Exercises for XSS,  OWASP A7

*First of all, make sure, if possible, to attend the lecture and the demo/competition held in the class.*

Reflected XSS

Implement your own simple server, or use [this](https://github.com/securitydatspring2019/xssDemoServer.git) (see readme.md for how to set it up), with a proof of concept page, to demonstrate reflected XSS.

If you use the provided server above, feel free to use this [mail](https://docs.google.com/document/d/19Xz6eTfeBxK1js8evmRjzJyGI7Hh25VwxpP-EFkzcCs/edit?usp=sharing) as inspiration for the phishing part.

Remember, we (the hacker) will probably only succeed if the victim is using FireFox or IE.

Stored XSS:

Demonstrate an attack as outlined in [this video](https://www.youtube.com/watch?v=cbmBDiR6WaY) (which you hopefully already have watched). This [snippet file](https://docs.google.com/document/d/1sGegv6K4DeDuTAmXMhepzAaeDfDjIqWNIpj-jLL-kcI/edit) includes setup instructions for the simple “evil hacker server” and also a script you can try to inject.

Feel free to use my [demo-server](https://github.com/Lars-m/xssServer.git) (same as above) for your demo’s.

Alternative ways to Demonstrate XSS

The OWASP Juice Shop Application includes several XSS related challenges. Some can be hard to find but Google for the Juice solutions, and then just do/demonstrate the problems.

Prevent XSS-Attacks

Make sure to read/skim (if not already done) the OWASP <https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html> before you start.

Getting Started:

Clone this maven project as start code for this exercise: <https://github.com/securitydatspring2019/sanitazion-startcode>

It includes only three things of interest. An index.html page, a Servlet class (Sanitizer.java) and the dependencies referred to below, which you must use to Encode and/or Sanitize user inputs to the servlet.

<dependency>

    <groupId>org.owasp.encoder</groupId>

    <artifactId>encoder</artifactId>

    <version>1.2.2</version>

 </dependency>

 <dependency>

   <groupId>com.googlecode.owasp-java-html-sanitizer</groupId>

   <artifactId>owasp-java-html-sanitizer</artifactId>

   <version>20191001.1</version>

 </dependency>

The dependencies above are for these two OWASP projects, and you will find all you need to know  (which is very little) at these links:

* <https://github.com/OWASP/owasp-java-encoder/wiki/2)-Use-the-OWASP-Java-Encoder>
* <https://github.com/OWASP/java-html-sanitizer>

Use the libraries above to solve the following problems:

Given this input string which you get from the index-page:

**Hacky <b>hack</b> the <em>cool</em> Hacker <script>alert('hi')</script>**

Change the servlet to return strings as outlined below and in the code:

* Encode the string so, if rendered in a browser, we will see exactly the text (with all characters) given above
* Sanitize the string to allow for simple formatting, but NOTHING else. That is, if the input string was rendered in a browser, we should see this text (bold and italic): Hacky **hack** the *cool* Hacker
* Sanitize the string to allow no tags at all. That is, if the input string was rendered in a browser, we should see this text: Hacky hack the cool Hacker

See this deployed project for how it should end: <http://www.dat-security.dk/sanitize/index.html>

**Vores løsning**

Billede af HTML-siden på browseren:

Et billede, der indeholder indendørs, skærmbillede, fugl

Automatisk genereret beskrivelse

Tilføjet dependencies i pom.xml:

Et billede, der indeholder fugl

Automatisk genereret beskrivelse

Billede af servlet, hvor encoder og sanitizing er implementeret:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse