# BUUCTF-RE-0x04



## title: BUUCTF-RE-0x04 date: 2021-05-10 19:58:34 tags:

刷题记录。狗头~~

[ACTF新生赛2020]easyre

首先看题

Challenge	Top 3 Solves	×
[ACTF	─新生赛2020	]easyre
	1	
得到的 flag 请使	回上flag{}提交。	
よ attachment.		
Flag		Submit

啥都没有,下载下来看见有两个运行文件,有一个是可以使用的。



查壳

Weid v0.95		—	$\times$
文件: C:\Users\张宏\Desktop\CTF\BVVC	IF-RE\[ACTF	新 <u>生</u> 赛2020]eas	潮览
入口点: 0000E480	EP 段:	UPX1	$\triangleright$
文件偏移: 00001680	首字节:	60, BE, 15, DO	$\sum$
连接器版本: 2.24	子系统:	Win32 console	$\sum$
PESniffer: UPX v0.89.6 - v1.02 / v1.0 PEiDDSCAN: UPX 0.89.6 - 1.02 / 1.05 -	5 - v1.22 2.90 -> Ma	rkus & Laszlo [	$\overline{\mathbf{b}}$
多文件扫描(M) 查看进程(T)	扩展信息	1. 插件	
✓ 总在最前(S) 送项(0)	关于(A)	しして「退出」	

UPX壳,因为会脱压缩壳了。所以就脱吧,没有用脱壳机。

放入OllyDbg脱壳失败了,地址直接是乱的。然后放入的是X64dbg脱得壳,这个就很明显。

使用ESP定律把壳脱了。然后修复之后,可以运行了。

放入IDA然后简单分析之后就做不来了。发现这些数据可能有问题。

IDA - easyre_dump_SCY.ex	ke C:\Users\张宏\D	esktop\easyre_c	lump_SCY.exe							_	٥	×
<u>File E</u> dit <u>J</u> ump Searc <u>h V</u> iev	v Deb <u>u</u> gger Lum	i <u>n</u> a <u>O</u> ptions <u>W</u>	indows Help									
📂 🔚 🗢 🔻 🔶 🖛 👫	🖀 🖀 🕽 🙆	🔺 🔵 📑 📑	ist. <b>t</b> • ± mi×	No debugger	• 👘	🔁 🚮 🕈 🕅						
Library function Rogu	lar function 📕 I	estruction D	ta Unovalored	Extornal cymbol Iumi	na function							
Z Runations window				External symbol Edul								
Functions window		IDA View-A	Pseudocode-B	Pseudocode−A	⊠Hex View-1 □	Structures	Enums E	Imports 🖾	If Exports			
Function name		lintcde	ci main(int argo	, const cnar **argv	, const char	r≁envp)						ſ
<u>f</u> sub_401000		2 BVTE W	1[12] · // [esp+1	2hl [ebn_2Ehl BVREE								
<b>f</b> TopLevelExceptionFilter			/5[3]· // [esp+1	Fh] [ehn-22h]								l
<u>f</u> start		5 BYTE V	5[5]: // [esp+2A	hl [ebp-16h] BYRFF								
<u>f</u> sub_4012C0		6 int v7:	// [esp+2Fh] [e	bp-11h]								
f sub_4012E0		7 int v8:	// [esp+33h] [e	bp-Dh1								
🗲 sub_401330		8 int v9:	// [esp+37h] [e	ebp-9h1								
🗾 main		9 char v10	); // [esp+3Bh]	[ebp-5h]								
F sub 401460		10 int i:	/ [esp+3Ch] [eb	p-4h]								
テ sub_4014B0		11										
T sub 401550	•	12 sub 401	10();		// 不知道這	(是个啥						
f sub 401630	•	13 gmemcpy	(v4, "*F'\"N,\"(	<pre>I?+@", sizeof(v4));</pre>								
f sub 401640	•	14 printf(	'Please input:")	3								
5 sub_401640	•	15 scanf("S	6s", v6);		// ACTF{}							
5 sub_401780	•	16 if ( v6	[0] != 'A'    ve	[1] != 'C'    v6[2]	!= 'T'    v6	[3] != 'F'    ve	6[4] != '{'	<pre>v10 != '}' )</pre>				
J SUD_401780	•	17 returi	n 0;									
F sub_401990	•	18 v5[0] =	v7;									
F sub_401A10	•	19 v5[1] =	v8;									
<u>f</u> sub_401A30	•	20 v5[2] =	v9;									
<u>f</u> sub_401BB0	•	21 for ( i	= 0; i <= 11; +	+i )								
<u>f</u> _getmainargs	~	22 {										
<	>	23 if (	/4[i] != byte_40	02000[*((char *)v5 +	i) - 1] )							
Line 2 of 46	•	24 reti	urn 0;									
🛦 Graph overview	0 Ø ×	25 }										
		26 printf(	'You are correct	:!");								
		27 return (	);									
	•	28}										
┟╼╧╧╾╛		0000085B r	main:27 (40145B)									
Output window												
405024: using guessed	type int dword	_405024;										^
401C88: call analysis	failed											
402088: using guessed	type int (*off	_402088)[172	];									
402088: using guessed	type int (*off	_402088)[172	];									
402088: using guessed	type int (*off	_402088)[172	];									
401C88: call analysis	failed											~
Python												
AU: idle Down Disk: 1	8GB											

然后查看思考,可能是我脱壳没有脱干净。可是直接使用脱壳机脱得壳反汇编的结果也是一样的。所以可能就是这样的。前面显 示的是一个读不懂的函数,然后后面的比较应该是进行比较。可是跟flag好像没啥关系,也跟输入的内容没啥关系。所以这里就 有点不是很懂了。

```
דוור אסי // הארבאלר אוד להא אוד
       9
          char v10; // [esp+3Bh] [ebp-5h]
      10
          int i; // [esp+3Ch] [ebp-4h]
      11
          sub_401A10();
qmemcpy(v4, "*F'\"N,\"(I?+@", sizeof(v4));
                                                          // 不知道这是个啥
     • 12
     • 13
                                                                              应该是12啊,这里是15,迷糊
          printf("Please input:");
     14
                                        char[15]
          scanf("%s", v6); // ACTF{}
if ( v6[0] != 'A' || v6[1] != 'C' || v6[2] != 'T' || v6[3] != 'F' || v6[4] != '{' || v10 != '}' )
     • 15
    • 16
     • 17
            return 0;
     18
          v5[0] = v7;
     19
          v5[1] = v8;
     20
          v5[2] = v9;
     21 🔵
          for ( i = 0; i <= 11; ++i )
      22
          Ł
     23
            if ( v4[i] != byte_402000[*((char *)v5 + i) - 1] )
    24
              return 0:
      25
          }
    26
          printf("You are correct!");
     27
          return 0;
     28
# 主要的一个函数的解析
int cdecl main(int argc, const char **argv, const char **envp)
{
 _BYTE v4[12]; // [esp+12h] [ebp-2Eh] BYREF
 DWORD v5[3]; // [esp+1Eh] [ebp-22h]
  _BYTE v6[5]; // [esp+2Ah] [ebp-16h] BYREF
 int v7; // [esp+2Fh] [ebp-11h]
 int v8; // [esp+33h] [ebp-Dh]
  int v9; // [esp+37h] [ebp-9h]
  char v10; // [esp+3Bh] [ebp-5h]
  int i; // [esp+3Ch] [ebp-4h]
                                               // 不知道这是个啥
  sub_401A10();
  qmemcpy(v4, "*F'\"N,\"(I?+@", sizeof(v4));
  printf("Please input:");
  scanf("%s", v6);
                                               // ACTF{}
  if ( v6[0] != 'A' || v6[1] != 'C' || v6[2] != 'T' || v6[3] != 'F' || v6[4] != '{' || v10 != '}' )
        //这里的字符是手动改变过的
   return 0;
  v5[0] = v7;
  v5[1] = v8;
  v5[2] = v9;
  for ( i = 0; i <= 11; ++i )
  {
   if ( v4[i] != byte 402000[*((char *)v5 + i) - 1] )// v4 = 数组[flag[i] -1]
     return 0;
  }
  printf("You are correct!");
  return 0;
}
```

```
# 原来的v4
v4 = "*F'\"N,\"(I?+@"
# 改一下
v4 = "*F\'\"N,\"(I?+@"
然后就是12个字符了,不过有点奇怪的是为什么IDA显示说是char【15】,而只有14个呀,所以可能数据末尾可能是有换行符的吧。暂时这么猜
想的。
```

接下来,就开始写逆向的算法。

可能是因为python不熟的缘故,所以算法最后的结果如下。然后中间还有一个就是在使用Shift+E将数据到处的时候,好像不知道 怎么就把其中一个空格给删不见了。这个就有点奇特了。

#### flag

```
# easyre
v4 = "*F\'\"N,\"(I?+@"
# List = "~}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)(\x27&%$# ! \""
list = "~}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)(\x27&%$# !\""
flag =""</pre>
```

```
for i in v4:
    temp = list.find(i)+1
    flag += chr(temp)
print("flag{"+flag+"}")  # fLag{U9X_15_W6@T?}
print("ACTF{"+flag+"}")  # ACTF{U9X_15_W6@T?}
```

这个之间踩得坑还是挺多的。

OK, 解决了。

Challenge	Top 3 Solves	×
[ACTF	=新生赛2020	]easyre
	1	-
得到的 flag 请任	回上flag{}提交。	
よ attachment.		
Flag		Submit

## [GWCTF 2019]pyre

### 工具和环境

uncompyle6 #一个将pyc文件还原成py文件的工具 python3 #看着代码,把他写出来

## 小知识点

做完之后,看见的知识点应该考察的是算法逆向过程中的异或运算。即一个数两次异或得到自己本身。

### 题目



[GWCTF 2019]pyre

得到的 flag 请包上 flag{} 提交。

下载

下载得到的是一个pyc的文件,所以就使用uncompyle6 将这个编译的文件,反汇编得到py文件。

源码如下

```
# uncompyle6 version 3.7.4
# Python bytecode 2.7 (62211)
# Decompiled from: Python 3.7.8 (tags/v3.7.8:4b47a5b6ba, Jun 28 2020, 08:53:46) [MSC v.1916 64 bit (AMD64)]
# Embedded file name: encode.py
# Compiled at: 2019-08-19 21:01:57
print 'Welcome to Re World!'
print 'Your input1 is your flag~'
l = len(input1)
for i in range(1):
   num = ((input1[i] + i) % 128 + 128) % 128
   code += num
for i in range(l - 1):
    code[i] = code[i] ^ code[(i + 1)]
print code
code = ['\x1f', '\x12', '\x1d', '(', '0', '4', '\x01', '\x06', '\x14', '4', ',', '\x1b', 'U', '?', 'o', '6', '*'
, ':', '\x01', 'D', ';', '%', '\x13']
```

```
算法逆向(python3)
```

```
code = ['\x1f', '\x12', '\x1d', '(', '0', '4', '\x01', '\x06', '\x14', '4', ',', '\x1b', 'U', '?', 'o', '6', '*'
, ':', '\x01', 'D', ';', '%', '\x13']
l = len(code)
# print(L)
for i in range(1-2,-1,-1):
    code[i] = chr(ord(code[i]) ^ ord(code[(i + 1)]))
# print(code)
flag = ""
for i in range(1):
    num = ord(code[i]) - i
    if num <0:
        num+=128;
    flag +=chr(num)
print(flag)
# GWHT{Just_Re_1s_Ha66y!}</pre>
```

得到flag

GWHT{Just\_Re\_1s\_Ha66y!}

因为题目要求提交flag形式的。所以还要改一下

flag{Just\_Re\_1s\_Ha66y!}

这下对了。

简单的注册器

工具及环境

jadk\_v1.2.0 #用来反汇编apk文件 python #计算出flag ApkScan-PKID.jar #查壳工具

#### 题目



要求

简单注册器

生活中难免会有需要使用一些付费的程序,但是没有绿色版怎么办?只能自己逆向看看注册程序的代码是什么逻辑了。 注意:得到的 flag 请包 上 flag{} 提交

#### 下载

下载后看见是一个APK,所以尝试着查下壳。

#### 查壳

用了查壳的软件没有查到壳。



使用jadx反汇编查看(这个工具感觉是非常友好的软件),在这个上面的安卓应用反汇编的更强吧。



#### python3写的

```
str = "dd2940c04462b4dd7c450528835cca15"
# 将str转化为List可以使用下标来处理。
list = []
for j in str:
   list.append(j)
x = list # 处理了之后,就用原来的函数来处理
x[2] = chr(ord(x[2]) + ord(x[3]) - 50)
x[4] = chr(ord(x[2]) + ord(x[5]) - 48);
x[30] = chr(ord(x[31]) + ord(x[9]) - 48);
x[14] = chr(ord(x[27])+ ord(x[28]) - 97);
for i in range(0,16):
   s = x[31-i]
   x[31-i] = x[i]
   x[i] = s
str2 = "".join(list) # 处理完之后,将原来的改成字符串打印输出
flag = ''
flag = flag+"flag{"+str2+"}"
print(flag)
# print(x)
```

然后运行就出来了。

#### flag

flag{59acc538825054c7de4b26440c0999dd}

## [BJDCTF2020]JustRE

#### 工具以及环境

Exeinfo PE # 查壳

#### 题目及描述

#### [BJDCTF2020]JustRE

得到的 flag 请包上 flag{} 提交。来源: https://github.com/BjdsecCA/BJDCTF2020

下载 attachment.exe

查壳

Ex	einfo PE - ve	er.0.0.6.2 by A.S.L - 108	83+97 sign	2020.07.10	_		×
	<u>F</u> ile : at	ttachment - 副本.exe	100 100 100 100 100 100 100 100 100 100		<u>/ н</u>		
	Entry Point :	00001462 00 <	EP Section :	.text			
ad	File Offset :	00000662	First Bytes :	55.8B.EC.6A.FF	•		Plug
Q	Linker Info :	6.00	SubSystem :	Windows GUI	PE	÷.[	
3	File Size :	00009F97h < 🛚	Overlay :	NO 00000000	0		S
ein	Image is 32b	it executable	RES/OVL : 1	<mark>6/0%</mark> 2019	X		000
x	Microsoft Vis	sual C++ ver 5.0/6.0 - no se	ec. Cab/7z/Zip	- 2019-09-15	Scan / t	1.12	Rip
(D)	Lamer Info -	Help Hint - Unpack info		0 ms;			1045310000
	Big sec. 1.t	ext , Not packed , try www	.ollydbg.de or	x64 debug v002	0 🜮	1.	<u>&gt;</u> >
			The Area	marks we a	10. 20 1. 20		1

没有壳那么我们就考虑使用IDA和OD

因为这里是32位的程序,先使用IDA32.

### 逆向过程

通过搜索字符串,然后发现字符串。

Winner and an and a set and a set and a set a se	or the course for the c			bearing foregoing and						- ··
File Edit Jump Search View Debugger	Lumina Options W	indows Help								
📂 🗖 🗢 🕶 🕈 🏪 🐴 🐥 🗎	🔬 🔼 🔵 🛲 📾	वि 📌 र 🖈 वि	úХ	No debugger	· 🗞	2 🔐 🕈 😭				
									2	•
Library function Regular function	Instruction Da	ata 📕 linexplo	ored H	External symbol 📕 Lumiu	a function					
Functions window	X TTALL		1	Céniure mindem 🖱					<b>T D</b>	
Eurotion name	↑ WIDA View-A □	Pseudocoo	de−A ⊔	Strings window 🖬	Mex view-1	M Structures 🖬	Enums 🛛	M Imports 🖬	Exports 🗳	
	Address	Length	Туре	String						<u>^</u>
<b>f</b> WinMain(x,x,x,x)	S .10ata.004069	. 00000006	ć	VintualFree						
f sub_4010C0	s .rdata:004009	. 00000000	ć	Virtuairree						
f sub_401150	S .10ata.004069	. 00000009	ć	Delland						
f sub_4011C0	S .10ata.004069	. 0000000A	ć	MiniteFile						
f DialogFunc	s .rdata:004069	. 0000000A	Ċ	CotlastError						
<u>f</u> _sprintf	s .ruata:004069	000000000	c	CotEiloDointor						
f start	S .10ata.004009	. 0000000F	ć	HeenAlles						
famsg_exit	S .10ata.004069	. 0000000A	ć	CatChinfa						
f_fast_error_exit	s .rdata:004069	. 0000000A	ć	GetCPINIO						
f_tlsbut	S .rdata:004069	. 00000007	Ċ	GetACP						
<u>f</u> sub_4016B6	s .rdata:004069	. 00000009	Ċ	GetUEIMCP						
f_write_char	S .rdata:004069	. 00000000	Ċ							
<u>f</u> _write_multi_char	's' .rdata:00406	00000000	Ċ	НеаркеАпос						
<u>f</u> _write_string	S .rdata:00406	000000F	C	GetProcAddress						
<u>f</u> _get_int_arg	.rdata:00406	00000000	C	LoadLibraryA						
<u>f</u> _get_int64_arg	s .rdata:00406	00000000	C	SetStaHanale						
<u>f</u> _get_short_arg	s .rdata:00406	00000014	C	MultiByte lowideChar						
<u>f</u> cinit	's' .rdata:00406	000000D	C	LCMapStringA						
<u>f</u> _exit	's' .rdata:00406	00000000	C	LCMapStringw						
<u>f</u> _exit	's' .rdata:00406	000000F	C	GetStringTypeA						
<u>f</u> _doexit	s .rdata:00406	000000F	C	GetStringTypeW						
<u>f</u> _initterm	's' .rdata:00406	0000011	C	FlushFileBuffers						
<u>f</u> _XcptFilter	's' .rdata:00406	00000000	C	CloseHandle						
<u>f</u> _xcptlookup	's' .rdata:00406	00000000	C	KERNEL32.dll	2 10 2 2					
f wincmdln	<ul> <li>data:00407030</li> </ul>		C	BJD{%d%d2069a4579	20233ac}					
< >>	's' .data:00407040	_ 00000010	C	怒已经点了%d次						~
Line 1 of 113	Line 99 of 100									
Output window										_ # ×
Using FLIRT signature: Microsoft	VisualC 2-14/ne	t runtime								^
Propagating type information										
Function argument information ha	is been propagate	d								
The initial autoanalysis has bee	n finished.									
4099F0: using guessed type int d	lword_4099F0;									*
Python										
AU: idle Down Disk: 33GB										
					-			-	22:17	~ -



🖼 xrefs to aBjdDD2069a4579			×
Directic Ty Address	Text		
🖼 Up o DialogFunc+5A	push offset aBjdDD2069a4579; " BJD{%d%d2069a45792d233ac}"		
Line 1 of 1			
	OK Cancel Search Help		

确定后点击确定。

可以猜出大概是这个意思。



#### Flag

BJD{1999902069a45792d233ac}

然后在BUUCTF里面需要将格式修改一下变成 flag{1999902069a45792d233ac}

#### 知识点

这个使用动态调试的时候电脑死机了。所以就静态调试搞出来了。大概需要我们点击19999次。

🔡 吾愛破解	- atta	chment - 副本.exe	- [LCG - 主线程, 模块 - attachme]						a ×
C 文件(F)	查看	(V) 调试(D) 插	牛(P) 选项(T) 窗口(W) 帮助(H) [+] 快	捷菜单 Tool	s BreakPo	oint-> BP P VB Notepad	Calc Folder CMD Exit		_ 8 ×
暂停		<u>&gt;</u> • × ▶ ∥	▶ <b>+ + ≥ ↓ +</b> → 1 e m t	whcH	k b r	s 🗄 🔣 ?	三 🛯 🗛 🔍 🔘 👭 🖃 吾 🗷	選 解	
00401380	-	50	push eax				· · · · · · · · · · · · · · · · · · ·	寄存器 (FPU)	
00401381	-	8D4424 04	<pre>lea eax,dword ptr ss:[esp+0x4]</pre>					EAX 0000000	
00401385	-	68 4C704000	push attachme.0040704C		ASCII "	您已经点了 🎖 次,加油不	要停下来!"	ECX 0000000	
0040138A	-	50	push eax					EDX 0000000	
0040138B	-	E8 8000000	call attachme.00401410					EBX 00250866	
00401390	-	A1 F0994000	mov eax,dword ptr ds:[0x4099F0					ESP 0019FBA4	
00401395	-	83C4 OC	add esp,0xC					EBP 0019FBE0	
00401398	-	3D 1F4E0000	cmp eax,0x4E1F					ESI 0000000	
0040139D	•~	/5 31	jnz short attachme.004013D0					EDI 00080B98	
0040139F	-	0H UU			40000			FTP 778F18CC	win32u.770F10CC
00401301		08 IF4E0000	Jos ook duoud ptv cci[ocp+0v9]		19999				
00401300	-	604624 86 40 98786888	nuch attachmo 00007020		00011	D ID (%d%d2060.5bE702d222	- <b>1</b>	C 0 ES 002B	32位 Ø(FFFFFFF)
00401300	-	C4	puch acy		HSGII	0001/00/02/00/04/07/72020000		P 0 CS 0023	32位 U(FFFFFFF)
00401380		F8 58000000	call attachme.00401410					H 0 55 002B	
00401385		8B4424 78	nov eax_dword_ntr_ss:[esn+8x78	1	attachr	ne . AA4A135A		2 0 FS 002B	321 <u>17</u> 8(FFFFFFF) 22倍 206888(FFF)
AA4A13B9		8364 10	add esn. 0x10	1	accuoin			5 0 FS 0053	321 <u>17</u> 394000(FFF)
004013BC	_	8D5424 00	lea edx.dword ptr ss:[esp]						32 <u>17</u> 0(FFFFFFF)
00401300		52	push edx		rText =	NULL		0 0 LastErr	
004013C1	-	50	push eax		hWnd =	NULL		o o Lasteri	Emon_3000E33 (8888888
004013C2	-	FF15 B060400	call dword ptr ds:[<&USER32.Se	tWindowTex	LSetWind			EFL 00000202	(NO,NB,NE,A,NS,PO,GE,G
004013C8	-	3300	xor eax,eax					STØ emptu Ø.Ø	
004013CA	-	83C4 64	add_esp,0x64					ST1 empty 0.0	
004013CD	-	C2 1000	retn 0x10					ST2 empty 0.0	
<u> 664613D6</u>	$\rightarrow$	885424 68	mov edx.dword ntr ss:[esn+8x68	1	attachr	ne . AA4AAAAAA		ST3 empty 0.0	
								ST4 empty 1.0	000000000000000000000000000000000000000
								ST5 empty 8.5	000000000000000000000000000000000000000
								ST6 empty 1.0	000000000000000000000000000000000000000
地址	HEX	数据		ASCII		∧ 0019FC5C	00401289 返回到 attachm	ne.00401289 来自	user32.DialogBoxPar 🗸
00406000	FØ	DF 39 76 E0 2	E 3A 76 A0 32 3A 76 40 0C 3A 76	/ / 病9v?:v?	:v@.:v	0019FC60	00400000 attachme.00400	3000	
00406010	DØ	98 39 76 70 0	E 3A 76 B0 98 39 76 80 DF 39 76	袠9vp∎:v	揮9υ∎?∪	0019FC64	00000081		
00406020	CØ	4C 3B 76 D0 0	B 3A 76 50 F5 39 76 C0 26 6E 77	繪;v?:vP	?v?nw	0019FC68	00080B98		
00406030	C 0	F3 39 76 90 2	1 3A 76 CO 03 3A 76 30 15 3A 76	荔9v?:v?	:v0 <b>:</b> :v	0019FC6C	00401350 attachme.00401	1350	
00406040	40	5D 6E 77 50 3	5 3A 76 10 E0 39 76 B0 35 3A 76	@]nwP5:vl	?v?:v	0019FC70			
00406050	C 0	05 3A 76 60 D	F 39 76 60 0A 3A 76 80 0C 3A 76	?:∪`?∪`.	:v <b>I</b> .:v	0019FC74	FFFFFFE		
00406060	ΕØ	1E 3A 76 70 1	9 3A 76 00 41 3A 76 10 99 39 76	?:vp∎:v.(	i:v∎?v	0019FC78	9999999999 7471-FEBB 后回石 user 99	7471-000	
00406070	80	2E 3A 76 20 4	F 3B 76 30 0E 3A 76 B0 32 3B 76	.:v 0;v	9∎:v?;v	0019FC7C	7074EEBB ]丞回到 USEP32.	.7074EEBB	
00406080	A0 -	18 3A 76 40 3	5 38 76 F0 1E 3A 76 90 1F 3A 76	?:0@5;0?	:0?:0	00105094	00000020		
00406090	00	17 3A 76 90 3	3 3A 76 90 89 39 76 60 17 3A 76	- <b>-</b>	×90 ∎:0	0019FC84	666868020		
004060A0	FU	0B 3H 76 4U U	H 3H 70 CU F4 39 76 UU UU UU UU	?:0@.:0∉	90	00191000	0000070		~
M1 M2 M3	M4	МБ	Command:	<b>_</b>				A	RSP RBP NONR
VA: 004013	A1 -	> 004013A6	Size: (0x0005 - 00005 bytes)	# (0x0	001 - 0000	01 dwords) Offset	: 000005A1 -> 000005A6	- Section: 🔁 🖣	P 🦘 🙂 🍨 🖽 📲 🏭

## 总结

BJD{1999902069a45792d233ac} 所以 flag{1999902069a45792d233ac}