Access Control

Access control is the application of constraints on who or what is authorized to perform actions or access resources. In the context of web applications, access control is dependent on authentication and session management

- **Authentication** confirms that the user is who they say they are
- **Session management** identifies which subsequent HTTP requests are being made by that same user
- **Access control** determines whether the user is allowed to carry out the action that they are attempting to perform

Vertical access controls

With vertical access controls, different types of users have access to different application functions. For example, an administrator might be able to modify or delete any user's account, while an ordinary user has no access to these actions. Vertical access controls can be more fine-grained implementations of security models designed to enforce business policies such as separation of duties and least privilege.

Horizontal access controls

With horizontal access controls, different users have access to a subset of resources of the same type. For example, a banking application will allow a user to view transactions and make payments from their own accounts, but not the accounts of any other use

Context-dependent access controls

Context-dependent access controls prevent a user performing actions in the wrong order. For example, a retail website might prevent users from modifying the contents of their shopping cart after they have made payment

Broken access controls :-

Vertical privilege escalation

If a user can gain access to functionality that they are not permitted to access then this is vertical privilege escalation. For example, if a non-administrative user can gain access to an admin page where they can delete user accounts, then this is vertical privilege escalation.

Unprotected functionality

At its most basic, vertical privilege escalation arises where an application does not enforce any protection for sensitive functionality

a website might host sensitive functionality at the following URL

```vbscript-html
https://insecure-website.com/admin
https://insecure-website.com/robots.txt
```

In some cases, sensitive functionality is concealed by giving it a less predictable URL. This is an example of so-called "security by obscurity". However, hiding sensitive functionality does not provide effective access control because users might discover the obfuscated URL in a number of ways.

```javascript
<script>
    var isAdmin = false;
    if (isAdmin) {
        ...
        var adminPanelTag = document.createElement('a');
        adminPanelTag.setAttribute('https://insecure-website.com/administrator-panel-yb556');
        adminPanelTag.innerText = 'Admin panel';
        ...
    }
</script>
```

This script adds a link to the user's UI if they are an admin user. However, the script containing the URL is visible to all users regardless of their role.

Parameter-based access control methods

Some applications determine the user's access rights or role at login, and then store this information in a user-controllable location

- A hidden field.
- A cookie.
- A preset query string parameter.

The application makes access control decisions based on the submitted value. For example:

```vbscript-html
https://insecure-website.com/login/home.jsp?admin=true
https://insecure-website.com/login/home.jsp?role=1
```

Broken access control resulting from platform misconfiguration

Some applications enforce access controls at the platform layer. they do this by restricting access to specific URLs and HTTP methods based on the user's role

```vbscript-html
DENY: POST, /admin/deleteUser, managers
```

This rule denies access to the `POST` method on the URL `/admin/deleteUser`, for users in the managers group.

Some application frameworks support various non-standard HTTP headers that can be used to override the URL in the original request, such as `X-Original-URL` and `X-Rewrite-URL`

```vbscript-html
POST / HTTP/1.1
X-Original-URL: /admin/deleteUser
...
```

If an attacker can use the `GET` (or another) method to perform actions on a restricted URL, they can bypass the access control that is implemented at the platform layer.

Horizontal privilege escalation

Horizontal privilege escalation occurs if a user is able to gain access to resources belonging to another user, instead of their own resources of that type. For example, if an employee can access the records of other employees as well as their own, then this is horizontal privilege escalation.

```vbscript-html
https://insecure-website.com/myaccount?id=123
```

If an attacker modifies the `id` parameter value to that of another user, they might gain access to another user's account page, and the associated data and functions.

instead of an incrementing number, an application might use globally unique identifiers (GUIDs) to identify users. This may prevent an attacker from guessing or predicting another user's identifier. However, the GUIDs belonging to other users might be disclosed elsewhere in the application where users are referenced, such as user messages or reviews.

Access control vulnerabilities in multi-step processes

the administrative function to update user details might involve the following steps

1. Load the form that contains details for a specific user.
2. Submit the changes.

3. Review the changes and confirm

Imagine a website where access controls are correctly applied to the first and second steps, but not to the third step. The website assumes that a user will only reach step 3 if they have already completed the first steps, which are properly controlled. An attacker can gain unauthorized access to the function by skipping the first two steps and directly submitting the request for the third step with the required parameters.

IDORs

Insecure Direct Object References are a SubCategory of acces control vulnerabilities. IDORs Occurs if an application uses user-supplied input to access objects directly and an attacker can modify the input to obtain unauthorized access.

Referer-based access control

Some websites base access controls on the `Referer` header submitted in the HTTP request. The `Referer` header can be added to requests by browsers to indicate which page initiated a request.

For example, an application robustly enforces access control over the main administrative page at `/admin`, but for sub-pages such as `/admin/deleteUser` only inspects the `Referer` header. If the `Referer` header contains the main `/admin` URL, then the request is allowed.

Location-based access control

Some websites enforce access controls based on the user's geographical location. This can apply, for example, to banking applications or media services where state legislation or business restrictions apply. These access controls can often be circumvented by the use of web proxies, VPNs, or manipulation of client-side geolocation mechanisms.

How to prevent access control vulnerabilities

- Never rely on obfuscation alone for access control.
- Unless a resource is intended to be publicly accessible, deny access by default.
- Wherever possible, use a single application-wide mechanism for enforcing access controls.
- At the code level, make it mandatory for developers to declare the access that is allowed for each resource, and deny access by default.
- Thoroughly audit and test access controls to ensure they work as designed.