

# bugkuCTF平台逆向题第五道love题解

原创

iqiqiya 于 2017-12-28 16:05:04 发布 7720 收藏

分类专栏: -----bugkuCTF 我的CTF进阶之路 文章标签: [bugku CTF love writeup reverse](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/xiangshangbashaonian/article/details/78923289>

版权



-----bugkuCTF 同时被 2 个专栏收录

9 篇文章 0 订阅

订阅专栏

我的CTF进阶之路

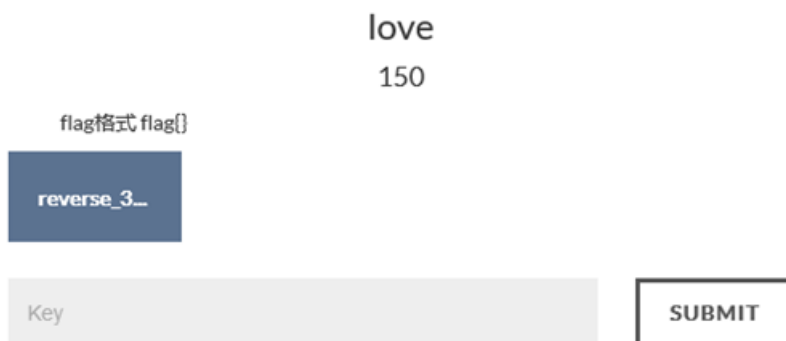
108 篇文章 18 订阅

订阅专栏

题目链接:

[http://123.206.31.85/files/99278e700add95793a765b13ab1314ea/reverse\\_3.exe](http://123.206.31.85/files/99278e700add95793a765b13ab1314ea/reverse_3.exe)

tips:



查壳无壳 (截图略)

载入OD

搜索字符串

```
0017492E push reverse_.0017845C RegCloseKey
00174951 push reverse_.00178470 SOFTWARE\Wow6432Node\Microsoft\Vi
001749A3 push reverse_.001784FC ProductDir
00174A38 mov eax,dword ptr ds:[0x178364] bin\MSPDB140.DLL
00174A44 mov eax,dword ptr ds:[0x178368] n\MSPDB140.DLL
00174A50 mov eax,dword ptr ds:[0x17836C] MSPDB140.DLL
00174A5C mov eax,dword ptr ds:[0x178370] PDB140.DLL
00174A68 mov eax,dword ptr ds:[0x178374] B140.DLL
00174A74 mov eax,dword ptr ds:[0x178378] 40.DLL
00174A80 mov eax,dword ptr ds:[0x17837C] .DLL
00174A8C mov eax,dword ptr ds:[0x178380] LL
00174C02 push reverse_.00178524 MSPDB140
00174C1C push reverse_.00178518 DLL
00174D7F push reverse_.00178554 PDBOpenValidate5
00174DEA push reverse_.00178568 r
00175758 push reverse_.00177B74 please enter the flag:
00175769 push reverse_.00177B8C %20s
00175822 push reverse_.0017A034 e3nifTMsB C@n@dH
00175842 push reverse_.00177B94 wrong flag!\n
00175851 push reverse_.00177C4C rigth flag!\n
```

随意输入11111111111111111111

```
please enter the flag:11111111111111111111
```

```
eax=0253E1E0, (ASCII "MTExMTExMTExMTExMTExMTE=")
```

发现进行了base64加密

再向下单步 发现、

0017581F	8BF4	mov esi,esp	
00175821	50	push eax	
00175822	68 34A01700	push reverse_.0017A034	e3nifiH9b_C@n@dH
00175827	8D8D 6CFFFFFF	lea ecx,dword ptr ss:[ebp-0x94]	
0017582D	51	push ecx	
0017582E	FF15 84B11700	call dword ptr ds:[<&ucrtbased.strncmp>	ucrtbase.strncmp
00175834	83C4 0C	add esp,0xC	
00175837	3BF4	cmp esi,esp	
00175839	E8 E9B8FFFF	call reverse_.00171127	
0017583E	85C0	test eax,eax	
00175840	74 0F	jc short reverse_.00175851	
00175842	68 947B1700	push reverse_.00177B94	wrong flag!\n
00175847	E8 E3BAFFFF	call reverse_.0017132F	
0017584C	83C4 04	add esp,0x4	
0017584F	EB 0D	jmp short reverse_.0017585E	
00175851	68 4C7C1700	push reverse_.00177C4C	righ flag!\n
00175856	E8 D4BAFFFF	call reverse_.0017132F	
0017585B	83C4 04	add esp,0x4	
0017585C	22C0	mov eax,eax	
0017A034=reverse_.0017A034 (ASCII "e3nifiH9b_C@n@dH")			

结合IDA看下

.text:004119E0	00000007	C	offset
.text:004158B0	0000000C	C	base64input
.text:004158C8	00000006	C	input
.text:004158CE	00000005	C	nlen
.text:004158D3	00000D02	C	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=
.rdata:00417B30	00000042	C	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=
.rdata:00417B74	00000017	C	please enter the flag:
.rdata:00417B8C	00000005	C	%20s
.rdata:00417B94	0000000D	C	wrong flag!\n
.rdata:00417BE1	0000001B	C	tack around the variable '
.rdata:00417BFC	00000011	C	' was corrupted.
.rdata:00417C11	0000000E	C	he variable '
.rdata:00417C20	0000002B	C	' is being used without being initialized.
.rdata:00417C4C	0000000D	C	righ flag!\n
.rdata:00417C70	000000DD	C	The value of ESP was not properly saved across a function call. This is usually a result of calli...
.rdata:00417D80	0000011D	C	A cast to a smaller data type has caused a loss of data. If this was intentional, you should m...
.rdata:00417ED8	0000001D	C	Stack memory was corrupted!\n\n

```
int sub_4156E0()
{
    size_t v0; // eax@6
    constchar*v1; // eax@6
    size_t v2; // eax@9
    char v4; // [sp+0h] [bp-188h]@6
    char v5; // [sp+Ch] [bp-17Ch]@1
    size_t v6; // [sp+10h] [bp-178h]@3
    size_t j; // [sp+DCh] [bp-ACh]@6
    size_t i; // [sp+E8h] [bp-A0h]@1
```

```

char Dest[108]; // [sp+F4h] [bp-94h]@5

char Str; // [sp+160h] [bp-28h]@6

char v11; // [sp+17Ch] [bp-Ch]@6

unsigned int v12; // [sp+184h] [bp-4h]@1

int savedregs; // [sp+188h] [bp+0h]@1

memset(&v5, 0xCC, 0x17);

v12 = (unsigned int) &savedregs ^ __security_cookie;

for( i = 0; (signed int) i < 100; ++i )

{

v6 = i;

if( i >= 0x64 )

sub_411154();

Dest[v6] = 0;

}

sub_41132F("please enter the flag:", v4);

sub_411375("%20s", (unsigned int) &Str);

v0 = j_strlen(&Str);

v1 = (const char*) sub_4110BE(&Str, v0, &v11);

strncpy(Dest, v1, '(');

sub_411127();

i = j_strlen(Dest);

for( j = 0; (signed int) j < (signed int) i; ++j )

Dest[j] += j;

v2 = j_strlen(Dest);

strncmp(Dest, Str2, v2);

if( sub_411127() )

sub_41132F("wrong flag!\n", v4);

else

sub_41132F("right flag!\n", v4);

```

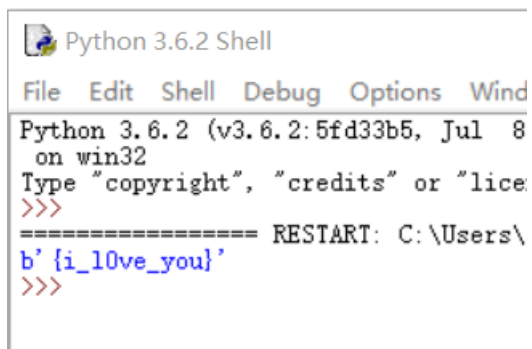
```
sub_41126C(&savedregs,&dword_415890);  
  
sub_411280();  
  
return sub_411127();  
  
}
```

分析可知:将输入的串Str1先进行base64加密 再与串Str2比较 若相等 则输出"right flag"

由此,我们只需将Str2也就是"e3nifIH9b\_C@n@dH"进行解密即可

Python脚本:

```
import base64  
  
s ="e3nifIH9b_C@n@dH"  
  
flag =""  
  
for i in range(len(s)):  
  
flag += chr(ord(s[i])- i)  
  
flag = base64.b64decode(flag)  
  
print(flag)
```



```
Python 3.6.2 Shell  
File Edit Shell Debug Options Wind  
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8  
on win32  
Type "copyright", "credits" or "lice  
>>>  
===== RESTART: C:\Users\  
b' [i_love_you]'  
>>>
```

所以得到flag{i\_love\_you}