Chained Exploitation: IDOR and Sensitive Data Exposure leading to Administrative Takeover

Project Information

- **Analyst:** Mduduzi William Radebe
- **Date:** 07 January 2026
- **Platform:** PortSwigger Web Security Academy
- **Vulnerability Type:** CWE-639 (IDOR) and CWE-312 (Cleartext Storage)
- **Severity: Critical (9.8/10)**

1. Executive Summary

This report documents a critical multi-stage attack. By exploiting an **Insecure Direct Object Reference (IDOR)** vulnerability, I gained unauthorized horizontal access to the administrator's account page. Once accessed, I identified a **Sensitive Data Exposure** vulnerability where the user's password was pre-filled in the HTML source code. By chaining these flaws, I performed a full **Account Takeover (ATO)** and executed restricted administrative commands.

2. Technical Analysis & Discovery

Initial Identification (IDOR)

While logged in as a standard user (wiener), I analyzed the account management URL:

GET /my-account?id=wiener

The use of a predictable, plaintext username in the id parameter suggested a lack of server-side authorization. I hypothesized that changing this value to administrator would allow me to view the admin's private profile.

Sensitive Data Exposure (Credential Leak)

Upon navigating to the administrator's profile, I observed a password "Change" field. Standard security practices dictate that password fields should be empty or masked. However, many legacy applications "pre-fill" these fields from the database.

3. Exploitation Methodology (The Attack Chain)

Step 1: Horizontal Escalation

I manipulated the request parameter to target the administrative user:

URL: https://[LAB-ID].web-security-academy.net/my-account?id=administrator

The server failed to verify if the requester (wiener) had permission to view the administrator object, granting me access to the page.

Step 2: Static Analysis & Data Exfiltration

I inspected the DOM (Document Object Model) of the administrator's account page. While the browser UI showed masked dots (••••••••), the source code revealed the password in plain text.

Vulnerable HTML Snippet:

HTML
<input type="password" name="password" value="8807z95vspm7kti20ovt">

I successfully exfiltrated the administrative password: 8807z95vspm7kti20ovt.

Step 3: Vertical Escalation (Account Takeover)

I logged out of my standard session and re-authenticated using the stolen administrative credentials. This elevated my privileges from a **Standard User** to a **Superuser**.

Step 4: Action on Objectives

With full administrative control, I navigated to the restricted /admin panel and deleted the user carlos, fulfilling the lab objective and demonstrating total system compromise.

4. Root Cause Analysis

Attack Stage	Failure Point	Root Cause
Stage 1 (IDOR)	Authorization	The server does not perform an "Ownership Check" on the id parameter.
Stage 2 (Leak)	Data Handling	Sensitive credentials are sent to the client-side within the HTML body.
Stage 3 (ATO)	Authentication	Lack of Multi-Factor Authentication (MFA) allowed the stolen password to be used immediately.

5. Remediation & Hardening Recommendations

A. Implement Strict Object-Level Authorization

The server must validate that the id in the URL matches the User_ID stored in the current session.

- **Secure Logic:** IF (Session_User != Request_ID) { Return 403 Forbidden }

B. Eliminate Sensitive Data Pre-filling

Password fields should **never** be pre-filled with the current password in the HTML value attribute. If a user wants to change their password, they should be required to provide the current one in an empty field.

C. Implement Multi-Factor Authentication (MFA)

Even if a password is leaked via IDOR, an MFA requirement (like an SMS code or Authenticator App) would prevent the attacker from completing the Account Takeover.

6. Conclusion

The "Web Security Academy" lab highlights how minor vulnerabilities can be chained to cause a catastrophic security breach. By failing to secure a single URL parameter, the application exposed the most sensitive secret in its database: the admin password. This proves that **Defense in Depth** is essential—relying on a single layer of security is never enough.