ENGAGEMENT CHECKLIST

OWASP API Security Top 10 Testing Checklist

A working checklist for testing an API against the OWASP API Security Top 10 (2023) and recording the result of each check.

Test represents the attacker's steps

Prove represents the controls to verify

Mark each Pass, Fail, or N/A and note your evidence.

TARGET / SYSTEM	ENVIRONMENT
TESTER	DATE
SCOPE / NOTES	VERSION / BUILD

> **DISCLAIMER!**
> Completing this checklist does not guarantee the API is secure. It catches the ten most common classes of failure; a real attacker chains findings in ways no list anticipates, and business logic flaws live in the gaps between categories. Treat it as a strong starting point and a record of what was checked, not a certificate of safety.

Compiled by Meli Imelda from the official OWASP API Security Top 10 2023

HOW TO MARK Pass: control present or test found nothing Fail: vulnerable or control missing N/A: not applicable to this API

API1 Broken Object Level Authorization [CRITICAL]

Can I reach another user's object by changing an ID?

CHECK	PASS	FAIL	N/A
TEST			
Run the two-account swap: as user A, take a request to A's object and swap in B's ID while keeping A's token Burp Repeater			
Notes:			
Enumerate sequential IDs over the ID position Burp Intruder, ffuf			
Notes:			
Attack unguessable IDs too: replay GUIDs that leak in other responses, logs or referrer headers Burp			
Notes:			
Test every verb and location: path, query, body, array-wrapped, JSON vs form Burp Repeater			
Notes:			
Confirm blind BOLA: a successful DELETE or PUT on another user's object with no data echoed Burp Repeater			
Notes:			
PROVE			
Centralized authorization enforces per-object ownership on every endpoint taking an object ID			
Notes:			
The check compares the authenticated user to the object's true owner server-side, never a client value			
Notes:			
Multi-tenant queries are scoped in the data layer (WHERE owner_id = session_user)			
Notes:			
Record IDs are random GUIDs, treated as defense in depth only			
Notes:			
Automated authorization tests run in CI per role and block the deploy on failure			
Notes:			

API2 Broken Authentication [CRITICAL]

Can I break, forge, or brute force my way into an account?

CHECK	PASS	FAIL	N/A
TEST			
Crack the JWT secret if HS256; a recovered secret forges any identity xjwt.io, jwt_tool, hashcat			
Notes:			

CHECK	PASS	FAIL	N/A
Try alg:none by stripping the signature xjwt.io, jwt_tool, Burp JWT Editor			
Notes:			
Tamper with a claim without re-signing (decode vs verify) xjwt.io, Burp JWT Editor			
Notes:			
Replay an expired token to check exp is validated Burp Repeater			
Notes:			
Credential-stuff and enumerate users; watch for differential errors Burp Intruder, ffuf			
Notes:			
Change email or password with no current-password confirmation Burp Repeater			
Notes:			
Treat forgot-password as a login endpoint; brute force it and test old versions Burp Intruder			
Notes:			
PROVE			
Standard vetted auth libraries; no hand-rolled token or password crypto			
Notes:			
Passwords hashed with bcrypt, scrypt or Argon2, with breached-password checks			
Notes:			
Server verifies JWT signature, rejects alg:none, validates exp, iss and aud			
Notes:			
Anti-automation on login and forgot-password: lockout, captcha, brute-force protection			
Notes:			
Re-authentication for email, password and MFA changes; MFA where possible			
Notes:			
No secrets in URLs; tokens invalidated on logout and rotated on password change			
Notes:			

API3 Broken Object Property Level Authorization [CRITICAL]

Does it leak fields I should not see, or accept fields I should not set?

CHECK	PASS	FAIL	N/A
TEST			
Inspect raw JSON, not the UI, for fields the client never shows Burp, Postman			
Notes:			

CHECK	PASS	FAIL	N/A
Fetch the same object as admin and as a low-priv user and diff the fields Burp Repeater			
Notes:			
Inject sensitive fields (is_admin, role, credit, balance) into a create/update request Burp Repeater			
Notes:			
Discover the full property set from Swagger, GraphQL introspection or an APK decompile JADX-GUI, Param Miner			
Notes:			
Re-fetch the object to confirm an injected value persisted Postman			
Notes:			
PROVE			
Responses built from explicit output DTOs; no entity-wide serialization			
Notes:			
A schema layer enforces which fields each endpoint may return			
Notes:			
Updates use an allowlist; price, role, ownership and status are never client bindable			
Notes:			
Sensitive fields gated by the caller's role, not just object ownership			
Notes:			

API4 Unrestricted Resource Consumption [HIGH]

Can I make it do expensive things without limit?

CHECK	PASS	FAIL	N/A
TEST			
Map the expensive endpoints: SMS, email, file generation, paid calls, uploads, pagination Burp			
Notes:			
Probe the rate limit: send rapidly and watch for a 429 Turbo Intruder, ffuf			
Notes:			
Bypass a weak limit: rotate X-Forwarded-For, change casing, add a trailing slash, rotate tokens Burp			
Notes:			
Brute force small secret spaces (OTP, PIN) where there is no limit ffuf, Burp Intruder			
Notes:			
Abuse pagination and batching: huge limit value, or many operations in one request Burp			

CHECK	PASS	FAIL	N/A
Notes:			
PROVE			
Container or serverless caps on memory, CPU and restarts			
Notes:			
Rate limiting global and per endpoint, keyed on user and IP, bypass-resistant			
Notes:			
Per-operation throttling, e.g. a cap on OTP validations			
Notes:			
Max string, array, nesting and upload size enforced server-side			
Notes:			
Pagination bounded; GraphQL depth and batching limits			
Notes:			
Spending limits or billing alerts on every paid integration			
Notes:			

API5 Broken Function Level Authorization [CRITICAL]

Can a normal user reach an admin function?

CHECK	PASS	FAIL	N/A
TEST			
Replay an admin request with the low-privilege token Burp Repeater, Autorize			
Notes:			
Method tampering: if you can GET a resource, try POST, PUT, DELETE Burp Repeater			
Notes:			
Guess admin routes and test each with the low-priv token ffuf, kiterunner			
Notes:			
Swap a single-object endpoint for the collection endpoint Postman			
Notes:			
Call admin GraphQL mutations as a normal user InQL			
Notes:			
PROVE			
Deny by default; every function requires an explicit role grant			
Notes:			
Admin controllers inherit role checks; admin functions in regular controllers check role too			
Notes:			
Authorization enforced on every HTTP method, not just GET			
Notes:			

CHECK	PASS	FAIL	N/A
Checks are role-based server-side, never inferred from the URL path			
Notes:			
The full route inventory, including legacy routes, is reviewed against the role matrix			
Notes:			

API6 Unrestricted Access to Sensitive Business Flows [HIGH]

Can I automate a business flow past its intended limit?

CHECK	PASS	FAIL	N/A
TEST			
Recon flows where volume equals value: checkout, reservations, referrals, sign-up, voting Burp			
Notes:			
Confirm the flow completes over the API alone, with no mandatory human step Burp Repeater			
Notes:			
Automate at scale and run the flow far past its intended limit Turbo Intruder, scripts			
Notes:			
Test the controls: captcha validated server-side, limits only per-IP, headless client blocked Burp			
Notes:			
PROVE			
Abuse-prone flows explicitly identified and threat-modeled			
Notes:			
Anti-automation on the API, not just the UI: captcha, device fingerprint, behavior analysis			
Notes:			
Limits keyed on more than IP (account, device, payment) so rotation does not defeat them			
Notes:			
Business guardrails: purchase caps, reservation holds, cancellation limits, sign-up throttling			
Notes:			
Monitoring on anomalous flow velocity			
Notes:			

API7 Server Side Request Forgery [CRITICAL]

Can I make the server fetch a URL I control?

CHECK	PASS	FAIL	N/A
TEST			
Find every URL-accepting input: webhooks, import-from-URL, avatar-by-URL, previews, SSO Burp, Param Miner			

CHECK	PASS	FAIL	N/A
Notes:			
Point it at an out-of-band listener to confirm the fetch and catch blind SSRF Burp Collaborator, interactsh			
Notes:			
Aim inward at loopback and internal ranges to enumerate services Burp Repeater			
Notes:			
Go for cloud metadata (169.254.169.254) for IAM credentials Burp Repeater			
Notes:			
Bypass filters: alternate IP encodings, inward redirects, credential-confusion URLs, file:// Burp			
Notes:			
PROVE			
The fetching component is network-isolated from internal ranges, loopback and metadata IP			
Notes:			
Allowlists for origins, schemes (http/https) and ports; media types validated			
Notes:			
HTTP redirects disabled so a whitelisted host cannot redirect inward			
Notes:			
A hardened parser validates the final resolved IP, with DNS-rebinding mitigation			
Notes:			
Raw upstream responses not returned to clients; metadata hardened (IMDSv2)			
Notes:			

API8 Security Misconfiguration [HIGH]

Are the headers, methods, errors and TLS locked down?

CHECK	PASS	FAIL	N/A
TEST			
Check security headers: HSTS, Cache-Control on sensitive responses, X-Content-Type-Options curl, nikto			
Notes:			
Test CORS: send an evil Origin and see if it is reflected with credentials allowed Burp, corsy			
Notes:			
Tamper with HTTP methods: OPTIONS, PUT, DELETE, TRACE nmap, Burp Intruder			
Notes:			

CHECK	PASS	FAIL	N/A
Force errors with malformed input; read stack traces and version banners Burp, ffuf			
Notes:			
Probe for exposed files and defaults: .git, .env, /actuator, /swagger, admin panels nikto, nuclei, dirsearch			
Notes:			
Test for injection where inputs are unvalidated; a verbose SQL error is the cue sqlmap			
Notes:			
Fingerprint components and cross-reference known CVEs nmap -sV, nuclei, whatweb			
Notes:			
PROVE			
An automated hardening baseline applied to every environment and re-checked			
Notes:			
Config reviewed across the stack: orchestration, gateway, proxies, cloud permissions			
Notes:			
TLS enforced everywhere with HSTS and modern ciphers			
Notes:			
Explicit per-endpoint verb allowlist; CORS never wildcard plus credentials			
Notes:			
Enforced response schemas including errors; no stack traces to clients			
Notes:			
Patch management and dependency scanning in CI; unnecessary features disabled			
Notes:			

API9 Improper Inventory Management [HIGH]

Are there old versions, staging hosts or shadow endpoints live?

CHECK	PASS	FAIL	N/A
TEST			
Enumerate versions and compare protections; the old version is usually the weak one ffuf, Burp Intruder			
Notes:			
Discover docs and schemas that reveal hidden endpoints (/swagger, /openapi.json, GraphQL) ffuf, nuclei, kiterunner			
Notes:			
Hunt shadow and zombie endpoints in historical URLs and JS bundles waybackurls, gau, LinkFinder			

CHECK	PASS	FAIL	N/A
Notes:			
Find non-production hosts via subdomain and certificate-transparency enumeration subfinder, amass, crt.sh			
Notes:			
Confirm retirement: deprecated endpoints really return 410/404 Burp Repeater			
Notes:			
PROVE			
A maintained, automated inventory of every host with environment, version, exposure, owner			
Notes:			
An inventory of every third-party integration with data exchanged and sensitivity			
Notes:			
Complete auto-generated docs kept in sync in CI, available only to authorized consumers			
Notes:			
A formal versioning and retirement policy; retired versions truly decommissioned			
Notes:			
Same controls on all versions and environments; no production data in non-prod			
Notes:			

API10 Unsafe Consumption of APIs [HIGH]

Does it blindly trust data from the APIs it consumes?

CHECK	PASS	FAIL	N/A
TEST			
Map the third-party integrations: enrichment services, webhooks, upstream partners code review, mitmproxy			
Notes:			
Check transport: confirm upstream calls use TLS with certificate validation mitmproxy, testssl.sh			
Notes:			
Inject through the upstream: plant SQLi/XSS in data the third party returns to you custom repo/profile			
Notes:			
Test redirect handling: make an upstream respond 308 to a host you control ngrok, Burp Collaborator			
Notes:			
Push resource limits: have the upstream return a huge or slow response mitmproxy			
Notes:			

CHECK	PASS	FAIL	N/A
PROVE			
Vendor security posture assessed before integrating			
Notes:			
All outbound calls over TLS with certificate validation			
Notes:			
Third-party data validated and sanitized before use; treated like user input			
Notes:			
No blindly followed redirects; an allowlist, with sensitive headers dropped cross-host			
Notes:			
Timeouts and response-size limits on every third-party interaction			
Notes:			