API rate limit bypass.

Rate limiting is the API control that limits how often a caller can hit an endpoint, used to prevent scraping, brute force, credential stuffing, and denial of service. Rate-limit bypass is the family of techniques attackers use to send more traffic than the limit was supposed to allow. The bypasses fall into a small number of patterns: counting at the wrong layer, identifying the wrong attribute, allowing batched requests through, and leaking limits to attackers who then time their bursts under the threshold.

HOW IT WORKS

01 What does a successful bypass enable?

- Account takeover at scale. Bypassing login rate limits enables credential stuffing across the user database.
- OTP brute force. Bypassing one-time-code rate limits enables brute-forcing of SMS or email codes (6-digit codes have one million possibilities; with no rate limit, that completes in minutes).
- Mass scraping. Bypassing read-endpoint limits enables harvesting the entire database.
- Denial of service. Bypassing expensive-operation limits enables resource exhaustion.
- Sensitive business flow abuse. Bypassing flow limits on purchases, transfers, or account creation enables the OWASP API6:2023 category.

02 How do you implement rate limits that hold?

- Layer the limits. Per-IP, per-account, per-API-key, and per-endpoint. The intersection makes a single bypass less useful.
- Limit at the origin. Edge rate limiting is useful for defense in depth, not the only layer. The origin should enforce limits regardless of edge.
- Verify the identity header. If you trust `X-Forwarded-For`, ensure it comes from a trusted proxy. Most CDNs strip and re-add the header so origin trust is safe; verify the configuration.

HOW TO DEFEND

- Credential stuffing. Without rate limits on login, an attacker can test millions of stolen username and password pairs against the live API at speed.
- Brute force. Without rate limits on OTP, password reset, or two-factor-code endpoints, attackers brute-force the code space.
- Scraping. Without limits on read endpoints, attackers harvest the entire dataset for resale or analysis.
- Resource exhaustion. Without limits on expensive operations, a single attacker can run up a cloud bill or take the service offline.

SOURCES

[1] OWASP API4:2023 Unrestricted Resource Consumption

[2] OWASP API6:2023 Unrestricted Access to Sensitive Business Flows

[3] OWASP Cheat Sheet on Authentication (rate-limit section)

- Track the right attribute. Login limits should track the username being tried, not just the source IP. Reset limits should track the account being reset, not just the requester.
- Account for batching. For batchable APIs (GraphQL, multi-call endpoints), limit at the operation level, not the HTTP request level.
- Lock progressively. First few attempts free, then progressive delay, then full lockout. Distinguishes legitimate user mistyping from attacker probing.
- Detect anomalies. Even with limits, watch for traffic that looks like distributed bypass: many sources hitting one account, one source hitting many accounts at low rate.

03 How does SecureLayer7 test rate limits?

Every API engagement runs the bypass matrix on sensitive endpoints (login, password reset, OTP, signup, expensive read, expensive write).

- IP rotation. Test with multiple source IPs (residential proxy, datacenter rotation).
- Header forgery. Test with `X-Forwarded-For-`, `X-Real-IP`, `X-Original-IP` variants.
- Origin bypass. Map the origin endpoint directly, test whether limits enforced at the edge are enforced there too.
- Batching. For GraphQL, test batched operations against the limit. For multi-call REST endpoints, test parallel requests against per-second limits.
- Path and method variation. Test trailing-slash, case, percent-encoding, and method variants.
- Identity variation. Test rotating API keys, accounts, and tokens against per-identity limits.

Deliverable maps findings to OWASP API4:2023 and API6:2023 with the specific rate-limit configuration change required.

Test your rate limits against the full bypass matrix.
`securelayer7.net/learn/api-security/rate-limit-bypass`

Open online