Requirements

EXAMPLE – SALE SYSTEM

CIA/AAA → measurable goals

- Sales systems process PII, payment tokens, and financial records. Translate CIA/AAA into measurable requirements

Hệ thống bán hàng xử lý PII, token thanh toán và dữ liệu tài chính.



EXAMPLE OF SECURITY REQUIREMENTS

- Traffic confidentiality (Conf): All customer-facing traffic is encrypted in transit and resistant to downgrade; coverage = 100%.
  Bí mật lưu lượng: Mọi lưu lượng đối diện khách được mã hoá khi truyền và chống hạ cấp; độ bao phủ = 100%.
- Cardholder data scope (Conf/Compliance): PAN is never processed, stored, or logged by the system; occurrences = 0 across code/configs/logs.
  Phạm vi dữ liệu thẻ: Hệ thống không xử lý, lưu trữ hay ghi log PAN; số lần xuất hiện = 0 trên mã/cấu hình/log.
- PII at-rest protection (Conf): All stored PII is protected at rest; coverage = 100%; key lifecycle meets defined rotation policy.
  Bảo vệ PII khi lưu trữ: PII được bảo vệ khi lưu; độ bao phủ = 100%; vòng đời khoá tuân thủ chính sách xoay định nghĩa.
- Order history integrity (Integ): Order events are tamper-evident and non-repudiable; daily integrity verification reports 0 mismatches; alert MTTD ≤ 5 min.
  Toàn vẹn lịch sử đơn hàng: Dấu vết chống sửa và không thể chối bỏ; kiểm chứng hằng ngày lệch = 0; thời gian phát hiện ≤ 5 phút.
- Service availability (Avail): Checkout SLO ≥ 99.9%; p95 latency ≤ 800 ms under normal and abusive conditions.
  Sẵn sàng dịch vụ: SLO thanh toán ≥ 99,9%; p95 ≤ 800 ms trong điều kiện bình thường và có lạm dụng.
- Administrative access assurance (AuthN/AuthZ): Sensitive admin actions require strong identity assurance and time-bound elevation; high-value actions require dual control above defined thresholds.
  Đảm bảo truy cập quản trị: Hành động nhạy cảm cần xác minh danh tính mạnh và nâng quyền theo thời gian; hành động giá trị cao cần kiểm soát hai người trên ngưỡng định nghĩa.
- Auditing & traceability (Audit): 100% admin operations and payment callbacks are recorded; audit trail is tamper-evident and time-ordered; sensitive fields are redacted.
  Ghi vết & kiểm toán: 100% thao tác admin và callback thanh toán được ghi; nhật ký chống sửa và có thứ tự thời gian; trường nhạy cảm được che.
- Partner integration trust (Conf/AuthZ): Partner API interactions are mutually authenticated, least-privilege scoped, and fully auditable.
  Tin cậy tích hợp đối tác: Tương tác API với đối tác được xác thực hai phía, phạm vi tối thiểu đặc quyền và ghi vết đầy đủ.
- Fraud & abuse containment (AuthZ/Avail): Abuse impact is bounded by per-entity limits and risk-driven friction; false-positive rate ≤ target.
  Giới hạn gian lận/lạm dụng: Tác động được ràng buộc bởi hạn mức theo thực thể và ma sát theo rủi ro; tỷ lệ dương tính giả ≤ mục tiêu.

Thread Modeling in Design

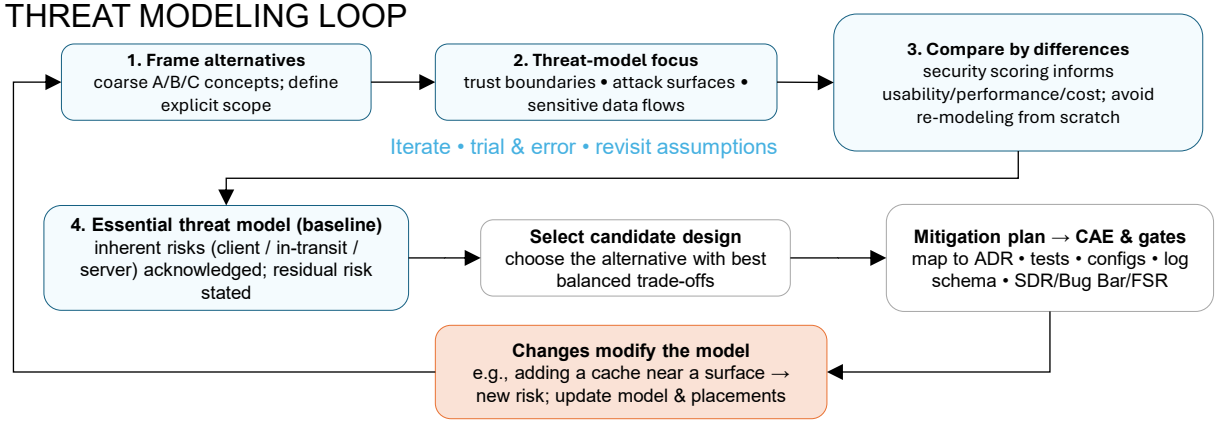
Embed threat modeling into design: it clarifies trade-offs and guides iterations toward a safer architecture—even with some trial-and-error.
Nhúng threat modeling vào thiết kế: nó soi rõ các đánh đổi và dẫn dắt các vòng lặp tới kiến trúc an toàn hơn—dù vẫn cần thử-sai để tối ưu.

METHOD

Simplistic vs Practical Integration

- Simplistic: generate multiple designs → threat-model each → score via summary assessment → pick the "best". Useful in theory but prohibitive in effort.
  Giản đơn: sinh nhiều phương án → mô hình hoá đe doạ từng phương án → chấm điểm → chọn. Hữu ích về lý thuyết nhưng tốn công sức.
- Practical: let security-focused assessments *inform* usability, performance, and cost; compare alternatives by their differences instead of re-modeling each from scratch.
  Thực tiễn: để đánh giá an ninh ảnh hưởng tới khả dụng, hiệu năng, chi phí; so sánh khác biệt giữa các phương án thay vì mô hình lại từ đầu.

THREAT MODELING LOOP



EARLY-STAGE FOCUS

Boundaries, surfaces, and sensitive flows

- Keep sensitive data away from exposed topologies where possible. Favor designs that reduce exposure along trust boundaries and minimize attack surfaces. Ex: HIGH - highest exposure of PII, MEDIUM - reduced offline PII, updated by region/schedule, LOWER - phone numbers not stored on device
  Giữ dữ liệu nhạy cảm tránh xa vùng phơi nhiễm khi có thể. Ưu tiên thiết kế giảm phơi nhiễm theo ranh giới tin cậy và tối thiểu hoá bề mặt tấn công.

What should do

- Inherent risks: client exposure, in-transit interception, server-side compromise cannot be wished away; plan mitigations and state residual risk.
  Rủi ro nội tại: phơi nhiễm phía client, trên đường truyền, phía server không thể biến mất; hãy lập kế hoạch giảm thiểu và nêu rõ rủi ro còn lại.
- When risk is high: reconsider the *need to collect* sensitive data; support sub-cases that avoid collection at the source where feasible.
  Khi rủi ro cao: cân nhắc nhu cầu thu thập dữ liệu nhạy cảm; hỗ trợ biến thể use-case tránh thu thập ngay từ đầu khi khả thi.

STRATEGIES

When compromise is needed

- Design for flexibility so protections can be added later; avoid painting into an insecure corner.
  Thiết kế linh hoạt để có thể bổ sung bảo vệ về sau; tránh bị "kẹt" trong góc không an toàn.
- Instrument to observe attempts of abuse if specific attacks are of concern.
  Gắn đo đạc/giám sát để quan sát hành vi lạm dụng nếu lo ngại tấn công cụ thể.
- When usability conflicts arise, explore UI alternatives and measure under realistic conditions.
  Khi xung đột khả dụng, thử phương án UI khác và đo đạc trong điều kiện thực tế.
- Explain risks using scenarios derived from threat models to illustrate downside of not mitigating.
  Diễn giải rủi ro qua kịch bản từ threat model để thấy cái giá của việc không giảm thiểu.

Balancing factors — security isn't everything

Good design depends on subjective judgment across security, usability, performance, and cost. Stay open to compromise: maximize security when costs are low; otherwise, negotiate trade-offs with clear scenarios and evidence from the threat model.

Thiết kế tốt dựa trên đánh giá chủ quan giữa bảo mật, khả dụng, hiệu năng và chi phí. Hãy sẵn sàng thoả hiệp: tối đa hoá bảo mật khi chi phí thấp; nếu không, thương lượng đánh đổi với kịch bản và bằng chứng xuất phát từ threat model.

Secure Design and Secure Design Review

Part 2 – Building in Mitigation

Building in Mitigation

Four Questions #3: "What are we going to do about it?"

- After scope and security requirements are clear, the designer incorporates protections and mitigations into the architecture. That focuses on two recurring Interfaces and Data

Khi phạm vi và yêu cầu bảo mật đã rõ, người thiết kế tích hợp các cơ chế bảo vệ vào kiến trúc. Cái tập trung vào 2 chủ đề interfaces và data.

DESIGN PATTERNS – TO BUILD

Design Attributes
- Economy of Design
- Transparent Design

Exposure Minimization
- Least Privilege
- Least Information
- Secure by Default
- Allowlists over Blocklists
- Avoid Predictability
- Fail Securely

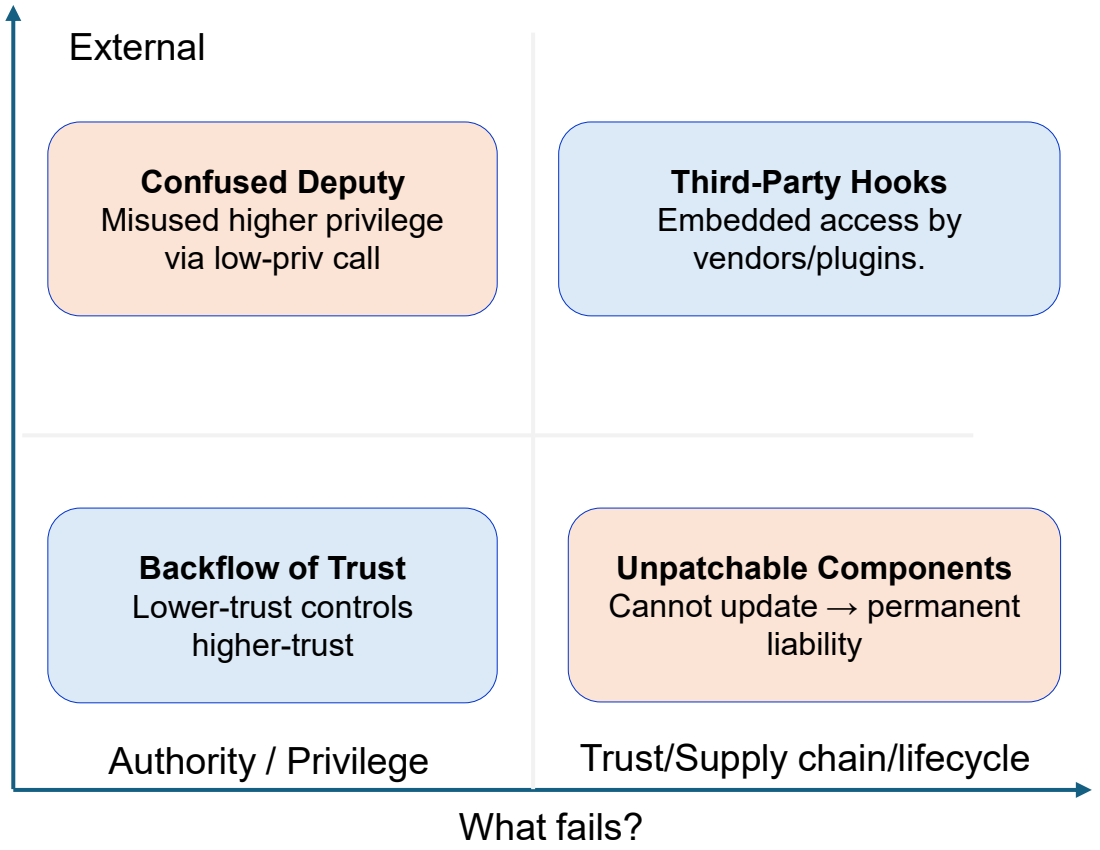
Trust and Responsibility
- Reluctance of Trust
- Accept Security Responsibility

Groups

Redundancy
- Defense in Depth
- Separation of Privilege

Strong Enforcement
- Complete Mediation
- Least Common Mechanism

ANTI PATTERNS – TO AVOID



Designing Interfaces – Safe Endpoints

CONCEPT

Interfaces define boundaries

Interfaces delineate system limits and component responsibilities: system calls, libraries, networks (client/server or P2P), inter- and intra-process APIs, shared structures in common datastores, and more. Complex interfaces (e.g., secure protocols) may warrant their own design. Within scope, define every interface, clarify security duties between peers, note whether inputs are validated or must be treated as untrusted, and, where a trust boundary is crossed, explain authentication and authorization.

Giao diện phác ranh giới hệ thống và trách nhiệm bảo mật của các thành phần chia sẻ cấu trúc dữ liệu dùng chung, v.v. Giao diện phức tạp có thể cần thiết kế riêng. Trong phạm vi, định nghĩa mọi giao diện, làm rõ trách nhiệm bảo mật, ghi chú đầu vào đã/ chưa được kiểm tra, và nếu có ranh giới tin cậy thì mô tả cơ chế xác thực/ủy quyền khi băng qua.

Security properties of interfaces

- State the necessary properties (CIA, privacy, Gold Standard).
  thuộc tính cần thiết: CIA, riêng tư, "chuẩn vàng".
- Security review verifies correct function & robustness vs threats.
  Review bảo mật xác minh hoạt động đúng & vững trước đe doạ.
- If requirements are unclear, reviewers & future devs will guess.
  Nếu yêu cầu mơ hồ, reviewer và dev sẽ phải đoán.

Endpoint as the front door

- Clear contract; validate inputs at the gate; default-deny errors; standardized responses to avoid leakage.
  Làm rõ Hợp đồng; kiểm tra ở "điểm gác"; lỗi mặc định từ chối; phản hồi chuẩn hoá để tránh rò rỉ.
- Admin paths isolated; require MFA & JIT elevation; only from verified networks/zones.
  Đường quản trị tách biệt; bắt buộc MFA & nâng quyền đúng lúc; chỉ từ vùng mạng đã kiểm chứng.
- Audit side-effects with 5W1H.
  Ghi audit hành động có tác dụng phụ bằng 5W1H.

Catalog Interfaces in Scope

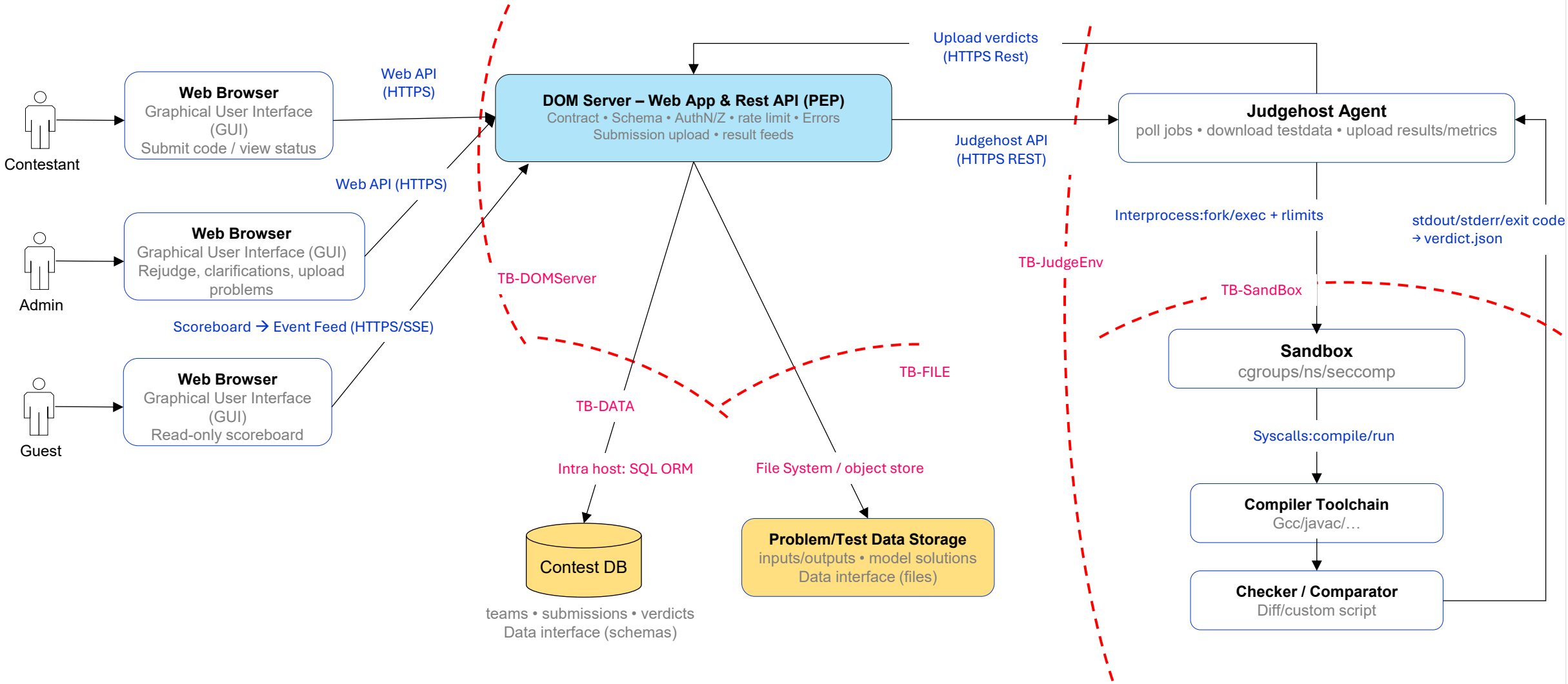
- System calls, libraries, IPC, RPC, HTTP/gRPC, P2P, datastores/shared structures.
  System call, thư viện, IPC, RPC, HTTP/gRPC, P2P, datastore/cấu trúc chia sẻ.
- Document each interface's security responsibilities and whether inputs are pre-validated.
  Ghi trách nhiệm bảo mật và tình trạng kiểm tra đầu vào.
- Where a trust boundary is crossed, describe authn/authz.
  Khi băng qua ranh giới tin cậy, mô tả xác thực/ủy quyền.

External components & assumptions

- Conform to existing specs of external systems; if unknown, document assumptions or add defensive wrappers.
  Tuân thủ đặc tả; nếu thiếu, ghi giả định hoặc bọc phòng vệ.
- When inputs may be unvalidated, treat as untrusted by default.
  Nếu không chắc đã kiểm tra, mặc định coi là không tin cậy.
- Optionally attack a test mock-up to learn security properties.
  Có thể tấn công bản thử để hiểu thuộc tính an toàn.

Example: DOMJugde System - Interface and Boundaries

ILLUSTRATION OF DOMJUGDE INTERFACES AND BOUNDARIES



DOMJugde - Interface Catalog & Recommended Design (1)

INTERFACE CATALOG AND RECOMMENDED DESIGN

Interface	Caller	Callee	Data & Protocol	Key Controls	Notes/Endpoints
I1. Web API (HTTPS) — GUI ↔ DOMserver (TB-WebAPI)	Contestant GUI; Jury GUI; Admin GUI	DOMserver Web App & REST API	JSON over HTTPS for most endpoints; multipart/form-data (ZIP or plain source) for submission upload; pagination via opaque cursors; idempotency keys for POST-like actions; errors in RFC7807 format.	AuthN: session cookie (SameSite=strict, HttpOnly, Secure) or OAuth2/OIDC with short-lived tokens + PKCE. AuthZ: role/contest scope per endpoint; allowlist of languages/problems per contest. Input caps: max source size, files count, compile time, memory. Rate limits per principal + IP; CSRF tokens for browser POST; content-disposition filename canonicalization; server-side MIME/type sniffing disabled.	Submission: POST /api/submissions (multipart) → returns {submission_id, sha256}. Status: GET /api/submissions/{id}. Rejudge/clarification/admin ops: POST with idempotency-key header. All responses include ETag; clients use If-None-Match.
I2. Event Feed — Scoreboard → Browser (TB-EventFeed)	DOMServer	Read-only browsers / public displays	Server-Sent Events (SSE) over HTTPS with Last-Event-ID; alternatively, WebSocket in read-only mode. Payloads: minimal JSON deltas.	Public/role-segmented channels (public vs jury). Do not embed personal data; redact IPs/PII. Cache-control: no-store for private; CDN for public. Anti-replay with event sequence + expiry; per-client rate caps.	Endpoint: GET /api/events/stream?contest_id=… with scope=public|jury. Fields: event_id, ts, type, payload.
I3. Judgehost API (HTTPS mTLS) — DOMserver ↔ Judgehost Agent (TB-Judgehost)	Judgehost Agent (poll jobs; download inputs; upload results/metrics)	DOMserver Judgehost controller	REST/JSON over HTTPS with **mutual TLS (mTLS)**. Agents authenticate with device certificates issued by the contest CA; DOMserver validates client cert CN/SAN and cert status (CRL/OCSP).	Short-lived, least-privilege credentials per host. Job descriptors carry pre-signed URLs (time- and scope-limited) to fetch artifacts. All artifacts are content-addressed (sha256) and include size checks. Results upload must include attestation: toolchain versions, checker hash, sandbox profile, resource usage. Replay-proof via one-time job token bound to mTLS identity.	Poll: POST /judgehost/poll → {job_id, inputs:[…], limits,…}. Download: GET pre-signed /artifacts/{hash}. Upload: POST /judgehost/verdict {submission_id, verdict.json, metrics, attestations}.
I4. Data Interface — DOMserver ↔ Contest DB (TB-Data)	DOMserver ORM layer	RDBMS (Contest DB)	SQL via ORM with parameterized queries; migrations versioned; read/write separation (primary) and read replicas for scoreboard.	DB roles: app-write (limited), app-read (scoreboard), admin (offline only). Row-level access per contest. All changes write to tamper-evident audit tables with actor+reason. Backups encrypted; tested RTO/RPO.	Schemas include teams, submissions, verdicts, problems, clarifications, events.

DOMJugde - Interface Catalog & Recommended Design (2)

INTERFACE CATALOG AND RECOMMENDED DESIGN

I5. Artifact Store — DOMserver ↔ File/Object Store (TB-File)	DOMserver	Object store / POSIX FS	Upload/download via content-addressed paths: /objects/sha256/<hash>; integrity verified on read. Problem/test bundles and submission archives are immutable objects; metadata kept in DB.	Bucket/prefix policies: write by DOMserver only; judgehosts read via pre-signed URLs. Lifecycle: retention period & secure deletion. Optional server-side encryption (AES-GCM) + KMS.	Objects: submission.zip, testcase.tar, checker.bin, verdict attachments.
I6. Inter-process API — Judgehost → Sandbox (TB-Sandbox)	Judge process	Sandbox runtime (cgroups/ns/ seccomp)	Local exec via fork/exec; IPC through pipes; resource limits configured per job.	Unprivileged user; no root. Namespace isolation; seccomp allowlist; cgroup CPU/mem/pids caps; no network or outbound only to metadata proxy off by default. Read-only mounts for toolchains/testdata; tmpfs for workdir; deterministic locale/timezone.	Inputs: source/tests; Outputs: stdout/stderr/exit code; usage stats.
I7. Checker/Comparator Contract	Sandboxed executed program (checker)	Judge harness	Std I/O contract: read contestant output and expected output; emit verdict code and metrics JSON to stdout/stderr.	Checker is treated as untrusted like submissions. Signed provenance for official checkers; fixed time/memory limits; no filesystem/network access beyond workdir.	Outputs a compact verdict.json {status, score, messages, time_ms, mem_kb}.

DOMJugde - Security Design Patterns

PATTERNS MATCHING

Design Patterns To Build

- Central Guard (PEP) at DOMServer: **complete mediation** of all contest actions.
- **Least Privilege** everywhere: separate roles for public scoreboard, contestant, jury, admin, judgehost.
- **Mutual Authentication** on sensitive machine interfaces (mTLS for I3).
- **Content-addressed, immutable artifacts (I5) + hash verification** → integrity & replay resistance.
- **Defense-in-Depth**: browser → server → job dispatch → sandbox → checker each re-validates assumptions.
- **Fail Securely**: job tokens expire; transient failures do not flip to permissive states; retries are idempotent.
- **Rate/Quota Limits**: per team, per IP, per contest to bound DoS/abuse blast radius.
- **Secure Logging & Accountability**: actor-tied audit logs; correlation IDs across I1–I8.
- **Data Minimization & Redaction** on Event Feed (I2).
- **Explicit Trust Boundaries** (TB-WebAPI, TB-EventFeed, TB-Judgehost, TB-Data, TB-File, TB-Sandbox) labeled in docs/DFDs.

Anti-Patterns To Avoid

- Trusting judgehosts based only on IP or shared long-lived tokens (no mTLS).
- **Embedding sensitive fields** (usernames/emails/IPs) in public scoreboard events.
- Accepting client-supplied filenames/paths without canonicalization → **path traversal** in artifact store.
- Running judgehost or checker with elevated privileges; shared writable mounts into the host OS.
- Unbounded submission size or job concurrency → self-inflicted DoS.
- **Single shared DB** role for all components; **lack of read-only separation** for scoreboard.
- **Silent overwrites** of artifacts instead of immutable content-addressed storage.
- **Webhooks/callbacks from third parties** into privileged endpoints without strict allowlists/signatures.
- Absence of idempotency keys on POST actions like rejudge → **duplicate state transitions**.

FPT Education
FPT UNIVERSITY

Threat List – STRIDE by Interface

THREAT LIST – CLASSIFIED BY STRIDE MODEL

Interface	Spoofing	Tampering	Repudiation	Information Disclosure	DoS	EoP
I1 Web API	stolen session/JWT; CSRF	modified uploads; param tampering	missing audit of admin ops	error leaks; IDOR	submission floods; heavy queries	privilege escalation via confused deputy
I2 Event Feed	rogue client impersonates jury channel	forged events if channel not signed	no event audit trail	leaking hidden tests/clarifications	clients open many SSE connections	using feed to trigger admin actions (should be read-only)
I3 Judgehost API	fake judgehost polls jobs	verdict.json altered in transit or replayed	unverifiable host identity/results	test data via over-broad URLs	many bogus hosts poll; slowloris	agent RCE path compromises DOMserver
I4 Contest DB	shared DB creds	direct writes bypassing ORM	missing change logs	broad SELECTs by scoreboard	expensive joins from UI	SQLi → superuser
I5 Artifact Store	attacker uploads lookalike object	replace test bundles without hash pinning	no provenance of checkers	wrong ACLs expose submissions/tests	storing huge blobs; path explosion	symlink/zip-slip on extract
I6 Sandbox	process identities inside sandbox reused	escape to modify host files	lack of job-scoped attestation	read host secrets via mounts	fork bombs; resource starvation	kernel escape/CVE in container runtime
I7 Checker Contract	malicious checker binary swapped	forged scores	no signed outputs	checker prints sensitive inputs	pathological checker complexity	checker exploits runtime to break out

DOMJugde – Misuse/Abuse Cases

REPRESENTATIVE MISUSE CASES

- As a contestant, I submit a fork-bomb or massive stdout to exhaust judgehost memory/disk and delay others (availability abuse).
  Với tư cách là người tham gia, tôi gửi một fork-bomb hoặc stdout khổng lồ để làm cạn kiệt bộ nhớ/đĩa của judgehost và làm chậm các bộ nhớ/đĩa khác (lạm dụng tính khả dụng).
- As a contestant, I try to infer hidden test data via timing side-channels or verbose compiler errors (information disclosure via side-channel).
  Với tư cách là người tham gia, tôi cố gắng suy ra dữ liệu thử nghiệm ẩn thông qua các kênh phụ thời gian hoặc lỗi biên dịch dài dòng (tiết lộ thông tin qua kênh phụ).
- As a malicious actor, I spin up fake judgehosts to poll jobs and exfiltrate test bundles (spoofing judgehosts).
  Với tư cách là một tác nhân độc hại, tôi tạo ra các judgehost giả để thăm dò các công việc và trích xuất các gói thử nghiệm (giả mạo judgehost).
- As a rogue insider, I swap a checker binary to accept my team's output regardless of correctness (tampering with checker).
  Với tư cách là một người trong cuộc, tôi hoán đổi một mã nhị phân của trình kiểm tra để chấp nhận đầu ra của nhóm mình bất kể tính chính xác (giả mạo trình kiểm tra).
- As a competitor, I spam clarifications or rejudge requests to create operational DoS (process abuse).
  Với tư cách là một đối thủ cạnh tranh, tôi spam các giải thích hoặc đánh giá lại các yêu cầu để tạo ra DoS hoạt động (lạm dụng quy trình).
- As a script, I scrape SSE and flood scoreboard subscriptions to overwhelm the server/CDN (event feed DoS).
  Dưới dạng một tập lệnh, tôi thu thập SSE và làm ngập các đăng ký bảng điểm để làm quá tải máy chủ/CDN (DoS nguồn cấp sự kiện).
- As an attacker, I craft a zip with '../' paths to overwrite server files on extraction (zip-slip).
  Với tư cách là kẻ tấn công, tôi tạo một tệp zip có đường dẫn '../' để ghi đè lên các tệp máy chủ khi giải nén (zip-slip).
- As a network adversary, I replay a previously captured verdict upload to alter scores (replay without nonce binding).
  Với tư cách là đối thủ mạng, tôi phát lại bản án đã tải lên trước đó để thay đổi điểm số (phát lại mà không ràng buộc nonce).

DOMJugde – Control Checklist, Data Schema and Residual Risks

Control Checklist by Interfaces

Interface	Controls (quick checklist)
I1 Web API	TLS 1.3; session cookie security; CSRF token; input caps; language allowlist; size/time quotas; rate limits; RFC7807 errors; ETag caching; audit all admin ops.
I2 Event Feed	Read-only channel separation; no secrets in payload; per-client caps; Last-Event-ID; CDN for public; signed event source where applicable.
I3 Judgehost API	mTLS; one-time job tokens; content-addressed artifacts; short-lived pre-signed URLs; attestation of toolchain/sandbox; replay protection; audit lifecycle of hosts.
I4 Contest DB	Least-privilege DB roles; parameterized queries; RLS per contest; append-only audit; encrypted backups; tested restores (RTO/RPO).
I5 Artifact Store	Immutable objects; SHA-256 pinning; server-only writes; lifecycle policies; SSE-KMS encryption; denial-by-default bucket ACLs.
I6 Sandbox	Unprivileged user; seccomp allowlist; no network; read-only mounts; cgroup CPU/mem/pids; time/mem limits; ephemeral working dir; kill-switch on runaway jobs.
I7 Checker Contract	Signed provenance; fixed limits; no FS/network; write verdict.json only; verify versions before run.

Concise data schema

Submission Upload (multipart form)

Fields: team_id, contest_id, problem_id, language, idempotency_key
Files: source.zip (sha256), optional metadata.json
Response: { "submission_id": "...", "received_at": "...", "sha256": "..." }

Job Descriptor (poll response)

{ "job_id":"...", "submission_id":"...",
"inputs":[{"name":"tests.tar","sha256":"...","url":"..."}],
"limits":{"time_ms":2000,"mem_kb":262144,"proc":1},
"token":"one-time-bound-to-mTLS" }

Verdict Upload

{ "submission_id":"...", "status":"AC|WA|TLE|MLE|RE|CE", "score":100,
"cases":[{"id":"tc1","status":"AC","time_ms":123,"mem_kb":10240}],
"env":{"toolchain":"gcc-13.2","checker_sha256":"...","sandbox_profile":"seccomp:v1"},
"artifact_hashes":{"stdout":"...","stderr":"..."},
"nonce":"...", "sig":"judgehost-sig" }

Residual Risks & Monitoring

- Sandbox escapes (kernel/runtime CVEs) — keep host minimal & patched; consider Kata/VM-based isolation for untrusted languages.
- Supply-chain risk in checkers/toolchains — sign, pin, and attest versions; reproducible builds.
- Side-channels (timing/size) — normalize outputs, cap logs, randomize scheduling where feasible.
- Operational overload during peak contests — auto scale judgehosts; queue backpressure; graceful degradation of scoreboard.

Design Data Handling

CONCEPT

Handle data by purpose, class, and lifecycle

Start by stating data protection goals. Treat personal data as sensitive unless explicitly decided otherwise. Reduce data movement and exposure by using opaque identifiers (Least Information pattern), and log using handles instead of raw data. Recognize public-data exceptions, but remember context can turn public facts into sensitive information. Plan for retention limits and secure deletion, including backups.

Bắt đầu bằng cách nêu rõ mục tiêu bảo vệ dữ liệu. Mặc định coi dữ liệu cá nhân là nhạy cảm trừ khi có quyết định khác. Giảm dịch chuyển/phơi nhiễm bằng mã định danh mờ (opaque ID) (mẫu Least Information), ghi log theo mã định danh thay vì dữ liệu thô. Thừa nhận ngoại lệ dữ liệu công khai, nhưng bối cảnh có thể biến công khai thành nhạy cảm. Lập kế hoạch lưu giữ có hạn và xoá an toàn kể cả bản sao lưu.

End-goals & Design Patterns

- Data minimization: collect only with specific purpose; avoid indefinite storage.
  chỉ thu thập khi có mục đích; tránh lưu vô thời hạn.
- Opaque identifiers: pass tokens across interfaces; dereference only when needed.
  Mã định danh mờ: truyền token qua giao diện; chỉ truy xuất khi cần.
- Classify & protect: tailor controls by sensitivity; default-deny for personal data.
  Phân loại & bảo vệ: điều chỉnh kiểm soát theo độ nhạy; mặc định từ chối với dữ liệu cá nhân.

Logs, Access, & Audit

- Log handles (tokens) and metadata (5W1H), not raw PII.
  Log xử lý (tokens) và metadata, không ghi PII thô.
- Separate auditor workflows to resolve tokens; enforce dual control.
  Tách luồng công việc kiểm toán để giải mã token; áp dụng kiểm soát hai người.
- Tamper-evident storage and retention for logs.
  Lưu trữ chống sửa và chính sách lưu giữ cho log.

Retention & Deletion Checklist

- State explicit retention periods per class; justify exceptions.
  Nêu thời hạn lưu giữ theo lớp; biện minh ngoại lệ.
- Define deletion triggers and backup propagation.
  Xác định điều kiện xoá và lan truyền tới bản sao lưu.
- Specify how to remove data from search indexes and caches.
  Quy định cách xoá khỏi chỉ mục tìm kiếm và cache.

Avoid these traps — anti-patterns

- Collecting personal data “just in case”.
  Thu thập dữ liệu cá nhân “phòng khi cần”.
- Logs containing raw identifiers (e.g., full credit card numbers).
  Log chứa định danh thô như số thẻ.
- Indefinite retention or unclear deletion of backups/replicas.
  Lưu giữ vô hạn hoặc xoá sao lưu không rõ ràng.
- Assuming public data is always safe when paired with context.
  Mặc định dữ liệu công khai luôn an toàn khi ghép bối cảnh.

Secure Data Handling — Retail Sales System (Case Study)

SCENARIO

System Context and Scope

- Scenario: an online retail sales system with Web/Mobile UI, a Web API (PEP), Order, Payment, and Customer services, a Data Vault (tokenization), and a Reporting subsystem.
  Hệ thống bán lẻ trực tuyến có Web/Mobile UI, Web API (điểm gác PEP), các dịch vụ Order/Payment/Customer, một Data Vault (mã định danh mờ), và hệ thống Báo cáo.
- In-scope: data handling for customers, orders, payments, fulfillment events.
- Out-of-scope: third-party payment processor internals (treated as dependency with contract).
- Trust boundaries: TB-WebAPI, TB-Data (DB/Vault), TB-File (object store), TB-Partner (Payment gateway).
- Owners: Product (purpose), Security (policy), Data Stewards (classification), Engineering (implementation evidence).
  Trong phạm vi: xử lý dữ liệu khách hàng/đơn hàng/thanh toán/sự kiện giao hàng. Ngoài phạm vi: nội bộ nhà cung cấp thanh toán. Ranh giới tin cậy: TB-WebAPI, TB-Data, TB-File, TB-Partner.

Misuse/Abuse Cases

- User tries mass address uploads to infer token mappings via timing.
  Người dùng thử tải lên hàng loạt địa chỉ để suy ra ánh xạ mã thông báo thông qua thời gian.
- Insider uploads spreadsheet with PII into analytics bucket bypassing vault.
  Người dùng nội bộ tải bảng tính có PII vào thùng phân tích, bỏ qua kho lưu trữ.
- Attacker replays PSP webhook to mark unpaid orders as paid.
  Kẻ tấn công phát lại webhook PSP để đánh dấu các đơn hàng chưa thanh toán là đã thanh toán.
- Operator resolves tokens without ticket/approval (insider misuse).
  Người vận hành giải quyết mã thông báo mà không cần vé/phê duyệt (lạm dụng nội bộ).

DATA CLASSIFICATION & DATA FLOW

Data Classification

Data class	Examples	Sensitivity	Owner	Notes (protections by design)
PII	Name, email, phone, address	High	Customer Ops	Default sensitive; opaque IDs used across services; logs avoid PII.
Financial (Fin)	Card PAN token, payment intent id	High	Payments	Never store raw PAN; only PSP token & last4 for UX.
Order data	Order id, items, totals	Medium	Commerce	Public when aggregated; private when tied to identity.
Public info	Store addresses, product catalog	Low	Merchandising	Context sensitive when paired with presence/location.
Operational logs	Request id, actor, 5W1H	Low	SRE	No raw identifiers; tamper-evident storage; retention policy.

Data Flow

- UI → Web API (TB-WebAPI): submit order → API exchanges PII for cust_tok.
- API → Data Vault (TB-Data): store PII; receive cust_tok; persist order with tokens.
- API → Payment Gateway (TB-Partner): create payment intent with PSP tokens; receive payment_intent_id.
- API → Order/Inventory/Shipping: propagate order_id, cust_tok, payment_intent_id (no raw PII).
- Events/Logs: publish events with tokens; reporting subscribes without PII.
- Auditor (exceptional): resolve tokens via break-glass process.

Retail Sales System – Controls

PROTECTION GOALS

Measurable security requirements

Goal area	End goal (requirement)	Verification / Evidence
Confidentiality	All customer-related exchanges occur over authenticated, confidential channels with forward secrecy.	Automated compliance report; pen-test summary; traffic inspection tests.
Integrity	State changes (orders, payments) are recorded with integrity and cannot be altered undetected.	Append-only audit tables; hash-chain verification; integrity tests.
Availability	Core purchase flows stay within defined error budgets and rate limits even under abuse.	Load/chaos test results; rate-limit alerts; SLO dashboards.
Authentication & Authorization	Only intended principals act within scoped permissions on their own data.	Access tests incl. IDOR probes; role mapping review.
Auditability	Every side-effecting action is attributable and reconstructible for 365 days.	Log schema sample; retention setting; reconstruction drill.
Privacy & Minimization	Personal data is collected only with specific purpose and removed at end of retention.	Data map; retention matrix; deletion tombstones; purge drill.

Opaque Identifier Pattern

- Design: convert sensitive fields to tokens at TB-Data; pass tokens across services; dereference only within Data Vault under privileged flows (auditor).
  Chuyển dữ liệu nhạy cảm thành token tại TB-Data; truyền token giữa dịch vụ; chỉ giải mã trong Data Vault khi có đặc.

Entity	Real fields (sealed in vault)	Token carried in system	Notes
Customer	name, email, phone, address	cust_tok	UI/API dùng cust_tok; Analytics uses anonymous data.
Payment method	PAN, expiry	card_tok (from PSP)	Save PSP token; last4 for UX; no PAN.
Log event	—	req_id, actor_id, cust_tok	No PII in logs; audit trail as needed.

Retention & Secure Deletion

Data class	Retention	Deletion trigger	Backup/Replica propagation
PII (customer)	24 months from last activity	Inactivity or user request	Purge DB/object store/caches; tombstone; backup purge
Payment tokens	Per PSP contract (shortest viable)	Card removal/expiry	Invalidate at PSP; remove local references
Orders	7 years (tax/compliance)	Retention end	Archive anonymized; keep totals only
Logs	365 days	Age-out	Archive then purge with verification

Retail Sales System – Threat List · Residual Risks · Evidence

THREAT CLASSIFICATION & RESIDUAL RISKS

Threat Classification by STRIDE

Area	S	T	R	I	D	E
Web API	Session/JWT theft	Param tampering	Missing admin audit	PII leakage via errors/IDOR	Flood submissions	Privilege escalation
Data Vault	Fake client at TB-Data	Swap token↔data map	Unlogged token resolution	Vault dump	Lock contention DoS	Bypass via debug tools
Partner	Rogue PSP callback	Tamper payment_intent_id	Dispute w/out evidence	Leak via webhook payloads	Replay callbacks	Abuse admin keys
Logs	Spoof actor	Alter entries	Deny actions	Sensitive data in logs	Log spam	Debug endpoints leak

Residual Risks & Acceptance

- Vault compromise — mitigated by isolation/monitoring; residual documented.
  Xâm phạm kho lưu trữ — được giảm thiểu bằng cách cô lập/giám sát; ghi chép lại dữ liệu còn sót lại.
- Side channels (timing/size) — normalized responses; anomaly detection.
  Các kênh phụ (thời gian/kích thước) — phản hồi được chuẩn hóa; phát hiện bất thường.
- Human error in token resolution — dual approval; periodic audit.
  Lỗi của con người trong quá trình giải quyết mã thông báo — phê duyệt kép; kiểm toán định kỳ.

Claim → Argument → Evidence

Claim	Argument (design rationale)	Evidence (what to attach)
Logs contain no PII	All flows pass tokens; log schema forbids PII fields	Schema lint report; sample logs
Customer PII is deletable	PII sealed in vault; referenced by token only	Delete job report; tombstones; purge drill
Payments resist replay	Webhooks bound to order nonce; idempotent processing	Unit/negative tests; signature verification report
Least privilege access	Separate roles for API, reporting, auditor	IAM policy review; access diff

Integrating Privacy into Design

CONCEPT

Why Privacy in Design Matter

Privacy is about human consequences of data handling: legal/regulatory duties, user expectations, and harm from disclosures. Good design limits access, makes misuse harder, and leaves audit-ready trails. When in doubt, escalate to the privacy officer for written sign-off.

Quyền riêng tư đề cập đến hệ quả đối với con người của việc xử lý dữ liệu: nghĩa vụ pháp lý/quy định, kỳ vọng người dùng và thiệt hại khi lộ lọt. Thiết kế tốt hạn chế truy cập, giảm sai sót và để lại dấu vết kiểm toán. Khi không chắc chắn, chuyển cho cán bộ phụ trách riêng tư phê duyệt bằng văn bản.

1) Privacy Policy & Guidance

Owner: Privacy/Legal • Sources: public & internal policy

Promises: collection • purpose • sharing • retention
Obligations for 3rd-party data (partners)
Ambiguity → written answers/sign-off

→

2) Translate to Requirements

Design section — measurable privacy constraints

- Purpose limit
- Data Minimization
- Retention limit
- Sharing Approvals

→

3) Design Controls

Implementable enforcement points

- Access Scoping
- Audit Trails (5W1H)
- Retention & Deletion Jobs
- Data Sharing and Approvals Gates

POLICY → REQUIREMENTS MAPPING

Mapping policies to measurable requirements

- Identify new data collection and verify policy compliance.
  Xác định dữ liệu mới và kiểm tra phù hợp chính sách.
- Confirm allowed purpose of use; build checks at interfaces.
  Xác nhận mục đích sử dụng; kiểm tra tại giao diện.
- Limit access to trained staff when unlimited use is technically possible.
  Giới hạn truy cập với nhân sự được đào tạo.

LIFECYCLE & RETENTION

Data lifecycle & retention

- Retention terms translate to timely deletion including backups.
  Chuyển thời hạn lưu giữ thành xoá đúng hạn kể cả bản sao lưu.
- Delete disused fields to reduce disclosure risk.
  Xoá trường không dùng để giảm rò rỉ.
- Track consent and data-sharing approvals.
  Theo dõi chấp thuận và phê duyệt chia sẻ dữ liệu.

AUDITING & APPROVALS

Audit Trails

- Log all sensitive access with 5W1H; require break-glass justification.
  Ghi mọi truy cập nhạy cảm kèm 5W1H; yêu cầu lý do khẩn cấp.
- Approval process before sharing data externally.
  Quy trình phê duyệt trước khi chia sẻ ra ngoài.
- Regular reviews of policy conformance and misconfiguration risks.
  Rà soát định kỳ mức phù hợp chính sách và rủi ro cấu hình sai.

AVOID PRIVACY PITFALLS — ANTI-PATTERNS

Common privacy pitfalls to avoid

- Short public policy but no internal detail → engineers left guessing.
- Using share-friendly cloud storage without guardrails.
- Unlimited access for broad roles; no training on privacy constraints.
- No access records — impossible to respond after a leak.

Making Trade-Offs

WHY TRADE-OFFS MATTER

Balancing security, complexity, and cost

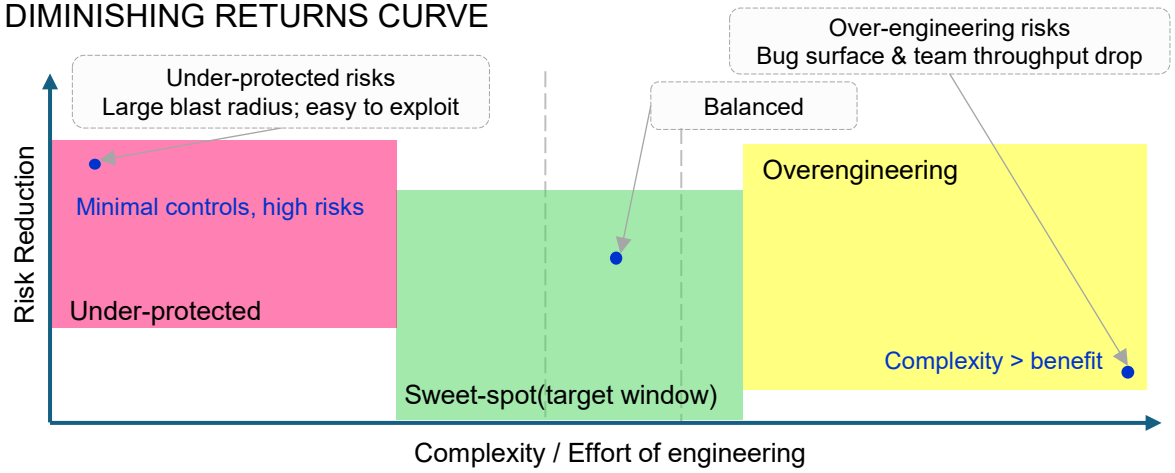
- More mitigations reduce risk—until complexity increases defect rates and maintenance cost.
  Thêm biện pháp giảm thiểu sẽ giảm rủi ro—nhưng đến một điểm sẽ làm tăng lỗi và chi phí bảo trì do phức tạp.
- Beware diminishing returns and schedule/headcount realities; perfection is not the goal.
  Cảnh giác hiệu ứng lợi ích cận dần và các ràng buộc tiến độ nhân sự; mục tiêu không phải là hoàn hảo tuyệt đối.
- Make compromises explicit in design: state goals, limits, and residual risk.
  Hãy ghi rõ các thoả hiệp trong thiết kế: nêu mục tiêu, giới hạn, và rủi ro còn lại.

WORSE CASE (CIA)

Cap the blast radius where possible

- Confidentiality: hardest to cap once data leaks (e.g., massive PII disclosure).
  Khó giới hạn nhất khi dữ liệu đã rò rỉ (ví dụ: lộ PII diện rộng).
- Integrity: cap via versioning, append-only history, and recoverability targets.
  Giới hạn bằng versioning, lịch sử chỉ-thêm, và mục tiêu khôi phục.
- Availability: cap impact with error budgets and recovery objectives (e.g., hourly backups cap loss ≤ 1h of transactions).
  Giới hạn tác động bằng error budget và mục tiêu phục hồi (ví dụ: sao lưu mỗi giờ giới hạn mất mát ≤ 1h giao dịch).

DIMINISHING RETURNS CURVE



ENTERPRISE CONTEXT AND EXCEPTION

When to raise or lower the security bar

- Most projects share a consistent baseline; call out deviations explicitly in the design preface.
  Đa số dự án dùng chuẩn cơ bản nhất quán; nếu lệch, hãy nêu rõ ngay phần dẫn nhập thiết kế.
- Raise the bar for high-liability targets (e.g., payment handling subject to special requirements).
  Nâng ngưỡng với mục tiêu rủi ro trách nhiệm cao (ví dụ: xử lý thanh toán có yêu cầu đặc biệt).
- Lower the bar for information-only sites that collect no end-user data (document the assumption).
  Giảm ngưỡng với website chỉ cung cấp thông tin, không thu thập dữ liệu người dùng cuối (và ghi nhận giả định).

FPT Education
FPT UNIVERSITY

Design Simplicity

WHY SIMPLICITY MATTERS

Less to misconfigure, fewer bug paths

- Simple designs reduce attack surface and cognitive load.
  Thiết kế đơn giản giúp giảm bề mặt tấn công và gánh nặng tư duy.
- Beware mechanisms that must be “exactly right”; prefer fewer, clearer rules.
  Cảnh giác cơ chế đòi hỏi “chuẩn tuyệt đối”; ưu tiên ít quy tắc nhưng rõ ràng.
- Separate concerns: functionality inside, security as an enforcing shell.
  Tách bạch mối quan tâm: chức năng bên trong, bảo mật là lớp vỏ cưỡng chế.

ANTI-PATTERNS TO AVOID

Complexity Traps

- Hard shell, soft inside — bypass around the PEP.
  Vỏ cứng ruột mềm — đi vòng qua PEP.
- Confused Deputy — services act with elevated unintended authority.
  Confused Deputy — dịch vụ bị “mượn tay” để lạm quyền.
- Backflow of Trust — internal components trusting external inputs implicitly.
  Dòng tin cậy ngược — bên trong tin tưởng ngầm đầu vào bên ngoài.
- Third-party hooks everywhere — sprawling plugins with unclear contracts.
  Móc nối bên thứ ba khắp nơi — plugin tràn lan, hợp đồng mơ hồ.

“Simplicity is the ultimate sophistication.” — Leonardo da Vinci

PATTERNS EMBRACING SIMPLICITY

Reduce moving parts, unify enforcement

- Economy of Design — prefer fewer, clearer mechanisms.
  Tiết kiệm trong thiết kế — ít cơ chế, rõ ràng hơn.
- Least Common Mechanism — avoid shared mutable “god” components.
  Cơ chế chung tối thiểu — tránh thành phần chung có trạng thái.
- Complete Mediation — centralize checks at PEP; no side doors.
  Thẩm tra đầy đủ — kiểm tra tập trung tại PEP; không cửa phụ.
- Defense in Depth (minimal layers) — each layer adds value or gets removed.
  Phòng thủ nhiều lớp (tối giản) — lớp nào không thêm giá trị thì bỏ.
- Fail Securely — safe defaults, explicit allowlists.
  Thất bại an toàn — mặc định từ chối, danh sách cho phép.

SIMPLICITY CHECKLIST

Make it measurable

- Count enforcement points (PEP) ≤ 3 per trust boundary.
  Số điểm cưỡng chế (PEP) ≤ 3 cho mỗi ranh giới.
- Data flows crossing trust boundaries minimized and cataloged.
  Giảm và liệt kê luồng dữ liệu qua ranh giới.
- Public API surface trimmed; stable, versioned contracts.
  Bề mặt API công khai gọn; hợp đồng có phiên bản ổn định.
- Single source of truth for authn/z decisions.
  Một nguồn quyết định duy nhất cho xác thực/ủy quyền.
- Configuration keys pruned; safe defaults enforced.
  Tối giản cấu hình; mặc định an toàn.
- Document explicit assumptions and rollback/kill-switch paths.
  Ghi rõ giả định và đường quay lui/kill-switch.

Part 3 – Secure Design Review

FPT Education
FPT UNIVERSITY

Secure Design Review - Overview

- One of the best ways to bake security into software is to review the design with a dedicated security perspective before coding.
  Một trong những cách tốt nhất để đưa bảo mật vào phần mềm là rà soát thiết kế với góc nhìn an ninh riêng, trước khi bắt đầu lập trình.
- In a Security Design Review (SDR), we apply security and privacy design concepts to the architecture, just like an engineer reviews an architect's blueprint for safety.
  Trong SDR, ta áp dụng các nguyên lý thiết kế an ninh và riêng tư lên kiến trúc, tương tự như kỹ sư kiểm tra bản vẽ của kiến trúc sư để bảo đảm an toàn.
- *'A good, sympathetic review is always a wonderful surprise.' — Joyce Carol Oates*
  'Một buổi review chân thành, thấu hiểu luôn là một bất ngờ tuyệt vời.'

What is SDR?

- Security-focused review of the design before coding.
  Rà soát thiết kế với góc nhìn an ninh, trước khi viết code.
- Separate session from normal feature design discussion.
  Tách buổi review an ninh khỏi thảo luận thiết kế tính năng thông thường.
- Uses security and privacy design concepts as a checklist for the architecture.
  Dùng các nguyên lý thiết kế an ninh và riêng tư như một checklist cho kiến trúc.
- Aims to bake security into the design, not patch it later in code.

Nhắm tới việc đưa bảo mật vào thiết kế, thay vì chắp vá thêm ở tầng code.

Architect–Engineer Analogy

- SDR is like an engineer reviewing an architect's blueprint for safety.
  SDR giống như kỹ sư kiểm tra bản vẽ của kiến trúc sư để bảo đảm an toàn.
- Designer focuses on function and experience; reviewer checks structure and risk.
  Nhà thiết kế tập trung vào chức năng và trải nghiệm; người review xem xét cấu trúc và rủi ro.
- Both need to understand the "building codes" of security and privacy.
  Cả hai phải hiểu "luật xây dựng" về an ninh và quyền riêng tư.
- Collaboration leads to higher quality and trust in the final system.
  Sự phối hợp giúp nâng chất lượng và mức độ tin cậy của hệ thống cuối cùng.

Who Review?

- Prefer a reviewer outside the design team but close to the system.
  Ưu tiên người review không thuộc nhóm thiết kế nhưng hiểu rõ hệ thống.
- Independence improves objectivity and reduces groupthink.
  Tính độc lập giúp khách quan hơn và giảm hiệu ứng "nghĩ theo nhóm".
- Reviewer should know real-world usage, integrations, and constraints.
  Reviewer cần nắm được cách hệ thống được dùng thật, các tích hợp và ràng buộc.
- Less-familiar reviewers add value by asking naive but sharp questions.
  Reviewer ít quen hệ thống vẫn hữu ích nhờ các câu hỏi tưởng như ngây thơ nhưng sắc bén.

Why it Matter?

- SDR turns security ideas into concrete improvements and raises the bar.

SDR biến ý tưởng bảo mật thành cải tiến cụ thể và nâng mặt bằng an ninh.

- You do the best SDR on systems whose design and failure modes you know well.

Bạn review tốt nhất trên những hệ thống mà bạn hiểu thiết kế và cách chúng có thể hỏng.

- SDR catches structural issues early, when they are cheapest to fix.SDR phát hiện lỗi cấu trúc sớm, khi chi phí sửa còn thấp.
- As teams adopt SDR routinely, overall software security reliability improves.

Khi SDR trở thành thói quen, độ tin cậy an ninh phần mềm của cả tổ chức đều tăng.

Security Design Review separates normal feature design from a focused security review, combining domain knowledge and security principles to build safer systems.
Security Design Review tách quá trình thiết kế tính năng khỏi phiên rà soát tập trung vào an ninh, kết hợp hiểu biết nghiệp vụ và nguyên lý bảo mật để xây hệ thống an toàn hơn.

SRD Logistic – Why, When, Docs & Conduct

Before diving into the SDR methodology, we need to clarify some logistics: why we do SDRs, when they should happen in the design lifecycle, and why preparation and documentation matter so much.

Trước khi đi sâu vào phương pháp SDR, ta cần làm rõ vài vấn đề hậu cần: vì sao cần SDR, SDR diễn ra ở giai đoạn nào trong vòng đời thiết kế, và vì sao chuẩn bị cùng tài liệu lại quan trọng đến vậy.

WHY

Tiny cost in time, big payoff in security and assurance.

- Reviews a small slice of design time but often finds major improvements.
  Chiếm một phần rất nhỏ trong thời gian thiết kế nhưng thường phát hiện cải tiến lớn.
- Simple designs review quickly; large designs get hotspots highlighted.
  Thiết kế đơn giản được review nhanh; thiết kế lớn thì làm rõ các "điểm nóng".
- Catching issues early is far cheaper than fixing them in code or production.
  Bắt lỗi sớm rẻ hơn rất nhiều so với sửa ở tầng code hoặc sau khi triển khai.
- Brings fresh perspectives and surfaces abuse cases and unintended consequences.

Đưa thêm góc nhìn mới và khơi ra các kịch bản lạm dụng, hệ quả ngoài ý muốn.

WHEN SDR

After functional review, before design freeze; plus, optional early pass.

- Main SDR: design iteration is complete and stable but not yet final.
  SDR chính: bản thiết kế/iteration đã tương đối ổn định nhưng chưa chốt.
- Keep security separate from functional review to avoid mixed focus.
  Tách SDR khỏi functional review để tránh phân tán trọng tâm.
- For complex or security-critical systems, add a preliminary early SDR.
  Với hệ thống phức tạp hoặc rất nhạy cảm, nên có thêm một SDR sớm, ít hình thức.
- Designers own security; reviewers support them, not replace them.
  Designer chịu trách nhiệm về bảo mật; reviewer chỉ đóng vai trò hỗ trợ, không thay thế.

DOCS & CONDUCT

Good documentation and respectful communication make SDR effective.

- Up-to-date design docs ensure everyone shares the same mental model.
  Tài liệu thiết kế cập nhật giúp mọi người có cùng mô hình tinh thần về hệ thống.
- Vague phrases like "store data securely" deserve a big red flag.
  Những câu chung chung kiểu "lưu dữ liệu an toàn" cần được gắn cờ đỏ nếu thiếu chi tiết.
- Reviewers preview docs to use meeting time for discussion, not discovery.
  Reviewer nên đọc trước tài liệu để dành thời gian họp cho thảo luận, không phải nghe giới thiệu.
- Feedback should be clear, persuasive, and nonjudgmental—not pontificating.
  Phản hồi cần rõ ràng, thuyết phục và không phán xét, tránh kiểu "lên lớp đạo lý".

SDR logistics clarify why the review is worth doing, when to schedule it in the design flow, and how documentation and behavior make the session productive instead of painful.

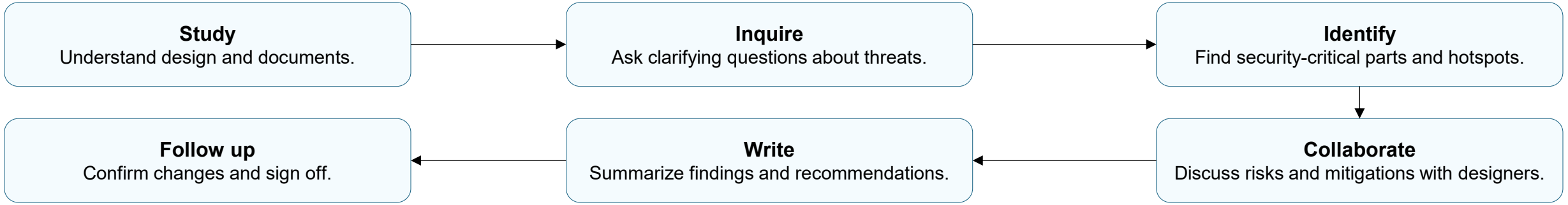
Góc nhìn "logistics" cho SDR giúp làm rõ vì sao nên review, đặt nó vào đâu trong luồng thiết kế, và cách tài liệu cùng thái độ ứng xử biến buổi review thành phiên làm việc hiệu quả thay vì căng thẳng.

SDR Process – Big Picture

The Security Design Review (SDR) process can be seen at a high level as six stages that take you from first understanding the design to finally signing off the review.

Quy trình Security Design Review (SDR) ở mức tổng thể gồm sáu giai đoạn, đưa bạn từ lúc mới tìm hiểu thiết kế cho tới khi ký duyệt kết quả review.

SIX STAGES - FROM FIRST READING TO FINAL FOLLOW-UP



CONTEXT

Formal SDR at a large software company, adaptable anywhere.

- Process described for a large company with a formal, mandatory SDR.
  Quy trình được mô tả cho một công ty lớn với SDR mang tính bắt buộc.
- Same analysis strategies can be adapted to less formal organizations.
  Cùng các chiến lược phân tích này có thể áp dụng cho tổ chức ít hình thức hơn.

WORKING STYLE

One marathon vs incremental sessions.

- Small designs: most stages can fit into a single session.
  Thiết kế nhỏ: thường gom được hầu hết bước trong một buổi.
- Large designs: break by stage; some stages need multiple sessions.
  Thiết kế lớn: chia theo giai đoạn; có bước phải làm nhiều buổi.
- Some reviewers prefer marathons; the author recommends incremental work.
  Có người thích review "marathon"; Có người khuyến nghị chia nhỏ, làm nhiều ngày.
- Sleeping on it often leads to better thinking and insights.
  "Ngủ một giấc rồi nghĩ lại" thường cho ra góc nhìn và ý tưởng tốt hơn.

The SDR process starts from understanding the design in context, runs through six clear stages, and can be executed in one intense session or over several smaller sessions depending on the design size.

Quy trình SDR xuất phát từ việc hiểu thiết kế trong bối cảnh cụ thể, đi qua sáu giai đoạn rõ ràng và có thể thực hiện trong một buổi tập trung hoặc chia nhỏ nhiều buổi tuỳ độ lớn của thiết kế.

Stages 1,2 – Study & Inquire

Read the design deeply, first to understand the system, then with your security hat on; after that, ask questions to clarify threats and fill in any gaps.

Đọc kỹ thiết kế: đầu tiên để hiểu hệ thống, sau đó với “mũ an ninh”; rồi đặt câu hỏi để làm rõ mối đe doạ và lấp những khoảng trống còn thiếu..

1. STUDY

Read, re-read, and take notes with a security-aware mindset.

- Read docs for a high-level understanding of the design.
- Re-read with your "security hat" and a threat-aware mindset.
- Take notes: ideas, observations, and questions for later stages.
- Flag potential issues now; deep security analysis comes later.

Remember: too much security can ruin a design (the all-concrete, no-window house).

Nhớ rằng “quá nhiều bảo mật” có thể phá hỏng thiết kế (ngôi nhà bê tông kín, không cửa sổ).

2. INQUIRE

Clarify threats, fix omissions, and confirm your understanding.

- Ask clarifying questions about basic threats and interactions.
- Ensure the design document is clear and complete; help fix omissions.
- Understand enough to be conversant, not necessarily an expert.
- Ask what the team worries about and why they might have no concerns.

Look for omitted details, such as implicit data storage with no security description.

Soi kỹ chi tiết bị bỏ qua, ví dụ lưu trữ dữ liệu ngầm nhưng không mô tả cách bảo vệ.

Stages 3 → 6 – Deep Analysis to Closure

IDENTIFY, COLLABORATE, WRITE, FOLLOW UP

Identify
Find security-critical parts and hotspots.

→

Collaborate
Discuss risks and mitigations with designers.

→

Write
Summarize findings and recommendations.

→

Follow up
Confirm changes and sign off.

3. IDENTIFY

Find security-critical parts and hotspots.

- Focus on interfaces, storage, and communications as central points.
  Tập trung vào interface, lưu trữ và truyền thông – các điểm trọng tâm.
- Work from exposed attack surfaces inward toward the most valuable assets.
  Đi từ bề mặt tấn công vào tới tài sản giá trị cao, như attacker thực thụ.
- Use CIA, Gold Standard, assets, attack surfaces, and trust boundaries as lenses.
  Sử dụng CIA, Gold Standard, tài sản, bề mặt tấn công, ranh giới tin cậy làm "lăng kính".
- Call out key protections explicitly in the design when needed.
  Yêu cầu ghi rõ các bảo vệ quan trọng trong tài liệu thiết kế khi cần.

4. COLLABORATE

Discuss risks and options; designer owns the design.

- Provide a security perspective on risks and mitigations, even for strong designs.
  Đưa ra góc nhìn bảo mật về rủi ro và cách giảm thiểu, ngay cả khi thiết kế đã tốt.
- Use concrete scenarios to show how security changes pay off over time.
  Dùng kịch bản cụ thể để minh hoạ lợi ích lâu dài của thay đổi bảo mật.
- Offer multiple alternatives and help weigh their pros and cons.
  Đề xuất nhiều phương án và hỗ trợ cân đo ưu/nhược.
- Accept the designer has the last word and document the decisions made.
  Chấp nhận designer là người quyết cuối và ghi lại các quyết định đã được thống nhất.

5. WRITE

Report findings and prioritize changes.

- Write a narrative report describing the security of the design and proposed changes.
  Viết báo cáo mô tả mức độ an ninh của thiết kế và các thay đổi đề xuất.
- Organize around specific design changes that address risks.
  Tổ chức nội dung xoay quanh các thay đổi thiết kế cụ thể để xử lý rủi ro.
- Use simple priorities like Must / Ought / Should for recommendations
  Dùng thang ưu tiên đơn giản như Must / Ought / Should cho các khuyến nghị.

6. FOLLOW UP

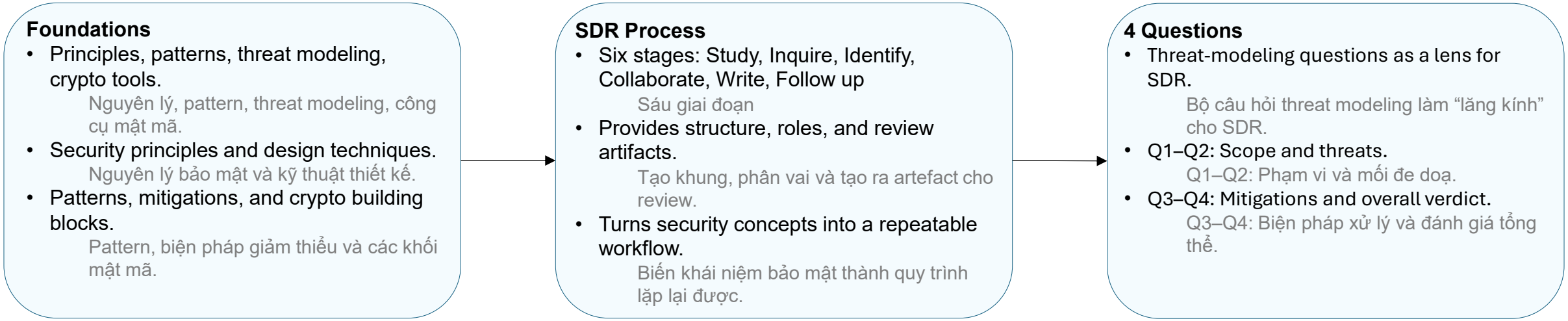
Verify changes and close the loop.

- Confirm that agreed design changes are applied correctly (often by checking docs).
  Xác nhận các thay đổi đã được thực hiện đúng (thường qua việc xem lại tài liệu).
- Track the SDR via a ticket or bug tracker item.
  Theo dõi SDR qua ticket hoặc mục trong bug tracker.
- Record differing opinions and unresolved recommendations as open issues.
  Ghi lại các quan điểm khác nhau và khuyến nghị chưa được áp dụng như các issue còn mở.
- Ideally, the designer continues to engage the reviewer as a long-term security resource.
  Lý tưởng nhất là designer tiếp tục xem reviewer như nguồn lực an ninh dài hạn.

Assessing Design Security (ADS)

- The SDR process gives you a clear structure; this section focuses on how to think about design security using the concepts and tools you have learned so far.
  Quy trình SDR cho ta một khung làm việc rõ ràng; phần này tập trung vào cách tư duy về độ an toàn của thiết kế dựa trên các khái niệm và công cụ bạn đã học.
- Foundations such as security principles, threat modeling, design techniques, patterns, mitigations, and cryptographic tools all converge into one goal: assessing how secure a design really is.
  Các nền tảng như nguyên lý bảo mật, threat modeling, kỹ thuật thiết kế, pattern, biện pháp giảm thiểu và công cụ mật mã đều hội tụ vào một mục tiêu: đánh giá thiết kế có an toàn tới đâu.
- Four Questions as a Mental Compass
  Bốn câu hỏi như một la bàn tinh thần.

FOUNDATIONS, SDR PROCESS, AND THE FOUR QUESTIONS



Assessing design security means combining your security foundations with the SDR process and using the Four Questions as a mental compass to keep your review focused.
Đánh giá độ an toàn của thiết kế là sự kết hợp giữa nền tảng bảo mật, quy trình SDR và việc dùng Four Questions như chiếc la bàn tư duy để giữ cho buổi review luôn đi đúng trọng tâm.

ADS: Q1 & Q2 — Scope and Threats

The first two Four Questions help you stay grounded: be clear about what you are building, then systematically ask what can go wrong with it.
Hai câu hỏi đầu trong Four Questions giúp bạn giữ vững trọng tâm: hiểu rõ mình đang xây gì, rồi có hệ thống hỏi xem điều gì có thể sai với thiết kế đó.

Q1 – WHAT WE ARE WORKING ON?

Clarify purpose so you can trim risk without breaking value.

Goal:

- Know the purpose of the design to cut risky, unnecessary features.
  Hiểu mục đích của thiết kế để có thể mạnh dạn cắt bỏ tính năng rủi ro, không cần thiết.
- When proposing changes, avoid breaking features that the business really needs.
  Khi đề xuất thay đổi, tránh làm hỏng các tính năng thật sự cần cho nghiệp vụ.
- Suggest safer alternative directions for risky features where possible.
  Đôi khi có thể gợi ý một hướng thay thế an toàn hơn cho các tính năng rủi ro.
- Example - Payroll health data: If health data is superfluous, cut it; if needed, protect it strongly (encrypt early, delete fast).
  Nếu dữ liệu sức khoẻ là dư thừa thì cắt bỏ; nếu cần thiết thì phải bảo vệ chặt (mã hoá sớm, xoá sớm).

Q2 – WHAT CAN GO WRONG?

Confirm threats, acceptable risks, and their future impact.

Goal:

- Confirm that important threats are anticipated and the design can withstand them.
  Xác nhận các mối đe doạ quan trọng đã được dự đoán và thiết kế đủ sức chống đỡ.
- Accept some threats only when the decision is explicit and justified.
  Chỉ chấp nhận một số mối đe doạ khi quyết định đó được ghi rõ và có lý do.
- Document accepted risks and why they are tolerable for now.
  Ghi lại các rủi ro đã chấp nhận và vì sao tạm thời có thể chấp nhận được.

Example — Applying Q1 & Q2 to a Design

Consider a student grade portal where each student logs in to view their own course results.
Xem xét một cổng tra cứu điểm, nơi mỗi sinh viên đăng nhập để xem kết quả học tập của chính mình.

Q1 – What we are working on?

- The goal is for each student to see only their own grades, anytime, from anywhere.
  Mục tiêu là mỗi sinh viên chỉ xem được điểm của chính mình, mọi lúc, mọi nơi.
- Features like viewing the whole cohort ranking are optional extras and introduce more risk.
  Các tính năng như xem xếp hạng cả khoá là phần bổ sung tuỳ chọn và kéo theo nhiều rủi ro hơn.

Q2 – What can go wrong?

- An attacker might guess IDs or manipulate URLs to see another student's grades (IDOR risk).
  Attacker có thể đoán ID hoặc chỉnh URL để xem điểm của sinh viên khác.
- If we do not constrain queries by the logged-in user, a simple bug could leak an entire class's grades.
  Nếu không ràng buộc truy vấn theo người dùng đăng nhập, một lỗi đơn giản có thể làm lộ điểm của cả lớp.
- Using Q1, we might decide not to implement cohort-wide views yet, or delay them until access controls are clearly designed.
  Dựa vào Q1, ta có thể quyết định chưa xây tính năng xem toàn bộ khoá, hoặc hoãn tới khi cơ chế kiểm soát truy cập được thiết kế rõ ràng.

ADS: Q3 & Q4 — Mitigations and Verdict

The last two Four Questions focus your attention on concrete mitigations and on giving a clear verdict about how secure the design really is.

Hai câu hỏi cuối trong Four Questions giúp bạn tập trung vào các biện pháp giảm thiểu cụ thể và đưa ra một đánh giá rõ ràng về mức độ an toàn thực sự của thiết kế.

Q2 – WHAT ARE WE GOING TO DO ABOUT IT?

Match threats to mitigations and avoid over- or under-securing.

Goal: Threat → Mitigation

- Match each important threat with design mitigations.
  Ghép từng mối đe doạ với biện pháp bảo vệ trong thiết kế.
- If something lacks mitigation, list it and explain why it matters.
  Khi thiếu mitigation, liệt kê rõ hạng mục đó và lý do quan trọng.
- Suggest options and strategies instead of one forced solution.
  Ưu tiên gợi ý hướng và phương án, không áp đặt một giải pháp duy nhất.
- Avoid homegrown or redundant security (e.g., extra TLS on HTTPS).
  Tránh cơ chế bảo mật "tự chế" hoặc lớp bảo vệ thừa.

Q4 — DID WE DO A GOOD JOB?

Summarize your verdict and tie it to weaknesses and standards.

Overall verdict box:

- Design is secure as is; no suggested changes.
  Thiết kế hiện tại đủ an toàn; không có đề xuất thay đổi.
- Design is secure with a few improvements recommended.
  Thiết kế an toàn, nhưng nên cải thiện thêm vài điểm.
- I have concerns; here are recommendations to make it more secure.
  Thiết kế còn đáng lo; đây là các khuyến nghị để cải thiện.
- Break out weaker areas and connect them to threats and impact.
  Phân tách từng khu vực yếu, gắn với threat và mức độ ảnh hưởng.
- Be explicit about the standard or comparison you are using.

Nói rõ chuẩn so sánh hoặc sản phẩm tham chiếu mà bạn dùng.

Example — Applying Q3 & Q4 to the Grade Portal

Recall the student grade portal: each student signs in to see only their own grades, and optional cohort ranking adds extra risk.

Q3 – What we are going to do about it?

- Map the IDOR threat to mitigations: enforce per-user access checks on every grade query; scope SQL queries by the logged-in student ID; delay or redesign cohort-wide ranking until access control is clear.VI:
  Ánh xạ threat IDOR sang các mitigation: bắt buộc kiểm tra truy cập theo user cho mọi truy vấn điểm; ràng buộc câu SQL theo mã sinh viên đăng nhập; hoãn hoặc thiết kế lại tính năng xếp hạng cả khoá cho tới khi cơ chế kiểm soát truy cập rõ ràng.
- Document any remaining gaps, such as APIs that reuse the same data but have weaker checks.
  Ghi lại các khoảng trống còn lại, chẳng hạn như API dùng chung dữ liệu nhưng kiểm soát lỏng hơn.

Q4 – Did we do a good job?

- Our current verdict might be: "Design is secure with a few improvements recommended" — once per-user access control and query scoping are in place, the main IDOR risk is mitigated, but future features like ranking views must follow the same pattern.
  Kết luận hiện tại có thể là: "Thiết kế an toàn, nhưng nên cải thiện thêm vài điểm" — khi đã có kiểm soát truy cập theo user và ràng buộc query, rủi ro IDOR chính được giảm đáng kể, nhưng các tính năng tương lai như trang xếp hạng phải tuân theo cùng mẫu bảo vệ.
- We explain the verdict by tying it back to the threat (unauthorized grade disclosure) and the consequence (breach of student privacy), and by being explicit about the standard we aim for (comparable or better than similar university portals).
  Ta giải thích kết luận bằng cách gắn nó với threat (lộ điểm trái phép) và hậu quả (vi phạm quyền riêng tư sinh viên), đồng thời nói rõ chuẩn mà ta hướng tới (tương đương hoặc tốt hơn các cổng tra cứu điểm của trường khác).

ADS: Where to Dig, Privacy, and Updates

An SDR cannot explore every detail. You must focus effort where it matters most, ensure privacy is covered, and keep reviews up to date as the design evolves.Một buổi SDR không thể soi mọi chi tiết.

Bạn cần dồn nỗ lực vào nơi quan trọng nhất, bao phủ cả khía cạnh privacy và duy trì review cập nhật theo sự thay đổi của thiết kế.

WHERE TO DIG

Focus review effort on security-critical areas.

- You cannot review every corner; follow your instincts.
  Không thể soi mọi ngóc ngách; hãy dùng trực giác để chọn chỗ.
- Note interesting areas, then dig into the biggest concerns.
  Ghi lại vùng “đáng chú ý”, rồi quay lại đào sâu chỗ đáng lo nhất.
- Skim low-impact parts; zoom in on weaknesses and key assets.
  Lướt qua phần ít liên quan; zoom vào dấu hiệu yếu và tài sản quan trọng.
- Watch attack surfaces and trust boundaries around valuables.

Đặc biệt chú ý bề mặt tấn công và ranh giới tin cậy quanh tài sản giá trị.

PRIVACY REVIEWS

Ensure design behavior matches privacy policy.

- Decide if privacy is inside SDR or a separate review.
  Quyết định xử lý privacy trong SDR hay bằng review riêng.
- Align with policy on collection, use, storage, sharing.
  Đảm bảo tuân thủ policy về thu thập, dùng, lưu trữ, chia sẻ.
- Walk the privacy policy and map clauses onto the design.
  Đi qua privacy policy, soi các điều khoản vào thiết kế.
- Get privacy/legal sign-offs when expertise is required.
  Nhờ chuyên gia privacy/legal phê duyệt khi cần chuyên môn.

REVIEWING UPDATES

Keep SDRs current as the software changes.

- Designs evolve; review the design delta, not just v1.
  Thiết kế luôn thay đổi; review phần delta, không chỉ bản đầu.
- Treat design docs as living, versioned architecture records.
  Xem tài liệu thiết kế như sổ theo dõi kiến trúc có phiên bản.
- Focus updates near security-critical areas; log findings.
  Tập trung review cập nhật quanh khu vực nhạy cảm và ghi lại kết quả.
- Do not underestimate "simple changes"; a quick SDR is cheap insurance.
  Đừng xem nhẹ "thay đổi nhỏ"; một SDR nhanh là lớp bảo hiểm rẻ.

Start by digging where security risk is likely to hide, make sure privacy promises are enforced in the design, and keep SDRs in sync with every meaningful design change.
Bắt đầu đào sâu ở những nơi rủi ro bảo mật dễ ẩn nấp, đảm bảo các cam kết về privacy được hiện thực hoá trong thiết kế và giữ cho SDR luôn đồng hành với mọi thay đổi thiết kế có ý nghĩa.

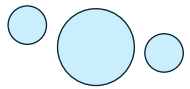
Example — Grade Portal: Where to Dig, Privacy, and Updates

For the student grade portal, focus your review on login, grade lookup APIs, and database access; treat grade records as sensitive personal data; and rerun SDRs whenever you add features like cohort ranking or export reports.
Với cổng tra cứu điểm, hãy tập trung review vào đăng nhập, API tra cứu điểm và quyền truy cập cơ sở dữ liệu; xem bản ghi điểm như dữ liệu cá nhân nhạy cảm; và chạy lại SDR mỗi khi bổ sung tính năng như xếp hạng cả khoá hoặc xuất báo cáo.

Managing Disagreement – Human Factor in SDR

- Security critique is deeply technical, but successful SDRs depend just as much on people: how we communicate, collaborate, and handle disagreement.
  Nhận xét về bảo mật là công việc rất kỹ thuật, nhưng một buổi SDR thành công phụ thuộc không kém vào con người: cách ta giao tiếp, hợp tác và xử lý bất đồng.
- Even the best technical analysis will fall flat if people feel attacked. SDRs work best when reviewers are seen as partners helping the team ship a safer product.
  Ngay cả phân tích kỹ thuật tốt nhất cũng sẽ vô nghĩa nếu người khác cảm thấy bị tấn công. SDR hiệu quả nhất khi reviewer được nhìn nhận như người đồng hành giúp team đưa ra sản phẩm an toàn hơn.

WHY PEOPLE MATTER



Technical review, human collaboration.

- SDRs are technical, but critiquing designs depends heavily on communication and collaboration.
  SDR là hoạt động kỹ thuật, nhưng việc góp ý thiết kế phụ thuộc rất nhiều vào giao tiếp và hợp tác.
- Security reviewers often get a reputation for being overly critical "outsiders".
  Người làm security dễ bị xem là "kẻ ngoài cuộc" luôn bắt bẻ.
- That perception can poison interactions and reduce the effectiveness of SDRs.
  Ấn tượng đó làm hỏng tương tác và giảm hiệu quả review.

MINDSET AND TONE

Adversarial content, collaborative tone.

- SDRs are adversarial in content, but the tone must be respectful and collaborative.
  Nội dung SDR mang tính "bắt lỗi", nhưng giọng điệu phải tôn trọng và cùng phe.
- Treat issues as teaching and learning opportunities, not proof of incompetence.
  Xem vấn đề như cơ hội học hỏi, không phải bằng chứng kém cỏi.
- "Nice goes further than mean" – the goal is a better design, not winning an argument.
  "Tử tế đi xa hơn gay gắt" – mục tiêu là thiết kế tốt hơn, không phải thắng tranh luận.

Example:

- Instead of saying "This design is careless", say "There are a few security gaps; if we fix them, the system will be much stronger".
  Thay vì nói "Thiết kế này cẩu thả", hãy nói "Ở đây còn vài lỗ hổng bảo mật; nếu mình xử lý, hệ thống sẽ chắc chắn hơn nhiều".
- Instead of "You didn't think about security at all", say "If we add these checks, our security will match the rest of the system".
  Thay vì "Anh/chị chẳng nghĩ gì về security cả", hãy nói "Nếu mình thêm các kiểm tra này, mức bảo mật sẽ đồng bộ với phần còn lại của hệ thống".
- Instead of "This makes no sense", say "I'm having trouble following this part; could we walk through the assumptions together?".
  Thay vì "Đoạn này vô lý", hãy nói "Em hơi khó theo dõi đoạn này; mình có thể cùng xem lại các giả định không?".

Managing Disagreement – Communicate Tactfully

- Turning hard feedback into constructive collaboration is a core SDR skill: what you say matters, but how you say it often matters even more.
  Biến những góp ý khó nghe thành hợp tác mang tính xây dựng là kỹ năng cốt lõi trong SDR: bạn nói gì quan trọng, nhưng cách bạn nói thường còn quan trọng hơn.
- Great SDR reviewers are generous with help but selective with criticism: they frame feedback as support, and they filter details so that teams can focus on what matters most.
  Reviewer SDR giỏi thường hào phóng trong việc hỗ trợ nhưng chọn lọc khi phê bình: họ trình bày feedback như một sự hỗ trợ và lọc bớt chi tiết để team tập trung vào những gì quan trọng nhất.

TACTFUL FEEDBACK PATTERNS (DO'S)

How to say hard things in a helpful way.

- Suggest fixes or additions with clear security rationale.
  Gợi ý chỉnh sửa/bổ sung kèm lý do bảo mật rõ ràng.
- Offer help (review docs, suggest edits) without doing the team's job.
  Sẵn sàng hỗ trợ (review tài liệu, gợi ý wording) nhưng không làm thay team.
- Present feedback as "my perspective", not as non-negotiable demands.
  Trình bày feedback dưới dạng "góc nhìn của tôi", không phải mệnh lệnh.
- Use the "sandwich" method: positive → issues → positive outcome.
  Dùng "sandwich": khen → góp ý → lợi ích sau khi cải thiện.
- Ask how they prefer to receive extensive feedback (doc, meeting, few key points).
  Hỏi trước họ muốn nhận feedback dày đặc theo cách nào (tài liệu, họp, hay chỉ vài ý chính).

FOCUS & INFORMATION GATHERING

Collect what you need, but filter what you send.

- Explore all leads, but limit feedback to the most significant points.
  Khai thác các dấu hiệu nhưng chỉ phản hồi điểm quan trọng, đừng ôm hết.
- Decide what to document versus what to just ask informally.
  Phân biệt nội dung cần ghi vào tài liệu và nội dung chỉ hỏi miệng/email.
- When docs are weak, use other channels: prototypes, code, reviews, meetings.
  Khi tài liệu kém, dùng kênh khác: prototype, code, review hoặc dự họp team.
- Before the meeting, sort notes: ask-in-advance, self-answer, discuss, report-only.
  Trước buổi họp, phân loại ghi chú: hỏi trước, tự tìm câu trả lời, bàn trong họp, chỉ ghi vào report.

Example — A difficult SDR, done tactfully

Instead of sending a 97-bullet-point email to a stressed lead, you might propose: "I've grouped my findings into five top issues and a longer appendix. Would you prefer a short doc to read first, or a meeting where we walk through the highlights together?

"Thay vì gửi một email 97 đầu mục cho một lead đang quá tải, bạn có thể đề xuất: "Em đã gom các phát hiện thành 5 vấn đề chính và một phụ lục dài hơn. Anh/chị muốn đọc nhanh một tài liệu ngắn trước, hay hẹn một buổi để mình cùng đi qua các ý chính?"

Managing Disagreement - Escalating Disagreements

- From "agree to disagree" on minor issues to formal escalation on major risks, SDRs need clear rules for how disagreements are recorded and, when necessary, elevated.
  Từ việc "đồng ý là đang bất đồng" với những vấn đề nhỏ cho tới quy trình escalate chính thức với rủi ro lớn, SDR cần những quy tắc rõ ràng về cách ghi nhận và, khi cần, nâng cấp bất đồng.
- Not every disagreement deserves escalation. Capture small differences for the record, but when big risks are at stake, write a balanced memo so leadership can make an informed, security-aware decision.Không phải mọi bất đồng đều cần escalate.
  Hãy ghi nhận những khác biệt nhỏ để làm hồ sơ, nhưng khi liên quan tới rủi ro lớn, hãy lập một bản memo cân bằng để lãnh đạo có thể ra quyết định với đầy đủ góc nhìn về bảo mật.

MINOR VS MAJOR DISAGREEMENT

Record small disagreements; reserve escalation for big risks.

- For minor issues, "agree to disagree" and record them in a "Recommendations Declined" section.
  Với vấn đề nhỏ, "đồng ý là đang bất đồng" và ghi vào mục "Khuyến nghị không được áp dụng".
- Explain the recommendation and potential consequences if the change is not made.
  Mô tả khuyến nghị và hậu quả nếu không triển khai.
- The goal is traceability, not "I told you so".
  Mục tiêu là truy vết quyết định, không phải "tôi đã nói rồi".

FORMAL ESCALATION & ROOT CAUSES

When stakes are high, write it down and bring in decision-makers.

- For major disputes, both designer and reviewer write their positions, common ground, risks, and options.
  Với tranh cãi lớn, designer và reviewer đều viết lập trường, điểm đồng thuận, rủi ro và phương án.
- Use this memo as input for management decisions and attach it to the SDR report.
  Dùng memo đó làm đầu vào cho quyết định của management và đính vào báo cáo SDR.
- Strong disagreements often come from hidden assumptions about use cases or data value.
  Bất đồng mạnh thường xuất phát từ giả định ngầm về cách dùng hoặc giá trị dữ liệu.
- Ask: "If we change our mind after release, how much harder will this be?" – usually lean safer.
  Hãy hỏi: "Nếu đổi ý sau khi release, sửa sẽ khó tới mức nào?" – thường nên nghiêng về phương án an toàn hơn.

Examples:

- Reviewer suggests logging a warning when a non-critical validation fails; designer prefers only debug-level logging.→ minor
  Reviewer đề xuất log cảnh báo khi một kiểm tra không quan trọng bị fail; designer muốn chỉ log mức debug.
- Reviewer recommends restricting an admin monitoring API to an internal network; designer wants to expose it publicly with only a basic API key. → major.
  Reviewer khuyến nghị giới hạn API giám sát admin trong mạng nội bộ; designer muốn mở ra internet chỉ với một API key đơn giản.
- Reviewer insists that every grade lookup must be scoped by the logged-in student ID; designer thinks a general “/grades?studentId=...” endpoint is fine because “students won’t guess each others’ IDs”. → major.
  Reviewer yêu cầu mọi truy vấn điểm phải ràng buộc theo ID sinh viên đang đăng nhập; designer cho rằng endpoint chung “/grades?studentId=...” là ổn vì “SV sẽ không đoán ID của nhau đâu”.