A01:2021 – Broken Access Control

- **Scenario**
  - The application uses unverified data in a SQL call that is accessing account information
    - https://example.com/app/accountInfo?acct=notmyacct
  - An attacker simply forces browses to target URLs. Admin rights are required for access to the admin page
    - From : https://example.com/app/getappInfo
      - To : https://example.com/app/admin_getappInfo
- **access controls can be divided into**
  - Vertical access controls
  - Horizontal access controls
  - Context-dependent access controls
    - Context-dependent access controls restrict access to functionality and resources based upon the state of the application or the user's interaction with it.
- **Description**
  - Access control enforces policy such that users cannot act outside of their intended permissions.
  - Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.
  - Access control design decisions have to be made by humans, not technology, and the potential for errors is high.
  - Common access control vulnerabilities include:
    - Violation of the principle of least privilege or deny by default
      - where access should only be granted for particular capabilities, roles, or users, but is available to anyone.
    - Bypassing access control checks by modifying the URL, internal application state, or the HTML page, or by using an attack tool modifying API requests.
    - Permitting viewing or editing someone else's account, by providing its unique identifier IDOR
    - Accessing API with missing access controls for POST, PUT and DELETE
    - Elevation of privilege. Acting as a user without being logged in or acting as an admin when logged in as a user
    - Metadata manipulation
      - EX :
        - JSON Web Token (JWT)
        - cookie or hidden field manipulated
        - abusing JWT invalidation.
    - CORS misconfiguration allows API access from unauthorized/untrusted origins.
    - Force browsing to authenticated pages as an unauthenticated user or to privileged pages as a standard user.
- **Access control is only effective in trusted server-side code or server-less API, where the attacker cannot modify the access control check or metadata.**
- **access control is dependent on**
  - **Authentication** identifies the user and confirms that they are 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.
- **examples of Vulnerabilities**
  - Transaction Authorization
  - Insecure Direct Object Reference
  - Authorization
  - Cross-Site Request Forgery
- **EX :**
  - Except for public resources, deny by default
  - Implement access control mechanisms once
  - minimizing Cross-Origin Resource Sharing (CORS) usage
  - Model access controls should enforce record ownership
  - Unique application business limit requirements should be enforced by domain models.
  - Disable web server directory listing and ensure file metadata (e.g., .git) and backup files are not present within web roots.
  - Log access control failures, alert admins when appropriate (e.g., repeated failures).
  - Rate limit API and controller access
  - Stateful session identifiers