

XSS漏洞

郭益華

XSS漏洞介紹

XSS漏洞是一種網站安全漏洞，攻擊者利用網站對使用者輸入的數據沒有過濾或過濾不嚴格，將惡意腳本程式碼插入到網頁中，當使用者瀏覽該網頁時，腳本程式碼會被執行，從而達到攻擊使用者的目的。

以下是XSS漏洞的幾種類型：

- **反射型XSS**：攻擊者將腳本程式碼注入到網址中，當使用者點擊該網址時，腳本程式碼會被執行。
- **存儲型XSS**：攻擊者將腳本代碼存儲到網站的數據庫中，當使用者訪問該網站時，腳本程式碼會被執行。
- **DOM型XSS**：攻擊者將腳本程式碼注入到網頁中的DOM元素中，當使用訪問該網頁時，腳本程式碼會被執行。

目錄

1. [發掘基本的反射型XSS](#)
2. [發掘中等的反射型XSS](#)
3. [發掘高級的反射型XSS](#)
4. [發掘儲存型XSS](#)
5. [發掘中等的儲存型XSS](#)
6. [XSS漏洞修復](#)

1. 發掘基本的反射型XSS

反射型XSS說明

- 被動型，非持久及不儲存的漏洞
- 只有當目標網站瀏覽某個特殊URL時，才會執行該漏洞
- 須滿足有人發送URL及有人點擊及該URL兩個條件，漏洞才會執行
- Example:
`http://target.com/page.php?somthing= <script>alert("XSS")</script>`

將 Security 調整為 low

DVWA Security

Script Security

Security Level is currently **low**.

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

The security level changes the vulnerability level of DVWA.

low



Submit

點選 XSS reflected

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

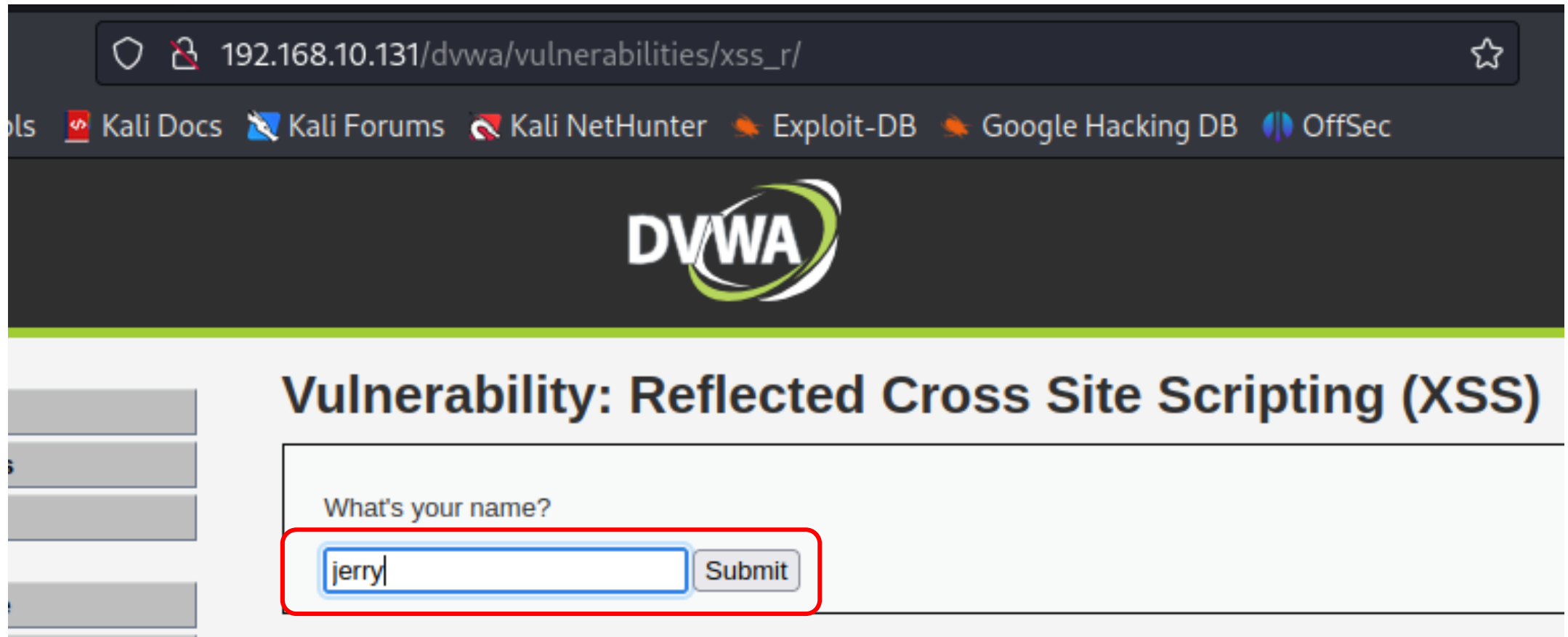
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

More info

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

輸入任意文字



The screenshot shows a web browser window with the address bar displaying `192.168.10.131/dvwa/vulnerabilities/xss_r/`. The browser's bookmark bar includes links to `Kali Docs`, `Kali Forums`, `Kali NetHunter`, `Exploit-DB`, `Google Hacking DB`, and `OffSec`. The page header features the DVWA logo. The main heading is **Vulnerability: Reflected Cross Site Scripting (XSS)**. Below this, a form asks "What's your name?". The input field contains the text "jerry" and is highlighted with a red border. A "Submit" button is located to the right of the input field.

192.168.10.131/dvwa/vulnerabilities/xss_r/

Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

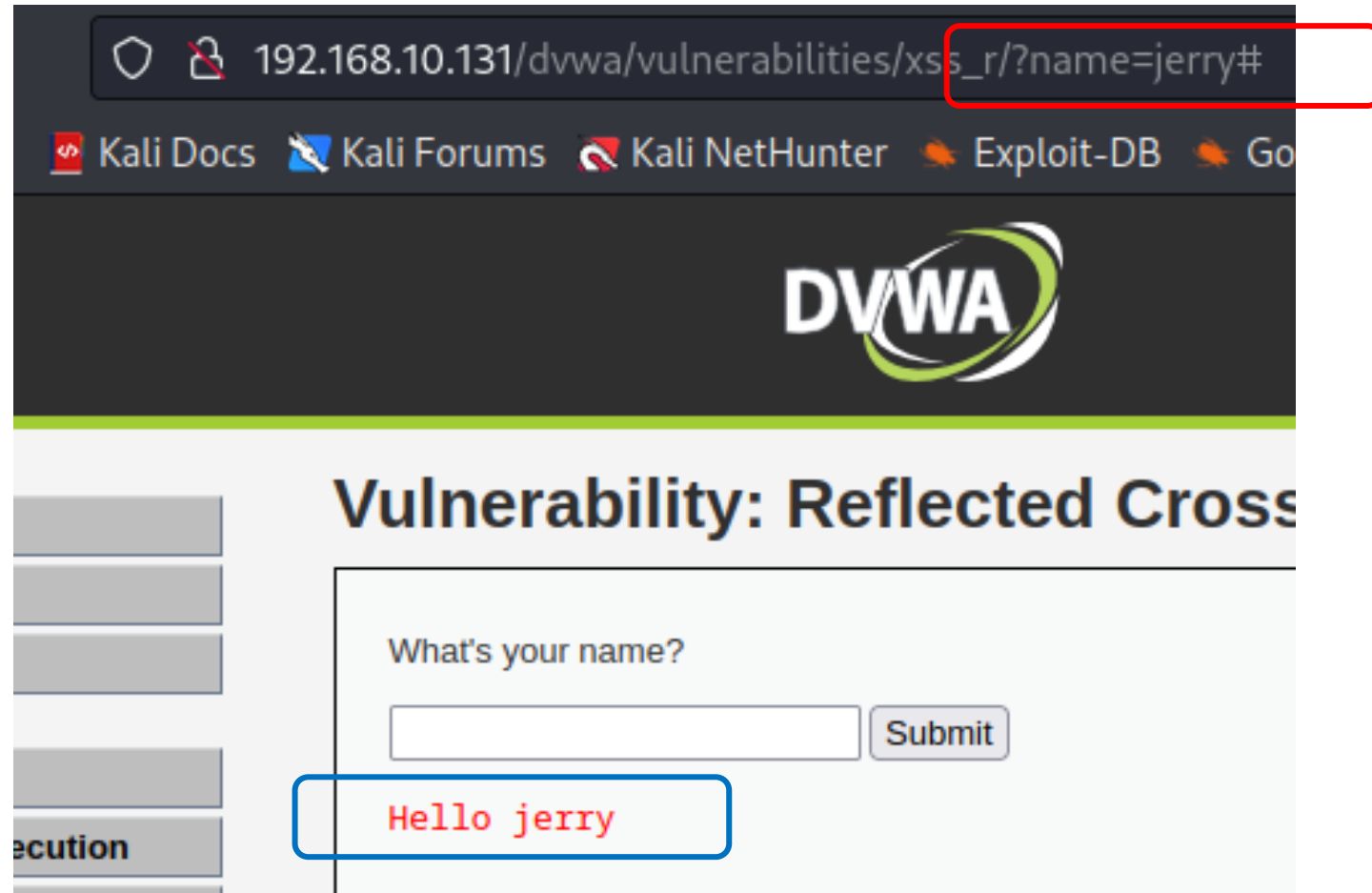
DVWA

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

jerry Submit

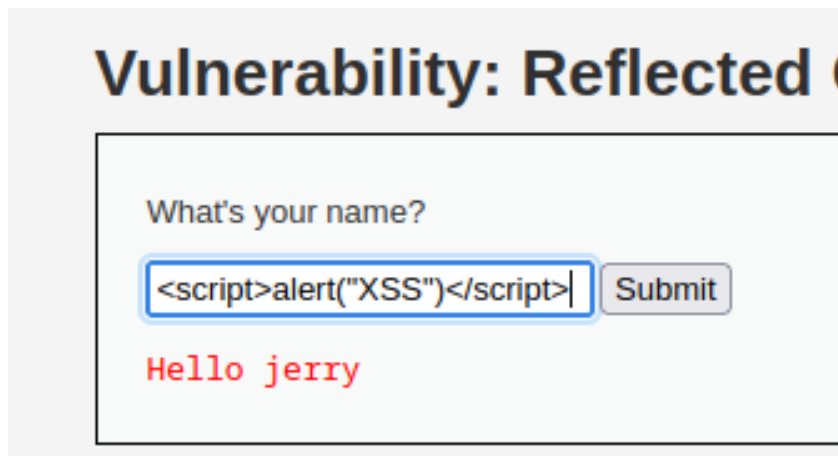
可觀察得知這是一個GET參數



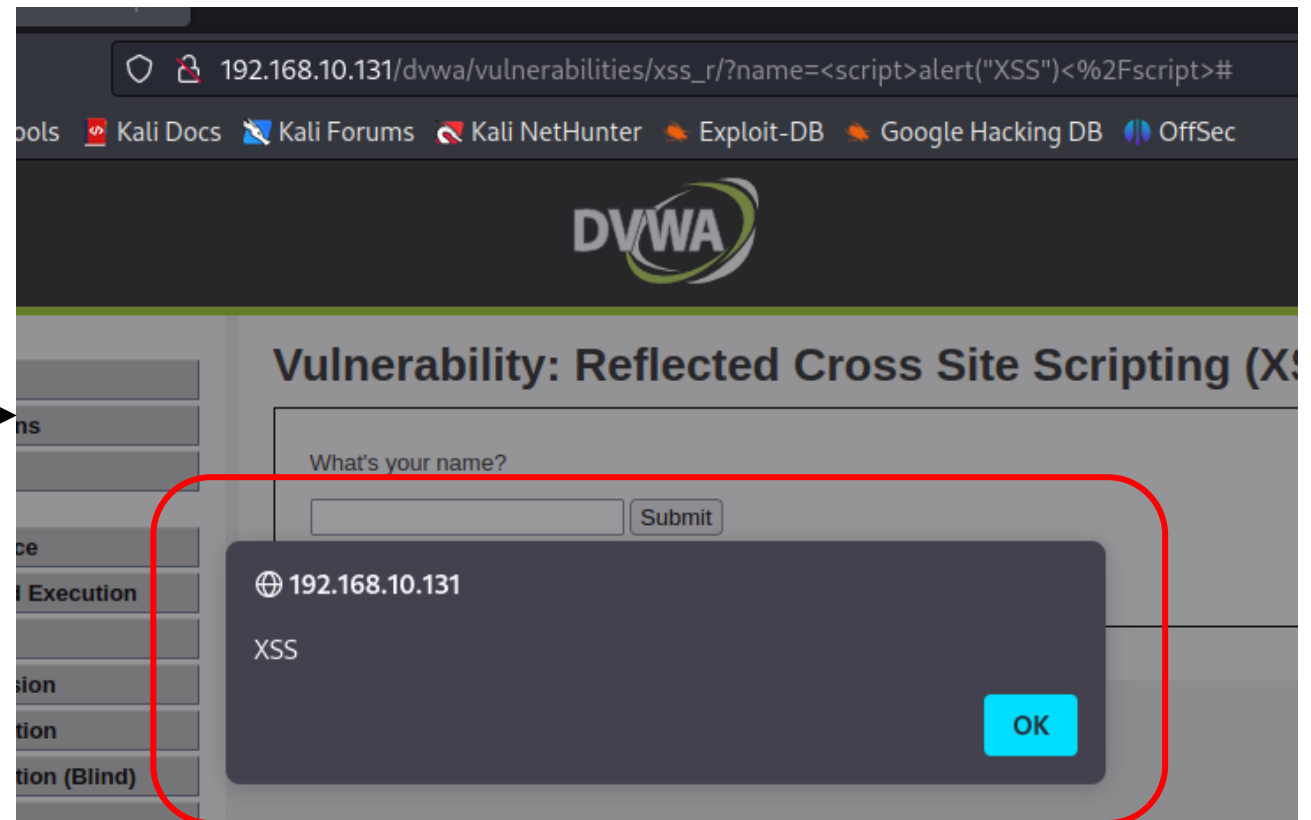
輸入javascript查看是否能收到目標網站的回應

跳出警告語法:

```
<script>alert("XSS")</script>
```

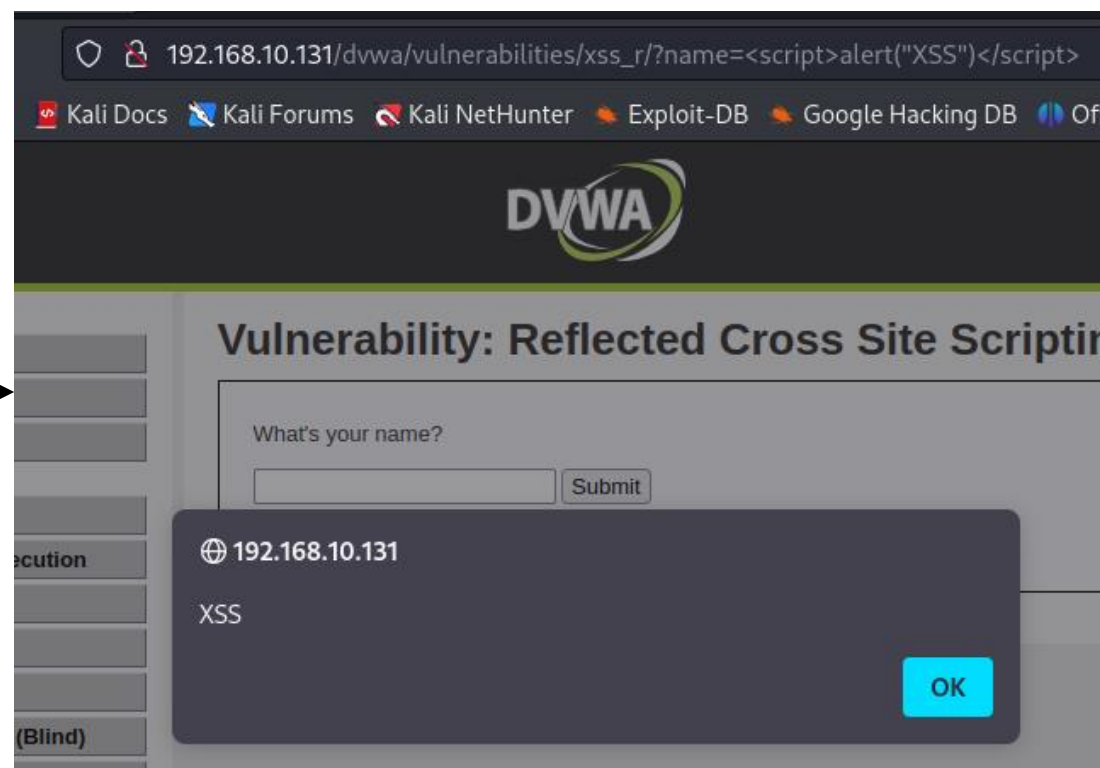
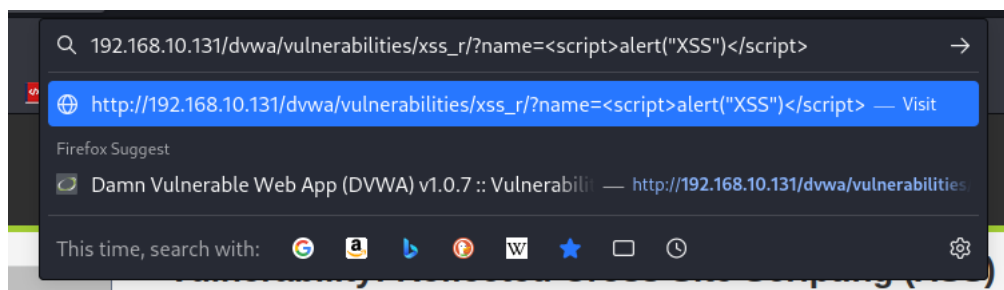


成功收到從目標網站所跳出的警告，代表可利用這樣的方式操控目標網站



改在網址欄輸入

成功收到從目標網站所跳出的警告，與前面一樣的結果



2.發掘中等的反射型XSS

輸入與前面一樣的程式碼

Vulnerability: Reflected Cross Site Script

What's your name?

More info

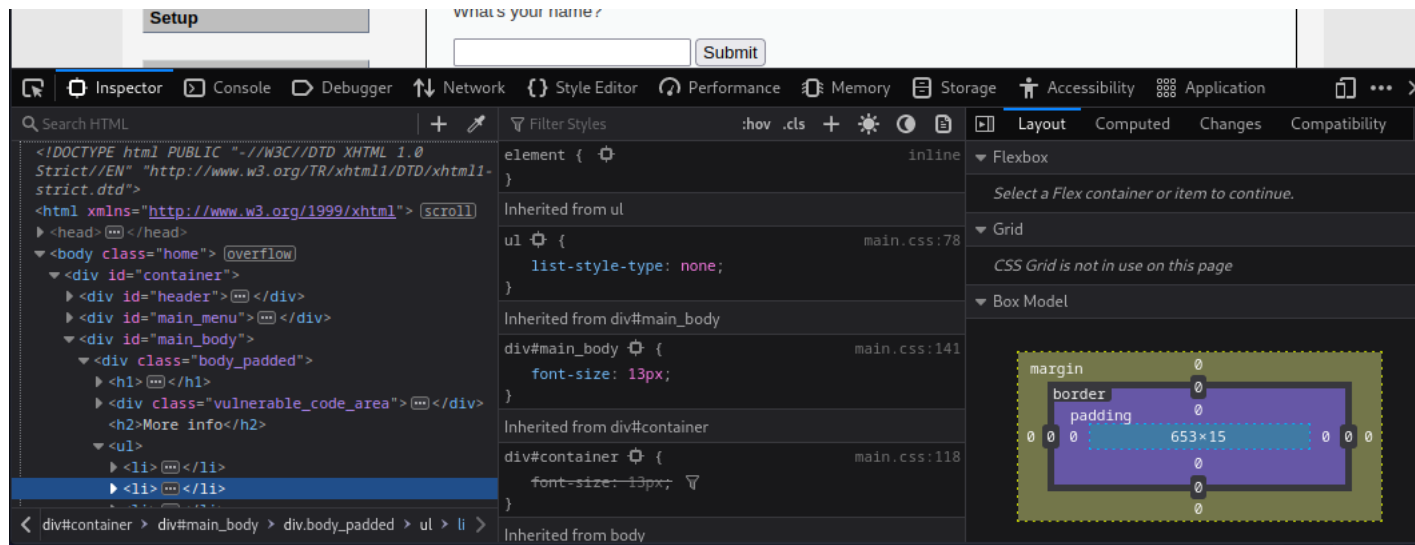
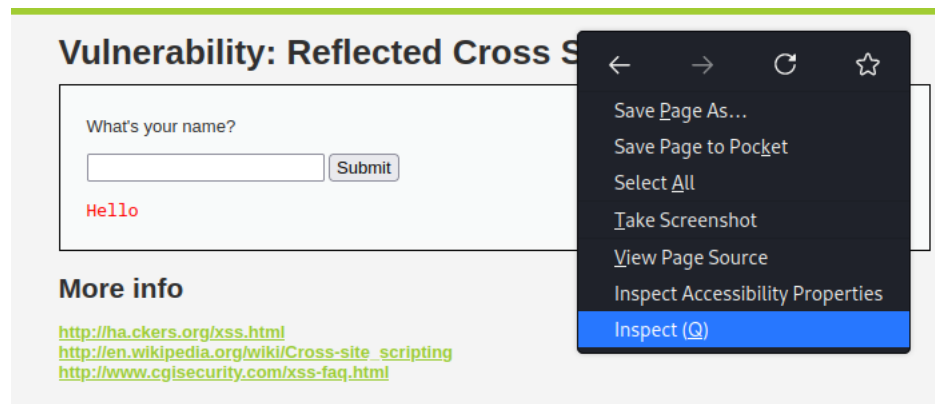
Vulnerability: Reflected Cross Site Script

What's your name?

🌐 192.168.10.131

XSS


滑鼠右鍵開啟檢查網頁程式碼



可看到這是我們所寫入的程式碼

```
    </form>  
    ▼ <pre>  
        Hello  
        <script>alert("XSS")</script>  
    </pre>
```

將 Security 調整為 medium

DVWA Security 

Script Security

Security Level is currently **medium**.

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

The security level changes the vulnerability level of DVWA.

medium ▼ Submit

點選 XSS reflected

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?
 Submit

More info

<http://hacker.org/xss.html>
http://en.wikipedia.org/wiki/Cross-site_scripting
<http://www.cgisecurity.com/xss-faq.html>

輸入與前面一樣的程式碼

Vulnerability: Reflected Cross

What's your name?

這次就沒有注入成功了，無跳出警告

Vulnerability: Reflected

What's your name?

Submit

Hello alert("XSS")

檢視網頁程式碼

推測為我們輸入的注入程式碼被過濾掉了

```
▶ <form name="XSS" action="#" method="GET">
  </form>
  <pre>Hello alert("XSS")</pre>
</div>
```

修改程式碼大小寫嘗試

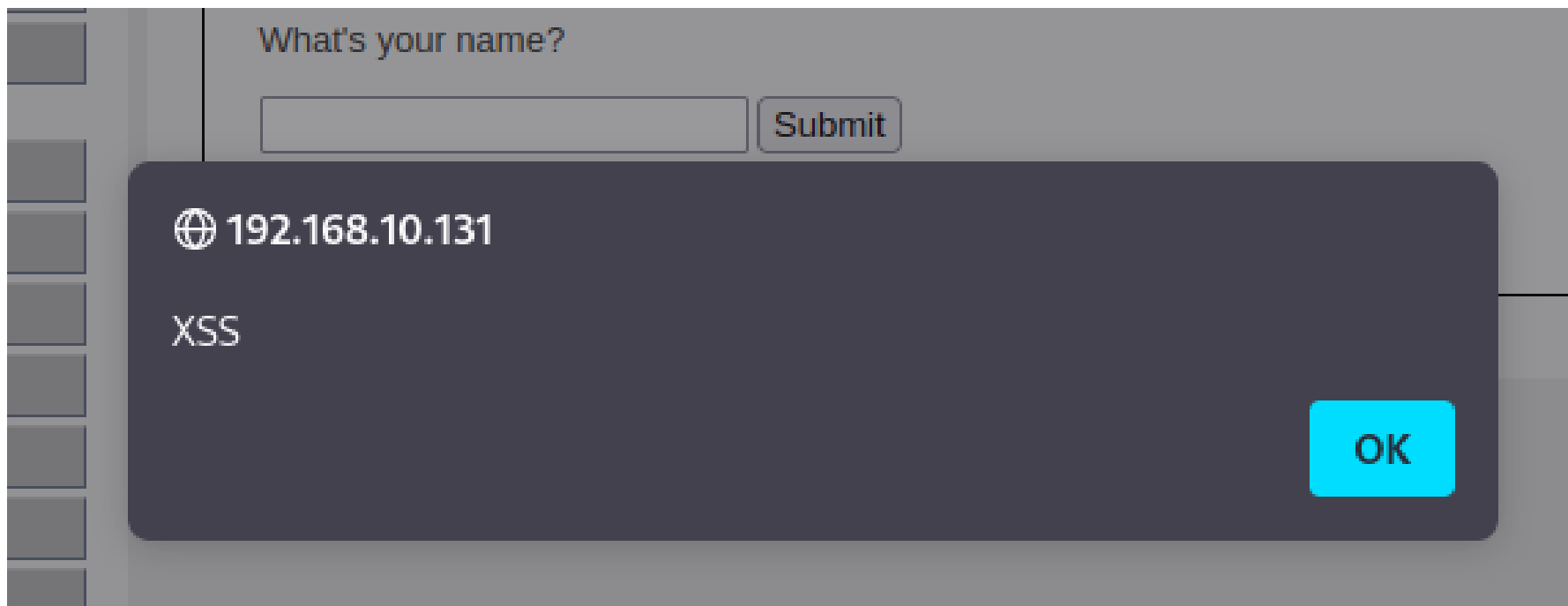
```
<sCripT>alert("XSS")</scRipt>
```

Vulnerability: Reflected C

What's your name?

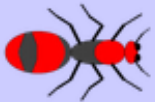
Hello alert("XSS")

成功繞過過濾，跳出警告



3.發掘高級的反射型XSS

開啟 Mutillidae



Mutillidae: Born to be Hacked

Version: 2.1.19 **Security Level: 0 (Hosed)** **Hints: Disabled (0 - I try harder)** **Not Logged In**

[Home](#) [Login/Register](#) [Toggle Hints](#) [Toggle Security](#) [Reset DB](#) [View Log](#) [View Captured Data](#)


[Core Controls](#)

[OWASP Top 10](#)

[Others](#)

[Documentation](#)

[Resources](#)



Site


Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10

Latest Version / Installation

- [Latest Version](#)
- [Installation Instructions](#)
- [Usage Instructions](#)
- [Get rid of those pesky PHP errors](#)
- [Change Log](#)
- [Notes](#)


開啟 Password Generator

OWASP Top 10 > A1 – Injection > JavaScript Injection > Password Generator

| | | | |
|--|---|--------------------------------|------------------------------------|
| OWASP Top 10 | A1 - Injection ▶ | SQLi - Extract Data ▶ | Vulnerable PHP |
| Others | A2 - Cross Site Scripting (XSS) ▶ | SQLi - Bypass Authentication ▶ | P Top 10 |
| Documentation | A3 - Broken Authentication and Session Management ▶ | SQLi - Insert Injection ▶ | |
| Resources | A4 - Insecure Direct Object References ▶ | Blind SQL via Timing ▶ | |
|  | A5 - Cross Site Request Forgery (CSRF) ▶ | SQLMAP Practice Target ▶ | |
| | A6 - Security Misconfiguration ▶ | HTML Injection (HTMLi) ▶ | |
| | A7 - Insecure Cryptographic Storage ▶ | HTMLi via HTTP Headers ▶ | |
| | A8 - Failure to Restrict URL Access ▶ | HTMLi Via DOM Injection ▶ | |
| | A9 - Insufficient Transport Layer Protection ▶ | HTMLi Via Cookie Injection ▶ | |
| | A10 - Unvalidated Redirects and Forwards ▶ | Command Injection ▶ | |
| | | JavaScript Injection ▶ | ed or you may build your own colle |
| | | HTTP Parameter Pollution ▶ | Those "Back" Buttons |
| | | Cascading Style Injection ▶ | Password Generator |

開啟畫面，點選 Generate

Password Generator

 **Back**

Password Generator

**Making strong passwords is important.
Click the button below to generate a password.**

This password is for anonymous

Generate

會產生一個密碼

Password Generator

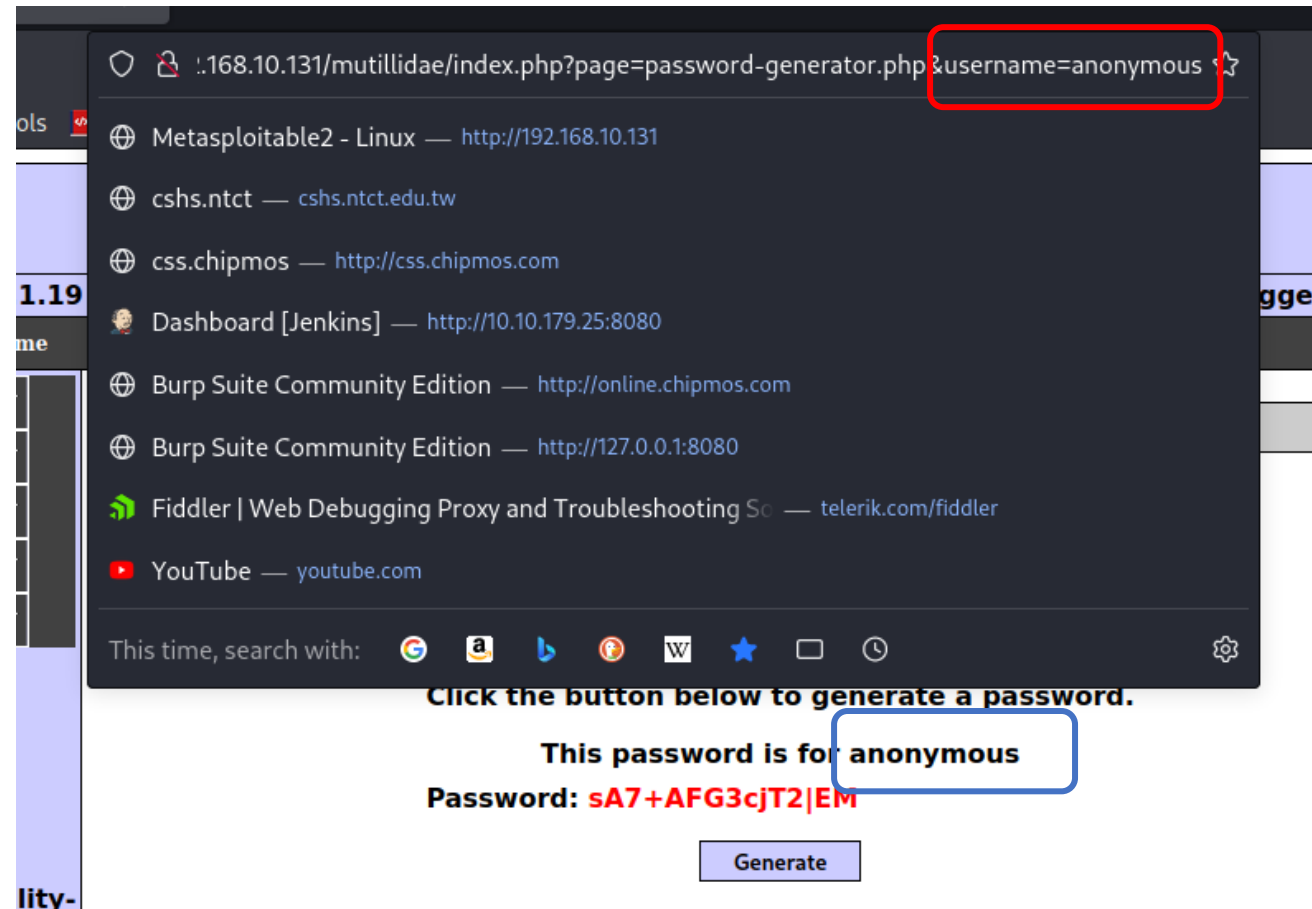
**Making strong passwords is important.
Click the button below to generate a password.**

This password is for anonymous

Password: sA7+AFG3cjT2|EM

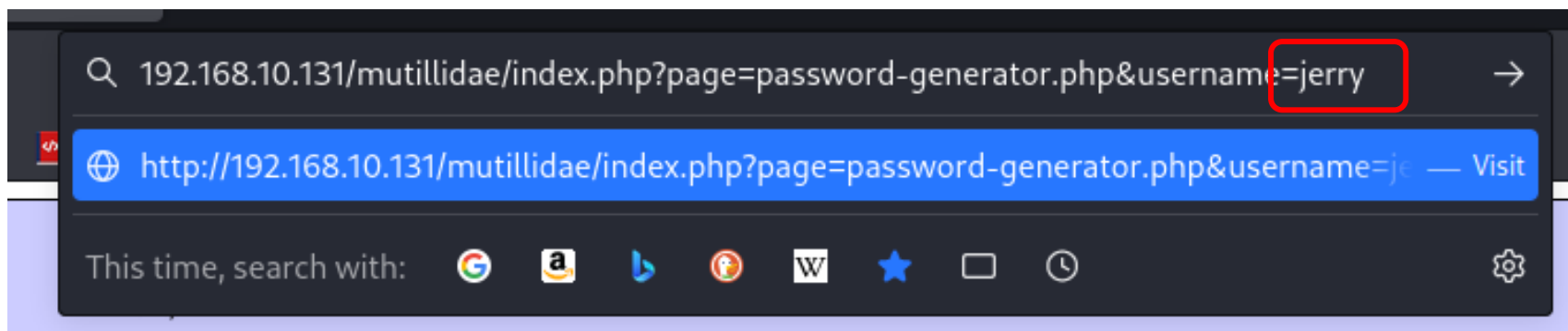
Generate

在網址欄可看到一個anonymous參數



輸入任意文字測試

可觀察到 anonymous 變為我們所輸入的文字



Password Generator

**Making strong passwords is important.
Click the button below to generate a password.**

This password is for jerry

Generate

點選 generate 一樣會產生一個密碼

Password Generator

Making strong passwords is important.
Click the button below to generate a password.

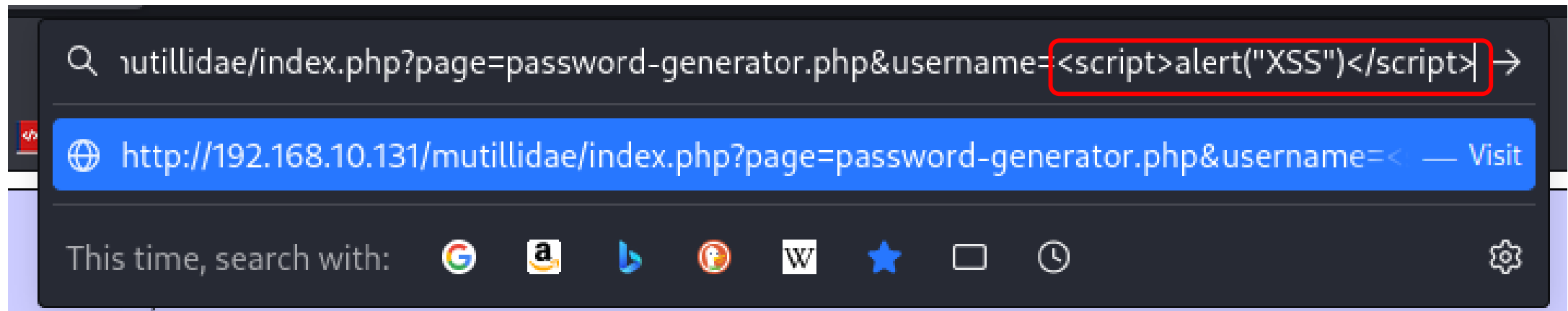
This password is for jerry

Password: **ps{:1&.e8*4Qg=t**

Generate

輸入與先前相同的程式碼注入

`<script>alert("XSS")</script>`



程式碼沒有執行，沒有跳出警告

Password Generator

**Making strong passwords is important.
Click the button below to generate a password.**

Generate

```
"; }catch(e){ alert("Error: " + e.message); }// end catch
```

但顯示出了不該出現在頁面的程式碼，代表我們所輸入的程式碼注入影響了網站程式碼

查看程式碼

原本:

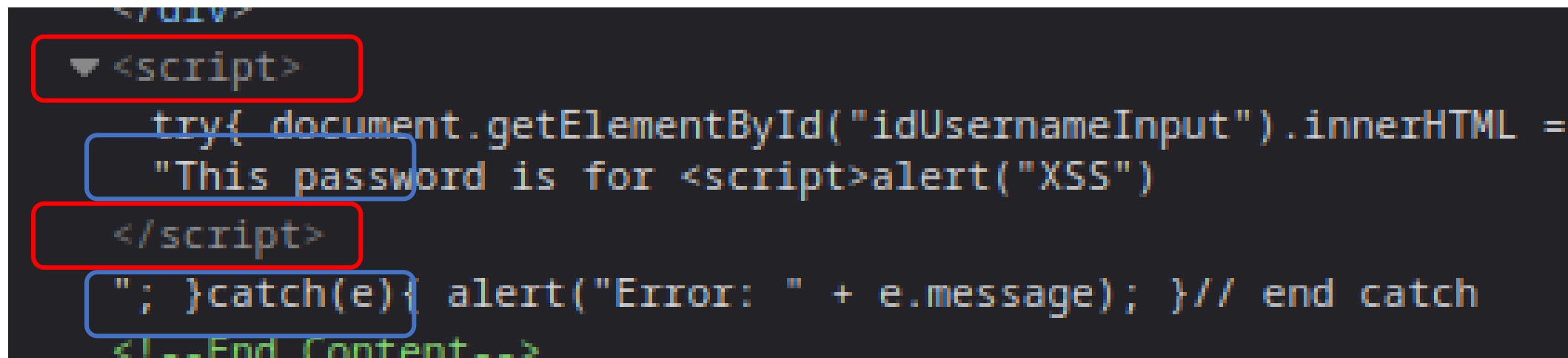
```
<script>alert("XSS")</script>
```

修改後:

```
" ;alert('XSS' );//
```

思路:

1. 可看到 password 這個程式碼本身已經寫在 `<script> </script>` 裡面，所以我們並不需要再加上 `<script> </script>`，修改為 `alert("XSS")`
2. 接著看到 `This password` 前與 `catch` 大括號前都有 `"`，代表他們本來是一句的，而為了讓我們的注入程式碼可順利執行，促使在我們的程式碼之前結束，程式碼需再修改為 `" alert("XSS")`
3. 除此之外，也須加上 `;` 代表一個程式碼段落的結束，程式碼需修改為 `" ;alert('XSS');`
4. 最後，為了使程式碼的獨立完整性並能順利執行，後面需再加上 `//` 註解掉我們程式碼後的網站程式碼，程式碼需修改為 `" ;alert('XSS');//`

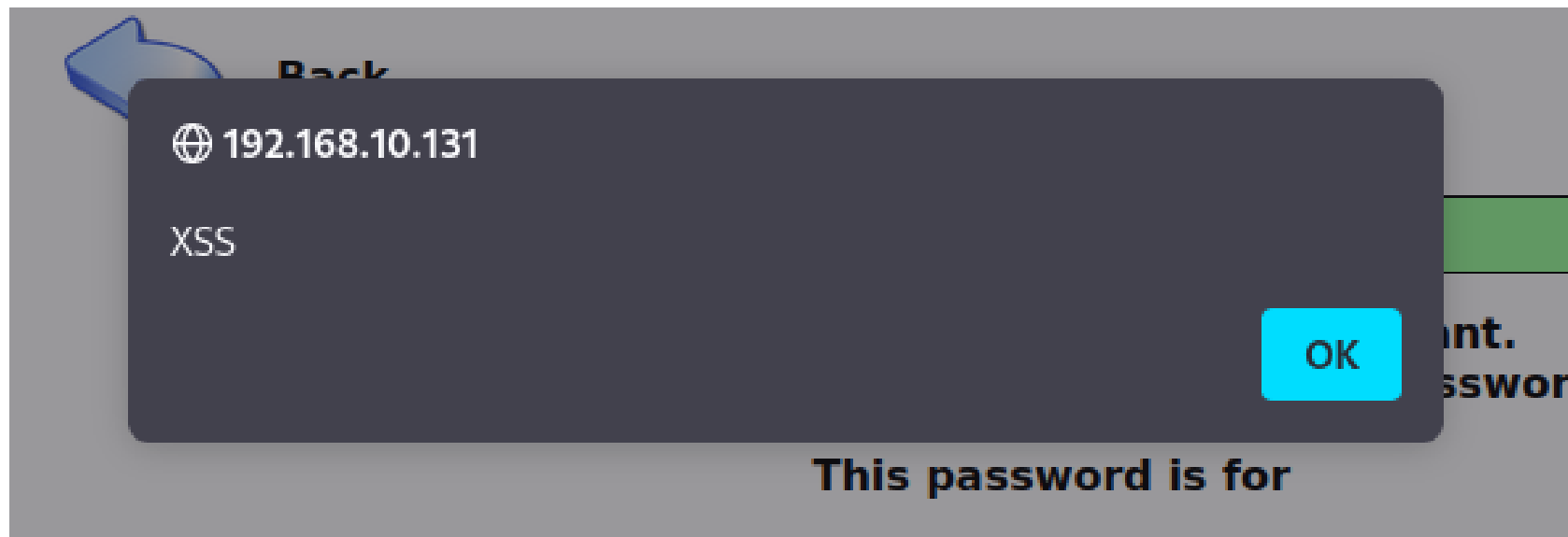


```
</div>
<script>
  try{ document.getElementById("idUsernameInput").innerHTML =
    "This password is for <script>alert("XSS")
  </script>
  "; }catch(e){ alert("Error: " + e.message); }// end catch
<!--End Content-->
```

每個網站的情況都會有所不同，必須仔細觀察，而不是直接複製注入程式碼

實際測試，成功注入跳出警告

8.10.131/mutillidae/index.php?page=password-generator.php&username=" ;alert('XSS');// ☆



4.發掘儲存型XSS

儲存型XSS說明

- 持久性的，儲存在頁面或是資料庫上
- 注入的程式碼在每次加仔的頁面上執行
- 會比反射型XSS更加危險

將 Security 調整為 low

DVWA Security

Script Security

Security Level is currently **low**.

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

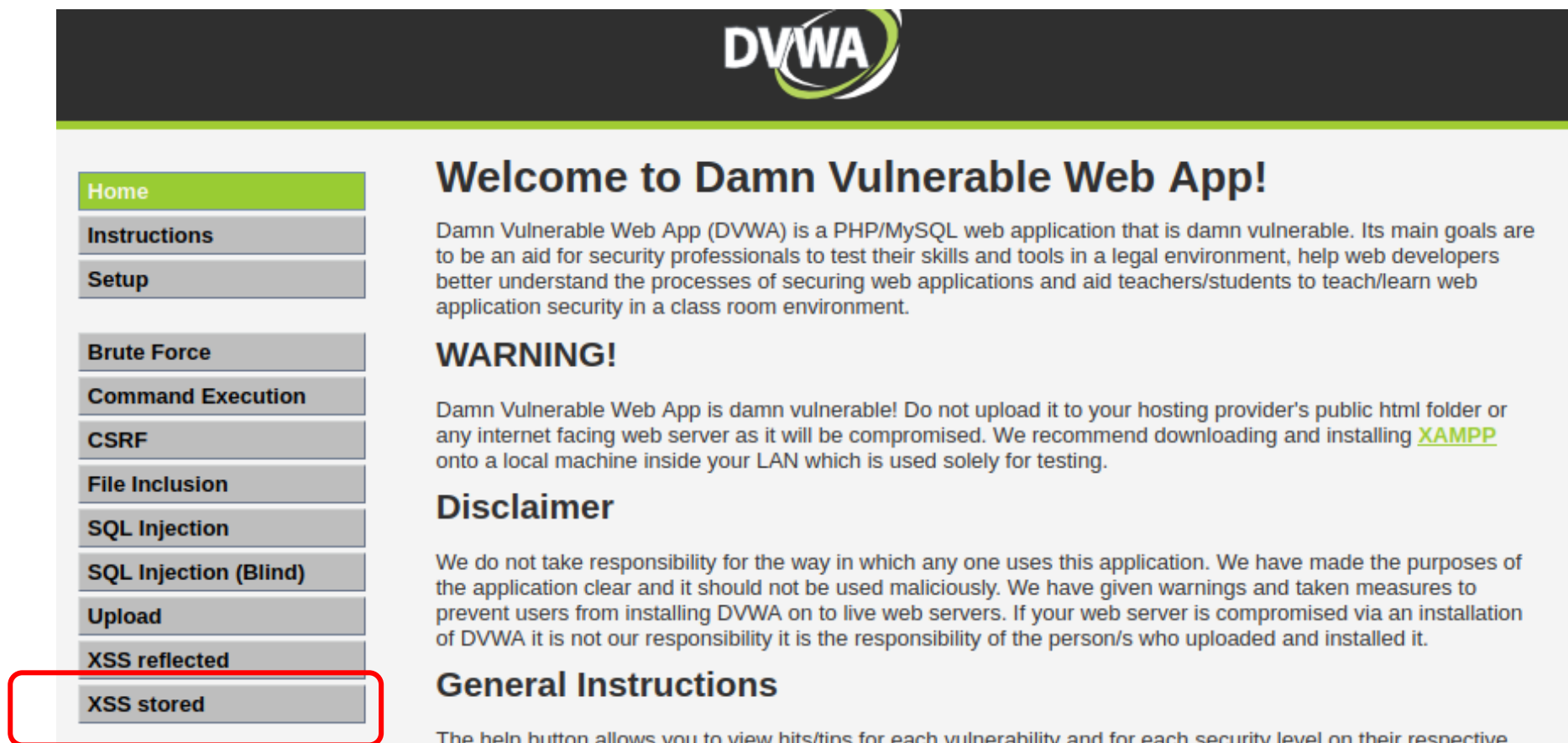
The security level changes the vulnerability level of DVWA.

low



Submit

點選 XSS stored



The image shows the main interface of the Damn Vulnerable Web App (DVWA). On the left is a sidebar menu with various security topics. The 'XSS stored' option at the bottom of this menu is highlighted with a red rectangular box. The main content area on the right has a dark header with the 'DVWA' logo. Below the header, it says 'Welcome to Damn Vulnerable Web App!' followed by a description of the app's purpose. There are sections for 'WARNING!' and 'Disclaimer'. At the bottom, there is a 'General Instructions' section.

DVWA

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

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

顯示畫面

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Sign Guestbook

Name: test

Message: This is a test comment.

輸入任意文字提交

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Name: jerry
Message: hello message

提交後可看到我們所輸入的訊息

開啟另外一台Windows登入，一樣可以看到Kali所提交的訊息

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

Name: jerry
Message: hello message

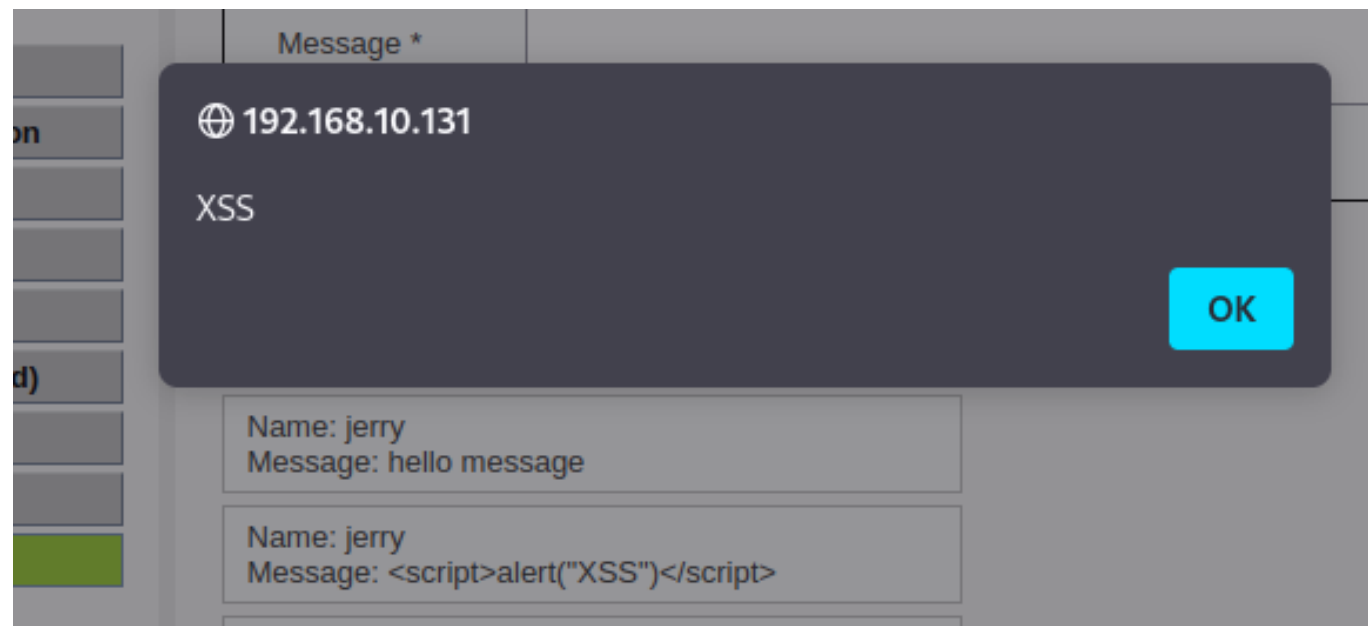
回到Kali輸入與先前相同的注入程式碼

`<script>alert("XSS")</script>`

Vulnerability: Stored Cross Site Scripting (XSS)

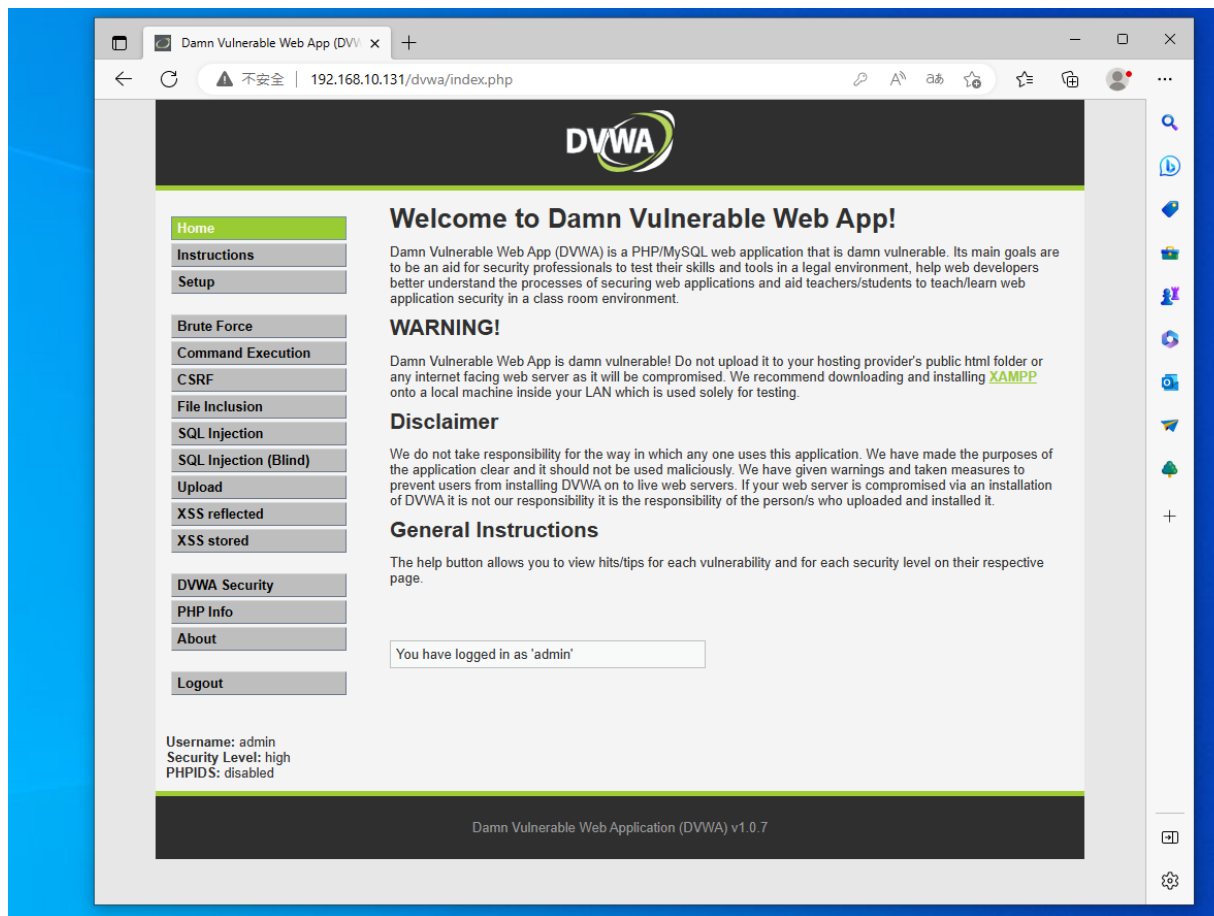
Name *

Message *



可成功注入跳出警告

開啟Windows的DVWA



一樣需要將 Security 調整為 low

DVWA Security

Script Security

Security Level is currently **low**.

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

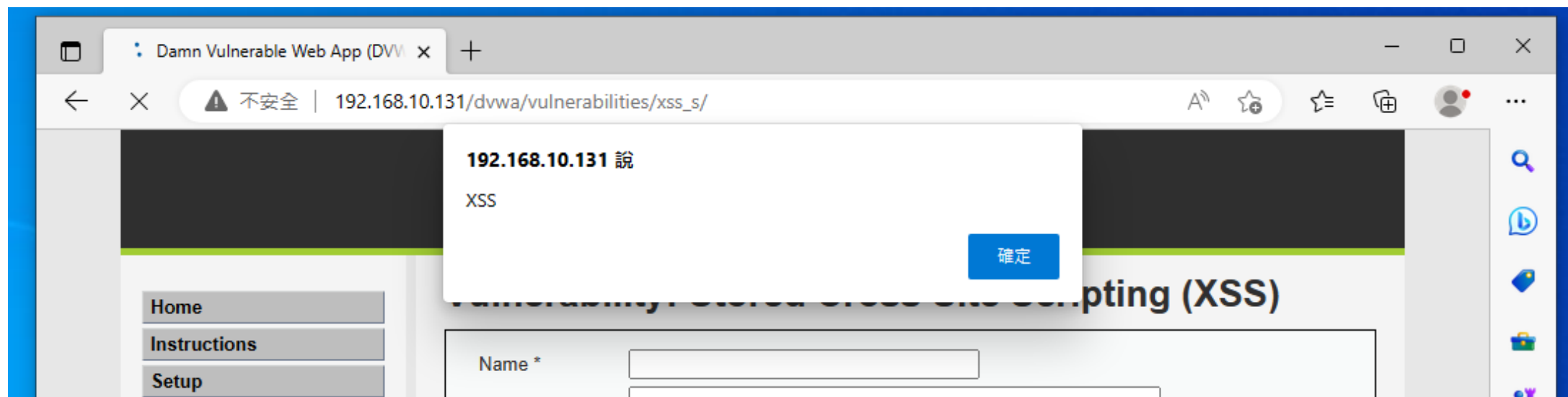
The security level changes the vulnerability level of DVWA.

low



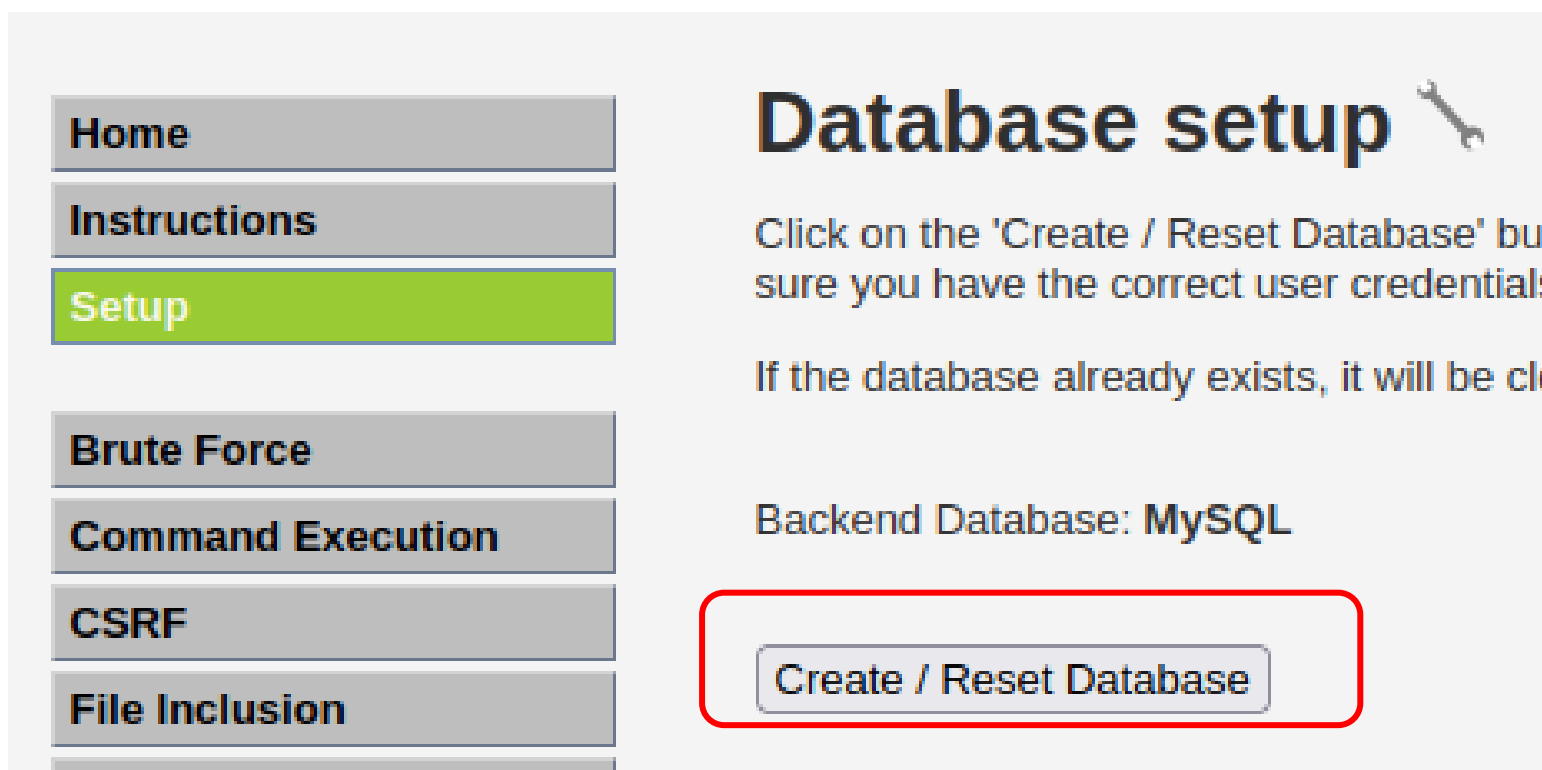
Submit

點選 XSS stored，即跳出先前注入的警告



5.發掘中等的儲存型XSS

點選 Setup > Create / Reset Database 重置 這樣才不會一直跳出先前的XSS



可看到先前的紀錄都刪除了



[Home](#)
[Instructions](#)
[Setup](#)

[Brute Force](#)
[Command Execution](#)
[CSRF](#)
[File Inclusion](#)
[SQL Injection](#)
[SQL Injection \(Blind\)](#)
[Upload](#)
[XSS reflected](#)
[XSS stored](#)

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message: This is a test comment.

More info

<http://hackers.org/xss.html>
http://en.wikipedia.org/wiki/Cross-site_scripting
<http://www.cgisecurity.com/xss-faq.html>

將 Security 調整為 medium

DVWA Security

Script Security

Security Level is currently **medium**.

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

The security level changes the vulnerability level of DVWA.

medium ▼

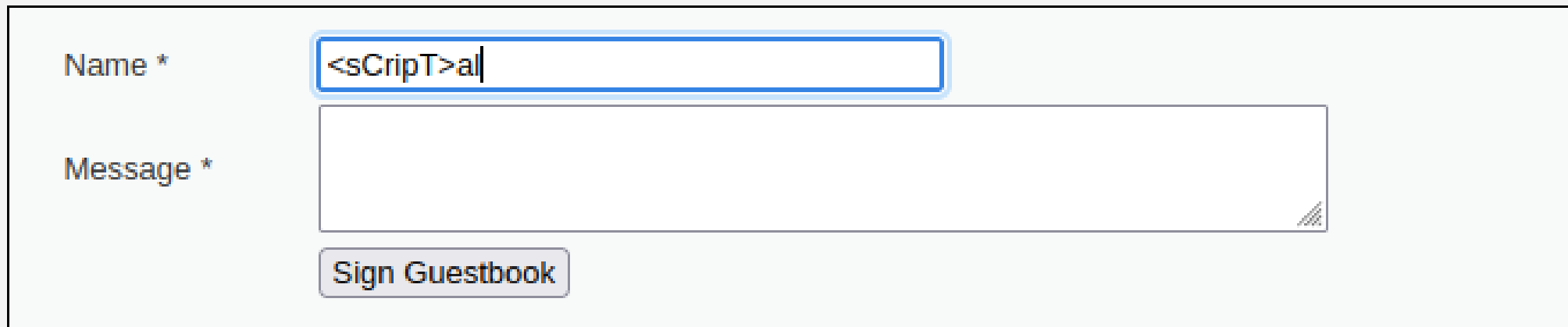
Submit

輸入與先前相同的語法，但有調整大小寫

發現有字數限制

```
<sCripT>alert("XSS")</scRipt>
```

Vulnerability: Stored Cross Site Scripting (XSS)



Name *

Message *

Sign Guestbook

備註: Medium等級的Message欄位無法注入，下一章節會說明，此章節從Name欄位注入

查看程式碼，修改字數

從最多10個字改為最多100個字，修改完畢按鍵盤Enter

```
<td width="100">Name *</td>  
▼ <td>  
  <input name="txtName" type="text" size="30"  
    maxlength="10">  
</td>  
</tr>
```

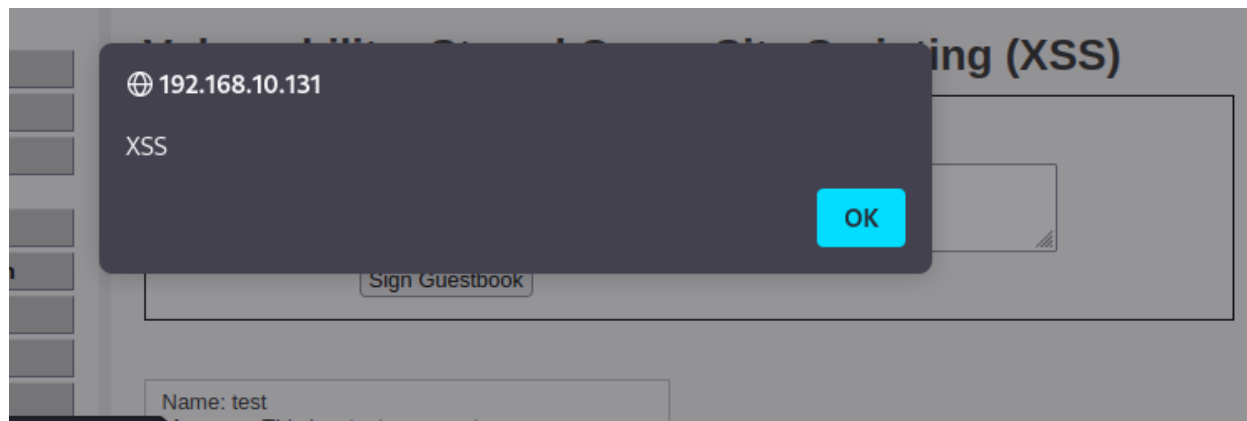
```
▼ <td>  
  <input name="txtName" type="text" size="30"  
    maxlength="100">  
</td>
```

成功注入，跳出警告

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *



換一個方式，時常會遇到限制引號

不用引號注入程式碼：

```
<SCRIPT>alert(String.fromCharCode())</SCRIPT>
```

將輸入的引號與文字改為Charcode

Charcode轉換網站: <https://charcode98.neocities.org/>



The screenshot shows a web browser window with the address bar displaying `charcode98.neocities.org`. The page title is "Uncle Jim's Web Designs". Below the title, the text "Javascript Utilities" and "CharCode Translator" are displayed. The author is listed as "Author: [Jim Stiles](#)". A paragraph explains the functionality: "This is a page that demonstrates the JavaScript `charCodeAt()` and `fromCharCode()` functions that I use. Type character numbers separated by commas in the first box, hit the first button and see the character equivalents in the second. Paste, or type, characters into the third, hit the second button, and see the character number codes in the last."

The interface contains two rows of input fields and buttons:

- Row 1: A text input field containing `85, 110, 99, 108, 101, 32, 74, 105, 109`, a button labeled `fromCharCode()`, and an empty text output field.
- Row 2: A text input field containing `XSS2`, a button labeled `charCodeAt()`, and a text output field containing `88 83 83 50`.

At the bottom, there is a large text area containing the code snippet: `<script language=javascript>eval(String.fromCharCode(PLACE CharCode HERE))</script>`. Below this text area is a link that says "Send this page to someone".

轉換Charcode

轉換前

XSS2

charCodeAt()

轉換後

88 83 83 50

修改後的注入程式碼

不用引號注入程式碼：

```
<SCRIPt>alert(String.fromCharCode(88, 83, 83, 50))</SCRIPt>
```

重新修改字數，並注入新程式碼

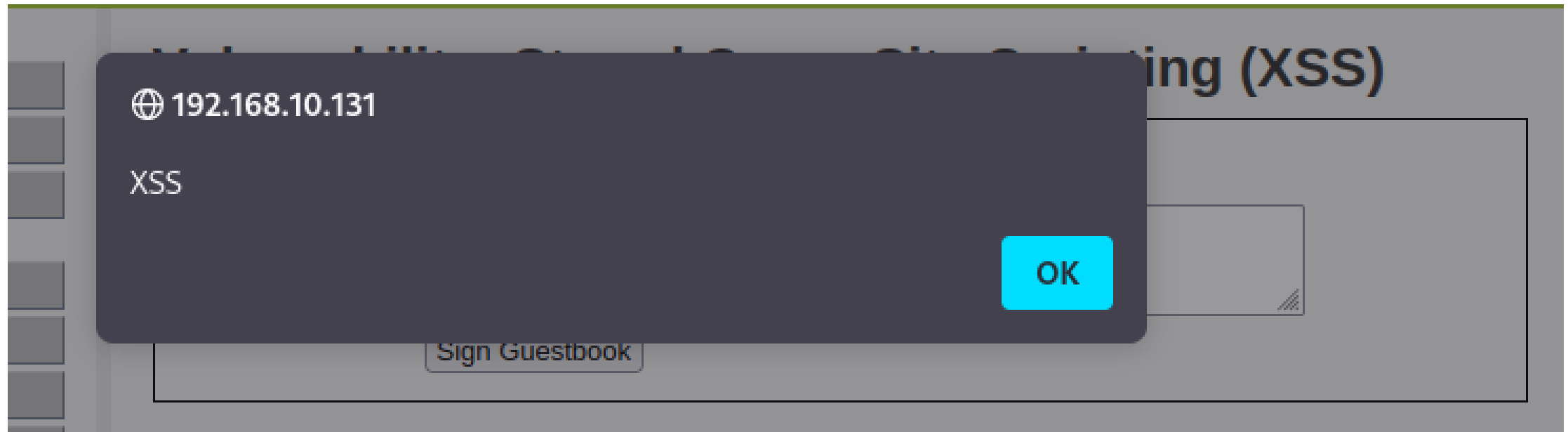
```
▼ <td>  
    <input name="txtName" type="text" size="30"  
        maxlength="100">  
    </td>
```

<SCRIPT>alert(String.fromCharCode(88, 83, 83, 50))</SCRIPT>

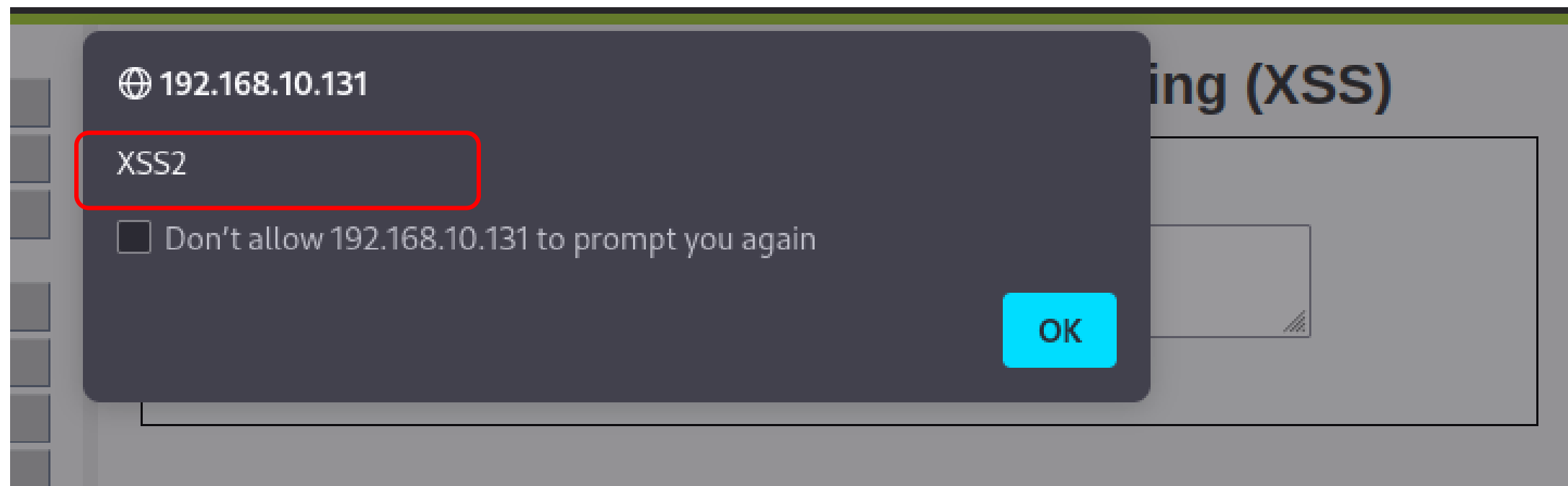
Vulnerability: Stored Cross Site Scripting (XSS)

| | |
|---|---|
| Name * | <input type="text" value="j.fromCharCode(88, 83, 83, 50))</SCRIPT>"/> |
| Message * | <input type="text" value="test2"/> |
| <input type="button" value="Sign Guestbook"/> | |

第一個注入已經存在資料庫所以會先跳出來



使用Charcode成功注入



6. XSS漏洞修復

防範關鍵

- 最小化使用者對HTML的輸入
- 在惡意語法注入到頁面之前，轉換任何惡意的輸入(轉換為HTML的表示方式)

| Char | Result |
|------|--------|
| & | & |
| < | < |
| > | > |
| ' | ' |

Example Code

Javascript:

```
function escapeHtml(unsafe) {  
    return unsafe  
        .replace(/&/g, "&amp;")  
        .replace(/</g, "&lt;")  
        .replace(/>/g, "&gt;")  
        .replace(/"/g, "&quot;")  
        .replace(/'/g, "&#039;");  
}
```

程式碼中，escapeHtml函式會將特殊符號字元跳脫，以防止惡意程式碼的執行。在輸入框中，使用escapeHtml函式將使用者輸入的內容進行跳脫，即可有效防止XSS攻擊。

Example Code

PHP:

```
<?php  
$str = "<script>alert('Hello World!');</script>";  
echo htmlspecialchars($str, ENT_QUOTES, 'UTF-8');  
?>
```

- 程式碼中，會將字串\$str中的特殊字元（如<, >, &, ', "）轉換成HTML實體，以避免JavaScript程式碼被執行。
- 第二個參數ENT_QUOTES表示將單引號和雙引號都轉換成HTML實體。
- 第三個參數'UTF-8'表示使用UTF-8編碼。

End