A5: Broken Access Control

- Improper enforcement of what authenticated users are allowed to do.
- As with other vulnerabilities, attacker can gain access to data or functions that he shouldn't.

(insecure direct object reference)

Exploitation of access control is a core skill of attackers. SAST and DAST tools can detect the absence of access control but cannot verify if it is functional when it is present. Access control is detectable using manual means, or possibly through automation for the absence of access controls in certain frameworks.

Access control weaknesses are common due to the lack of automated detection, and lack of effective functional testing by application developers.

Access control detection is not typically amenable to automated static or dynamic testing. Manual testing is the best way to detect missing or ineffective access control, including HTTP method (GET vs PUT, etc), controller, direct object references, etc.

The technical impact is attackers acting as users or administrators, or users using privileged functions, or creating, accessing, updating or deleting every record.
The business impact depends on the protection needs of the application and data.

Is the Application Vulnerable?

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 of the limits of the user.
Common access control vulnerabilities include:

- Bypassing access control checks by **modifying the URL**, **internal application state**, or the **HTML page**, or simply using a custom API attack tool.

- Allowing the primary key to be changed to another users record, permitting viewing or editing someone else's account.

- Elevation of privilege. Acting as a user without being logged in, or acting as an admin when logged in as a user.

- Metadata manipulation, such as replaying or tampering with a JSON Web Token (JWT) access control token or a cookie or hidden field manipulated to elevate privileges, or abusing JWT invalidation

- CORS misconfiguration allows unauthorized API access.

- Force browsing to authenticated pages as an unauthenticated user or to privileged pages as a standard user. Accessing API with missing access controls for POST, PUT and DELETE.

Example Attack Scenarios

Scenario #1: The application uses unverified data in a SQL call that is accessing account information:

pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );

An attacker simply modifies the 'acct' parameter in the browser to send whatever account number they want. If not properly verified, the attacker can access any user's account.

http://example.com/app/accountInfo?acct=notmyacct

Scenario #2: An attacker simply force browses to target URLs. Admin rights are required for access to the admin page.

http://example.com/app/getappInfo http://example.com/app/admin_getappInfo

If an unauthenticated user can access either page, it’s a flaw. If a non-admin can access the admin page, this is a flaw

How to Prevent

Access control is only effective if enforced in trusted server-side code or server-less API, where the attacker cannot modify the access control check or metadata.

- With the exception of public resources, deny by default.
- Implement access control mechanisms once and re-use them throughout the application, including minimizing CORS usage.
- Model access controls should enforce record ownership, rather than accepting that the user can create, read, update, or delete any record.
- 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 to minimize the harm from automated attack tooling.
- JWT tokens should be invalidated on the server after logout. Developers and QA staff should include functional access control unit and integration tests.