API5:2023 Broken Function Level Authorization

Threat agents/Attack vectors	Security Weakness	Impacts
API Specific : Exploitability **Easy**	Prevalence **Common** : Detectability **Easy**	Technical **Severe** : Business Specific
Exploitation requires the attacker to send legitimate API calls to an API endpoint that they should not have access to as anonymous users or regular, non-privileged users. Exposed endpoints will be easily exploited.	Authorization checks for a function or resource are usually managed via configuration or code level. Implementing proper checks can be a confusing task since modern applications can contain many types of roles, groups, and complex user hierarchies (e.g. sub-users, or users with more than one role). It's easier to discover these flaws in APIs since APIs are more structured, and accessing different functions is more predictable.	Such flaws allow attackers to access unauthorized functionality. Administrative functions are key targets for this type of attack and may lead to data disclosure, data loss, or data corruption. Ultimately, it may lead to service disruption.

Is the API Vulnerable?

The best way to find broken function level authorization issues is to perform a deep analysis of the authorization mechanism while keeping in mind the user hierarchy, different roles or groups in the application, and asking the following questions:

- Can a regular user access administrative endpoints?
- Can a user perform sensitive actions (e.g. creation, modification, or deletion) that they should not have access to by simply changing the HTTP method (e.g. from `GET` to `DELETE`)?
- Can a user from group X access a function that should be exposed only to users from group Y, by simply guessing the endpoint URL and parameters (e.g. `/api/v1/users/export_all`)?

Don't assume that an API endpoint is regular or administrative only based on the URL path.

While developers might choose to expose most of the administrative endpoints under a specific relative path, like `/api/admins`, it's very common to find these administrative endpoints under other relative paths together with regular endpoints, like `/api/users`.

Example Attack Scenarios

Scenario #1

During the registration process for an application that allows only invited users to join, the mobile application triggers an API call to `GET /api/invites/{invite_guid}`. The response contains a JSON with details about the invite, including the user's role and the user's email.

An attacker duplicates the request and manipulates the HTTP method and endpoint to `POST /api/invites/new`. This endpoint should only be accessed by administrators using the admin console. The endpoint does not implement function level authorization checks.

The attacker exploits the issue and sends a new invite with admin privileges:

```
POST /api/invites/new

{
  "email": "attacker@somehost.com",
  "role":"admin"
}
```

Later on, the attacker uses the maliciously crafted invite in order to create themselves an admin account and gain full access to the system.

Scenario #2

An API contains an endpoint that should be exposed only to administrators - `GET /api/admin/v1/users/all`. This endpoint returns the details of all the users of the application and does not implement function level authorization checks. An

attacker who learned the API structure takes an educated guess and manages to access this endpoint, which exposes sensitive details of the users of the application.

How To Prevent

Your application should have a consistent and easy-to-analyze authorization module that is invoked from all your business functions. Frequently, such protection is provided by one or more components external to the application code.

- The enforcement mechanism(s) should deny all access by default, requiring explicit grants to specific roles for access to every function.
- Review your API endpoints against function level authorization flaws, while keeping in mind the business logic of the application and groups hierarchy.
- Make sure that all of your administrative controllers inherit from an administrative abstract controller that implements authorization checks based on the user's group/role.
- Make sure that administrative functions inside a regular controller implement authorization checks based on the user's group and role.