Project Report

Computer Data Security

Laim Mayron

**Introduction:**

Security Vulnerability is basically a security flaw in a application, which maybe design or implementation bug, which allows an attack cause security damage to customers and organization. **Broken Access Control** is the one of the major and most crucial security vulnerability in OWASP (Open Worldwide Application Security Project) (OWASP, n.d.).

**Overview:**

Broken Access Control is the most critical security flaw that allows unauthorized users to view sensitive organization files and modify data, delete the data which they shouldn’t have access to. Which is occurred due to improper management of sessions, improper enforcement of role-based access controls. Which allows attacker to easily bypass the authorization and modify the data like he was a legitimate user/admin (Maric, 2023).

**Key Findings:**

Throughout my studies in computer data security and my experience as a software developer, I have learned that broken access control is a highly rated security flaw in modern computer applications. There are two key points in a website’s security: authentication and authorization. It mainly categorized as vertical and horizontal vulnerabilities. vertical means it’s a completely unauthorized, and horizontal means user logged in into another users credential and doing admin level changes even though he don’t have access, which is a security error in code level. While working on my final project, which involved a student results portal, I encountered a security issue related to role-based authorization, a type of broken access control vulnerability.

Additionally, another concern is session hijacking, which involves stealing an admin's session to make unauthorized changes at the administrative level. During my research, I also identified an issue known as Insecure Direct Object Reference (IDOR). This vulnerability allows someone to access other users' information by simply altering a user ID in the URL, such as “fau/userdata/1234”. It’s clear that including user IDs in URLs is not a secure practice.

**What did you do?**

I am going to explain total broken access control with a simple project. Basically, it’s a student’s result portal application.

For step 1: I am going to show Broken Access Control vulnerability in my own project. I have created a simple project; I am going to create Role-based access vulnerability in that project. Which means how the attacker steals or modifies the data without authorization as admin. And I am going to find out the issue background, by testing the application regarding the Broken Access Control Vulnerability.

Tools Using: React JS (Front-end), Node JS (Server Side)

Vulnerability Fixing Process: Here in this step, I am going fix that Broken Access Control, role-based authorization issue. By providing straight authorization between users and admins. And providing authorization for different URL endpoints.

**Background (References Summary)**

Through various blogs and websites, research papers I found over 94% of application tested in the form any broken control access issue with the average incident rate of 3.81% and maximum incident rate of 55.97%. I also observed year 2021 statistics mostly web applications of E-commerce sectors (27.91% incident rate) are affected by broken access control, like unauthorizedly access to cart and order item. And second most is Education (25%) sector fallowed y Government and private organizations (OWASP, n.d.)

A table with numbers and text

Description automatically generated

Fig: Analysis of Broken Access Control in each sector (‘Hassan et al’., 2018).

And moreover there more companies suffered in this vulnerability like google amazon and flip cart. And recently one American Financial Company leaked morethan 800 million rela-estate documents (Zhong ,2023)..

A close-up of a text

Description automatically generated

Fig: Security incidents caused by broken access control in Real World (Zhong ,2023).

So, all studies argue that this vulnerability occurs by improper session and role-based access handling, but we are not completely blaming developer’s and designers, but due to lack of awareness in regarding harmful broken access control (Vaadata, 2023).

**System Design and Implementation:**

**A diagram of a student's results

Description automatically generated**

**A diagram of a user access

Description automatically generated**

Fig: Broken Access Control User and Admin Access relation (Sert, 2021).

A block diagram that depicts the implementation of student result portal where

**Access Control Design:**

Access controls are in place at the initial login but not consistently enforced at each transaction level, leading to potential elevation of privilege.

**BAC Vulnerabilities or BAC Attack:**

There are no server-side checks to verify if the requesting user matches the student ID in the session, allowing for unauthorized data access through URL manipulation.

**Role Verification Absence**: Once logged in, the application assumes the user's role based on initial authentication and does not re-verify the role for subsequent requests.

A student who is authenticated could potentially send a request to edit grades, and due to the lack of role re-verification, the server processes the request as if it were from a teacher. So student can able to edit his grade (Snyk, n.d.).

**Mitigation/Defense:**

Enforced role verification at every transaction.

implemented non-sequential and non-predictable identifiers for resources.

Added server-side checks to match session IDs with the requested resources to prevent unauthorized access.

**Discussion/What Happened (my project):**

A screenshot of a computer

Description automatically generated

**Without Role Based Authentication:**

app.post("/marks/teachers",(req,res)=>{

marks=req.body

res.send("Marks Updated Successfully")

})

A screenshot of a computer

Description automatically generated

Student can able to update his grades successfully

I provided only one type of attack in my project there would be more attacks scenarios like Vertical Privilege Escalation, Direct Object References, nsecure Direct Object References (IDOR) (Ali, 2023).

app.post("/marks/teachers",(req,res)=>{

if(req.headers.usertype==="teacher")

{

marks=req.body

res.send("Marks Updated Successfully")

}

else{

res.status(401).send("You are unauthorised to access this")

}

})

A screenshot of a computer

Description automatically generated

Additionally there are more mitigation scenarios like Limiting CORS Usage, Enable permission based access control, and continuous monitoring and continuous testing and deny access by default (Roy, 2022).

**Conclusion:**

Throughout this course and broken access Control Vulnerability,

* Understand How Secure Coding Practices are essential to develop application.
* Understanding Testing. How Effective can security Testing reduce and prevent Some of the vulnerabilities.
* Understanding Authentication vs Authorization Concept.
* Security Monitoring and continuous role verification in a large organization.

And Iam going to change and modify existing project. In a way that not only broken access control, I need to control my application from aa security vulnerabilities.

And In future i will do research employ new different mitigation strategies, Encourage the audience to consider security as an integral part of user experience and business logic.

Additionally in future we may see more advanced and AI based monitoring and testing. Which can helps to reduce this security attacks even more.

**References:**

OWASP. (n.d.). Vulnerabilities. Retrieved from <https://owasp.org/www-community/vulnerabilities/>

Maric, N. (2023, December 29). Broken Access Control: Attack Examples and 4 Defensive Measures. Brightsec <https://brightsec.com/blog/broken-access-control-attack-examples-and-4-defensive-measures/>

Hassan, Md Maruf & Ali, Md. Asraf & Bhuiyan, Touhid & Sharif, Md Hasan & Biswas, Saikat. (2018). Quantitative Assessment on Broken Access Control Vulnerability in Web Applications.

OWASP. (2021). A01:2021 - Broken Access Control. Retrieved from <https://owasp.org/Top10/A01_2021-Broken_Access_Control/>

Zhong, L. (2023). A Survey of Prevent and Detect Access Control Vulnerabilities. University of California, San Diego. Retrieved from <https://arxiv.org/pdf/2304.10600.pdf>

Sert, Y. E. (2021, August 20). A comprehensive guide to broken access control. PurpleBox Security. <https://medium.com/purplebox/broken-access-control-f82235ddf888>

Ali, Z. (2023, July 22). Broken Access Control: Unraveling Web Application Vulnerabilities and Mitigation Strategies. LinkedIn. <https://www.linkedin.com/pulse/broken-access-control-unraveling-web-application-mitigation-zahid-ali/>

Roy, D. (2022, January 14). How to Prevent Broken Access Control. GeeksforGeeks. <https://www.geeksforgeeks.org/how-to-prevent-broken-access-control/>

Snyk. (n.d.). Broken access control. <https://learn.snyk.io/lesson/broken-access-control/>

Vaadata. (2023, March 31). OWASP Top 10 #1: Broken Access Control and Security Tips. Retrieved April 8, 2024, from <https://vaadata.com/blog/owasp-top-10-1-broken-access-control-and-security-tips/>