NetSPI™

API Security

Best Practices for a Proactive Approach

APIs are central to digital business operations, enabling connectivity and innovation. Yet their rapid adoption expands the attack surface, making targeted API security testing essential.

APIs are a common entry point for attackers due to issues like weak authentication, excessive data exposure, missing rate limits, and business logic flaws. However, one of the most significant and often overlooked challenges is the lack of an accurate API inventory, and most organizations, when asked, cannot confidently account for all their APIs. This gap is critical. You cannot secure assets that you do not know exist. When left unaddressed, these inventory and security gaps can lead to data breaches and significant business impact.

"Across more than 4,000 monitored environments, Thales recorded over 40,000 API incidents in the first half of 2025 alone. Although APIs represent only 14% of overall attack surfaces, they now attract 44% of advanced bot traffic, demonstrating how attackers are focusing their most sophisticated automation on the workflows that underpin critical business operations."

THALES,
API THREAT REPORT (H1 2025)

Key Elements for Securing APIs

A robust API security approach leverages established standards (such as NIST (SP) 800-228 and the OWASP API Security Top 10) to identify real-world risks. Effective programs look beyond surface vulnerabilities to find complex logic and authorization flaws. Regardless of where an organization is currently in terms of both API inventory and API security, there is a methodical way to improve both. Fortunately, irrespective of that maturity level, the phases and protections are the same.

How to Build an Accurate API Inventory:

To start, organizations must embed an inventory or cataloging process directly into the API development life-cycle. There are **2 distinct phases** for building an API inventory:

1. **Pre-Runtime:** including plan, develop, build, test, and release.
2. **Runtime:** including deliver, deploy, operate, monitor, and feedback.

In the pre-runtime phase, each new API should be accompanied by a comprehensive catalog or specification. This catalog documents endpoints, expected behaviors, data flows, and authorization models, providing a clear picture from the outset. Even if previous APIs were deployed without this rigor, all new API projects should make this phase a mandatory starting point. Similarly, new versions of existing APIs should adopt this practice, ensuring ongoing catalog accuracy as environments evolve.

Despite these best intentions, many organizations operate at varying maturity levels and often deal with incomplete inventories, especially where legacy APIs are concerned. Legacy APIs may have been launched without standardized documentation, contain remanent endpoints from initial testing phases, or have been deprecated without final removal from production environments. These orphaned or undocumented APIs create substantial blind spots, opening up more pathways for exploitation.

In addition to cataloging, ongoing monitoring and maintenance are necessary to gain visibility into real-world API usage. **Continuous observation enables organizations to understand what requests are being made, identify unauthorized or abnormal behavior, and maintain an up-to-date inventory, which further closes exposure gaps.** Left unaddressed, these inventory and security gaps can lead to data breaches and significant business impact as demonstrated by recent high-profile incidents linked to API misconfigurations.

Key Elements for Securing APIs

How to Protect an API Inventory

Building the inventory is the first step. Once an accurate inventory is compiled, there are protections for both pre-runtime and runtime phases which can be further broken down into basic and advanced protections that organizations can implement depending on current API inventory and security maturity.

Pre-runtime phase protections include:

- Using a standard specification language such as OpenAPI
- Defining schemas and valid ranges for request and responses
- Maintaining the API inventory and assigning ownership across the entire organization

Runtime phase protections include:

- Encryption
- End-user and service authentication and authorization
- Request and response validation
- Resource consumption protections such as request throttling and timeouts
- Logging and monitoring

The best way to implement these basic and advanced protections is to start inventorying what you know, including endpoints, request actions, and expected input and output. Once the inventory is complete, documentation can begin using a standard specification such as an OpenAPI catalog. A protected accurate API inventory is the first step in closing the security gap. Testing APIs can validate the protections in place to even further secure your API ecosystem.

Focus Areas for Testing:

- **Encryption:** Validate the encryption protocols and ciphers adequately protect the confidentiality and integrity of the API data in transit.
- **Authentication & Authorization:** Identify weaknesses that allow unauthorized access or privilege escalation
- **Input Validation:** Prevent injection and data manipulation through rigorous validation
- **Rate Limiting:** Defend against brute force and denial-of-service attacks
- **Business Logic:** Expose workflow gaps and logic flaws that attackers could exploit

Best Practices for Proactive API Security

By applying these practices, organizations build resilient APIs against evolving attack techniques, reduce the likelihood of breaches, and maintain trust with stakeholders.

Adopt Strong Authentication Mechanisms

Implement widely recognized standards, such as OAuth 2.0 and JWT, to verify both user and application identities. Avoid static API keys or basic authentication, which can be easily compromised if leaked. Ensure token expiration is enforced and sensitive credentials are never transmitted in URLs.

Enforce Least-Privilege Access and Rigorous Authorization

Restrict access rights for both users and systems so they can only interact with the resources necessary for their roles. Apply object-level and function-level authorization checks at every endpoint. For example, validate user ownership before granting access to specific resources, reducing the risk of unauthorized data exposure like Broken Object Level Authorization (BOLA) vulnerabilities.

Validate and Sanitize All Incoming Data

Treat every input as untrusted. Use strict data schemas to validate request parameters and sanitize values before processing. This prevents injection attacks, such as SQL/NoSQL injection and cross-site scripting (XSS) and ensures only valid data can interact with your backend infrastructure.

Monitor APIs and Implement Robust Rate Limits

Set up appropriate throttling, IP block listing, and rate limiting to prevent automated attacks, brute-force entry, data scraping, and denial-of-service events. Continuous monitoring and comprehensive logging support early detection of abnormal and excessive activity, so teams can respond before minor incidents escalate.

Continuously Test APIs Using Automated Tools and Expert-Led Assessments

Integrate automated security scans within your CI/CD pipelines to catch routine vulnerabilities early but also conduct targeted manual testing to identify nuanced business logic and authorization flaws that automation can miss. Regularly reassess your APIs as environments change and new endpoints are introduced.

Gain Confidence with Proactive Security

Unsecured APIs introduce immediate and strategic risks. A proactive security testing program strengthens defenses, reduces risk, and supports business resilience.

NetSPI's penetration testing services uncover hidden API vulnerabilities, validate security posture, and deliver actionable insights for continuous improvement.

Secure your APIs. Protect your business. Contact NetSPI to learn more.