

AWFUL

题外话：这一周下来收获挺多的，所以想把这一周写题目整个测试过程都尽量写下来，由于中间的很多测试都是失败，所以可能废话会很多，同时也是第一次 WP，尽情见谅。

Web

分数 50: Are you from Europe?

URL: <http://123.206.203.108:10001/European.html>



抽到的。。。。。

起初第一眼看到标题抽到 SSR 就算过关，没有多想，直接开干，手动点了 50+ 没有结果。

后来尝试其他方法，看到是 html 网页即静态网页，查看源代码，由于对 html 框架不熟悉。直接在上面进行修改没有结果，无奈之下转到抓包方面，结果在测试图中抽到了 SSR。拿到 FLAG。

flag: hgame{Th3_Ch0seN_0nE!}

运气挺好的，另外我也希望看看这道题真正是怎么解决的。

分数 100:special number

URL: <http://118.25.18.223:10001>

hint: PHP 弱类型

打开只有 php 代码，典型的 php 代码审计代码题目，这里我放在 sublime 里面方便分析

```
1  include_once("flag.php");
2  if(isset($_GET['key'])){
3      $pattern = '/^(?=.*[0-9].*)(?=.*[a-zA-Z].*){7,}$/';
4      $key = $_GET['key'];
5      if(preg_match($pattern,$key)==0){
6          echo "格式错误";
7      }else{
8          $lock="*****";
9          $b = json_decode($key);
10         if($b== $lock)
11             echo $flag;
12         else
13             echo "this is no special number";
14     }
15 }
```

① `$pattern = '/^(?=.*[0-9].*)(?=.*[a-zA-Z].*){7,}$/';`

这一行是正则表达式，由于我对正则表达式不是特别熟悉，查询了很多资料，后面会给出，这里给出我的分析：必须带数字和字母而且总长度大于等于 7。

② `$_GET['key']` 为获取 URL 参数 "key" 的内容

③ `json_decode` 是一种解法方式

综上所述，这一整个 php 代码要求我们所需要的就是，输入 <http://118.25.18.223:10001/?key=XXXXX>，而这个 XXXXX 是我们需要输入的内容，这个内容必须带数字和字母而且总长度大于等于 7，同时这个内容在经过 json_decode 处理之后 == 某一个字符串成立（即 \$_GET['key']==\$lock 成立），就可以拿到 flag

接下来我们来看看 hint

- ① php 弱类型：废话，这道题目只有三个坎，一个看懂正则表达式；一个知道 json_decode 具体是怎么运作的（但是做完之后实际上这个也不需要知道）；最后一个就是 php 弱类型。不解释
- ② 科学记数法：稍微明显了点，当初做完这道题时这个还没出这个 hint，由于当初我并没有注意到题目标题 special number 的意义所在，所以一直再无意义的尝试，等下会进行讲解。

这里我使用的是倒退的思想

```
if($b==$lock)
```

```
    echo $flag;
```

先看看 php 弱类型，什么东西 == 任意字符串是成立的，给一下几个进行参考下

```
0=="*****"
```

```
"0000000000"=="*****"
```

```
"0admin"=="*****"
```

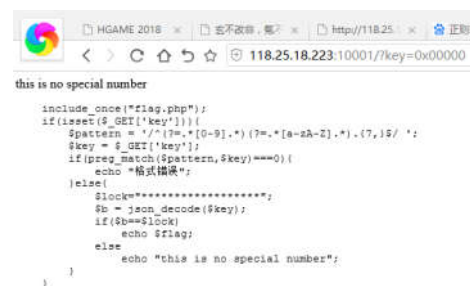
接下来是 js_decode，由于当时没有注意到标题名字的意义，我尝试很多种方法

1、输入数组，这是 js_decode 原本的用处（我看网上的例子基本都是数组，这里如果说错了请指正），最后放弃是因为数组 == 一个字符串是永远不成立的

2、故意输错，即利用 js_decode 无法处理一些特定类型，导致它报错并返回 NULL，这是一种常用的方法，最后放弃是因为我发现自己记混了弱类型，实际上 NULL == "" 成立但是 NULL == "任意有内容的字符串" 是错的

3、经过两种尝试了解到必须输入一个 js_decode 能处理的类型，在本地测试中，发现数字和 boolean 类型他可以处理，表明有两个切入口。之前说过 true 类型 == 任意字符串，但是由于到达不到长度 7 且没有数字最终放弃（说的这么简单，但是实际上我还是傻傻的试了十几分钟），最后只有数字。

怎样才能输入一个带有字母且长度为 7 的数字呢？第一个想到的是十六进制，那是特别高兴，本地测试也是对的，然而结果。。。



心态直接崩了!!! 我试了两三小时，你居然给我看这个？

冷静下来，我第一个想到是不是出题者出的小把戏，php 弱类型中 "1admin"==1 是成立的，会不会那一串 "*****" 中开头是一个数字呢，我继续尝试了 0x0000001 到 0x0000009，都不对，最终放弃了。

最后我又想到了科学记数法，试了下居然成功了。。。



题外话：我已经不知道多少次因为忽视题目名被坑了很久，然而真正做题目的时候像我这种悟性不高的基本不会一点就通的。
测试过程我也发现了很多知识点所以全都写下来，可能会很长，希望见解，科学记数法我最后没写完，可以去查一查。

参考文章：

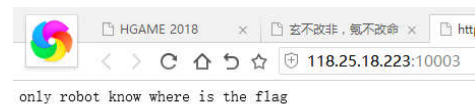
<https://zhidao.baidu.com/question/808516916006224252.html>

<http://www.jb51.net/tools/regexcsc.htm>

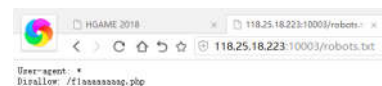
<http://www.jb51.net/article/36172.htm>

分数 100: can u find me?

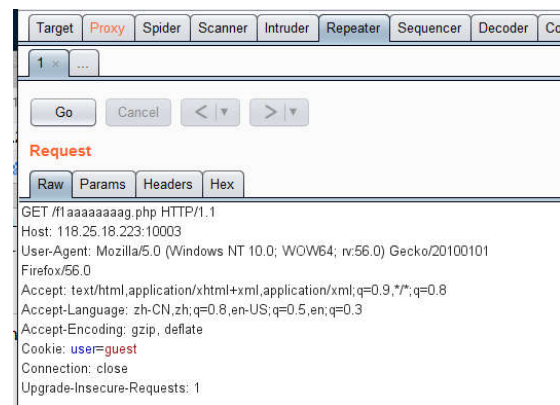
URL: <http://123.206.203.108:10001/European.html>



只有这么一点，源代码没有任何发现，抓包也是如此，之后查阅资料得知了 **robot.txt** 这种路径，尝试了下



很明显了，需要手动抓包修改



包的内容如下，可以明显看到 **cookie: user=guest**

将其进行修改就拿到 **flag** 了



参考文章

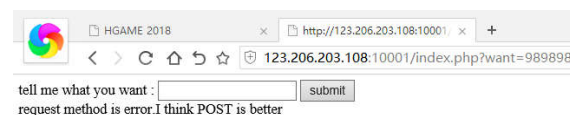
<http://blog.csdn.net/u012763794/article/details/50959166>

分数 100: 想要 **flag** 吗?

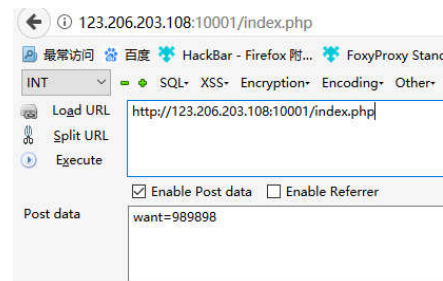
URL: <http://123.206.203.108:10001/>



随便输入，跳出提示



提示挺明显的，需要我们 **post** 数据（然而实际上我自己写的时候是用抓包进行发送，却也成功了）



tell me what you want : submit
https://www.wikiwand.com/en/X-Forwarded-For
only localhost can get flag

居然还给网站了，挺良心的，可惜我知道这种方法。进行抓包

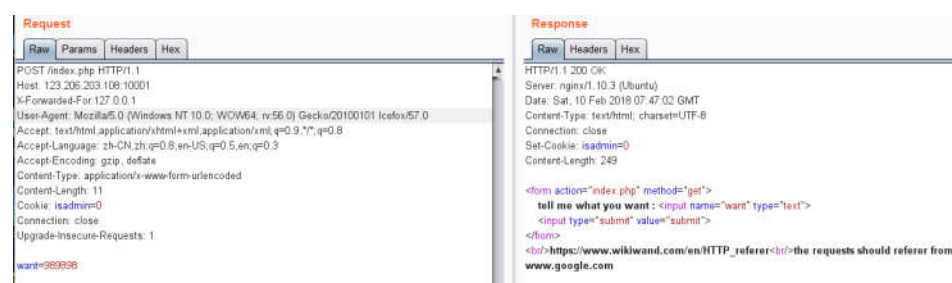


在任意位置添加 X-Forwarded-For:127.0.0.1

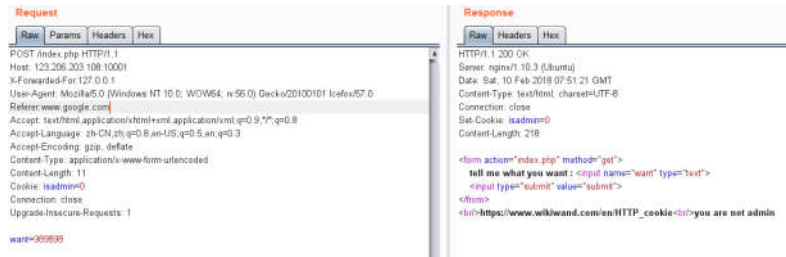
127.0.0.1 为 localhost



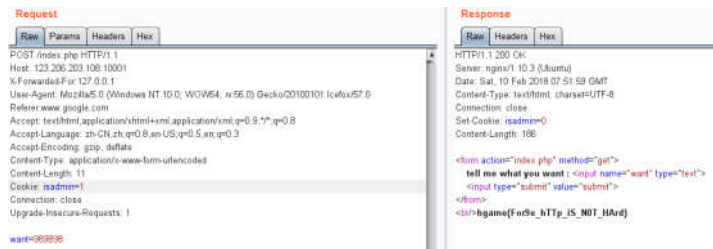
要求我们用 icefox/57.0，写过类似的同学应该知道要继续修改，对于 user-agent 进行修改



这里其实我卡了一段时间，因为之前在研究伪装 **Ip** 的时候，查询如何修改 **referer**，网站给的全是脚本，而且还不说怎么使用。这里我又查了很多资料没有结果，最后我发现只需要在包里面添加 **Referer** 一行就可以了（这题也算解决了我之前留下来的疑问了），修改后继续



将 isadmin 的数值修改



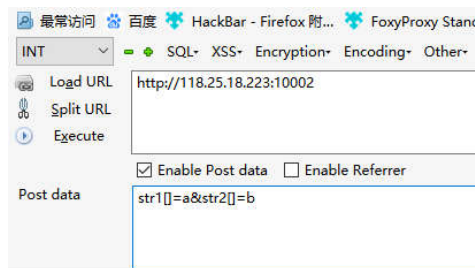
拿到 flag

分数 50: 我们不一样

URL: <http://118.25.18.223:10002/>



又是 php 代码审计，我先说我当时的解题方法，这题目我第一直接就是输入数组报错，灵感来源是之前写过 md5 和 sha1 碰撞类似的题目。注意这里代码中是 post 方式。



flag is:hgame{g3t_f14g_is_so0000_ez}

5 分钟都不到，这题也是我在写题目时产生信心的一道题目。

当然现在回来写 WP 时我还是要详细解释这个原理，毕竟以前写题目不懂看别人 flag 时，那种作者一笔带过，最后就贴了一个 python 代码的心情我能理解。

- 1、首先 `$_post['str1']` 的意思是提取出在网页提交的时候以 post 方式提交的参数名字为 str1 的内容
- 2、`strcmp($_POST['str1'],$_POST['str2'])` 比较 str1 和 str2 字符串内容，如果相同返回 0，（想要具体了解这个函数，可以看看文章第二个）

在解释完这两个之后来解释原理：

在提交数据的时候，`$_POST['str1']` 和 `$_POST['str2']` 都是数组类型的，数组内容分别为 a 和 b，所以他们两个自然是不相同的，

但是当他们经过 `strcmp` 函数处理的时候，由于 `strcmp` 无法处理数组类型导致它报错返回 `NULL`，（之前我看到一个 WP 上面写的是报错返回 `false`，但是实际我测试过来为 `NULL`）而 `NULL==0` 是成立的，所以拿到 flag

参考文章

<https://www.cnblogs.com/zhanqiaomiao/p/6013475.html>

http://www.w3school.com.cn/php/func_string_strcmp.asp

re

说明 re 之前，由于去年初学渗透一段时间，使用的是当年教材使用的软件，因此这里说明一下，使用的软件为 **OD 1.0 版本**（好像是这个版本，反正是 OD 的第一个版本），插件方面应该没有影响，因为写出来的题目没有壳和反调试函数，用 **PEID 测试** 是否有壳我就没有贴出了

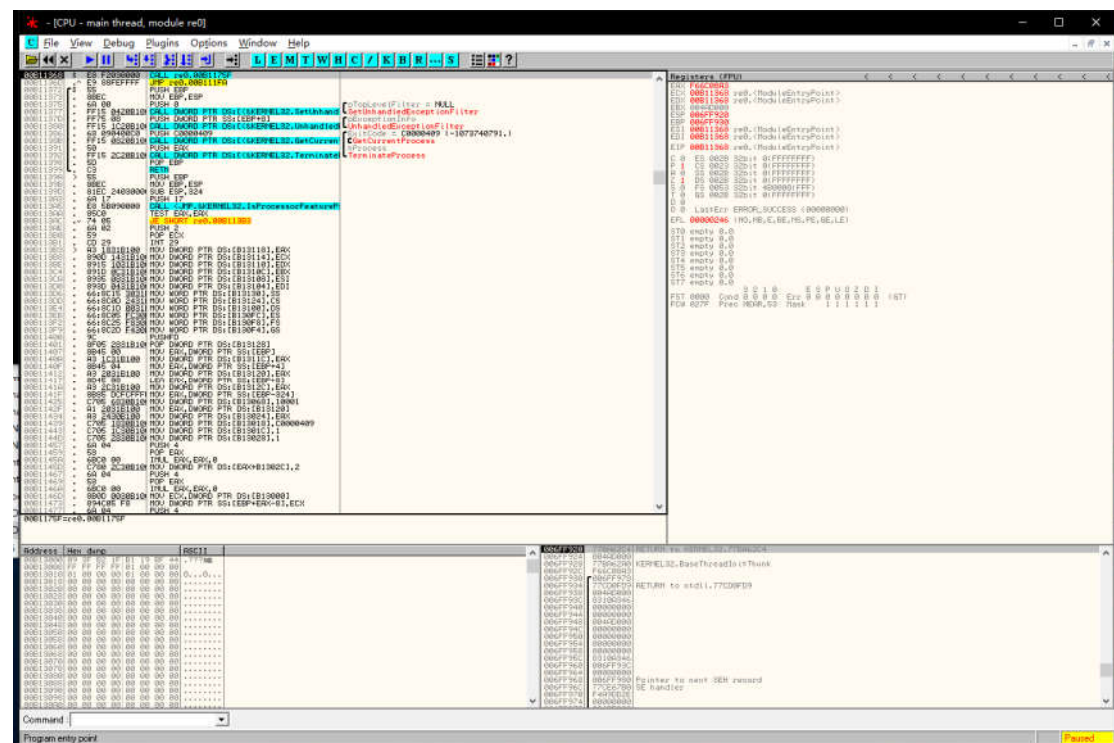
分数 50: re0

URL: <http://ol795rwtm.bkt.clouddn.com/re0.exe>

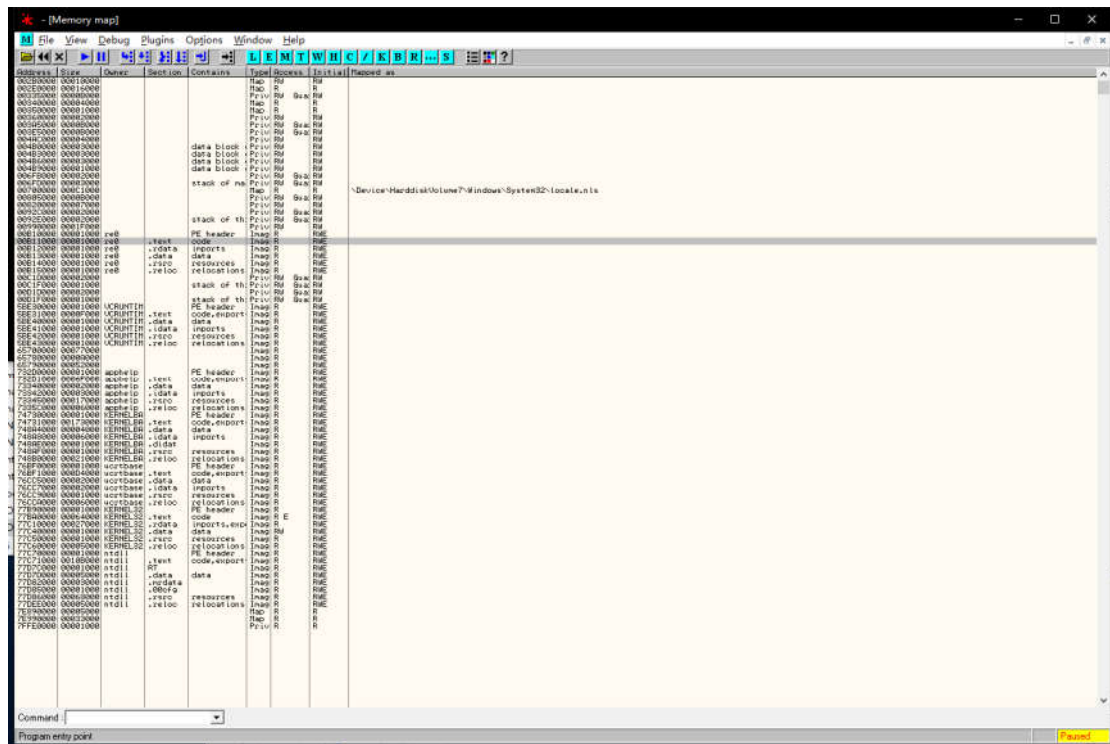
打开后界面为一个 cmd 窗口

```
Welcome to hgame
Input your flag:
```

进行调试

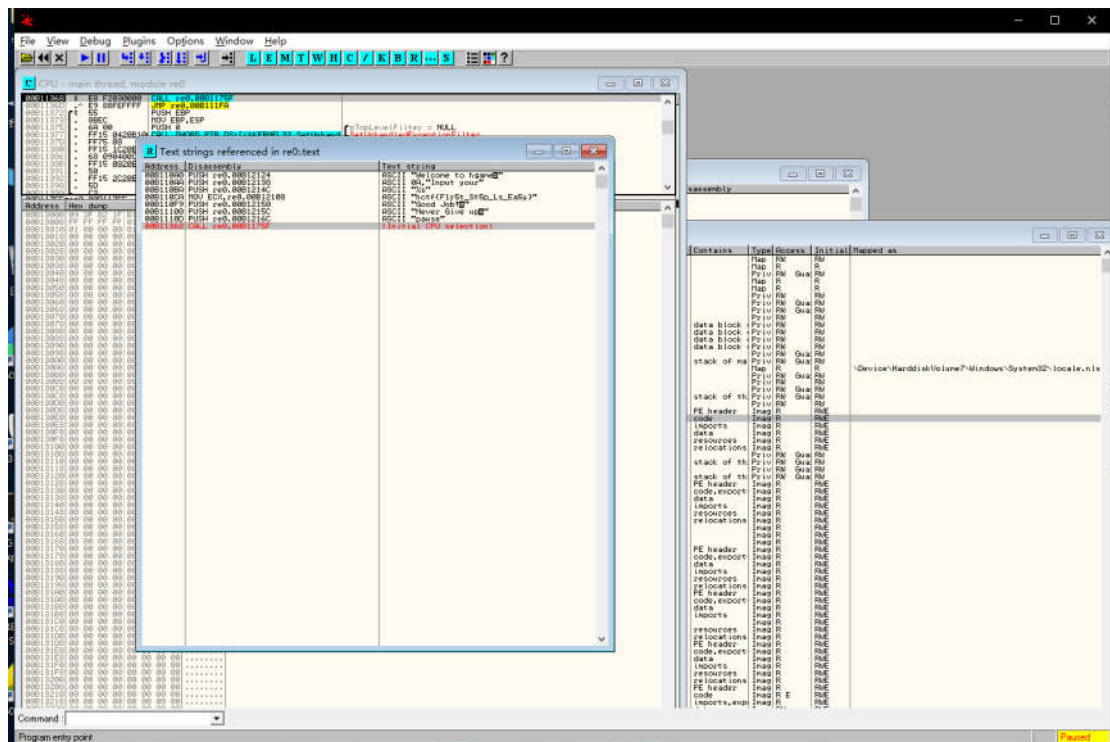


点击 Memey map（就是那个上面大写的 M）



如图，点击的位置可以确认为目前 EIP 到达的位置是主程序位置

右键--search for--all referenced text strings



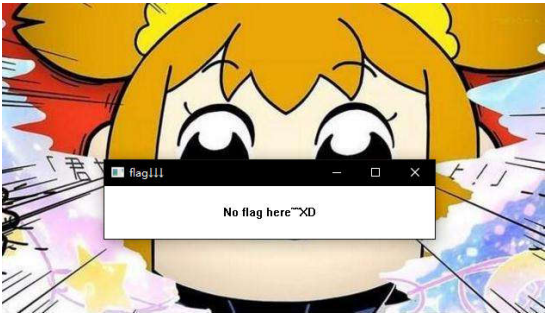
程序都还没运行就得到 flag...

分数 100: nop_pop

URL: <http://oi795rwtm.bkt.clouddn.com/re0.exe>

题外话：这程序在写的时候不知道被 360 误删了多少次...，尝试在虚拟机运行，结果报错：不是有效的 32 位程序，无奈只能

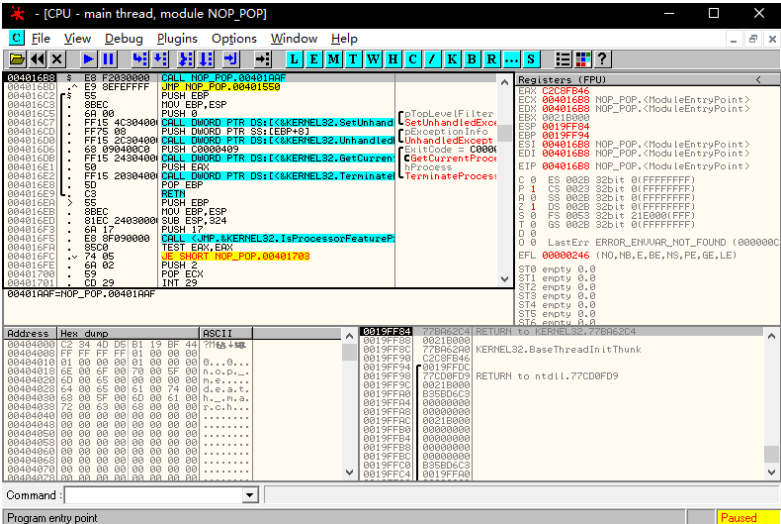
在 win10 调试
运行界面



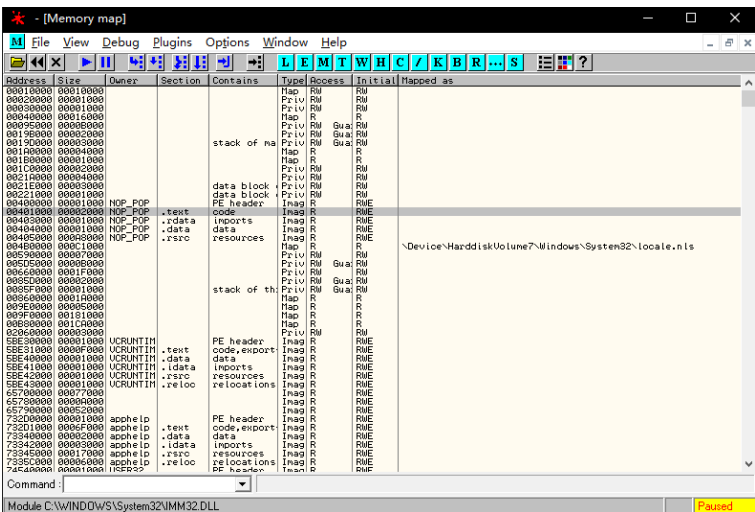
由于这是我做的最后一道题，所以有看到 hint

描述
听说把pop子的窗口去掉就能找到flag了
知识点: winrar去广告窗，了解一下

既然都说 winrar 去广告，就去搜呗（后面会给出参考文章）在知道方法后，进行调试

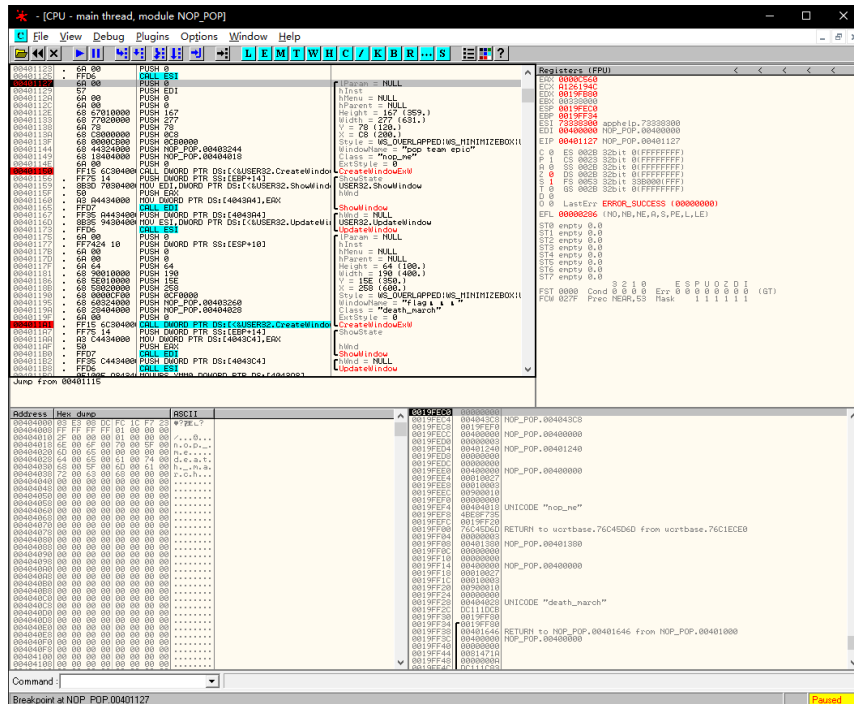


点击上面的 M

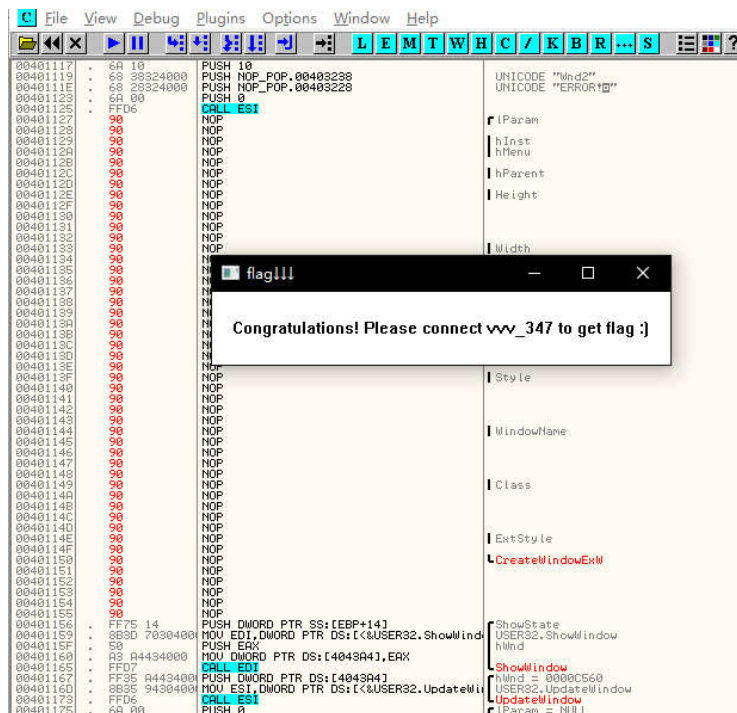


确定停在了处在主程序中
右键—search for—name in current module 查看调用函数

再次运行后



这里我们需要把 401127 到 401150 全部 nop 掉，所以右键—binary—fill with nops



到了这一步，当初以为一个坑（后面题目中你们就会遇到），想了一段时间，但是后来想起来在群里面看到也有人遇到这个问题，结果真要去群里面联系一个叫 vvv_347 的同学拿 flag。。。之后联系他，他说要把 patch 过的文件发给他，这里也说下怎么保存
右键—copy to executable—all modifications—copy all，之后点击跳出来的窗口，右键—save file 就好了

hctf{Far5we1L_G0od_Cr4cker}

参考文章

http://www.360doc.com/content/16/11/15/19/28736017_606814895.shtml

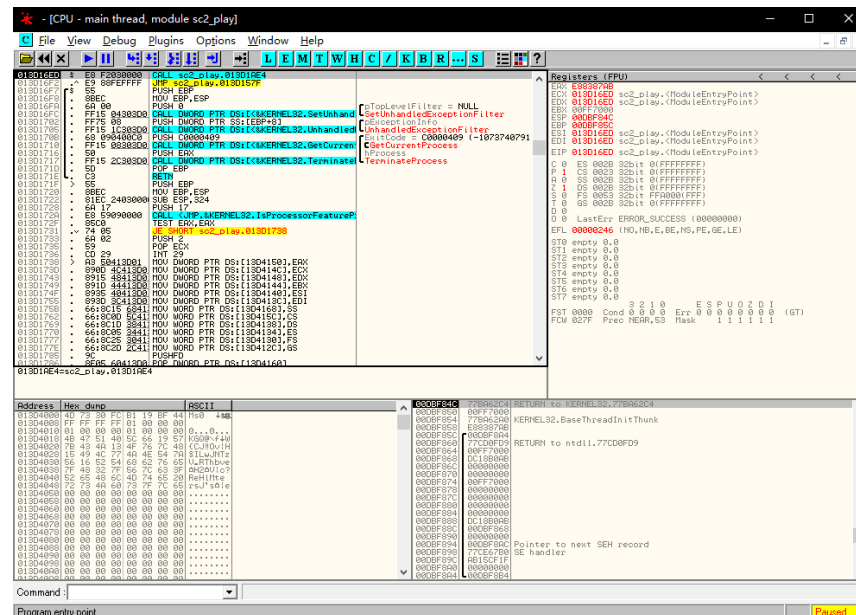
分数 100: sc2_player

URL: http://o795rwtm.bkt.clouddn.com/sc2_player.exe

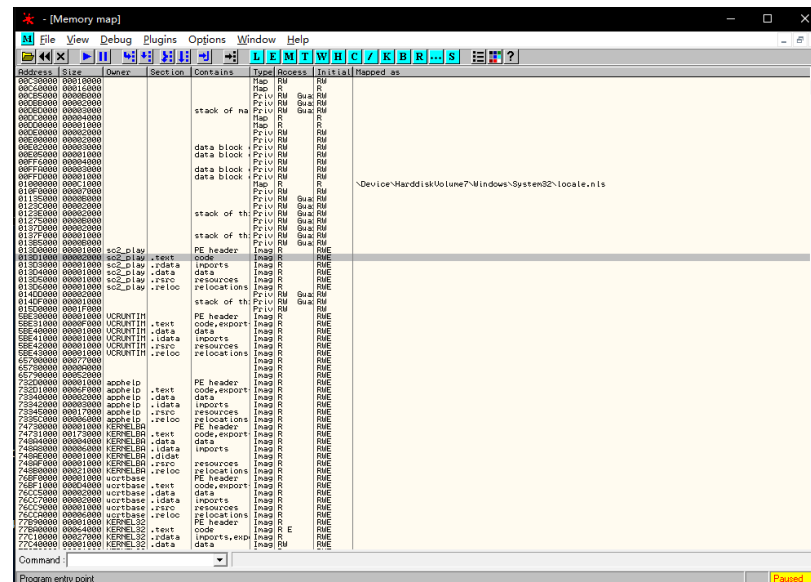
知识点: hotkey: n

题外话: 这题我解完了以后也不知道这个知识点是什么鬼, 而且 flag 也带着 IDA 字样。。有点不怎么舒服

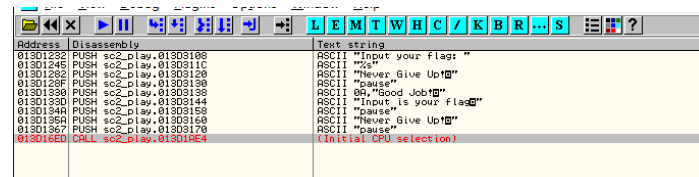
调试界面



点击 M, 确认处在位置为主程序中 (一般最初处在的位置都是 40 开头, 如果不是可能是壳或者反调试所导致的, 所以需要查看 memory map 进行检查)



右键--search for--all referenced text strings



可以看到这里 flag 明文, 我们点击 013D1330 进行定位

可以看到它产生了 4 对长度为 7 的字符串，接下来 F7 进入 13D1323 程序

013D11E7	CC	INT3
013D11E8	\$ 55	PUSH EBP
013D11E9	8BEC	MOV EBP,ESP
013D11EA	51	PUSH ECX
013D11EB	56	PUSH ESI
013D11EC	8B45 18	MOV EAX,DWORD PTR SS:[EBP+18]
013D11ED	50	PUSH EAX
013D11EE	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]
013D11EF	51	PUSH ECX
013D11F0	E9 9EFFFFFF	CALL sc2_play.013D1110
013D11F1	83C4 08	ADD ESP,8
013D11F2	8BF0	MOV ESI,EAX
013D11F3	8B55 18	MOV EDI,DWORD PTR SS:[EBP+18]
013D11F4	83C2 07	ADD EDI,7
013D11F5	52	PUSH EDI
013D11F6	8B45 0C	MOV ECX,DWORD PTR SS:[EBP+C]
013D11F7	50	PUSH EAX
013D11F8	E9 9EFFFFFF	CALL sc2_play.013D1110
013D11F9	83C4 08	ADD ESP,8
013D11FA	2BF0	AND ESI,EAX
013D11FB	8B4D 18	MOV ECX,DWORD PTR SS:[EBP+18]
013D11FC	83C1 0E	ADD ECX,0E
013D11FD	51	PUSH ECX
013D11FE	8B55 10	MOV EDI,DWORD PTR SS:[EBP+10]
013D11FF	52	PUSH EDI
013D1200	E9 74FFFFFF	CALL sc2_play.013D1110
013D1201	83C4 08	ADD ESP,8
013D1202	2BF0	AND ESI,EAX
013D1203	8B4D 18	MOV ECX,DWORD PTR SS:[EBP+18]
013D1204	83C0 15	ADD EAX,15
013D1205	50	PUSH EAX
013D1206	8B4D 14	MOV ECX,DWORD PTR SS:[EBP+14]
013D1207	51	PUSH ECX
013D1208	E9 8FFFFFFF	CALL sc2_play.013D1110
013D1209	83C4 08	ADD ESP,8
013D120A	2BF0	AND ESI,EAX
013D120B	8BFE 01	CMPEB ESI,1
013D120C	75 09	JNC SHORT sc2_play.013D11C4
013D120D	BB 01000000	MOV EAX,1
013D120E	EB 32	JMP SHORT sc2_play.013D11F4
013D120F	EB 30	JMP SHORT sc2_play.013D11F4
013D1210	> C745 FC 0000	MOV DWORD PTR SS:[EBP-4],0
013D1211	EB 09	JMP SHORT sc2_play.013D11D6
013D1212	> 8B55 FC	MOV EDI,DWORD PTR SS:[EBP-4]
013D1213	> 83C2 01	ADD EDI,1
013D1214	> 8B55 FC	MOV EDI,DWORD PTR SS:[EBP-4]
013D1215	> 837D FC 1C	CMPL EDI,DWORD PTR SS:[EBP-4],1C
013D1216	> 70 16	JGE SHORT sc2_play.013D1110
013D1217	> 8B45 1C	MOV EAX,DWORD PTR SS:[EBP+1C]
013D1218	> 8345 FC	ADD EAX,DWORD PTR SS:[EBP-4]
013D1219	> 0FB608	MOVZX ECX,BYTE PTR DS:[EAX]
013D121A	> 8BF1 34	XOR ECX,34
013D121B	> 8B55 1C	MOV EAX,DWORD PTR SS:[EBP+1C]
013D121C	> 8B55 FC	ADD EDI,DWORD PTR SS:[EBP-4]
013D121D	> 8B0A	MOV BYTE PTR DS:[EDX],0
013D121E	> EB 08	JMP SHORT sc2_play.013D11C0
013D121F	> 33C0	XOR EAX,EAX
013D1220	> 58	POP EAX
013D1221	> 8BE5	MOV ESP,EBP
013D1222	> 5D	POP EBP
013D1223	> C3	RET
013D1224	CC	INT3
013D1225	CC	INT3

我们可以看到，如果我们要想让 EAX 为 1，13D11B9 跳转必须成立，分析一下就会发现它后面的有一个部分为花指令，继续往上推，13D11B9 前面 ESI 必须为 0，ESI 的值会改变的只有 13D1175,13D118A,13D119F，测试图中，经过 call 指令的时候，EAX 会变成 0，也就是说经过这 4 个 CALL 指令，EAX 绝对不能为 0，否则就会失败（如果 ESI 为 0，后面的 ADD 指令怎么都不可能把他变成 1 的），再去看那 4 个 call 调用的是是什么程序

013D110F	CC	INT3
013D1110	\$ 55	PUSH EBP
013D1111	8BEC	MOV EBP,ESP
013D1112	51	PUSH ECX
013D1113	56	PUSH ESI
013D1114	C745 FC 0000	MOV DWORD PTR SS:[EBP-4],0
013D1115	EB 09	JMP SHORT sc2_play.013D1126
013D1116	> 8B45 FC	MOV EDI,DWORD PTR SS:[EBP-4]
013D1117	> 83C0 01	ADD EAX,1
013D1118	> 8B45 FC	MOV EDI,DWORD PTR SS:[EBP-4]
013D1119	> 837D FC 07	CMPL EDI,DWORD PTR SS:[EBP-4],7
013D111A	> 70 20	JGE SHORT sc2_play.013D1140
013D111B	> 8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]
013D111C	> 834D FC	ADD ECX,DWORD PTR SS:[EBP-4]
013D111D	> 0FB611	MOVZX EDX,BYTE PTR DS:[ECX]
013D111E	> 8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]
013D111F	> 8345 FC	ADD EAX,DWORD PTR SS:[EBP-4]
013D1120	> 0FB608	MOVZX ECX,BYTE PTR DS:[EAX]
013D1121	> 3BD1	CMPL EDX,ECX
013D1122	> 75 04	JNZ SHORT sc2_play.013D1146
013D1123	> EB D9	JMP SHORT sc2_play.013D1110
013D1124	> EB 04	JMP SHORT sc2_play.013D1140
013D1125	> 33C0	XOR EAX,EAX
013D1126	> EB 07	JMP SHORT sc2_play.013D1151
013D1127	> EB D1	JMP SHORT sc2_play.013D1110
013D1128	> BB 01000000	MOV EAX,1
013D1129	> 8BE5	MOV ESP,EBP
013D112A	> 5D	POP EBP
013D112B	> C3	RET
013D112C	CC	INT3
013D112D	CC	INT3
013D112E	CC	INT3
013D112F	CC	INT3

这里注意这个指令

013D112C	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]
013D112F	834D FC	ADD ECX,DWORD PTR SS:[EBP-4]
013D1132	0FB611	MOVZX EDX,BYTE PTR DS:[ECX]
013D1135	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]
013D1138	8345 FC	ADD EAX,DWORD PTR SS:[EBP-4]
013D113B	0FB608	MOVZX ECX,BYTE PTR DS:[EAX]
DS:[013D43C0]=31 ('1')		
EDX=013D43AE (sc2_play.013D43AE)		

[ECX]他保存的是之前加密的那 4 对字符串，每次只有一位的保存到 EDX，

013D112C	8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]
013D112F	834D FC	ADD ECX,DWORD PTR SS:[EBP-4]
013D1132	0FB611	MOVZX EDX,BYTE PTR DS:[ECX]
013D1135	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]
013D1138	8345 FC	ADD EAX,DWORD PTR SS:[EBP-4]
013D113B	0FB608	MOVZX ECX,BYTE PTR DS:[EAX]
013D113E	3BD1	CMPL EDX,ECX
013D1140	75 04	JNZ SHORT sc2_play.013D1146
DS:[013D4034]=68 ('h')		
ECX=013D43C0 (sc2_play.013D43C0), ASCII "1317131"		

[EDX]保存的是一个固定的内存位置，通过输入不同字符串发现这个里面的内容和位置是不变的

如图所示，这样我们就知道它的机制：将我们输入长度为 **28** 的字符串分成 **4** 部分进行加密，然后将他们对上面这部分进行比较，全部对才为 **flag**

013D10F0	7D 19	JGE SHORT sc2_play.013D110B	
013D10F2	8B45 14	MOV EAX, DWORD PTR SS:[EBP+14]	
013D10F5	50	PUSH EAX	
013D10F6	68 18403D01	PUSH sc2_play.013D4018	Arg4 Arg3 = 013D4018
013D10FB	8B4D 10	MOV ECX, DWORD PTR SS:[EBP+10]	Arg2 = 013D43C0 ASCII "httf\N0"
013D10FE	51	PUSH ECX	
013D10FF	8B55 08	MOV EDX, DWORD PTR SS:[EBP+8]	Arg1 sc2_play.011A1050
013D1102	52	PUSH EDX	
013D1103	E9 48FFFFFF	CALL sc2_play.013D1050	
013D1108	83C4 10	ADD ESP, 10	
013D110B	8B55 08	MOV EAX, DWORD PTR SS:[EBP+8]	
013D110E	7D 19	JGE SHORT sc2_play.013D110B	
013D10F0	7D 19	JGE SHORT sc2_play.013D110B	
013D10F2	8B45 14	MOV EAX, DWORD PTR SS:[EBP+14]	
013D10F5	50	PUSH EAX	
013D10F6	68 18403D01	PUSH sc2_play.013D4018	Arg4 Arg3 = 013D4018
013D10FB	8B4D 10	MOV ECX, DWORD PTR SS:[EBP+10]	Arg2 = 013D43B8 ASCII "t_f14g_"
013D10FE	51	PUSH ECX	
013D10FF	8B55 08	MOV EDX, DWORD PTR SS:[EBP+8]	Arg1 sc2_play.011A1050
013D1102	52	PUSH EDX	
013D1103	E9 48FFFFFF	CALL sc2_play.013D1050	
013D1108	83C4 10	ADD ESP, 10	
013D110B	8B55 08	MOV EAX, DWORD PTR SS:[EBP+8]	

013D110E	7D 19	JGE SHUK1.sc2_play.013D110B	
013D110F	8B45 14	MOV EAX,DWORD PTR SS:[EBP+14]	
013D110F	50	PUSH EAX	
013D110F	68 18403D01	PUSH sc2_play.013D4018	Arg4 = 013D4018
013D110F	9B4D 10	MOV ECX,DWORD PTR SS:[EBP+10]	Arg3 = 013D4018
013D110F	57	PUSH ECX	Arg2 = 013D43B0 ASCII "_look_c"
013D110F	8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]	
013D110F	52	PUSH EDX	Arg1 = sc2_play.011A1050
013D110F	E8 48FFFFF	CALL sc2_play.013D1050	

013D10F2	8B45 14	MOV EAX,DWORD PTR SS:[EBP+14]	[Arg4 Arg3 = 013D4018 Arg2 = 013D43A8 ASCII "mp_p1z")"
013D10F5	50	PUSH EAX	
013D10F6	68 12403D01	PUSH sc2_play.013D4018	
013D10F8	8B4D 10	MOV ECX,DWORD PTR SS:[EBP+10]	
013D10FE	51	PUSH ECX	[Arg1 sc2_play.011A1050
013D10FF	8B55 08	MOV EDI,DWORD PTR SS:[EBP+8]	
013D1102	50	PUSH EDI	
013D1103	E8 48FFFFFF	CALL sc2_play.013D1050	
013D1108	83C4 10	ADD ESP,10	

当初以为是 **flag**，兴奋的去输入却是错的。查看了下语意，要去看 **cmp**，这它不说也知道。。之后我没有去看加密机制，而是转到比较这里，经过测试，发现他的加密方式有一点为每一位对应另外一位，而且是互相对应的，加入输入的为 **a**，加密后为 **b**，那么输入 **b** 的时候，他会加密为 **a**，知道这个以后便有了思路：首先我们知道那个比较内容是固定的

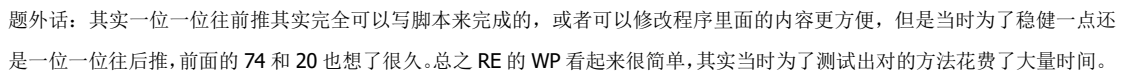
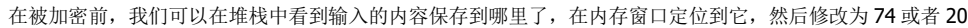
第一位 **h**，只要我输入的字符串第一位为 **h** 就可以看到它加密成什么，这样反过来输入这么内容就可以跟他对上。这里我是一位一位测试的。

[illegible]

可以看到为 **c** 所以第二位为 **c**，然后把第三位写成 **v**

最后得出 flag，以下为我当时写的草稿

这里注意下他比较的内容中(如上图),有几个位置是无法输入的比如十六进制的 74 和 20, 这里需要输入后手动到程序中修改。



MISC

分数 50: 白菜 1

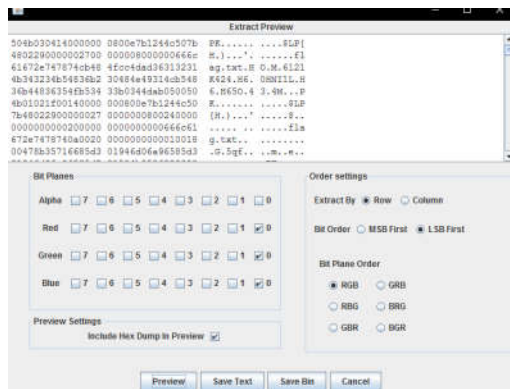
知识点: 图片隐写 lsb

URL: <http://p1kaloi2x.bkt.clouddn.com/flag.png>

图片为



使用 binwalk 查询没有结果, 打开 Stegsolve, 改变色道没有结果, 点击 stegosolve 中的 analyse—data extract, 经过设置查看到以下内容



可以看到 PK 开头, 这是 rar 类型文件, 将他保存出来, 修改后缀名为 rar, 即可看到 flag



题外话: 这道题卡了 4 天, 最后才知道在网上看到 PK 字眼突然才想到的, 接下来我会把所有的尝试都提一遍(基本上是快把网上的常用的隐写软件全都下了一遍)没有兴趣的就不用看这些失败的尝试了

1、起初 binwalk 分析出现如下结果

```
root@kali: ~/Desktop
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@kali:~# cd Desktop
root@kali:~/Desktop# binwalk flag.png
文件
-----
DECIMAL      HEXADECEMAL    DESCRIPTION
-----
0            0x0            PNG image, 1080 x 1920, 8-bit/color RGB, non-inter
laced
41          0x29            Zlib compressed data, default compression
root@kali:~/Desktop#
```

以为跟 **zlib** 解压数据有关，找到能够解压 **zlib** 的 **python** 脚本，结果老是报错，又去查询耗费了几个小时，最终放弃。

2、 下载 LSBSteg 文件进行解密，结果

```
root@kali:~/Desktop/LSB-Steganography-master# python LSBSteg.py decode -i flag.p
ng -o flag.txt
Traceback (most recent call last):
  File "LSBSteg.py", line 189, in <module>
    main()
  File "LSBSteg.py", line 183, in main
    raw = steg.decode_binary()
  File "LSBSteg.py", line 165, in decode_binary
    for i in range(l):
MemoryError
root@kali:~/Desktop/LSB-Steganography-master#
```

尝试多次无果，最终放弃

3、 下载 lsb.pyc 尝试进行解密，结果

```
root@kali:~/Desktop/cloacked-pixel-master# python lsb.py extract flag.png flag.t
xt 123456
[+] Image size: 1080x1920 pixels.
Traceback (most recent call last):
  File "lsb.py", line 194, in <module>
    extract(sys.argv[2], sys.argv[3], sys.argv[4])
  File "lsb.py", line 123, in extract
    data dec = cipher.decrypt(data out)
  File "/root/Desktop/cloacked-pixel-master/crypt.py", line 24, in decrypt
    return self.unpad(cipher.decrypt(enc[AES.block_size:]))
  File "/usr/lib/python2.7/dist-packages/Crypto/Cipher/blockalgo.py", line 295,
in decrypt
    return self.cipher.decrypt(ciphertext)
ValueError: Input strings must be a multiple of 16 in length
root@kali:~/Desktop/cloacked-pixel-master#
```

由于不熟悉 **python**，一直不知道这个错误是什么引起的，以为密码错误，尝试其他文件之后排除密码问题，之后再尝试多次无果最终放弃

4、 用 pngcheck 检查，没有发现异常

5、 用 steghide 尝试解密，尝试多次没有结果。

6、 用 braintools 进行解密，结果

```
C:\Users\... Desktop\LSB\braintools-v2.1>bftools decode braincopter flag.png
]+. +, ++++++[.]-----,,)]],,,--,,,]>[, [[[[[-
C:\Users\... Desktop\LSB\braintools-v2.1>
```

得出了一串 **brainfuck** 密文，仿佛有了进展，结果到翻译了下报错，心态崩了。

直到最后才知道 **PK** 是 **RAR** 文件的开头

分数 50：白菜 2

知识点：初识文件结构

URL: <http://p1kaloi2x.bkt.clouddn.com/misc2.jpg>

打开图片



放到 binwalk 查看

```
root@kali:~/Desktop# binwalk misc2.jpg
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
0            0x0          JPEG image data, JFIF standard 1.01
1037199      0xFD38F      Zip archive data, at least v2.0 to extract, compressed size: 41, uncompressed size: 39, name: flag.txt
1037368      0xFD438      End of Zip archive
root@kali:~/Desktop#
```

捆绑的为 zip 文件，实用 foremost 进行分解

```
root@kali:~/Desktop# foremost misc2.jpg
Processing: misc2.jpg
|foundat=flag.txt|H00M0NL3JL000L402LSNIM322313100NM50JI0
*|
root@kali:~/Desktop#
```

打开刚才分解的 rar 文件，得到 flag

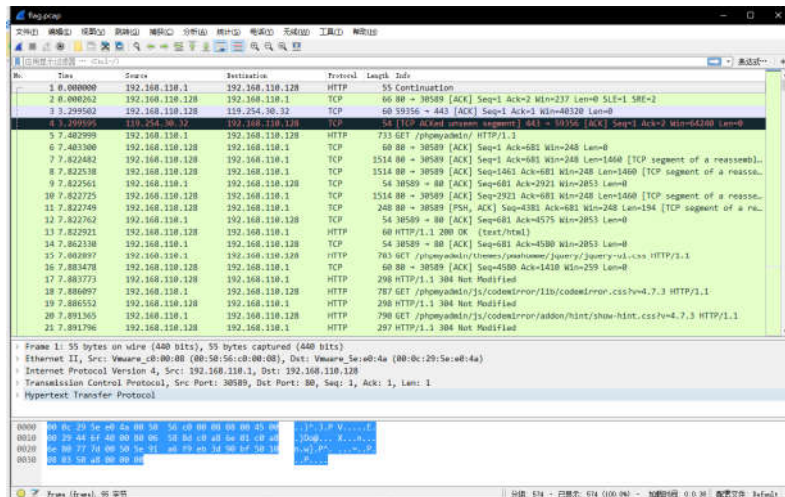


分数 50: pacp1

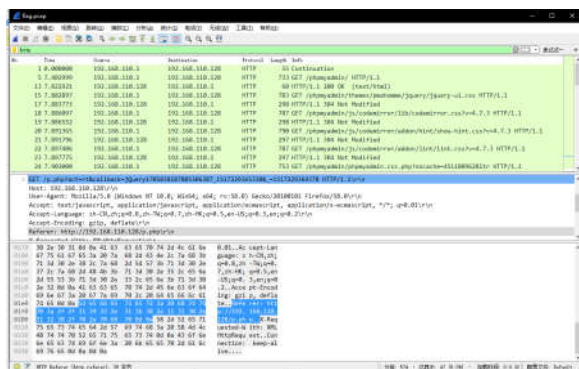
描述 ngc 从不知道哪里抓来的流量，好像里面有一个不得了的东西呢 知识点: wireshark 初识流量包

URL: <http://p1kaloi2x.bkt.clouddn.com/flag.pcap>

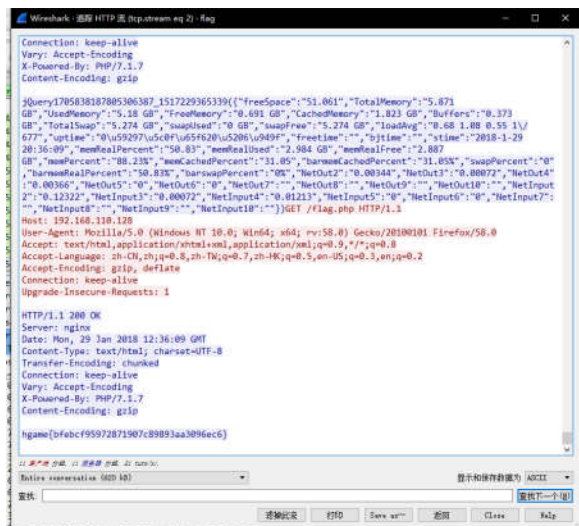
打开界面



先对其进行过滤，在过滤器中输入 http



数量不是特别多，一直往下拉，倒数第二个为flag.php，对内容进行查看，没有发现有关线索，于是右键--跟踪流--http 流拉到最下面，可以看到 flag（如果觉得这样做没有理由，可以在下面查看关键字 hgame 或者 hctd



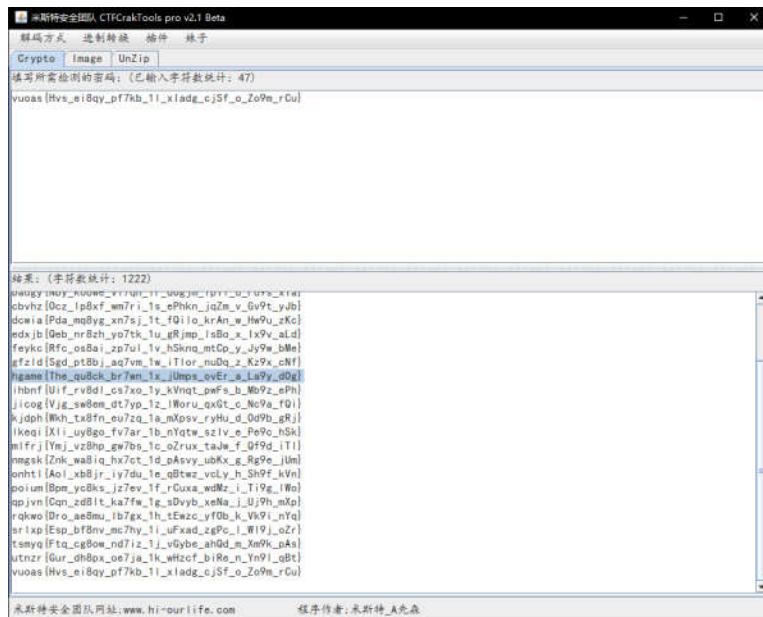
crypto

分数 50: easy Caesar

描述：相信你们都知道啥是凯撒加密的，so Ciphertext: vuoa5[Hvs_ei8qy_pf7kb_1l_xIadg_cjSf_o_Zo9m_rCu]

URL https://www.wikiwand.com/en/Caesar_cipher

拿到 CTFTool 里面解决



很明显可以找到一个答案，上交上去发现不对，尝试多次无果，以为是平台问题，后来看群里面有人说到这个问题，确认平台没问题。之后察觉这句话有点眼熟，网上查询这是“the quick brown fox jumps over a lady dog”的变种。以下是我当时进行的尝试：

第一次猜想，在这个字符串中，每个单词都有一位要么变大写，要么变数字，在这个基础上 **1x** 和 **fox** 缺了一位，那么如果在其他地方都正确的情况下，只需要添加另一位进去就行了，只是不知道 **1** 代表的是什么。于是分别尝试了

```
hgame{The_qu8ck_br7wn_1ox_jUmps_ovEr_a_La9y_dOg}
```

```
hgame{The_qu8ck_br7wn_f1x_jUmps_ovEr_a_La9y_dOg}
```

```
hgame{The_quick_brown_fox_jumps_over_a_Lady_dog}
```

```
hgame{the_quick_brown_fox_jumps_over_a_Lady_dog}
```

都是错误的，那么进行下一个猜想，假设其他地方都是正确的，**1** 这个数字是需要删除的，我们需要输入 **fo**，但是 **f** 和 **o** 其中有一个要么是大写要么是数字。于是分别尝试了以下东西

```
hgame{The_qu8ck_br7wn_Fox_jUmps_ovEr_a_La9y_dOg}
```

```
hgame{The_qu8ck_br7wn_2ox_jUmps_ovEr_a_La9y_dOg}
```

```
hgame{The_qu8ck_br7wn_3ox_jUmps_ovEr_a_La9y_dOg}
```

```
hgame{The_qu8ck_br7wn_4ox_jUmps_ovEr_a_La9y_dOg}
```

.....

```
hgame{The_qu8ck_br7wn_f0x_jUmps_ovEr_a_La9y_dOg}
```

```
hgame{The_qu8ck_br7wn_f0x_jUmps_ovEr_a_La9y_dOg}
```

```
hgame{The_qu8ck_br7wn_f1x_jUmps_ovEr_a_La9y_dOg}
```

```
hgame{The_qu8ck_br7wn_f2x_jUmps_ovEr_a_La9y_dOg}
```

.....

```
hgame{The_qu8ck_br7wn_f9x_jUmps_ovEr_a_La9y_dOg}
```

都是错误的。之后突然发现 **fox** 的读音跟 **4** 很像，恍然大悟，输入了

```
hgame{The_qu8ck_br7wn_4x_jUmps_ovEr_a_La9y_dOg}
```

还是错的。之后群里面有人爆出这道题的提示（暂且叫 **hint**）

hint: 狐狸，语意

呵呵，说了跟没说一样。

之后我思考了很长时间，突然回忆起群里面有人说这道题跟去年的原题差不多，于是去网上将 **WP** 下载过来参考，以下是我看到的內容

hgame(Caesar_cipher_8s_just_for_fun), 尝试提交, 发现这 flag 竟然不对... 于是不得不开脑洞... 数字 8 刚好对应着字母 i... 然后试试 hgame(Caesar_cipher_is_just_for_fun), 还是错.... (' ▽ ') \rightarrow $\perp = \perp$, 然后再开脑洞... 试试了 1
hgame(Caesar_cipher_1s_just_for_fun), flag 正确 OTZ

图片中提到由于 i 跟 1 长的类似，于是进行了以下尝试

以“hgame{The_qu1ck_br7wn”为开头的所有可能，还是全错。

之后尝试推理数字的意义，在字符串中 i 对应 8，o 对应 7，z 对应 9，试图推断出为什么是单词是这个字母进行变形，为什么这个字母是变成大写而不是数字，考虑因素有字母顺序，倍数等等，之后都没有任何头绪。

最后的最后，我假设了下里面所有数字都是错误的，并将他们改成其他数字。

hgame{The_qu1ck_br0wn_4x_jUmps_ovEr_a_La2y_dOg}

积分 50+

题外话：这题我同样卡了4天，心态崩了3次，最后灵感还是外面吃饭的时候想出来。

分数 50: Polybius

描述：其实这个和凯撒差不多 Ciphertext: hgame{FDXDGDADDG_FXXFAAXFAG_GDFXFFXFFXADXFDA_GDAD}

URL: <https://www.wikiwand.com/en/Polybius#/Cryptography>

查了下标题，这是个棋盘密码，看密文发现全为 ADFGX，查询得出为棋盘密码的变种，并进行解密，当初先得出的密文为

```
hgame{fritz_nebel_invented_it}
```

提交上去不对，以为解密方式不对，经过确认还是不对，再到网上查询，后来发现棋盘密码的表中 j 和 i 是在一起的，将 j 替换成 i 后，得到 flag: hgame{fritz_nebel_invented_it}

参考文献:

<http://tieba.baidu.com/p/1595383711>

分数 50: Hill

描述: Not hard key: 9 17 6 5 Ciphertext: phnfetzhzzwz 解出来之后手动加上 hgame{}

URL: https://www.wikiwand.com/en/Hill_cipher

经过查询，hill 代表的希尔密码，加密方式通过矩阵换算，当初没有找到在线网站，以为要自己手算（矩阵都忘得差不多了，逆矩阵都忘了怎么求了），但在之后找到解密网站

Plaintext

overtthehills

key = 9 17 6 5

Ciphertext

plu5et4h1rr5

加上 `hgame{}` 之后得到 flag

参考文献:

<http://blog.csdn.net/pdsu161530247/article/details/75667218>

<http://www.practicalcryptography.com/ciphers/hill-cipher/>

分数 100: confusion

[illegible]

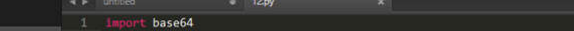
很明显这是摩斯密码，进行在线解密

`MRLTK6KXNVZXQWBSNA2FU5U2GGBSWA5BSLAZFU6J/BNDAZSRHU6Q==`

一看以为 base64 密文，经过解密后为乱码。通过查询

base32中只有大写字母 (A-Z) 和数字234567

The RFC 4648 Base 32 alphabet							
Value	Symbol	Value	Symbol	Value	Symbol	Value	Symbol
0	A	9	J	18	S	27	3
1	B	10	K	19	T	28	4
2	C	11	L	20	U	29	5
3	D	12	M	21	V	30	6
4	E	13	N	22	W	31	7
5	F	14	O	23	X		
6	G	15	P	24	Y		
7	H	16	Q	25	Z		
8	I	17	R	26	2	pad	=

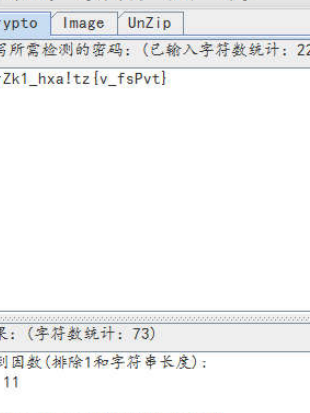


```
OPEN FILES
untitled
12.py
1 import base64
2 print base64.b32decode("MRLTK6KXNVZXQWBSNA2FSU2GGBSW458SLAZFU6SVJ3BND4ZSRHU6====")

C:\WINDOWS\system32\cmd.exe
C:\Users\user\Desktop\java脚本>python 12.py
d5Ymxmx2H4tSF0ent2X2ZzUHz0f==
C:\Users\user\Desktop\java脚本>(python)
```

明显为 base64，直接解密得到密文

根据之前写题目的经验看到{}第一个想到栅栏密码或者是列置换，先进行栅栏密码解密



米斯特安全团队 CTFCrackTools pro v2.1 Beta

解码方式 进制转换 插件 林子

Crypto Image UnZip

填写所需检测的密码: (已输入字符数统计: 22)

unrZk1_hxa!tz[v_fsPvt]

结果: (字符数统计: 73)

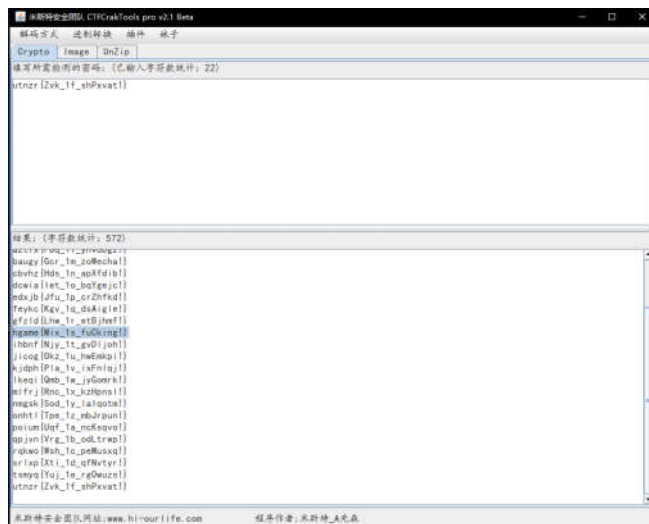
得到国数(排除1和字符串长度):

2 11

第1栏: urk_x!zvfPtnZ1hat[_sv]

第2栏: utnznr[Zvk_1f_shPxvat!]

根据 flag 为 hgame{XXXXXX}或者 hctf{XXXXXX} 框架，第二种更显得合理一些。将 utnzt{Zvk_1f_shPxvat!}进行凯撒分解



最终得到 flag