World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(Review Article)

Securing APIs in the modern threat landscape: Best practices and challenges

Ishva Jitendrakumar Kanani *

Department of Computer Science Engineering, Kent State University, Kent, Ohio, USA.

World Journal of Advanced Research and Reviews, 2022, 13(03), 654-657

Publication history: Received on 11 February 2022; revised on 25 March 2022; accepted on 30 March 2022

Article DOI: https://doi.org/10.30574/wjarr.2022.13.3.0239

Abstract

As application ecosystems evolve toward microservices, serverless architectures, and cloud-native models, Application Programming Interfaces (APIs) have become essential conduits for data and functionality exchange. However, their ubiquity and accessibility also make them prime targets for cyberattacks. This paper explores the evolving threat landscape for APIs, outlines the security challenges associated with API-first development, and recommends best practices for securing APIs across their lifecycle. It also includes a case study on high-profile API breaches from 2021–2022 and offers practical implementation strategies aligned with NIST and OWASP frameworks.

Keywords: API Security; API Gateway; Microservices; Cloud-Native Security; Shadow APIs; Threat Modeling;

1. Introduction

By 2022, APIs had become the backbone of digital transformation, enabling connectivity between microservices, cloud applications, mobile clients, and third-party platforms. A report from Akamai revealed that over 83% of global internet traffic involved API communication, indicating their widespread use in both internal and external systems [1]. As development practices evolve toward API-first methodologies, security concerns have intensified, especially given that APIs often expose business-critical functions and sensitive user data.

APIs offer speed, flexibility, and scalability—but they also broaden the attack surface. Their accessibility, especially in public-facing contexts, makes them attractive targets for adversaries. A 2022 report by Salt Security found that 94% of organizations had experienced at least one API-related security incident in the prior year [2]. Vulnerabilities such as broken object-level authorization, data overexposure, and improper authentication are increasingly exploited to compromise systems. These issues are exacerbated in complex environments where APIs proliferate rapidly without consistent governance, testing, or monitoring.

This paper examines how organizations can secure their APIs across the lifecycle—from design and development to deployment, monitoring, and retirement. It explores emerging threats, details technical implementation practices, and presents a real-world case study illustrating the importance of proactive API security.

2. The API Threat Landscape

2.1 Emerging Threat Patterns

APIs are highly flexible and customizable, but this flexibility can also introduce risk when security best practices are not enforced. The OWASP API Security Top 10, last updated in 2021, highlights the most prevalent and dangerous categories of API vulnerabilities [3]. These include broken object-level authorization (BOLA), which occurs when access controls fail to restrict users from accessing objects that do not belong to them, and excessive data exposure, where APIs return

* Corresponding author: Ishva Jitendrakumar Kanani

more information than necessary, leaving sensitive data unprotected. Other common threats include broken user authentication, lack of rate limiting, and security misconfigurations such as permissive cross-origin resource sharing (CORS) settings or verbose error messages.

Real-world exploitation of these vulnerabilities continued to rise in 2022. For example, Peloton exposed user account data via an insecure API endpoint lacking adequate authorization checks [4]. More significantly, the Optus breach in Australia affected nearly 10 million individuals and was directly traced to an unauthenticated API endpoint, demonstrating how even basic misconfigurations can lead to catastrophic data exposure [5]. T-Mobile, too, suffered repeated breaches tied to weak API controls and monitoring deficiencies [6].

As attackers shift focus from traditional web attacks to API-centric exploits, organizations must evolve their security postures to address this growing threat vector comprehensively.

3. Implementation Strategies for Securing APIs

3.1 Secure Authentication and Authorization

Effective authentication and authorization are foundational to API security. Standards such as OAuth 2.0 and OpenID Connect enable token-based authentication and allow for delegated access via access and refresh tokens [7]. When properly implemented, these frameworks allow APIs to validate user identities without storing credentials. Access tokens should be signed, time-bound, and context-aware, including fields like audience, issuer, and expiration time.

Role-based access control (RBAC) and attribute-based access control (ABAC) can enforce fine-grained policies by ensuring users can only access resources they are explicitly permitted to. For multi-tenant applications, tenant isolation must be strictly enforced through scoped tokens and consistent authorization validation.

3.2 Schema Enforcement and Input Validation

Defensive API design requires strict input validation to avoid injection, enumeration, and manipulation attacks. OpenAPI 3.0 specifications allow developers to define expected request and response formats, enabling schema enforcement at both gateway and application levels [9]. Strict JSON validation ensures that unexpected parameters or payloads are rejected early, while maximum content length and boundary checks help prevent denial-of-service (DoS) attacks.

3.3 Gateway-Level Protections

API gateways such as AWS API Gateway, Kong, Apigee, and Azure API Management provide centralized control points for security enforcement. Gateways can verify JWT tokens, enforce rate limits, throttle suspicious traffic, and manage CORS settings [10]. They can also apply mutual TLS (mTLS) for authenticating service-to-service communication in microservices architectures. Key rotation, versioning, and endpoint access logging are additional gateway features that bolster operational visibility.

3.4 Secure Development and Testing

Security must be integrated into the software development lifecycle (SDLC). Static code analysis tools like SonarQube and CodeQL help identify insecure code patterns at the source level [11]. Meanwhile, dynamic testing with OWASP ZAP, Postman Security, or Burp Suite allows teams to simulate real-world API attacks. Fuzzing tools like RESTler generate malformed or edge-case inputs to assess API robustness, while business logic flaws often require manual testing and threat modeling [13].

Secret management tools such as HashiCorp Vault and AWS Secrets Manager should be used to avoid hardcoded credentials and ensure key rotation policies are enforced [12].

3.5 Runtime Protection and Monitoring

In production, continuous monitoring is essential. Security information and event management (SIEM) platforms like Splunk, Datadog, and ELK stack can aggregate and analyze API logs to detect anomalies. Specialized API security platforms, including Salt Security, 42Crunch, and Noname Security, provide visibility into shadow APIs, monitor usage behavior, and trigger alerts for unusual patterns such as brute-force attacks or data scraping attempts [14].

Organizations must also be prepared to identify and deprecate outdated or undocumented APIs, known as zombie APIs, which can silently expose data or functionality if left unmanaged [15].

4. Case Study: The 2022 Optus API Breach

In September 2022, Optus experienced one of the most significant data breaches in Australian history when attackers exploited a publicly accessible API that required no authentication. Through unauthenticated HTTP requests, the attacker extracted personal data including names, addresses, phone numbers, dates of birth, and identity document numbers. The attack was unsophisticated yet effective—simply enumerating user identifiers in the request path allowed the attacker to scrape millions of records. The API lacked basic controls such as access authentication, rate limiting, and anomaly detection, making the extraction of data not only possible but efficient.

The incident revealed multiple systemic failures. Sensitive data was returned in plaintext and without filtering, violating principles of data minimization. Moreover, the exposed API was tied to a legacy system, indicating poor visibility and governance of aging infrastructure. Optus responded by disabling the endpoint, alerting regulators, and offering identity protection services to customers. The Australian government introduced emergency measures to allow banks to access compromised ID data, and proposed sweeping reforms to the Privacy Act. These reforms included greater penalties, mandatory data minimization, and expanded investigative powers for the Office of the Australian Information Commissioner (OAIC) [5][16].

The breach highlighted the real-world consequences of neglecting API security fundamentals. It emphasized the need for organizations to implement authentication rigorously, monitor all API traffic, enforce proper data handling, and maintain a comprehensive inventory of all exposed interfaces especially those tied to deprecated systems

5. Conclusion

APIs are now integral to digital business operations, powering everything from cloud-native applications and mobile platforms to partner integrations and internal microservices. However, as demonstrated by the Optus breach and numerous others in 2022, APIs also introduce a wide and often underprotected attack surface. The growing complexity of API ecosystems across multi-cloud environments, serverless platforms, and hybrid infrastructures demands that security teams evolve their strategies beyond perimeter defense.

Securing APIs requires a holistic, lifecycle-based approach that incorporates best practices such as robust authentication (e.g., OAuth 2.0), fine-grained access controls (RBAC/ABAC), schema validation, rate limiting, encryption, and runtime monitoring. Organizations must also prioritize continuous discovery and decommissioning of undocumented or deprecated APIs—known as “shadow” and “zombie” APIs which remain among the most exploited vulnerabilities.

Additionally, the Optus breach underscores the importance of data minimization, anomaly detection, and Zero Trust principles. Security must be embedded as a design requirement rather than an afterthought, supported by cross-functional collaboration between development, operations, and security teams. This requires ongoing investment in developer education, threat modeling, and security automation integrated within CI/CD pipelines.

Looking ahead, API security is poised to benefit from advances in AI-driven anomaly detection, behavioral traffic analysis, and automated policy enforcement. These innovations can reduce the burden on manual review and enable faster, more accurate identification of emerging threats. Furthermore, as regulatory pressure mounts globally, aligning API security controls with frameworks like OWASP API Top 10, NIST SP 800-207, and ISO/IEC 27001 will be critical for maintaining trust and compliance.

In summary, API security must evolve from isolated tools to a strategic pillar of enterprise cybersecurity. Organizations that proactively secure their APIs will not only reduce their breach risk but also enable safer innovation, faster development cycles, and stronger digital resilience in an increasingly interconnected world.

References

[1] Akamai. *State of the Internet: API Security*. 2022.

[2] Salt Security. *State of API Security Report*. Q1 2022.

[3] OWASP Foundation. *OWASP API Security Top 10 – 2021* [Internet]. 2021 [cited 2025 Jul 18]. Available from: https://owasp.org/www-project-api-security/

[4] TechCrunch. *Peloton API Bug Exposed User Data* [Internet]. April 2022 [cited 2025 Jul 18]. Available from: https://techcrunch.com/2022/04/peloton-api-bug/

[5] Australian Cyber Security Centre. *Optus Data Breach: Key Takeaways*. October 2022.

[6] T-Mobile. *SEC Breach Filing*. January 2023.

[7] Hardt D. *The OAuth 2.0 Authorization Framework*. IETF RFC 6749; 2012.

[8] National Institute of Standards and Technology (NIST). *Zero Trust Architecture*. NIST SP 800-207. Gaithersburg, MD: NIST; 2020.

[9] OpenAPI Initiative. *OpenAPI Specification v3.0*. 2022.

[10] Google Cloud. *API Gateway Documentation*. 2022.

[11] GitHub. *CodeQL: Semantic Code Analysis*. 2022.

[12] GitGuardian. *State of Secrets Sprawl Report*. 2022.

[13] Bishop Fox. *API Business Logic Testing: Why It Matters*. 2022.

[14] Noname Security. *The API Security Landscape*. Industry Report; 2022.

[15] Gartner. *Stop Attacks by Securing Unknown and Shadow APIs*. Market Guide; 2022.

[16] Australian Government. *Privacy Act Reforms Post-Optus Breach*. Legislative Proposal; 2022.