CS 410: Web Security
A3/A10: Labs and Homework

A3 WFP1: XSS

For all WFP1 XSS levels, you will need to disable your browser's XSS auditor. (See Developer Tools to see if the XSS auditor is blocking the exploit). On Chrome, you can disable it via google-chrome --disable-xss-auditor.

- **Example #1**
  - **Inject JavaScript into the script's name parameter that uses JavaScript's document.documentElement.innerHTML attribute to display the contents of the page's HTML in an alert box**
- **Example #2**
  - **Filtering has now been applied, but does not take into account that tags in HTML are not case-sensitive**
  - **Bypass the filter and invoke JavaScript's location.assign() function to display the contents of the CS 410 CTF site (`cs410.oregonctf.org`)**
- **Example #3**
  - **Filtering has been altered, but only uses a single pass to remove problematic tags.**
  - **Bypass the filter and inject JavaScript that will pop up an alert box with your name in it**
- **Example #4**
  - **Tag filtering has eliminated the ability to inject script tags. However, JavaScript functions can be issued on specific events within other tags (`onmouseover, onmouseout, onmousemove, onclick, onerror, href='javascript:…'`)**
  - **Use an attribute of an `<a>` ,`<div>`, or `<img>` tag to inject JavaScript that will pop up an alert box with your name in it**
- **Example #5**
  - **Filter allows `<script>` tag, but not `alert` keyword**
  - **Static keyword filters can be bypassed by code that dynamically generates the keyword at run-time**
  - **Use JavaScript's `eval` and `String.fromCharCode()` to inject JavaScript that will pop up an alert box with your name in it. Alternatively use the `prompt` or `confirm` functions to do the same**
- **Example #6**
  - **In looking at the HTML, the value entered is echoed inside the JavaScript code**

    - **This makes it vulnerable to code injection**
    - **Inject the appropriate character to terminate the instruction being injected**
    - **Issue another JavaScript command that pops up an alert box with your name in it**
    - **You may need to insert a comment character at the end to remove the original characters in command being injected (//)**
- **Example #7**
    - **The page employs PHP's htmlentities to encode characters. Unless the `ENT_QUOTES` flag is enabled, the function will not encode single quotes**
    - **Perform the same attack as in #6, but with single-quotes**
- **Example #8**
    - **To build the form, the developer employs and trusts `PHP_SELF` which is controlled by the user via the URL**
    - **Go to**

`http://<wfp1 site>/xss/example8.php/stuff`

    - **View the form attribute to see how `PHP_SELF` has been used**
    - **Use this observation to invoke JavaScript's `location.assign()` function to redirect access to the CS 410 CTF site (`cs410.oregonctf.org`)**
    - **Your payload may need to be properly URL-encoded**
- **Example #9**
    - **In the page source, examine the JavaScript that produces the page output**
    - **Use this to inject JavaScript that will display the contents of the page's HTML in an alert box**

A3 XSS X-XSS-Protection

- **For these levels, only screenshots and answers to questions are needed**
- **Setup a NodeJS/Express server on localhost at port 1234**
    - **mkdir xss**
    - **cd xss**
    - **wget http://thefengs.com/wuchang/courses/cs410/files/xss_server.js**
    - **npm install express**
    - **npm install request**
    - **nodejs xss_server.js**
- **Example #1**

    - **Generate a pop-up by visiting http://localhost:1234/?user=<script>alert(document.documentElement.innerHTML)</script>&xss=0**
- **Example #2**
    - **Open Developer Tools console**
    - **Goto http://localhost:1234/?user=<script>alert(document.documentElement.innerHTML)</script>&xss=1**
        - **What part of the page did the auditor remove?**
- **Example #3**
    - **Open Developer Tools console**
    - **Goto http://localhost:1234/?user=<script>alert(document.documentElement.innerHTML)</script>&xss=1; mode=block**
        - **What is different from Example #2?**
- **Example #4**
    - **Open Developer Tools and go to Network => All**
    - **Goto http://localhost:1234/?user=<script>alert(document.documentElement.innerHTML)</script>&xss=1; report=http://localhost:1234/report**
    - **Show the report request that is initiated**

A3 XSS Content-Security-Policy

- **For these levels, only screenshots and answers to questions are needed**
- **Setup NodeJS/Express servers on localhost at port 1234 and 4321 using previous location**
    - **cd xss**
    - **wget http://thefengs.com/wuchang/courses/cs410/files/csp_server.js**
    - **nodejs csp_server.js**
    - **Substitute your username for OdinID in links below**
- **Example #1**
    - **Generate page that allows all three scripts to change base HTML**
    - **http://localhost:4321/?user=OdinID**
- **Example #2**
    - **Set policy to be none (most restrictive)**
        - **Open Developer Tools => Console**
        - **Dump page output and console output of request to show all scripts blocked (as well as favicon.ico image)**
        - **http://localhost:4321/?user=OdinID&csp=default-src 'none'**

- **Example #3**
    - **Set policy to allow same-origin**
        - **Open Developer Tools => Console**
        - **Dump page output and console output of request to show origin JavaScript and images loaded while others blocked**
        - **http://localhost:4321/?user=OdinID&csp=default-src 'self'**
- **Example #4**
    - **Set policy to allow same-origin**
        - **Open Developer Tools => Console**
        - **Dump page output and console output of request to show inline and origin JavaScript and images loaded**
        - **http://localhost:4321/?user=OdinID&csp=default-src 'self'; script-src 'self' 'unsafe-inline'**

AX Unsolicited Framing (Clickjacking)

- **For this lab, only screenshots are necessary**
- **Use `nc -C` to connect to google.com and find what it uses for its `X-Frame-Options:` header**
- **Then, create an HTML with the following and open it in the browser. See if the site loads and if not, what the error is in the console.**

```
<html></html> <body></body> <iframe src="https://www.google.com"
width="800" height="800"></iframe>
```

A3 Homework

- **Lessons: Cross Site Scripting**
- **Challenges: XSS #1-5**

A10 Homework

- **Lessons: Unvalidated Redirects and Forwards**