[Lab Report] Vertical Privilege Escalation via Client-Side Cookie Manipulation

Project Information

- **Analyst:** Mduduzi William Radebe
- **Date:** 07 January 2026
- **Platform:** PortSwigger Web Security Academy
- **Vulnerability Type:** CWE-807: Reliance on Untrusted Inputs in a Security Decision
- **Severity: Critical (9.0/10)**

1. Executive Summary

This report details the identification and exploitation of a critical **Broken Access Control** vulnerability. The application was found to store the user's authorization status (Administrator vs. Standard User) within a client-side cookie. By manually tampering with this cookie, I successfully bypassed the application's security logic, elevated my privileges to "Administrator," and performed restricted management actions.

2. Technical Analysis & Discovery

Initial State Observation

Upon logging in with a standard account (wiener), I analyzed the application's state-tracking mechanism. Most modern applications track roles on the server-side, but this application was observed using a client-side flag.

Parameter Audit

Using the browser's Developer Tools (Application Tab), I audited the cookies associated with the current session.

Findings:

- **Cookie Name:** Admin
- **Current Value:** false

The Architectural Flaw:

The presence of a boolean Admin cookie suggests that the backend code is performing a check similar to:

if (request.cookies['Admin'] == 'true') { showAdminPanel(); }

This constitutes a "Trusting the Client" anti-pattern. The server is abdicating its responsibility to verify the user's role against a secure database.

3. Exploitation Methodology (Proof of Concept)

Step 1: Privilege Forgery

I manually manipulated the session state by changing the Admin cookie value from false to true. This was done directly in the browser's storage manager.

Step 2: Forced Browsing

With the forged authorization flag set, I attempted to access the restricted administrative endpoint:

https://[LAB-ID].web-security-academy.net/admin

Step 3: Verification of Access

The server successfully validated the forged Admin=true cookie and granted full access to the **Administrator Panel**. I was presented with the user management interface, which is normally hidden from standard users.

Step 4: Unauthorized Action (Impact Confirmation)

To prove the vulnerability's impact, I executed a "Delete" request for the user carlos. The server processed the request without further verification, confirming that the cookie was the **only** check preventing administrative actions.

4. Root Cause Analysis

Component	Status	Security Risk
Data Integrity	**Failed**	The application does not sign or encrypt the Admin cookie, allowing for trivial modification.
Trust Model	**Insecure**	The server trusts user-supplied input to define the user's privilege level.
Authorization	**Broken**	Role-based access control (RBAC) is enforced on the client-side rather than the server-side.

5. Remediation & Hardening

A. Server-Side Role Mapping (Primary Defense)

Authorization levels must be stored in a secure server-side database. When a user logs in, the server should look up the role associated with the **Session ID** and verify permissions on every request.

B. Cryptographic Signing (Defense in Depth)

If user roles *must* be stored in a cookie (e.g., in a JWT), the cookie must be **cryptographically signed** by the server. This ensures that if a user changes false to true, the signature will become invalid and the server will reject the request.

C. Use of 'HttpOnly' and 'Secure' Flags

While they don't prevent manual tampering by the user, these flags prevent scripts from accessing cookies, providing another layer of security against XSS-based cookie theft.

6. Conclusion

This exploit demonstrates a complete breakdown of the **Principle of Least Privilege**. By relying on client-side cookies for security decisions, the application allows any user to become a full administrator in seconds. This vulnerability emphasizes that **all security-sensitive decisions must be made on the server**, where the user cannot interfere with the logic.