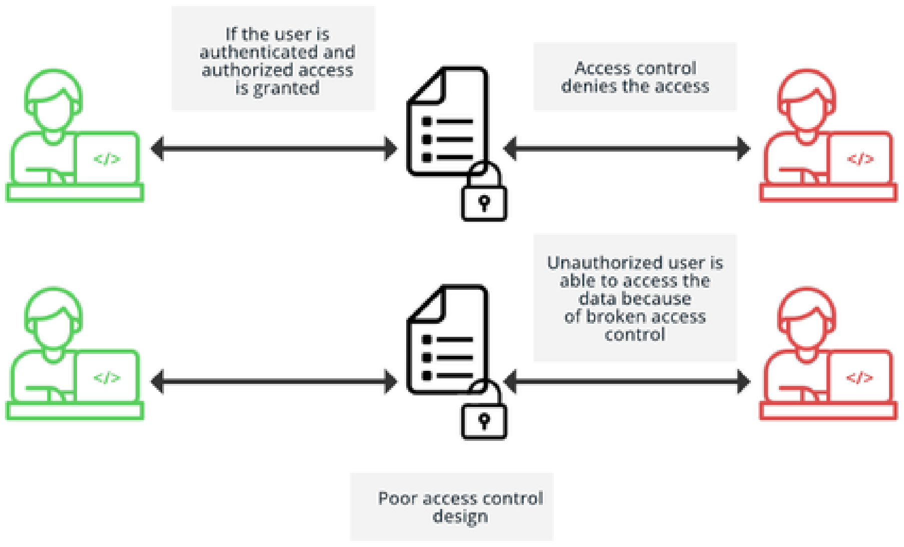
# Access control vulnerabilities

Sabuj191761 | sabuj-modak | sabujmodak

## What is access control?

In web applications, **access control** is defined as the process of managing users to access and restrict specific resources or functionalities within the application.


If the user is authenticated and authorized access is granted
Access control denies the access
</>
</>
Unauthorized user is able to access the data because of broken access control
</>
</>
Poor access control design

## Core components under Access Control:

- Authentication
- Authorization
- Session Management
- User Roles & Permissions

## Broken access control:

Broken access control refers to a vulnerability in a system's access control mechanisms that allows an attacker to bypass restrictions and gain unauthorized access to resources or perform actions they shouldn't be able to. It's a top security concern consistently ranked among the OWASP Top 10 web application security risks.

Broken Access Control Attacks
Brute Force Attacks
Man-in-the-Middle Attacks
Session Hijacking
Privilege Escalation Attacks

**Key characteristics:**

- **Failure to enforce access control policies effectively.**
- **Weak authentication or authorization mechanisms.**
- **Flaws in design or implementation of access controls.**
- **Incorrect configuration or management of access control settings.**

## Examples of Broken Access Control

Here are some real-world examples of broken-access control:

- Unrestricted URL access: When a web app allows access to restricted pages via unrestricted URLs, it's broken-access control. This happens when a developer fails to implement proper authentication and authorization controls. It allows anyone with the URL to access restricted pages. Attackers can manipulate URL parameters to access sensitive information.
- Inadequate authorization checks: Another type of broken-access control occurs when a web application fails to perform proper authorization checks, allowing attackers to access restricted pages or functions. This can happen if an application only checks for authentication rather than authorization. For

example, an authenticated user may be able to access pages or functions they should not be able to access. An attacker can take advantage of this flaw to gain unauthorized access to sensitive data or functionality.

- Insecure direct object reference (IDOR): An IDOR vulnerability enables attackers to access sensitive data by manipulating object references. Attackers modify the URL's ID parameter to gain access. This vulnerability arises due to inadequate user input validation and access control measures in web applications.
- Horizontal and vertical access control: These vulnerabilities occur when users have unauthorized access to data or functions based on their role or permission level. This happens when access controls are misconfigured, allowing users to access data or functions beyond their assigned level.
- Broken session management: Broken session management vulnerabilities occur when a web application fails to properly manage user sessions. As a result, attackers may be able to steal or reuse session tokens, granting them unauthorized access to the web application. For example, if a web application fails to properly invalidate session tokens when a user logs out, an attacker can reenter the application using the session token.

## Missing Access Control

Vulnerabilities associated with Missing Access Control (MAC) arise when a system lacks a robust access control mechanism. Developers might forget to add important checks like who is allowed to perform certain actions or if the data coming from the user to the application is safe.

These vulnerabilities let attackers get into the application, manipulate the data, or cause other problems. To prevent this, developers need to be careful and make sure they include strong access control checks, such as validating user identity, giving the proper and required permissions to employees or fellow developers, and validating the information for any issues before using it. This helps to keep the system safe from unauthenticated access and potential harm.

## Examples of Access Control Vulnerabilities

1. Insecure Function Level Authorization: This vulnerability occurs when an application lacks proper checks and authorization at different function levels. It can allow unauthorized users to access sensitive functionalities or APIs directly.

2. Insecure Direct Object Manipulation (IDOR): This vulnerability allows an attacker to modify object identifiers or parameters associated with resources to gain unauthorized access to sensitive data. For example, manipulating URLs or form inputs to access or modify data to other users.
3. Vertical Access Control Bypass (Vertical Privilege Escalation): This vulnerability arises when a lower-privileged user gains unauthorized access to resources or actions of higher-privileged users. For example, a user with no additional permissions is able to access and modify admin settings or sensitive data like resetting the passwords of other users.
4. Horizontal Access Control Bypass (Horizontal Privilege Escalation): In this case, an attacker can access resources or perform actions of another user with the same level of privilege. For instance, one user viewing or editing the personal information of another user (with same permissions) without proper authorization comes under the same role group.
5. Failure to Enforce Authorization/Validation Checks: When an application fails to enforce proper authorization checks at different layers (such as on API gateway, mid-tier applications and API level), it allows users to access restricted resources or perform actions they shouldn't have permissions for.

## Insecure Direct Object References

Insecure Direct Object References (IDOR) occur when an application provides direct access to objects based on user-supplied input. As a result of this vulnerability attackers can bypass authorization and access resources in the system directly, for example database records or files. Insecure Direct Object References allow attackers to bypass authorization and access resources directly by modifying the value of a parameter used to directly point to an object. Such resources can be database entries belonging to other users, files in the system, and more. This is caused by the fact that the application takes user supplied input and uses it to retrieve an object without performing sufficient authorization checks.

### Access controls can be classified into different types, as stated below.

1. Vertical Access Controls
2. Horizontal Access Controls
3. Context-Dependent Access Controls

## Vertical Access Controls

They are responsible for restricting access to sensitive functionalities which are not available to other types of users. These types of controls are generally used for limiting access to admin functionalities from the non-administrative set of users.

## Horizontal Access Controls

They are used when there is a requirement to restrict resource access to only specific users allowed to access those particular resources. For example, in any banking application, it should only allow the user to view his transactions but not for any other users.

## Context-Dependent Access Controls

They are responsible for limiting access to resources and functionalities based on the application's context and user activity.

When an attacker breaks these access controls, then this will lead to vulnerabilities related to authorization issues. There are different ways to break the access control implied upon the resources, and these ways are specific to the nature and context of the application and the functionalities being accessed.

Different types of broken access controls are in detail as we go further with this blog.

1. Vertical Privilege Escalation
2. Horizontal Privilege Escalation
3. IDOR (Insecure Direct Object Reference)
4. Access Control Vulnerabilities in a Multi-Step Process
5. Referrer-Based Access Control
6. Location-Based Access Control

## IDOR (Insecure Direct Object Reference)

IDOR is another sub-group of access control vulnerability; this vulnerability emerges when user-supplied input is used by the application to directly access the objects. An attacker might take advantage of the same and will modify the inputs to get direct unauthorized access. Although IDOR is an implementation error from a pool of other implementation errors, it is more popular than the rest because of its listing in OWASP Top Ten in the year 2007.

Direct reference in IDOR can be to anything like; it can be directly referenced to database objects or even to the static file, which is directly stored on the server-side filesystem.

## Access Control Vulnerabilities In a Multi-Step Process:

There are various functionalities in the application where the process is submitted in multiple steps. For example, there are many applications that use incremental forms to commit changes to users' data or even to access existing data. In the case of these incremental forms, sometimes the access controls are implemented in the initial step only. So, an attacker might try and alter the content in the final submission, which will ultimately save the final data altered that the attacker has provided.

## Referrer-Based Access Control:

Some of the applications use access controls that are based on the Referrer header of the HTTP request. The referrer header is basically used to indicate where the page is coming from, i.e., the Referrer header is always the previous URL of the page. In a few of the applications, the restriction to some critical or sensitive functionalities is based on the referrer header. For example, if a user wants to access a URL "/admin/dashboard," then the application will first check the referrer header of the request and will validate whether the value of the referrer is "/admin" or not. This way, the access is granted after validating the referrer of the request. In such a type of access control, the attacker will alter the referrer header and gain unauthorized access to such sensitive pages and functionalities.

## Location-Based Access Control:

These days websites are also enforcing access controls based on the geolocation of the user. In cases, whether banking applications or entertainment applications, depending upon the state legislation, only the approved content and functionalities will be available to the user. These controls can be bypassed using web proxies, VPNs, or manipulation of the client-side geolocation mechanisms.

## Mitigation for Access Control Vulnerabilities:

1. Until and unless public access is required for any resource, by default, deny access to resources.
2. Mandatory defines the access for each resource at the code level.
3. Test all the access controls properly to check whether they are working properly or if there are any loopholes in the implementation.
4. Use single application-wise access control wherever possible.
5. Don’t rely only on obfuscation; implement a proper access control mechanism.

## Reference Links:

PortSwigger

InfoSec Write-ups

https://www.varutra.com/access-control-vulnerabilities/