The Chinese University of Hong Kong

Department of Information Engineering

Summer Workshop – Fun with Information Engineering and Security

Lab 5 – Web Security 1

Introduction

The web is important in today's world, serving as a channel for communication and information sharing. As the web becomes increasingly integrated into our daily lives, it also attracts malicious actors seeking to exploit vulnerabilities for personal gain. Web security is essential to protect users and data from the myriad of threats that exist online.

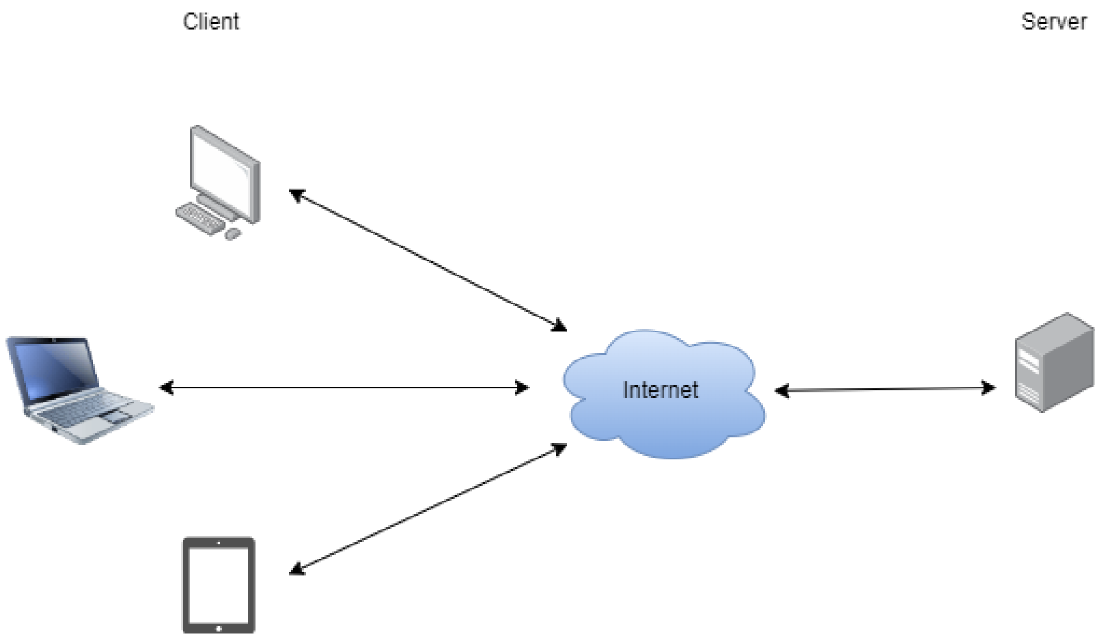
In this lab, we will focus on web security. We will learn about common vulnerabilities in web applications, including cross-site scripting. Through practical exercises, we will gain skills in identifying and mitigating these security risks, equipping them with essential knowledge for safeguarding web environments.

Section 1 Web Security 1

Section 1.1: Basic concepts of web

Client-Server model

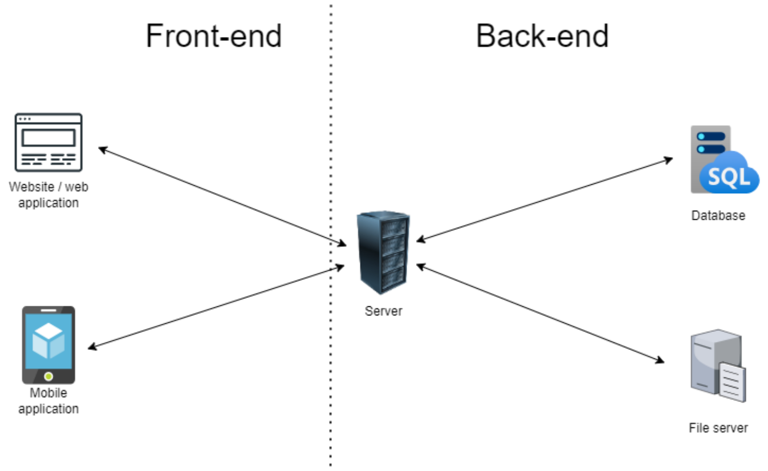
The client-server modal is a network architecture where clients request services or resources from servers, which then provide the necessary responses or data.



Client is usually the web browser or application that interacts with the server to request services or data. End user can interact with the client directly. Server usually operates with powerful computers. It hosts applications, executes server-side logic and stores data.

Frontend and Backend

The frontend and backend architecture is a design pattern in application development:



Frontend is the interface that runs on the client side. It provides the interaction elements for the end user, including buttons and client-side logic. Backend is where the application runs on the server side. It handles data processing, authentication and database interaction.

For example, when you are shopping online:

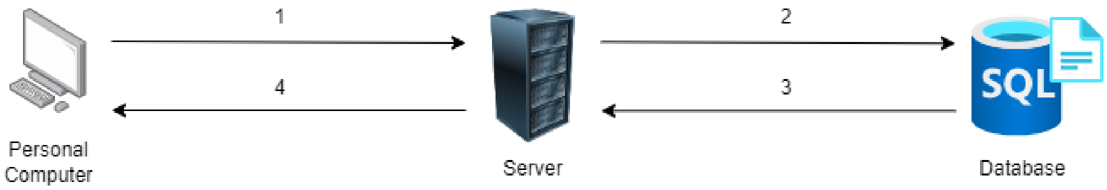
Client: browser (e.g. Chrome, Firefox)

Front-end: the online shopping website

Server: the computer that stores your personal information, product quantity…etc.

Backend: the program that the server uses, e.g. handles the user login logic, interacts with the product database…etc.

What happens behind the scenes when users want to get information about a product:



1. User would like to check the information on product A, the browser sends a request to the server
2. The server received the request, the backend logic interacts with the database server to query the information of product A
3. The database server returns the information of product A (e.g. product description, price) to the server
4. The server sends a response to the computer, the information is rendered as a website on the user's computer

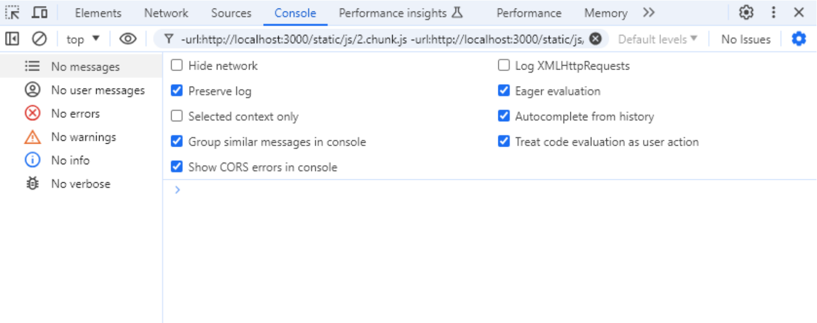
Section 1.2 Javascript and Event

JavaScript is a programming language primarily used for creating and controlling dynamic website content. It is often used alongside HTML and CSS for front-end user interfaces. It enables interactive web pages and is an essential part of web applications.

Try it!

Let's try some simple javascript function in the developer console:

1. Start chrome and enter about:blank in the URL
2. Click ⋮ on the top right-hand corner, hover on More tools, and select <> Developer tools. You can also simply enter developer tools by F12
3. Go to console



□ Hide network
☑ Preserve log
□ Selected context only
☑ Group similar messages in console
☑ Show CORS errors in console
□ Log XMLHttpRequests
☑ Eager evaluation
☑ Autocomplete from history
☑ Treat code evaluation as user action

4. You may try some simple JavaScript programming. What can the following code snippet do?

```
var a = 2; // Declare a variable "a"
var b = 3; // Declare a variable "b"
alert(a + b); // Prompt a pop up window, with content of "a + b" (i.e. 5)
```

In programming, events are things that happen in the system (e.g. when the user clicks on a button, when the image loads to the page). If we want to implement logic to handle the event change, we will need an event listener.

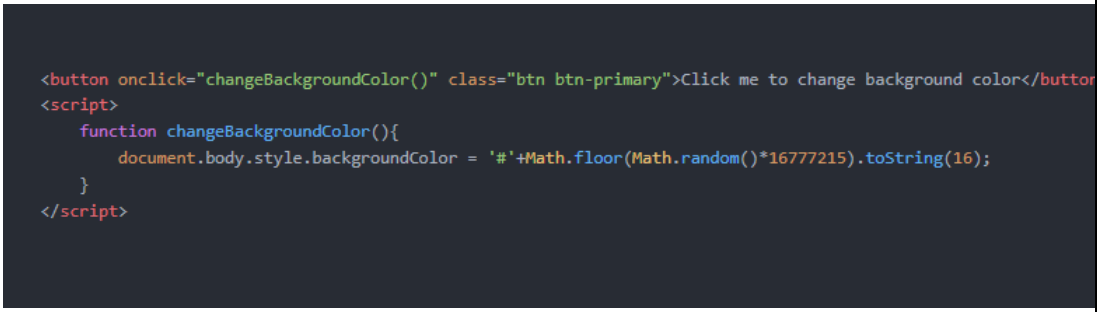
Try it!

Let’s try some javascript event listener

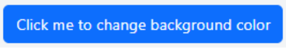
Go to https://iesummerworkshop.github.io/event-listener.html

1. Onclick: when the user clicks on the object

In this page, the background will be changed when the user clicks on the button



```
<button onclick="changeBackgroundColor()" class="btn btn-primary">Click me to change background color</buttor
<script>
    function changeBackgroundColor(){
        document.body.style.backgroundColor = '#'+Math.floor(Math.random()*16777215).toString(16);
    }
</script>
```

Click on the button . What are the changes?

2. Onmouseover: when the user points the mouse to the object

Move the mouse pointer to , what happened?

3. Onkeydown: when the user presses any key of the keyboard

Press the left and right arrow keys, what happened?

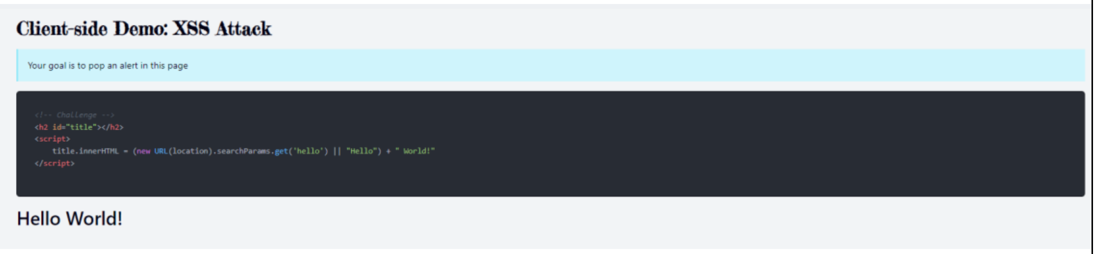
You may check out more Javascript events on https://developer.mozilla.org/en-US/docs/Web/Events

Section 1.3: XSS attack

Cross-site scripting (XSS) is a web application vulnerability that allows attackers to inject malicious scripts into web pages and violate SOP. It is one of the OWASP top 10 web application security risks. XSS occurs when the webpage does not sanitize the user’s input properly.

Try it!

Go to https://iesummerworkshop.github.io/xss.html



Client-side Demo: XSS Attack

Your goal is to pop an alert in this page

```
<!-- Challenge -->
<h2 id="title"></h2>
<script>
    title.innerHTML = (new URL(location).searchParams.get('hello') || "Hello") + " World!"
</script>
```

Hello World!

Your goal for this lab exercise is to pop an alert on this page:

1. Given the script of this website, observe if there are any input fields we can launch this attack
2. Craft the attack payload using javascript

Solution

As you may observe, the script is taking the value of the query parameter "hello". We may launch an XSS attack by changing the query parameter. Craft the attack payload: `?hello=<svg%20onload="alert(123)"></svg>`

The URL should look like:

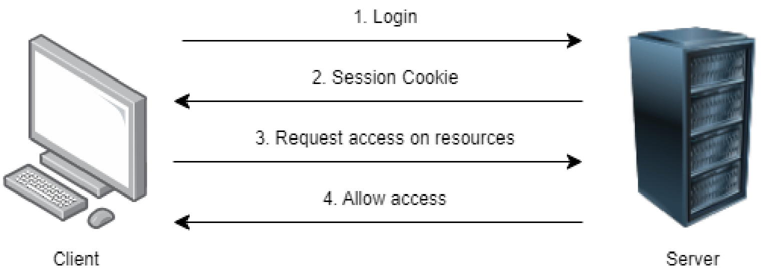
`https://iesummerworkshop.github.io/xss.html?hello="<svg%20onload="alert(123)"></svg>`

Refresh the page and see the result. You may also try the below payload:

- `?hello=<iframe%20src=""%20onload="alert(123)"></iframe>`
- `?hello=<xss%20onmousemove="alert(1)"%20style=display:block>test</xss>`

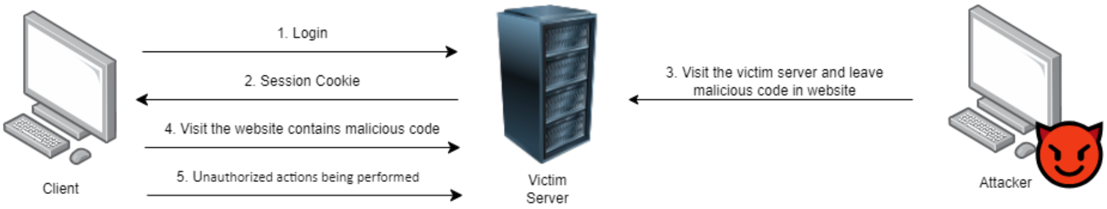
What is a cookie

Cookie is a small piece of information stored on the client side. They are used to store session information that allows a web server to recognize and authenticate users across multiple requests. The following steps explain how cookies help in session management:



1. User inputs credentials (username and password) into the website, the client sends a login request to the server
2. Server authenticates the user successfully. The server stores the session information of the user and returns a session ID to the client. The client stores the session ID by cookie
3. The user requests access to server resources that are limited to logged-in users only. The client attaches the cookie automatically to the request
4. The server received the cookie information and query on the session information. The user is having a valid session and the resources are returned to the user

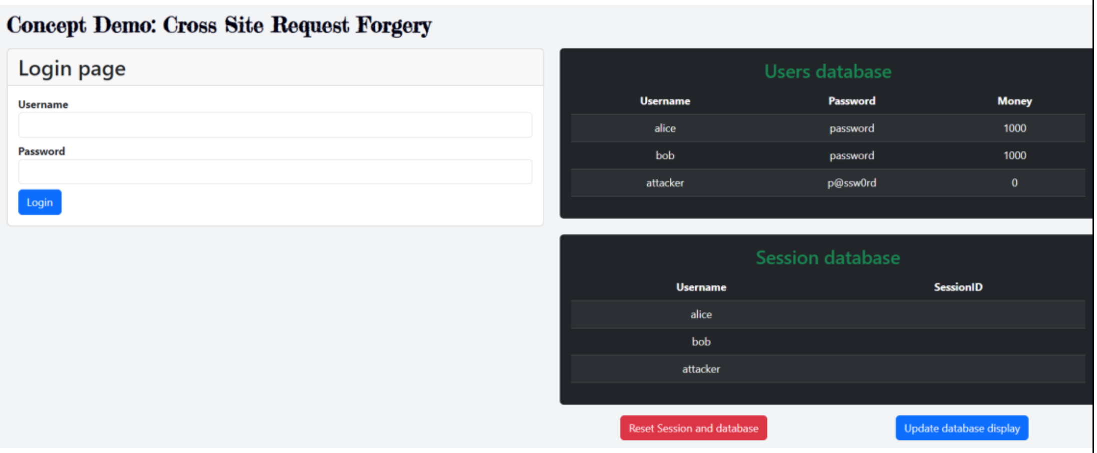
If cookies are not handled properly, it can lead to unauthorized actions being performed on behalf of the authenticated user without the user's consent. The following shows the steps shoes on how bad things could happen:


Client
1. Login
2. Session Cookie
4. Visit the website contains malicious code
5. Unauthorized actions being performed
Victim Server
3. Visit the victim server and leave malicious code in website
Attacker

1. User inputs credentials (username and password) into the website, the client sends a login request to the victim server
2. Victim server authenticates the user successfully. User receives a session cookie
3. The attacker visit the server and leave malicious code in the website using XSS attack
4. The user visit the website that contains the malicious code
5. Unauthorized actions being performed on behalf of the authenticated user without the user's consent

Try it!

Start a new browser in incognito mode. Go to https://iesummerworkshop.github.io/bank.html



Concept Demo: Cross Site Request Forgery

Login page

Username

Password

Login

Users database

Username	Password	Money
alice	password	1000
bob	password	1000
attacker	p@ssw0rd	0

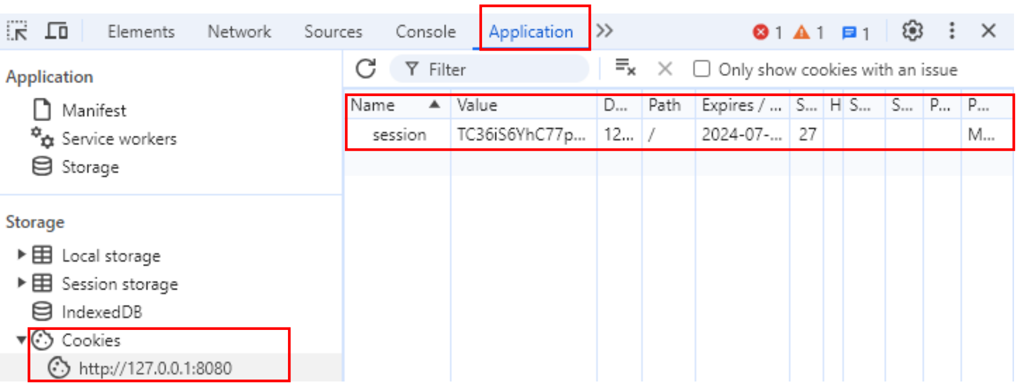
Session database

Username	SessionID
alice	
bob	
attacker	

Reset Session and database

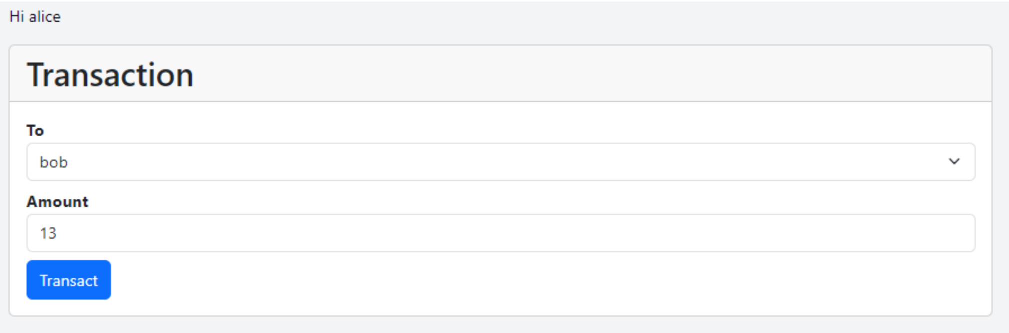
Update database display

1. Login with Alice or Bob
2. After clicking login, you should see the session database updated with a session ID
3. Open the developer console by F12, go to the application → Cookies. Click on the URL, You should see a new cookie called "session", the value should be the same as the one we have in the session database



Name	Value	D...	Path	Expires / ...	S...	H	S...	S...	P...	P...
session	TC36iS6YhC77p...	12...	/	2024-07-...	27					M...

4. Select a person you would like to transfer money and input the amount. Click on Transact afterward.

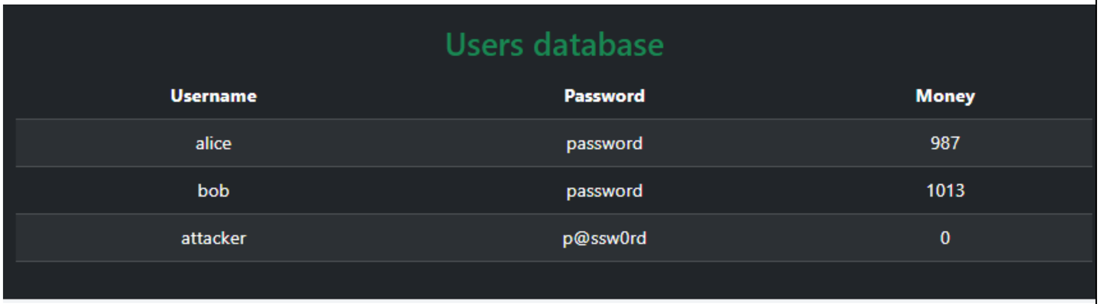

Hi alice

Transaction

To
bob

Amount
13

5. You should see the money transferred from alice the bob successfully


Users database

Username	Password	Money
alice	password	987
bob	password	1013
attacker	p@ssw0rd	0

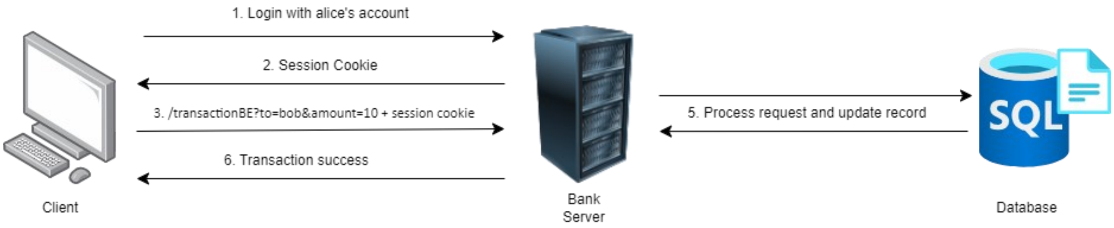
…and now, the thing goes wrong…

6. Open a new tab within the same browser and visit https://iesummerworkshop.github.io/bank-forum.html, it is a forum page hosted by the bank, where the attacker leaves some malicious comments it will be loaded to the user's computer
7. Wait until the page is loaded. Go back to https://iesummerworkshop.github.io/bank.html tab.
8. Click on Update database display, what are the changes on the user database?
9. Repeat steps 6 to 8, any changes?

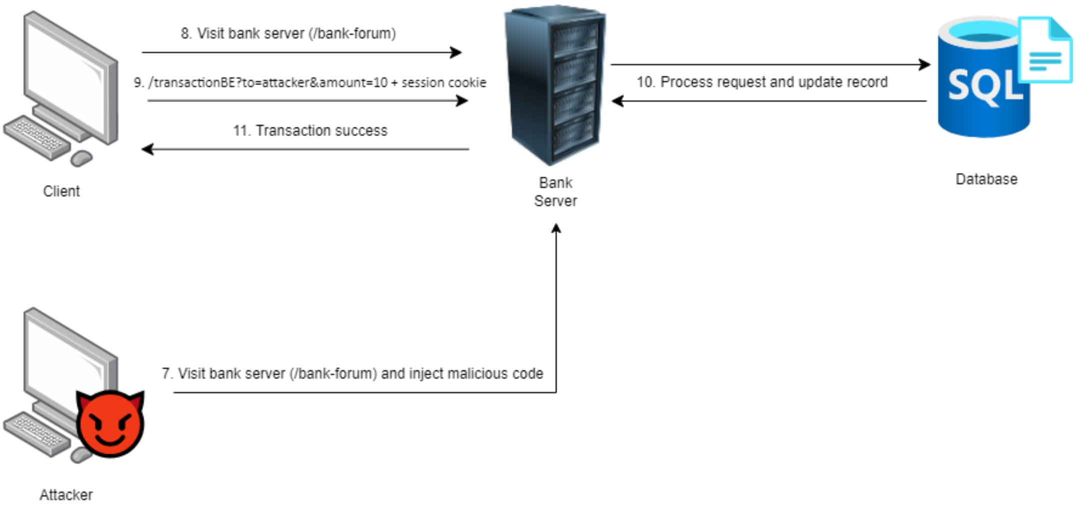
What happened during a normal transaction?

1. User logged in with alice's account
2. Server authenticates alice successfully and returns a session cookie
3. When alice requests to transact, a request is sent to the server, e.g. `/transactionBE?to=bob&amount=10`. The session cookie is being sent to the server together
4. The server received the request. From the session ID, the server recognized alice was making the transaction request
5. The server processes the transaction request and updates the record in the database

6. Transaction successful


1. Login with alice's account
2. Session Cookie
3. /transactionBE?to=bob&amount=10 + session cookie
6. Transaction success
5. Process request and update record
Client
Bank Server
SQL
Database

7. The attacker visited the bank forum and injects a malicious code in the comment section by XSS attack (i.e. `<iframe src="/transactionBE.html?to=attacker&amount=10" width="0" height="0"></iframe>`)
8. Alice visit the bank forum which contains the malicious code
9. The browser sends the forged request `/transactionBE.html?to=attacker&amount=10` to the bank server
10. The server processes the transaction request and updates the record in the database
11. Transaction to attacker successful


8. Visit bank server (/bank-forum)
9. /transactionBE?to=attacker&amount=10 + session cookie
11. Transaction success
10. Process request and update record
7. Visit bank server (/bank-forum) and inject malicious code
Client
Bank Server
SQL
Database
Attacker

Mitigation of XSS

To defend XSS in the web application, the following are some strategies we can apply:

1. Sanitize user input

Ensure that any user input is sanitized before processing. This can involve removing or escaping potentially dangerous characters e.g. < >.

2. Output encoding

If adding a variable to HTML content is needed, HTML entity encoding would help:

- Converting `<` to `&lt;`
- Converting `>` to `&gt;`

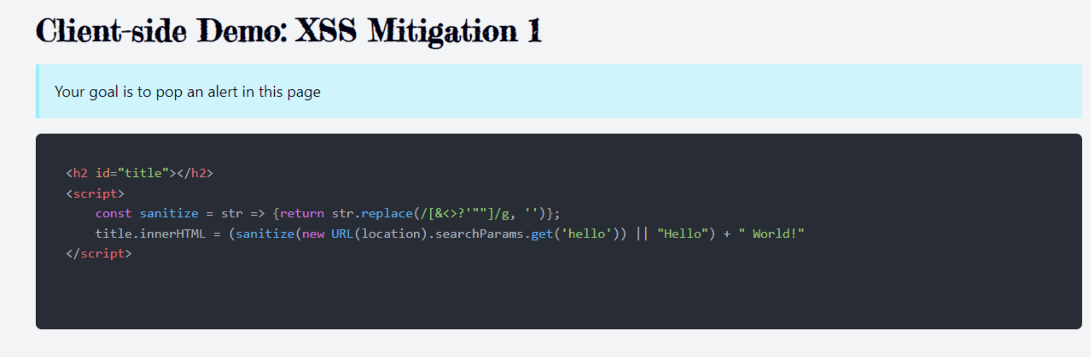
3. Input validation

Input validation should work by blocking invalid input. For example, if the form requests the user to input a phone number, only numbers and some symbols are allowed in the input field.

Try it!

1. Sanitize user input

Go to https://iesummerworkshop.github.io/xss-mitigate-1.html



Client-side Demo: XSS Mitigation 1

Your goal is to pop an alert in this page

```
<h2 id="title"></h2>
<script>
    const sanitize = str => {return str.replace(/[&<>?'""]/g, '')};
    title.innerHTML = (sanitize(new URL(location).searchParams.get('hello')) || "Hello") + " World!"
</script>
```

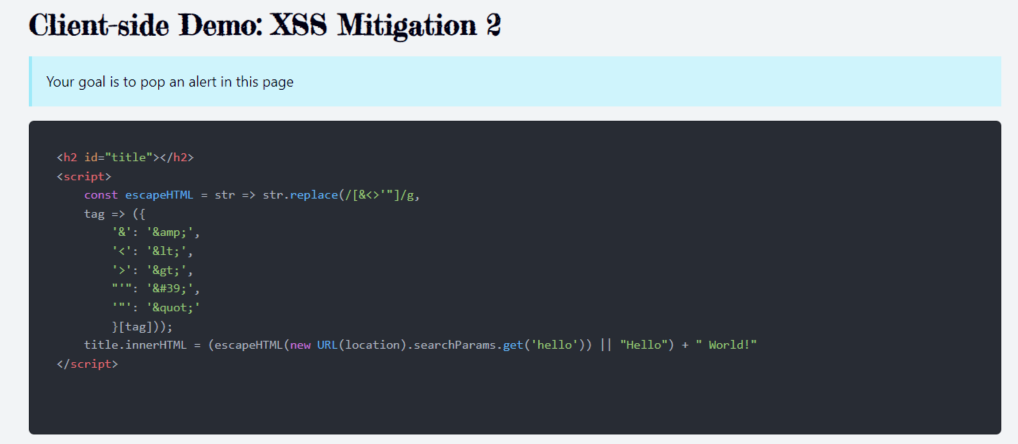
On this page, we are removing the potentially dangerous character. Try the same attack.

- What is the output?
- Does the attack work?

2. Output encoding

Go to https://iesummerworkshop.github.io/xss-mitigate-2.html

On this page, we are encoding the potentially dangerous character to HTML entity



Client-side Demo: XSS Mitigation 2

Your goal is to pop an alert in this page

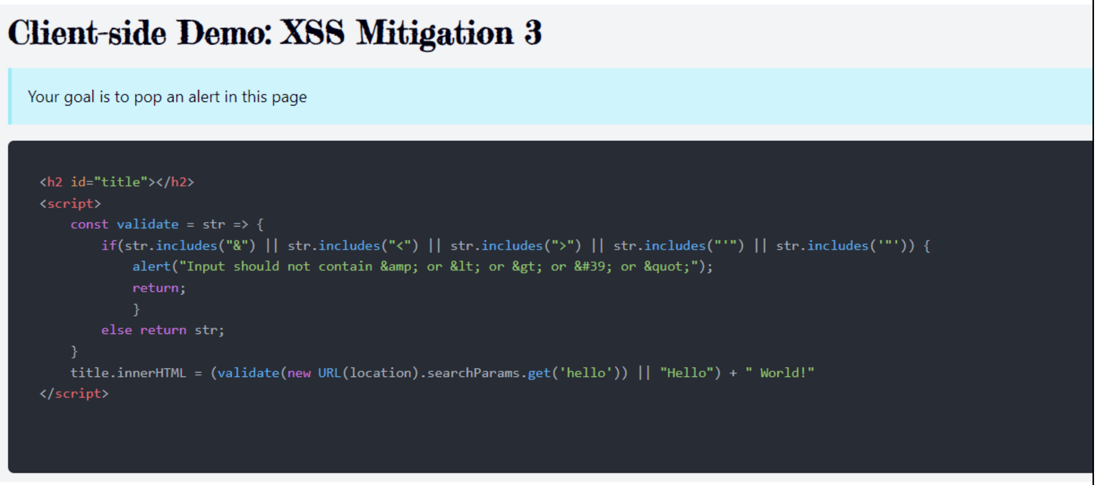
```
<h2 id="title"></h2>
<script>
    const escapeHTML = str => str.replace(/[&<>'"]/g,
    tag => ({
        '&': '&amp;',
        '<': '&lt;',
        '>': '&gt;',
        "'": '&#39;',
        '"': '&quot;'
        }[tag]));
    title.innerHTML = (escapeHTML(new URL(location).searchParams.get('hello')) || "Hello") + " World!"
</script>
```

- What is the output?
- Does the attack work?

3. Input validation

Go to https://iesummerworkshop.github.io/xss-mitigate-3.html

On this page, we are checking if the string contains a potentially dangerous character. If yes, an alert will be prompted.



Client-side Demo: XSS Mitigation 3

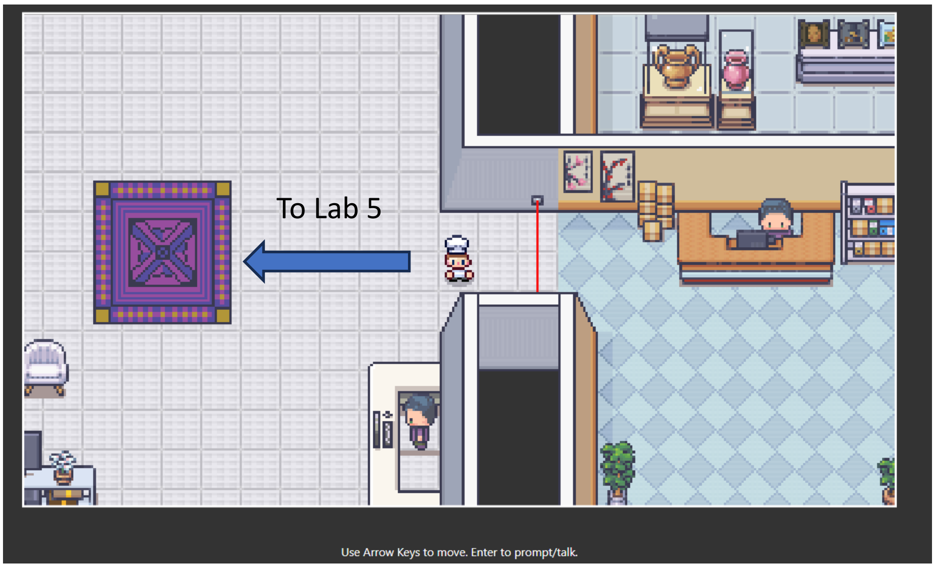
Your goal is to pop an alert in this page

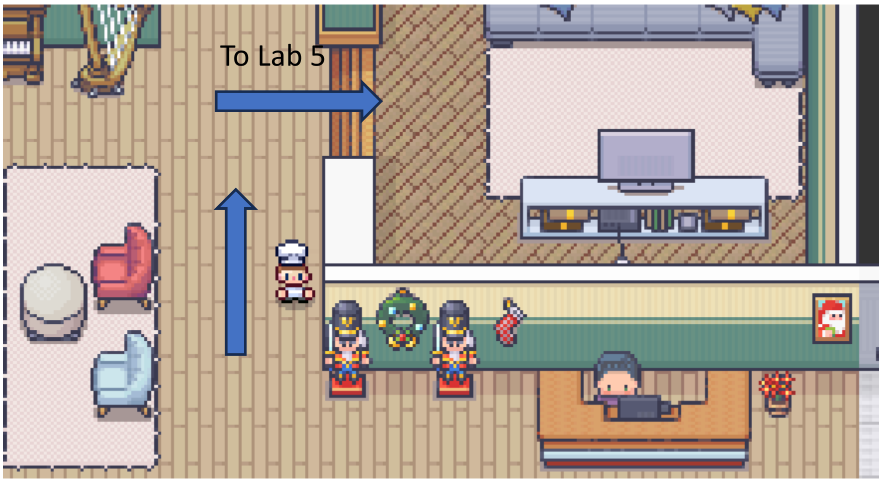
```
<h2 id="title"></h2>
<script>
    const validate = str => {
        if(str.includes("&") || str.includes("<") || str.includes(">") || str.includes("'") || str.includes('"')) {
            alert("Input should not contain &amp; or &lt; or &gt; or &#39; or &quot;");
            return;
            }
        else return str;
    }
    title.innerHTML = (validate(new URL(location).searchParams.get('hello')) || "Hello") + " World!"
</script>
```

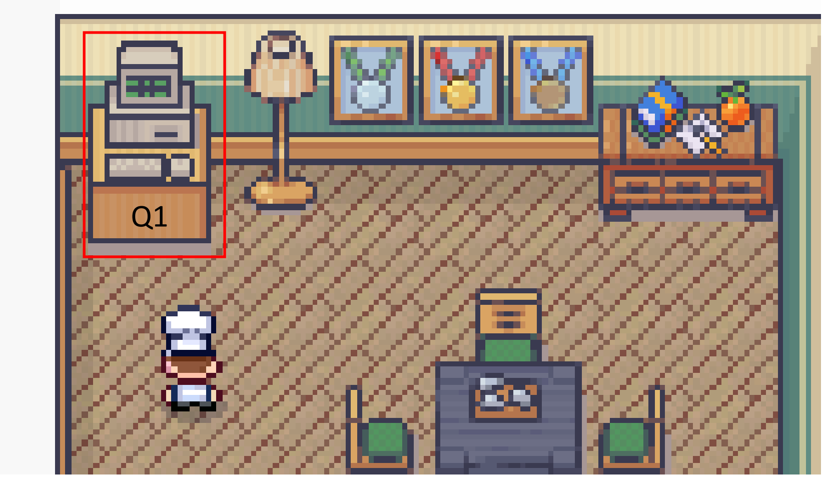
- What is the output?
- Does the attack work?

Section 2: Challenge Time

Now, please go to https://iesummerworkshop.github.io/rpg/page.html to enjoy your challenge! Walk in front of the computer and press enter to interact with it. Mark down the key you get!

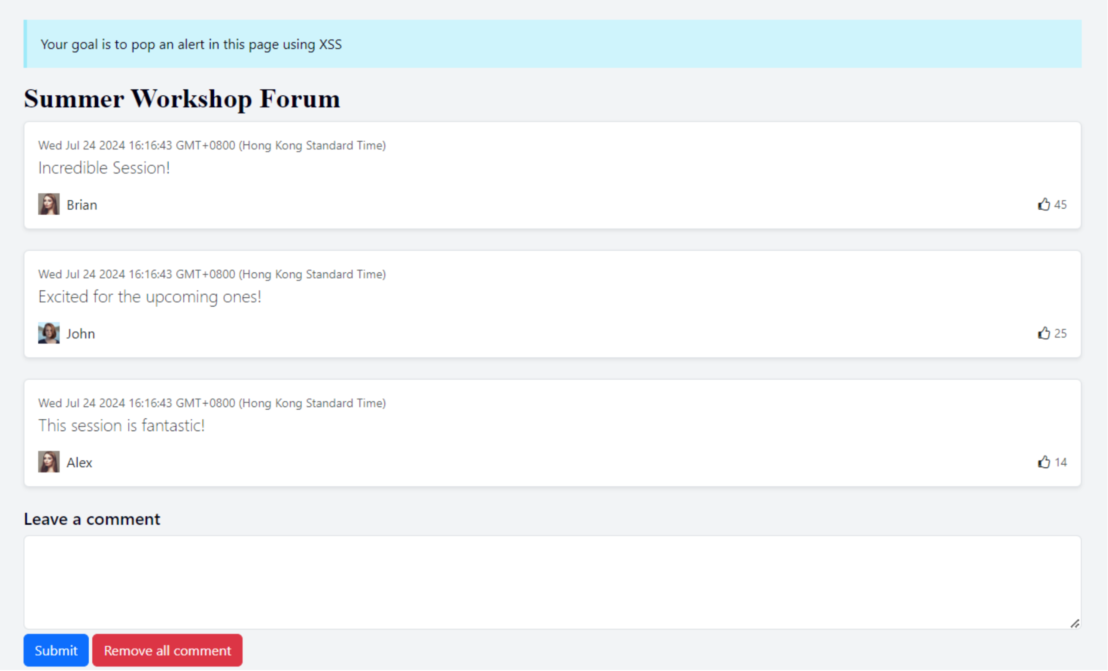






1. [☆ ☆ ☆] XSS

Go to https://iesummerworkshop.github.io/xss-lab.html, your goal is to pop an alert on this page.



Your goal is to pop an alert in this page using XSS

Summer Workshop Forum

Wed Jul 24 2024 16:16:43 GMT+0800 (Hong Kong Standard Time)
Incredible Session!
Brian 45

Wed Jul 24 2024 16:16:43 GMT+0800 (Hong Kong Standard Time)
Excited for the upcoming ones!
John 25

Wed Jul 24 2024 16:16:43 GMT+0800 (Hong Kong Standard Time)
This session is fantastic!
Alex 14

Leave a comment

Hints

A. Is there any input field where we can launch an attack?

B. You may check on the link in section 1.2, it should provide some useful event listeners that may help with completing this question.

Interact with this counter to submit the key for lab 5 and lab 6