BROKEN ACCESS CONTROL: ANALYSIS, RISKS AND PREVENTION METHODS

Uzbekistan, Sharda University,
Faculty of Engineering and Technology,
1st-year Master's student
Obidjonov Nursulton G'ulomjon o'g'li

ANNOTATION: *This paper provides a comprehensive analysis of Broken Access Control, one of the most critical and widespread vulnerabilities in modern information systems. This vulnerability arises from improper enforcement of user permissions, allowing unauthorized access to restricted resources and posing serious threats to data security. The study examines the main causes, types, and real-world manifestations of Broken Access Control, as well as methods for its detection and prevention. In addition, modern approaches and technologies for enhancing system security are discussed.*

Key words: *information security, access control, authentication, authorization, vulnerability, web security, RBAC, IDOR*

АННОТАЦИЯ: *В данной статье проводится комплексный анализ уязвимости Broken Access Control, которая является одной из наиболее распространённых и опасных проблем в современных информационных системах. Данная уязвимость возникает вследствие некорректного управления правами доступа пользователей и может привести к несанкционированному доступу к защищённым ресурсам. В статье рассмотрены основные причины возникновения, виды и практические проявления данной уязвимости, а также методы её выявления и предотвращения. Кроме того, проанализированы современные подходы и технологии обеспечения информационной безопасности.*

КЛЮЧЕВЫЕ СЛОВА: *информационная безопасность, контроль доступа, аутентификация, авторизация, уязвимость, веб-безопасность, RBAC, IDOR.*

INTRODUCTION

In recent years, the rapid digital transformation of society has significantly increased reliance on web applications, cloud computing, and distributed information systems. From e-commerce platforms and banking systems to e-government services and educational portals, modern applications process vast amounts of sensitive data. As a result, ensuring the security and integrity of these systems has become a top priority for organizations worldwide.

One of the fundamental pillars of information security is access control, which governs how users interact with system resources. Access control mechanisms are designed to ensure that only authorized users can access specific data and perform permitted actions. These mechanisms are typically enforced through authentication (verifying identity) and authorization (granting permissions). However, when these controls are improperly implemented or inconsistently enforced, critical vulnerabilities can arise. Among such vulnerabilities, Broken Access Control stands out as one of the most severe and prevalent threats. According to the OWASP Top 10 Web Application Security Risks, Broken Access Control has consistently ranked as a leading security issue, highlighting its widespread impact across various industries. This vulnerability occurs when an application fails to correctly restrict user actions, allowing attackers to bypass authorization checks and gain unauthorized access to protected resources.

The consequences of Broken Access Control can be devastating. Attackers exploiting this vulnerability may view, modify, or delete sensitive

information, perform unauthorized transactions, or even gain administrative control over the system. In large-scale systems, such breaches can lead to financial losses, reputational damage, and legal consequences. For instance, unauthorized access to personal data may violate data protection regulations such as GDPR, further amplifying the impact. The root causes of Broken Access Control are often linked to flawed design decisions, inadequate testing, and over-reliance on client-side validation. Developers may unintentionally expose sensitive endpoints, fail to validate user roles on the server side, or improperly manage session tokens. Additionally, the increasing complexity of modern applications — including microservices architectures and API-driven systems — further complicates the implementation of consistent and secure access control policies.

Another contributing factor is the lack of standardized security practices during the software development lifecycle (SDLC). In many cases, security considerations are addressed only in later stages of development, rather than being integrated from the beginning. This approach increases the likelihood of vulnerabilities remaining undetected until deployment or even after exploitation.

Given the critical nature of this issue, it is essential to thoroughly understand the mechanisms, types, and causes of Broken Access Control vulnerabilities. This paper aims to provide a comprehensive analysis of this security flaw, examine its real-world implications, and propose effective mitigation strategies. By adopting secure coding practices and robust access control models, developers and organizations can significantly reduce the risk of unauthorized access and enhance the overall security posture of their systems.

RESEARCH METHODOLOGY

Theoretical Foundations of Access Control. Access control is a core component of information security that regulates how users and systems interact with protected resources. It ensures confidentiality, integrity, and availability (CIA triad) by restricting unauthorized access. Modern access control systems are typically based on several formal models: Discretionary Access Control (DAC): Access rights are assigned based on user identity and ownership of resources. While flexible, DAC is prone to misconfiguration and privilege leakage. Mandatory Access Control (MAC): Access decisions are enforced by a central authority based on predefined policies, often used in high-security environments.

Role-Based Access Control (RBAC): Permissions are assigned to roles rather than individual users, simplifying management in large systems. Attribute-Based Access Control (ABAC): Access is granted based on attributes such as user role, location, device, and time, providing fine-grained control. Despite the availability of these models, improper implementation often leads to inconsistencies, which can result in Broken Access Control vulnerabilities.

Broken Access Control occurs when an application fails to enforce authorization policies consistently across all endpoints. This vulnerability can manifest in various forms: URL Manipulation: Attackers modify query parameters to access unauthorized data. Hidden Endpoint Exposure: Sensitive endpoints are not properly protected and can be accessed directly. Method Tampering: Switching HTTP methods (e.g., GET to POST) to bypass restrictions. JWT Misconfiguration: Improper validation of JSON Web Tokens allows privilege escalation. CORS Misconfiguration: Weak Cross-Origin Resource Sharing policies expose sensitive APIs. These issues often arise due to fragmented security checks and lack of centralized authorization mechanisms.

This occurs when a user gains higher privileges than intended. For example, a regular user accessing administrative dashboards due to missing role validation. This is particularly dangerous as it may grant full system control. In this case, users access data belonging to other users with similar roles. For example, accessing another user's profile by modifying an identifier. This type is common in poorly secured REST APIs.

Some systems fail to enforce access rules based on contextual conditions such as time, location, or workflow stage. For example, allowing a user to modify a completed transaction. One of the most common issues, IDOR allows attackers to directly access internal objects (e.g., files, database records) without proper authorization checks.

The emergence of Broken Access Control is rarely due to a single factor. Instead, it is the result of multiple weaknesses: Lack of centralized access control logic: Security rules scattered across codebase Client-side enforcement: Relying on front-end validation instead of server-side checks Inadequate testing: Insufficient penetration testing and code reviews Complex system architecture: Microservices and APIs increase attack surface Human factors: Developer inexperience or lack of security awareness Furthermore, rapid development cycles and Agile methodologies sometimes prioritize functionality over security, increasing the likelihood of such vulnerabilities. Identifying Broken Access Control requires a combination of automated tools and manual testing:

Static Application Security Testing (SAST): Analyzes source code for vulnerabilities

Dynamic Application Security Testing (DAST): Tests running applications for flaws

Penetration Testing: Simulates real-world attacks Manual Code Review: Identifies logical flaws missed by tools Security testing should be integrated into the Software Development Life Cycle (SDLC) to ensure early detection.

Preventing Broken Access Control requires a systematic and layered approach:

Centralized Authorization Mechanism: Implement a unified access control layer Deny-by-default Principle: Block all access unless explicitly allowed Least Privilege Principle: Users should have only necessary permissions Secure API Design: Validate every request at the server level Token Security: Proper validation and expiration of authentication tokens Logging and Monitoring: Detect suspicious access patterns in real time Additionally, organizations should follow security standards recommended by OWASP and regularly update their systems.

In real-world scenarios, Broken Access Control has been responsible for numerous data breaches. For example, in some web applications, attackers exploited IDOR vulnerabilities to access millions of user records by simply iterating through user IDs. In other cases, misconfigured APIs allowed unauthorized access to financial transactions. These incidents highlight the importance of rigorous security practices and continuous monitoring.

To further strengthen access control, modern systems are increasingly adopting:

- Zero Trust Architecture (ZTA)
- Multi-Factor Authentication (MFA)
- AI-based anomaly detection systems

These advanced approaches help reduce the risk of unauthorized access even in complex environments.

CONCLUSION AND RECOMMENDATIONS

In conclusion, Broken Access Control remains one of the most critical and widespread vulnerabilities in modern web applications and information systems. As highlighted by the OWASP Top 10 security risks, this issue continues to dominate due to its severe impact and frequent occurrence across different platforms and industries. The failure to properly enforce access restrictions allows attackers to bypass authorization mechanisms, leading to unauthorized data access, privilege escalation, and, in many cases, full system compromise.

This study has demonstrated that the root causes of Broken Access Control are often deeply embedded in both technical and organizational aspects of software development. Weak architectural design, lack of centralized authorization mechanisms, insufficient server-side validation, and inadequate security testing all contribute to the emergence of this vulnerability. Moreover, the growing complexity of modern systems — including cloud-based infrastructures, microservices architectures, and API-driven applications — significantly increases the attack surface, making consistent access control enforcement more challenging. The analysis also revealed that Broken Access Control is not limited to a single form but manifests in multiple variations, such as vertical and horizontal privilege escalation, Insecure Direct Object References (IDOR), and context-dependent access failures. Each of these types poses unique risks and requires specific mitigation strategies. Therefore, a one-size-fits-all approach is insufficient; instead, a comprehensive and layered security strategy must be adopted.

To effectively mitigate these risks, organizations must implement robust access control frameworks based on well-established principles such as **deny-by-default** and **least privilege**. The adoption of Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC) models can significantly

enhance the granularity and flexibility of authorization mechanisms. Additionally, enforcing strict server-side validation, securing APIs, and implementing proper session and token management are essential steps in preventing unauthorized access.

Furthermore, integrating security practices into every phase of the Software Development Life Cycle (SDLC) is crucial. This includes conducting regular security audits, penetration testing, and code reviews, as well as adopting automated security tools such as SAST and DAST. Continuous monitoring and logging mechanisms should also be employed to detect and respond to suspicious activities in real time. From a strategic perspective, emerging approaches such as Zero Trust Architecture (ZTA), Multi-Factor Authentication (MFA), and AI-driven threat detection systems offer promising solutions for strengthening access control in complex environments. These technologies ensure that trust is never assumed and that every access request is thoroughly verified, regardless of its origin. Ultimately, ensuring effective access control is not only a technical requirement but also a critical component of organizational risk management. Developers, system architects, and security professionals must work collaboratively to design secure systems that can withstand evolving cyber threats. By prioritizing secure coding practices, adhering to international security standards, and fostering a culture of cybersecurity awareness, organizations can significantly reduce the risk of Broken Access Control and protect their digital assets.

REFERENCES

1. OWASP. *OWASP Top 10: The Ten Most Critical Web Application Security Risks*. OWASP Foundation, 2021.

2. National Institute of Standards and Technology. *Security and Privacy Controls for Information Systems and Organizations (SP 800-53)*. NIST, 2020.

3. William Stallings. *Cryptography and Network Security: Principles and Practice*. 7th Edition. Pearson, 2017.

4. Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. 2nd Edition. Wiley, 2008.

5. ISO. *ISO/IEC 27001: Information Security Management Systems*. ISO, 2013.

6. Andrew Hoffman. *Web Application Security: Exploitation and Countermeasures for Modern Web Applications*. O'Reilly Media, 2020.

7. Cloud Security Alliance. *Security Guidance for Critical Areas of Focus in Cloud Computing*. CSA, 2019.

8. IETF. *RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. 2014.

OWASP. *Access Control Cheat Sheet*. OWASP Foundation, 2023.