IDOR

Overview

Room URL: https://tryhackme.com/room/[room-name]

Difficulty: Easy

Category: Web Application Security/Access Control

Date Completed: [Date]

Objective

Learn what IDOR (Insecure Direct Object Reference) vulnerabilities are, how to identify them through various obfuscation methods, and how to exploit them to access unauthorized data.

Table of Contents

Introduction

What is IDOR?

IDOR stands for **Insecure Direct Object Reference** and is a type of access control vulnerability.

This vulnerability occurs when a web server:

1. Receives user-supplied input to retrieve objects (files, data, documents)
2. Places too much trust in that input data

3. Fails to validate on the server-side whether the requested object belongs to the user requesting it

Why IDOR Matters

IDOR vulnerabilities can lead to:

- Unauthorized access to other users' data
- Privacy breaches and data leaks
- Horizontal privilege escalation (accessing data of users at the same privilege level)
- Vertical privilege escalation (accessing admin or higher-privilege data)
- Financial fraud and identity theft

Real-World Example

Scenario:

You sign up for an online service and navigate to your profile:

```
http://online-service.thm/profile?user_id=1305
```

You can see your information. Out of curiosity, you change the `user_id` parameter to `1000`:

```
http://online-service.thm/profile?user_id=1000
```

Result: You can now see another user's information!

The Problem: The website doesn't verify that the requested user information belongs to the authenticated user making the request.

Detection Method 1: Encoded IDs

What is Encoding?

When passing data between pages (via POST data, query strings, or cookies), web developers often encode raw data to ensure the receiving server can understand the contents.

How Encoding Works:

Encoding converts binary data into an ASCII string using characters like:

- a-z, A-Z, 0-9
- `=` for padding

Most Common Encoding: Base64

Base64 encoding is easily recognizable and can be identified by:

- Length of the string
- Character set used (alphanumeric + `/` and `+`)
- Padding with `=` characters

Testing for IDOR with Encoded IDs:

1. Identify a base64 encoded ID in the URL, cookie, or POST data
2. Decode the value using https://www.base64decode.org/
3. Modify the decoded value (change user ID, etc.)
4. Re-encode using https://www.base64encode.org/
5. Submit the modified request
6. Check if you can access unauthorized data

Example:

```
Original: user_id=MTIzNDU=
Decoded: 12345
Modified: 12346
Re-encoded: MTIzNDY=
```

Detection Method 2: Hashed IDs

What are Hashed IDs?

Hashed IDs are more complex than encoded IDs, but they may follow predictable patterns. The hash is often the hashed version of an integer value.

Common Pattern:

Sequential integer IDs hashed with MD5, SHA-1, or other algorithms.

Example:

```
ID: 123
MD5 Hash: 202cb962ac59075b964b07152d234b70
```

Testing for IDOR with Hashed IDs:

1. Identify what appears to be a hashed ID
2. Try to determine the hashing algorithm (length can provide clues)
3. Use hash lookup services like https://crackstation.net/
4. If you find the original value, try adjacent values (ID+1, ID-1)
5. Hash those values with the same algorithm
6. Submit requests with the new hashed IDs

Hash Length Indicators:

- 32 characters: Likely MD5
- 40 characters: Likely SHA-1
- 64 characters: Likely SHA-256

Tools:

- CrackStation - Database of billions of hashes
- HashCat - Offline hash cracking
- Online MD5/SHA generators

Detection Method 3: Unpredictable IDs

When to Use This Method:
When IDs cannot be detected or predicted using encoding or hashing methods (e.g., UUIDs, random strings).

The Technique:
Create two separate user accounts and swap their ID numbers between them.

Test Process:

1. Create Account A and note its ID
2. Create Account B and note its ID

3. While logged in as Account A, try to access Account B's resources using Account B's ID
4. If you can view Account B's content while authenticated as Account A, you've found an IDOR vulnerability

Why This Works:

Even with unpredictable IDs, the application may still fail to verify that the authenticated user has permission to access the requested resource.

Variations:

- Test while logged in with different accounts
- Test while not logged in at all
- Test with session tokens from different users

Where IDORs Are Located

IDOR vulnerabilities aren't always obvious. They can be hidden in various places:

1. URL Parameters

```
/profile?user_id=123
/document?doc_id=456
```

2. AJAX Requests

Content loaded asynchronously via JavaScript that isn't visible in the address bar.

3. API Endpoints

RESTful APIs often use IDs in the path or parameters:

```
/api/v1/customer?id=123
/api/users/456/details
```

4. JavaScript Files

Referenced endpoints in client-side JavaScript code.

5. POST Data

IDs sent in form submissions or API requests.

6. Cookies

Session or reference IDs stored in cookies.

Parameter Mining

What is it?

Discovering unreferenced parameters that may have been useful during development but were pushed to production.

Example:

```
Normal endpoint: /user/details (displays your info via session)
Discovered parameter: /user/details?user_id=123 (displays any user's info)
```

How to Find Hidden Parameters:

- Use tools like Burp Suite's Param Miner
- Try common parameter names (id, user_id, uid, account_id, etc.)
- Review JavaScript files for API calls
- Check API documentation if available
- Analyze network traffic in browser developer tools

Walk Through

Task 1: Understanding IDOR

Question: What does IDOR stand for?

Answer: `Insecure Direct Object Reference`

Task 2: Basic IDOR Exploitation

Objective: Find and exploit an IDOR vulnerability on the example website

1. Click the "View Site" button
2. Explore the website and look for user-specific endpoints
3. Identify URLs or API calls with ID parameters

4. Modify the ID parameter to access other users' data
5. Capture the flag

Answer: `THM{REDACTED}`

Task 3: Encoded IDs

Question: What is a common type of encoding used by websites?

Answer: `base64`

How to Identify Base64:

- Look for strings ending in `=` or `==`
- Alphanumeric characters with `/` and `+`
- Can be decoded at https://www.base64decode.org/

Task 4: Hashed IDs

Question: What is a common algorithm used for hashing IDs?

Answer: `md5`

MD5 Hash Characteristics:

- Always 32 characters long
- Hexadecimal (0-9, a-f)
- Can be looked up at https://crackstation.net/

Task 5: Unpredictable IDs

Question: What is the minimum number of accounts you need to create to check for IDORs between accounts?

Answer: `2`

Testing Process:

1. Create first account (note the ID)
2. Create second account (note the ID)

3. While logged in as one account, try accessing the other's resources

---

Task 6: Locating IDORs

Objective: Understand that IDORs can be in various locations including API endpoints and hidden parameters

Read and understand where IDORs might be located beyond just the URL bar.

Task 7: Practical IDOR Exploitation

Setup:

1. Press "Start Machine" button
2. Navigate to the lab URL: https://LAB_WEB_URL.p.thmlabs.com
3. Click on "Customers" section
4. Create an account
5. Log in with your new account

Step 1: Navigate to Your Account

1. Click on "Your Account" tab
2. Notice your username and email are pre-filled

Step 2: Investigate the API Call

1. Open browser Developer Tools (F12)
2. Go to the "Network" tab
3. Refresh the page
4. Observe the API call to: `/api/v1/customer?id={user_id}`

Step 3: Analyze the Response The endpoint returns JSON format containing:

- User ID
- Username
- Email address

Step 4: Test for IDOR

1. Note the `id` parameter in the query string
2. Modify the `id` value to test other user IDs
3. Try `id=1` to see user 1's information
4. Try `id=3` to see user 3's information

Answers:

What is the username for user id 1? `adam84`

What is the email address for user id 3? `j@fakemail.thm`

Lessons Learned

- IDOR is an access control vulnerability where user-supplied input is trusted without server-side validation
- The fundamental problem is lack of authorization checks on the server side
- IDORs can occur in URL parameters, API endpoints, AJAX requests, and hidden parameters
- Base64 encoding is NOT security - it's easily reversible and commonly used in IDOR vulnerabilities
- Hashed IDs may seem secure but can follow predictable patterns or be cracked using rainbow tables
- Even unpredictable IDs (UUIDs) can be vulnerable if the server doesn't verify authorization
- Always check API calls in browser Developer Tools (Network tab) for potential IDOR endpoints
- Testing with multiple accounts is essential for identifying IDOR vulnerabilities
- Parameter mining can reveal hidden parameters that weren't properly secured
- JSON responses from APIs often expose more data than intended
- Proper IDOR prevention requires server-side authorization checks for every resource access
- Never rely solely on obscurity (encoding, hashing) for access control
- Sequential IDs make IDOR testing easier but even random IDs aren't protection without proper authorization
- Browser developer tools are essential for discovering API endpoints and hidden requests

---

Resources

TryHackMe

Base64 Decode

Base64 Encode

CrackStation - Hash Lookup

OWASP - Insecure Direct Object References

PortSwigger - Access Control Vulnerabilities

Burp Suite - Param Miner Extension

OWASP API Security Top 10

Revision #2

Created 2026-01-21 23:02:14 UTC by David Rizzo

Updated 2026-01-21 23:25:56 UTC by David Rizzo