Received: 2 August 2024 | Revised: 15 January 2025 | Accepted: 18 February 2025 | Published online: 11 March 2025

Archives of Advanced Engineering Science
2025, Vol. 00(00) 1–6
DOI: 10.47852/bonviewAAES52024016

RESEARCH ARTICLE

Access Restricted: A Study of Broken Access Control Vulnerabilities

Sadeeq Jan[1],*, Safi Ullah Khan[1], Abdul Wahab[1] and Dr. Mohammad[2]

[1]*Department of Computer System Engineering, University of Engineering and Technology Peshawar, Pakistan*

[2]*Department of Computer Science & Information Technology, University of Engineering and Technology Peshawar, Pakistan*

Abstract: Broken access control vulnerability is ranked No.1 in OWASP Top 10 list in 2021. This means that it is the most commonly used exploited weakness in the web applications by attackers today. Because if an attacker can exploit this vulnerability, they can gain control rights and potentially compromise the entire web application. From that point forward, the attacker can execute various attacks depending on their objectives. That's what makes it the most purposeful. In this research, we will reveal the security vulnerability of the access control in a web applications system. We will explore case studies of real-world attacks that leverage broken access control, providing a contextual understanding of the impact and implications of these vulnerabilities. Through this research, we aim to contribute to the ongoing efforts to enhance web application security and mitigate the risks associated with broken access control vulnerabilities. We will examine vulnerabilities in web applications that attackers can use to compromise access. Finally, we discuss the protection and security measures that should be taken against attackers who use this vulnerability.

Keywords: broken access control, OWASP Top 10, web application security, vulnerability exploitation, attack vectors, security vulnerabilities, access control mechanisms

1. Introduction

Due to the rapid evolution of modern technology, businesses are adapting their service delivery methods to meet the increasing expectations and ever-changing behaviors of consumers. Businesses in different field now rely on web applications to improve their activities like managing their logistics, interacting with customers, and organizing their workforce. By implementing session management capabilities, web applications can securely and efficiently respond to diverse service requests from authorized users. Generally, the application sessions are launched by authenticating the users with methods like username and password enabling it to provide a good customer service and a good environment to the customers, making them feel comfortable and satisfied while accessing only the permitted resources. Access control ensures that access to resources like web pages and tables of the database is limited and securely managed and securely configured to prevent unauthorized access by those who are logged in [1]. Using web applications to handle business tasks is an important milestone in modern business history, and it also increases the risk of loss if there is no good development in applications due to continuous design, development, and coding [2].

Evidence from OWASP now shows that (BAC) broken access control vulnerabilities have been marked as a Category 1 vulnerability [3] due to its presence and negative consequences in a Web application. Creating Weaknesses in the access control part of the web application design (such as misunderstandings, leakage of sensitive information, miscommunication, readable text, etc.,) can result in higher permissions for general users or logged-in systems. Exploiting the BAC bug can cause severe issues for the web application disruptions, such as preventing unauthorized access to administrative area authorization section, complete disruption of the website request, etc. An unusual authentication and access control vulnerability was listed as one of the OWASP 2021 Top Ten vulnerabilities that accommodates problems which have been discovered in state-of-the-art apps like IIS and WordPress [4, 5]. Major vulnerabilities are injection (SQL), cross-site scripting (xss), local file inclusions, and remote file inclusions. These vulnerabilities often result from the incorrect implementation of user authentication protocols [6, 7]. According to Ahmed et al. [8], managing active sessions is one of the two most dangerous problems.

Many studies have investigated the problem of user authentication and access in the protection system. SQL Injection (SQLi), Broken Authentication, Control, and Cross-Site Scripting (XSS) are common web application vulnerabilities. In this, we discuss the level of analysis of the process and propose guidelines for developers to secure Web applications [9]. Research is conducted through root cause analysis to identify governance interactions and evidence gaps, and solutions are provided to mitigate re-attacks behind web applications [10]. Vulnerability recognition, attack techniques, and documented techniques have been defined to protect the website from intruders [11]. This article introduces Nemesis technology to prevent access control vulnerabilities and exploit authentication issues in web applications. The author used this solution as a tool with which developers can

*__Corresponding author:__ Sadeeq Jan, Department of Computer System Engineering, University of Engineering and Technology Peshawar, Pakistan. Email: sadeeqjan@uetpeshawar.edu.pk; safimohmand34@gmail.com

check the presence of security vulnerabilities in a short time [12]. Studies describe authentication issues and attack control such as the destruction of web applications. At the end of this study, preventive measures for the problem in question are also mentioned [10, 13].

A detailed comparison of the Attribute-Based Access Control (ABAC) model with a well-documented version of ABAC model. This study presents the logical framework of ABAC and security protocols for operationalizing access control to web services, which is outlined in Anas et al. [7]. One method introduces FIX ME UP, a revolutionary tool that finds access control check errors and generates candidate fixes and has been evaluated by ten globally surveyed PHP applications [14, 15]. Following the security model to depend on all types of authentications and access control threats, tests confirmed that the model can secure applications around the world [3]. Many studies describe and investigate different types of Web access control. It introduces control code that focuses on Extensible Access Control Markup Language rules, which have become the basis for determining and controlling access to various uses and available services. Web Standards for management strategies are based on computational methods. Access security techniques have been studied [14, 16]. Using data from Windows 98 and NT 4.0, Alhazmi and colleagues proposed two models for the process of detecting vulnerabilities. This study focuses on the speed of software flaws leading to vulnerabilities, based on data from five different versions of windows and two versions of red hat linux [17].

When I reviewed the above content, I found that the research work on BAC is very insufficient. This article examines and analyzes the weaknesses of the Management Approach, the reasons for its various categories, and the implementation process. Risk factors for causes of BAC were also identified. Chapter 4 discusses the analysis results. Finally, the article concludes on the importance of research and future work.

2. Methodology

2.1. Data collection from different sources

400 websites were analyzed to conduct an experiment where dork was used to collect samples from search engines. Dork is a technique that utilizes advanced search operators to help user to locate exact information on the Internet [18]. The following queries were used:

1) php inurl:admin/config.php
2) inurl:admin/ControlPanel.php
3) “inurl:admin/login.php”
4) “inurl:site/backup.zip”
5) “inurl:site/db.sql”

The above Google dork syntax will be different depending on the specific requirements and different search engines (e.g., Yahoo, DuckDuckGo, Bing). Once the first location is found, it is sent to the first stage to ensure that a negative BAC is present.

2.2. Preprocessing stage

Data preprocessing is very important to create a real usable data set. If the data are not analyzed or analyzed properly, the raw data are incomplete, inconsistent, and altered. After getting our selected list of websites from the Dork output, we checked them with the four methods used to detect the presence of BAC on these applications.

2.3. Adjust access

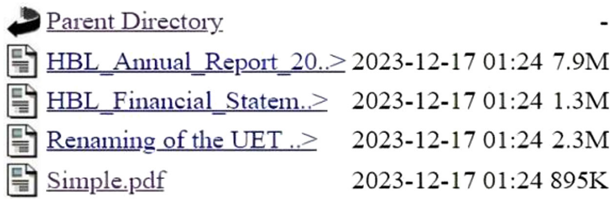
Separating admin site pages from the user general panel to prevent unauthorized access is a best practice for web application designers/developers [19]. However, except for some exceptions found in this study all users are able to access the admin sections without any restrictions. The following is an example of a BAC vulnerability in a web application that accesses a sensitive page without requiring a session.

The below code is vulnerable because it lacks session validation and authorization checks, allowing unauthorized users to access and view sensitive admin details. Implementing proper authentication and authorization measures is essential to prevent such unauthorized access.

```
<?php // No Access Control Check
    include “config.php”;
    include “db_connection.php”;
    // Fetch admin level details
    $result1 = mysqli_query($conn, “SELECT*FROM admin”) if
($result1) {
    while ($row = mysqli_fetch_assoc($result1)) { echo “Admin ID: ”.
$row[“admin_id”].
    echo “Admin Name: ”. $row[“admin_name”]
    }}?>}
```

In the above statement, the SQL query is executed to fetch admin details. Without proper access control checks, this query allows any user, including unauthorized ones, to retrieve sensitive admin information from the database. Figure 1 illustrates a broken access vulnerability where a user has accessed a restricted folder without any access controls in place. The absence of proper access restrictions allows unauthorized users to retrieve the sensitive information.

Figure 1
User access to restricted folder without restriction



```
Parent Directory                                          -
HBL_Annual_Report_20..>  2023-12-17 01:24  7.9M
HBL_Financial_Statem..>  2023-12-17 01:24  1.3M
Renaming of the UET ..>  2023-12-17 01:24  2.3M
Simple.pdf               2023-12-17 01:24  895K
```

2.4. Cookie-based access control vulnerability

Permission controls users’ level of access to all applications and limits their permissions on resources. Unauthorized access to a web application occurred due to an invalid session in a web application code. Sample PHP code below:

Figure 2 shows the PHP code vulnerabilities in access control. The session check for “member” is bypassed after the initial check, allowing unauthorized access to the users. The use of “mysqlquery” without proper parameter sanitization exposes the application to SQL injection attacks. A new session is created in the third line of this code. Once authenticated, it will allow the user to access the “login.php” page. Line 8 contains configuration information without restrictions. In line 11, any user can delete or modify the information without permission. Attackers often set the “SuperAdmin” Cookie ID value to the administrator account ID (e.g., ID = 1) and then alter the admin value in Sections 2, 3, 4, etc. While browsing the cookie file, the user/attacker can change

Figure 2
PHP code vulnerability: Broken access control

```
<?php session_start (); ?>

// Checking For Access Validity
if(!$_SESSION['member']) {

    header('Location:login.p
    hp'); exit;

}

// Control Access in
required herr. # Include
'inc/config.php';

# Include 'inc/conn.php';

// Not secure Database Connections for
     accessing Data

$del1 = mysql_query("DELETE FROM close_bid
     WHERE item.name = '" . $item_name .
     "'");

if($delete1) {
    mysql_close($conn);

}

?>
```

the ID; for example, the value might change from ID = 1009 to ID = 1. If the session is not defined correctly on the admin page, this can lead to massive changes in accounts, resulting in all permissions being assigned to ID = 1 (MainAdmin). It was determined that the number of web applications affected by BAC in the sample was 129, with 4 main risk factors. These risks are classified according to the importance of some independent variables, which are the main causes of access control (BAC) vulnerabilities: sensitive data, exceptional redirection, incorrect session configuration, language usage, operating system, and server/platform. All information listed has been verified by the Internet Security Center DIU.

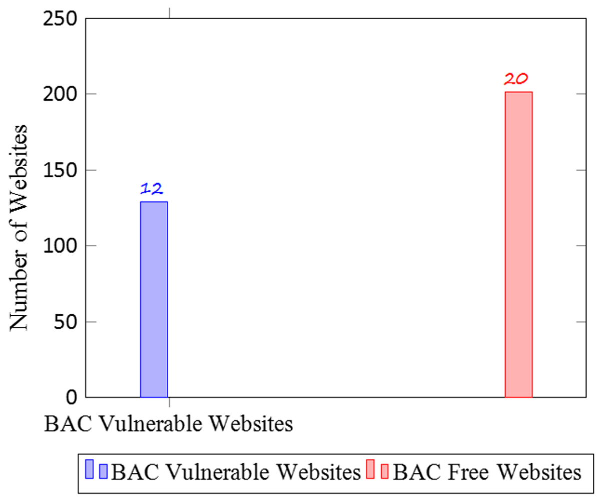
2.5. Quantitative examination of processed data

The main purpose of this study is to determine the relationship between various factors and BAC severity and to identify important factors. After preprocessing the collected data, the correlation and *p*-value between BAC and various variables were evaluated using the χ^2 test. Binary logistic regression was used for significant features (p¡0.05), and the odds ratio (OR) was calculated with a 95% confidence interval. The Pearson χ^2 test was then used to determine correlations between factors. All analyses were performed using IBM Statistical Package for the Social Sciences.

3. Result

This study used a small sample technique to determine its sampling method. The given method has been formulated using the below Equation (1):

Figure 3
The figure outlines the sample percentage, distinguishing between BAC-free and those with BAC vulnerabilities



$$S = \frac{y^2 + Mp(1-p)}{D^2(M-1) + y^2 p(1-p)} \tag{1}$$

In this equation, “S” denotes the required sample size, “M” represents the population size, “p” is the population proportion, “D” stands for the degree of accuracy that is expressed as a proportion, and “Y ” is the table value of chi-square for 1 degree of freedom at 3.84₽ (desired confidence level). We used G*Power statistical software to determine the sample size for our study using the above equation. A linear multiple regression test was used with the family of F tests; in our case, the number of predictors was set to 4 because the primary predictors in the testing models are the types of exploitations. The α error probability was set at 0.05, and power (1-β error probability) was fixed at 0.95 in the tool. According to its results, at least 129 valid samples should be taken.

Figure 3 represents the percentage of samples between BAC Bugs free and vulnerable websites. Among the 400 samples, 37.09% of web applications were affected by BAC vulnerability, while the remaining 63.91% were observed to be free from BAC Bugs. To achieve this, they used redirection settings, sensitive data retrieval misconfiguration, and illegal cookie access. This study employed the manual penetration testing method which used a double-blinded strategy to gather information. The initial analysis of this dataset was done with an emphasis on demographics and sectors. Pearson χ^2-value, binary logistic regression, *p*-value tests, and OR were also employed to perform analyses that focus on BAC reasons, exploitation techniques, platforms as well as applications hosted in operating systems. The following are the findings from the analysis.

Table 1 was analyzed for frequency in our sample: education, e-commerce, government, health, and private companies. The study shows that among these sectors, it is the e-commerce web applications that are most vulnerable—compromising 27.91.

We have 400 web applications under consideration, among which the BAC vulnerability affected 37.09.

The details of BAC web application vulnerabilities are illustrated in Table 2 showing the different levels of risks for

Table 1
BAC vulnerability across five sectors: Frequency analysis

Sector	Frequency	Percentage
Educations	35	25.58%
E-commerce	35	27.91%
Government Counterpart	30	21.71%
Health	20	7.75%
Private Company	25	17.05%
Total	**145**	**100.00%**

Table 2
Frequency distribution and *p*-value analysis of probability: BAC causes, exploits, markets, platforms, and operations associated with BAC vulnerabilities in web applications

Factors	Found	Not found	***P*-value**
Reason of BAC			
Improper Input Validation	71	79	0.021*
Sensitive Data Disclosure	10	67	
Session Misconfiguration	48	10	
Directory Readable	0	45	
Exploitation Techniques			
Access on Redirection	60	109	0.000*
Misconfig. of Sensitive Data	42	74	
Unauthorized Cookie Access	27	18	
Sectors			
Education	33	97	0.917
E-commerce	36	38	
Govt. Counterpart	28	20	
Health	10	8	
Private Company	22	38	
Platform			
PHP	113	65	0.000*
Java	4	37	
.NET	12	99	
OS			
UNIX	72	101	0.000*
Windows	23	38	
Cent-OS	34	62	

BAC vulnerabilities. The results indicate that there is a relationship between "Exploitation Technology," "Platform", and "Operating System" with BAC vulnerabilities at a very high level of significance (p ¡ 0.0000). Moreover, the analysis also points out that "Causes of BAC" have some association with BAC vulnerabilities but not the sectors themselves (p ¡ 0.021). As depicted in the table, it can be seen that ".NET" and "Java" platforms have significant links (p ¡ 0.05) with BAC vulnerabilities in web application model—a distinguishing feature: an indication to consider during analysis.

Table 3 shows that the main reason for BAC vulnerability is "Input validation", "sensitive information leakage", "session configuration error", "redirect access settings", "sensitive information configuration error", and other "BAC vulnerabilities exploiting technology"; Development platforms "UNIX", "Windows", and "Cent-OS" operating systems such as "Java" and ".Net" are quite important in terms of BAC vulnerabilities for web applications with the values of 1.8904, 0.1681, 11.3185, 0.7339.

Table 3
Confidence interval 95% of predictors and odds ratio

Predictors	Category	Odd ratio	95% C.I.
Reason of BAC			
Improper Input Validation	0.0000	1.8904	1.2081–2.958
Sensitive Data Disclosure	0.0000	0.1681	0.0827–0.341
Session Misconfiguration	0.0000	11.3185	5.4589–23.47
Directory Readable	0.5010	0.0000	n/a
Exploitation			
Access on Redirection	0.0000	0.7339	0.4710–1.1436
Misconfig. of Sensitive Data	0.0000	0.8285	0.5196–1.3212
Unauthorized Cookie Access	0.5041	2.6912	1.4138–5.1227
Platform			
PHP	0.8262	14.7769	1.8006–26.9577
Java	0.0104	0.1418	0.0493–0.4087
.Net	0.0399	0.1057	0.0549–0.2035
OS			
UNIX	0.0000	1.2507	0.8022–1.9505
Windows	0.0000	0.9307	0.5250–1.6502
Cent-OS	0.0000	0.8024	0.4901–1.3156

The risk of those who avoid them is 0.8285, 0.1418,0.1057, 1.2507, 0.9307, 0.8024 times higher, respectively. Similarly, factors, as well as ease of reading cookies, Through the use of Unauthorized Cookie Access, or websites built upon PHP, the risks increase by a factor of 3.6812, 2.6912, and 14.7769 respectively compared to non-associated ones.

4. Discussion

This study was conducted on more than 330 websites, including 129 BAC malicious web applications and 201 non-BAC malicious web applications, with the participation of web applications such as education, e-commerce, and government. Development using PHP, Java, and .NET platforms for business, healthcare, and private companies. Standard application hosting servers run on UNIX, Windows, and Cent-OS. This research reveals that "Causes of BAC", "BAC Development Technologies", "Platforms", and "Operating Systems" are applications used by BAC vulnerabilities. From the sample data, it can be seen that applications created with ".Net" have a bad BAC of 68.22 will most likely have BAC at a rate of "51.16" Session setting error", "login validation" and "sensitive data leak" problems will allow unauthorized users to gain permission through BAC vulnerabilities. In contrast, Access Redirect Settings and Sensitive Data Misconfiguration Retrieval are excellent strategies for exploiting BAC vulnerabilities. This analysis found that web applications with session errors were at higher risk of exploiting BAC vulnerabilities than applications without session errors (OR = 11.3185). Websites that are misunderstood are at greater risk than apps that use appropriate strategies (OR = 1.8904). The above five factors are significant in the chi-square test and binary logistic regression analysis. In this study, a factor (i.e., the development of BAC technology) that was significant in binary logistic regression but not significant in chi-square test (p¡ 0.007) was investigated. On the other hand, BAC reasons (e.g., "array readability") are not what BAC is responsible for. Limitations:

The analysis in this study was made from only 330 documents from BAC information and web applications containing BAC information. If the analysis involves large data, the results of the analysis will be different. Additionally, while this research provides insight into BAC vulnerabilities across a wide range of web applications and platforms, it is also important to be aware of the changing nature of the website. It is important to continually monitor and update safety procedures to reduce the risks associated with BAC and other adverse events. As web technology advances and new threats emerge, ongoing research and collaboration between developers, security experts, and researchers is important to prevent potential vulnerabilities and protect sensitive data in web applications. Future research may also explore other factors that influence the negative impact of BAC, such as innovation processes and the integration of new technologies such as cloud computing and microservices. Understanding how these factors interact with BAC vulnerabilities can provide insight into improving security and developing better web applications. Web applications that connect datasets to include larger and more diverse samples can provide insight into improving security and developing better web applications. This expansion will increase the generalizability of the findings and provide a better understanding of the vulnerability of BAC across different professions and regions. Additionally, it is important to review the specific coding practices and procedures used in each development environment to identify the issue causing the BAC to be poor. Collaboration with industry partners can also facilitate the sharing of best practices and the development of security systems. Another important area for future research will involve new technologies such as machine learning and artificial intelligence in detecting and mitigating BAC impairment. Additionally, longitudinal studies that monitor changes in BAC parameters over time may help understand the effectiveness of safety measures. Given the dynamic nature of web technology, regularly revalidating the security of your web application is critical to staying ahead of threats. The use of electronic devices that provide continuous monitoring and instant alerts for BAC vulnerabilities can reduce the window of impact and improve overall security.

5. Conclusion

BAC vulnerabilities in web applications are often caused by omissions in security design practices such as full input validation, sensitive robust measures to protect data, secure session configuration and management, and tight control over directory readability. These oversights are often unintentional but reflect designers' and developers' lack of awareness of the impact of BAC vulnerabilities in web applications and reduce risks associated with BAC impairment. This article identifies factors that contribute to negative BAC and evaluates their importance. These findings aim to raise web designers' and developers' awareness of important points to consider and enable them to implement preventive measures before applying. Other factors affect the foundation. Additionally, this study laid the foundation for future research to deepen and understand other factors contributing to negative BAC. These future studies may contribute to the development of a stronger and more robust web by expanding the knowledge base in web security. But awareness and effective precautions can go a long way in reducing these risks. By integrating security measures and being aware of emerging threats, web developers can secure their applications and protect sensitive user data. Additionally, it is important for non-web developers to continually monitor and constantly update security procedures to minimize BAC vulnerabilities and ensure that sensitive user information is continually protected against changing threats environment. By creating a culture of security awareness and risk management, organizations can build resilience to vulnerabilities and protect the integrity of their web applications [20].

It is important to review the practices and procedures used in each development environment to identify the issue causing the BAC to be negative. Partners can also facilitate the sharing of best practices and the development of security systems. Longitudinal studies that monitor changes in BAC parameters over time may help understand the effectiveness of safety measures used. It is important to stay ahead of threats.

Acknowledgement

I am grateful to all collaborators and National Cyber Security Centre UET Peshawar for their cooperation and contributions to this study. My final year advisor Prof. Dr. Sadeeq Jan with his guidance, wisdom, and unwavering support has greatly enhanced this study. His guidance and insights were crucial in developing the findings and conclusions presented in this research paper. I would also like to thank my colleagues and friends for their valuable suggestions and support throughout the research. Their constructive criticism and honest support helped overcome problems encountered during this study. Finally, I would like to thank the technical staff of our organization (National Cyber Security Center UET Peshawar, NCCS) for their assistance in obtaining the necessary resources and ensuring the success of the project.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data available on request from the corresponding author upon reasonable request.

Author Contribution Statement

Sadeeq Jan: Methodology, Formal analysis, Supervision. **Safi Ullah Khan:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Project administration. **Abdul Wahab:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Project administration. **Dr. Mohammad:** Investigation, Project administration.

References

[1] Goyal, D., Lavania, G., & Sharma, G. (2023). Review of modern web application cybersecurity risks and counter measures. In *AIP Conference Proceedings*, *2782*(1).

[2] Krishnaraj, N., Madaan, C., Awasthi, S., Subramani, R., Avinash, H., & Mukim, S. (2023). Common vulnerabilities in real world web applications. In T. A. Vakaliuk & S. O.

Semerikov (Eds.), *Doors* (pp. 9–22). Edge Computing Workshop (doors).

[3] Alahmad, M., Alkandari, A., & Alawadhi, N. (2022). Survey of broken authentication and session management of web application vulnerability attack. *Journal of Engineering Science and Technology*, *17*(2), 0874–0882.

[4] Hassan, M. M., Ali, M., Bhuiyan, T., Sharif, M., & Biswas, S. (2018). Quantitative assessment on broken access control vulnerability in web applications. In *International Conference on Cyber Security and Computer Science 2018*.

[5] Zhong, L. (2023). A survey of prevent and detect access control vulnerabilities. *arXiv Preprint:2304.10600*.

[6] Almushiti, E., Zaki, R., Thamer, N., & Alshaya, R. (2023). An investigation of broken access control types, vulnerabilities, protection, and security. In *International Conference on Innovation of Emerging Information and Communication Technology*, 253–269.

[7] Anas, A., Elgamal, S., & Youssef, B. (2024). Survey on detecting and preventing web application broken access control attacks. *International Journal of Electrical and Computer Engineering*, *14*(1), 772–781.

[8] Ahmed, M. I., Hassan, M. M., & Bhuyian, T. (2017). Local file disclosure vulnerability: A case study on the web applications of public sector. In *10th International Conference on Computer and Electrical Engineering*, 11–13.

[9] Hossain, M. M., Hasan, M., Mahajabin, M., Rahman, A., Maisha, N., & Nayan, P. N. (2023). Broken authentication and its significance in protecting online applications: An overview paper. In *International Conference on Cyber Intelligence and Information Retrieval*, 57–65.

[10] Helmiawan, M. A., Firmansyah, E., Fadil, I., Sofivan, Y., Mahardika, F., & Guntara, A. (2020). Analysis of web security using open web application security project 10. In *2020 8th International Conference on Cyber and IT Service Management*, 1–5.

[11] Čović, Z. (2022). Threats and vulnerabilities in web applications and how to avoid them. In *IFIP International Conference on Human Choice and Computers*, 93–103.

[12] Aljabri, M., Aldossary, M., Al-Homeed, N., Alhetelah, B., Althubiany, M., Alotaibi, O., & Alsaqer, S. (2022). Testing and exploiting tools to improve OWASP top ten security vulnerabilities detection. In *2022 14th International Conference on Computational Intelligence and Communication Networks*, 797–803.

[13] Flores Jr, C. P., & Richard, N. (2024). Evaluation of common security vulnerabilities of state universities and colleges websites based on OWASP. *Journal of Electrical Systems*, *20*(5s), 1396–1404.

[14] Nedeljković, N., Vugdelija, N., & Kojić, N. (2020). Use of "OWASP Top 10" in web application security. In *Fourth International Scientific Conference on Recent Advances in Information Technology, Tourism, Economics, Management and Agriculture*, 25.

[15] Choiriyah, A., & Qomariasih, N. (2023). Security analysis on websites belonging to the health service districts in Indonesia based on the open web application security project (OWASP) top 10 2021. In *2023 International Conference on Information Technology and Computing*, 267–272.

[16] Zhang, Z., Zou, F., Hong, J., Chen, L., & Yi, P. (2024). Detection and analysis of broken access control vulnerabilities in app-cloud interaction in IoT. *IEEE Internet of Things Journal*. 1–6.

[17] Khanum, A., Qadir, S., & Jehan, S. (2023). OWASP-based assessment of web application security. In *2023 18th International Conference on Emerging Technologies*, 240–245.

[18] Ilca, L. F., & Balan, T. (2021). Windows communication foundation penetration testing methodology. In *2021 16th International Conference on Engineering of Modern Electric Systems*, 1–4.

[19] Votipka, D., Fulton, K. R., Parker, J., Hou, M., Mazurek, M. L., & Hicks, M. (2020). Understanding security mistakes developers make: Qualitative analysis from build it, break it, fix it. In *29th USENIX Security Symposium*, 109–126.

[20] Huang, H., Shen, B., Zhong, L., & Zhou, Y. (2023). Protecting data integrity of web applications with database constraints inferred from application code. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 632–645.

How to Cite: Jan, S., Khan, S. U., Wahab, A., & Dr. Mohammad. (2025). Access Restricted: A Study of Broken Access Control Vulnerabilities. *Archives of Advanced Engineering Science*. https://doi.org/10.47852/bonviewAAES52024016