Authorization Flaws

Broken Access Control

Access control is supposed to prevent that users can act outside of their intended permissions.

Possible Impact of Broken Access Control

- Access unauthorized functionality and/or data, such as access other users' accounts
- View sensitive files
- Modify other users' data
- Change access rights

Common Attacks

- Modifying URL, internal application state, or HTML page
- Changing the primary key to another users record
- Elevation of privilege
  - Acting as a user without being logged in
  - Acting as an admin when logged in as a user

ℹ️ *Obtaining a higher level of access is also referred to as **Vertical** Privilege Escalation while same-level access to another user's data is called **Horizontal** Privilege Escalation.*

- Metadata manipulation
  - Replaying or tampering with access control tokens
  - Cookie or hidden field manipulation
- Force browsing to authenticated pages as an anonymous user or to privileged pages as a standard user
- Accessing API with missing access controls for `POST`, `PUT` and `DELETE`

Risk Rating

Broken Access Control

Exploitability	Prevalence	Detecability	Impact	Risk
🔸 Average	🔸 Common	🔸 Average	🔴 Severe	A5
(2	+ 2	+ 2) / 3	* 3	= 6.0

Exercise 5.1

Assuming no access control is in place, which privilege escalations are possible by tampering with the following URLs?

1. `http://logistics-worldwi.de/showShipment?id=40643108`
2. `http://my-universi.ty/api/students/6503/exams/view`
3. `http://document-warehou.se/landingpage?content=index.html`

Exercise 5.2

1. Access the administration section of the store (★★)
2. View another user's shopping basket (★★)
3. Get rid of all 5-star customer feedback (★★)
4. Post some feedback for another user but without previously logging in as that user (★★★)

Prevention

- **Access control** is only effective if **enforced in trusted server-side code**
- With the exception of public resources, **deny by default**
- **Implement** access control mechanisms **once and re-use** them throughout the application
- **Enforce record ownership**
- **Disable web server directory listing** and ensure file metadata and backup files are not present within web roots

- **Log access control failures**, alert admins when appropriate
- Rate limit API and controller access to minimize the harm from automated attack tooling
- Access tokens should be invalidated on the server after logout
- Developers and QA staff should include functional access control unit and integration tests

Access Control Design Principles

1. Design Access Control thoroughly up front
2. Force all Requests to go through Access Control checks
3. Deny by Default
4. Principle of Least Privilege
5. Don't hardcode roles
6. Log all Access Control events

Exercise 5.3 (🏠)

1. Place an order with a negative total (★★★)
2. Access one or more misplaced files (★★★★ - ★★★★★★)