Securing APIs as the New Perimeter

Srinivas Karthik Putlur

Principal Product Security Engineer, CBA

Why the perimeter has shifted

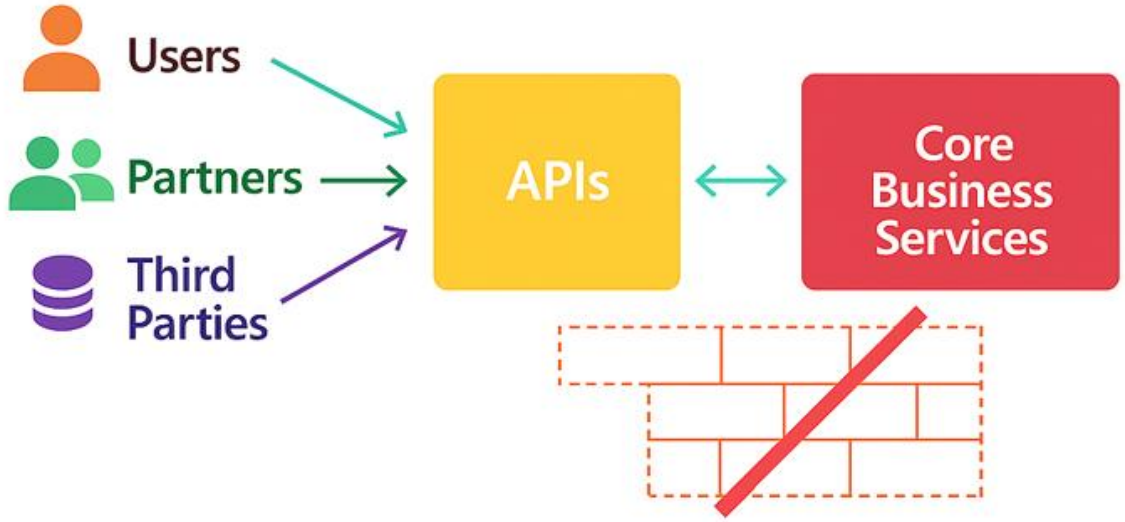
“Modern architectures—cloud, SaaS, microservices, and mobile—have eliminated the classic perimeter. The enterprise no longer lives behind a firewall; identity and APIs *are* the new boundary.”

APIs connect:

- Frontends → Backends
- Internal services → Third parties
- Partners → Core business systems

If attackers compromise APIs, they bypass UI, auth assumptions, and business controls

The traditional network perimeter is gone. APIs now expose core business logic directly to the internet.



Soundbite

"In modern architectures, APIs are the front door—not the firewall."

The Cost of Getting API Security Wrong

Most API breaches happen due to authorization failures — not exotic exploits

- Broken Object Level Authorization (BOLA / IDOR) remains the #1 API vulnerability
- Excessive data exposure leaks sensitive fields via overly broad responses
- Automation-driven abuse such as scraping, credential-stuffing, and API misuse happens over valid sessions
- Valid authentication ≠ authorized access — tokens simply don’t enforce business rules
- Traditional controls (WAFs, API gateways, infra filtering) are blind to these logic flaws
- Late detection is expensive — because the traffic looks “legitimate” until it isn’t

Why traditional security controls fall short

What doesn’t work on its own

- Firewalls & network segmentation
- Signature-based WAF controls
- Annual penetration tests

Why these controls fall short

- They don’t understand business logic
- They lack context of API schemas, data models, and workflows
- They can’t keep pace with CI/CD and daily API changes

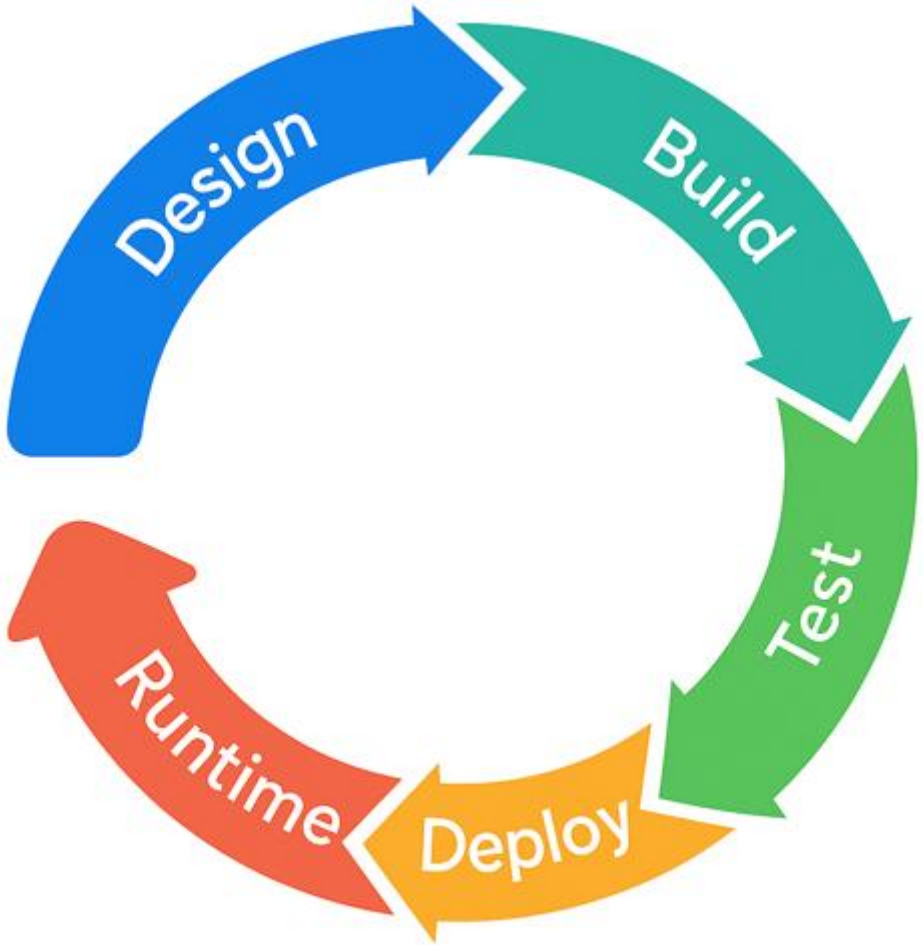
“APIs fail at the logic layer, not the network layer.”

OWASP API Security Top 10 - 2023

API1	Broken Object Level Authorization
API2	Broken Authentication
API3	Broken Object Property Level Authorization
API4	Unrestricted Resource Consumption
API5	Server-Side Request to Sensitive Business Flows
API7	Server-Side Request Forgery
API8	Security Misconfiguration
API9	Improper Inventory Management

API Security Lives in the Pipeline

"Each stage catches a different class of API risk"



Design

API Specification as a Security Contract

Treat OpenAPI / AsyncAPI specs as **security-critical artifacts**, not documentation.

Actions

- Require OpenAPI specs for all APIs (PR gate)
- Version control specs alongside code
- Enforce spec completeness:
  - Authentication schemes
  - Authorization scopes
  - Input/output schemas
  - Error handling

Security Benefits

- Prevents undocumented (“shadow”) endpoints
- Enables automated security testing
- Forms the baseline for runtime enforcement

Threat Modelling

Perform **API-specific threat modelling**, not generic STRIDE diagrams.

Focus areas

- Object-level authorization (BOLA)
- Function-level authorization (BFLA)
- Excessive data exposure
- Business logic abuse (state transitions, rate abuse)

DevSecOps pattern

- One threat model per API domain
- Updated on breaking API changes only
- Stored as markdown in repo

Build

Static Analysis for API Code

Traditional SAST focuses on syntax-level bugs, but most modern API breaches result from **logic-layer failures**. Static analysis must evolve to detect **API-specific misuse patterns** early in development.

API-Aware Static Analysis Should Detect

- Missing authentication/authorization checks on API handlers
- Insecure deserialization paths that allow malicious payloads
- Mass assignment vulnerabilities caused by automatic model binding
- Improper pagination or filtering logic leading to data exposure or scraping

By extending SAST to understand API semantics—not just code syntax—engineering teams can catch critical API risks earlier, reduce downstream defects, and improve the overall security posture of the API ecosystem.

API Specification Security Testing

Run automated security checks on the OpenAPI spec *before* any code runs.
Catches design-level vulnerabilities early, fast, and consistently.

What We Detect

- Missing Auth: Endpoints without security requirements → accidental public APIs
- Insecure Methods: Use of risky verbs (e.g., TRACE, unsafe DELETE)
- Over-Permissive Schemas: additionalProperties: true, undefined object shapes
- Sensitive Data Exposure: Fields like *password, token, secret* exposed in responses

“Perform Dependency Scanning”

Test

Dynamic API Security Testing (DAST for APIs)

Traditional DAST fails for APIs. Use **spec-driven testing**.

Approach

- Generate test cases from OpenAPI specs
- Focus on:
  - Auth bypass
  - IDOR / BOLA
  - Rate limit weaknesses
  - Input tampering

Deploy

API Gateway as a Control Plane (Not Just Routing)

Use the API gateway as a **policy enforcement point**, not a firewall replacement.

Enforce

- Authentication (OAuth2, mTLS, JWT)
- Authorization scopes
- Rate limits per consumer
- Schema validation

> “**Gateways cannot protect business logic by themselves — they are guardrails, not guards**.”

Runtime

Runtime

- **Continuous monitoring** of API behaviour to detect anomalies, abuse, and unexpected access patterns
- **Real-time threat detection** using behavioural analytics, AI/ML, and API security gateways
- **Runtime protection** for microservices via service mesh, mTLS, and dynamic policy enforcement
- **Automated incident response** (blocking, throttling, isolating workloads) when threats are detected
- **Runtime visibility** into API calls, dependencies, and service-to-service communication
- **Continuous drift detection** to identify config or infrastructure changes that violate security posture

Feedback Loop to DevSecOps

Runtime findings must **feed back into CI/CD**.

```
Runtime abuse detected →
  Create security finding →
  Add test case →
  Prevent regression in future releases

```

Common misconceptions

❌ “We have a WAF, so APIs are secure”

❌ “Auth is handled by the gateway, we’re done”

❌ “Internal APIs don’t need strong security”

❌ “We’ll fix security issues after release”

✅ APIs need **defense in depth + lifecycle coverage**

Key Takeaways

- APIs are the **new attack surface**
- Most API risks are **design & logic flaws**
- DevSecOps enables **early, automated, scalable security**
- API security must be **continuous**, not reactive

Thank you

Srinivas Karthik Putlur
https://au.linkedin.com/in/skputlur