# DVWA XSS模块渗透测试报告

# ## 1. 测试概述

### 1.1 测试目标

本测试针对Damn Vulnerable Web Application (DVWA) 中的跨站脚本(XSS)模块进行安全性测试，旨在发现并验证XSS漏洞，评估其风险等级，并提供修复建议。

### 1.2 测试环境

- 操作系统: Windows 11

- 浏览器: Google Chrome 118.0.5993.88

- DVWA版本: 1.10

- 安全级别: Low/Medium/High/Impossible

### 1.3 测试方法

- 手动测试: 通过输入不同类型的XSS payloads进行测试

- 工具测试: 使用Burp Suite进行拦截和修改请求

- 验证方法: 检查是否能够成功执行JavaScript代码

## 2. 测试详情

### 2.1 Reflected XSS (Low)

#### 2.1.1 测试步骤

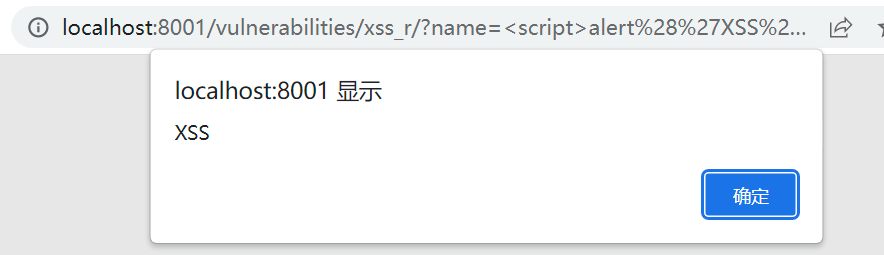
1. 访问DVWA平台，登录后进入XSS (Reflected)模块

2. 在输入框中输入测试payload: `<script>alert('XSS')</script>`

3. 点击提交按钮

2.1.2 测试结果

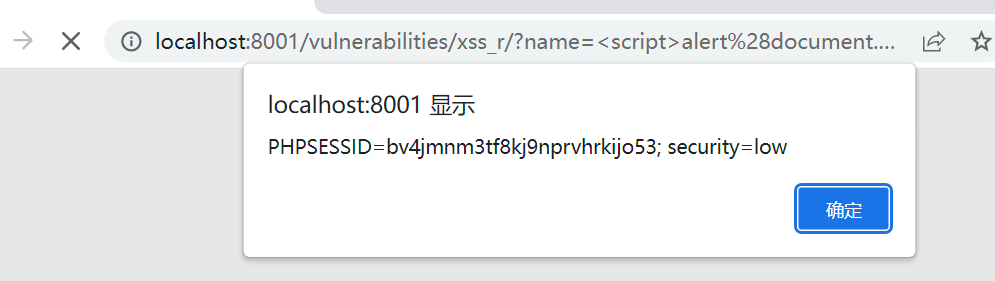
成功弹出包含'XSS'的警告框，证明存在Reflected XSS漏洞。



进阶测试:

<script>alert(document.cookie)</script>

可以获取页面cookie



将URL拼接payload 当访问url就会触发

攻击者使用窃取的 Cookie，通过浏览器 “开发者工具→Application→Cookies” 替换自己的 Cookie，即可冒充受害者登录 DVWA，执行任意操作（如查看敏感信息、修改权限

#### 2.1.3 漏洞分析

- 输入的JavaScript代码未经过滤直接嵌入到HTML响应中

- 服务器未对特殊字符进行转义处理

- 风险等级: 高

### 2.2 Reflected XSS (Medium)

#### 2.2.1 测试步骤

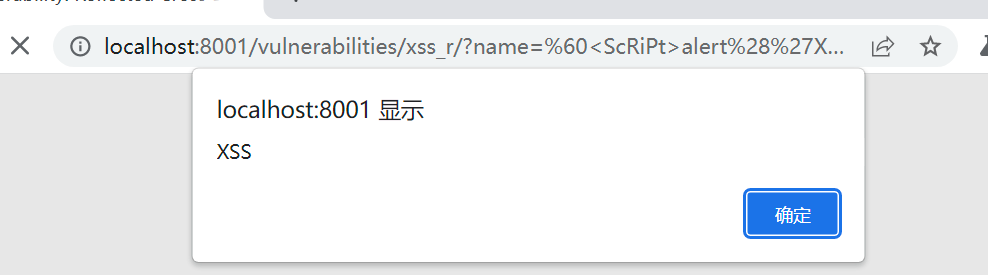
1. 尝试基本XSS payload: `<script>alert('XSS')</script>`

2. 尝试绕过过滤的payload: `<ScRiPt>alert('XSS')</ScRiPt>` 或 `<img src=x onerror=alert('XSS')>`

#### 2.2.2 测试结果

基本payload被过滤，但大小写混合或使用其他标签的payload可以成功执行。

截图如下:



#### 2.2.3 漏洞分析

- 服务器对输入进行了简单过滤，但未完全防止XSS攻击

- 仅过滤了小写的`<script>`标签，但未处理大小写混合或其他可执行JavaScript的标签

- 风险等级: 中高

### 2.3 Reflected XSS (High)

#### 2.3.1 测试步骤

1. 尝试各种常见XSS payloads

2. 尝试使用HTML事件处理器: `<img src=x onerror=alert('XSS')>`

3. 尝试使用编码绕过: `&#x3C;script&#x3E;alert('XSS')&#x3C;/script&#x3E;`

4 尝试 其他标签可以绕过: 如:

<svg onload=alert('XSS')>

<input type="text" onfocus=alert('XSS') autofocus>

页面会持续地弹窗xss

#### 2.3.2 测试结果

大多数payloads被过滤，但特定条件下仍可能存在绕过方式。





#### 2.3.3 漏洞分析

- 服务器实施了更严格的过滤，但仍可能存在绕过方法

- 风险等级: 中

源代码如下:

if( array\_key\_exists( "name", $\_GET ) && $\_GET[ 'name' ] != NULL ) {

// Get input

$name = preg\_replace( '/<(.\*)s(.\*)c(.\*)r(.\*)i(.\*)p(.\*)t/i', '', $\_GET[ 'name' ] );

// Feedback for end user

$html .= "<pre>Hello ${name}</pre>";

}

?> 匹配过滤了任何包含<script地不区分大小写地字符,但是没有对其他标签进行过滤

### 2.4 Reflected XSS (Impossible)

#### 2.4.1 测试步骤

1. 尝试各种XSS payloads

2. 分析服务器响应

#### 2.4.2 测试结果

所有XSS payloads均被有效过滤，未发现可执行JavaScript的情况。

#### 2.4.3 漏洞分析

- 服务器实施了全面的输入验证和输出编码

- 使用了适当的安全措施防止XSS攻击

- 风险等级: 低

### 2.5 Stored XSS

(此处需要根据实际测试情况填写)

## 3. 漏洞总结

| 漏洞类型 | 安全级别 | 风险等级 | 是否可利用 |

|---------|---------|---------|----------|

| Reflected XSS | Low | 高 | 是 |

| Reflected XSS | Medium | 中高 | 是 |

| Reflected XSS | High | 中 | 可能 |

| Reflected XSS | Impossible | 低 | 否 |

## 4. 修复建议

### 4.1 通用修复建议

1. 对所有用户输入进行严格验证和过滤

2. 使用适当的输出编码机制，如HTML实体编码

3. 实施内容安全策略(CSP)

4. 避免使用eval()等危险函数

5. 对Cookie设置HttpOnly和Secure标志