DeskLodge

2 Free Day Hot Desking Trial

It’s time to work wonderful

Securing Your API
The OWASP Top 10

Rob Allen, February 2026

57 % of organizations suffered an API-related data breach in the past two years

Traceable 2025 Global State of API Security report

Why APIs are different?

The OWASP API Security Project seeks to provide value to software developers and security assessors by underscoring the potential risks in insecure APIs

OWASP API Security Top 10

OWASP API Security Top 10

Who are you and what can you access?

Authentication and authorisation failures

Broken Authentication (#2)

APIs that don't properly verify who you are

Broken Authentication (#2)

APIs that don't properly verify who you are

- Weak/no token validation
- Missing expiration on tokens
- Credential stuffing attacks

Broken Authentication (#2)

APIs that don't properly verify who you are

- Weak/no token validation
- Missing expiration on tokens
- Credential stuffing attacks

Example: API accepts JWT without verifying the signature

Broken Authentication (#2)

Prevention

Broken Authentication (#2)

Prevention

- Use established standards (OAuth 2.0, OpenID Connect)
- Implement proper token validation and expiration
- Rate limiting on auth endpoints

Broken Function Level Authorisation (#5)

Users can access functionality they shouldn't

Broken Function Level Authorisation (#5)

Users can access functionality they shouldn't

- Incorrect authorisation checked on a function or resource
- Legitimate calls to endpoints that the user shouldn't have access to
- Undocumented open endpoints

Broken Function Level Authorisation (#5)

Users can access functionality they shouldn't

- Incorrect authorisation checked on a function or resource
- Legitimate calls to endpoints that the user shouldn't have access to
- Undocumented open endpoints

Example: `/debug/dump`

Broken Function Level Authorisation (#5)

Prevention

Broken Function Level Authorisation (#5)

Prevention

- Deny by default
- Check roles/permissions on every endpoint
- Don't rely on hiding endpoints from documentation

Broken Object Level Authorisation (#1)

Users can access objects belonging to other users

Broken Object Level Authorisation (#1)

Users can access objects belonging to other users

- User can access another user's resource
- Changing an ID or key name allows access to privileged data

Broken Object Level Authorisation (#1)

Users can access objects belonging to other users

- User can access another user's resource
- Changing an ID or key name allows access to privileged data

Example: `/users/123/orders` - change to 124 and see someone else's orders

Broken Object Level Authorisation (#1)

Prevention

Broken Object Level Authorisation (#1)

Prevention

- Implement proper authorisation based on user policies
- Check if the user has access the requested resource
- Check that the operation is also allowed

Broken Object Property Authorisation (#3)

Users can read or modify properties they shouldn't

Broken Object Property Authorisation (#3)

Users can read or modify properties they shouldn't

- Sending properties that this user shouldn't see
- Allowing this user to change a property they shouldnt

Broken Object Property Authorisation (#3)

Users can read or modify properties they shouldn't

- Sending properties that this user shouldn't see
- Allowing this user to change a property they shouldnt

Example: User updates profile, includes `"role": "admin"` in payload

Broken Object Property Authorisation (#3)

Prevention

Broken Object Property Authorisation (#3)

Prevention

- Cherry pick object properties to return
- Explicit allowlists for input properties

Exploiting how your API works

Business logic and resource abuse

Unrestricted resource consumption (#4)

APIs that can be abused through resource consumption

Unrestricted resource consumption (#4)

APIs that can be abused through resource consumption

- Expensive operations without throttling
- Exhausting memory through requests for too much data
- Denial of service

Unrestricted resource consumption (#4)

APIs that can be abused through resource consumption

- Expensive operations without throttling
- Exhausting memory through requests for too much data
- Denial of service

Example: `/widgets?page=1&per_page=1000000`

Unrestricted resource consumption (#4)

Prevention

Unrestricted resource consumption (#4)

Prevention

- Rate limiting (per IP, per user, per endpoint)
- Pagination with maximum limits
- Resource quotas / Timeouts

Unrestricted access to business flows (#6)

Critical workflows lack protection against automation

Unrestricted access to business flows (#6)

Critical workflows lack protection against automation

- Some business flows are more sensitive than others
- Legitimate calls, but unexpected order"
- Excessive access may harm the business

Unrestricted access to business flows (#6)

Critical workflows lack protection against automation

- Some business flows are more sensitive than others
- Legitimate calls, but unexpected order"
- Excessive access may harm the business

Example: Ticket scalping bots, inventory hoarding

Unrestricted access to business flows (#6)

Prevention

Unrestricted access to business flows (#6)

Prevention

- Device fingerprinting
- Behavioral analysis
- Transaction limits

Unsafe consumption of APIs (#10)

Your API trusts third-party APIs too much

Unsafe consumption of APIs (#10)

Your API trusts third-party APIs too much

- Dependency on another's vulnerabilities
- Malicious data can be injected
- Not accounting for failure

Unsafe consumption of APIs (#10)

Your API trusts third-party APIs too much

- Dependency on another's vulnerabilities
- Malicious data can be injected
- Not accounting for failure

Example: Geolocation API takes 30 seconds to time out and locks your API

Unsafe consumption of APIs (#10)

Prevention

Unsafe consumption of APIs (#10)

Prevention

- Validate all external data
- Whitelist redirect URLs
- Implement timeouts

Operational security gaps

Configuration and infrastructure vulnerabilities

Security misconfiguration (#8)

Insecure defaults and missing security hardening

Security misconfiguration (#8)

Insecure defaults and missing security hardening

- Default configurations
- Missing security updates
- Unnecessary features enabled
- Header misconfiguration (CORS, etc.)

Security misconfiguration (#8)

Insecure defaults and missing security hardening

- Default configurations
- Missing security updates
- Unnecessary features enabled
- Header misconfiguration (CORS, etc.)

Example: Error messages return stack traces

Security misconfiguration (#8)

Prevention

Security misconfiguration (#8)

Prevention

- Regular security auditing and updates
- Audit and remove unnecessary features
- For APIs against browser-based clients, implement CORS and security headers

Improper inventory management (#9)

Do you know your API?

Improper inventory management (#9)

Do you know your API?

- Old API versions still running
- Shadow APIs (undocumented endpoints)
- Non-production environments accessible

Improper inventory management (#9)

Do you know your API?

- Old API versions still running
- Shadow APIs (undocumented endpoints)
- Non-production environments accessible

Example: v1 API wasn't decommissioned

Improper inventory management (#9)

Prevention

Improper inventory management (#9)

Prevention

- Maintain API inventory/catalog
- API Gateway / automated discovery tools
- Retire old versions with clear timelines

Server side request forgery (#7)

API fetches remote resources without validation

Server side request forgery (#7)

API fetches remote resources without validation

- User-controlled URLs in API requests
- API fetches a remote resource from user-supplied URL
- Can access internal network endpoints

Server side request forgery (#7)

API fetches remote resources without validation

- User-controlled URLs in API requests
- API fetches a remote resource from user-supplied URL
- Can access internal network endpoints

Example: `/images?url=http://127.0.0.1:8080/metrics`

Server side request forgery (#7)

Prevention

Server side request forgery (#7)

Prevention

- Validate and sanitize URLs
- Whitelist for domains & media types, etc
- Disable HTTP redirection where possible
- Don't sent raw responses to clients

In Closing

OWASP API Security Top 10

- Authentication & authorisation failures
- Business logic & resource abuse
- Configuration & infrastructure vulnerabilities

Security requires

- Defense in depth
- Testing with the mindset of an attacker
- Ongoing attention

Resources

OWASP API Security Project website
owasp.org/www-project-api-security/

REST Security Cheat Sheet
cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html

API Security news
apisecurity.io

"Securing APIs isn't optional; it is the frontline defense for protecting data integrity and maintaining digital trust."

Randy Barr, Cequence Security

Thank you!

Rob Allen ~ akrabat.com