wallarm

SOLUTION BRIEF

# OWASP API Security Top-10 2023 Reference Guide

Quick Guide for API Professionals to the new 2023 OWASP API Security Top-10 list

wallarm

# Introduction

The 2nd edition of the OWASP API Security Top-10 list was finalized and released in Jun-2023.

It's important for everyone concerned about API security – from CISOs to practitioners, from DevSecOps to API builders, breakers, and defenders – to understand where this all-important framework has landed. We present this guide to help you understand what's changed from the 1st edition released in 2019, what's stayed the same, and what's missing. We also provide useful guideposts for you to assess how each will impact your situation, and some tools to help build up your API security program.

# Why Do We Care?

Quite simply, our 2022 API threat data[1] show that API-focused attacks are up, API-specific vulnerabilities are growing and continue to present a High risk, and the time to respond is decreasing.

Number of Vulnerabilities
**650**

Number of Vendors
**337**

Average CVSS
**7.4**

Critical / High Vulnerabilities
**57%**

Average time to Respond (Q4)
**-3 days**

# Key Takeaways

1. The 2023 OWASP API Security Top-10 list is a good starting point, but not the be-all and end-all of API security. After all, APIs are just the beginning point – you need to consider your infrastructure, configurations, and operating systems. Indeed, all your system components need to be considered – from the software that makes up the API to the database(s) to which the software is connected, to your container configuration(s).
2. While the final 2023 OWASP API Security Top-10 list has changed a bit since 2019, we recommend you don't hastily overhaul your existing tools & processes. As we all know, security is a journey, not a destination – so rather than recklessly ripping and replacing, add to what you currently have. Build up your defenses based on your unique and evidence-based needs.
3. A holistic security approach from Dev testing ("shift left") to real-time inline protection ("shield right") is needed. By bringing both sides together, you can identify which vulnerabilities can be eliminated via your SDLC tools and those that need additional run-time protections.

[1] Source: Wallarm, 2022 Year-End API ThreatStats™ Report (Mar-2023)

wallarm

No changes

# API1:2023
# Broken Object Level Authorization (BOLA)

BOLA (aka IDOR) refers to a failed access control process, allowing a user to get access to objects which they should not be authorized to access. This could lead to an unauthorized user accessing sensitive data, manipulating data, or executing functions.

Exploitability **3** | Prevalence **3** | Detectability **3** | Technical Impact **2**

Risk Rating **6.0**

Example Case

**Tenet Healthcare (Apr-2022)**

This “cybersecurity incident” is alleged to have been caused by a BOLA attack. It led to several weeks of downtime and service delays at some of their facilities and caused $100 million in unfavorable impact due to lost revenues from interruptions to business operations and remediation costs.

Prevention

- Implement an authorization mechanism that checks whether the logged-in user has permission to perform the action.
- Use this authorization mechanism in all functions that access sensitive data.
- Use randomly generated GUIDs (as they are hard to guess) as object identifiers for user requests.

**How Wallarm Helps**

- Discover API endpoints that are potentially vulnerable to BOLA.
- Apply triggers to protect endpoints against BOLA exploitation.

**CWE Examples**

- CWE-285: Improper Authorization
- CWE-639: Authorization Bypass Through User-Controlled Key
- CWE-284: Improper Access Control

Notes:

1. The icon in the upper left of each page indicates the degree to which the category was materially changed in the 2023 release.
2. The Risk Rating ranges from 1 (low) to 9 (high) and is based on: Likelihood (average of exploitability, prevalence, and detectability) times Impact; see our OWASP 2023 Risk Ratings blog post for more.

wallarm

No changes

# API2:2023
# Broken Authentication

Improperly implemented user authentication often renders other security measures irrelevant. Technical flaws in a user authentication system can allow malicious parties to impersonate legitimate users. Some technical flaws could include using expired or leaked tokens/sessions, guessable or predictable authentication tokens, or otherwise broken credential verification before minting valid user sessions.

Exploitability **3** | Prevalence **2** | Detectability **3** | Technical Impact **3**

Risk Rating **8.0**

## Example Case

**Veeam Backup & Replication (Mar-2023)**

This flaw, tracked as CVE-2023-27532, affects all Veeam Backup & Replication (VBR) versions (enterprise and community). It could allow access by unauthenticated attackers to backup infrastructure hosts after obtaining encrypted credentials stored in the VBR configuration database.

## Prevention

- Use commonly accepted standards like OAuth and JWT for the authentication process.
- Identify and document all paths that can be used to authenticate with your API and ensure they are reviewed for possible credential leaks.
- Do not return any sensitive information like passwords, keys, or tokens directly in API responses.
- Protect all login, password recovery, and registration paths using rate limiting, brute force protection, and by adding lockout measures for abusive traffic sources.
- Implement and use multi-factor authentication (MFA) wherever possible, and use revocable tokens where implementing MFA is not feasible.

### How Wallarm Helps

- Detection and triggers for Brute Force attacks.
- Detection of weak JWT vulnerabilities.
- Detection of leaked API secrets (tokens, keys, credentials, etc.).

### CWE Examples

- CWE-204: Observable Response Discrepancy
- CWE-307: Improper Restriction of Excessive Authentication Attempts

wallarm

Changes

# API3:2023
# Broken Object Property Level Authorization

Attackers can exploit vulnerable API endpoints by reading or changing values of object properties which they are not supposed to be able to access.

- The API endpoint exposes properties of an object that are considered sensitive and should not be read by the user (previously named: "Excessive Data Exposure").
- The API endpoint allows a user to change, add, and/or delete a sensitive object's property value which they should not be able to access (previously named: "Mass Assignment").

Exploitability **3** | Prevalence **2** | Detectability **3** | Technical Impact **2**

Risk Rating **5.3**

## Example Case

**Reviver Digital License Plates (Jan-2023)**

User accounts were assigned to a unique “company” JSON object which allows other sub-users to be added to the account. Via the password reset URL, a malicious actor could elevate privileges, the ability to administer vehicles, fleets, and user accounts, and gain access to many more API endpoints and functionality. The attacker could then remotely update, track, or delete anyone’s REVIVER plate

## Prevention

- Define exactly which object properties are to be returned in your API functions rather than returning entire objects.
- Do not directly assign user input to objects in your API functions or create or update objects by directly assigning user input.
- Explicitly define the object properties that the user may update in your API code.
- Return only the data the client requests from your API functions rather than returning all available data and expecting the client to filter it.
- Limit the number of records that can be affected by a query in API functions to prevent mass updating or disclosure of database records.
- Validate API responses from a central schema that filters out object properties that should not be visible to the requesting user.

### How Wallarm Helps

- Detection of Information Exposure vulnerabilities and attacks.
- Detection Mass Assignment attacks.

### CWE Examples

- CWE-213: Exposure of Sensitive Information Due to Incompatible Policies
- CWE-915: Improperly Controlled Modification of Dynamically-Determined Object Attributes

wallarm

No changes

# API4:2023
# Unrestricted Resource Consumption

It's common to find APIs that do not limit client interactions or resource consumptions. Although most of the time interactions are logged, due to the lack of or improper monitoring, malicious activity can go unnoticed. Exploitation can lead to DoS due to resource starvation, but it can also impact service providers' billing.

Exploitability **2** | Prevalence **3** | Detectability **3** | Technical Impact **3**

Risk Rating **8.0**

## Example Case

**Polish Government Sites (Q1-2023)**

In Mar-2023, a distributed denial-of-service (DDoS) attack temporarily knocked podatki.gov.pl (the government tax portal) offline, disrupting access.

## Prevention

- Use container-based solutions that make it easy to limit memory, CPU, number of restarts, file descriptors, and processes.
- 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 is stored locally or in cloud storage).
- Implement a limit on how often a client can interact with the API within a defined timeframe (rate limiting).
- Rate limiting should be fine-tuned based on the business needs. Some API Endpoints might require stricter policies.
- Limit/throttle how many times or how often a single API client/user can execute a single operation (e.g., validate an OTP, or request password recovery without visiting the one-time URL).
- 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.

### How Wallarm Helps

- Detection of API Abuse attacks, such as Brute Force, Resource Overlimit, Data Bombing, and more.
- Sophisticated Rate-Limiting capabilities.

### CWE Examples

- CWE-770: Allocation of Resources Without Limits or Throttling
- CWE-400: Uncontrolled Resource Consumption
- CWE-799: Improper Control of Interaction Frequency

wallarm

No changes

# API5:2023
# Broken Function Level Authorization (BFLA)

Exploitation requires the attacker send legitimate API calls to an endpoint to which they should not have access; for instance, replacing the HTTP method from GET to PUT, or changing the "users" string in the URL to "admins."

Exploitability **3** | Prevalence **2** | Detectability **3** | Technical Impact **3**

Risk Rating **8.0**

Example Case

**Reddit (Apr-2022)**

Malicious actors can bypass the review process simply by altering the admin_approval value to APPROVED and effective_status to ACTIVE, changing the ads status to "approved and active" without review nor payment.

Prevention

- The enforcement mechanism(s) should deny all access by default, requiring explicit grants to specific roles for access to every function.
- Review your API endpoints against function level authorization flaws, while keeping in mind the business logic of the application and groups hierarchy.
- Make sure that all your administrative controllers inherit from an administrative abstract controller that implements authorization checks based on the user's group/role.
- Make sure that administrative functions inside a regular controller implement authorization checks based on the user's group and role.

**How Wallarm Helps**

- Attack detections, including Path Traversal, Forced Browsing, etc.
- Vulnerability detections, including Path Traversal, Forced Browsing, Open Redirect, CSRF, etc.

**CWE Examples**

- CWE-285: Improper Authorization

wallarm

New

# API6:2023
# Unrestricted Access to Sensitive Business Flows

Exploitation usually involves understanding of the business model of the API, finding sensitive business flows, and automating access to these flows. This could impact the business in several ways; for example: 1. Prevent legitimate users from purchasing a product; 2. Lead to inflation in the internal economy of a game; 3. Allow the attacker to send excessive amounts of messages/comments and easily spread fake news.

Exploitability **3** | Prevalence **3** | Detectability **2** | Technical Impact **2**

Risk Rating **5.3**

Example Case

**Twitter (Jun-2021)**

From Jun-2021 until Jan-2022, a Twitter API bug allowed attackers to submit contact information like email addresses and receive the associated Twitter account, if any, in return. Attackers exploited this flaw to “scrape” data from Twitter. And while the bug didn't allow hackers to access passwords or other sensitive information like DMs, it did expose the connection between Twitter accounts, which are often pseudonymous, and the email addresses and phone numbers linked to them, potentially identifying users.

Prevention

- Business – identify the business flows that might harm the business if they are excessively used.
- Engineering – choose the right protection mechanisms to mitigate the business risk.
- Secure and limit access to APIs that are consumed directly by machines, such as developer and B2B APIs. They tend to be easy targets for attackers because the required protection mechanisms are often not (properly) implemented.

**How Wallarm Helps**

- API Abuse Prevention, including Account Takeover (ATO), Brute Force attacks, Crawlers / Scrapers, and more.
    - Covers both anonymous sessions and legitimate users (e.g., with tokens).
    - Does not rely on JavaScript or CAPTCHAs.

**CWE Examples**

none listed

wallarm

New

# API7:2023
# Server Side Request Forgery (SSRF)

Successful exploitation might lead to internal services enumeration (e.g., port scanning) or information disclosure, bypassing firewalls or other security mechanisms, etc. This vulnerability allows attackers to cause the server-side application to make requests to an unintended location. Exploitation requires the attacker to find an API endpoint that receives a URI as a parameter and then accesses the provided URI.

Exploitability **3** | Prevalence **2** | Detectability **3** | Technical Impact **2**

Risk Rating **5.3**

## Example Case

**WordPress (Sep-2022)**

A 6-year-old blind server-side request forgery (SSRF) vulnerability, first surfaced in 2017, in the pingback functionality exposed on the XMLRPC API, a core WordPress feature, could enable distributed denial-of-service (DDoS) attacks by maliciously asking 1000s of blogs to check for pingbacks on a single victim server.

## Prevention

- Isolate the resource fetching mechanism in your network: usually these features are aimed to retrieve remote resources and not internal ones.
- Whenever possible, use allow lists of
  - Remote origins users are expected to download resources from (e.g., Google Drive, Gravatar, etc.).
  - URL schemes and ports.
  - Accepted media types for a given functionality.
- Disable HTTP redirections.
- Use a well-tested and maintained URL parser to avoid issues caused by URL parsing inconsistencies.
- Validate and sanitize all client-supplied input data.
- Do not send raw responses to clients.

### How Wallarm Helps

- SSRF vulnerability detection.

### CWE Examples

- CWE-918: Server-Side Request Forgery (SSRF)

wallarm

No changes

# API8:2023
# Security Misconfiguration

Prevent attackers from finding unpatched flaws, common endpoints, or unprotected files and directories to limit unauthorized access or knowledge of the system.

Exploitability **3** | Prevalence **3** | Detectability **3** | Technical Impact **3**

Risk Rating **9.0**

Example Case

**IBM Cloud Databases for PostgreSQL (Dec-2022)**

A vulnerability, consisting of three (3) exposed secrets (K8s service account token, private container registry password, CI/CD server credentials) combined with overly permissive network access to internal build servers, could result in RCE access to customers' environments and possibly allowing malicious actors to read and modify the data stored in the PostgreSQL database.

Prevention

- Ensure your deployment process is security hardened and well-documented so that a secure hosting environment can be reproduced.
- Review your deployment configurations and process regularly, including any software dependencies used in your API, deployment and configuration files, and the security of your cloud infrastructure.
- Limit all client interactions with your API and any other resources (such as linked media) to secure, authorized channels.
- Only allow API access using necessary HTTP verbs to reduce attack surfaces.
- Set CORS policies for APIs that are publicly accessible from browser-based clients.

**How Wallarm Helps**

- Detection of Scanner attacks.
- Detection of Scanner vulnerabilities and vulnerable components.

**CWE Examples**

- CWE-2: Environmental Security Flaws
- CWE-16: Configuration
- CWE-209: Generation of Error Message Containing Sensitive Information
- CWE-319: Cleartext Transmission of Sensitive Information
- CWE-388: Error Handling
- CWE-444: Inconsistent Interpretation of HTTP Requests ('HTTP Request/Response Smuggling')
- CWE-942: Permissive Cross-domain Policy with Untrusted Domains

wallarm

No changes

# API9:2023
# Improper Inventory Management

How many APIs do you have? Threat agents usually get unauthorized access through old API versions or endpoints left running unpatched and using weaker security requirements. Visibility across your entire API portfolio – internal & external, documented & undocumented, etc. – including endpoints and data flows is critical to your security posture.

Exploitability **3** | Prevalence **3** | Detectability **2** | Technical Impact **2**

Risk Rating **5.3**

## Example Case

**Optus (Sep-2022)**

Optus was hit by a massive data breach exposing as many as 10 million customer accounts, which reports suggest was due to an exposed API which did not require authorization or authentication to access customer data, meaning that anyone who knew the endpoint URL could abuse it.

## Prevention

- Inventory all API endpoints.
- Know which of APIs poses most of the risk.
- Know which of APIs handle PII data.
- Track which APIs are new or got updated. Prioritize them for the pentests and bug bounty.
- Document all aspects of your API such as authentication, errors, redirects, rate limiting, cross-origin resource sharing (CORS) policy, and endpoints, including their parameters, requests, and responses.
- Generate documentation automatically by adopting open standards. Include the documentation build in your CI/CD pipeline.
- Use external protection measures such as API security specific solutions for all exposed versions of your APIs, not just for the current production version.

### How Wallarm Helps

- Detection of sensitive data flows, such as PII, PHI, credentials, etc.
- Detection of Shadow / Zombie APIs.

### CWE Examples

- CWE-1059: Incomplete Documentation

wallarm

New

# API10:2023
# Unsafe Consumption of APIs

Developers tend to trust but not verify endpoints which interact with external or third-party APIs. Successful exploitation of security flaws in these APIs can impact those relying on them. Root causes include weak(er) security requirements for external or 3rd party APIs, especially regarding transport security, AuthN / AuthZ, and input validation and sanitization. Note that Injections (formerly API8:2019) has been absorbed into this category. See our Spotlight on Injections (next page) for a more in-depth discussion on this critical class of vulnerabilities.

Exploitability **3** | Prevalence **2** | Detectability **2** | Technical Impact **3**

Risk Rating **7.0**

Example Case

**Facebook (Sep-2018)**

Nightwatch Security researcher Yakov Shafranovich reported that a third-party Android application with Facebook API access was copying and storing data outside of the social network in an insecure manner. The application accessed user data through the Facebook API and copied it to a Firebase database and API server without any authentication or HTTPS protections in place. One of the databases accessed contained over 1,000,000 records.

Prevention

- When evaluating service providers, assess their API security posture.
- Ensure all API interactions happen over a secure communication channel (TLS).
- Always validate and properly sanitize data received from integrated APIs before using it.
- Maintain an allowlist of well-known locations integrated APIs may redirect yours; do not blindly follow redirects.

**How Wallarm Helps**

- Detection of Injection vulnerabilities and attacks, including XSS, RCE, SQLi / NoSQLi, CRFLi, LDAPi, SSTi, SSI, Email Injection, XXE, and more.

**CWE Examples**

- CWE-20: Improper Input Validation
- CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
- CWE-319: Cleartext Transmission of Sensitive Information

# Spotlight on Injection

Injection attacks can lead to information disclosure, data loss, DoS, or even complete host takeover. Attackers input malicious code or commands via whatever injection vectors are available in the API (e.g., direct input, parameters, integrated services, etc.) which is then sent to the interpreter and can bypass security, change permissions, and much more.

It's true that Injection risks, previously a stand-alone category (API8:2019), have been incorporated into API10:2023, but our data[2] show Injections constitute the largest single API risk group. We recommend you treat them as a critical part of your API Security program.

Exploitability **3** | Prevalence **3** | Detectability **3** | Technical Impact **3**

Risk Rating **9.0**

**Example Case**

**JeecgBoot SQL Injection Vulnerability (Mar-2023)**

A remote authenticated attacker could send specially crafted SQL statements to the jmreport/qurestSql endpoint, which due to improper user-input sanitization via the apiSelectId parameter could allow the attacker to view, add, modify, or delete information in the back-end database. An exploit was published before the CVE was created. (CVE-2023-1454; CVSS score: 9.8)

**Prevention**

- Perform data validation using a single, trustworthy, and actively maintained library.
- Validate, filter, and sanitize all client-provided data, or other data coming from integrated systems.
- Special characters should be escaped using the specific syntax for the target interpreter.
- Always limit the number of returned records to prevent mass disclosure in case of injection.
- Validate incoming data using sufficient filters to only allow valid values for each input parameter.
- Define data types and strict patterns for all string parameters.

**How Wallarm Helps**

- Detection of Injection vulnerabilities and attacks, including XSS, RCE, SQLi / NoSQLi, CRFLi, LDAPi, SSTi, SSI, Email Injection, XXE, and more.

**CWE Examples**

- CWE-79: Cross-site Scripting (XSS)
- CWE-22: Path Traversal
- CWE-89: SQL Injection (SQLi)

and many others.

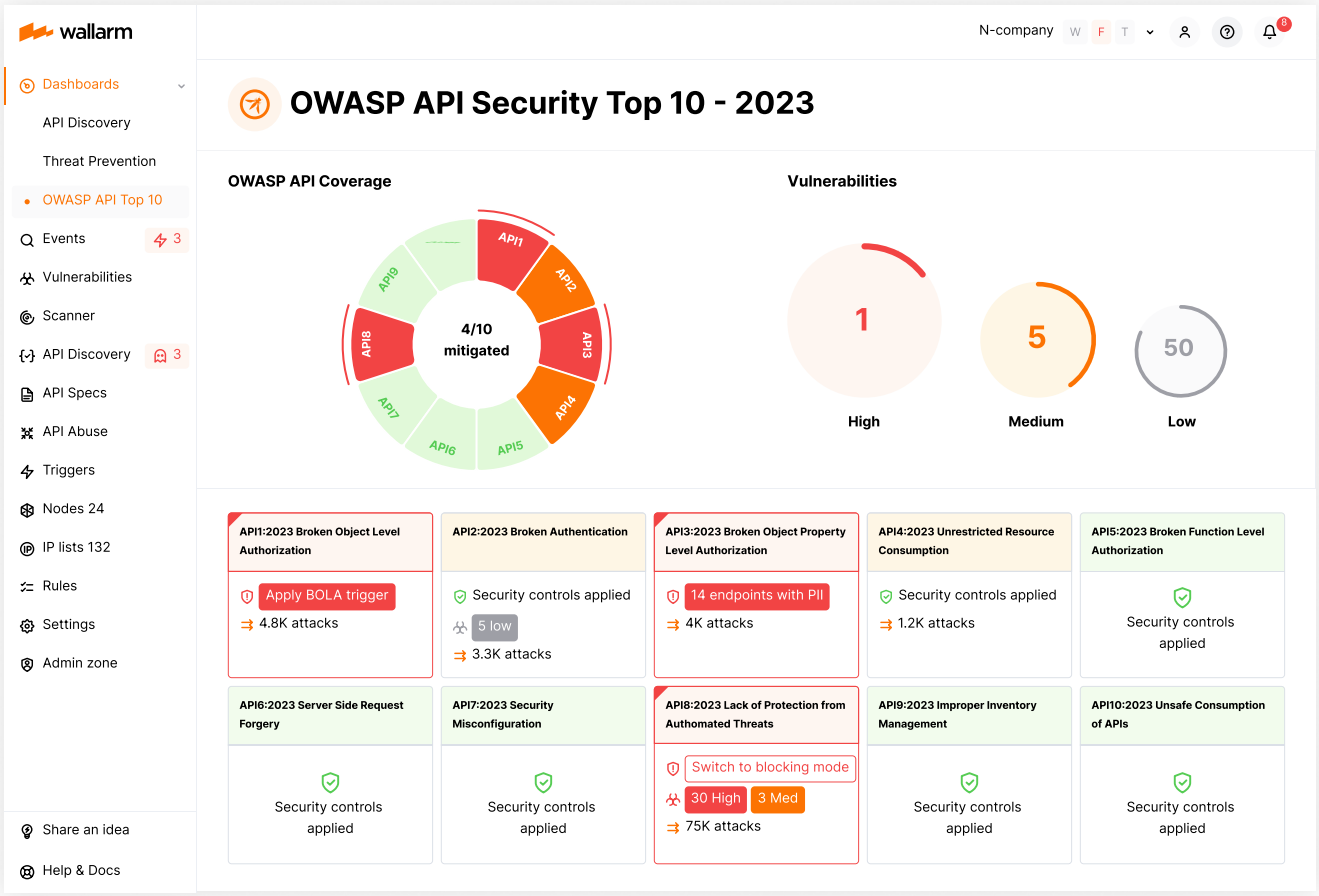


[2] Source: Wallarm, 2022 Year-End API ThreatStats™ Report (Mar-2023)

# Protect Your APIs from OWASP API Security Top-10 Threats

Wallarm End-to-End API Security solution provides comprehensive protection against the OWASP API Security Top-10 threats. And in 2023, we are making this even easier for you!

The **2023 OWASP API Security Top-10 dashboard** provides you with complete visibility into the security state of your APIs, easy identification of your most critical security risks, and ability to immediately apply protective measures.




Implementing a robust API Security program becomes much easier with the OWASP API Security Top-10 dashboard from Wallarm. The automated security report enables you to pinpoint the most critical risks in your APIs, thoroughly analyze all associated events, and effortlessly apply appropriate security controls to mitigate them. By combining the strengths of complete visibility with real-time threat prevention, this capability reduces your emerging threat risks, your workload, and your security costs.

If you are interested in learning more about how we can help you protect your APIs, please schedule a demo with one of our security experts today!

wallarm

# Learn More

## Additional Resources

**OWASP Software Assurance Maturity Model**

https://owaspsamm.org/ and/or https://owaspsamm.org/model/

**OWASP Cheat Sheet Series**

https://cheatsheetseries.owasp.org/index.html

**OWASP Automated Threats to Web Applications**

https://github.com/OWASP/www-project-automated-threats-to-web-applications

**Wallarm 2022 Year-End API ThreatStats™ Report**

https://www.wallarm.com/resources/2022-year-end-api-threatstats-full-report

## API Security Training

- OWASP Juice Shop – Can be used in security trainings, awareness demos, CTFs and as a guinea pig for security tools.
- Kontra – Free interactive application security training modules on OWASP API.
- 3 training resources to improve your API hacking tradecraft – Blog post by Dana Epp.

## Vulnerable API Sandbox for Team Training

- crAPI from OWASP – Challenges. Deployed with Docker.
- vAPI – Exercises. Postman collection deployed as Helm.
- VAmPI – Postman collection, deployed with Docker.
- Damn Vulnerable GraphQL Application – GraphQL-specific, deployed as Docker.
- DVWS-node – Deployed as Docker.

wallarm

## About Wallarm

Wallarm delivers security software and services providing robust protection for APIs, web applications, microservices, and serverless workloads running in cloud-native environments. Hundreds of Security and DevOps teams choose Wallarm to discover all APIs and web applications running in their environment, to get full visibility into malicious traffic, to protect their entire public, private and partner API portfolio, and to automate incident response for their cybersecurity programs. We support modern tech stacks and API protocols, and offer deployment options for SaaS, multi- and hybrid-cloud, Kubernetes and more. Wallarm is headquartered in San Francisco, California, and is backed by Toba Capital, Y Combinator, Partech, and other investors.

(415) 940-7077
188 King St. Unit 508, San Francisco, CA 94107
www.wallarm.com