REFLECTED XSS



Submitted by:

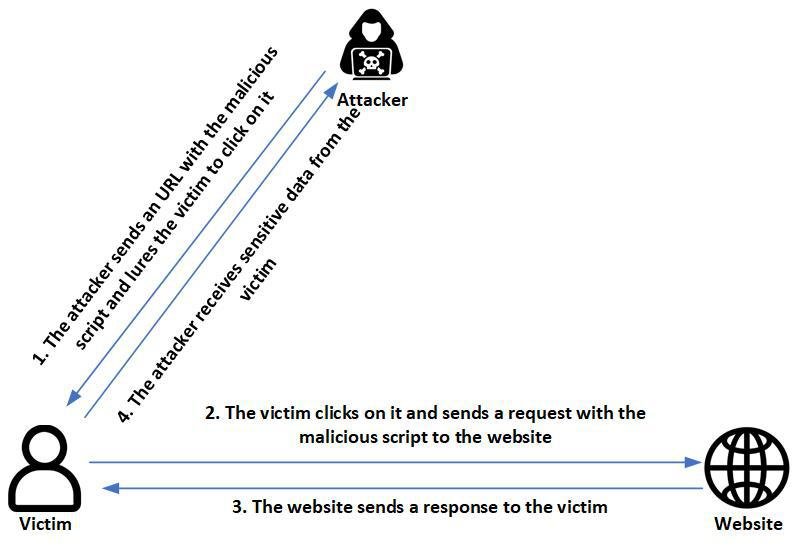
Muhammad Sameer

Reflected XSS

Reflected Cross-Site Scripting (XSS) is a type of security vulnerability commonly found in web applications. It occurs when an attacker injects malicious scripts into a web application, which then reflects these scripts back to the user's browser. This can lead to various security issues, including data theft, session hijacking, and unauthorized actions.

# Working

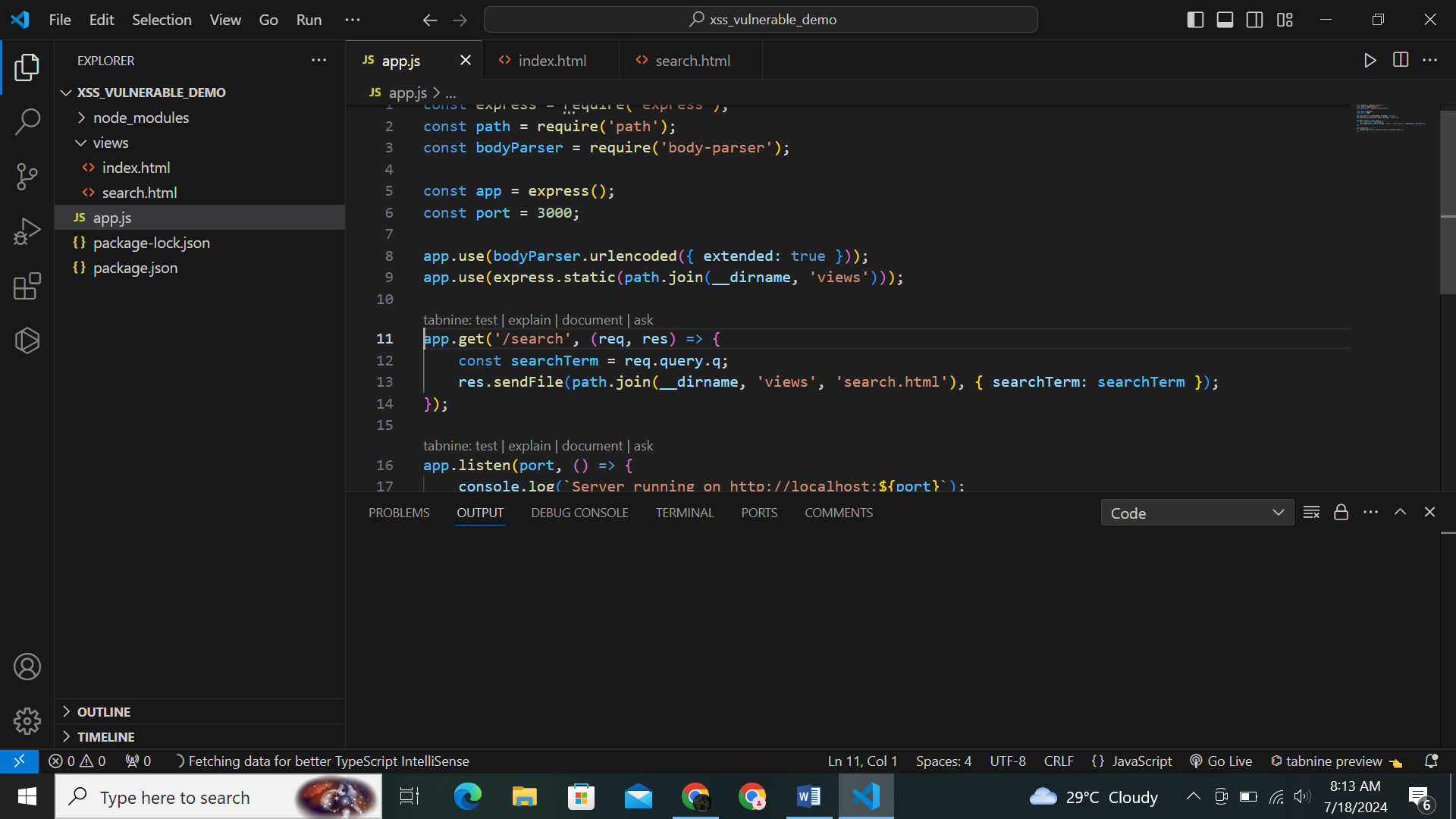
1. An attacker sends a malicious link to a victim.
2. The victim clicks on the link, which sends a request to the server with the malicious script.
3. The server reflects the script back in its response.
4. The victim’s browser executes the script, leading to potential data theft or other malicious activities.



# Example

* Attacker crafts a phising email.
* In the email attacker attaches the link of crafted email i.e (http://example.com/search?q=<script>alert('XSS')</script>).
* The attacker sends the email to user.
* Victim opens the email and clicks on the link.
* When link is opened it run the attacker script and sends the information to attacker.

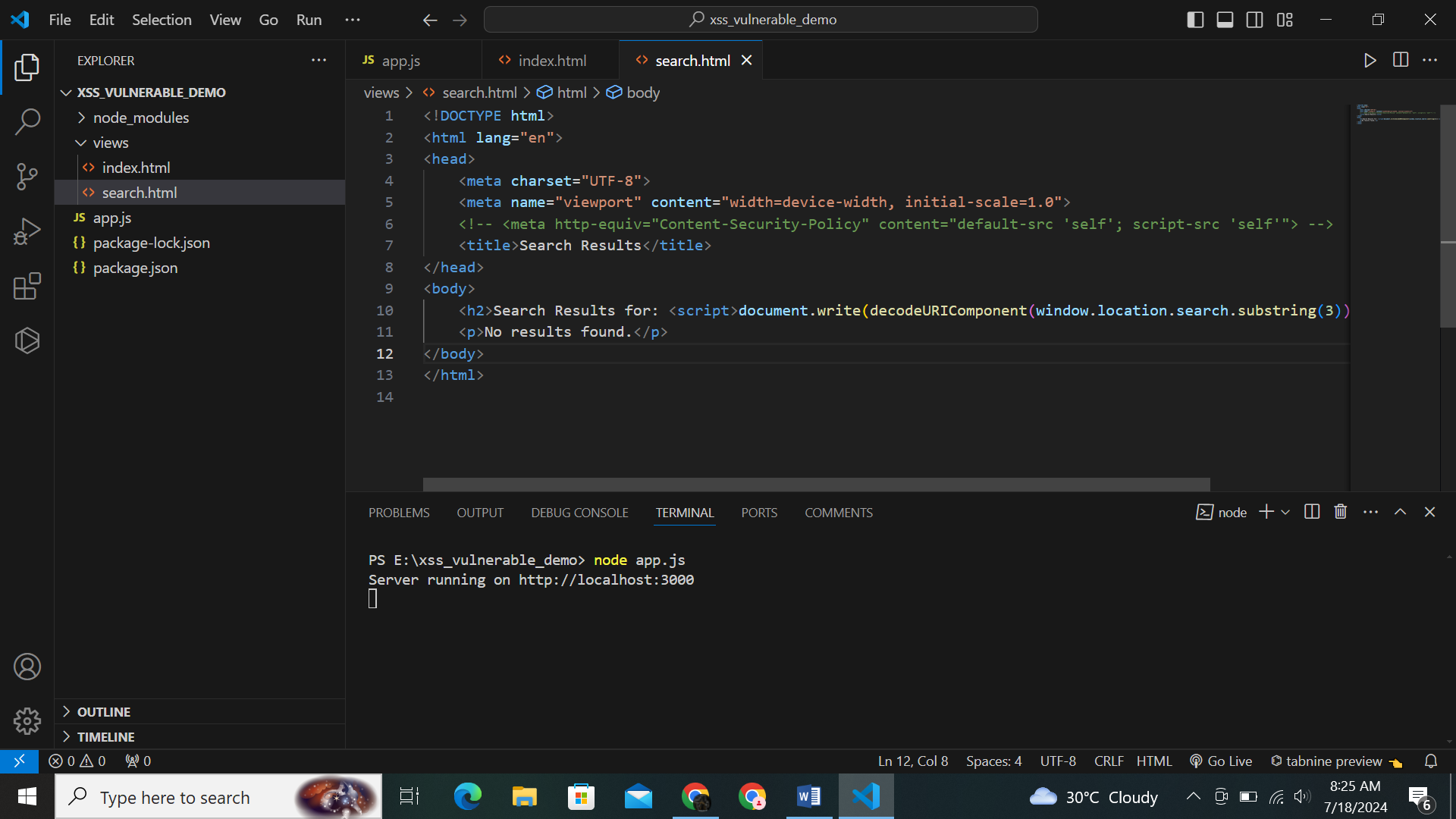
# How the Script Works in the Search Case



## Route Handling

* When a GET request is made to the /search route, the searchTerm is extracted from the query parameter q.
* Example URLs: http://localhost:3000/search?q=hello
* The searchTerm variable stores the value requested from the form.

## Client-Side Script in search.html

****

decodes and writes the query parameter value directly to the HTML.

## Vulnerability

If the query parameter contains a script, it will be executed by the browser. For example, the URL **http://localhost:3000/search?q=<script>alert('XSS')</script> will execute an alert with 'XSS'.**

# Prevention Techniques

## Input Validation

Validate and sanitize all user inputs to ensure they do not contain malicious code.

**const safeSearchTerm = searchTerm.replace(/[<>]/g, function(c) { return {'<':'&lt;', '>':'&gt;'}[c]; });**

## Output Encoding

Encode outputs to prevent scripts from being interpreted as executable code.

**Replaces < and > characters to prevent script execution.**

## Content Security Policy (CSP)

Implement CSP to restrict the sources from which scripts can be loaded.

**<meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self'">**