SVS | Exercise 4

Task 1

1. Was ist Cross-Site Scripting (XSS)?
2. Platzieren Sie in das Verzeichnis Ihres PHP auslieferenden Servers aus Tutorial 3 die Datei *questbook.php.* Die PHP-Datei implementiert eine einfache Gästebuch-Seite. Gästebuch-Einträge werden in einer XML-Datei abgespeichert.
   a. Wie kann man mit XSS-Technologien einen Knopf erzeugen, dessen Betätigung zur Änderung der Farbe der Überschrift der Gästebuchseite führt?
   b. Wie kann man mit XSS-Technologien einen Knopf erzeugen, dessen Betätigung zur Änderung der Links im oberen Bereich der Seite führt? Die Links sollen ihre Aufschrift beibehalten. Ein Klick auf die Links soll aber den Nutzer zu http://www.google.de schicken!
   c. Geht beides auch ohne Knöpfe?
3. Welche Schutzmaßnahmen gegen XSS sind Ihnen bekannt?

1. What is Cross-Site Scripting (XSS)?
2. Place the file *guestbook.php* into your PHP delivering server folder from tutorial 3. The script implements a simple guestbook application. The data will be stored in a XML file.
   a. Using the XSS technique create a button, which, if pressed, changes the color of the page header.
   b. Using the XSS technique create a button, which, if pressed, changes the links in the top area of the page. The links should keep their labels but, if clicked, should direct a user to http://www.google.de
   c. Are the actions a and b possible without buttons?
3. Which defense methods against XSS do you know?

Task 2

Die Seite http://mytuc.org/jmcp simuliert eine Suchmaschine, die die 5 letzten eingegebenen Suchbegriffe eines Nutzers merkt.

a. Was ist der Unterschied zwischen den *stored*, *reflected* bzw. *DOM-based* XSS Angriffen?
b. Wie kann ein Angreifer die letzten von einem Nutzer eingegebenen Suchbegriffe ausspionieren?

The page http://mytuc.org/jmcp simulates a search engine, which remembers 5 search queries entered last.

a. What is the difference between *stored*, *reflected* and *DOM-based* XSS attacks?
b. How can an attacker spy on the search history of a user?

Task 3

Unter http://mytuc.org/sgdf finden Sie eine Seite, die zwei weitere Seiten als iframes lädt (Quellcode: *iframes.php* sowie *content.html*). In dem Eingabefeld unten kann Javascript eingegeben werden, dass sofort vom Browser ausgewertet wird (als ob dieser Script gleich in der ursprünglichen Seite eingebettet wäre).

a. Was kann man eingeben um die Inhalte der H1-Elemente aus den beiden iFrames auszugeben?
b. Bei welchem der iFrames klappt das und warum (Stichwort Same-Origin-Policy)?
c. Was bezweckt die dahinterstehende Sicherheitsmaßnahme?

At http://mytuc.org/sgdf one can find a page, which loads two further pages into iframes (Source: *iframes.php* and *content.html*). Into the input field below one can enter Javascript, which will be immediately evaluated by the browser (as if it would be embedded into the original page).

a. What can be entered to read the H1-elements of the both iframes?
b. For which frame do your commands work and why (keyword Same-Origin-Policy)?
c. What is the goal of the security measure behind?

Task 4

Wie erfolgt ein Cross-Site-Request-Forgery (CSRF) Angriff? Angenommen, ein Nutzer hat sich auf der Seite http://mytuc.org/vwfx angemeldet (siehe Übung 3, z.B. mit dem Nutzername „Max“ und Passwort „Mustermann“). Wie kann man mittels der Gästebuchseite aus der Aufgabe 1 diesen Nutzer gegen seinen Willen ausloggen?

How a Cross-Site-Request-Forgery (CSRF) attack takes place? Assumed, a user has logged in into http://mytuc.org/vwfx (cf. Tutorial 3, e.g. using username „Max“ and password „Mustermann“). How can one log out the user against his will using the guest book from the task 1?