Web Application Security

Dr. Digvijaysinh Rathod

Professor

School of Cyber Security and Digital Forensics

National Forensic Sciences University

digvijay.rathod@nfsu.ac.in

Insecure Direct Object References (IDOR)

IDOR

- ✓ Insecure Direct Object Reference (IDOR) occurs when an application **exposes a reference to an internal implementation object—such as a file, database record, directory, device identifier**, or key—without adequate access control.
- ✓ Attackers can **manipulate this reference** to gain unauthorized access, modify, or delete data belonging to other users or systems.

IDOR

- ✓ Formally, OWASP classifies IDOR under Broken Access Control (A01:2021).
- ✓ It represents a failure to enforce proper authorization before allowing access to backend objects or resources.

IDOR - Core Concept

- ✓ In secure applications, every resource request should be mediated by **authorization logic** that verifies the requester’s rights.
- ✓ In an IDOR-vulnerable system, object references are exposed directly and **trust the client-supplied identifier** (e.g., numeric IDs, filenames, record keys).

IDOR - Example

https://forensiclab.in/report?case_id=102

- ✓ If the server does not verify whether the logged-in user actually owns or is authorized for case_id=102, an attacker could change the value to:

  https://forensiclab.in/report?case_id=103

and download another investigator’s confidential forensic report.

IDOR - Typical Manifestations

Context	Example of Reference	Vulnerable Behavior
Web App / API	/user/1234/profile	Attacker modifies 1234 to view others' data
IoT / Device Endpoint	/device?id=5	Returns telemetry for any device ID without ownership check
File System	/download?file=report .pdf	Any file path accepted (path traversal)
Database Record	SELECT * FROM cases WHERE id = ?	Query uses attacker-supplied ID without session validation

IDOR - Impact

- **Confidentiality breach:** Exposure of private reports, logs, or evidence files.
- **Integrity violation:** Attackers alter data belonging to other users (e.g., forensic records).
- **Account compromise:** Enumeration of user IDs or session tokens.
- **Legal risk:** Breach of evidentiary confidentiality in LEA or judicial processes.

IDOR -Causes

- Client-controlled object references (predictable IDs, file names, UUIDs).
- Missing authorization checks on each object request.
- Assumption of obscurity (believing that sequential IDs are “hidden”).
- Improper session-to-object mapping.

IDOR -Mitigation & Best Practices

Control	Description
Indirect References	Use mapping tables or opaque tokens instead of predictable IDs. Example: /report?token=F2A9C7... instead of /report?id=102.
Authorization Checks	Validate user/session permissions server-side before granting access to any object.
Least Privilege Principle	Limit each account to only its own resources.
Input Validation & Logging	Reject unexpected object references and log all access attempts.
Secure API Design	Employ resource-based access controls (RBAC/ABAC) for REST APIs.
Encryption & Tokenization	Hash or encrypt internal identifiers when exposed externally.
Testing During Development	Integrate IDOR test cases into continuous integration (CI) pipelines.

Example – Secure Implementation

- Insecure Code:

```
case_id = request.GET['id']

report = db.get("SELECT * FROM reports WHERE id =

%s" % case_id)

return report
```

Example – Secure Implementation

- Secure Code:

```
case_id = request.GET['id']

if not user.is_authorized_for(case_id):

    raise AccessDenied()

report = db.get_secure_report(user, case_id)

return report
```

Mobile Phone Security

Dr. Digvijaysinh Rathod
Professor
School of Cyber Security and Digital Forensics
National Forensic Sciences University

digvijay.rathod@nfsu.ac.in