# Security testing report

**Vulnerability Assessment Report: Cross-Site Scripting (XSS)**

| | |
|---|---|
| **Vulnerability** | **Cross-Site Scripting (Reflected & Stored)** |
| **Target Application** | DVWA (Damn Vulnerable Web Application) |
| **Severity** | **High (Reflected) / Critical (Stored)** |
| **Affected Endpoints** | .../vulnerabilities/xss_r/ (Reflected)<br><br>.../vulnerabilities/xss_s/ (Stored) |
| **Affected Parameters** | name (Reflected)<br><br>Name, Message (Stored) |

---

**Executive Summary**

**Two types** of Cross-Site Scripting (XSS) vulnerabilities were identified and successfully exploited in the DVWA environment. These flaws stem from the application's failure to properly sanitize user-supplied input before rendering it back to the browser.

1. **Reflected XSS (High Severity):** The name parameter on the "Reflected XSS" page is vulnerable. An attacker can craft a malicious URL and trick a victim into clicking it. The victim's browser then executes the malicious script, which is "reflected" off the web server. This can be used for session hijacking or credential theft.
2. **Stored XSS (Critical Severity):** The Name and Message parameters on the "Stored XSS" (Guestbook) page are vulnerable. An attacker can submit a malicious script as a guestbook entry, which is then **permanently stored in the database**. This script automatically executes in the browser of **every user** who views the guestbook page, making it a far more severe vulnerability.

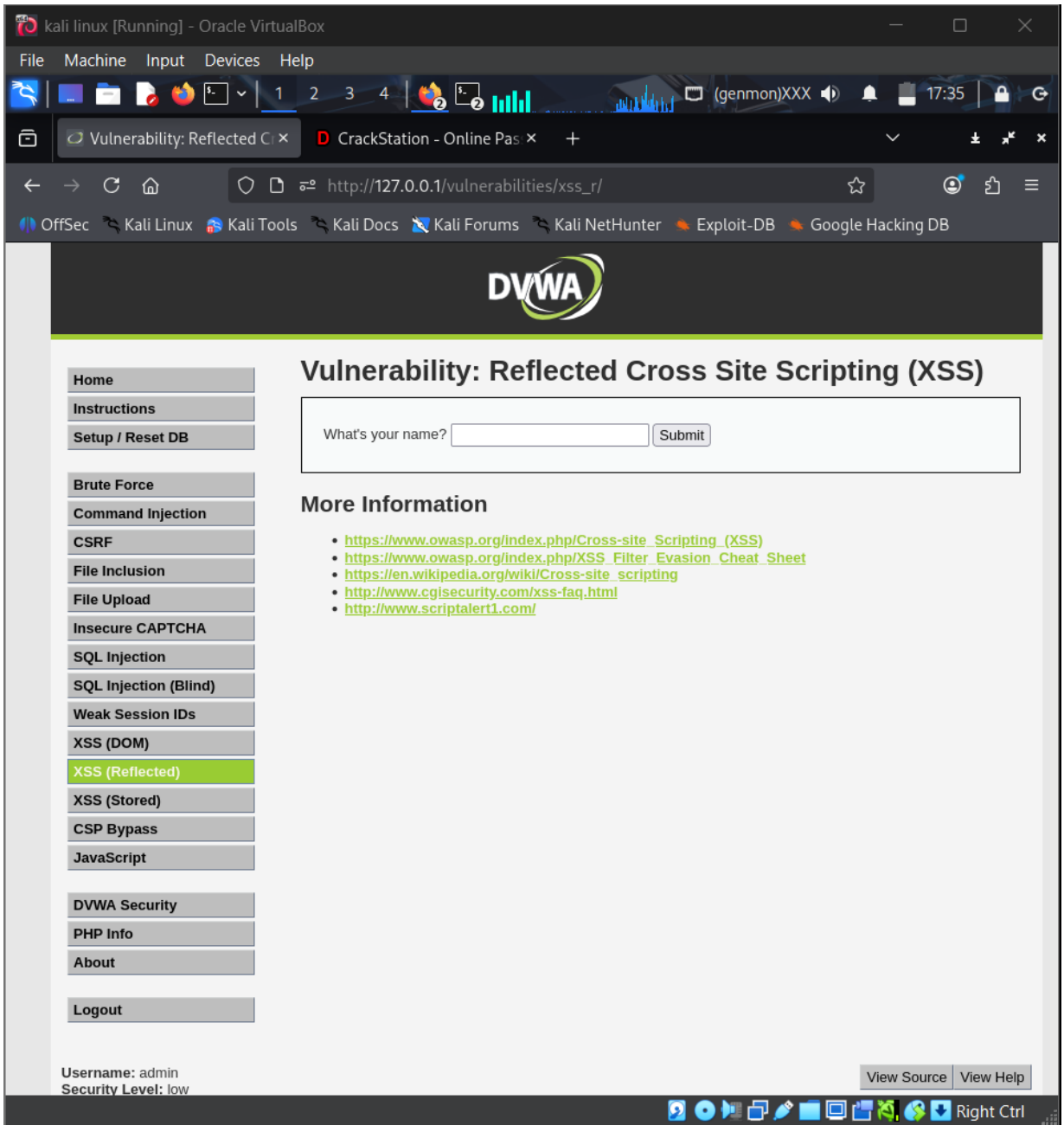
Successful exploitation allows an attacker to execute arbitrary JavaScript in a victim's browser, leading to session cookie theft, keylogging, and full account compromise. Remediation requires implementing **context-aware output encoding** on all user-supplied data.

---

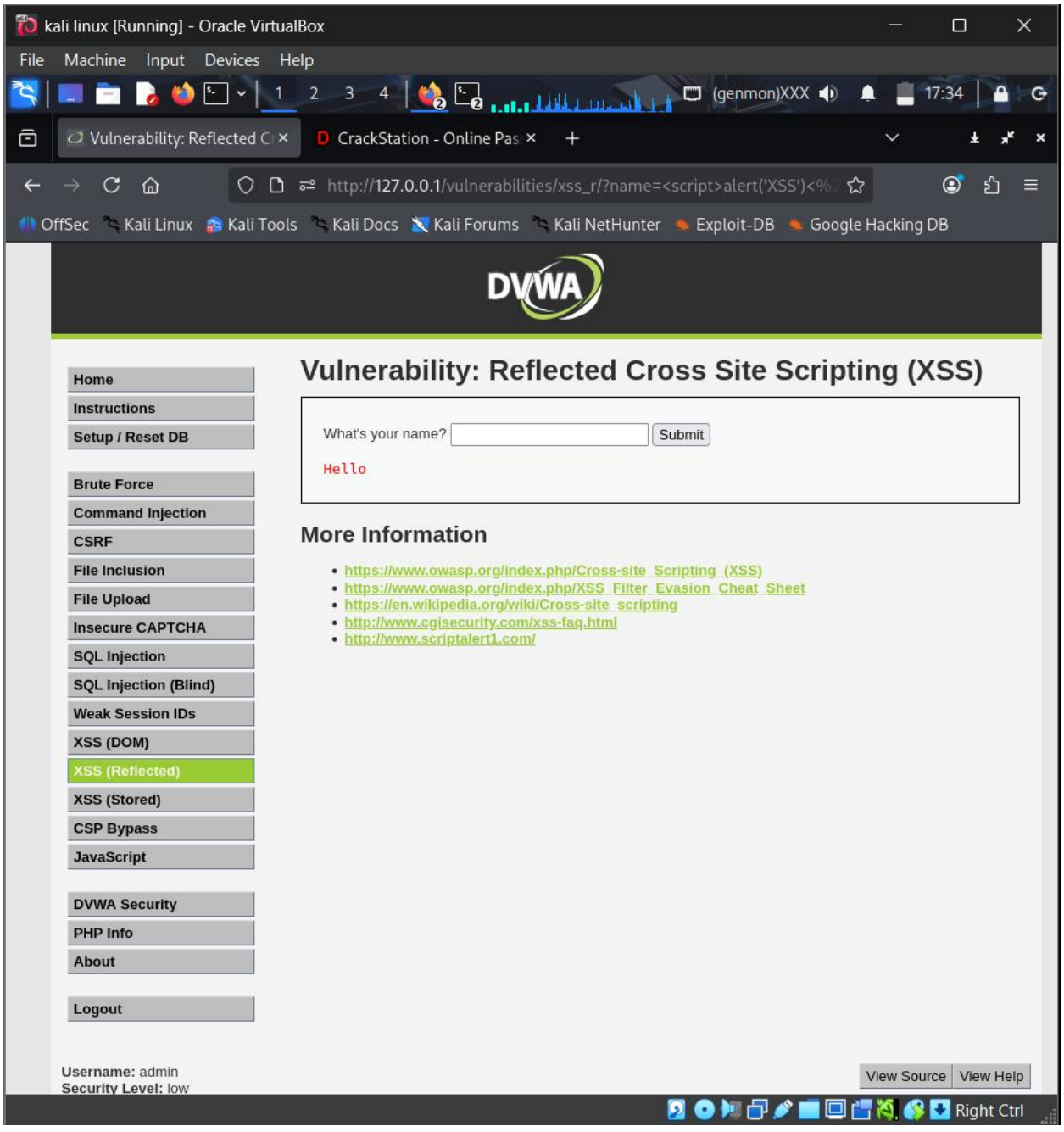
**Attack Narrative 1: Reflected XSS**

This attack demonstrates how a payload can be delivered via a single link.

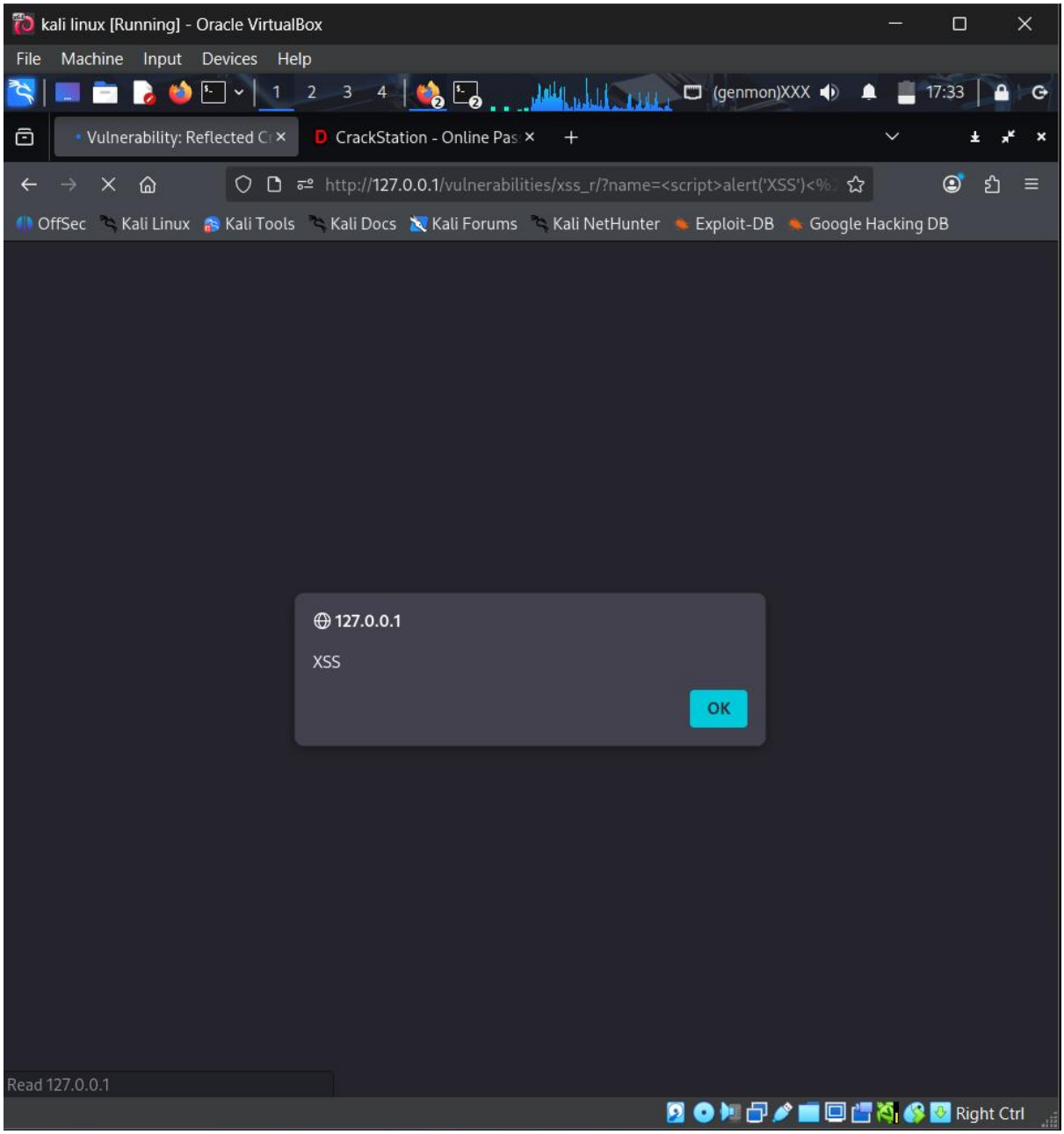
1. **Target Identification:** The "Vulnerability: Reflected Cross Site Scripting (XSS)" page (/xss_r/) was targeted. This page features an input field that takes a user's name and displays it back.



2. **Payload Injection:** A simple proof-of-concept (PoC) payload was injected into the name parameter via the URL:
    - **Payload:** <script>alert('XSS')</script>
    - **Malicious URL:** http://127.0.0.1/vulnerabilities/xss_r/?name=<script>alert('XSS')</script>



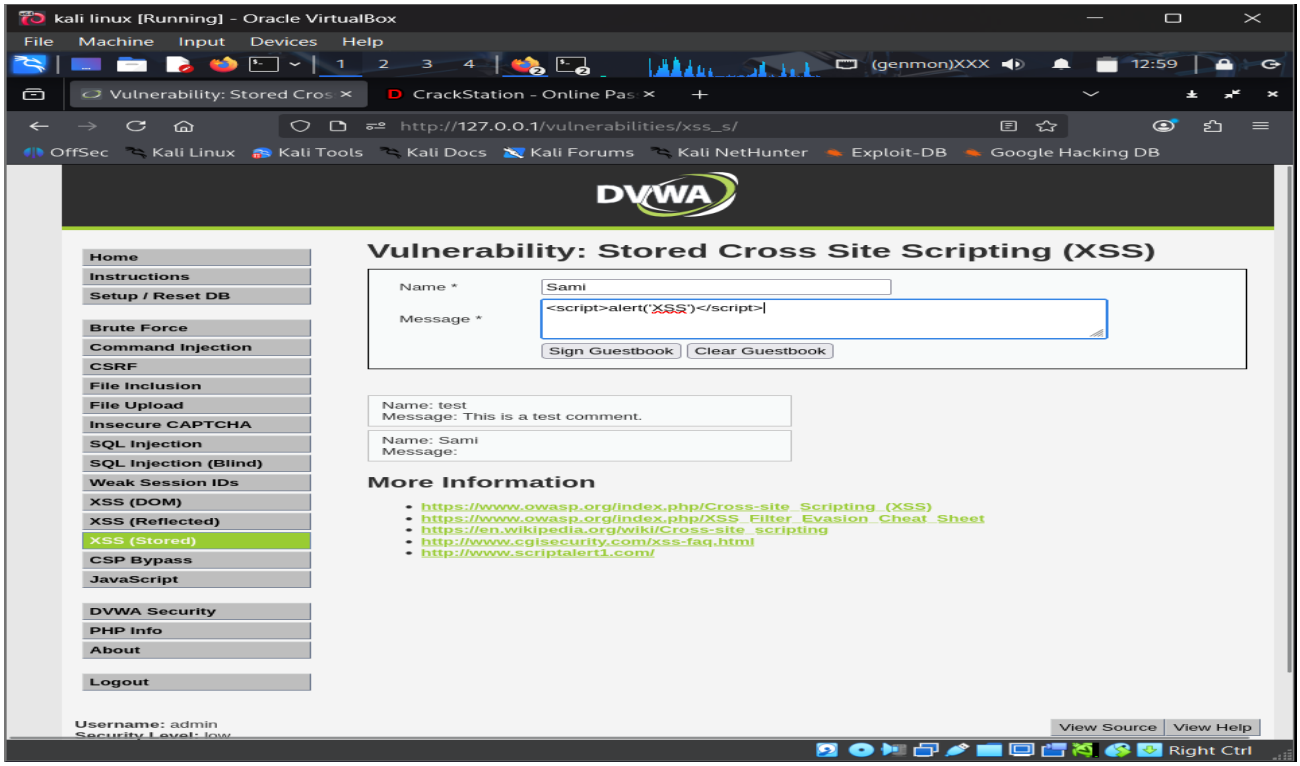
3. **Execution:** The web server received the request, took the malicious script from the name parameter, and inserted it directly into the HTML response without any encoding or sanitization.
4. **Result:** The victim's browser, trusting the content from the server, executed the embedded JavaScript. As shown in the screenshot, this triggered an alert box with the text "XSS", confirming the vulnerability.



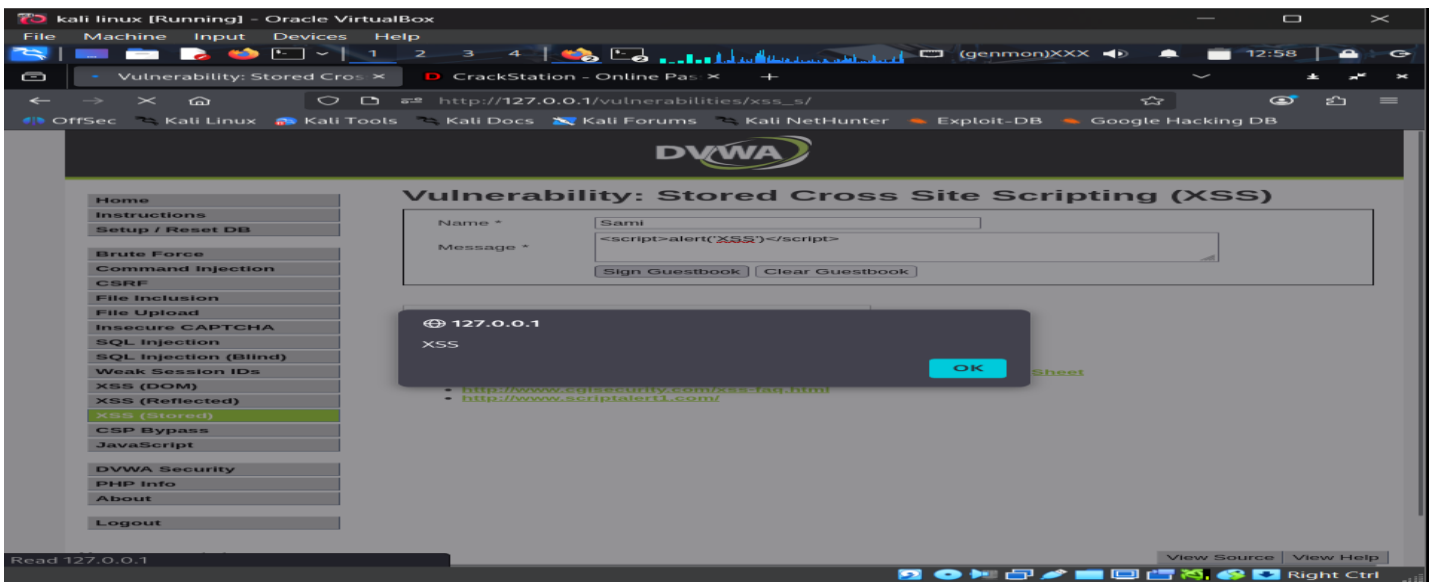
**Attack Narrative 2: Stored XSS**

This attack is more dangerous as it is persistent and affects all users.

1. **Target Identification:** The "Vulnerability: Stored Cross Site Scripting (XSS)" page (/xss_s/) was targeted. This page functions as a guestbook, storing and displaying all user-submitted entries.




2. **Payload Submission:** The attacker submitted a malicious payload into the guestbook form:
    - **Name:** Sami (and later test 2)
    - **Message:** <script>alert('XSS')</script>



3. **Payload Persistence:** The application accepted this input and saved the malicious script **directly into the database** as a valid guestbook message.
4. **Execution:** When *any* user (including the attacker or an administrator) loads or reloads the guestbook page, the application queries the database, retrieves the malicious script, and inserts it into the page's HTML.
5. **Result:** The browser executes the script, triggering the "XSS" alert box. The screenshots show this script executing multiple times (once for each malicious entry stored in the database), demonstrating that every user visiting this page is automatically attacked.

---

**Mitigation 3 : XSS**

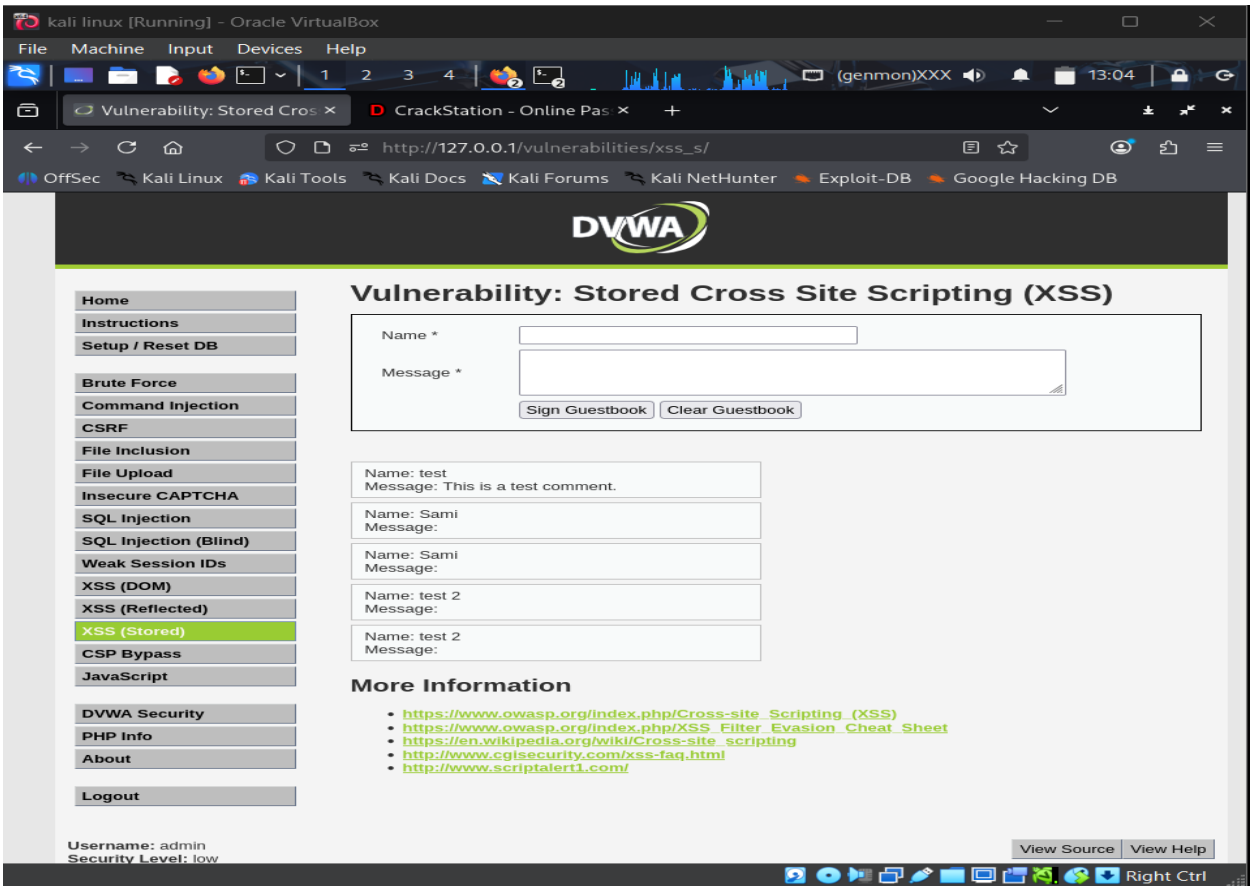
**1. Primary Mitigation: Context-Aware Output Encoding**

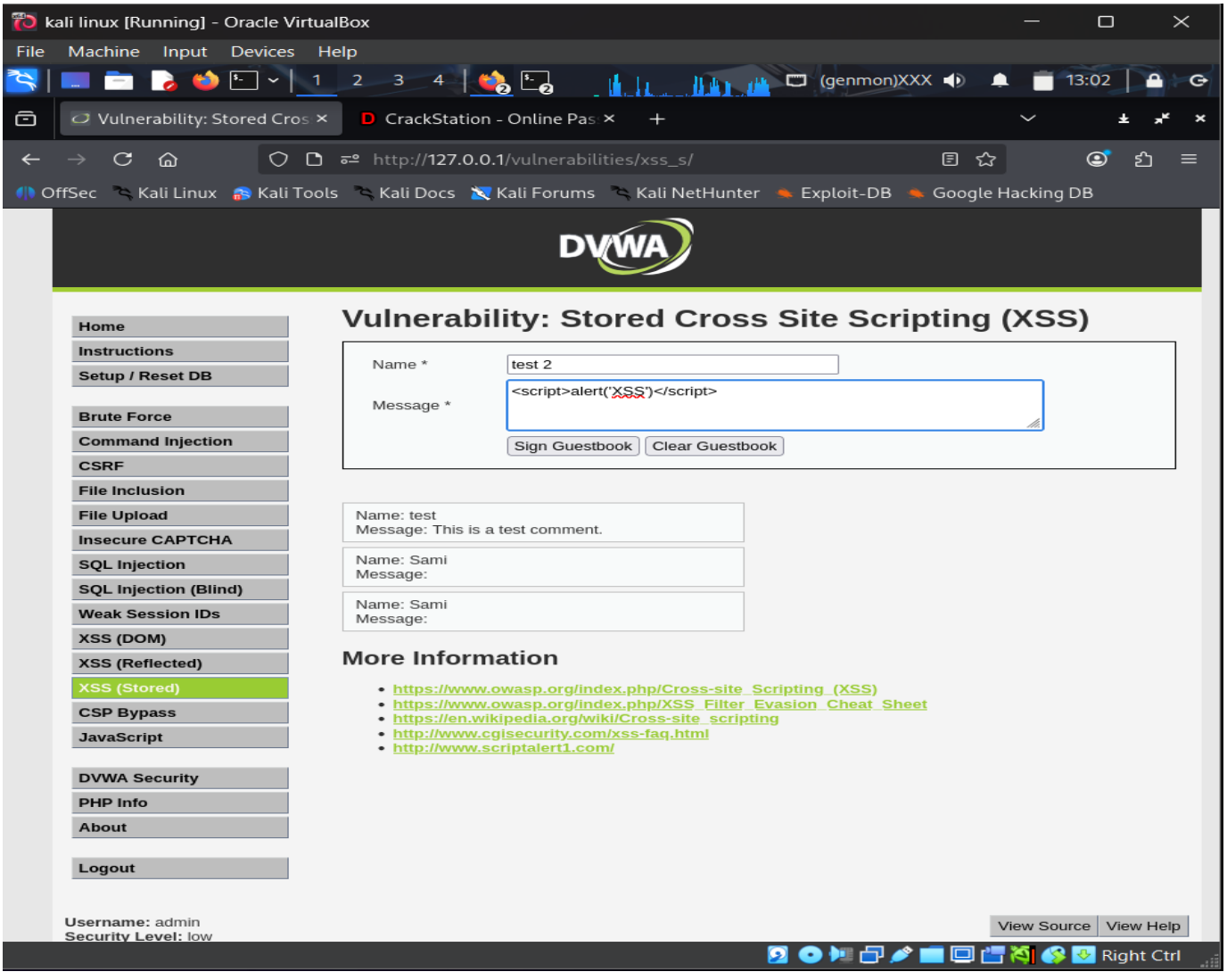
This is the **most critical and effective** defense. The solution is not to block input, but to safely encode the output before it is rendered in the HTML, so the browser interprets it as text, not as code.

- **What it is:** The process of converting special characters into their HTML entity equivalents.
    - < becomes <
    - > becomes >
    - " becomes "
    - ' becomes '
    - / becomes /
- **How it works:** When the browser encounters <script>alert(1)</script> in the HTML, it will **display the literal text** <script>alert(1)</script> to the user instead of executing it as a script.
- **Implementation (PHP Example):**

**Vulnerable Code:** echo ' ' . $_POST['message'] . ' ';

**Secure Code:** Use the htmlspecialchars() function to encode the data. echo ' ' . htmlspecialchars($_POST['message'], ENT_QUOTES, 'UTF-8') . ' ';


kali linux [Running] - Oracle VirtualBox
File Machine Input Devices Help
Vulnerability: Stored Cros
CrackStation - Online Pas
http://127.0.0.1/vulnerabilities/xss_s/
OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Stored Cross Site Scripting (XSS)
Name *
Message *
Sign Guestbook
Clear Guestbook
Name: test
Message: This is a test comment.
Name: Sami
Message:
Name: Sami
Message:
Name: test 2
Message:
Name: test 2
Message:
More Information
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
http://www.scriptalert1.com/
Username: admin
Security Level: low
View Source
View Help
Right Ctrl


kali linux [Running] - Oracle VirtualBox
File Machine Input Devices Help
Vulnerability: Stored Cros
CrackStation - Online Pas
http://127.0.0.1/vulnerabilities/xss_s/
OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Stored Cross Site Scripting (XSS)
Name *
test 2
Message *
<script>alert('XSS')</script>
Sign Guestbook
Clear Guestbook
Name: test
Message: This is a test comment.
Name: Sami
Message:
Name: Sami
Message:
More Information
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
http://www.scriptalert1.com/
Username: admin
Security Level: low
View Source
View Help
Right Ctrl