

Experiment 1.2

Student Name : Ritik Pathania
UID : 20BCS1743
Section : 601 B
Subject Code : 20CSP-338
Subject Name : Web and Mobile Security Lab

Aim: Design a method to simulate the HTML injection and cross-site scripting to exploit the attacker.

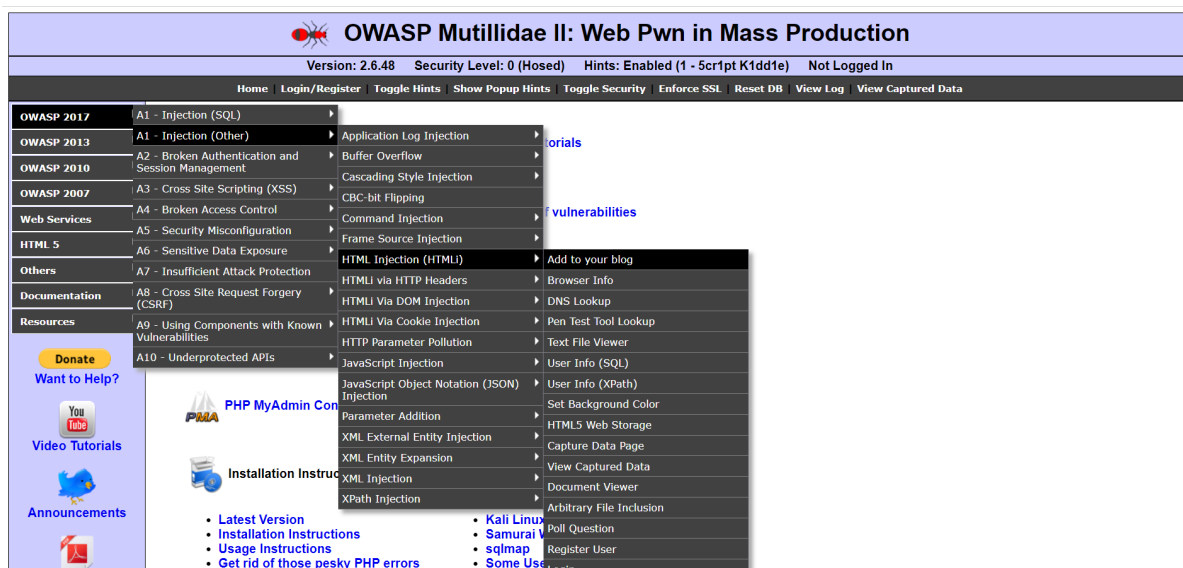
Requirements: PC with Windows 7 or above.

Steps for the experiment: HTML injection

1. Open the website:

[OWASP Mutillidae II: Web Pwn in Mass Production](#)

Now, we'll be redirected to the web page which is suffering from an **HTML Injection vulnerability** which allows the user to submit his entry in the blog as shown in the screenshot.



2. Now, let's try to inject malicious code Enter the HTML code inside the given text area to set up the HTML attack.

The screenshot shows the OWASP Mutillidae II: Web Pwn in Mass Production interface. The top navigation bar includes links like Home, Login/Register, Toggle Hints, Show Popup Hints, Toggle Security, Enforce SSL, Reset DB, View Log, and View Captured Data. The main content area is titled "Welcome To The Blog" and features a "Back" button, a "Help Me!" button, and a "Hints and Videos" section. A "Add New Blog Entry" form is visible, containing a text area with the HTML code: `<td/> Ritik <marquee> こんにちは元気ですか </marquee>`. Below the form is a "Save Blog Entry" button. A "View Blogs" link is also present. At the bottom, a table titled "5 Current Blog Entries" shows a single entry with the name "anonymous", date "2022-08-30 16:43:05", and comment "Ritik".

- That HTML code is thus now in the application's web server, which gets rendered every time the victim visits this malicious page, he'll always have this code which looks official to him.

The screenshot shows the OWASP Mutillidae II: Web Pwn in Mass Production interface after the blog entry has been saved. The "Add New Blog Entry" form is now empty. The "View Blogs" link is still present. At the bottom, a table titled "6 Current Blog Entries" shows the same entry as before, but the comment field now displays the rendered HTML code: `<td/> Ritik <marquee> こんにちは元気ですか </marquee>`.

Steps for the experiment: XMS injection

- Open the website:
[Google XSS Game Website](#)

Now, we'll be redirected to the web page which is suffering from an **HTML Injection vulnerability** which allows the user to submit his entry in the blog as shown in the screenshot.

[1/6] Level 1: Hello, world of XSS

Mission Description

This level demonstrates a common cause of cross-site scripting where user input is directly included in the page without proper escaping.

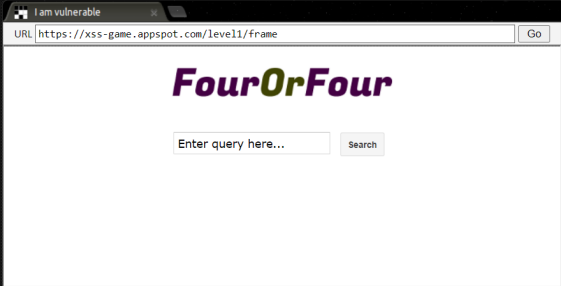
Interact with the vulnerable application window below and find a way to make it execute JavaScript of your choosing. You can take actions inside the vulnerable window or directly edit its URL bar.

Mission Objective

Inject a script to pop up a JavaScript alert() in the frame below.

Once you show the alert you will be able to advance to the next level.

Your Target



2. If the search field is vulnerable when the user enters any script, then it will be executed. Consider, a user entering a very simple script as shown below:

<script>alert("こんにちは元気ですか")</script>

[1/6] Level 1: Hello, world of XSS

Mission Description

This level demonstrates a common cause of cross-site scripting where user input is directly included in the page without proper escaping.


Interact with the vulnerable application window below and find a way to make it execute JavaScript of your choosing. You can take actions inside the vulnerable window or directly edit its URL bar.

Mission Objective

Inject a script to pop up a JavaScript alert() in the frame below.

Once you show the alert you will be able to advance to the next level.

Your Target



3. Then after clicking on the "Search" button, the entered script will be executed. The script typed into the search field gets executed. This just shows the vulnerability of the XSS attack.



We learn what is HTML injection and XSS injection. An overview of how these attacks are constructed and applied to real systems. If the app or website lacks proper data sanitization, the malicious link executes the attacker's chosen code on the user's system. As a result, **the attacker can steal the user's active session cookie** which can be harmful to the website.

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Worksheet completion including writing learning objectives/Outcomes. (To be submitted at the end of the day).		
2.	Post-Lab Quiz Result.		
3.	Student Engagement in Simulation/Demonstration/Performance and Controls/Pre-Lab Questions.		
	Signature of Faculty (with Date):	Total Marks Obtained:	