Study Plan — OWASP API Security Top 10

A User Stories Template for Practical Mastery

Compiled for Jordan Suber

Contents

How to Use This Template

Each card below turns an OWASP API Top 10 (2023) risk into an actionable user story with tasks and verifiable outcomes. Copy any card into your backlog and adjust fields (*Persona*, *Dependencies*, estimates) to your context. Keep stories INVEST-compliant and attach evidence (reports, logs) to each PR/build.

References. OWASP API Security Top 10 (2023): API1–API10. See https://owasp.org/www-project-api-security/ for canonical definitions and guidance.

1 Writing Effective User Stories (Quick Primer)

Required Story Data

- **Epic/Feature** (traceability), **Business Value** (why it matters), **Priority**, **Estimate (SP)**, **Persona**, **Dependencies**, **Assumptions/Risks**.
- **Acceptance Criteria** written in BDD form (Given/When/Then), observable in CI or runtime telemetry.
- **Evidence** links: build artifacts, scanner reports, dashboards, and test logs.

INVEST Heuristics

- **Independent:** avoids cross-team blocking.
- **Negotiable:** scope can be right-sized.
- **Valuable:** risk reduced, compliance improved, or time saved.
- **Estimable:** bounded by clear AC and DoD.
- **Small:** completes within a sprint (1–5 SP typical).
- **Testable:** pass/fail can be automated.

Good AC Patterns (Examples)

- **Given** a user with token `A` and another user's object ID, **When** the GET is attempted, **Then** the API returns `403` and logs an authZ denial with subject/object IDs.
- **Given** login endpoint, **When** 30 requests are sent within 10s from one IP, **Then** rate limiter returns `429` with retry headers and no auth state change.

2 Kickoff & Lab Setup

W0-Setup — establish a practice repo and CI scaffolding

Epic / Feature	Enablement / Foundations
Business Value	Create a safe space to iterate on risks with automated evidence and reproducibility
Priority / Estimate	Priority: Must · SP: 3
Persona	security champion
Dependencies	VCS, CI runner, sample API
Assumptions / Risks	Tooling install time; align languages and package managers across team

Story *As a security champion, I want to establish a practice repo and CI scaffolding so that Create a safe space to iterate on risks with automated evidence and reproducibility.*

Non-Functional Security · Performance · Reliability · Privacy · Observability

Acceptance Criteria (Generic, BDD)

Scenario	Outcome is evidenced in CI
Given	the target API, spec, and test environment are available
When	the tasks below are implemented and tests are executed in CI
Then	the stated outcome is observable in reports, logs, and job summary

Definition of Ready: Persona clear; AC drafted; Dependencies known; Estimate set. • **Definition of Done:** All ACs pass; Tests green; Security checks; Docs updated; Deployed/flagged.

Setup Tasks

- ☐ Initialize repo structure: `/openapi`, `/tests`, `/policies`, `/load`, `/docs`.
- ☐ Add CI workflow: lint spec (Spectral), run unit tests, publish artifacts and job summary.
- ☐ Choose a vulnerable demo API *and* a greenfield API you control.
- ☐ Seed `README.md` with architecture diagram, risk register, and Definition of Done (Security).

Risk-Specific AC for Setup

Given	CI is configured for the repo
When	a PR updates tests/spec
Then	CI emits a summary linking to lint, unit, and coverage reports with pass/fail status

3 API1 — Broken Object Level Authorization (BOLA)

API1-BOLA — enforce object-level authorization checks

Epic / Feature	Access Control
Business Value	Prevent cross-tenant and cross-user data exposure and tampering
Priority / Estimate	Priority: Must SP: 5
Persona	backend engineer
Dependencies	User/tenant model, auth context in handlers
Assumptions / Risks	Legacy endpoints may bypass middleware; map all object IDs and owners first

Story *As a backend engineer, I want to enforce object-level authorization checks so that Prevent cross-tenant and cross-user data exposure and tampering.*

Non-Functional Security Performance Reliability Privacy Observability

Acceptance Criteria (Generic, BDD)

Scenario	Outcome is evidenced in CI
Given	the target API, spec, and test environment are available
When	the tasks below are implemented and tests are executed in CI
Then	the stated outcome is observable in reports, logs, and job summary

Definition of Ready: Persona clear; AC drafted; Dependencies known; Estimate set. • **Definition of Done:** All ACs pass; Tests green; Security checks; Docs updated; Deployed/flagged.

Tasks

- ☐ Inventory resources and ownership (subject→object) per endpoint; record in a matrix.
- ☐ Implement server-side ownership checks in controllers/middleware for `GET/PUT/PATCH/DELETE`.
- ☐ Add negative tests that swap IDs in path/body/query to simulate IDOR.
- ☐ Log authZ decisions with subject, object, policy, and outcome; redact PII.
- ☐ Fail CI if any endpoint lacks an ownership test.

Risk-Specific AC

Given	user `A` and user `B` exist with distinct object IDs
When	`A` requests `/objects/{id_B}`
Then	response is `403` and audit log contains a denied authZ entry with subject/object mapping

4 API5 — Broken Function Level Authorization (BFLA)

API5-BFLA — enforce action/role checks for functions

Epic / Feature	Access Control
Business Value	Stop privilege escalation and unauthorized state changes
Priority / Estimate	Priority: Must SP: 5
Persona	service owner
Dependencies	RBAC/ABAC policy store, role catalog
Assumptions / Risks	Hidden/legacy admin endpoints; surface and block

Story *As a service owner, I want to enforce action/role checks for functions so that Stop privilege escalation and unauthorized state changes.*

Non-Functional Security Performance Reliability Privacy Observability

Acceptance Criteria (Generic, BDD)

Scenario	Outcome is evidenced in CI
Given	the target API, spec, and test environment are available
When	the tasks below are implemented and tests are executed in CI
Then	the stated outcome is observable in reports, logs, and job summary

Definition of Ready: Persona clear; AC drafted; Dependencies known; Estimate set. • **Definition of Done:** All ACs pass; Tests green; Security checks; Docs updated; Deployed/flagged.

Tasks

- ☐ Build role→action matrix (list/create/update/delete/admin) per endpoint.
- ☐ Block undocumented endpoints by default; return `404` or `403` as policy.
- ☐ Add tests that attempt admin-only actions with basic user tokens.
- ☐ Emit structured authZ logs for each action decision.

Risk-Specific AC

Given	a basic role token
When	calling `POST /admin/users`
Then	API returns `403`, no side-effects occur, and a denial is recorded

5 API3 — Broken Object Property Level Authorization (BOPLA)

API3-BOPLA — restrict access to sensitive fields

Epic / Feature	Data Protection
Business Value	Ensure only authorized roles see/modify protected properties
Priority / Estimate	Priority: Must SP: 5
Persona	API developer
Dependencies	DTO/serializer layer, field visibility rules
Assumptions / Risks	Client filtering is insufficient; enforce on the server

Story *As a API developer, I want to restrict access to sensitive fields so that Ensure only authorized roles see/modify protected properties.*

Non-Functional Security Performance Reliability Privacy Observability

Acceptance Criteria (Generic, BDD)

Scenario	Outcome is evidenced in CI
Given	the target API, spec, and test environment are available
When	the tasks below are implemented and tests are executed in CI
Then	the stated outcome is observable in reports, logs, and job summary

Definition of Ready: Persona clear; AC drafted; Dependencies known; Estimate set. • **Definition of Done:** All ACs pass; Tests green; Security checks; Docs updated; Deployed/flagged.

Tasks

- ☐ Define field visibility per role (e.g., `ssn`, `isAdmin`, secrets).
- ☐ Implement server-side projection via DTO/serializers or GraphQL resolvers.
- ☐ Add tests asserting sensitive fields never appear for unauthorized roles.
- ☐ Create write-guards to block updates to restricted fields.

Risk-Specific AC

Given	a basic user token
When	fetching `/users/me`
Then	response excludes `ssn` and `isAdmin`; schemas validate absence

6 API2 — Broken Authentication

API2-Auth — harden authentication and sessions

Epic / Feature	Identity & Sessions
Business Value	Reduce account takeover and token abuse by enforcing strong auth flows
Priority / Estimate	Priority: Must SP: 5
Persona	platform engineer
Dependencies	OAuth2/OIDC or session service, keys/rotation
Assumptions / Risks	Token validation gaps; ensure `exp/aud/iss/nbf` checks

Story *As a platform engineer, I want to harden authentication and sessions so that Reduce account takeover and token abuse by enforcing strong auth flows.*

Non-Functional Security Performance Reliability Privacy Observability

Acceptance Criteria (Generic, BDD)

Scenario	Outcome is evidenced in CI
Given	the target API, spec, and test environment are available
When	the tasks below are implemented and tests are executed in CI
Then	the stated outcome is observable in reports, logs, and job summary

Definition of Ready: Persona clear; AC drafted; Dependencies known; Estimate set. • **Definition of Done:** All ACs pass; Tests green; Security checks; Docs updated; Deployed/flagged.

Tasks

- ☐ Standardize on OIDC/OAuth2 or signed sessions; rotate signing keys.
- ☐ Enforce MFA where appropriate; rate limit login & password reset flows.
- ☐ Validate JWT claims and implement secure refresh; prevent replay.
- ☐ Add failed-login correlation logs and alerting.

Risk-Specific AC

Given	30 login attempts from one IP within 10s
When	the attempts are executed
Then	API returns `429` with retry headers; no session is established

7 API4 — Unrestricted Resource Consumption

API4-URC — apply quotas, limits, and pagination

Epic / Feature	Availability
Business Value	Protect availability and reduce noisy-neighbor effects
Priority / Estimate	Priority: Must SP: 5
Persona	SRE
Dependencies	Gateway/limiter, pagination patterns
Assumptions / Risks	Large payloads and expensive queries must be controlled

Story *As a SRE, I want to apply quotas, limits, and pagination so that Protect availability and reduce noisy-neighbor effects.*

Non-Functional Security Performance Reliability Privacy Observability

Acceptance Criteria (Generic, BDD)

Scenario	Outcome is evidenced in CI
Given	the target API, spec, and test environment are available
When	the tasks below are implemented and tests are executed in CI
Then	the stated outcome is observable in reports, logs, and job summary

Definition of Ready: Persona clear; AC drafted; Dependencies known; Estimate set. • **Definition of Done:** All ACs pass; Tests green; Security checks; Docs updated; Deployed/flagged.

Tasks

- ☐ Define per-endpoint budgets (RPS, burst, concurrency, body size).
- ☐ Implement pagination and filters; add `413` on oversize payloads.
- ☐ Load-test abusive patterns (k6/Locust); verify graceful `429`.
- ☐ Dashboards: success/4xx/5xx, rate-limit hits, latency p95/p99.

Risk-Specific AC

Given	a client exceeds the configured RPS for `/search`
When	requests continue beyond the burst window
Then	responses are `429` with `Retry-After` and no resource exhaustion occurs

8 API6 — Unrestricted Access to Sensitive Business Flows

API6-Flows — protect high-value flows from automation abuse

Epic / Feature	Abuse Mitigation
Business Value	Prevent fraud/abuse in money/credit/referral flows
Priority / Estimate	Priority: Must SP: 5
Persona	product security
Dependencies	Risk catalog, friction controls
Assumptions / Risks	Balance friction vs. false positives with telemetry

Story *As a product security, I want to protect high-value flows from automation abuse so that Prevent fraud/abuse in money/credit/referral flows.*

Non-Functional Security Performance Reliability Privacy Observability

Acceptance Criteria (Generic, BDD)

Scenario	Outcome is evidenced in CI
Given	the target API, spec, and test environment are available
When	the tasks below are implemented and tests are executed in CI
Then	the stated outcome is observable in reports, logs, and job summary

Definition of Ready: Persona clear; AC drafted; Dependencies known; Estimate set. • **Definition of Done:** All ACs pass; Tests green; Security checks; Docs updated; Deployed/flagged.

Tasks

- ☐ Identify sensitive flows; tag endpoints in OpenAPI with risk annotations.
- ☐ Add friction: velocity rules, step-up auth, proof-of-work, or device signals.
- ☐ Simulate abuse; measure precision/recall of detections.

Risk-Specific AC

Given	5 coupon redemptions within 60s by same account/IP
When	the 6th attempt occurs
Then	flow is blocked or requires step-up auth; event is logged and alerted

9 API7 — Server-Side Request Forgery (SSRF)

API7-SSRF — constrain server egress driven by user input

Epic / Feature	Network Egress
Business Value	Block pivoting to internal services and metadata endpoints
Priority / Estimate	Priority: Must · SP: 5
Persona	platform engineer
Dependencies	Egress proxy, allow-list, DNS re-resolution
Assumptions / Risks	Disable raw URL fetches; sanitize and validate destinations

Story *As a platform engineer, I want to constrain server egress driven by user input so that Block pivoting to internal services and metadata endpoints.*

Non-Functional Security · Performance · Reliability · Privacy · Observability

Acceptance Criteria (Generic, BDD)

Scenario	Outcome is evidenced in CI
Given	the target API, spec, and test environment are available
When	the tasks below are implemented and tests are executed in CI
Then	the stated outcome is observable in reports, logs, and job summary

Definition of Ready: Persona clear; AC drafted; Dependencies known; Estimate set. • **Definition of Done:** All ACs pass; Tests green; Security checks; Docs updated; Deployed/flagged.

Tasks

- ☐ Route outbound HTTP via proxy; enforce allow-list and DNS re-resolution.
- ☐ Block RFC1918/localhost/metadata IP ranges.
- ☐ Add tests attempting to reach `169.254.169.254` and internal hosts.
- ☐ Log denials with destination details; alert on patterns.

Risk-Specific AC

Given	an endpoint accepting a URL
When	a URL targeting `169.254.169.254` is submitted
Then	request is denied and logged; no egress connection occurs

10 API8 — Security Misconfiguration

API8-Misconfig — standardize secure defaults and hardening

Epic / Feature	Hardening
Business Value	Reduce attack surface via configs, headers, and container hygiene
Priority / Estimate	Priority: Must SP: 5
Persona	devops
Dependencies	Baseline headers, pinned images, minimal privileges
Assumptions / Risks	Config drift; codify checks in CI

Story *As a devops, I want to standardize secure defaults and hardening so that Reduce attack surface via configs, headers, and container hygiene.*

Non-Functional Security Performance Reliability Privacy Observability

Acceptance Criteria (Generic, BDD)

Scenario	Outcome is evidenced in CI
Given	the target API, spec, and test environment are available
When	the tasks below are implemented and tests are executed in CI
Then	the stated outcome is observable in reports, logs, and job summary

Definition of Ready: Persona clear; AC drafted; Dependencies known; Estimate set. • **Definition of Done:** All ACs pass; Tests green; Security checks; Docs updated; Deployed/flagged.

Tasks

- ☐ Enforce TLS and baseline headers (CORS, HSTS where applicable); minimize error leakage.
- ☐ Pin container images; drop root privileges; read-only FS where possible.
- ☐ Scan IaC/images (Trivy, tfsec); fail on new high/critical.
- ☐ Create a golden service template with secure defaults.

Risk-Specific AC

Given	a new service scaffolded from the template
When	CI runs
Then	baseline header tests pass and image/IaC scans report no high/critical issues

11 API9 — Improper Inventory Management

API9-Inventory — maintain authoritative API inventory and lifecycle

Epic / Feature	Governance
Business Value	Know every API/version/owner to reduce shadow risk
Priority / Estimate	Priority: Must · SP: 5
Persona	platform owner
Dependencies	API catalog/CMDB, gateway logs
Assumptions / Risks	Rogue endpoints; auto-discovery required

Story *As a platform owner, I want to maintain authoritative API inventory and lifecycle so that Know every API/version/owner to reduce shadow risk.*

Non-Functional Security · Performance · Reliability · Privacy · Observability

Acceptance Criteria (Generic, BDD)

Scenario	Outcome is evidenced in CI
Given	the target API, spec, and test environment are available
When	the tasks below are implemented and tests are executed in CI
Then	the stated outcome is observable in reports, logs, and job summary

Definition of Ready: Persona clear; AC drafted; Dependencies known; Estimate set. • **Definition of Done:** All ACs pass; Tests green; Security checks; Docs updated; Deployed/flagged.

Tasks

- ☐ Build/extend API catalog with version, owner, data classification, lifecycle.
- ☐ Auto-register services from CI on build/release.
- ☐ Detect shadow endpoints via gateway logs and repo search.
- ☐ Publish deprecation schedules and retirement runbooks.

Risk-Specific AC

Given	a new service is released
When	CI completes
Then	the service appears in the catalog with owner, version, and classification

12 API10 — Unsafe Consumption of APIs

API10-Consumption — treat upstream APIs as untrusted

Epic / Feature	Third-Party Risk
Business Value	Prevent cascading failures and data misuse from dependencies
Priority / Estimate	Priority: Must SP: 5
Persona	integration engineer
Dependencies	Timeouts/retries/circuit breakers, schema validation
Assumptions / Risks	Provider changes; contract tests mitigate

Story *As a integration engineer, I want to treat upstream APIs as untrusted so that Prevent cascading failures and data misuse from dependencies.*

Non-Functional Security Performance Reliability Privacy Observability

Acceptance Criteria (Generic, BDD)

Scenario	Outcome is evidenced in CI
Given	the target API, spec, and test environment are available
When	the tasks below are implemented and tests are executed in CI
Then	the stated outcome is observable in reports, logs, and job summary

Definition of Ready: Persona clear; AC drafted; Dependencies known; Estimate set. • **Definition of Done:** All ACs pass; Tests green; Security checks; Docs updated; Deployed/flagged.

Tasks

- ☐ Validate upstream responses against JSON Schema; reject on drift.
- ☐ Configure short timeouts, bounded retries, circuit breakers, and fallbacks.
- ☐ Add consumer-driven contract tests in CI; pin scopes/permissions.
- ☐ Maintain SBOM of integrations and usage terms.

Risk-Specific AC

Given	the upstream adds a new required field
When	contract tests run in CI
Then	the build fails with a clear schema diff and no production deploy occurs

13 Cross-Cutting Practices (Do Weekly)

Ongoing Tasks

- ☐ Update threat models (DFDs, misuse cases) with each change; link to PRs.
- ☐ Standardize structured logs: auth decisions, rate-limit hits, SSRF denials.
- ☐ Export scanner and test reports (HTML/JSON) as build artifacts.
- ☐ Review dashboards (latency p95/p99, % 4xx/5xx) and error budgets.

14 Capstone Exercise

CAP-RedBlue — run a red/blue exercise across API1–API10

Epic / Feature	Readiness
Business Value	Prove detection, response, and hardening in realistic attack flows
Priority / Estimate	Priority: Must SP: 8
Persona	security team
Dependencies	Staging env, attack scripts, runbooks
Assumptions / Risks	Time-box per scenario; record lessons and backlog follow-ups

Story *As a security team, I want to run a red/blue exercise across API1–API10 so that Prove detection, response, and hardening in realistic attack flows.*

Non-Functional Security Performance Reliability Privacy Observability

Acceptance Criteria (Generic, BDD)

Scenario	Outcome is evidenced in CI
Given	the target API, spec, and test environment are available
When	the tasks below are implemented and tests are executed in CI
Then	the stated outcome is observable in reports, logs, and job summary

Definition of Ready: Persona clear; AC drafted; Dependencies known; Estimate set. • **Definition of Done:** All ACs pass; Tests green; Security checks; Docs updated; Deployed/flagged.

Tasks

- ☐ Simulate BOLA/BOPLA/BFLA/SSRF and business-flow abuse; capture evidence.
- ☐ Execute response playbooks (block patterns, rate limits, feature flags).
- ☐ Produce a post-mortem with top 5 improvements and owners.

Checklist Snapshot (Printable)

☐ API spec linted; security schemes, schemas constrained.

☐ Ownership/role/field matrices exist with passing tests.

☐ Rate limits, quotas, and pagination verified under load.

☐ Egress proxy + allow-list; SSRF denials logged and alerted.

☐ Golden service template with scans and secure defaults.

☐ API catalog up to date; deprecation schedules published.

☐ Third-party contracts tested; timeouts/retries/circuit breakers configured.