INSECURE DIRECT

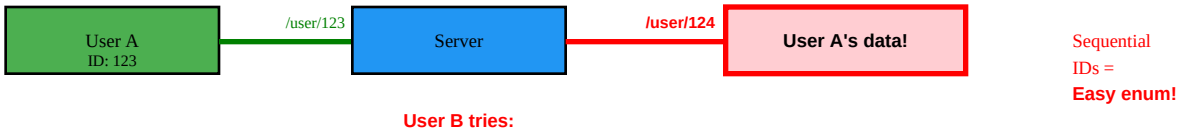
OBJECT REFERENCE

Complete IDOR Attack Reference

> **What is IDOR?**
> Insecure Direct Object Reference (IDOR) occurs when an application exposes direct references to internal objects (files, database records, URLs) without proper access control. Attackers manipulate parameters (IDs, filenames, keys) to access unauthorized resources. IDOR is consistently found in bug bounties due to its simplicity and high impact.

1. Numeric ID IDOR

Description: Most common IDOR type. Sequential numeric IDs easily guessable. User accesses /user/123, changes to /user/124 to view other user's data. No authorization check between user identity and requested resource ID.



Example Attack:

```
/api/user/1234 → /api/user/1235 | /order/5000 → /order/5001
```

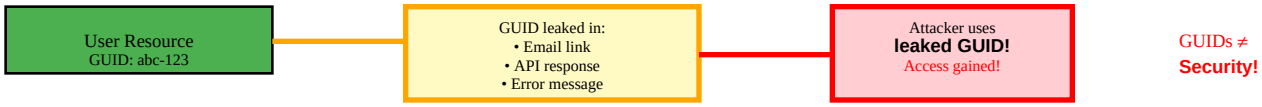
Attack Techniques:

```
Increment/decrement IDs | Try ID ranges | Brute force sequential IDs
```

> **Impact:** Unauthorized data access, privacy violation, account enumeration

2. GUID/UUID IDOR

Description: GUIDs appear random but may still be vulnerable. Check if GUID referenced elsewhere (emails, responses, logs). Try leaked/disclosed GUIDs. Some UUID versions (v1) contain timestamp and MAC address.



Example Attack:

```
/document/550e8400-e29b-41d4-a716-446655440000
```

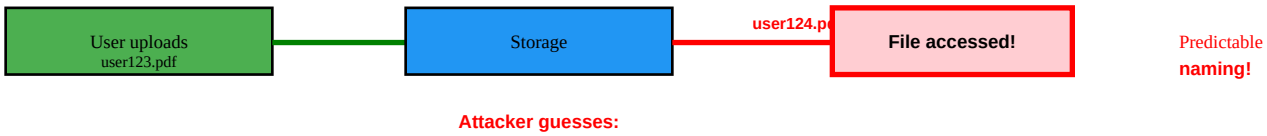
Attack Techniques:

```
Extract GUIDs from responses | Check UUID version | Try leaked identifiers
```

Impact: Access to 'protected' resources, false sense of security

3. Filename/Path IDOR

Description: Direct file references without access control. User uploads file as user123_doc.pdf, tries user124_doc.pdf. Directory traversal combined with IDOR. Predictable naming patterns exploitable.



Example Attack:

```
/download?file=invoice_123.pdf → invoice_124.pdf
```

Attack Techniques:

```
Modify filenames | Change user prefixes | Path manipulation | Pattern guessing
```

> **Impact:** Unauthorized file access, sensitive document disclosure

4. Email/Username IDOR

Description: APIs accept email or username instead of ID. Enumerate valid emails/usernames. No rate limiting enables mass data harvesting. Common in profile lookup, password reset, user search endpoints.

Example Attack:

```
/api/profile?email=victim@domain.com
```

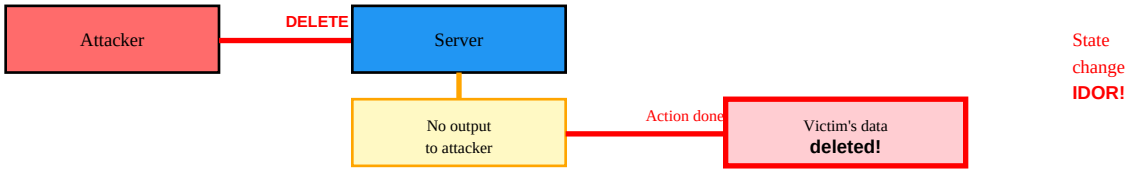
Attack Techniques:

```
Email enumeration | Username guessing | Common name lists | Company directory
```

> **Impact:** Profile disclosure, data scraping, targeted attacks

5. Blind IDOR

Description: No direct output but action succeeds. Change password for other user, delete other's resources, modify settings. Confirmation may be indirect (email notification, log entry, side channel).



Example Attack:

```
POST /api/user/456/delete (deletes other user's account)
```

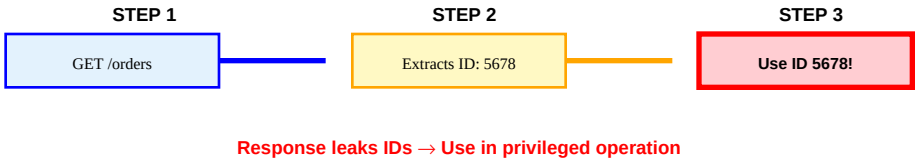
Attack Techniques:

```
State-changing operations without direct feedback | Monitor side effects
```

> **Impact:** Unauthorized modifications, data deletion, privilege escalation

6. Multi-Step IDOR

Description: First request appears authorized, subsequent uses leaked reference. Step 1: View own order, response contains other order IDs. Step 2: Use discovered ID in privileged operation.



Example Attack:

```
GET /orders (reveals all IDs) → POST /order/5678/refund
```

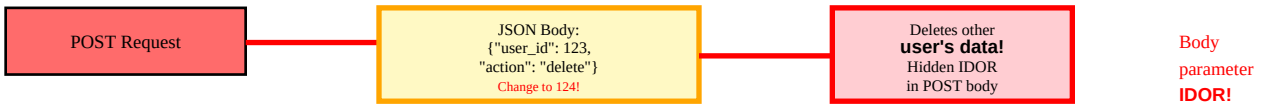
Attack Techniques:

```
Chain requests | Extract IDs from responses | Use in privileged operations
```

> **Impact:** Complex attack chains, bypasses simple access controls

7. JSON/POST Body IDOR

Description: IDOR in POST request bodies, not just URLs. JSON payload contains user_id, account_id, organization_id. Modify these values in request body. Less obvious than URL parameters.



Example Attack:

```
{"user_id": 123, "action": "delete"} → Change user_id to 124
```

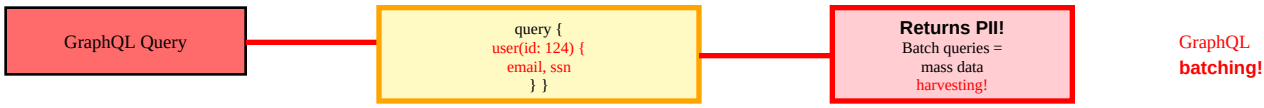
Attack Techniques:

```
Modify JSON/XML body IDs | Change nested object references | Array manipulation
```

Impact: Hidden IDOR vectors, bypasses URL-based protection

8. IDOR in GraphQL

Description: GraphQL exposes object references through queries. Query accepts IDs without proper authorization. Introspection reveals available IDs and relationships. Batching enables mass enumeration.



Example Attack:

```
query {user(id: 123) {email, ssn}} → Change id to 124
```

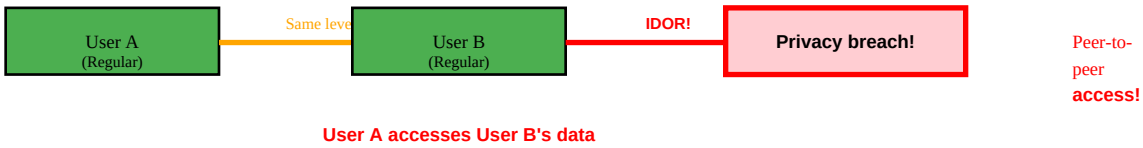
Attack Techniques:

```
Modify ID in GraphQL query | Batch queries | Introspection for schema
```

Impact: API data exposure, mass data harvesting via batching

9. Horizontal Privilege Escalation

Description: Access resources of users at same privilege level. User A views user B's orders, messages, documents. Most common IDOR scenario. Both users have same role but should only access own data.



Example Attack:

```
User views /profile/456 (another regular user's profile)
```

Attack Techniques:

```
Enumerate IDs of same-level users | Access peer resources
```

Impact: Privacy violation, data breach, competitive intelligence

10. Vertical Privilege Escalation

Description: Regular user accesses admin/privileged resources. User modifies admin_id parameter, accesses /admin/settings. More severe than horizontal. Leads to full system compromise.

Example Attack:

```
/api/user/123/role → Change to /api/admin/role
```

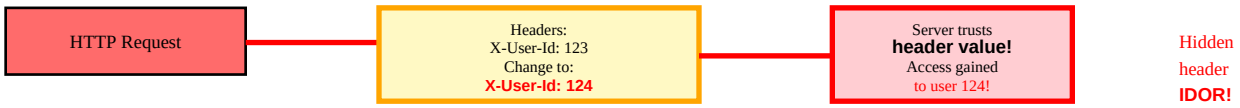
Attack Techniques:

```
Try admin IDs | Access privileged endpoints | Modify role parameters
```

Impact: Admin access, system compromise, full control

11. IDOR via HTTP Headers

Description: References in custom headers. X-User-Id, X-Account-Id, X-Organization headers trusted without validation. Less obvious attack vector. Often overlooked in security reviews.



Example Attack:

```
X-User-Id: 123 → Change to X-User-Id: 124
```

Attack Techniques:

```
Modify custom headers | Try common header names | Burp Suite header manipulation
```

> **Impact:** Hidden IDOR, bypasses request body/URL protection

12. IDOR in Mobile APIs

Description: Mobile app APIs often less protected than web. Hardcoded API keys, weak authentication. Decompile app to find endpoints. Mobile-specific IDORs in push notifications, deep links, SDK calls.

Example Attack:

```
Mobile API: /api/v1/user/profile?uid=123
```

Attack Techniques:

```
Intercept mobile traffic | Decompile app | Test discovered endpoints
```

> **Impact:** Mobile user data exposure, app-specific vulnerabilities

HIGH-VALUE IDOR TARGETS

User Data Endpoints

- /api/user/{id} - User profiles and personal information
- /api/user/{id}/settings - Account configuration
- /profile/{username} - Public/private profiles
- /account/{id}/details - Account details
- /api/user/{id}/addresses - Shipping/billing addresses
- /user/{id}/payment-methods - Saved credit cards
- /api/user/{id}/notifications - User notifications

Document & File Endpoints

- /download?file={filename} - File downloads
- /api/document/{id} - Document access
- /files/{user_id}/{filename} - User files
- /invoice/{id} - Invoice/receipt access
- /report/{id}/download - Report generation
- /api/attachment/{id} - Email/message attachments
- /media/{id} - Images, videos, uploads

Financial & Transaction Endpoints

- /api/order/{id} - Order details and history
- /transaction/{id} - Transaction records
- /payment/{id}/details - Payment information
- /invoice/{id} - Billing invoices
- /subscription/{id} - Subscription details
- /refund/{id} - Refund requests
- /api/wallet/{id}/balance - Wallet/balance info

Communication Endpoints

- /api/message/{id} - Private messages
- /conversation/{id} - Chat conversations
- /api/email/{id} - Email content
- /ticket/{id} - Support tickets
- /api/comment/{id} - Comments (edit/delete)

- `/notification/{id} - User notifications`

Administrative Endpoints

- `/admin/user/{id} - Admin user management`
- `/api/admin/logs/{id} - System logs`
- `/admin/organization/{id} - Org settings`
- `/api/admin/reports - Administrative reports`
- `/admin/settings/{id} - System configuration`

API-Specific Patterns

- `/api/v1/users/{id}/data - RESTful patterns`
- `?user_id={id} - Query parameters`
- `user={id} - Simple parameter names`
- `{"user_id": 123} - JSON body references`
- `X-User-Id: {id} - Custom headers`

IDOR TESTING METHODOLOGY

1. Identify Object References: Map all parameters that reference objects: user IDs, document IDs, order numbers, filenames. Look in URLs, POST bodies, JSON, headers, cookies. Create list of all potential IDOR points.

2. Create Multiple Test Accounts: Register 2-3 accounts with different privileges. User A (victim), User B (attacker), Admin account if possible. Test access between accounts systematically.

3. Map Access Patterns: As User A, access own resources and note IDs/references. Log all object identifiers visible to User A. Document which resources should be private.

4. Test Cross-Account Access: Log in as User B. Try accessing User A's resources by changing IDs. Test both read and write operations. Document successful unauthorized access.

5. Test Sequential IDs: If IDs are numeric, test ranges. Try ID-1, ID+1, ID+10, ID+100. Use Burp Intruder for automated enumeration. Monitor for successful responses.

6. Check Response Differences: Compare authorized vs unauthorized responses. Look for: HTTP status codes (200 vs 403), response size, error messages, timing differences. Blind IDOR may show subtle hints.

7. Test All HTTP Methods: Try GET, POST, PUT, DELETE, PATCH on resources. Sometimes GET is protected but POST isn't. Test OPTIONS to discover allowed methods.

8. Manipulate Request Components: Test IDORs in: URL path, query parameters, POST body, JSON/XML, HTTP headers, cookies. Don't assume protection on one means all are protected.

9. Check for Indirect References: Look for leaked references in responses. Search for IDs in: HTML comments, JavaScript variables, API responses, error messages. Use these in subsequent requests.

10. Test Privilege Escalation: Try accessing admin endpoints with regular user credentials. Modify role/privilege parameters. Test /admin/, /api/admin/, /management/ paths.

IDOR PREVENTION

• **Implement Proper Authorization:** ALWAYS verify user has permission to access requested resource. Check object ownership at application layer. Never rely solely on obscurity of IDs. Authorization should be on every request.

• **Use Indirect References:** Map internal IDs to session-specific references. User session contains mapping of accessible resources. User sees reference A which maps to internal ID 123. Different user gets different reference for same resource.

• **Apply Access Control Lists (ACLs):** Maintain explicit ACLs for resources. Database stores who can access what. Check ACL on every resource access. Deny by default, allow explicitly.

• **Use Resource-Based Authorization:** Check: Does current user own this resource? Does user have required role? Is resource in user's organization/group? Implement at data access layer.

• **Avoid Predictable Identifiers:** Use UUIDs instead of sequential IDs. Makes enumeration harder (but not impossible). Still need authorization checks. UUIDs are not security controls.

• **Implement Rate Limiting:** Limit number of requests per user per time period. Prevents mass enumeration. Alert on suspicious patterns. Block after threshold exceeded.

• **Minimize Data Exposure:** Return only necessary data. Filter response based on user permissions. Don't include other users' IDs in responses. Avoid exposing internal identifiers.

• **Audit and Monitor:** Log all access attempts to sensitive resources. Alert on unusual access patterns. Monitor for horizontal/vertical privilege escalation attempts. Regular security audits.

IDOR DETECTION CHECKLIST

✓ Map all endpoints that accept ID parameters

✓ Create at least 2 test accounts (different users, same privilege)

✓ Test numeric ID enumeration (increment, decrement)

✓ Try accessing other user's resources by ID manipulation

✓ Test both GET and POST/PUT/DELETE methods

✓ Check for IDOR in URL parameters, POST body, headers

✓ Test file download/upload endpoints with different filenames

✓ Look for GUIDs/UUIDs leaked in responses

✓ Test email/username as direct references

✓ Check for blind IDOR (state changes without output)

✓ Try admin/privileged IDs as regular user

✓ Test multi-step operations (extract IDs, use in next step)

✓ Monitor response codes, sizes, timing for differences

✓ Test GraphQL queries with different user IDs

✓ Check mobile API endpoints separately

✓ Use Burp Suite Autorize extension for automated testing

✓ Test with different session cookies simultaneously

✓ Look for IDORs in API documentation/Swagger

IDOR TESTING TOOLS

Tool	Purpose	Use Case
Burp Suite	Proxy, repeater, intruder	Manual IDOR testing, enumeration
Autorize Extension	Automated authz testing	Compare responses across users
AuthMatrix Extension	Session comparison	Multi-user authorization matrix
Postman	API testing	Test RESTful API IDORs
FFUF	Fuzzing	Enumerate IDs at scale
Arjun	Parameter discovery	Find hidden ID parameters
GraphQL Voyager	GraphQL schema viz	Identify object relationships

Note: This cheat sheet is for educational and authorized security testing only. Unauthorized access to accounts and data is illegal. Always obtain written permission before testing.