Exercises for OWASP A2 + A5

List of Worst and/or Broken Passwords

Bare på første side i Google:

1. <https://www.securitymagazine.com/articles/89694-the-top-100-worst-passwords>
2. <https://thethreatreport.com/some-of-the-worst-passwords-of-2018/>
3. <https://www.teamsid.com/splashdatas-top-100-worst-passwords-of-2018/>
4. <https://www.teamsid.com/1-50-worst-passwords-2019/>
5. <https://www.secureworldexpo.com/industry-news/top-15-worst-passwords-list>
6. <https://www.nbcnews.com/better/lifestyle/worst-passwords-2019-they-re-so-weak-even-novice-hacker-ncna1106626>
7. <https://www.csoonline.com/article/3244004/top-25-worst-most-insecure-passwords-used-in-2017.html>

Bare på første side i Google:

1. <https://xwyr.polderziegen.de/username-password-combo-list.html>
2. <https://xfinitytechnologies.com/wp-content/uploads/2020/02/rhcwq/common-username-and-password-list.html>

Preventing bad passwords

* Why is this not always as easy as it sounds? - which two “project requirements” often draws in two quite different directions?
  + When you develop system you often have a product owner who prioritizes usability over security. These include security measures such as, passwords of different characters, passwords of a certain length, multifactor authentication, A list to match weak passwords up against, timeouts on login fails. Weak passwords themselves, where a user does not consider their own protection.
* Implement a simple control (feel free to use predefined packages) to verify passwords, up against a set of rules decided by you (length, required character, illegal words etc.)
  + Made a simple java Class that can check for weak or unsecure passwords by length, required characters and illegal words.

### Prevent Brute Force Attacks

Consider ways to prevent brute force attacks and implement a VERY simple proof of concept to illustrate your ideas (in any language/platform you prefer)

* Cooldown on login requests pr. Failed attempt
* CAPTCHA are you human authentication

#### 1) Locate the problem in how Access Control is implemented, and use this knowledge to “steal” other users private info.

First use the link: <http://dat-security.dk/a5demo/>, and follow the instructions given to see whether you can “hack” the site, and get private information from users other than yourself (You are user: [a@b.dk](mailto:a@b.dk) with password: test)

“Hacking“the site:

* Man kan komprimere denne side ved at tilgå Google Chrome develober tools og gå til Network sektionen, hvor vi kan kigge på de get eequests der bliver lavet. Da vi allerede er logget ind som en bruger kan vi kigge på den tilhørende Get request til ”get personal” data funktionen og brug det request url: ”http://dat-security.dk/a5demo/api/data/b@b.dk” der ligger i Get requestet. For dette url kan vi så ændre brugernavnet og lave en ny Get request på det brugernavn i browseren for at få denne persons data.

Hvad kunne man gøre?

* Man kan bruge en Grit til at ændre sit brugernavn/session-id i den tilhørende request, så man ikke kan gætte sig frem til andre brugeres brugernavne. Dette er dog ikke nødvendigvis en endelig løsning da man stadig vil kunne tilgå andre brugeres sider, hvis man på en eller anden måde får fat i dette brugernavn eller id selv efter Grit.
* Man kan hashe sin basic authentication ‘YUBiLmRrOnRlc3Q=’ I stedet for bare at enkryptere det med base 64 (base 64 er lige så usikkert som plain tekst siden man bare kan dekryptere det nemt). Her skal man også være klar over at siden bruger http og ikke http’s’ hvilket ikke er sikkert over for nogle der sidder og lytter med og eventuelt kunne opfange mit brugernavn og password som sendes med i requesten. Basic authentication minder meget om WEB-tokens.
* Lad være med at gemme eller videresende værdier i plain text.
* Man kan kontrollere om det er noget som den konkrete bruger har adgang til at fremvise.

2) Fix the problem

The first fix we can do is to look at the default name JESSIONID, since it discloses the underlying technology and language used by the server. This can be done in context.xml by Setting the cookie name of the session-id to something else so the underlying values cannot be seen directly. This can also help make it a little less obvious.

Regarding a fix for the ability for another user to use the Get request itself to access other users account information, we would have to control the user’s authentication every time they try to access something of priority. We could also change the name variables to something else than plane text, so they aren’t so easy to come across and predict.