DASCTF Oct X 吉林工师_Misc_复现

原创

M3ng@L 于 2022-03-02 23:31:43 发布 499 收藏

分类专栏： CTF比赛复现 文章标签： Misc python

DASCTF Oct X 吉林工师_Misc_复现

Perference: (11条消息) DASCTF Oct X 吉林工师 欢迎来到魔法世界～ WriteUp_七月的博客-CSDN博客

WELCOME DASCTFxJlenu

keywords: `python2_input()函数漏洞`

利用python的`input`漏洞，可以直接传入系统命令

```
__import__('os').system('cat flag.txt')
```

如果题目已经引入了os模块那么直接可以

```
os.system('cat flag.txt')
```

giveyourflag

keywords: 压缩包套娃

题目是不断嵌套压缩为zip文件，那么使用zipfile模块进行解压缩

```
import zipfile

f = zipfile.ZipFile("flag1","r")
while True:
    try:
        file_name = f.namelist()[0]
        print(file_name)
        f.extractall()
        f = zipfile.ZipFile(file_name,"r")
    except:
        break
```

如果使用绝对路径会稍微麻烦一些，把解题脚本和zip文件放在同一目录运行就好了

闯入魔塔的魔法少女

keywords: swf反编译

得到swf文件，是flash运行的文件

使用flash反编译软件`JPEXS flash`反编译swf文件

使用软件自带的tools搜索flag转到一段内部程序，在内部程序中继续搜索flag，跳转到flag处

魔法秘文

keywords: ttf文件

压缩包解压得到一张png图像文件，用010editor打开发现有`50 4B 03 04`导出zip文件的十六进制部分

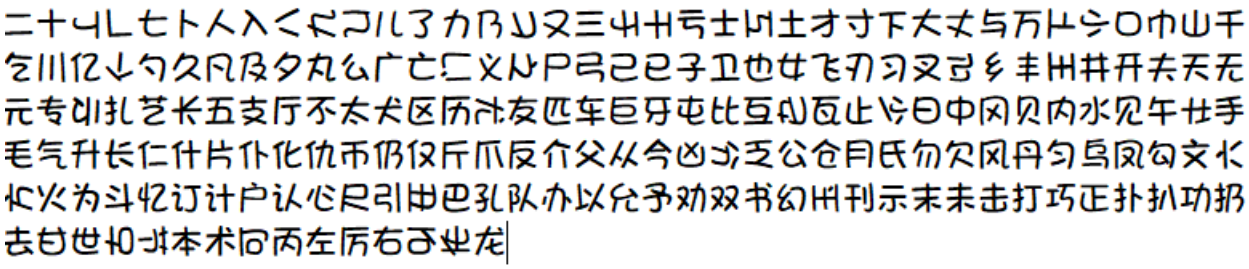
得到一个新的压缩包，内有flag.txt和`魔法萝莉简体`，flag.txt被加密了且压缩包注释提示密码为32个汉字，而后者可以解压出来

用010editor打开，在文件尾找到一串url编码，解码得到一串汉字（200个，显然不是整串汉字）

把`魔法萝莉简体`文件添加文件后缀名`ttf`；`ttf文件`是字体文件，可以复制到系统Fonts文件夹中（相当于安装了字体，之后就可以通过切换字体使用了）

> 文件夹路径 C:\Windows\Fonts

安装好之后使用该字体显示url解码得到的200个汉字



仔细观察会发现其中有些字和平时的不太一样，是倒过来的，把它们挑出来

> 丁厂八九几刀于干工上小个门之马王云木尤切少牛分六方丑玉古节可石布

有一两个字有点刁钻不好认，总而言之凑足32个字就可以解压flag.txt了，得到flag

> DASCTF{4b7e33769d9bd2b7dbc1790ae39397b9}

不可以色色

keywords: `url链接下载隐藏文件`， `mp4文件格式`， `pdf417条形码`

打开是一个网页，表面有个gif文件，另存为下来，把知道的都试了一遍，没有多的条件和提示，只有file format中有

> THEREIS NO FLAG,

回到网页打开网页源码，注释中有一句

> <!–video.?–>

可能有其他文件隐藏了起来，只有后缀不知道，`mp4`和`zip`， `rar`等等都试试，最后找到

> http://dasctf202110-bkyss-1251267611.file.myqcloud.com/video.zip

压缩包中有个mp4文件，解压，但是发现不能正常打开，010editor打开看了看，发现文件头似乎不对，全部充斥着`L`和`FLAG`无关字符

正常的`mp4`文件头格式

> 00 00 00 20 66 74 79 70 69 73 6F 6D

修改文件头使其和正常的文件头一样，就可以打开mp4文件了

在mp4动画中有两帧画面上有类似条形码的图片，截取下来，按照常规样子拼接在一起，使用在线网站扫描

在线读取条形码 (aspose.app)

得到flag

魔法信息

keywords: `010editor的pdf模块pdf.bt`

得到一个流量包，追踪tcp流，总共有20个流，先是找到了几个png文件的十六进制，之后发现第20个流有个zip文件（关键字`PK`）

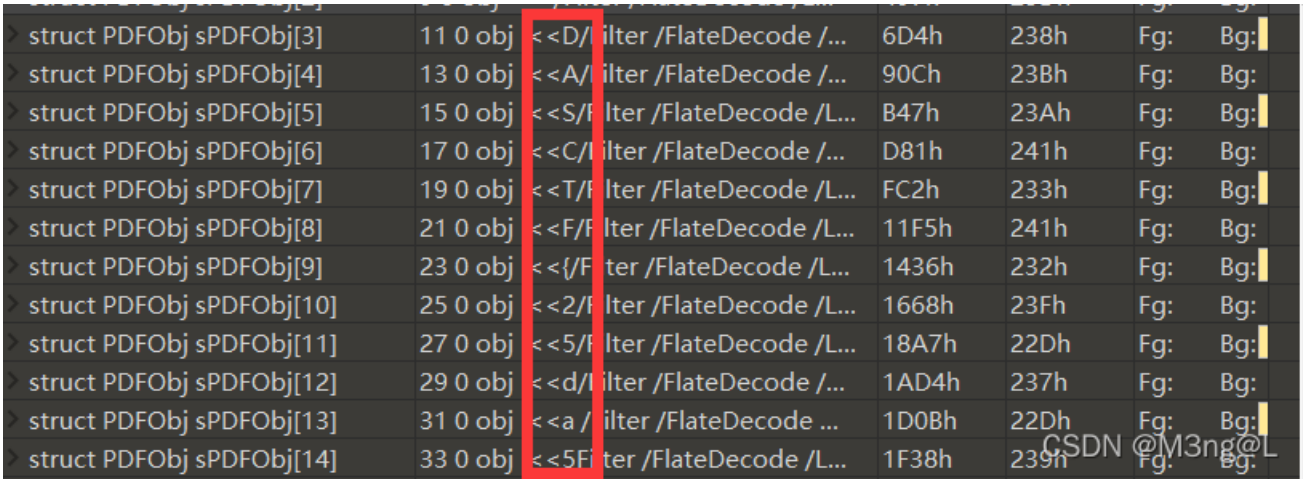
将其转换为原始数据然后导出，粘贴zip文件部分的十六进制到`010editor`的新建十六进制文件

里面有个pdf文件，但是用`Winrar`无法正常解压，猜测可能是zip文件哪里被篡改了，但是找了半天还是没有问题

试着用`7z`直接解压文件（虽然会报错，但是文件确实解压出来了）

无法打开pdf文件，使用010editor看看；

运行模块`pdf.bt`，可以发现



flag藏在了属性名中

英语不好的魔法少女

keywords: 零宽字符加密， 找不同

开始得到一张png的图，在010editor中打开，发现在data部分有很多英文单词

手动复制粘贴到新建文件，发现很多单词前后跟上了一些莫名其妙的乱码

> 类似的：notebooksâ€Œâ€Œâ€Œâ€Œâ€ â€¬â€ ï»¿

后来发现这些乱码就是零宽字符加密的密文（Unicode Steganography with Zero-Width Characters (330k.github.io)）

~~把自己新建的文件丢入零宽字符解密，没有反应~~

重新从图片处分离文件，使用在线分离网站：图片文件夹 (masterqian.github.io)

分离得到`stego_text.txt`，再用ta去零宽字符解密，得到一串

> yjPW8Rlz0og8HX3o6BcwTmveeyyEDiCurJNTwPJeY/PMyOhHXYVKPLIn6isBRyL0

似乎是一串某个加密的密文

再仔细观察之前的英文单词，发现有拼错的地方（对应了题目）

在github上找到常见的英文单词表集，Unicode Steganography with Zero-Width Characters (330k.github.io)

编写脚本判断题目给出的单词是否在正确的单词表中进行对比

```
f = open("right_words.txt","r")
table = []
ori = []
for line in f.readlines():
    table.append(line.strip())
f.close()
f = open("our_words.txt","r",encoding="utf8")
for line in f.readlines():
    ori.append(line.strip())
f.close()
temp = []
for i in ori:
    if i not in table:
        temp.append(i)
        print(i)
```

得到不正确的单词（如果找的单词集不够多的话，也可以进一步手动筛选脚本跑出来的不正确的单词）

注意：这里如果用的是`stego_text.txt`里的单词，那么会因为有零宽字符的密文而导致所谓“不正确”的单词增多（可以自己进一步挑选，也可以直接新建一个文件删除零宽字符的表示）

把不正确单词中不对的字母一一挑选出来，其中有几个单词挑选哪个字母会有点歧义，但是之后都试试就行，把重复的字母删除为一个

> key：mfsvxueshang

丢入AES_ECB解密（没有告诉偏移量，首先默认ECB模式），得到flag

弼弼

keywords: `拼二维码`， `rot47`， `gitee项目`

开始得到一个mp4文件（使用`kinovea`打开需要修改文件名字才能正常打开）

图像中观察到几帧带有二维码的图像，截取下来，但是太模糊了，无法直接扫描

自己手动对应二维码格式新建

QRazyBox - QR Code Analysis and Recovery Toolkit (h3110w0r1d.com)

扫描二维码得到类似于（加上推断）

> snowywar.gitee/gege

这个网址是点不开的，但是后缀名`gitee`搜索之后发现存在一个`gitee`项目，那么在项目内搜索`snowywar`找到 Snow_war雪殇/gege - 码云 - 开源中国 (gitee.com)

里面有两张图与本题的图片风格一致，导出来丢入`stegsolve`分析，

`4444.png`将通道RGB顺序均设置为0，文件头有一个网址 https://jam.wiipedia.org/wiki/%E6%AD%BB

进入网址（VPN没加载出来，没进去）但是很显然是wiki的网站，里面的内容一般不会有用，但是值得注意的是这里wiki搜索的是`死`

> jam.wiipedia.org/wiki/死

`twitter.jpg`在`file format`中看到zip文件的十六进制，导出来，需要密钥

试一试之前wiki中搜索的`死`，成功解密，得到

> =6270yFdE0<?@H0=@G60562C=J0v60v6

`ROT47`解密

> leaf_Ju5t_know_love_dearly_Ge_Ge

加上前缀即为flag

魔法少女的迷音

kewords: `ATOM-128加密`， `音频反向`， `ASCII可见字符范围`

开始得到一个加密了的zip压缩包，打开注释看到（建议看注释使用`Winrar`，因为`7Z`注释中的换行符不会执行）

> nlhtnmTm+m0a+m0a0lA5LIA5LIA5LIA5LIA5LIA5LIA5LIA5L/CC
> atom128

网上搜到`ATOM-128`加密，ATOM-128 / Fast Encryption (Encode or Decode) (persona-shield.com)

解密得到

> passswowoowowddddddddddddddddddddddddd

显然应该就是压缩包的密码了，解压得到一个wav音频文件

打开听一下，全是听不懂的（~~试着看过频谱图，什么也没有，silenteye也什么都没有~~）

最后原来是`音频反向`了，在`audacity`中再进行一次音频反向即可听到正常的音频

音频中是在念数字，记录下来

> 100 51 55 97 51 49 53 54 48 98 100 53 100 53 51 100 50 50 48 99 57 97 52 57 50 102 97 100 53 54 48 49

这里有一点就是，`100 51`很容易觉得是`151`，但实际上151不在ASCII码可见字符范围内，因此推断诸如`151 153`的数字应该都是`100 51 100 53`

转ASCII码（也可以是叫做`From Decimal`）

> d37a31560bd5d53d220c9a492fad5601

加上前缀DASCTF即为flag

卡比卡比卡比

keywords: `内存取证(浏览器记录、cmd记录、提取用户密码)`， `gif查看`

开始得到一个raw内存文件和一个`!@#$unimportance`的文件，里面全是乱码，猜测暂时没有利用价值

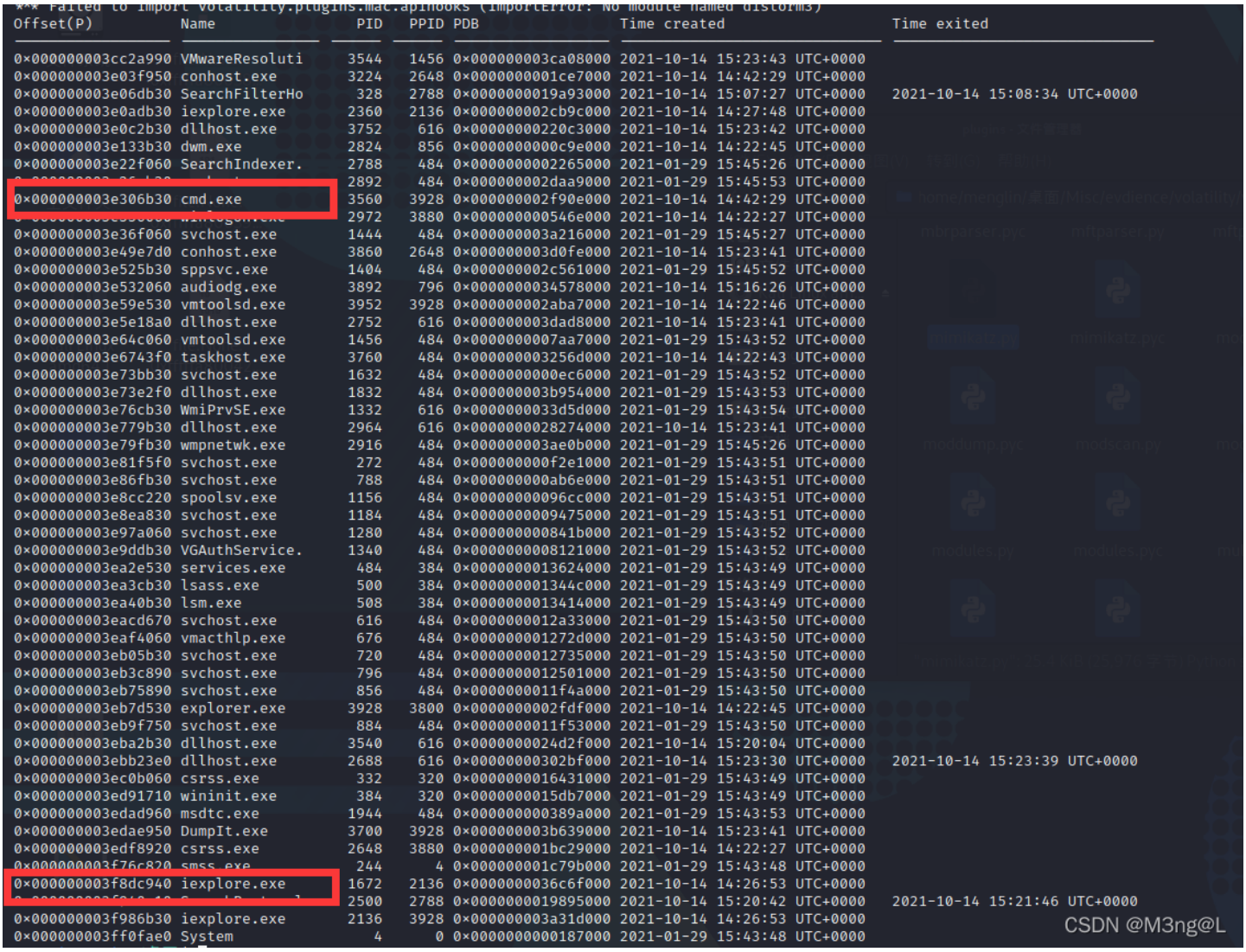
开始对raw文件使用`volatility`进行内存取证（我这里使用的volatility是自己下载的，不是kali自带的，所以是用的`vol.py`来进行取证），取得系统信息

```
python2 vol.py -f 1.raw imageinfo
```

扫描进程

```
python2 vol.py -f 1.raw --profile=Win7SP1x64 psscan
```

发现可疑进程



```
Offset(P)          Name                PID   PPID PDB                Time created                   Time exited
0x000000003cc2a990 VMwareResoluti     3544   1456 0x000000003ca08000 2021-10-14 15:23:43 UTC+0000
0x000000003e03f950 conhost.exe        3224   2648 0x0000000001ce7000 2021-10-14 14:42:29 UTC+0000
0x000000003e06db30 SearchFilterHo      328   2788 0x0000000019a93000 2021-10-14 15:07:27 UTC+0000   2021-10-14 15:08:34 UTC+0000
0x000000003e0adb30 iexplore.exe       2360   2136 0x000000002cb9c000 2021-10-14 14:27:48 UTC+0000
0x000000003e0c2b30 dllhost.exe        3752    616 0x000000000220c3000 2021-10-14 15:23:42 UTC+0000
0x000000003e133b30 dwm.exe            2824    856 0x0000000000c9e000 2021-10-14 14:22:45 UTC+0000
0x000000003e22f060 SearchIndexer.     2788    484 0x0000000002265000 2021-01-29 15:45:26 UTC+0000
                                      2892    484 0x000000002daa9000 2021-01-29 15:45:53 UTC+0000
0x000000003e306b30 cmd.exe            3560   3928 0x000000002f90e000 2021-10-14 14:42:29 UTC+0000
                                      2972   3880 0x000000000546e000 2021-10-14 14:22:27 UTC+0000
0x000000003e36f060 svchost.exe        1444    484 0x000000003a216000 2021-01-29 15:45:27 UTC+0000
0x000000003e49e7d0 conhost.exe        3860   2648 0x000000003d0fe000 2021-10-14 15:23:41 UTC+0000
0x000000003e525b30 sppsvc.exe         1404    484 0x000000002c561000 2021-01-29 15:45:52 UTC+0000
0x000000003e532060 audiodg.exe        3892    796 0x0000000034578000 2021-10-14 15:16:26 UTC+0000
0x000000003e59e530 vmtoolsd.exe       3952   3928 0x000000002aba7000 2021-10-14 14:22:46 UTC+0000
0x000000003e5e18a0 dllhost.exe        2752    616 0x000000003dad8000 2021-10-14 15:23:41 UTC+0000
0x000000003e64c060 vmtoolsd.exe       1456    484 0x0000000007aa7000 2021-01-29 15:43:52 UTC+0000
0x000000003e6743f0 taskhost.exe       3760    484 0x000000003256d000 2021-10-14 14:22:43 UTC+0000
0x000000003e73bb30 svchost.exe        1632    484 0x0000000000ec6000 2021-01-29 15:43:52 UTC+0000
0x000000003e73e2f0 dllhost.exe        1832    484 0x000000003b954000 2021-01-29 15:43:53 UTC+0000
0x000000003e76cb30 WmiPrvSE.exe       1332    616 0x0000000033d5d000 2021-01-29 15:43:54 UTC+0000
0x000000003e779b30 dllhost.exe        2964    616 0x0000000028274000 2021-10-14 15:23:41 UTC+0000
0x000000003e79fb30 wmpnetwk.exe       2916    484 0x000000003ae0b000 2021-01-29 15:45:26 UTC+0000
0x000000003e81f5f0 svchost.exe         272    484 0x000000000f2e1000 2021-01-29 15:43:51 UTC+0000
0x000000003e86fb30 svchost.exe         788    484 0x000000000ab6e000 2021-01-29 15:43:51 UTC+0000
0x000000003e8cc220 spoolsv.exe        1156    484 0x000000000096cc000 2021-01-29 15:43:51 UTC+0000
0x000000003e8ea830 svchost.exe        1184    484 0x0000000009475000 2021-01-29 15:43:51 UTC+0000
0x000000003e97a060 svchost.exe        1280    484 0x000000000841b000 2021-01-29 15:43:52 UTC+0000
0x000000003e9ddb30 VGAuthService.     1340    484 0x0000000008121000 2021-01-29 15:43:52 UTC+0000
0x000000003ea2e530 services.exe        484    384 0x0000000013624000 2021-01-29 15:43:49 UTC+0000
0x000000003ea3cb30 lsass.exe           500    384 0x000000001344c000 2021-01-29 15:43:49 UTC+0000
0x000000003ea40b30 lsm.exe             508    384 0x0000000013414000 2021-01-29 15:43:49 UTC+0000
0x000000003eacd670 svchost.exe         616    484 0x0000000012a33000 2021-01-29 15:43:50 UTC+0000
0x000000003eaf4060 vmacthlp.exe        676    484 0x0000000001272d000 2021-01-29 15:43:50 UTC+0000
0x000000003eb05b30 svchost.exe         720    484 0x0000000012735000 2021-01-29 15:43:50 UTC+0000
0x000000003eb3c890 svchost.exe         796    484 0x0000000012501000 2021-01-29 15:43:50 UTC+0000
0x000000003eb75890 svchost.exe         856    484 0x000000001f4a000 2021-01-29 15:43:50 UTC+0000
0x000000003eb7d530 explorer.exe       3928   3800 0x0000000002fdf000 2021-10-14 14:22:45 UTC+0000
0x000000003eb9f750 svchost.exe         884    484 0x000000011f53000 2021-01-29 15:43:50 UTC+0000
0x000000003eba2b30 dllhost.exe        3540    616 0x0000000024d2f000 2021-10-14 15:20:04 UTC+0000
0x000000003ebb23e0 dllhost.exe        2688    616 0x000000000302bf000 2021-10-14 15:23:30 UTC+0000   2021-10-14 15:23:39 UTC+0000
0x000000003ec0b060 csrss.exe           332    320 0x0000000016431000 2021-01-29 15:43:49 UTC+0000
0x000000003ed91710 wininit.exe         384    320 0x0000000015db7000 2021-01-29 15:43:49 UTC+0000
0x000000003edad960 msdtc.exe          1944    484 0x000000000389a000 2021-01-29 15:43:53 UTC+0000
0x000000003edae950 DumpIt.exe         3700   3928 0x000000003b639000 2021-10-14 15:23:41 UTC+0000
0x000000003edf8920 csrss.exe          2648   3880 0x000000001bc29000 2021-10-14 14:22:27 UTC+0000
0x000000003f76c820 smss.exe            244      4 0x000000001c79b000 2021-01-29 15:43:48 UTC+0000
0x000000003f8dc940 iexplore.exe       1672   2136 0x0000000036c6f000 2021-10-14 14:26:53 UTC+0000
                                      2500   2788 0x0000000019895000 2021-10-14 15:20:42 UTC+0000   2021-10-14 15:21:46 UTC+0000
0x000000003f986b30 iexplore.exe       2136   3928 0x000000003a31d000 2021-10-14 14:26:53 UTC+0000
0x000000003ff0fae0 System                4      0 0x0000000000187000 2021-01-29 15:43:48 UTC+0000
```


分别是cmd命令行和浏览器进程

关于浏览器可以先看看ta的浏览记录（也就是网址）

```
python2 vol.py -f 1.raw --profile=Win7SP1x64 iehistory
```

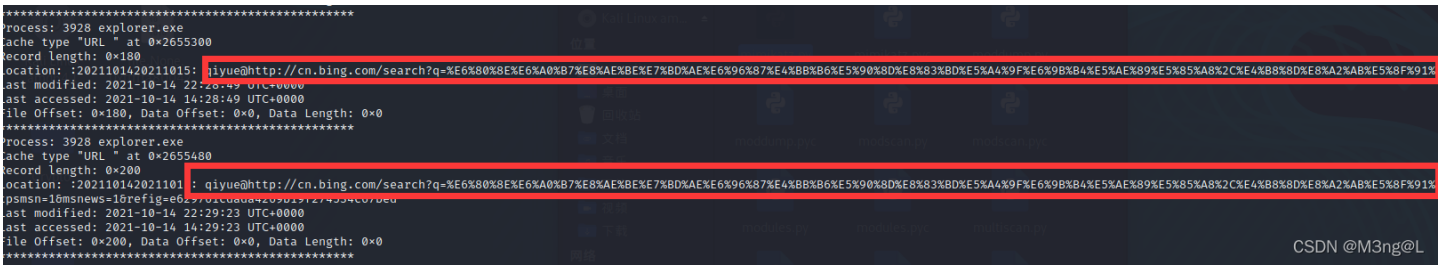
得到的结果太多了，优化一下命令

```
python2 vol.py -f 1.raw --profile=Win7SP1x64 iehistory | more
```

仔细观察有个`key.png`的访问地址是`file:`开头的，说明是本地文件

```
**************************************************
Process: 3928 explorer.exe
Cache type "URL " at 0×2656180
Record length: 0×100
Location: :2021101420211015: qiyue@file:///C:/Program%20Files%20(x86)/MSBuild/key.png
Last modified: 2021-10-14 23:16:45 UTC+0000
Last accessed: 2021-10-14 15:16:45 UTC+0000
File Offset: 0×100, Data Offset: 0×0, Data Length: 0×0
**************************************************
```

还有几个在`bing`上搜索的网址，粘贴下来`url-decode`看一看



> http://cn.bing.com/search?q=怎样设置文件名能够更安全,不被发现&qs=ds&form=QBRE
> http://cn.bing.com/search?q=在文件名前加一个前缀
> &form=PRCNZH&ocid=iehp&httpsmsn=1&msnews=1&refig=e629761cdada4269b19f274534c67bed

其他的地址解密出来就是乱码了

那先把`key.png`提取出来

```
python2 vol.py -f 1.raw --profile=Win7SP1x64 filescan | grep key.png
```

找到对应的`PID`

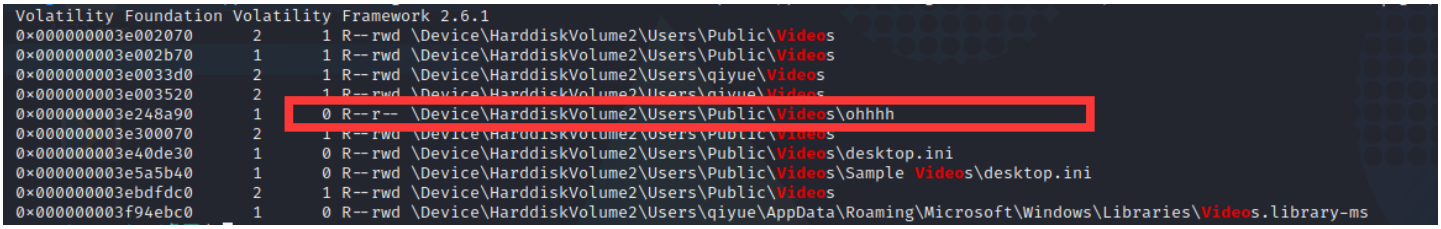
```
python2 vol.py -f 1.raw --profile=Win7SP1x64 dumpfiles -Q 0x000000003e5e94c0 -D ./
```

保存下来，把后缀换成png但是却无法显示，使用`010editor`打开看看，发现根本没有png文件头，只有很短的一串乱码，猜测是`文本`，把后缀名改成`.txt`试试，得到

> 我记得我存了一个非常棒的视频，但怎么找不到了，会不会在默认文件夹下

这样指向性就很强了，去搜索一下`Video`文件，而视频的默认文件夹也就是`Video`（注意时大写字母开头，这里`grep`对大小写敏感）

```
python2 vol.py -f 1.raw --profile=Win7SP1x64 filescan | grep Video
```



得到多个文件，最可疑的是`ohhhh`，`dump`下来

```
python2 vol.py -f 1.raw --profile=Win7SP1x64 dumpfiles -Q 0x000000003e248a90 -D ./
```

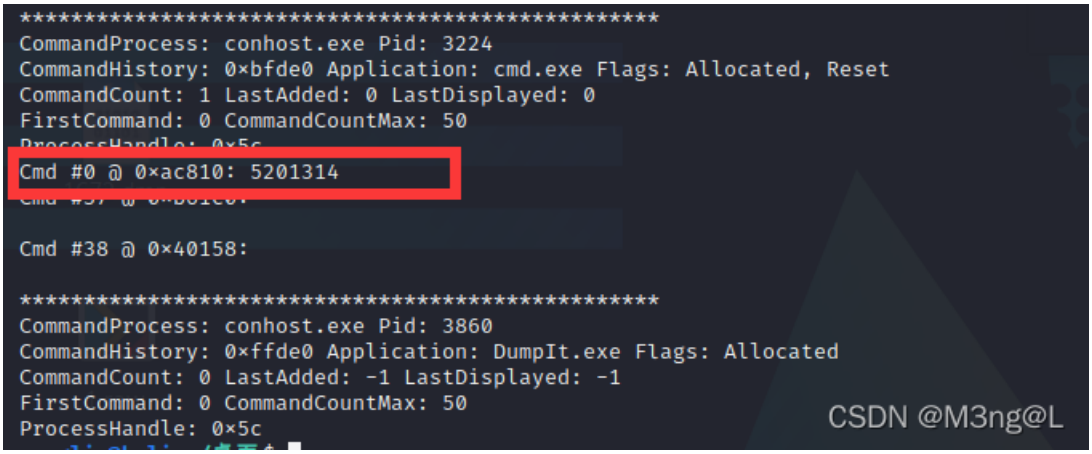
改后缀名为`mp4`发现，又不能打开，`010editor`再次打开，这次直接可以看到

> xzkbyyds!

到这里，关于`浏览器记录`的部分找的差不多了；找一下之前的`cmd`进程

```
python2 vol.py 1.raw --profile=Win7SP1x64 cmdscan
```

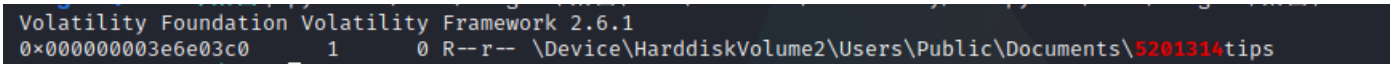
得到



关于这串数字感觉想是密钥，但是这里暂时还没有需要密钥的地方

把到此为止的得到所有信息都在内存中搜索一下，发现

```
python2 vol.py 1.raw --profile=Win7SP1x64 filescan | grep 5201314
```



`dump`下来

```
python2 vol.py 1.raw --profile=Win7SP1x64 dumpfiles -Q 0x000000003e6e03c0 -D ./
```

在`010editor`中发现是zip文件头，加后缀

解压需要密钥，把目前为止知道的信息都试一遍，过不了

重新回到内存取证中寻找与密钥有关的东西，最后找到是管理员`qiyue`的`登陆密码`

```
python2 vol.py 1.raw --profile=Win7SP1x64 mimikatz
```

```
WARNING : Volatility.debug    : NoneObject as string: Invalid offset 2518040 for dereferen
Module   User             Domain           Password
-------  ---------------  ---------------  ----------------------------------------------
wdigest  qiyue            qiyue-PC         MahouShoujoYyds
wdigest  QIYUE-PC$        WORKGROUP
```

关于`volatility`的插件`mimikatz`一般需要手动安装，详情可见 volatility安装插件(以mimikatz为例子)_ruokeqx-CSDN博客_volatility 安装插件，安装好之后就可以使用以上命令而不会报错了

使用管理员登录密码去解压之前得到的zip文件，得到文件`exp`

```
import struct
key = 'xxxxxxxxx'
fp = open('!@#$importance', 'rb')
fs = open('!@#$unimportance', 'wb')
data = fp.read()
for i in range(0, len(data)):
    result = struct.pack('B', data[i] ^ ord(*key[i % len(key)]))
    fs.write(result)
fp.close()
fs.close()
```

这里就和题目最先给出的`乱码文件`联系在了一起，而这个`exp`脚本就是解密用的（说白了就是在异或一次即可）

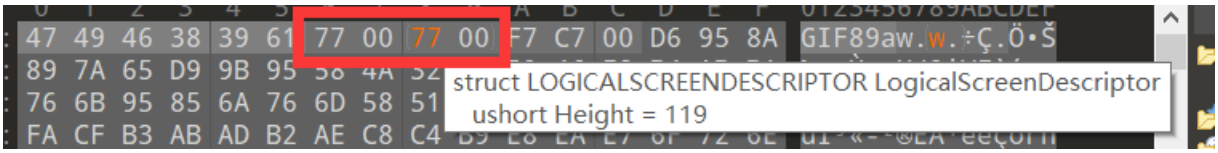
需要知道`key`，猜测之前得到的`xzkbyyds!`是`key`，因为字符串长度一致，稍微修改一下`exp`文件（首先修改成`exp.py`）

```
import struct
key = 'xzkbyyds!'
fp = open('C:\\Users\\Menglin\\Desktop\\!@#$importance', 'wb')
fs = open('C:\\Users\\Menglin\\Desktop\\!@#$unimportance', 'rb')
# 不知道为什么我这里单独使用open()函数相对路径不生效...所以用的绝对路径
data = fs.read()
for i in range(0, len(data)):
    result = struct.pack('B', data[i] ^ ord(*key[i % len(key)]))
    fp.write(result)
fp.close()
fs.close()
```

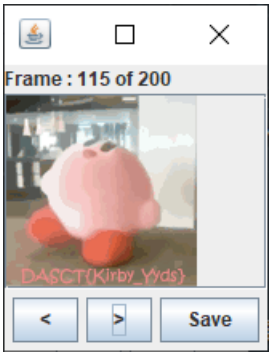
得到`!@#$importance`文件中有`GIF`文件头，修改后缀名为`.gif`

终于看到了卡比

仔细观察在不点开的时候右下方有个小图标，但是使用系统自带的`图片`点开之后没有了，似乎被截掉了一部分，猜测是修改了gif的高度；由于肉眼看起来像是正方形的gif文件，那就把高宽改成一致



`stegsolve`使用一帧一帧地看gif文件

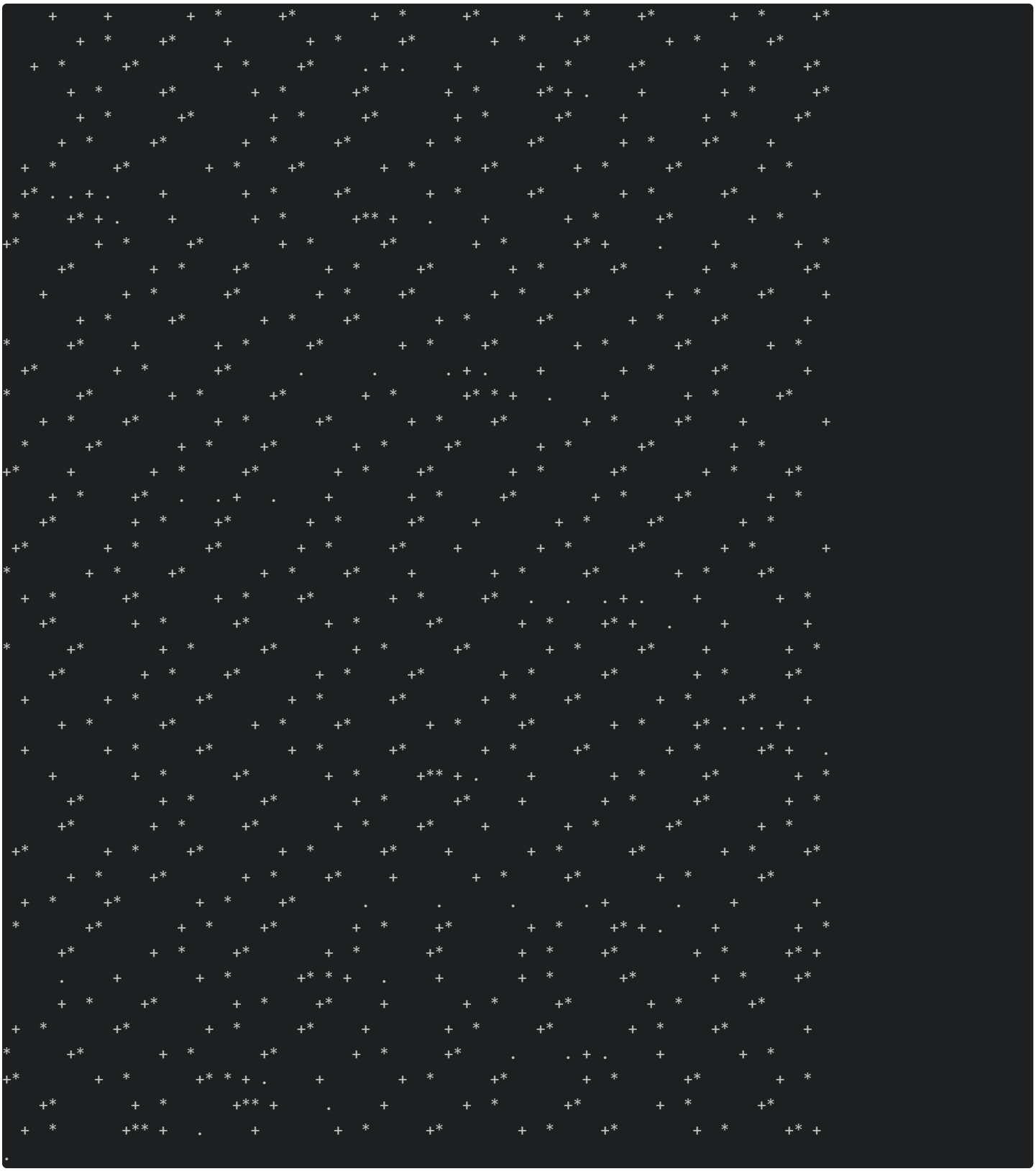


flag即在图片下方

Twinkle Twinkle Starry Night

keywords: `starry语言`， `栈指令`，

使用nc连接，反馈一长串base64编码，解密得到



是`starry语言`，(11条消息) 搞怪语言——Starry语言简介_Script Ahead, Code Behind-CSDN博客，是从栈指令转换而来的语言

网上没有找到现成的转换脚本，可以根据下面这幅图来写脚本转换

```
dup       : " +"
swap      : "  +"
rotate    : "   +"
pop       : "    +"
push      : 5个或以上个空格，后面接一个"+"；空格数量减去5就是push的参数值

+         : "*"
-         : " *"
*         : "  *"
/         : "   *"
%         : "    *"

num_out   : "."
char_out  : " ."

num_in    : ","
char_in   : " ,"

label     : 任意多个空格之后接一个"`"；空格的个数是该标签的标识符
jump      : 任意多个空格之后接一个"'"；空格的个数是跳转目标标签的标识符
```


转换脚本

```python
f = open("cipher.txt","r")
cipher = f.read()
f.close()
f = open("plaint.txt","w+")
# print(cipher)
space = 0
for cha in cipher:
    if cha == "\n":
        continue
    elif cha == " ":
        space = space + 1
        continue
    elif cha == "+":
        if space == 1:
            f.write("dup\n")
            space = 0
        if space == 2:
            f.write("swap\n")
            space = 0
        if space == 3:
            f.write("rotate\n")
            space = 0
        if space == 4:
            f.write("pop\n")
            space = 0
        if space >= 5:
```

```
            temp = space - 5
            f.write("push " + str(temp))
            f.write("\n")
            space = 0
        continue
    elif cha == "*":
        if space == 0:
            f.write("+\n")
            space = 0
        if space == 1:
            f.write("-\n")
            space = 0
        if space == 2:
            f.write("*\n")
            space = 0
        if space == 3:
            f.write("/\n")
            space = 0
        if space == 4:
            f.write("%\n")
            space = 0
        continue
    elif cha == ".":
        if space == 0:
            f.write("num_out\n")
            space = 0
        else:
            f.write("char_out\n")
            space = 0
        continue
    else:
        f.write("error!\n")
        break
# 由于很多其他的栈指令是没有出现的，这里就没有写相应的转换了
```

得到一长串栈指令；形如

```
push 0
push 0
push 3
*
push 1
+
push 3
*
push 0
+
push 3
*
push 2
+
push 3
*
```

这样的指令，大概看了看，有`*`，`+`，`-`，`push`，`dup`，`char_out`

在栈中的意义是（详情可以见栈与四则运算 - CoderLcp - 博客园 (cnblogs.com)，(16条消息) java虚拟机指令dup的理解_Gabriel576282253的专栏-CSDN博客_dup指令）

- `*`

使栈最顶端的两个数值相乘，得到的结果覆盖这两个数值（也就是两个地址变成了一个）

- `+`

使栈最顶端的两个数值相加，得到的结果覆盖这两个数值

- `-`

使栈最顶端的两个数值相减（具体是栈从顶端第二个减去第一个），得到的结果覆盖这两个数值

- `push`

压栈，从栈顶加入新的数值

- `dup`

将处于栈顶的那一个数值复制，再进行压栈

- `char_out`

提取栈顶的数值（这里也就是`print`了），并且除去ta在栈中的存在

那么编写一个脚本把栈指令的操作表现出来

```
f = open("plaint.txt","r")
data = []
flag = []
for line in f.readlines():
    line = line.strip('\n')
    # print(line)
    if "push" in line:
        number = int(line.split(" ")[1])
        data.append(number)
    elif "*" in line:
        x = data[-1]
        y = data[-2]
        number = x * y
        data = data[:-2]
        data.append(number)
    elif "+" in line:
        x = data[-1]
        y = data[-2]
        number = x + y
        data = data[:-2]
        data.append(number)
    elif "-" in line:
        x = data[-1]
        y = data[-2]
        number = y - x
        # if number < 0:
        #     number = -number
        data = data[:-2]
        data.append(number)
    elif "dup" in line:
        x = data[-1]
        data.append(x)
    elif "char_out" in line:
        number = data[-1]
        print(str(number),end=" ")
        flag.append(number)
        data = data[:-1]
    else:
        print("eorro")
        break
print()
for i in flag:
    print(chr(i),end="")
```

得到flag（注意这里是动态flag）