CRASHTEST **SECURITY**

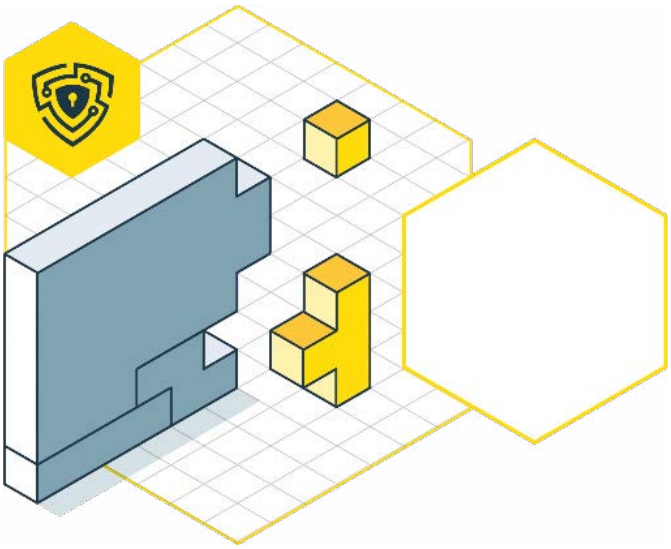
GUIDE FOR PREVENTING API VULNERABILITIES

WHAT ARE THE STEPS TO KEEP YOUR WEB APP OR API SAFE FROM SUCH VULNERABILITY

GUIDE FOR THE API VULNERABILITY PREVENTION

Table of Contents

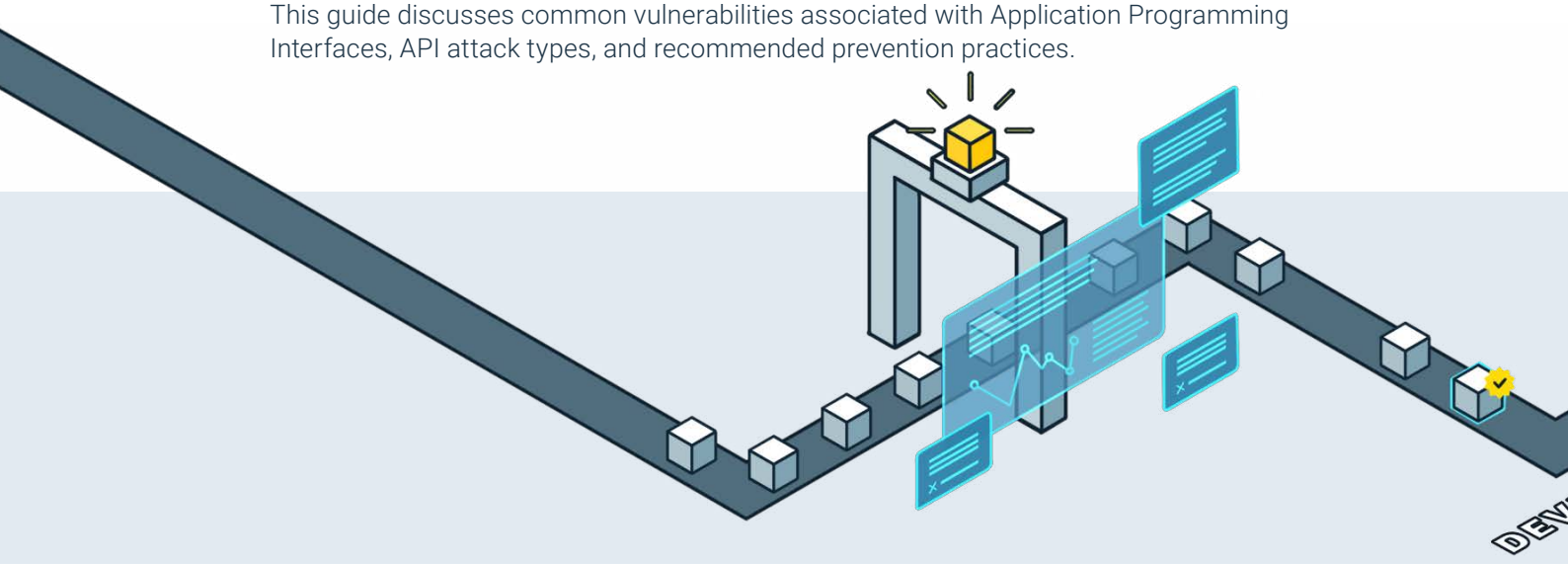
What are API Vulnerabilities? 3
Types of API Attacks 5
API Attack - Severity Levels 6
Identify API Vulnerabilities with Crashtest Security 6
API Vulnerabilities Prevention Techniques 6
Best Practices in Preventing API Attacks 8
Prevent API Attacks with Crashtest Security 9



INTRODUCTION TO THIS GUIDE

An Application Programming Interface (API) is a set of programming codes that allows the seamless integration between various software applications, processes, and users. The machine-readable interface is fundamentally built to exchange data and functionality without modifying an existing application, enabling cross-platform consistency while reducing the manual overhead of developing and maintaining a tech stack. While APIs offer enormous benefits to modern application delivery, the interface is also susceptible to cyberattacks without security practices and tools.

This guide discusses common vulnerabilities associated with Application Programming Interfaces, API attack types, and recommended prevention practices.



WHAT ARE API VULNERABILITIES?

Since they expose sensitive data and application logic, APIs are now a prime target of cybersecurity attacks that reveal critical information and compromise enterprise systems. Insecure implementation of APIs often makes them susceptible to various exploits by attackers. As broken or hacked APIs expose sensitive data and application logic, API attacks continue to be a significant cause of worry for organizations that rely on modern computing.

Common vulnerabilities for API security include:

Broken Object Level Authorization (BOLA)

he BOLA vulnerability is one of the most common and severe API vulnerabilities per the OWASP API Security Top 10. Also known as Insecure Direct Object Reference (IDOR), the BOLA vulnerability develops when the API incorrectly exposes sensitive fields stored within an object. When a vulnerable API relies on client-side object IDs rather than the client's state to make decisions on object access, attackers can replace the object id in the API call with that of a different user. In instances where the API endpoint does not perform adequate object-level authorization checks, attackers can gain access to a vulnerable user's data or orchestrate a complete account takeover.

Excessive Data Exposure

APIs are data sources that often rely on the client to filter data within the response. In generic API implementations, developers include all data and functionality within the API response sent to the client browser without filtering or masking sensitive information. Hackers can intercept such API communications and misuse sensitive data to perform reconnaissance and enumeration for advanced attacks.

Improper Asset Management

APIs expose multiple endpoints having numerous versions, functionalities, and parameters that affect the operation of these endpoints. This necessitates development teams to build and maintain detailed API assets documentation covering all hosts, endpoints, and versions. Lack of proper documentation and improper management of API assets makes it difficult for teams to identify and mitigate implementation issues such as exposed endpoints or deprecated API versions.

Broken Authentication

This is one of the most common API vulnerabilities targeted by attackers to exploit the inadequate implementation of user authentication tokens. An exposed API key or broken authentication functionality allows malicious actors to illegally access the endpoint, enabling them to compromise security controls to widen the reach of their subsequent attacks.

Insufficient Resources and Rate Limiting

Poor implementation of resources and rate-limiting affects the API's incapability to restrict the number of requests or bot traffic generated from a single source. While this makes APIs vulnerable to brute-force attacks, DoS attacks, and performance issues, the misconfiguration also aids the exploitation of other API vulnerabilities such as broken object-level authorization. This also enables attackers to overload the APIs by submitting numerous requests using bot traffic, making the server unavailable and causing disruption to the business.

Mass Assignment

This vulnerability exploits the feature of the application framework that allows developers to assign user input values to multiple program variables or objects simultaneously. Attackers target this vulnerability to overwrite existing server-side variables, initialize new variables, execute malicious code, or access sensitive data. Attackers can also craft malicious requests to include additional unintended parameters, thereby adversely affecting the functionality of an application.

Broken Function Level Authorization

This vulnerability occurs when the API

- fails to properly implement access control functions that an authorized user can perform, or
- can not segregate the access limits of administrative and regular users.

Malicious users exploit these authentication flaws to gain access to restricted and sensitive actions such as administrative functionalities, facilitating further attack advancement or complete account take over.

Security Misconfiguration

Developers often miss security hardening across the entire API stack, making the API vulnerable to security threats. Such instances allow attackers to uncover misconfigurations and exploit them as vectors for unauthorized user and system data access. Some common security misconfigurations of APIs include:

- Usage of inadequate authentication/default credentials
- Use of unnecessary HTTP methods
- Misconfigured response headers
- Data leakage
- Cross-origin resource sharing
- Exposed cloud resources
- Unsanitized incoming requests

Insufficient Logging & Monitoring

Inefficient or missing logging and monitoring mechanisms are a prevailing vulnerability for APIs. Security incidents often go undetected without adequate monitoring mechanisms, enabling attackers to orchestrate further attacks on the system. Without consistent logs to track, it is also near-impossible for security teams to back-trace security incidents and inherent flaws that lead to issues such as failed authentication checks, input validation failures, and other warnings that make the system susceptible to API attacks.

WHAT ARE API VULNERABILITIES?

Due to the ubiquity and the data they expose, APIs are one of the most popular attack vectors in modern tech stacks. Attacks targeting API endpoints include:

Injection Attacks

Injection attacks are a broad class of API attack vectors that extend long-standing potential threats against APIs and web applications. In an injection attack, attackers inject malicious code to a poorly developed, unsecured API, ensuring unauthorized access to the software code and data. To do so, hackers typically inject malicious OS commands, XML parameters, or SQL queries to be executed by the server. A successful attack's impacts are most likely data loss, theft, denial of service, or comprehensive compromise of legitimate user/system accounts.

DDoS Attacks

Denial of Service (DoS) or Distributed Denial of Service (DDoS) attacks involve sending numerous incoming requests to the target server to cause overload and become inaccessible to legitimate users. Instead of targeting specific functions or data, these attack types typically target the entire API endpoint. The severity of such attacks is often severe, which leads to the loss of revenue, reputation, and intellectual property of an organization.

Parameter Tampering Attacks

In this attack, the malicious actor targets the application's business logic exposed by the API. Hackers usually leverage vulnerabilities in backend validation to manipulate parameters within an API request, subsequently enabling them to bypass security controls that use these parameters. While parameters raise the level of control over the application's behavior, they also allow malicious users to modify API request query parameters or input fields within a web form.

Man-in-the-Middle (MITM) Attacks

The man-in-the-middle attack is an eavesdropping exploit where threat actors position themselves between an authorized user and the API endpoint. Such unauthorized users act as a third party who intercepts, alters, and selectively relays API calls between two interacting entities. MITM attacks often occur in web applications with insufficient cryptography, allowing the hacker to intercept and decrypt messages in transit.

Authentication Hijacking Attacks

APIs configured to have state as part of their workflow also contain information on generating and storing sessions. In vulnerable applications, threat actors can perform session replays to uncover personal data and credentials the API associates with a regular user, such as API keys, session cookies, and access tokens. The attacker can further exploit this information to assume legitimate users' identities. Malicious actors also leverage identity and credential management errors to obtain stolen authentication tokens of user accounts and application data.

Cross-Site Request Forgery (CSRF) Attacks

Stateful APIs are vulnerable to CSRF attacks if they accept and use cookie data to manage the session. These APIs persist credentials within the endpoint, which the bad actors can obtain and use to send requests from malicious sites. If a leaky API server accepts non-HTTPS requests, the attacker can spoof the API response header, enabling legitimate users to perform malicious actions on their behalf.

API ATTACK - SEVERITY LEVELS

Application Programming Interfaces offer a seamless approach to exposing system data and functionalities. The severity and impacts of a successful attack typically vary depending on the security vulnerability exploited, the actions performed by the unauthorized user, and the data targeted. Consequences of a successful API attack include:

- Loss of revenue/business reputation
- Data breaches/exposure of sensitive data
- Service/function unavailability
- Account takeover/impersonation of regular users
- Data deletion
- Manipulation of response headers
- Alteration of API calls and database queries
- Content scraping and intellectual property theft
- Credit Card Fraud
- Bypassing authentication/authorization controls
- Privilege escalation

HOW TO IDENTIFY API VULNERABILITIES WITH CRASHTEST SECURITY?

Crashtest Security Suite's automated scanner discovers known security risks to help identify and fix a vulnerable API with a click of a button. The suite includes an API vulnerability scanner that benchmarks the application's API security against the OWASP API top 10 to help quickly detect API threat vectors. The platform also ships with various other scanners, including a CSRF Scanner, Javascript Security Scanner, Microservices Security Scanner, and HTTP Header Scanner, to help detect malicious code and workloads in API traffic.

Crashtest Security fits seamlessly into modern workflows and pipelines, allowing API developers to integrate automated security testing into their programming interfaces. The platform also generates a comprehensive vulnerability report of uncovered vulnerabilities and offers remediation advice for adequate API security.

API VULNERABILITIES PREVENTION TECHNIQUES

Some standard prevention controls to reduce the API attack surface include:

API RATE-L IMITING

Rate-limiting effectively helps combat API vulnerabilities such as Denial of Service (DoS), Distributed Denial of Service (DDoS), and brute force attacks. Rate-limiting acts as a gatekeeper for controlling the number of incoming/outgoing requests an API can handle at a given time and is usually achieved using throttling and quotas. Throttling and quotas protect endpoints from being overwhelmed by numerous requests causing disruptions in standard API processing and unusual resource spikes. If the request exceeds the permitted limits, the API response returns an error message while enforcing a waiting period before the user/process can re-establish the connection through the API.

DEPLOYING A WEB APPLICATION FIREWALL

A web application firewall generates logs for transactions, errors, and access, which can be used for user behavior analytics, application troubleshooting, and other practices for effective API security. An efficient firewall administers security controls based on the OpenAPI or Swagger definition of the API.

Once deployed, the firewall checks the API definition to distinguish between unauthorized users and regular traffic. The protection configuration defines a whitelist of allowed operations and data input and applies to outgoing and incoming traffic. API developers can define a customized set of access control instructions in the API‘s OpenAPI or Swagger definition to allow or block specific web requests.

WAF is commonly used to protect API Gateways from different web exploits such as XSS and MITM attacks by restricting the interception of application traffic. Additionally, WAF helps mitigate DDoS attacks on application-layer and SQL injection attacks by utilizing strong SSL/TLS encryption.

OPENID CONNECT (OIDC)

OIDC and OAuth 2.0 are the two fundamental elements for securing API logins and the data exposed by such APIs. OIDC is an authentication protocol that uses OAuth 2.0 grant types to add an identity layer for storing and verifying user profiles and login information.

OIDC also helps with authentication and authorization of users through Single Sign-On to API workloads. The protocol offers optional security measures for encryption and signing while relying on access codes, passwords, and implicit credentials to obtain the legitimate user‘s identity.

ENFORCING TRANSPORT LEVEL SECURITY

Transport Level Security (TLS) is often considered the first line of defense against API attacks. The protocol facilitates secure and encrypted API communications, protecting the data in transit while supporting mutual authentication mechanisms to ensure legitimate access to APIs.

Enforcing TLS protocol ensures client information reaches only the intended and authorized recipients, preventing attacks such as parameter tampering, eavesdropping, and man-in-the-middle attacks. Developers should supplement Transport Level Security measures alongside an access control list to prevent unauthorized access to the API server for effective protection.

BEST PRACTICES IN PREVENTING API ATTACKS

Best practices to prevent the API server attacks include:

CREATE AND UPDATE API INVENTORIES

APIs are a cornerstone of modern digital transformation, with enterprises deploying numerous public-facing APIs to make their applications flexible and easy to integrate. An organization's security experts must maintain an updated inventory of all deployed APIs, their state, and services for efficient management and contingency planning. To administer robust security of an organization's API, it is recommended to conduct periodic perimeter scans that help identify vulnerability and tag APIs appropriately.

ENFORCE THE PRINCIPLE OF LEAST PRIVILEGE

Enforcing the principle of least privilege is recommended as it minimizes the attack surface and reduces the security risk of unauthorized access to critical data/processes. Adhering to the least privilege principle, security teams should grant authorized users only the minimum required access to perform a permitted action. APIs should also be provided access only to the data and processes required for completing a transaction, thereby preventing excessive data exposure.

PERFORM USER INPUT VALIDATION

As often unauthorized users submit requests from compromised accounts while parading as legitimate users, the API server should not blindly consume input data provided by any API client. It is recommended to implement proper input validation techniques both at the server and client-side to validate the input before passing it to the endpoints properly. API developers should also define rules that help identify malicious incoming requests, preventing hackers from submitting harmful code or modified queries that can compromise the API endpoint.

ENFORCE STRONG AUTHENTICATION AND AUTHORIZATION MECHANISMS

Since APIs are publicly accessible and act as entry points for an organization's internal systems, it is crucial to control and verify the identity and permissions assigned to a user/process before granting access to these APIs.

As a recommended practice, organizations should implement robust, proven, and standard authentication mechanisms, such as JWT access tokens, multi-factor authentication, and OAuth protocol, for verifying the identity of API users sending the requests.

API authorization should be governed by role-based access control, where every role is assigned with a pre-defined set of permissions. Teams should also adhere to the principle of least privilege to ensure that only legitimate API users are allowed access to the functions and data necessary for their role, reducing the chance of malicious actors obtaining sensitive information.

USE AN API VULNERABILITY SCANNER

Software teams should enable continuous automated scanning and monitoring of APIs to identify and mitigate vulnerabilities across the complete API lifecycle. It is recommended for security teams to utilize vulnerability scanners that automatically identify and document security gaps against all known vulnerabilities.

API VULNERABILITY SCANNING WITH CRASHTEST SECURITY

Crashtest Security ships with an API vulnerability scanner out of the box, leveraging documented OpenAPI and Swagger profiles to scan an application for vulnerabilities. The scanner comprehensively scans all endpoints to generate an actionable report, including the vulnerabilities discovered, their respective severity levels, and recommended remediation actions.

The Crashtest Security Suite integrates several tools and systems to help identify security risks across an entire tech stack. The platform's automated API vulnerability scanner seamlessly integrates with your existing workflow to detect potential threats within API endpoints and other supporting components of the interface.

To learn how Crashtest Security can help prevent API vulnerabilities of your applications, try a 14-day free trial here.

Start 2-Week Trial for Free

CRASHTEST **SECURITY**

WWW.CRASHTEST-SECURITY.COM