What is insecure direct object reference?

Insecure direct object reference (usually written IDOR) is the vulnerability where the application identifies which resource to act on (an invoice, a user, a file) by an ID supplied in the request, but does not check whether the calling user is allowed to access that resource. Changing the ID to a different value returns somebody else's data. Despite being one of the simplest flaws to understand, IDOR is the highest-frequency authorization finding on most application engagements.

HOW IT WORKS

01 What do attackers do with IDOR?

Real exploit patterns we see on engagements:

- Read other users' data. Walk the ID space, dump every invoice, every message, every uploaded file.
- Modify other users' data. Change someone else's account settings, update their address, change a balance.
- Privilege escalation. Edit role assignments through an admin-only endpoint that forgot to check the caller's role.
- Workflow bypass. Mark another user's order as shipped, approve another user's expense, advance another user's record to a state it should not reach yet.
- Mass exfiltration. Scripted enumeration over the ID space pulls out the entire dataset in minutes.

02 What forms does IDOR take in practice?

Three patterns cover almost everything we find:

- Sequential numeric IDs in URLs. The classic case. The application uses auto-increment integers, and the only authorization check is the session cookie. Changing `?id=4287` to `?id=4288` returns somebody else's data.
- IDs in the request body. Same vulnerability, harder to spot because the ID is not visible in the URL bar. Common in PUT / POST endpoints that update objects.

HOW TO DEFEND

- Centralize the check. A single authorization layer that runs on every request and decides 'is this user allowed to act on this resource in this way' is dramatically easier to audit than per-endpoint checks scattered across the codebase.
- Make the check explicit in the data layer. Queries that load a resource should require the owner (or appropriate scope) as a parameter, not just the resource ID. `SELECT * FROM invoices WHERE id = ? AND organization_id = ?` is much harder to forget than checking the result after the fact.
- Test for it on every endpoint. Automated tests that try a wrong-user request on every authenticated route catch regressions early. Most teams discover that several endpoints they assumed were safe are not.
- Treat unpredictable IDs as a defense in depth. UUIDs and slugs reduce the cost of leaked IDs but do not replace authorization checks.

SOURCES

[1] OWASP A01:2021 Broken Access Control

[2] OWASP API1:2023 Broken Object Level Authorization

[3] CWE-639: Authorization Bypass Through User-Controlled Key

- Predictable but non-numeric IDs. UUIDs and slugs help when the IDs are unguessable. Many applications log or return them in places the attacker can read, so 'unguessable' becomes 'enumerable' fast. Treat unpredictability as a small bonus, not a fix.

Variants worth knowing: BOLA (Broken Object Level Authorization, the OWASP API Top 10 name for the same flaw), BFLA (Broken Function Level Authorization, when an entire admin function is reachable by a non-admin), mass assignment (when the request lets the user write fields the application did not intend to expose).

03 How does SecureLayer7 test for IDOR?

Every application engagement runs IDOR testing across the entire authenticated surface.

- Map the resource model. What are the objects (invoices, users, files, messages, roles)? Which endpoints read them, write them, or list them?
- Test cross-account access. For each endpoint, fire the same request with another user's resource ID. Document everything that returns data, mutates, or even errors in a way that confirms the resource exists.
- Test cross-role access. For each admin-only endpoint, fire the same request as a non-admin. Many BFLA findings hide on routes the team forgot to lock down.
- Test mass assignment. For each PUT / POST that updates an object, try adding fields the API did not advertise (`role: 'admin'`, `verified: true`, `organization_id: 999`).

Deliverable maps findings to OWASP A01:2021 (Broken Access Control) and OWASP API1:2023 (BOLA) with the realistic blast radius for each.

Find IDOR before someone enumerates your customer table.
securelayer7.net/learn/application-security/idor

Open online