Introduction to the OWASP Top 10 – 2021

# Risk A1: Broken Access Control

## Key Concepts

### Key Concept 1: Access Control

- Selectively restricting access
- Features or data require permission to access

### Key Concept 2: Authorization

- Authorization is the actual permission to access certain features or data
- Permission to access certain features or data  is call authorization

Additional Information

Access Control is the process by which you disallow access, typically to data. A simple example, the key to your house, provides you access. The access control is in the door, the lock, and the windows of the house being shut. With access control features or data require a permission to be given to access them.

Authorization is the actual permission to access. In our previous example of the house, whoever has the key is authorized to access what is in the house. Permission to access certain features or data is called authorization.

Access Control and Authorization are the two key information security concepts you need for A1, the first risk of the OWASP Top 10.

## Definition

Users of the application can operate outside of their defined permissions.

- Access to unauthorized information or specific features of the application is made available
- Violation of the principle of least privilege or least authority

Additional Information

The definition of A1 of 2021 – Broken Access Control – is that users of the application can operate outside their defined permissions. An example, you may try one of the locked rooms at a party and it opens anyways because the lock is broken. That is broken access control.

Based on our party example, you see how a standard consequence of Broken Access Control is access to unauthorized information or specific features of the application being made available when they should not. As a result, Broken Access Control is a violation of the principle of least privilege. Based on this principle, you should only give the permissions necessary for the user to perform their tasks and nothing more.

## Example

### Bad Example

```
adminReport(param) {
 if (user.isRole("USER"), param)
 {
 //execute admin activity on param
 }
}
```

### Good Example

```
adminReport(param) {
 if (user.isAuthorized("ADMIN"), param)
 {
 //execute admin activity on param
 }
}
```

#### Additional Information

Above is your first example in pseudocode. This is a plain language description and you do not need to be a developer to understand this, so please do not shy away from examples in pseudocode.

As a bad example, of what not to do, the user is checked that they have the role of "USER". Based on this check, they are allowed to execute some activity that only administrators should be allowed to perform. That is Broken Access Control.

For our good example, the user is checked as to whether they are an administrator. Only if they are authorized as an administrator, can they generate the admin report. Ergo, the user of the application can only operate within their defined permission of user and not administrator. With our good example we make sure only administrators can generate the admin report.

## Challenges

### Access Control is difficult for developers to build

- Frameworks rarely provide detailed access control functionality

### Access Control is difficult to test

- Automated scanning tools are rarely aware of your custom access control policies

Additional Information

We now understand how broken access control happens let us look at why it happens.

Access control is difficult for developers to build because frameworks do not come with detailed access control patterns and functionality. Access control is typically tailored for different applications. The type of access control an application has depends on the users, the features, the data and more importantly, the specific business model.

Access control is also difficult to test. Automated scanning tools are rarely aware of the state or states of your application. Scanning tools must be trained for that. As an example of why testing for broken access control is hard, consider the quite common penetration testing activity of login in as user and then attempting to access functionality only provided to administrators. This would be typically hard to automate.

## Best Protection Strategies

DEBAR

**Design** access control so all requests must be authorized

**Enforce** access by activity and only for valid workflow paths, never by role

**Build** a centralized access control mechanism

**Assign** permissions to users in the context of data

**Refuse** access by default, fail securely

Additional Information

These are some of the best protection strategies against A1 of the Top 10 – Broken Access Control.