What is GraphQL penetration testing?

GraphQL penetration testing is the practice of attacking a GraphQL API the way a real attacker would. GraphQL is a query language and runtime that routes every API operation through one or two endpoints, with the caller specifying which fields to return. The flexibility is the design goal. It also introduces attack patterns REST does not have: introspection that hands the attacker the full schema, batched queries that bypass rate limits, deeply nested queries that exhaust resources, and authorization checks that have to live at the resolver level rather than the endpoint level.

HOW IT WORKS

01 What are the GraphQL-specific attacks?

- Introspection-driven enumeration. Pull the entire schema, identify endpoints, types, and arguments that look interesting, target them.
- Authorization at the resolver level. Test every resolver for object-level authorization (BOLA). The endpoint accepting the request says nothing about whether the specific field should return the requested data.
- Batching attacks. A single HTTP request can carry many GraphQL operations. Rate limits applied at the HTTP layer miss them. Useful for credential stuffing on login mutations.
- Depth attacks. Deeply nested queries that fan out into expensive joins. Variants: query a user's friends' friends' friends, recursively, until the resolver chokes.
- Alias-based abuse. GraphQL aliases let the same field be queried many times with different names in one operation. Bypasses naive rate limiting and per-field cost analysis.
- Mutation discovery. A schema browse often surfaces mutations the team did not realize were exposed: admin functions, internal-only flows, dangerous test mutations left in production.

02 How do you secure a GraphQL API?

- Disable introspection in production. Many GraphQL servers leave it on by default. Production introspection helps attackers, not legitimate users (legitimate clients have build-time schema access).

SOURCES

[1] OWASP GraphQL Cheat Sheet

[2] OWASP API Security Top 10 (2023)

[3] GraphQL Specification

- Authorize at the resolver level. Every resolver verifies the caller is allowed to access the specific object and field. A central authorization library applied via middleware is easier to audit than per-resolver checks.
- Query depth and complexity limits. Bound how deep a query can recurse and how expensive the total query cost can be. Several open-source libraries handle this.
- Persisted queries. Restrict production clients to a fixed allowlist of pre-registered queries. Defeats most ad-hoc abuse. Requires build-time tooling.
- Per-operation rate limiting. Limit at the operation level, not the HTTP request level, so batched requests are accounted for correctly.
- Field-level authorization audit. Run a regular check that every field in the schema has an authorization rule and the rule is the right one.

03 How does SecureLayer7 test GraphQL APIs?

Every GraphQL engagement runs through the standard matrix.

- Schema enumeration. Pull the schema via introspection. If introspection is off, fingerprint to identify the server implementation, then attempt schema discovery through error messages.
- Per-resolver authorization. For every interesting field, test cross-account access. Particularly important for fields that fan out (`user.invoices`, `organization.members`).
- Batching and alias abuse. Test rate-limit and authorization bypass via batched operations and alias-tricked queries.
- Depth and complexity. Test resource-exhaustion via nested queries, then test the team's complexity limits if present.
- Mutation surface. Enumerate every mutation. Many engagements find dangerous mutations the team forgot were exposed.

- Injection through arguments. Test SQL injection, NoSQL injection, and LDAP injection in resolver arguments. GraphQL syntax does not protect underlying queries.

Deliverable maps findings to the OWASP API Top 10 and to the OWASP GraphQL Cheat Sheet, with the specific resolver or configuration change required.

Test your GraphQL API against the full attack matrix.
securelayer7.net/learn/api-security/graphql-pentesting

Open online