OWASP prioritere efter hvor farlige security trusler er. De ændre sig hele tiden.

Exploitability, Prevalence, Detectability \* Technical (Busness impact)

Usability vs. security

**A2-Broken Authentication**

Authentication = Are you who you say you are?

Link: <https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A2-Broken_Authentication>

The application is vulnerable if?

* If it permits Credential Stuffing (Trying lists of known leaked username/password combinations).
* If it permits brute force attacks or other automated attacks (When someone multiple times try to access other accounts).
* If it permits weak passwords.
* If it uses week credential recovery for forgotten passwords (F.eks “knowledge based awnsers”).
* If it uses plain text, encrypted, or weak hashed passwords.
* If it exposes session id’s in the url.
* Does not rotate Session IDs after successful login.
* Does not properly invalidate Session IDs.

Use multifactor authentication (Login with multiple factors, not just password, something you know, something you know are, something you have)

You can delay logging attempts after each try or reject the client after multiple attempts.

Don’t allow weak passwords, require special characters.

On a login page, do not tell the user that the password or username is wrong (Don’t give the hacker a lead).

CAPTCHA er et antihacker- og digitaliseringssystem, som fungerer ved brug af et billede/ billeder af ord med forvreden tekst.

WEB-sessions with orderly timeouts and uniqe session id’s (http er stateless, session objektet ligger i forbindelsen mellem en Clint og en host med en lifetime på 30 min, her gemmes information vedrørende brugen og brugervariabler og lidt authentiation) Use http’s’ to secure WEB-sessions so they can’t be tampered with. (Man kan faktisk ved et redirect fra http til https ændre I sessionen, men dette er patchet I de fleste nye browsere)

# A5-Broken Access Control

Authorization = What do you have access to do?

Link: <https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A5-Broken_Access_Control>

The application is vulnerable if?

* 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’s 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.