Introduction

APIs are like *teenagers*; they need constant supervision or they'll break the rules in ways you never imagined !

API Security in DevSecOps

BUILT IN , NOT BOLTED ON

ANIRUDHA KARANDIKAR

Agenda

Why API Security is non-negotiable?

80%

Rapid API Growth

By 2025, over **80%** of all web traffic is expected to pass through APIs, making them a vulnerable target for attackers.

91%

Widespread API Attacks

91% of organizations experienced an API security incident in the last year, underscoring the critical need for proactive security measures.

30

Cost of Security Flaws

It costs **30 times** more to fix a security vulnerability in production than during the design phase, highlighting the importance of early security integration.

Why API Security is non-negotiable?

80%

Human Error in API Security

Nearly **80%** of API vulnerabilities are caused by misconfigurations and coding mistakes. Proactive security measures are needed in the API development lifecycle.

1

Rising API Threats

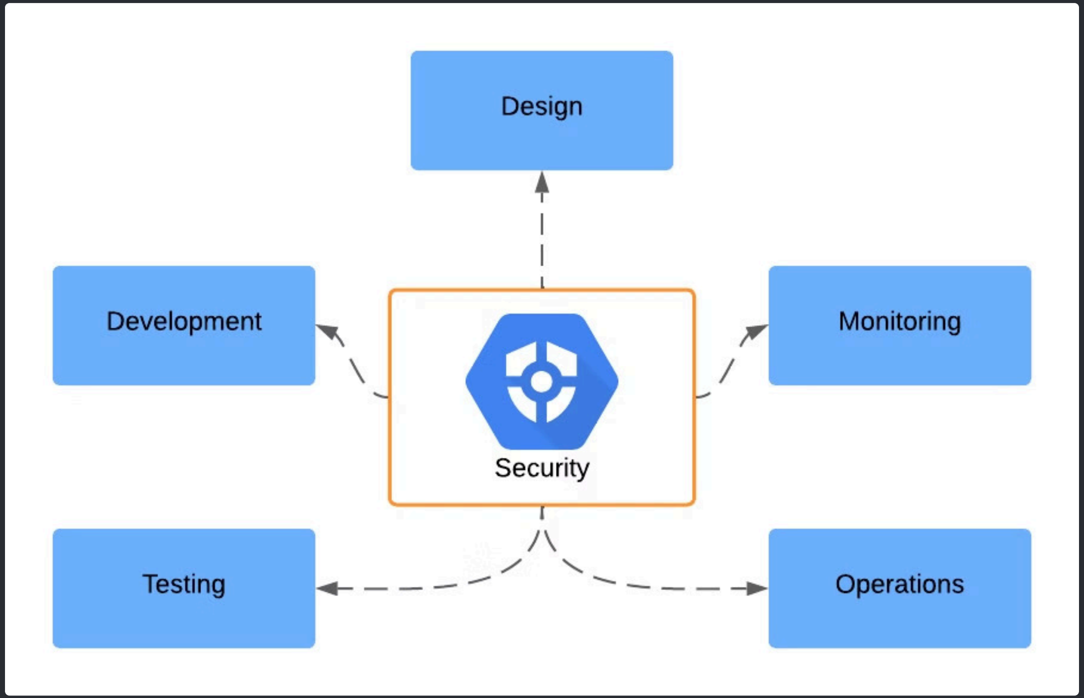
APIs are becoming the most targeted attack vector, exceeding web applications in frequency. This emphasizes the urgency for security-first approaches in API management.

40%

Security Testing Gaps

Despite the high risk, only **40%** of organizations test their APIs for vulnerabilities during development. Many systems remain exposed to attacks post-deployment.

Security Mindset



DevSecOps is a cultural shift that integrates security practices into the DevOps process, ensuring collaboration between development, security, and operations teams.

Key Principle:

The Security First Principle emphasizes embedding security at every stage of the software development lifecycle (SDLC), ensuring that security is a shared responsibility.

Objective:

To proactively identify and mitigate vulnerabilities early in development, reducing the likelihood of security issues in production environments.

API Design Security Best Practices

▼ Authentication and Authorization (Oauth 2.0)

Use strong authentication methods (e.g., OAuth 2.0) to validate user identities and control access.

▼ Data Encryption

Implement TLS for data in transit and AES for data at rest to protect sensitive information.

▼ Limit Sensitive Data

Design APIs to return only necessary data to users, minimizing exposure of sensitive information.

▼ Configuration Management

Maintain secure configurations for API servers, databases, and other components.

▼ Logging & Observability

Establish logging frameworks to monitor API usage, detect anomalies, and enable forensic analysis.

▼ Error Handling

Use generic error messages to avoid revealing implementation details while logging specifics securely.

▼ Access Control

Clearly define public and internal APIs, applying strict access controls to sensitive endpoints.

▼ Rate Limiting and Throttling

Implement rate limiting and throttling to control request frequency and prevent abuse, reducing the risk of DoS attacks.

▼ Input Validation and Data Sanitization

Validate and sanitize all incoming data to prevent injection attacks like SQL injection, XSS, and XXE attacks.

▼ Token Expiry and Refresh Policies

Set expiration times for tokens (e.g., JWTs) and establish secure token refresh mechanisms to limit misuse.

▼ API Versioning

Use versioning to support security patches in older versions, allowing secure deprecation of outdated APIs.

▼ Secure API Documentation

Restrict access to API documentation, especially for private APIs, limiting it to authenticated users.

▼ Minimum Privilege Principle

Design APIs to grant only the necessary permissions based on user roles or scopes.

▼ Request and Response Payload Size Restriction

Restrict the size of request and response payloads to prevent resource exhaustion attacks.

API Development Security Practices

Software Versions

Regularly update software libraries and frameworks to eliminate known vulnerabilities.

Dependency Management

Tools like ***Renovate*** help automate dependency updates, ensuring your software is always secure.

Secrets Management

Use tools like ***Vault*** or ***Secrets Manager*** to store secrets and sensitive information. Also tools such as ***gg-shield*** detect and prevent accidental commits of sensitive information in source code.

Security Reviews and Scans

Integrate static and dynamic analysis tools into CI/CD pipelines to continuously assess code for vulnerabilities.

Infrastructure as Code

Manage infrastructure with IaC tools like Terraform or Ansible, allowing for version control and security checks.

Automated Code Quality Checks

Use tools like SonarQube to evaluate code quality alongside security, ensuring code is maintainable and adheres to best practices, which can indirectly improve security.

API Testing Security Practices

1

Static Application Security Testing (SAST)

Analyze source code early in the development cycle to catch vulnerabilities before deployment.

2

Dynamic Application Security Testing(DAST)

Test running applications to identify security weaknesses through simulated attacks.

3

Container Scanning

Regularly scan container images for vulnerabilities to ensure secure deployments.

4

Performance Testing

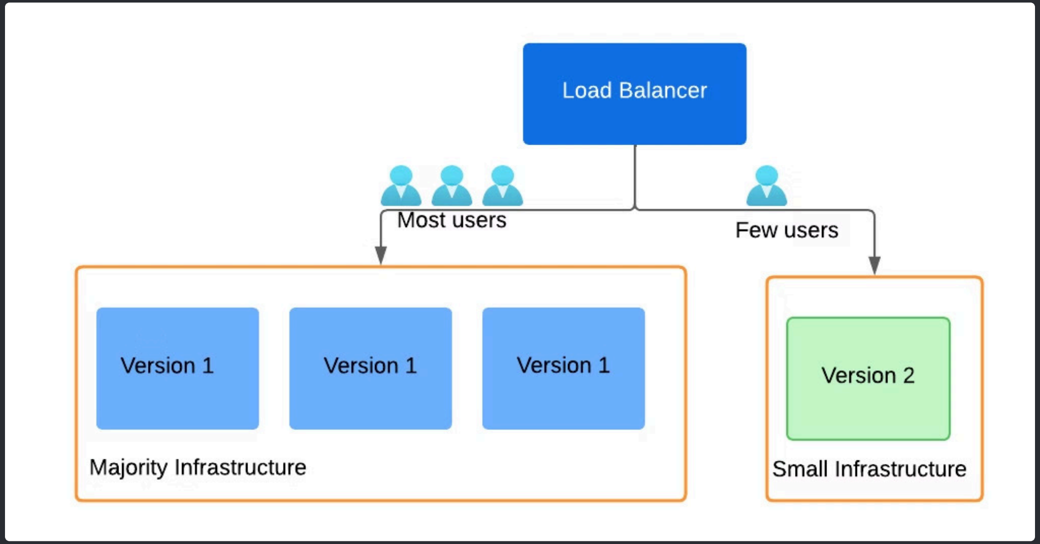
Evaluate API performance under various conditions, adjusting parameters like rate limiting to prevent abuse.

5

Security Testing

Conduct negative access testing and penetration testing to uncover and remediate security gaps effectively.

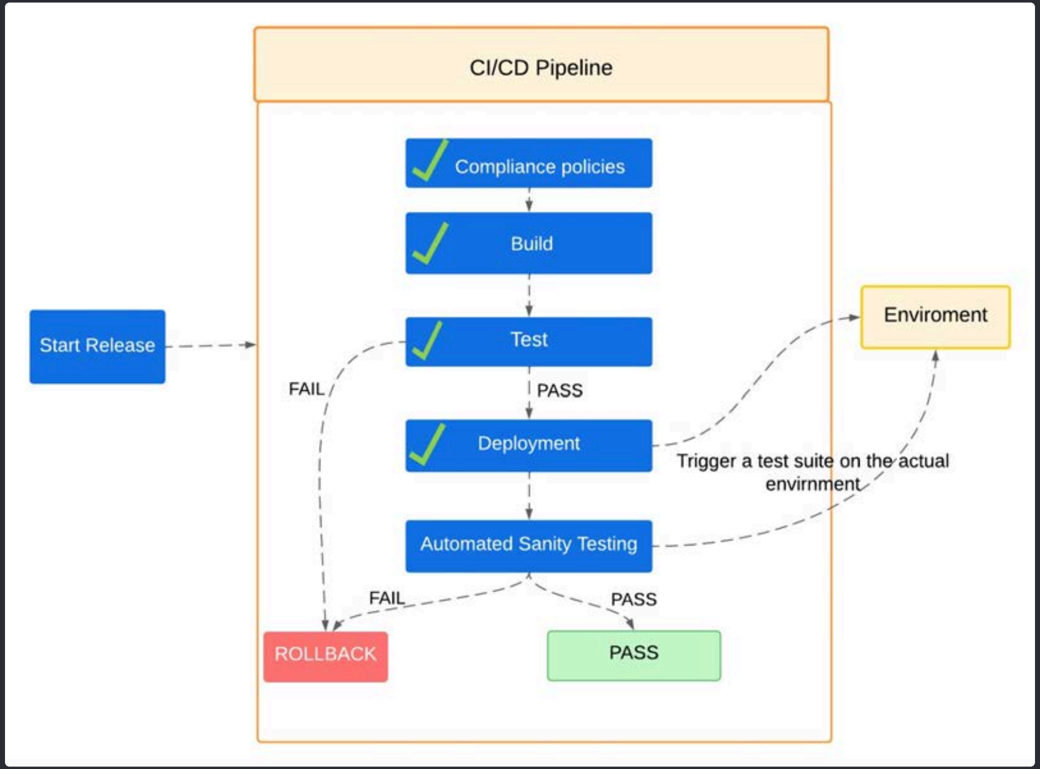
API Release Security Practices



Canary Deployment

Gradually release updates to a subset of users, allowing for monitoring and rollback if issues arise

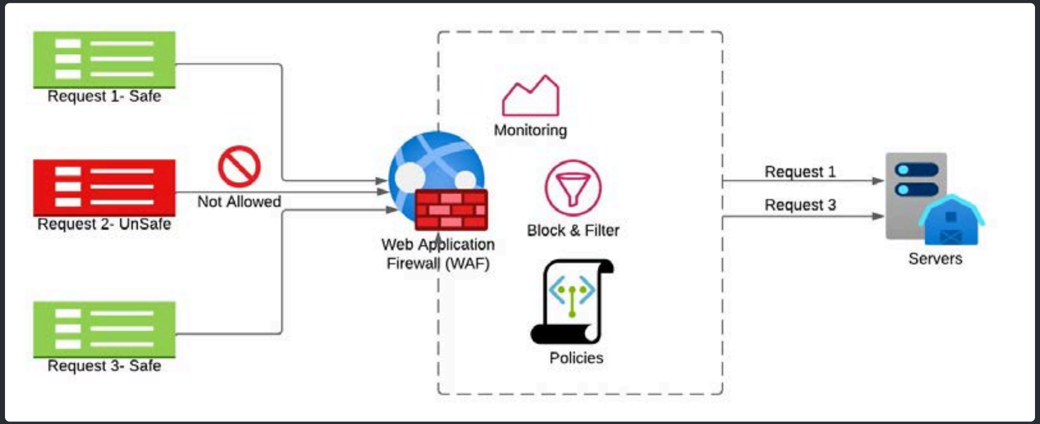
API Release Security Practices



Automated Sanity Testing

Integrate security test cases into CI/CD pipelines to ensure every deployment meets security standards

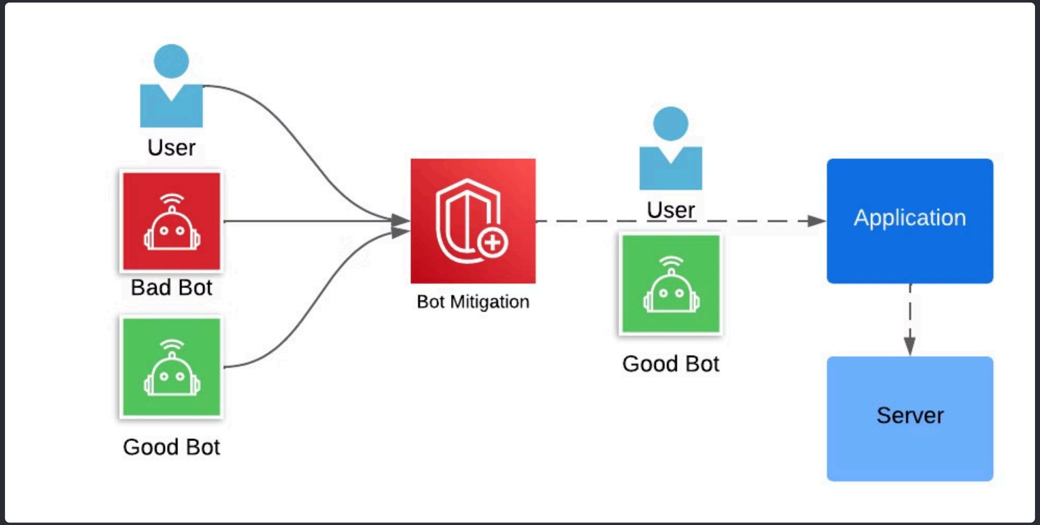
API Operational Security Practices



Web Application Firewall (WAF)

Use WAFs to shield APIs from common threats such as SQL injection and cross-site scripting (XSS).

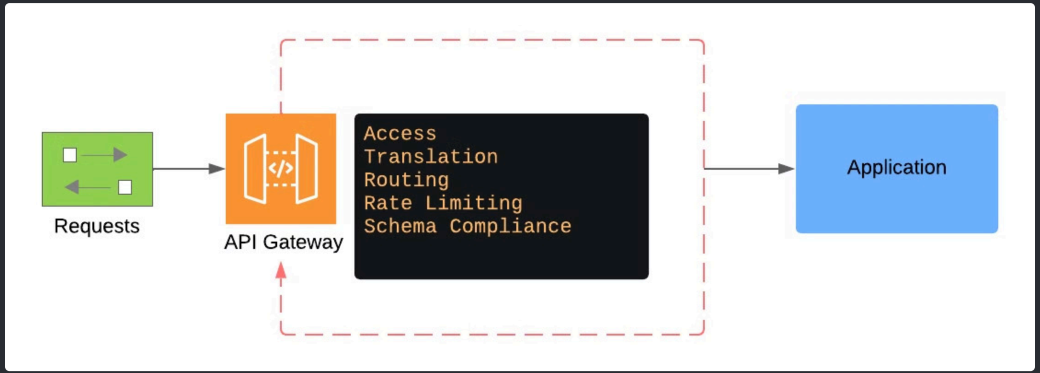
API Operational Security Practices



Bot Detection

Tools like Cequence identify and block malicious bot traffic, protecting API resources.

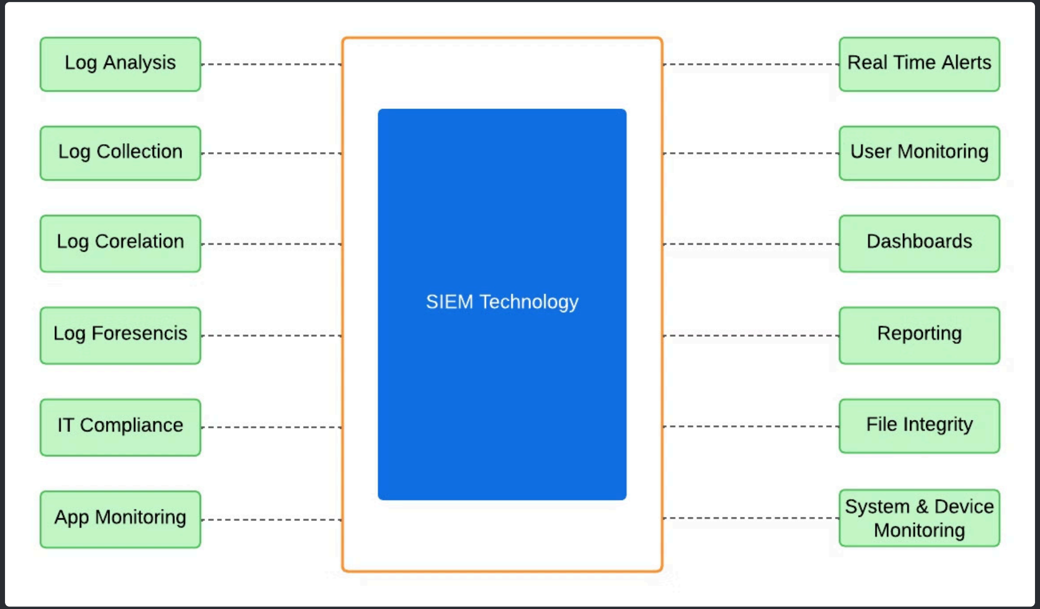
API Operational Security Practices



API Gateway

API gateways can enforce security policies, including rate limiting, authentication, and traffic filtering.

API Operational Security Practices



SIEM Solution

Implement Security Information and Event Management systems to gain real-time insights into security events.

API Monitoring Security Practices

1

Monitoring Metrics

Employ RED (Rate, Error, Duration) metrics to track API performance and availability continuously.

2

Security Control Testing

Regularly validate security headers and access controls to ensure compliance with security policies.

3

Sample Tracing

Implement tracking mechanisms to follow API requests, aiding in identifying potential vulnerabilities and performance bottlenecks.

4

Access to Tools

Ensure development and operations teams have access to monitoring and security tools to facilitate proactive security management.

Final Word

In today’s digital landscape, securing APIs is not just an option—it's a necessity. The **Security First Principle** in DevSecOps emphasizes the importance of integrating security at every stage of the software development lifecycle. By embedding security practices into design, development, testing, release, and operations, organizations can proactively identify and mitigate vulnerabilities before they reach production. This approach not only enhances the security posture of applications but also fosters a culture of shared responsibility among development, security, and operations teams.

Security **DevSecOps**