XSS Attack & Defense Lab: Identifying, Exploiting, and Mitigating Cross-Site Scripting Vulnerabilities

Presented By
Lamiya
Laboni Akter Mitu

Presented TO
MD. Nessar Rahman Porag

Objectives

- To understand the fundamentals of Cross-Site Scripting (XSS) vulnerabilities.
- To identify XSS weaknesses in a vulnerable web application.
- To practically exploit Stored, Reflected, and DOM-Based XSS using DVWA.
- To analyze HTTP requests and responses through Burp Suite.
- To apply effective security mechanisms to mitigate XSS attacks.

Lab Environment & Tools

Attacker Machine: Kali Linux

Target Server: Metasploitable2

Vulnerable Application: Damn Vulnerable Web Application (DVWA)

Tools Used: Firefox Browser, Burp Suite, Apache Server, ModSecurity WAF

What is Cross-Site Scripting (XSS)?

- Cross-Site Scripting (XSS) is a common web application security vulnerability
- It allows attackers to inject malicious JavaScript code into a web application
- The injected script executes in the victim's browser
- This can lead to cookie theft, session hijacking, and unauthorized actions

The Attack Surface: Three Primary Types

Understanding the delivery method is key to identifying where your application is most vulnerable.

Reflected XSS

- Script is reflected in server response
- Executes when victim clicks a malicious link

Stored (Persistent) XSS

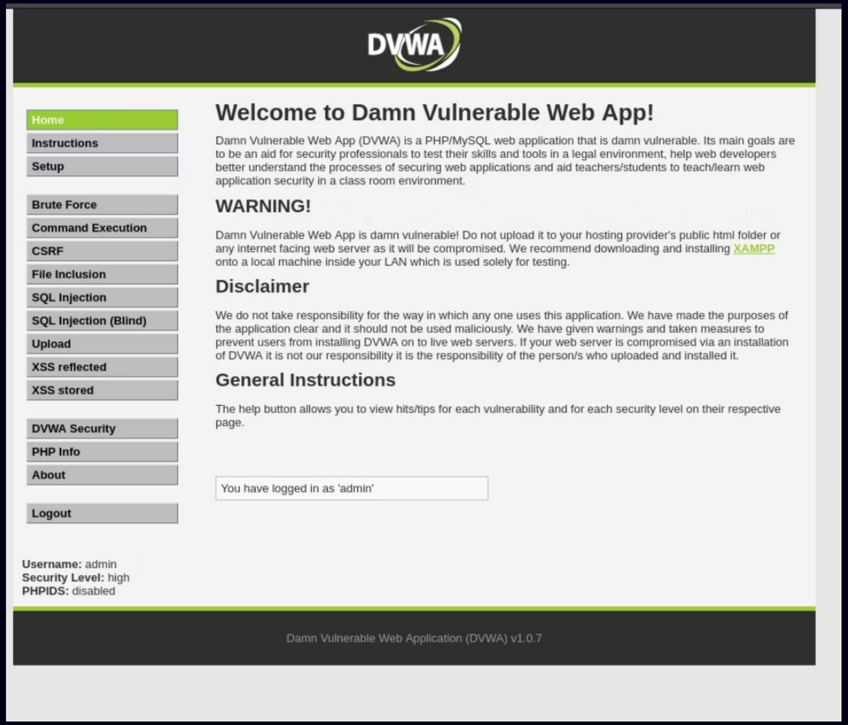
- Malicious script is stored in the database
- Executes automatically when page loads
- Affects multiple users

DOM-Based XSS

- Occurs entirely on the client side
- Vulnerability exists in JavaScript handling of user input
- Server-side processing is not involved

DVWA Overview

Damn Vulnerable Web Application (DVWA) is a purposely insecure web application.It is designed for learning and practicing web security attacks. The security level was set to **Low** to demonstrate XSS vulnerabilities



Welcome to Damn Vulnerable Web App!

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

WARNING!

Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing XAMPP onto a local machine inside your LAN which is used solely for testing.

Disclaimer

We do not take responsibility for the way in which any one uses this application. We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

General Instructions

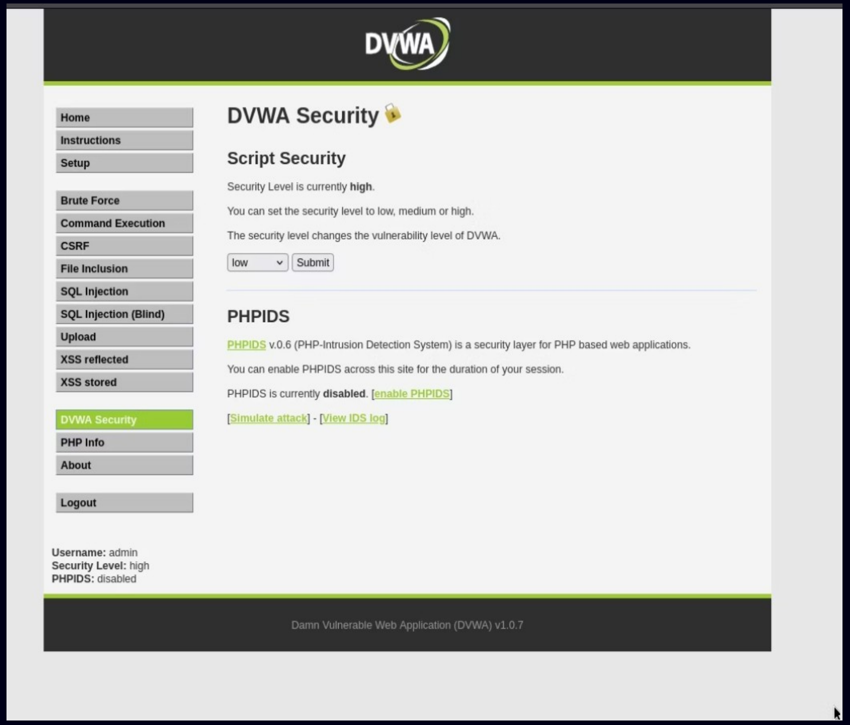
The help button allows you to view hits/tips for each vulnerability and for each security level on their respective page.

You have logged in as 'admin'

Username: admin
Security Level: high
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

Fig -1 : Home page



DVWA Security

Script Security

Security Level is currently **high**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

low Submit

PHPIDS

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently **disabled**. [enable PHPIDS]

[Simulate attack] - [View IDS log]

Username: admin
Security Level: high
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

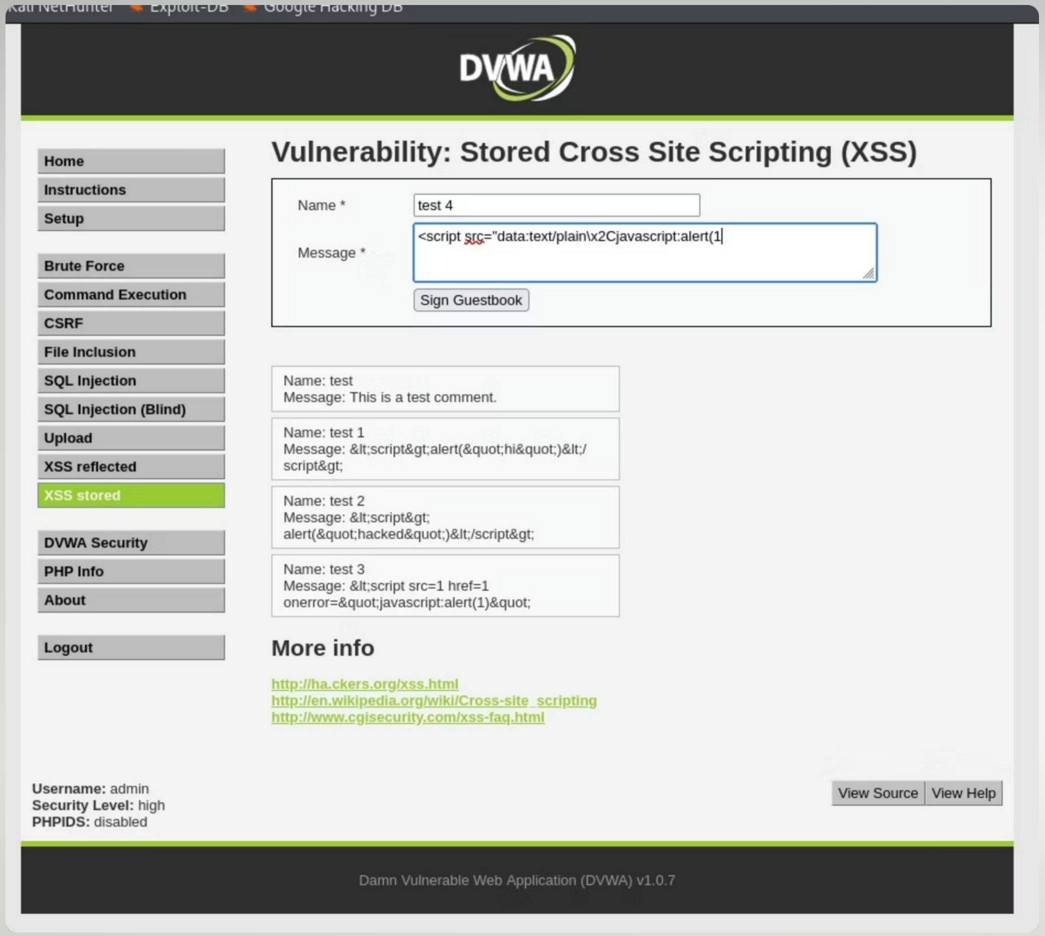
Fig-2 : Security page

XSS Vulnerability Identification

- Application input fields were tested using JavaScript payloads.
- User input was reflected without proper validation or sanitization
- Multiple XSS vulnerabilities were successfully identified

Stored XSS Exploitation

- A malicious JavaScript payload was submitted through an input form
- The payload was stored in the application database
- The script executed automatically whenever the page was loaded



Vulnerability: Stored Cross Site Scripting (XSS)

Name * test 4

Message * `<script src="data:text/plain\x2Cjavascript:alert(1`

Name: test
Message: This is a test comment.

Name: test 1
Message: &lt;script&gt;alert(&quot;hi&quot;)&lt;/script>

Name: test 2
Message: &lt;script&gt;alert(&quot;hacked&quot;)&lt;/script>

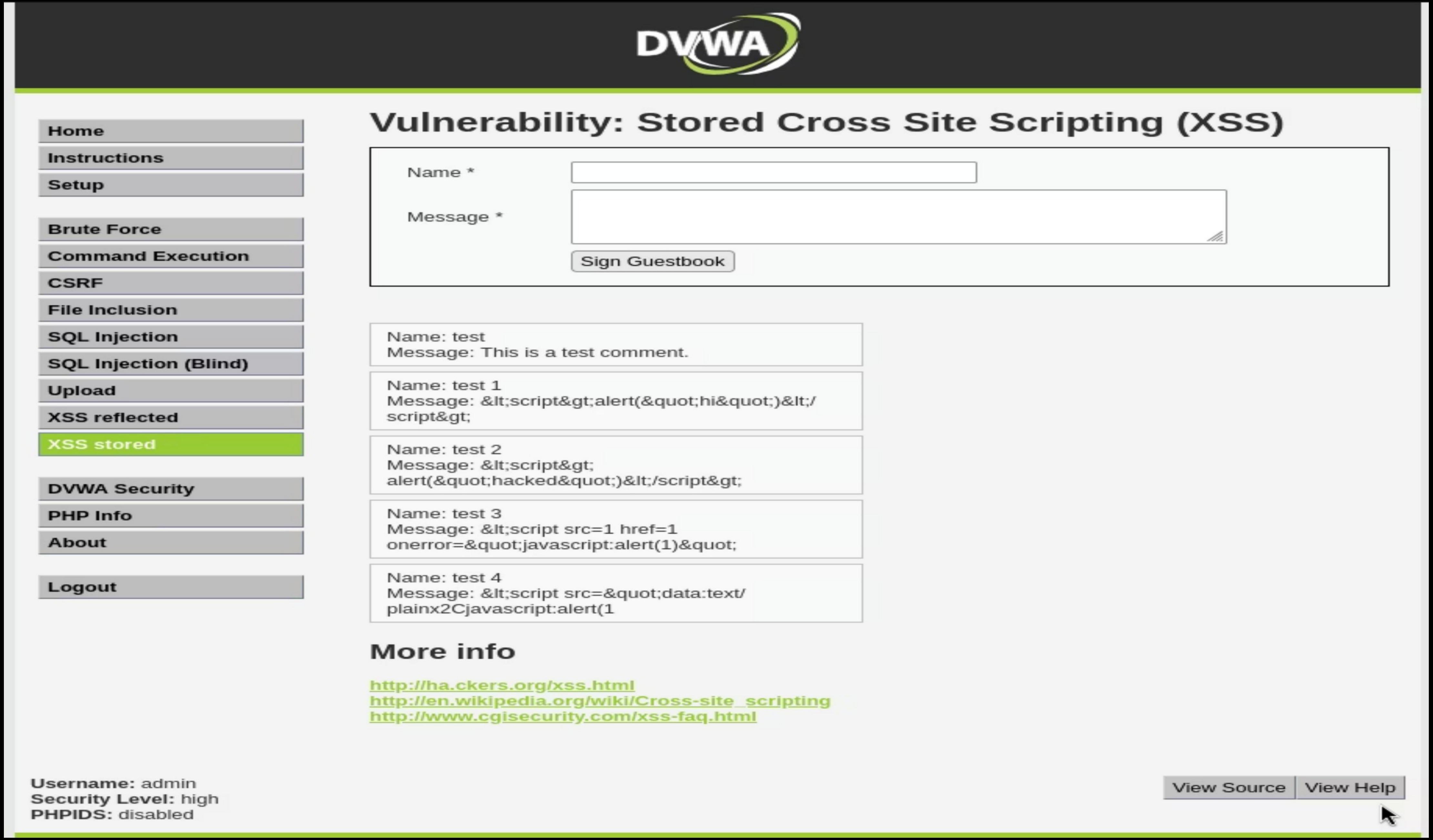
Name: test 3
Message: &lt;script src=1 href=1 onerror=&quot;javascript:alert(1)&quot;

More info

http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html

Username: admin
Security Level: high
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Name: test 1
Message: &lt;script&gt;alert(&quot;hi&quot;)&lt;/script&gt;

Name: test 2
Message: &lt;script&gt;alert(&quot;hacked&quot;)&lt;/script&gt;

Name: test 3
Message: &lt;script src=1 href=1 onerror=&quot;javascript:alert(1)&quot;

Name: test 4
Message: &lt;script src=&quot;data:text/plainx2Cjavascript:alert(1

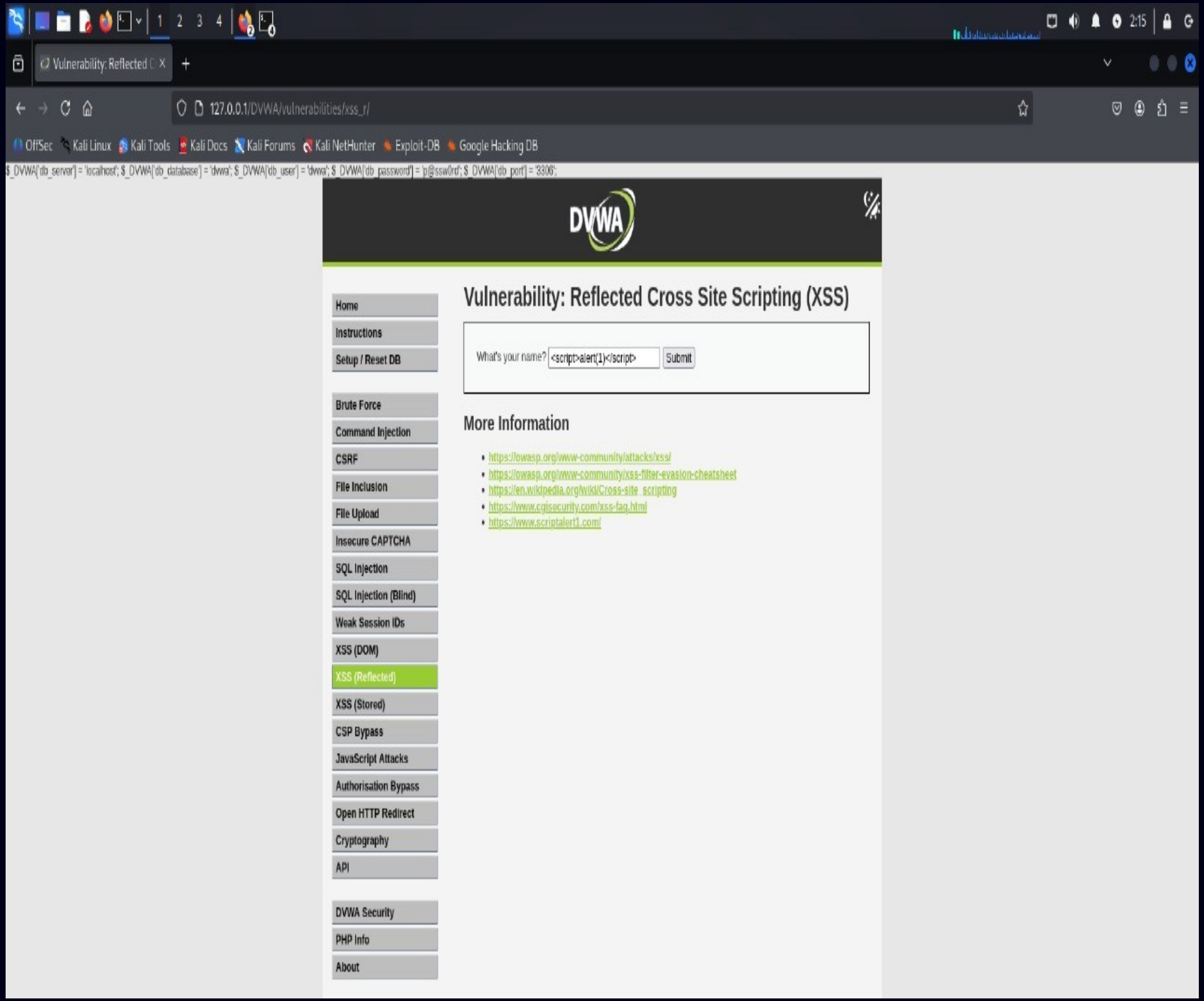
More info

http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html

Username: admin
Security Level: high
PHPIDS: disabled

Reflected XSS Exploitation

- A crafted payload was sent through an HTTP request
- The server reflected the input in its response
- JavaScript executed immediately in the browser



```
$_DVWA['db_server'] = 'localhost'; $_DVWA['db_database'] = 'dvwa'; $_DVWA['db_user'] = 'dvwa'; $_DVWA['db_password'] = 'p@ssw0rd'; $_DVWA['db_port'] = '3306';
```

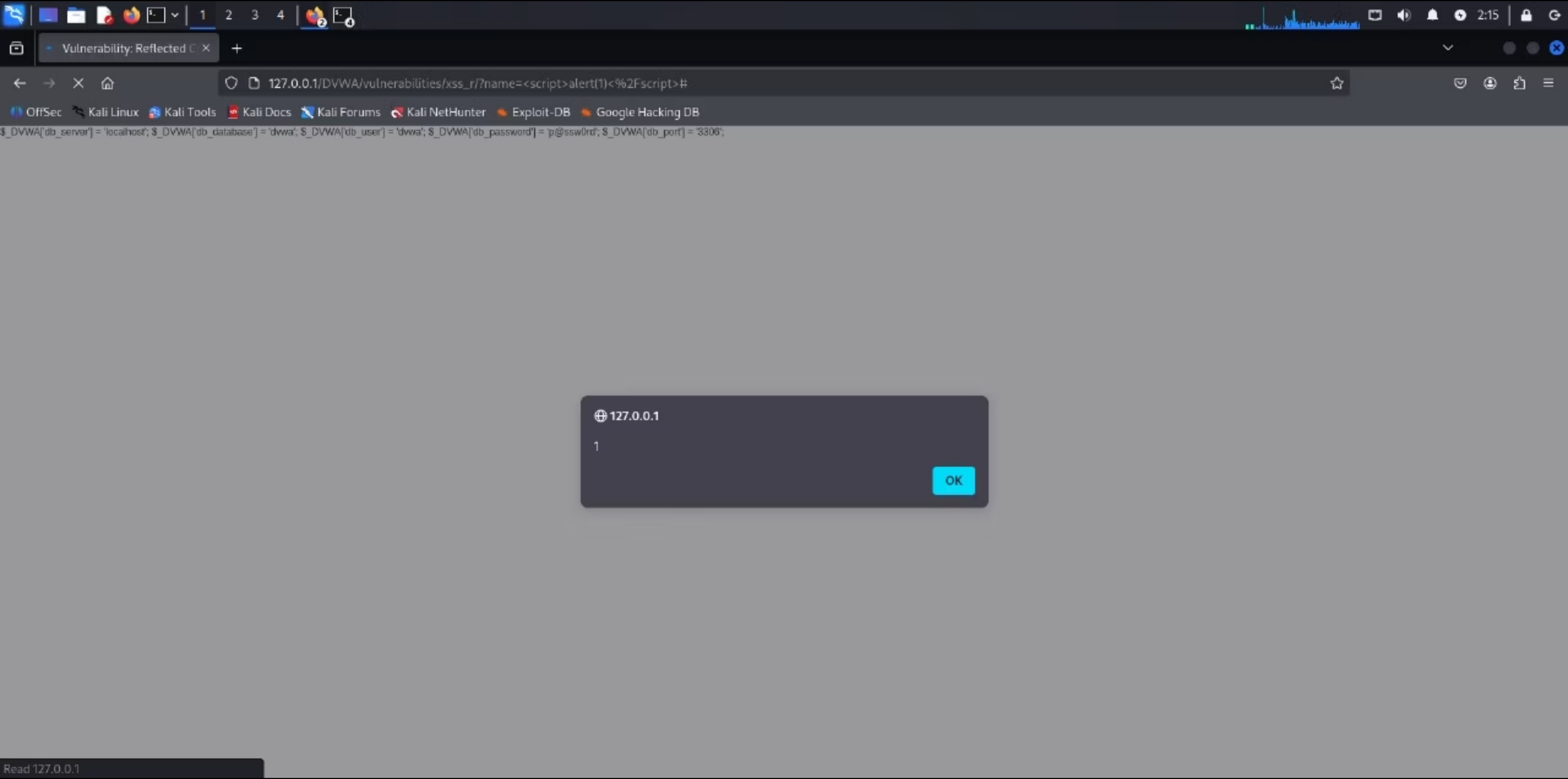
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? `<script>alert(1)</script>` Submit

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/

Made with GAMMA



```
127.0.0.1/DVWA/vulnerabilities/xss_r/?name=<script>alert(1)<%2Fscript>#
```

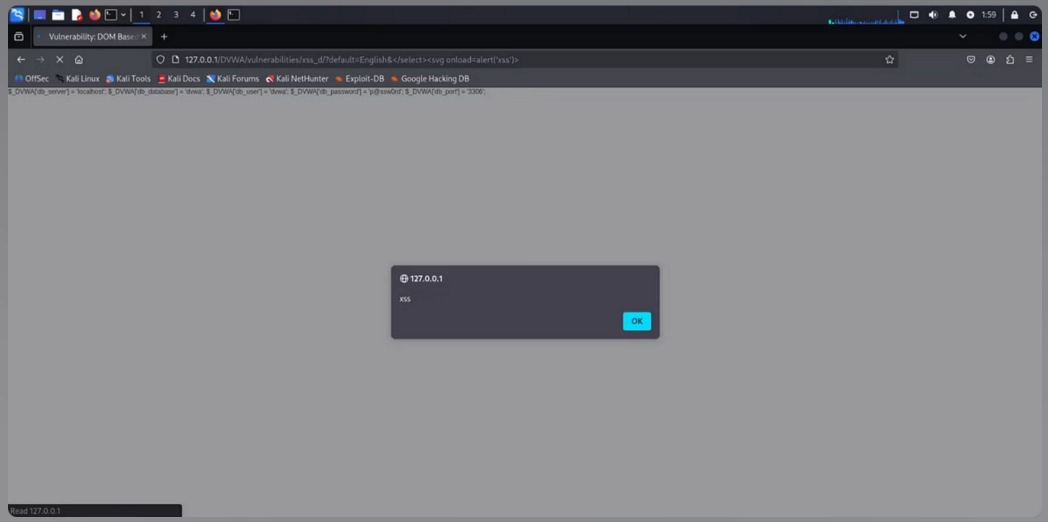
```
$_DVWA['db_server'] = 'localhost'; $_DVWA['db_database'] = 'dvwa'; $_DVWA['db_user'] = 'dvwa'; $_DVWA['db_password'] = 'p@ssw0rd'; $_DVWA['db_port'] = '3306';
```

127.0.0.1

1

DOM-Based XSS Exploitation

- Payloads were injected through URL parameters
- Client-side JavaScript processed the malicious input
- Script execution occurred without server involvement



```
127.0.0.1/DVWA/vulnerabilities/xss_d/?default=English&</select><svg onload=alert('xss')>
```

```
$_DVWA['db_server'] = 'localhost'; $_DVWA['db_database'] = 'dvwa'; $_DVWA['db_user'] = 'dvwa'; $_DVWA['db_password'] = 'p@ssw0rd'; $_DVWA['db_port'] = '3306';
```

127.0.0.1

xss

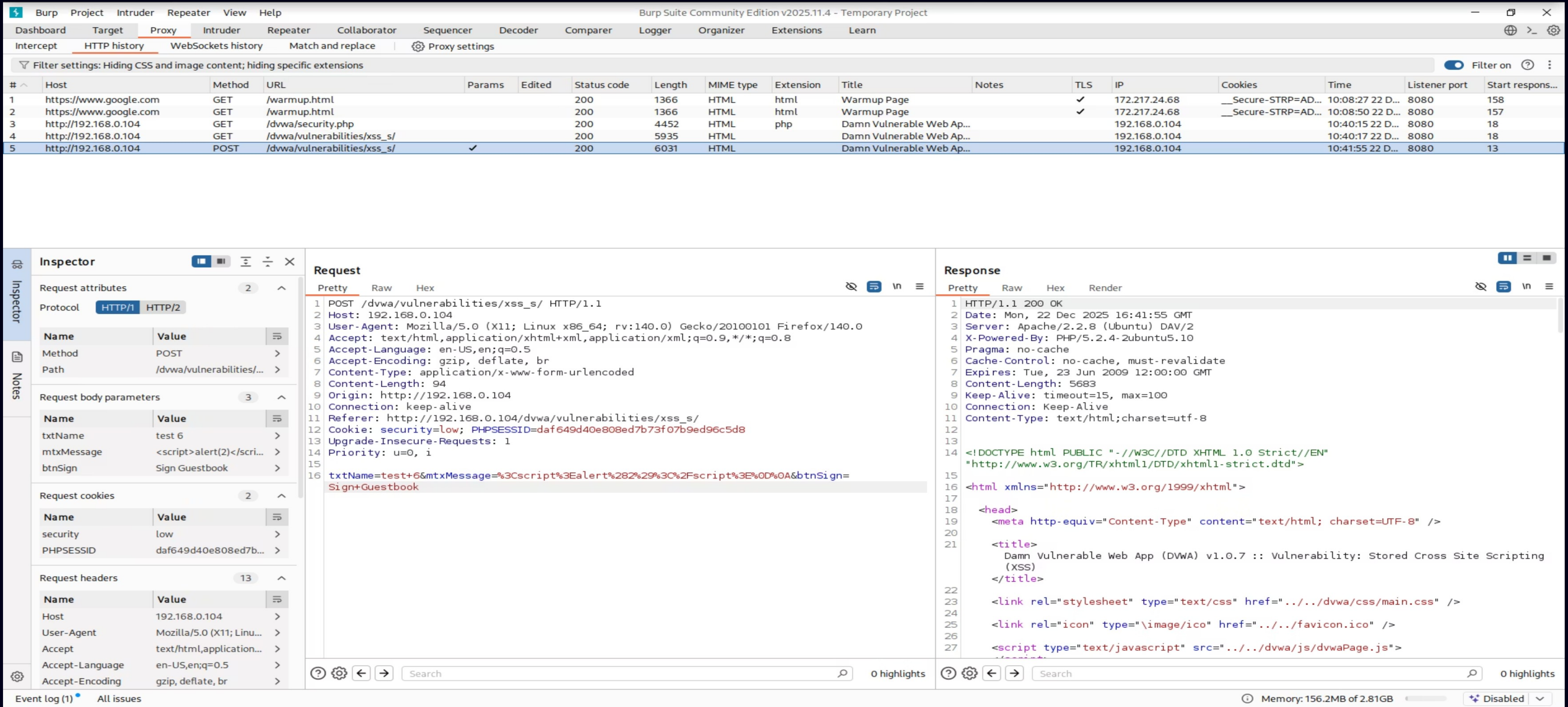
Request & Response Analysis Using Burp Suite

- Burp Suite Proxy was enabled to intercept HTTP traffic
- Requests containing malicious payloads were captured
- Server responses confirmed payload reflection or storag

Importance of Burp Suite

- Provides visibility into HTTP request and response flow
- Helps trace how malicious payloads travel through the application
- Acts as strong evidence of XSS vulnerabilities

XSS Stored



Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.11.4 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Match and replace Proxy settings
Filter on

Filter settings: Hiding CSS and image content; hiding specific extensions

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port	Start respons...
1	https://www.google.com	GET	/warmup.html			200	1366	HTML	html	Warmup Page		✓	172.217.24.68	__Secure-STRP=AD...	10:08:27 22 D...	8080	158
2	https://www.google.com	GET	/warmup.html			200	1366	HTML	html	Warmup Page		✓	172.217.24.68	__Secure-STRP=AD...	10:08:50 22 D...	8080	157
3	http://192.168.0.104	GET	/dvwa/security.php			200	4452	HTML	php	Damn Vulnerable Web Ap...			192.168.0.104		10:40:15 22 D...	8080	18
4	http://192.168.0.104	GET	/dvwa/vulnerabilities/xss_s/			200	5935	HTML		Damn Vulnerable Web Ap...			192.168.0.104		10:40:17 22 D...	8080	18
5	http://192.168.0.104	POST	/dvwa/vulnerabilities/xss_s/	✓		200	6031	HTML		Damn Vulnerable Web Ap...			192.168.0.104		10:41:55 22 D...	8080	13

Request attributes 2

Protocol HTTP/1 HTTP/2

Name	Value
Method	POST
Path	/dvwa/vulnerabilities/...

Request body parameters 3

Name	Value
txtName	test 6
mtxMessage	<script>alert(2)</scri...
btnSign	Sign Guestbook

Request cookies 2

Name	Value
security	low
PHPSESSID	daf649d40e808ed7b...

Request headers 13

Name	Value
Host	192.168.0.104
User-Agent	Mozilla/5.0 (X11; Linu...
Accept	text/html,application...
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate, br

Request

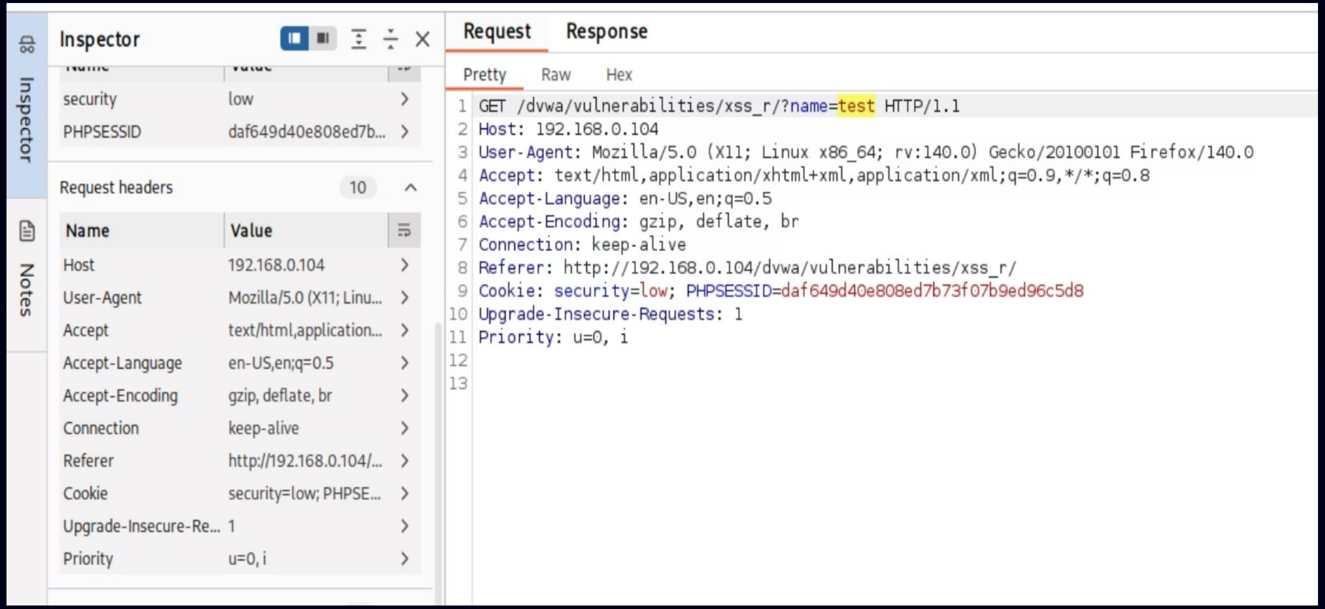
```
POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1
Host: 192.168.0.104
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 94
Origin: http://192.168.0.104
Connection: keep-alive
Referer: http://192.168.0.104/dvwa/vulnerabilities/xss_s/
Cookie: security=low; PHPSESSID=daf649d40e808ed7b73f07b9ed96c5d8
Upgrade-Insecure-Requests: 1
Priority: u=0, i

txtName=test+6&mtxMessage=%3Cscript%3Ealert%282%29%3C%2Fscript%3E%0D%0A&btnSign=
Sign+Guestbook
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 22 Dec 2025 16:41:55 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Pragma: no-cache
Cache-Control: no-cache, must-revalidate
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Content-Length: 5683
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=utf-8


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

    <title>
      Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: Stored Cross Site Scripting
      (XSS)
    </title>

    <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />

    <link rel="icon" type="\image/ico" href="../../favicon.ico" />

    <script type="text/javascript" src="../../dvwa/js/dvwaPage.js">
```

Made with GAMMA

XSS Reflected



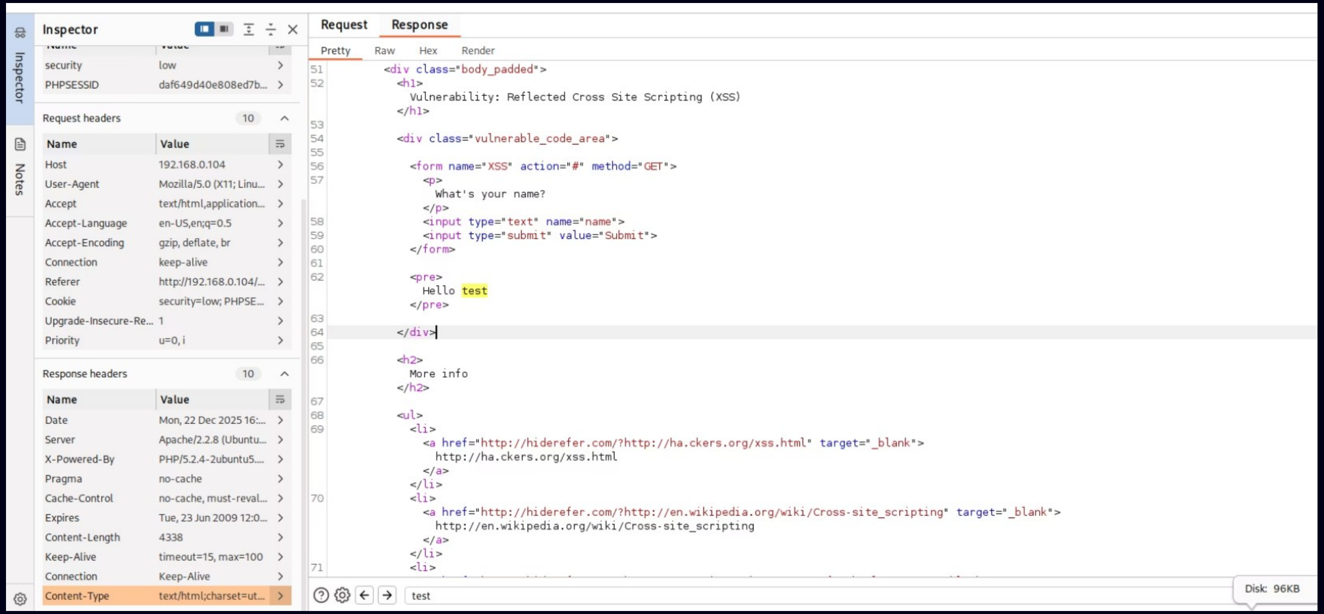
security	low
PHPSESSID	daf649d40e808ed7b...

Request headers 10

Name	Value
Host	192.168.0.104
User-Agent	Mozilla/5.0 (X11; Linu...
Accept	text/html,application...
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Referer	http://192.168.0.104/...
Cookie	security=low; PHPSE...
Upgrade-Insecure-Re...	1
Priority	u=0, i

```
GET /dvwa/vulnerabilities/xss_r/?name=test HTTP/1.1
Host: 192.168.0.104
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://192.168.0.104/dvwa/vulnerabilities/xss_r/
Cookie: security=low; PHPSESSID=daf649d40e808ed7b73f07b9ed96c5d8
Upgrade-Insecure-Requests: 1
Priority: u=0, i


```

Fig-1: Request



security	low
PHPSESSID	daf649d40e808ed7b...

Request headers 10

Name	Value
Host	192.168.0.104
User-Agent	Mozilla/5.0 (X11; Linu...
Accept	text/html,application...
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Referer	http://192.168.0.104/...
Cookie	security=low; PHPSE...
Upgrade-Insecure-Re...	1
Priority	u=0, i

Response headers 10

Name	Value
Date	Mon, 22 Dec 2025 16:...
Server	Apache/2.2.8 (Ubuntu...
X-Powered-By	PHP/5.2.4-2ubuntu5....
Pragma	no-cache
Cache-Control	no-cache, must-reval...
Expires	Tue, 23 Jun 2009 12:0...
Content-Length	4338
Keep-Alive	timeout=15, max=100
Connection	Keep-Alive
Content-Type	text/html;charset=ut...

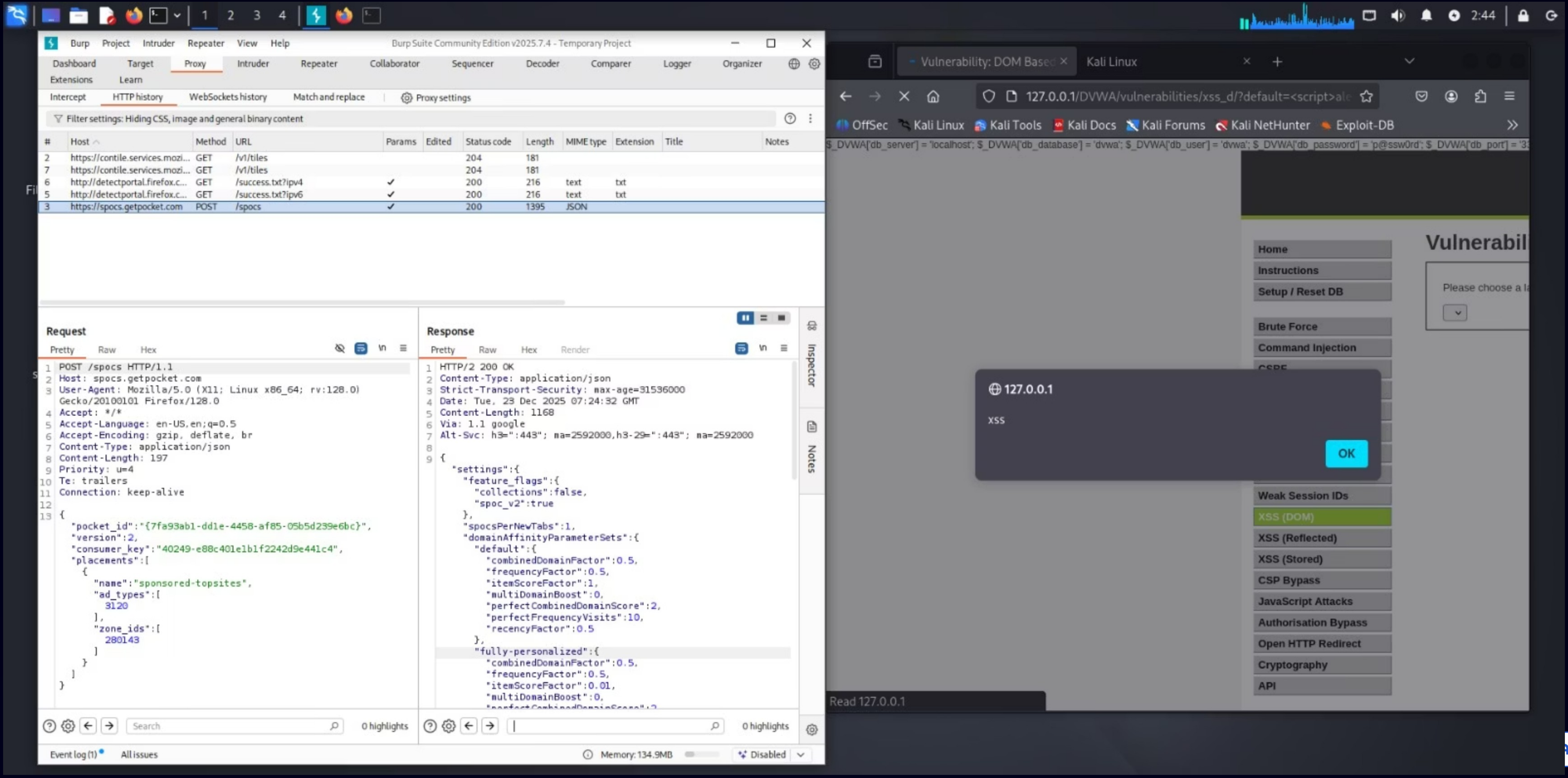
```
    <div class="body_padded">
      <h1>
        Vulnerability: Reflected Cross Site Scripting (XSS)
      </h1>

      <div class="vulnerable_code_area">

        <form name="XSS" action="#" method="GET">
          <p>
            What's your name?
          </p>
          <input type="text" name="name">
          <input type="submit" value="Submit">
        </form>

        <pre>
          Hello test
        </pre>

      </div>

      <h2>
        More info
      </h2>

      <ul>
        <li>
          <a href="http://hiderefer.com/?http://ha.ckers.org/xss.html" target="_blank">
            http://ha.ckers.org/xss.html
          </a>
        </li>
        <li>
          <a href="http://hiderefer.com/?http://en.wikipedia.org/wiki/Cross-site_scripting" target="_blank">
            http://en.wikipedia.org/wiki/Cross-site_scripting
          </a>
        </li>
        <li>
```

test

Fig-2: Respons

Made with GAMMA

XSS Dom



Filter settings: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes
2	https://contile.services.mozi...	GET	/v1/tiles			204	181				
7	https://contile.services.mozi...	GET	/v1/tiles			204	181				
6	http://detectportal.firefox.c...	GET	/success.txt?ipv4	✓		200	216	text	txt		
5	http://detectportal.firefox.c...	GET	/success.txt?ipv6	✓		200	216	text	txt		
3	https://spocs.getpocket.com	POST	/spocs	✓		200	1395	JSON			

```
POST /spocs HTTP/1.1
Host: spocs.getpocket.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0)
Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 197
Priority: u=4
Te: trailers
Connection: keep-alive

{
  "pocket_id":"{7fa93ab1-dd1e-4458-af85-05b5d239e6bc}",
  "version":2,
  "consumer_key":"40249-e88c401e1b1f2242d9e441c4",
  "placements":[
    {
      "name":"sponsored-topsites",
      "ad_types":[
        3120
      ],
      "zone_ids":[
        280143
      ]
    }
  ]
}
```

```
HTTP/2 200 OK
Content-Type: application/json
Strict-Transport-Security: max-age=31536000
Date: Tue, 23 Dec 2025 07:24:32 GMT
Content-Length: 1168
Via: 1.1 google
Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000

{
  "settings":{
    "feature_flags":{
      "collections":false,
      "spoc_v2":true
    },
    "spocsPerNewTabs":1,
    "domainAffinityParameterSets":{
      "default":{
        "combinedDomainFactor":0.5,
        "frequencyFactor":0.5,
        "itemScoreFactor":1,
        "multiDomainBoost":0,
        "perfectCombinedDomainScore":2,
        "perfectFrequencyVisits":10,
        "recencyFactor":0.5
      },
      "fully-personalized":{
        "combinedDomainFactor":0.5,
        "frequencyFactor":0.5,
        "itemScoreFactor":0.01,
        "multiDomainBoost":0,
```

127.0.0.1/DVWA/vulnerabilities/xss_d/?default=<script>ale

$_DVWA['db_server'] = 'localhost'; $_DVWA['db_database'] = 'dvwa'; $_DVWA['db_user'] = 'dvwa'; $_DVWA['db_password'] = 'p@ssw0rd'; $_DVWA['db_port'] = '3

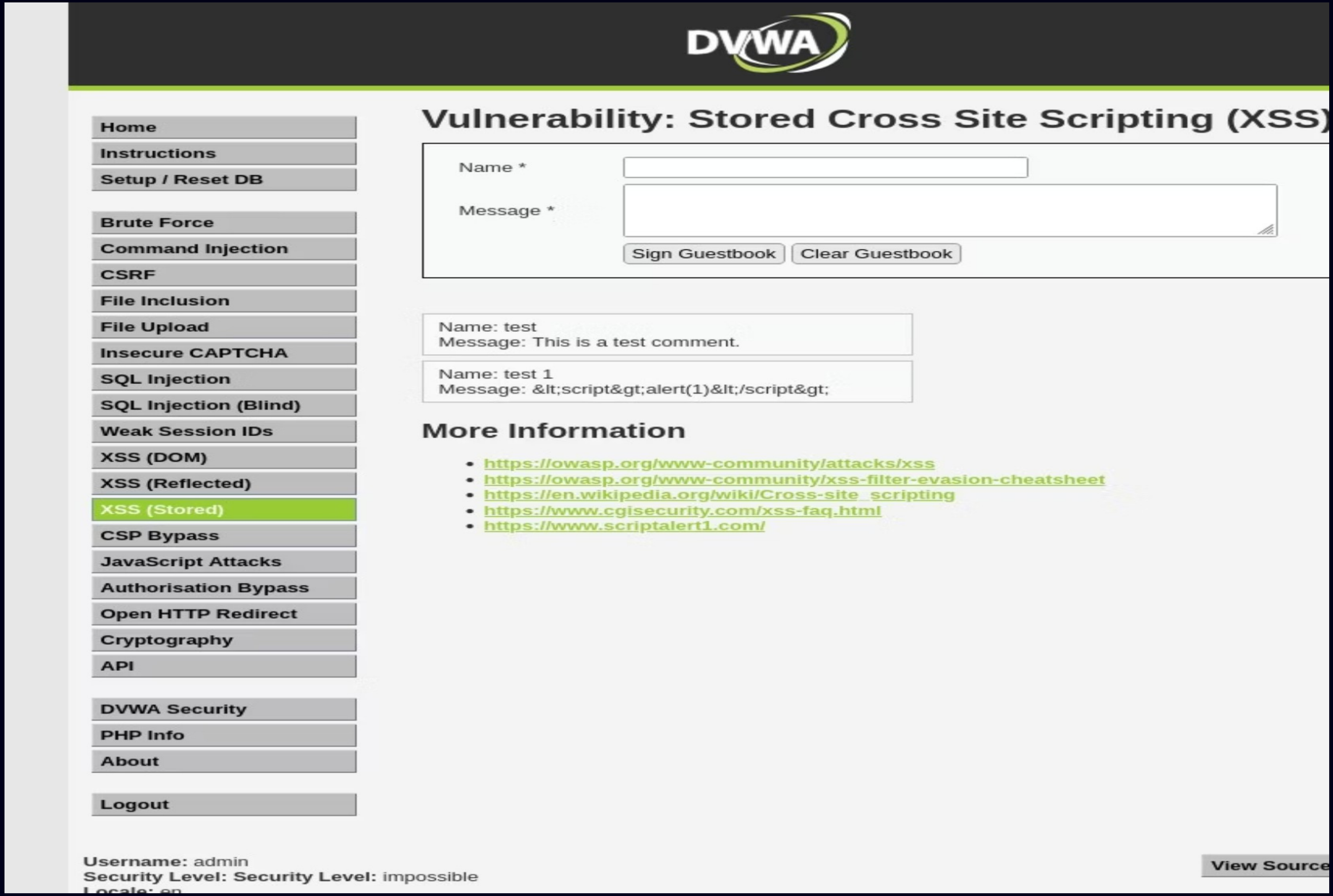
127.0.0.1

xss

Defense Phase Overview

- Defense mechanisms were applied after successful exploitation
- The goal was to prevent JavaScript execution
- Multiple security layers were implemented

Stored xss



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

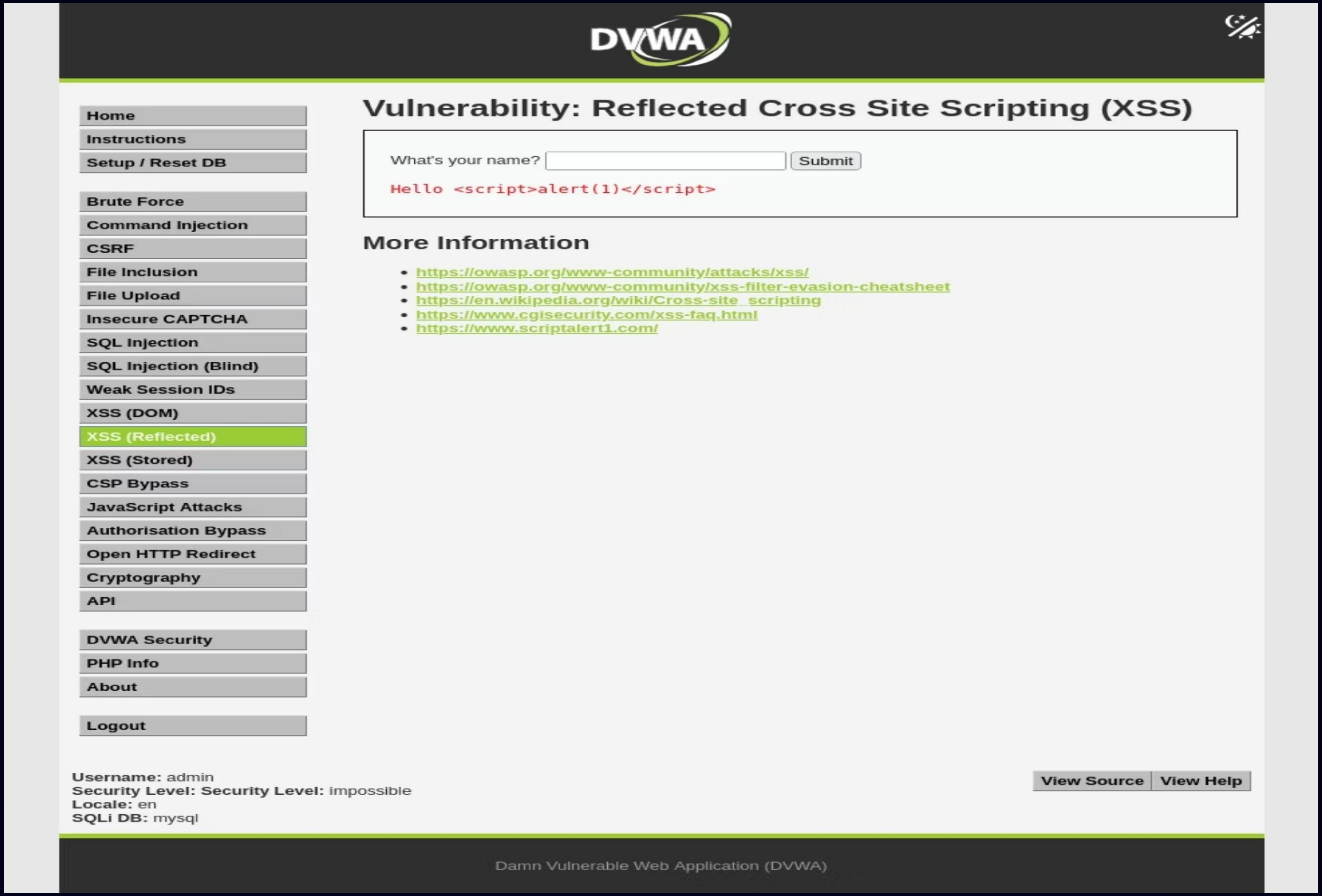
Name: test 1
Message: <script>alert(1)</script>

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/

Username: admin
Security Level: Security Level: impossible
Locale: en

Reflected XSS



Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

```
Hello <script>alert(1)</script>
```

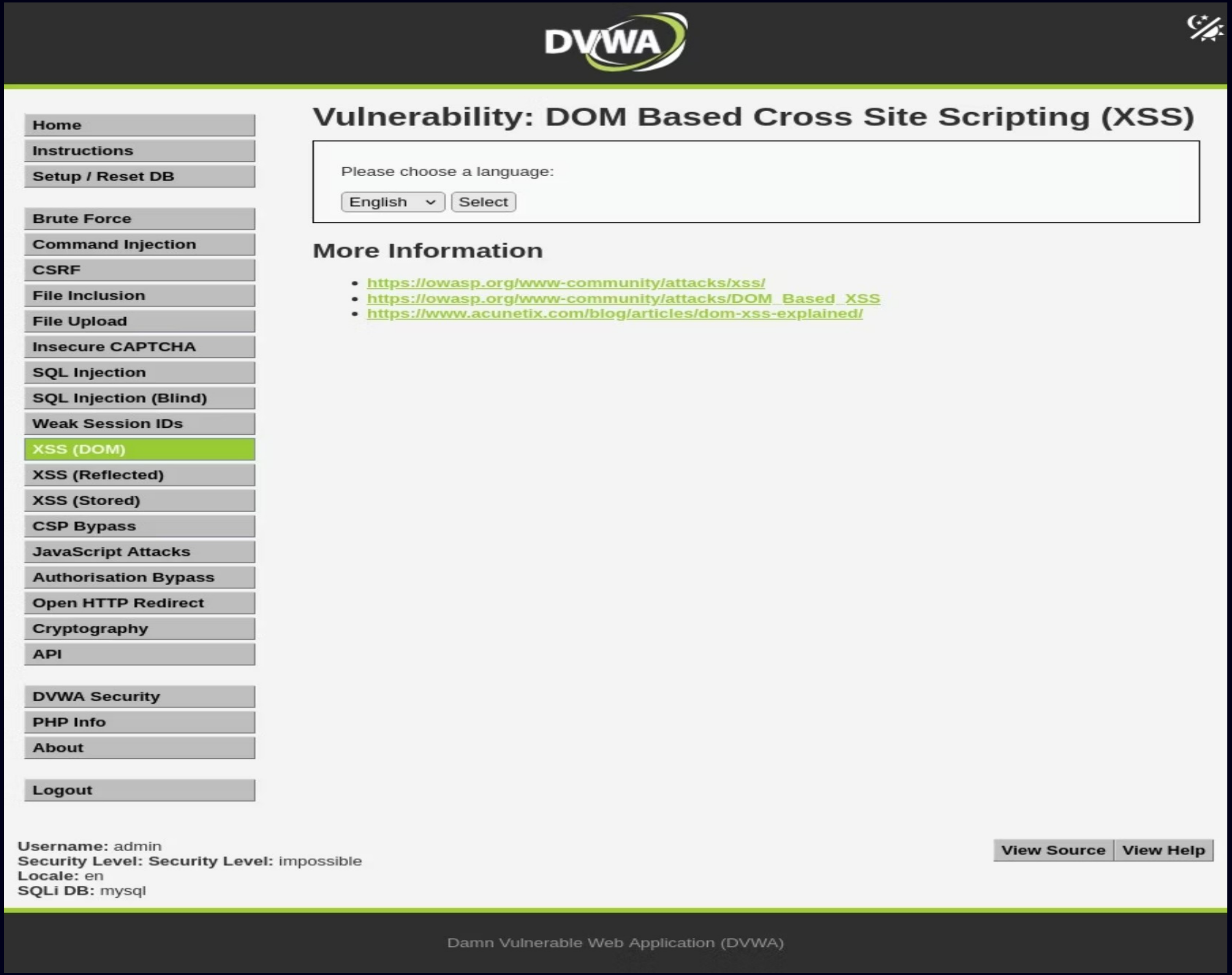
More Information

- 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/

Username: admin
Security Level: Security Level: impossible
Locale: en
SQLi DB: mysql

Damn Vulnerable Web Application (DVWA)

DOM-Based XSS



Vulnerability: DOM Based Cross Site Scripting (XSS)

Please choose a language:

English 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/

Username: admin
Security Level: Security Level: impossible
Locale: en
SQLi DB: mysql

Damn Vulnerable Web Application (DVWA)

Defense & Mitigation Strategies

Input Validation

User inputs were validated before processing. Suspicious script tags were filtered. Reduced the risk of malicious injection.

Output Encoding

Special characters were converted into HTML entities. Scripts were displayed as plain text. Browser execution was successfully prevented

ModSecurity Web Application Firewall (WAF)

ModSecurity was integrated with the Apache web server. OWASP Core Rule Set (CRS) was enabled. Incoming requests were inspected for attack patterns

ModSecurity Configuration & Log Analysis

- ModSecurity engine was enabled successfully
- Suspicious XSS payloads were detected
- OWASP CRS rules were triggered
- Audit logs were generated for analysis

Result Analysis

Stored XSS

- No alert box was executed
- Payload was automatically HTML-encoded
- Stored XSS vulnerability was mitigated

Reflected XSS

- JavaScript execution was blocked
- Payload appeared as harmless text
- WAF successfully detected malicious patterns

DOM-Based XSS

- No JavaScript execution occurred
- Web page loaded normally
- Client-side protection proved effective

Overall Defense Evaluation

The overall defense implementation in this project proved to be effective in mitigating Cross-Site Scripting (XSS) vulnerabilities. After enabling input validation, output encoding, and deploying ModSecurity Web Application Firewall with the OWASP Core Rule Set, all previously exploited XSS payloads failed to execute. The browser treated injected scripts as harmless text, preventing unauthorized JavaScript execution and protecting user sessions.

Additionally, ModSecurity actively inspected incoming HTTP requests and successfully detected suspicious patterns related to XSS attacks. Relevant security rules were triggered, and detailed audit logs were generated, confirming that the Web Application Firewall was functioning as intended. This layered defense approach significantly reduced the attack surface of the application and demonstrated how combining secure coding practices with automated security controls can effectively protect web applications from XSS attacks.

Conclusion

This project provided a comprehensive understanding of Cross-Site Scripting (XSS) vulnerabilities by combining both theoretical knowledge and practical experimentation. Through the use of Damn Vulnerable Web Application (DVWA), different types of XSS attacks—Stored, Reflected, and DOM-Based—were successfully identified and exploited in a controlled lab environment. These experiments demonstrated how improper input handling and lack of output sanitization can allow malicious JavaScript to execute in a user’s browser, leading to serious security risks such as session hijacking and data theft.

Furthermore, the project emphasized the importance of defensive mechanisms in securing web applications. By implementing input validation, output encoding, and deploying ModSecurity with the OWASP Core Rule Set, the effectiveness of layered security defenses was clearly observed. After enabling these protections, malicious scripts failed to execute, and suspicious activities were detected and logged by the Web Application Firewall. Overall, this project highlights the critical need for secure coding practices and proactive security measures to mitigate XSS vulnerabilities and strengthen the security of modern web applications.