LEARNING MADE EASY

Noname Security Special Edition

Securing APIs

for dummies®
A Wiley Brand

Identify the top API threats

Understand how to secure your APIs

Address regulatory requirements

Brought to you by

noname

Lawrence Miller, CISSP

About Noname Security

Noname Security is the only company taking a complete, proactive approach to API security. Noname works with 20 percent of the Fortune 500 and covers the entire API security scope — discovery, posture management, runtime protection, and security testing. Noname Security is privately held and remote-first, with headquarters in Silicon Valley, California, and offices in Amsterdam.

Securing APIs

Noname Security Special Edition

by Lawrence Miller, CISSP

for dummies®
A Wiley Brand

Securing APIs For Dummies®, Noname Security Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN 978-1-394-21635-2 (pbk); ISBN 978-1-394-21637-6 (ebk)

For general information on our other products and services, or how to create a custom *For Dummies* book for your business or organization, please contact our Business Development Department in the U.S. at 877-409-4177, contact info@dummies.biz, or visit www.wiley.com/go/custompub. For information about licensing the *For Dummies* brand for products or services, contact BrandedRights&Licenses@Wiley.com.

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Project Editor: Elizabeth Kuball
Acquisitions Editor: Traci Martin
Editorial Manager: Rev Mengle
Senior Client Account Manager: Matt Cox
Production Editor: Tamilmani Varadharaj

Introduction

Application programming interfaces (APIs) are the backbone of today's app-driven world, establishing key conduits for working both inside the enterprise and with partners and customers. Simply put, APIs connect the world, from revenue-generating customer experiences to cost-saving backend integrations and everything in between.

However, the API attack surface is rapidly growing, and threat actors are increasingly targeting these critical byways. Securing APIs has become a top priority for security and risk professionals everywhere, as leading industry analysts predict that more than half of all data theft incidents will be attributed to insecure APIs by 2025, and 78 percent of enterprises have suffered an API security incident in the past 12 months according to survey findings in *The API Security Disconnect – API Security Trends in 2023.*

About This Book

Securing APIs For Dummies, Noname Security Special Edition, consists of five chapters that explore the following:

- The rapidly growing API attack surface (Chapter 1)
- Limitations in existing API security tools (Chapter 2)
- Critical capabilities needed to secure APIs (Chapter 3)
- Top API security risks (Chapter 4)
- Ten keys to securing APIs (Chapter 5)

Each chapter is written to stand on its own, so if you see a topic that piques your interest, feel free to jump ahead to that chapter. You can read this book in any order that suits you.

Foolish Assumptions

It's been said that most assumptions have outlived their uselessness, but I assume a few things nonetheless!

Mainly, I assume that you're an IT or security executive, architect, manager, engineer, or developer. Perhaps you're a DevOps manager, an application security professional, or even an application developer interested in how you can better secure your organization's business-critical APIs. As such, you recognize the critical nature of APIs and their potential exposure to cyberattacks and threat actors.

If any of these assumptions describes you, then this is the book for you! If none of these assumptions describes you, keep reading anyway — it's a great book, and after reading it, you'll know quite a bit about securing your APIs.

Icons Used in This Book

Throughout this book, I occasionally use special icons to call attention to important information. Here's what to expect:

REMEMBER

This icon points out important information you should commit to your nonvolatile memory, your gray matter, or your noggin.

TECHNICAL STUFF

This icon explains the jargon beneath the jargon and is the stuff legends — well, legendary nerds — are made of.

TIP

Tips are appreciated, but never expected, and I sure hope you appreciate these useful nuggets of information.

Beyond the Book

There's only so much I can cover in this short book, so if you find yourself at the end of it wondering, "Where can I learn more?," go to `https://nonamesecurity.com`.

IN THIS CHAPTER

- Defining APIs
- Recognizing the top threats to API security
- Ensuring compliance with regulatory requirements

Chapter 1
Addressing the Growing API Attack Surface

Application programming interfaces (APIs) have become the default connectivity and data exchange method of modern application environments. With that in mind, securing APIs is paramount to securely operating in our digital-first world. In this chapter, I start by defining APIs and what they do. Then I explore the top API security threats and address regulatory requirements.

What Is an API?

An API is a software interface, written using a programming language of choice, that allows two other disparate software components to communicate with each other. These two other software components can also be written in any language of choice, making the concept of APIs really flexible and powerful. APIs help make applications and digital services easier to consume, and

they make it easier for developers to build, enhance, and maintain applications. APIs do two things (see Figure 1-1). They allow people to build applications that:

- Communicate with existing applications and services
- Perform certain actions on data

With APIs, you can build applications that automatically update without requiring any manual work. You can also empower users to interact with existing applications and services in a more efficient way. This increases developer productivity by allowing them to focus on the functionality of their applications rather than on the different software components.

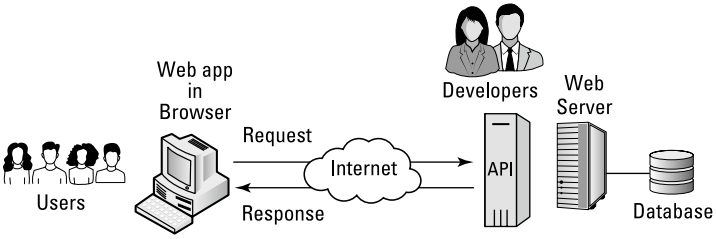
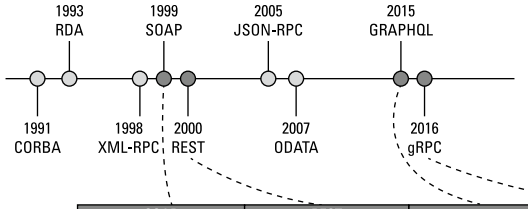


FIGURE 1-1: APIs allow applications to communicate with other applications and services and perform certain actions on data.

There are different types of APIs, some of which are used for communication between microservices. These types include Simple Object Access Protocol (SOAP), Representational State Transfer (RESTful), and Graph Query Language (GraphQL) APIs (see Figure 1-2). Some APIs are intended to manipulate data, such as create, read, update, delete (CRUD) APIs.

APIs can be written in practically any programming language (such as Java, Go, and C#), and many API standards exist that use Extensible Markup Language (XML), JavaScript Object Notation (JSON), and so on as a data protocol, making it possible to seamlessly transmit data between disparate systems.



	SOAP (Simple Object Access Protocol)	REST (Representational State Transfer)	GraphQL	RPC (Remote Procedure Call)
Organized in terms of	enveloped message structure	compliance with six architectural constraints	schema and type system	local procedure call
Format	XML only	XML, JSON, HTML, plain text	JSON	JSON, XML, Protobuf, Thrift, FlatBuffers
Learning curve	Difficult	Easy	Medium	Easy
Community	Small	Large	Growing	Large
Use cases	– payment gateways – identity management – CRM solutions – financial and telecommunication services – legacy system support	– public APIs – simple resource-driven apps	– mobile APIs – complex systems – micro-services	– command and action-oriented APIs – high performance communication in massive micro-services systems

FIGURE 1-2: A comparison of API architectural styles.

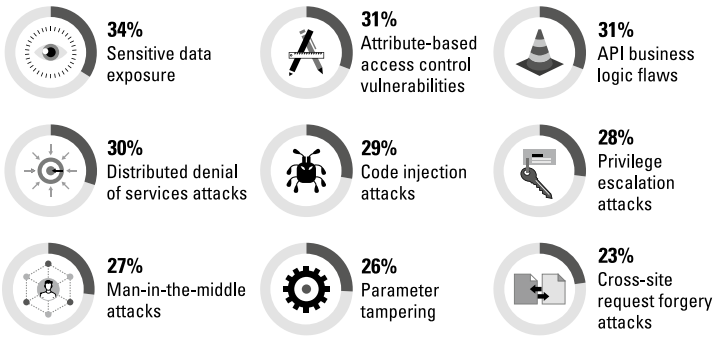
Identifying the Top API Threats

APIs are specifically designed to provide access to software services that may contain sensitive data. Unfortunately, many organizations' API ecosystems are fraught with vulnerabilities and, too often, enterprises only become aware after a breach has already occurred. API security is complex, and the threats, vulnerabilities, and risks associated with APIs can be daunting for any organization.

The Open Worldwide Application Security Project (OWASP) API Security Top Ten (see Chapter 4) is composed of the top ten vulnerabilities in APIs — but there are many more vulnerabilities and threats that organizations need to address in their APIs. In fact, a "less severe" (that is, not "top ten") vulnerability that is actively being exploited by a threat actor can be a far greater risk to your organization than a "more severe" vulnerability that is not being actively exploited.

According to Enterprise Strategy Group (ESG), 92 percent of cybersecurity professionals have experienced at least one API security-related incident in the last 12 months. Addressing API

vulnerabilities — including sensitive data exposure, attribute-based access control vulnerabilities, and API business logic flows, among others (see Figure 1-3) is a top priority.



Source: Enterprise Strategy Group, "Securing the API Attack Surface," May 2023

FIGURE 1-3: Types of API vulnerabilities that are of greatest concern.

Addressing Regulatory Requirements

Data security and privacy regulations continue to raise the compliance bar for organizations worldwide. The omnipresent threat of costly fines, reputational damage, and lost revenue has motivated companies to address sensitive data protection much more proactively.

As if the existing challenges weren't enough, regulated industries must now factor APIs into their compliance programs. Organizations must provide a comprehensive risk assessment and implement controls to help ensure that data is protected and stored securely. APIs must be a part of the risk assessment process. Regulated organizations must determine what security controls are required, and then be able to demonstrate how possible vulnerabilities and associated risks will be identified and addressed directly.

REMEMBER

API compliance refers to an organization's ability to meet the requirements defined by data security and privacy regulators in a given industry or jurisdiction. Compliance requirements must be strictly adhered to for the protection of your customers', partners', and employees' data and other sensitive and private information.

The regulatory landscape is rapidly evolving as new data security and privacy regulatory requirements are enacted across different jurisdictions at state, national, regional, and multinational levels. Some of the most relevant regulations for API security include the following:

- **U.S. Health Insurance and Portability and Accountability Act (HIPAA):** HIPAA includes provisions that mandate the adoption of federal privacy protections for protected health information (PHI). The most typical HIPAA violation is the failure to properly secure and encrypt data.
- **European Union General Data Protection Regulation (GDPR):** The GDPR has been critical as a unifying approach to European data privacy because it has driven greater protection for European residents by spelling out, with high clarity, what organizations that process personal data must do to safeguard that data to protect those rights.
- **California Consumer Privacy Act (CCPA):** The CCPA establishes and further enhances California residents' consumer privacy rights and protections. The CCPA provides consumers more control over the personal information businesses collect.
- **Payment Card Industry Data Security Standard (PCI DSS):** PCI DSS provides a baseline of technical and operational requirements designed to protect payment data. PCI DSS v4.0, published in April 2022, is the latest iteration. Organizations covered by PCI DSS must comply with the new standards by March 31, 2024. A key addition to the new standard is the inclusion of APIs. Requirement 6 ("Develop and maintain secure systems and software") specifically calls out APIs in the following subsections:
  - *6.2.2:* This requirement covers training for application developers working on bespoke (that is, applications developed by a third-party vendor, but not standard commercial off-the-shelf [COTS] applications) and custom software. It states that developers need to be trained on security topics relevant to their job function, including secure software design and secure coding techniques, at least annually.

- *6.2.3:* This requirement addresses the need to review your bespoke custom application code to ensure known vulnerabilities are not released into production. Organizations must also verify that software uses external components' functions (such as libraries, frameworks, APIs, and so on) in a secure manner.
- *6.2.4:* This requirement touches on using software engineering techniques or other methods, to prevent or mitigate common software attacks and related vulnerabilities in bespoke and custom software. The attacks called out in this requirement range from relatively straightforward injection-based attacks to more sophisticated API business logic–based attacks.
- *6.3.2:* This requirement applies to maintaining an inventory of bespoke and custom software, and third-party software components incorporated into bespoke and custom software, to facilitate vulnerability and patch management.

IN THIS CHAPTER

- Understanding what a WAF does
- Defining API management and gateway functions
- Looking at the limitations of WAFs and API management gateways

Chapter 2
Recognizing the Limitations of Existing Tools

Application programming interfaces (APIs) aren't exactly "new" technology, but the tremendous API growth in numbers, data volume, and ubiquity, as well as the increased sophistication of API functionality, can overwhelm enterprise application security teams. As a result, there are significant gaps in API observability capabilities and the application of API-specific security controls.

Contributing to the lack of existing API security proficiency and maturity is the distributed nature of many modern APIs. API security controls are distributed between the delivery technology stack that includes web application firewalls (WAFs), API management, and API gateways.

In this chapter, you learn about existing tools that are used to enforce API security policies and controls and their limitations.

What Is a WAF?

WAFs have become part of the core stack for application and API protection. WAFs are proxy-based tools, designed specifically for web applications, that inspect incoming web and API requests (headers and payloads) for malicious or unwanted traffic.

REMEMBER

The basic function of a WAF is to provide an application layer filter for web and API traffic. This filter looks for malicious/unwanted content within incoming requests and ensures that only approved actions, defined in policies, can be performed.

WAFs are utilized to provide rudimentary protections for applications and APIs. They're fairly proficient at detecting known attacks (with signatures) and malicious scripts, but they lack context for more sophisticated multistep API attacks. Whereas most web-based attacks, such as injection attacks, are "one and done," most API-based attacks are "low and slow," focused on manipulating API flows without being detected to essentially trick the backend systems. Advanced WAFs add anti-automation capabilities to prevent threat actors from automating attack techniques such as breached credential testing, brute force attacks, and account lockout (denial-of-service) attacks. A WAF can only apply policy to traffic that passes through it.

What Is an API Gateway?

The primary functions of API management platforms and gateways are to publish APIs, ensure API availability, monitor usage, and enforce access controls. API management platforms and gateways play a very important role in the delivery of APIs. Each plays a defined role, and they're tightly linked together.

API management platforms operate in the control plane (management and policy) and are usually delivered as a portal service where developers and API managers can "check in" their APIs when they're ready to be in production. API management is used to manage and monitor the operations of the API.

The API gateway operates in the data plane (proxy with policy enforcement), serving as an access control point in front of an API endpoint. The API gateway provides core functionality to ensure

the API is available to its intended consumers. The API gateway also is a control point for the API management policies such as access controls and usage (for example, rate limiting and quotas).


TIP

Routing traffic through an API gateway is a best practice, especially for open APIs (exposed to the internet). However, not all APIs sit behind a gateway, so they don't benefit from the controls and visibility provided by gateway and management functions.

Identifying Gaps in Existing API Security Tools

Both WAFs and API gateways are important components of the API delivery stack, but neither is designed to provide the security controls and observability required to adequately protect APIs. For example, broken object level authorization (BOLA) attacks look like "ordinary" API traffic to WAFs and gateways, enabling them to pass through these controls undetected.

Key limitations in these tools include the following:

- **Limited observability:** Both WAFs and API gateways can only observe API traffic that is routed through them. Leading industry analysts predict that 50 percent of enterprise APIs will be "unmanaged" by 2025, which means that observability will be limited at best. Some unmanaged APIs are deployed intentionally, but others may be unknown "shadow" or "zombie" APIs that could be putting the organization at risk. Even if all APIs are routed through WAFs and gateways, most enterprise organizations will only have fragmented views of their API estate that could span across multiple teams or business units.
- **Incomplete inventory:** Simply knowing the number of APIs within the organization is not very useful for security and IT teams. An accurate inventory needs to include contextual API data that includes data types handled, authentication controls, configurations, traffic mappings, routing details, exposure to the internet, and all other relevant metadata. Neither WAFs nor API gateways can provide an aggregated and current inventory of the full API estate.

- **Inadequate security posture management analysis:** Without full context-aware visibility of the API estate, the combination of WAFs and API gateways simply cannot provide detailed analysis of the API security posture. Posture management analysis helps IT teams to efficiently identify and resolve misconfigurations that could result in security risk or compliance violations. Misconfigurations, for example, could include inadequate authentication, unnecessary exposure (to the internet), lack of rate limiting, or encryption — just to name a few.
- **Lack of API-specific runtime security controls:** The combination of WAFs and gateways provides basic API security controls: WAFs apply signature-based attack detection and identify appropriate user-based session behavior, and gateways can enforce rate limiting and authentication controls. However, these controls alone are not enough to adequately protect the business from API-specific attacks and abuse. For example, BOLA attacks look like "ordinary" API traffic to WAFs and gateways, enabling them to pass through these controls undetected. WAFs and gateways lack contextual awareness between API requests and responses. This gap can leave APIs vulnerable not only to BOLA exploits, but other attacks and business logic abuse that simply can't be easily identified using standard WAF and gateway controls.
- **Need for policy:** Both WAFs and gateways require defined security policies to implement desired controls and enforcement actions. This requires a strong understanding of the different threats and vulnerabilities that your APIs are exposed to, and continuous evaluation and updating of policies to address the latest threats.


TIP

Organizations need additional API-specific controls that uncover blind spots and attack paths open to threat actors, and mitigate threats in real time. A complete API security platform, encompassing API discovery, posture management, runtime protection, and API security testing, has become essential. Chapter 3 outlines the critical capabilities necessary to secure your APIs.

IN THIS CHAPTER

- Starting with access control
- Knowing what you're protecting
- Ensuring API runtime security
- Testing your APIs
- Managing your API security posture

Chapter 3
Defining Critical Capabilities for Securing APIs

In this chapter, I explain the key capabilities and features you need in your API security tools.

Access Control

Gaining access to a target environment is a key objective for attackers. Skilled threat actors may try to exploit a vulnerability in your identity and access management (IAM) platform, but, more often than not, attackers simply log into systems using stolen or compromised credentials.

TIP

To help protect your APIs from unauthorized access, look for the following access control capabilities:

- **Authentication:** The process of verifying the identity of a user or device that attempts to access a system, service, or network. Authentication is an important part of any application and can be done in many ways, such as username and password authentication, two-factor or multifactor authentication (2FA or MFA, respectively), and API authentication. API authentication uses an API key to verify the identity of the user. This type of authentication can be used for both public and private APIs.
- **Authorization:** API authorization is the process of checking the identity of the user and authorizing them to access the application. Table 3-1 shows different API authorization methods and their associated characteristics.
- **Accountability:** An API access control platform must also provide robust monitoring, alerting, and logging capabilities to ensure the pertinent details of all access control activities are available for incident response, forensic investigation, and root cause analysis (RCA).

TABLE 3-1 API Authorization Methods

Method	Security	Complexity	Scalability
Basic authentication	Low	Low	Low
API keys	Medium	Medium	Medium
Open Authorization (OAuth) 2.0	High	High	High
JavaScript Object Notation (JSON) Web Tokens (JWTs)	High	Medium	High
Security Assertion Markup Language (SAML) 2.0	High	High	Low

Discovery

As the saying goes, "You can't protect what you can't see." If your organization is among the 74 percent of organizations that do not have a complete API inventory or know which of their APIs return sensitive data (according to a recent study conducted by Noname Security), you have limited visibility at best. Many of

these "rogue" APIs are almost certainly unprotected or under-protected. Why? Enterprises manage thousands of APIs, many of which aren't properly routed through a proxy such as an API gateway or web application firewall (WAF).

TIP

An API discovery tool should incorporate the following capabilities and features:

- **API asset discovery and granular inventory:** An API discovery tool must be able to identify all the APIs you have, regardless of configuration or type — including Representational State Transfer (RESTful), Graph Query Language (GraphQL), Simple Object Access Protocol (SOAP), Extensible Markup Language–Remote Procedure Call (XML-RPC), and JavaScript Object Notation–Remote Procedure Call (JSON-RPC). It should also create an inventory that is updated automatically to prevent it from getting stale, and provide the ability to search, tag, filter, assign, and export APIs based on any attribute.
- **Detection of dormant, legacy, and zombie APIs:** Legacy and zombie APIs can predate your organization's API security initiatives. These APIs typically lack ownership and function without any visibility or security controls, just waiting to be exploited.
- **Shadow domain discovery:** In addition to shadow APIs, you can have entire *shadow domains* (API domain names that you know nothing about). API discovery tools must identify forgotten, neglected, or otherwise unknown shadow domains that may pose a security risk.
- **Automatic scans:** Scanning is essential to eliminate blind spots and identify critical issues, including leaked API keys and credentials, API code and schema exposure, infrastructure misconfigurations, vulnerabilities in documentation, GitHub repositories, Postman workspaces, and so on.

Runtime Protection

API runtime protection is the process of securing APIs as they operate and manage requests during their normal operations. After posture management, runtime protection is a critical way to protect your API estate against bugs and misconfigurations that slip into production. Runtime protection identifies unusual

patterns and anomalies in API use and data access, so ongoing attacks that may otherwise slip under the radar can be identified and remediated before thousands or millions of data records have been compromised.

TIP

An API runtime protection tool should include the following capabilities and features:

- **Out-of-band monitoring in real time:** API security monitoring shouldn't impact, slow down, or add latency to API traffic. It should run completely out of band with no required network changes. Runtime protection tools should mirror traffic from identified data sources and perform analysis on that traffic data in the background with real-time alerting of any issues discovered.
- **API anomaly and exploitation detection:** Passive data collection is not enough, especially as the number of APIs and the total volume of API traffic continues to scale. API activity must be analyzed continuously to detect anomalous events and alert security and operations teams. State-of-the-art platform tools incorporate artificial intelligence (AI) and machine learning capabilities to analyze traffic in real-time and leverage contextual insights into data leakage, data tampering, data policy violations, suspicious behavior, and API security attacks.
- **API attack remediation:** After an anomaly or other problem has been identified and an alert has been generated, time is of the essence. Unauthorized movement of sensitive data via API or other suspected misuse of APIs must be detected and blocked. Runtime protection blocks misuse through integration with your existing firewalls and API gateways, and partially or fully automates remediation.
- **Integrations for incident response:** The best API security tools integrate with existing security systems. When an incident occurs, runtime protection tools must include the necessary integrations to ensure remediation tasks are assigned to appropriate teams. If misconfigurations, data policy violations, or suspicious behaviors are detected, they should be reported to the API gateway; security information and event management (SIEM) system; security orchestration, automation, and response (SOAR) platforms; and other information security engines to ensure the right level of awareness. As a general rule, runtime protection tools should integrate easily with the other security, monitoring, and management tools your organization uses.

Testing

API security testing refers to the process of assessing and evaluating the security measures implemented in an API to ensure that it's protected against potential vulnerabilities, threats, and attacks. It involves conducting comprehensive tests to identify weaknesses in authentication mechanisms, authorization controls, data integrity, encryption protocols, input validation, error handling, and other security-related aspects of an API.

TIP

Common API security testing platforms include the following types:

- **Static analysis security testing (SAST):** SAST is a software testing technique that uses static analysis to find security vulnerabilities in the source code of the software without executing it. It can be used to detect programming errors, design flaws, and security vulnerabilities in the source code of a program or system.
- **Dynamic application security testing (DAST):** DAST testing is different from SAST in that API testing takes place in production. Testers identify problems that occur during use and then trace them back to their origins in the software design, instead of detecting issues linked to a code module.
- **Software composition analysis (SCA):** SCA is a software engineering technique that helps identify the software components and their relationships. It can be used for analyzing the design of an application, identifying *code smells* (that is, an indicator of a deeper problem), or finding out how much code is needed for a given task.

Posture Management

API posture management provides the tools to manage, monitor, and maintain the security of your APIs throughout the API life cycle. It combines API discovery with sensitive data identification and vulnerability detection, so your remediation efforts focus on the most critical APIs first.

Good posture management provides visibility into all activity around your APIs so you can enforce security policies, ensure compliance with regulations, and audit changes to your API ecosystem. It protects and secures your APIs against malicious attacks, unauthorized users, and data breaches — any one of which can lead to significant reputational damage, loss of business, and regulatory penalties.

Implementing posture management best practices minimizes the API attack surface and mitigates much of your API risk. Thorough inventories of APIs and sensitive data stores are essential best practices for good posture management.

TIP

A robust API posture management platform should include the following capabilities and features:

- **Sensitive data classification:** An API that supplies weather data from public sources is of much less concern than one that transmits credit card information. API posture management tools should be able to quickly identify how many APIs are able to access credit card data, phone numbers, Social Security numbers (SSNs), and other sensitive data, along with the number of users who have accessed sensitive data via your APIs.
- **Configuration assessment:** Many cyberattacks gain entry as the result of simple misconfiguration of the networks, API gateways, or firewalls that broker and protect API traffic. Strong posture management requires the ability to scan infrastructure and software configurations regularly, including log files and configuration files. Regular scanning helps uncover misconfigurations and vulnerabilities and identifies risks created by configuration drift.
- **Auto-generated API specifications:** API specifications tell the consumers of an API what it does and how to use it. Secure APIs must be evaluated for compliance against specifications and accurately documented. Poor or nonexistent specifications make security testing more difficult, increasing the risk that an API reaches production with an undetected vulnerability.

IN THIS CHAPTER

- Introducing OWASP
- Getting familiar with the OWASP Risk Rating Methodology
- Unpacking the top ten API security risks

Chapter 4
Understanding the OWASP API Security Top Ten Risks

This chapter takes a deep dive into the Open Worldwide Application Security Project (OWASP) API Security Top Ten risks. You'll learn about the OWASP Foundation, the OWASP Risk Rating Methodology, and the risks that comprise the 2023 OWASP API Security Top Ten.

What Is OWASP?

The OWASP Foundation is a global, nonprofit organization dedicated to improving software security through open-source initiatives and community education, tools, and collaboration.

OWASP projects are a collection of related tasks that have a defined road map and team members. All projects are open source and

built by volunteers. OWASP currently has more than 100 active projects organized into the following categories:

- **Flagship projects:** The OWASP Flagship designation is given to projects that have demonstrated strategic value to OWASP and application security as a whole.
- **Production projects:** OWASP Production projects are production-ready projects.
- **Other projects:** These include Lab and Incubator projects.

The OWASP Top Ten is perhaps the most well-known project published by OWASP. It's a reference standard for the most critical web application security risks. Similar to the OWASP Top Ten, the OWASP API Security Top Ten is a reference standard that focuses specifically on API security risks.

The OWASP Risk Rating Methodology

The OWASP API Security Top Ten is an excellent starting point to help organizations identify the most critical threats to their API footprint.

OWASP classifies each API security threat by four criteria:

- **Exploitability:** How easy is this vulnerability to exploit/attack?
- **Weakness prevalence:** How common is this vulnerability?
- **Weakness detectability:** How easy would it be to detect this vulnerability?
- **Technical impact:** What is the technical impact of this vulnerability?

Each factor is given a score from 1 to 3, with 3 being the most severe. A vulnerability that is easy to exploit, widespread, and easily detectable with severe technical impact is the most urgent to address. These dimensions allow API security risks to be force-ranked in terms of severity.

REMEMBER

The OWASP 2023 API Security Top Ten is based on publicly available data about API security incidents. This data was collected from bug bounty platforms and publicly available reports. For the 2023 update, only issues reported between 2019 and 2022 were considered.

The Top API Security Risks

The following sections look at the OWASP API Security Top Ten (2023) risks and how to remediate or mitigate them.

Broken object level authorization

APIs often expose endpoints that handle object identifiers (unique values that describe objects), resulting in a wide attack surface in access-level control. As a result, attackers may be able to read, update, delete, or create data objects without permission by exploiting broken object level authorization (BOLA) vulnerabilities. Thus, an attacker may be able to forge a ticket that grants them unauthorized permissions, analogous to forging a permission slip to go on a field trip in school. Individual authorization checks are necessary for every function that interacts with a data source.

TIP

To address BOLA vulnerabilities in your APIs, do the following:

- Use an authorization mechanism to check if the logged-in user has access to perform the requested action on the record in every function that uses an input from the client to access a record in the database.
- Prefer the use of random and unpredictable values (that is, not easy to guess or infer) as universally unique identifiers (UUIDs) for records' IDs.
- Write tests to evaluate the vulnerability of the authorization mechanism. Do not deploy changes that make the tests fail.

Broken authentication

Authentication is more complex to implement for APIs because multifactor authentication (MFA) and other human-driven authentication options aren't possible. Also, authentication

mechanisms are often implemented incorrectly, allowing attackers to compromise authentication tokens or exploit implementation flaws to take over other users' identities temporarily or even permanently. Thus, an attacker can impersonate an authorized user or object to gain access to an API. Compromising the ability to identify the user compromises overall API security.


TIP

To address broken authentication vulnerabilities, do the following:

- Make sure you understand all the possible flows to authenticate to the API (mobile/web/deep links that implement one-click authentication and so on).
- Implement anti–brute force mechanisms to mitigate credential stuffing, dictionary attacks, and brute force attacks on your authentication endpoints. This mechanism should be stricter than the regular rate-limiting mechanisms on your APIs.
- Avoid using API keys for user authentication. They should only be used for API client authentication.

Broken object property level authorization

A typical broken object property level authorization (BOPLA) exploit would involve API endpoints that return all object's properties in the API response, thereby exposing potential sensitive information. After these (sometimes hidden) properties are obtained, a threat actor can craft an API request to try to change them. Getting access to these private object properties could lead to sensitive data disclosure, data loss, or data corruption.


TIP

To address BOPLA vulnerabilities, do the following:

- Avoid generic methods such as `to_json()` and `to_string()`. Instead, cherry-pick specific object properties you want to return.
- If possible, avoid using functions that automatically bind a client's input into code variables, internal objects, or object properties (known as *mass assignment*).
- Implement a schema-based response validation mechanism as an extra layer of security. As part of this mechanism, define and enforce the data returned by all API methods.

Unrestricted resource consumption

Satisfying API requests requires resources such as network bandwidth, central processing unit (CPU), memory, and storage. In many cases, APIs don't restrict the size or number of resources that can be requested by the client in a session or a certain period of time (that is, *rate limiting*). This means users can make huge requests, either inadvertently or maliciously. This can negatively impact API server performance and expose your APIs to authentication flaws, such as brute force attacks.

TIP

To address unrestricted resource consumption vulnerabilities, do the following:

- Define and enforce a maximum size of data on all incoming parameters and payloads, such as maximum length for strings, maximum number of elements in arrays, and maximum upload file size (regardless of whether it's stored locally or in the cloud).
- Implement a limit on how often a client can interact with the API within a defined time frame (rate limiting).
- Add proper server-side validation for query string and request body parameters, specifically the one that controls the number of records to be returned in the response.

Broken function level authorization

Authorization flaws tend to occur in complex access control policies with hierarchies, roles, and groups, and an unclear distinction between administrative and regular functions. An attacker can gain access to other users' resources or admin functions by exploiting these weaknesses.

TIP

To address broken function level authorization vulnerabilities, do the following:

- Review your API endpoints against function level authorization flaws, while keeping in mind the business logic of the application and groups hierarchy.
- Make sure all your administrative controllers inherit from an administrative abstract controller that implements authorization checks based on the user's group/role.

- Make sure administrative functions inside a regular controller implement authorization checks based on the user's group and role.

Unrestricted access to sensitive business flows

Unrestricted access to sensitive business flows is mostly concerned with automated bot-type attacks misusing the normal process flow of the API and the connected backend systems.


TIP

To address unrestricted access to sensitive business flow vulnerabilities, do the following:

- Identify the business flows that could harm the business if they're excessively used, and choose appropriate protection mechanisms to mitigate the business risk.
- Analyze the user flow to detect nonhuman patterns (for example, the user accessed the "add to cart" and "complete purchase" functions in less than 1 second).
- Consider blocking IP addresses of Tor exit nodes and well-known proxies.

Server-side request forgery

Server-side request forgery (SSRF) flaws occur when an API fetches a remote resource without validating the user-supplied Uniform Resource Locator (URL). It enables an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall or virtual private network (VPN).


TIP

To address SSRF vulnerabilities, do the following:

- Isolate the resource-fetching mechanism in your network. These features usually aim to retrieve remote resources, not internal ones.
- Whenever possible, use allow lists of remote origins from which users are expected to download resources (for example, Google Drive, Gravatar, and so on), URL schemes and ports, and accepted media types for a given functionality.
- Disable Hypertext Transfer Protocol (HTTP) redirections.

Security misconfiguration

Security misconfiguration is just that: a misconfiguration in the security parameters of the API, often found in relation to open cloud storage, insecure default configurations, incomplete or ad hoc configurations, misconfigured HTTP headers, unnecessary HTTP methods, permissive cross-origin resource sharing (CORS), and/or verbose error messages containing sensitive information.

TIP

To address security misconfiguration risks, do the following:

- Ensure that all API communications from the client to the API server and any downstream/upstream components happen over an encrypted communication channel (that is Transport Layer Security [TLS]), regardless of whether it's an internal or public-facing API.
- APIs expecting to be accessed from browser-based clients (for example, web app front end) should implement a proper CORS policy, include applicable security headers, and restrict incoming content types/data formats to those that meet the business/functional requirements.
- Where applicable, define and enforce all API response payload schemata, including error responses, to prevent exception traces and other valuable information from being sent back to attackers.

Improper inventory management

An organization will typically create multiple APIs and API versions over time. It's critical that life-cycle management is applied to these APIs and proper security controls are implemented, or APIs are sunsetted when no longer in use.

TIP

To address improper inventory management risks, do the following:

- Inventory all API hosts and document important aspects of each of them, focusing on the API environment (that is, production, staging, testing, development), who should have network access to the host (that is, public, internal, partners), and the API version.

- Inventory integrated services and document important aspects, such as their role in the system, what data is exchanged (data flow), and their sensitivity.
- Document all aspects of your API, such as authentication, errors, redirects, rate limiting, COR policy, and endpoints, including their parameters, requests, and responses.

Unsafe consumption of APIs

Developers tend to trust data received from third-party APIs more than user input. This is especially true for APIs offered by well-known companies. Because of that, developers tend to adopt weaker security standards, for instance, in terms of input validation and sanitization.


TIP

To address risks associated with unsafe consumption of APIs, do the following:

- Ensure all API interactions happen over a secure communication channel (that is, TLS).
- Always validate and properly sanitize data received from integrated APIs before using it.
- Maintain an allow list of well-known locations. Integrated APIs may redirect yours, too — don't blindly follow redirects.


TIP

Download the *2023 OWASP API Security Top Ten Best Practices Guide* at `https://nonamesecurity.com` to learn more about the OWASP API Security Top Ten and how to remediate these vulnerabilities.

IN THIS CHAPTER

- Identifying business and technical stakeholder requirements
- Addressing API security across the API lifecycle
- Introducing security requirements and testing early in the software development lifecycle
- Extending security requirements and testing beyond production release
- Ensuring effective incident response and a healthy security ecosystem

Chapter 5
Ten Keys to API Security in Practice

Here are ten key considerations to help you improve your organization's application programming interface (API) security posture:

- **Business stakeholders:** Your business stakeholders should minimally include the following representatives:
  - **Executive leadership:** Executive leadership, including C-level executives and senior management, are responsible for setting strategic goals and allocating resources for API security initiatives. They provide guidance on establishing policies, procedures, and budgets to support robust API security measures across the organization.
  - **Business analysts (or product owners):** Business analysts have valuable insights into your business requirements related to the functionality of APIs and the integration

needs with external partners or third-party services. Engaging them helps identify potential risks associated with specific features or integrations early on during the development process.

- **Compliance officers:** Compliance officers ensure that APIs adhere to relevant industry standards, regulations, and privacy laws such as the General Data Protection Regulation (GDPR) and Payment Card Industry Data Security Standard (PCI DSS). They collaborate with the IT security team to ensure that appropriate safeguards are implemented within API design and operation processes.
- **Legal team:** Your legal team plays a critical role in ensuring that the organization's API initiatives comply with contractual obligations, intellectual property rights, and other legal aspects related to data protection.

» **Technical stakeholders:** Your technical stakeholders should include the following team members:

- **App developers:** An application software developer's primary responsibility is to design software systems and applications in accordance with the needs of their customers or employer. In addition, they run tests, alter test programs, and provide fixes for any flaws or malfunctions that occur. They collaborate closely with other application software developers, app testers, and software engineers.
- **Application architects:** Planning, designing, and managing groups of applications are responsibilities that architects frequently take on. They're also responsible for interactions between users and business units. Architects must have strong communication and negotiation skills so they can cooperate with everyone from software project managers to end users. In most cases, this also entails maintaining documentation, creating best practices for application development, and organizing staff training.
- **Application security:** Application security seeks to defend data and software application code from online dangers. Application security may and should be used throughout the whole development process, including design, development, and deployment. Therefore, people who work in AppSec are finding ways to improve the security posture in some capacity.

- **Network and infrastructure engineers:** The network of an organization must be planned, implemented, and secured by a network infrastructure engineer. To verify that the network site of the company has the right specifications, this person conducts site audits and surveys. They implement automated security scans at different stages of development to detect vulnerabilities early on, before pushing code into production environments.

- **Development:** To improve your organization's API security posture, several API design and development considerations need to be addressed. First and foremost, implementing strong authentication, such as OAuth 2.0 or two-factor authentication (2FA), helps protect APIs from unauthorized access. Another important design consideration is properly managing access controls to ensure that only authorized entities or roles can invoke specific API functions. This helps prevent unauthorized users from accessing sensitive data or functionalities exposed through the API endpoints.
- **Testing:** Conduct comprehensive security testing during the development life cycle of your APIs, including vulnerability assessments, penetration testing, and code reviews by experienced security professionals. Regularly update dependencies and libraries used within your API ecosystem to protect against known and newly discovered vulnerabilities.
- **Deployment:** Organizations should deploy their APIs to dedicated servers or cloud platforms with robust security controls, such as firewalls, intrusion detection systems, and regular vulnerability assessments.
- **Production:** Ensure that all components used in the API production environment — including operating systems, web servers, databases, frameworks, and libraries — are kept up to date with the latest patches and security fixes. This helps address known vulnerabilities and ensures that you're using the most secure versions of these components. Ongoing training and education are vital for maintaining an improved API security posture in production. Provide regular training sessions to developers, system administrators, and other stakeholders involved with the production environment on best practices for secure coding.

- **Security shift-left:** Shift-left testing saves time and reduces costs by identifying bugs and rigorously detecting errors and bottlenecks earlier in the API life cycle so technical stakeholders can improve initial designs and develop alternatives. The shift-left approach ensures quality by allowing the development team to bake automation tools like API testing or unit testing right into the building process.
- **Security shift-right:** Shift-right testing ensures real-world software stability and performance by testing in production environments, with scenarios that aren't possible in the development environment, and improving user experience by collecting feedback and reviews from application users. Conduct regular vulnerability assessments, penetration testing, and code reviews of your deployed APIs. These activities help identify and remediate security weaknesses or vulnerabilities that may have been missed during the development phase.
- **Incident response:** Implement a robust incident response plan that outlines procedures, roles, and responsibilities in the event of a security breach or other incidents. Test this plan regularly to ensure its effectiveness and train relevant personnel in their roles during such events.
- **Security ecosystem:** Continuous monitoring plays a critical role in maintaining an improved API security posture post-deployment. Monitor server logs regularly for any suspicious activities or signs of unauthorized access attempts. Employ security information and event management (SIEM) tools to aggregate and analyze log data from different sources, as well as IT service management (ITSM) integrations for incident response.

noname

Purpose-built API Security Testing

Deliver Secure APIs Faster

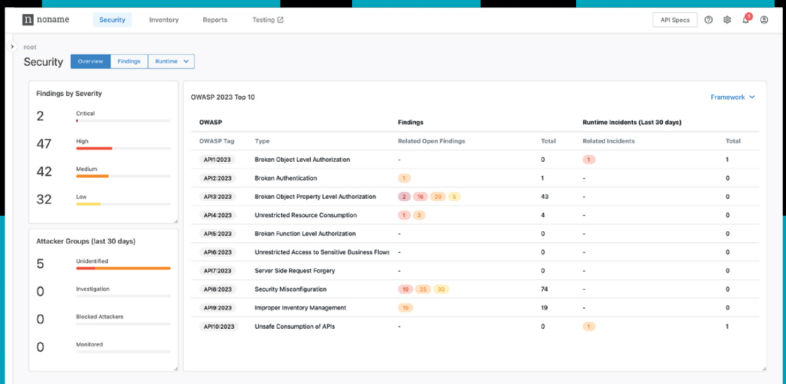
Noname Security Active Testing is a purpose-built API security testing solution that help you stop vulnerabilities before they reach production.

Test Automatically

Automatically run 150+ dynamic tests that simulate malicious traffic, including the OWASP API Top Ten.

Integrate with CI/CD

Fully integrate with your existing CI/CD pipelines, such as Jenkins and Postman, as well as all your ticketing and workflow tools such as ServiceNow, Slack, and Jira.



Proactively manage your API security posture

Major companies across the globe have all had significant API-related security incidents in recent years. In fact, 78 percent of cybersecurity professionals admitted to experiencing an API security-related incident in 2023. Why exactly? API vulnerabilities are being exploited at such an alarming rate for a number of reasons. As your organization's use of APIs expands, your attack surface expands with it, creating new security challenges — and opportunities for attackers. This guide helps you improve your organization's API security posture.

Inside...

- Reduce the risk of successful API attacks
- Learn why WAFs and API gateways aren't enough
- Ensure API runtime security
- Test your API security
- Understand the OWASP API Security Top Ten

Lawrence Miller served as a Chief Petty Officer in the U.S. Navy and has worked in information technology in various industries for more than 25 years. He is the coauthor of *CISSP For Dummies* and has written more than 250 *For Dummies* books on numerous technology and security topics.

ISBN: 978-1-394-21635-2
Not For Resale

for dummies®
A Wiley Brand

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.