Broken Access Control: Low-Privilege User Can Edit Other User's Settings and Permissions

Download as PDF | Move to In Progress

Test program

Details | Edit

- **Vulnerability Name**: Broken Access Control
- **Vulnerable Asset/Endpoint**: https://sample.com/directory/{UUID}/settings/user
- **Vulnerability Category**: Broken Access Control (OWASP A01:2021)

Vulnerability General Overview

Access control, also called authorization, is a security measure that makes resources available to users that should have access to those resources and denies access to users who should not have access. For example, a user may access a secure web application and authenticate themselves by logging in. After authentication, when the user tries to access a resource, the access control policy checks whether the user is authorized to use the requested resource.

Potential Impact

Broken access control occurs when an issue with the access control enforcement allows a user to perform an action outside of the user's limits. For example, an attacker may be able to exploit a flaw in an application with the intention of gaining elevated access to a protected resource to which they are not entitled. As a result of the privilege escalation, the attacker can perform unauthorized actions. If applicable, the actual impact of this vulnerability can be found in the steps to reproduce section of this report.

Steps To Reproduce

1. Login to https://sample.com/ as a Low-Privilege user with only "View" permissions for all functionalities.
2. Browse to "Settings" - "Edit settings"
3. Click "Edit" on any existing user, edit the name and role permissions, and save the changes.
4. Notice this action is successful via the following HTTP request:

Request

```
POST /api/directory/{UUID}/settings/user HTTP/2
Host: sample.com

{"id":123123,"email":"sample123@iaminspectiv.com","name":"Sample
User","password":"","status":"1","data":{"34":[1]}}
```

Response

```
HTTP/2 200 OK
Date: Tue, 14 Jan 2025 01:13:12 GMT

{"code":200,"message":"Permissions updated successfully"}
```

5. Open a new browser and log in to the same merchant profile as the Admin user.
6. Browse to the same "Settings" location, and notice that the changes have been made by the Low-Privilege "View" only user.
7. Low-Privilege users can use this exploit to perform unauthorized edit actions to other users.

Remediation Recommendations

- Except for public resources, deny by default.
- Implement access control mechanisms once and re-use them throughout the application, including minimizing Cross-Origin Resource Sharing (CORS) usage.
- Model access controls should enforce record ownership rather than accepting that the user can create, read, update, or delete any record.
- Unique application business limit requirements should be enforced by domain models.
- Disable web server directory listing and ensure file metadata (e.g., .git) and backup files are not present within web roots.
- Log access control failures, alert admins when appropriate (e.g., repeated failures).
- Rate limit API and controller access to minimize the harm from automated attack tooling.
- Stateful session identifiers should be invalidated on the server after logout. Stateless JWT tokens should rather be short-lived so that the window of opportunity for an attacker is minimized. For longer lived JWTs it's highly recommended to follow the OAuth standards to revoke access.

References

- https://owasp.org/Top10/
- https://owasp.org/Top10/A01_2021-Broken_Access_Control/
- https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Transaction_Authorization_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

Details

Created 1/16/2025

Status New

Severity High