# Broken Object-Level Authorization BOLA / IDOR

Attackers access other users' data by modifying object IDs.  
when an application exposes references to internal objects (such as files, database records, or IDs) without enforcing proper access control checks. Attackers manipulate these references to access data or actions not meant for them.

## Example:

**Horizontal Access (IDOR)**: GET /invoices/98567  
If User A changes the ID to /invoices/98568 and accesses another user’s invoice, that’s IDOR.   
  
**Accessing Other Users’ PII**: GET /user/profile/12345  
Change 12345 to 12346 to view/edit another profile.  
  
API Token Tampering: GET /api/v1/files/download?fileId=abcd1234  
Change fileId to guess or brute-force file IDs.  
  
**Mobile App or SPA**: Attacker manipulates JSON response or API request and directly accesses protected objects (especially common in RESTful APIs).

## Mitigation:

Enforce access control on server side based on user context.

## Impact

| **Type** | **Description** |
| --- | --- |
| **Data Breach** | Exposure of sensitive data (PII, health records, payments). |
| **Business Risk** | Compliance violations (GDPR, HIPAA, PCI-DSS). |
| **Privilege Escalation** | Gaining access to other users' assets. |
| **Reputation Damage** | Public breaches from IDOR/BOLA are common. |

# Types of BOLA / IDOR

## Horizontal Privilege Escalation (Classic IDOR)

* **Definition**: Access peer-level resources using object identifiers.
* **Example**: /user/record?id=122 → change to another user's ID.
* **Mitigation**: Validate ownership on server-side per session/user context.

## 2. Vertical Privilege Escalation (Elevated Object Access)

* **Definition**: Non-admin accessing admin-owned resources.
* **Example**: /admin/settings/102
* **Mitigation**: Role-based access check before object access.

## 3. Mass Assignment (Object Injection)

* **Definition**: Modifying multiple object properties via API that should not be allowed.
* **Example**: {"userId": 101, "role": "admin"}
* **Mitigation**: Use allow-lists to define updatable properties.

## 4. Reference Prediction

* **Definition**: Guessing predictable object IDs (e.g., incremental IDs).
* **Mitigation**: Use UUIDs or unpredictable tokens for references.

# Testing Methods

| Method | Tool | Description |
| --- | --- | --- |
| Manual ID Change | Burp Suite Repeater | Change object ID and observe response |
| AuthMatrix (Burp Plugin) | Role/object matrix testing | Automate role-object relationships |
| Postman | API token testing | Try different object IDs via API calls |
| Automated Scanning | OWASP ZAP | Detect basic IDOR patterns (limited) |

# Mitigation Strategies

| Area | Control |
| --- | --- |
| Enforce Object Ownership Checks | Every object access should validate user session and ownership. |
| Use Indirect Object References | Don’t expose database IDs directly; use UUID or hash. |
| Avoid Mass Assignment | Explicitly define which fields are editable. |
| Authorization Middleware | Centralize role/resource checks in backend logic. |
| Rate Limiting + Logging | Detect mass ID brute-force attempts. |

# Tools for Detection

| Tool | Use |
| --- | --- |
| Burp Suite | Modify IDs manually |
| Postman | API object access |
| AuthMatrix | Role-object mapping automation |
| OWASP ZAP | Scan parameter-based object access |
| Fiddler / Charles Proxy | Great for mobile/desktop app APIs |

# Points

“BOLA is the **#1 risk in OWASP API Top 10** because APIs expose object references by design.”

“Most IDOR flaws pass through automated scanners — **manual testing is essential**.”

“We encourage UUIDs and server-side ACL checks in our secure coding standards.”

“Use AuthMatrix in Burp Suite to test role-object access patterns systematically.”

# Notable Breaches (IDOR)

| Company | Year | Description |
| --- | --- | --- |
| Facebook | 2015 | Deleted any album via predictable object ID |
| Instagram | 2019 | Viewed private stories by modifying URL |
| T-Mobile | 2020 | Exposed user data via BOLA flaw in API |