TryHackMe Advent of Cyber 2025

Day 5 Challenge Report

Insecure Direct Object References (IDOR)

1. Executive Summary

This report documents the completion of Day 5 of the TryHackMe Advent of Cyber 2025 event. The challenge focused on Insecure Direct Object References (IDOR), a critical access control vulnerability. Through practical exploitation using browser developer tools, successfully demonstrated horizontal privilege escalation by accessing unauthorized user data and identified multiple IDOR variants including base64 encoding, hashing, and UUID-based vulnerabilities.

2. Challenge Overview

Objective: Understand authentication and authorization concepts, identify IDOR vulnerabilities, exploit IDOR for horizontal privilege escalation, and learn proper remediation techniques.

Vulnerability Type: Access Control - Insecure Direct Object Reference

3. Theoretical Foundation

3.1 Understanding IDOR

IDOR (Insecure Direct Object Reference) is an access control vulnerability where web applications use references to determine data retrieval but fail to verify user authorization before returning that data. When users can simply modify identifiers (like user IDs, file paths, or database keys) to access resources belonging to other users, an IDOR vulnerability exists.

Common Example:

```
https://awesome.website.thm/TrackPackage?packageID=1001
```

Alternative Terminology:

Some security professionals prefer the term **Authorization Bypass** as it more accurately describes the root cause. The vulnerability isn't about direct object references themselves, but rather the missing authorization checks. Hiding or encoding IDs (e.g., /user/ea21f09b2) doesn't fix the issue if the server still doesn't verify permissions.

3.2 Authentication vs Authorization

Understanding the distinction between these concepts is fundamental to comprehending IDOR vulnerabilities.

Authentication:

The process of verifying identity. Users provide credentials (username/password) and receive session information (cookie/token). This verification occurs on every subsequent request, not just at login. Session expiration requires reauthentication.

Authorization:

The process of verifying permissions. Determines what authenticated users can access or modify. Examples include admin page access or payment authorization for specific accounts.

Critical Principle: Authorization cannot occur before authentication. The application must know user identity before verifying permissions. If IDOR doesn't require authentication, authentication must be fixed before addressing authorization issues.

3.3 Privilege Escalation Types

Vertical Privilege Escalation:

Gaining access to additional features or elevated permissions. Example: normal user performing administrator actions.

Horizontal Privilege Escalation:

Using authorized features to access unauthorized data at the same privilege level. Example: viewing other users' accounts when you should only see your own. ***Most IDOR cases fall into this category.***

4. Practical Exploitation

4.1 Basic IDOR - Direct User ID Manipulation

Used browser Developer Tools to intercept and analyze application requests, revealing the mechanism for data retrieval.

Investigation Process:

1. Opened Developer Tools (Right-click → Inspect → Network tab)
2. Refreshed page to capture requests
3. Analyzed view_accountinfo request
4. Identified user_id parameter with value 10

Exploitation Method:

5. Navigated to Storage tab in Developer Tools
6. Expanded Local Storage dropdown
7. Located auth_user data entry
8. Modified user_id value from 10 to various test values
9. Refreshed page to view other users' data

Result: Successfully accessed other users' account information without authorization, demonstrating the simplest form of IDOR vulnerability.

Answer - User with 10 Children: User ID 15

4.2 Encoded IDOR - Base64 Obfuscation

IDORs aren't always obvious. Applications may encode identifiers to obscure direct references, creating a false sense of security.

Example:

```
Original: /user/2
Encoded: /user/Mg==
```

The value **Mg==** is simply base64 encoding of the number 2. Attackers can still exploit IDOR by encoding sequential numbers.

Key Insight: *Encoding is obfuscation, not security. Without proper authorization checks, encoded parameters remain vulnerable.*

4.3 Hashed IDOR - MD5/SHA1 References

Some applications use hash values as identifiers, appearing more secure than sequential numbers. However, if the hashing algorithm and input are predictable, the vulnerability persists.

Hashed identifiers may be MD5 or SHA1 values. If attackers can predict or enumerate possible inputs, they can generate valid hashes and exploit the IDOR vulnerability.

4.4 UUID-Based IDOR - Algorithmic Weakness

The most sophisticated IDOR variant encountered involved UUIDs. While appearing random, the underlying algorithm revealed exploitable patterns.

Investigation:

- Voucher codes used UUID format
- Used UUID Decoder to identify version
- Discovered UUID version 1 in use

UUID Version 1 Vulnerability:

UUID v1 incorporates timestamp and MAC address. If generation time is known, UUIDs can be reconstructed. For instance, if vouchers are generated between 20:00-21:00, attackers can generate 3,600 UUIDs (60 minutes × 60 seconds) for brute force attacks.

Attack Vector: Generate all possible UUIDs for known time window and attempt to redeem valid vouchers, bypassing intended access controls.

5. Remediation Strategies

Proper IDOR prevention requires server-side authorization checks, not reliance on obscurity or encoding.

5.1 Core Principles

1. **Server-Side Verification:** Verify user authorization on every request
2. **Permission Checks:** Confirm logged-in user owns or has permission for requested resource
3. **Avoid Security Through Obscurity:** Don't rely on Base64, hashing, or encoding alone
4. **Session Association:** Link resources to authenticated user sessions
5.

5.2 Implementation Best Practices

- **Use Unpredictable Identifiers:** Random IDs for public links, but maintain authorization checks
- **Testing Protocol:** Attempt accessing other users' data during testing
- **Monitoring:** Log and monitor failed access attempts
- **Indirect Reference Maps:** Use session-specific mappings instead of direct database IDs
- **Framework Security Features:** Leverage built-in authorization mechanisms

6. Key Takeaways

- IDOR is an access control vulnerability exploiting missing authorization checks
- Most IDOR cases result in horizontal privilege escalation
- Authorization must follow authentication
- Encoding, hashing, or obfuscation don't fix IDOR
- Server-side authorization checks are mandatory
- Even sophisticated identifiers (UUIDs) can be vulnerable
- Testing must verify unauthorized access is blocked

7. Conclusion

Day 5 of the TryHackMe Advent of Cyber 2025 provided comprehensive training in identifying and exploiting Insecure Direct Object References. Through practical browser-based exploitation, demonstrated multiple IDOR variants and horizontal privilege escalation techniques.

The challenge reinforced that proper access control requires server-side authorization verification on every request. No amount of encoding, hashing, or identifier complexity can substitute for verifying that authenticated users have permission to access requested resources.

Challenge Status: **COMPLETED ✓**