CTF—逆向入门题目(超详细)



 Thunder J
 丁 2018-09-13 18:56:50 发布
 63081
 收藏 382

 分类专栏:
 题目篇 文章标签:
 CTF 逆向入门

 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/CharlesGodX/article/details/82692953

 版权

<u>
 题目篇 专栏收录该内容</u>
 <u>
 题目第 专栏收录该内容</u>
 14 篇文章 1 订阅
 订阅专栏

0x00: 介绍

以下为一些简单的Windows逆向入门题目,帮助一些刚接触逆向又无法下手的朋友,如果对安卓逆向感兴趣的 朋友可以看一下我的这一篇安卓逆向入门题目哦: https://blog.csdn.net/CharlesGodX/article/details/86602958

0x01:题目

1.Bugkuctf平台中的逆向题easy_vb:

打开文件发现需要输入注册码获取flag

🖏 QCTF	- [×
Welcome to QCTF			
	0 🔻	0	•
确 定			
https://blog.csdn.	1irage Team	rles	GodX

话不多说先放入PEID看看,养成这个好习惯,发现是用VB6写的

🌃 PEiD vO).95			_	×	
File: C:\Us	ers\xsj\Desktop\e	easy_vb.exe				
Entrypoint:	00001238		EP Section:	.text	>	
File Offset:	00001238	j	First Bytes:	68,9C,15,40	>	
Linker Info:	6.0		Subsystem:	Win32 GUI	>	
Microsoft Visual Basic 5.0 / 6.0						
Multi Scan Task Viewer Options About Exit						
▼ <u>Stay on top</u> https://blog.csdn.net/Charl?s600					Gorðxi	

我们载入IDA进行分析,用alt+t搜索字符串CTF,然后crtl+t搜索下一个字符串,直到看到flag .text:0040238F fnclex .text:00402391 jge short loc_4023A5 . .text:00402393 push 0A 0h • .text:00402398 offset dword_401A48 push . .text:0040239D push edi • .text:0040239E push eax . .text:0040239F call ds:__vbaHresultCheckObj .text:004023A5 ; CODE XREF: .text:004023911j .text:004023A5 loc_4023A5: . .text:004023A5 eax, [ebp-18h] mov .text:004023A8 push eax offset aMctf_n3t_rev_1 ; "MCTF{_N3t_Rev_1s_E4ay_}" .text:004023A9 push .text:004023AE call ds:__vbaStrCmp • .text:004023B4 mov edi, eax . .text:004023B6 lea ecx, [ebp-18h] . .text:004023B9 neg edi .text:004023BB edi, edi sbb .text:004023BD inc edi .text:004023BE neg edi . Tuest de a 0.51

2.Bugkuctf平台中的逆向题Easy_Re:

先把文件下载下来载入PEID

Weid v0.94		_	\times
文件: C:\Users\xsj\Deskto	p\re1. exe		
入口点: 000014A5	EP 段:	. text	$\left \right>$
文件偏移: 000008A5	首字节:	E8, 0A, 3E, 00	$\left \right>$
连接器版本: 12.0	子系统:	Win32 console	>
PESniffer: Unknown Microsoft Visual Studio .	NET 2005 — 2008 -> Mi	crosoft Corpora) tion
多文件扫描(M) 查看进程(I) 选项(0) 关于	F(<u>A</u>) 退出	1
☑总在最前(S) 欢迎访问 http://bb	看雪学院论坛 s.pediy.com/g.csdn.	net/C <mark>扩展信息</mark>	()

运行文件发现有字符串flag,于是考虑用IDA打开文件用alt+F12查找字符串flag

		1.21	1.1	
欢迎ヲ	F到DUT(CTF呦		
这是一	一道很可	「爱很的	育单的逆	向题呦
输入f	lag吧:			
1		/1 1		
h	ittps:/	/blog.	csdn.ne	et/CharlesGodX

来到这里发现xmmword后面有两串奇怪的字符串,我们将其选中按R键将其变成字符串发现flag

•	nan	db '1#SNAN',0 ; DATA XREF: sub_40F02C+C71o
•		align 4
•	nd	db '1#IND',0 ; DATA XREF: sub_40F02C+E0îo
•		align 4
•	har a1Inf[]	
	nf	db '1#INF',0 ; DATA XREF: sub_40F02C+EFîo
•		align 4
•	har a1Qnan[]	
	nan	db '1#QNAN',0 ; DATA XREF: sub_40F02C:loc_40F13Cîo
•		align 4
•	word_413E34	xmmword 3074656D30633165577B465443545544h
		; DATA XREF: sub_401000+101r
•	rd_413E44	dq 7D465443545544h ; DATA XREF: sub_401000+27îr
•	utctf	db '欢迎来到DUTCTF呦',0Ah,0 ; DATA XREF: sub_401000+1Aîo
•		align 10h
•	_413E60	db 0D5h ; ; DATA XREF: sub_401000+3Dîo
•		db 0E2h ;
		db OCAh ;
		db 0C7h ;
		db 0D2h ;
1		db 088h ;
1		db 085h ;
1		db 0C0h; https://blog.csdp.net/CharlesGo
1		db uBAh ; https://biog.obduk.net/ondriebook

3.南邮CTF逆向题Hello,RE!

下载文件用PEID载入,无壳,运行一下发现让输入flag,老办法用IDA打开查找字符串flag



查找到之后用f5查看伪代码

```
тва
                                                 ax, [esp+rin]
.text:00401576
                                      mov
                                               [esp], eax
                                                                   : char *
.text:00401579
                                      call
                                                strcmp
                                                eax, eax
.text:0040157E
                                      test
.text:00401580
                                      jz
                                                short loc 401590
                                                dword ptr [esp], offset aFlagA ; "flag错
.text:00401582
                                      mov
.text:00401589
                                      call
                                                  Z6printfPKcz
                                                                   ; printf(char const*,...)
.text:0040158E
                                                short loc_401592
                                      jmp
.text:00401590
.text:00401590
.text:00401590 loc_401590:
                                                                    ; CODE XREF: _main+801j
                                                short loc_4015B0
.text:00401590
                                      jmp
.text:00401592 ;
.text:00401592
.text:00401592 loc_401592:
                                                                    ; CODE XREF: _main+681j
                                                                    ; _main+8Eîj
.text:00401592
.text:00401592
                                               eax, [esp+11h]
                                      lea
.text:00401596
                                                [esp+4], eax
                                      mov
                                                dword ptr [esp], offset format ; "%s"
.text:0040159A
                                      mov
.text:004015A1
                                                  Z5scanfPKcz
                                      call
                                                                   ; scanf(char const*,...)
.text:004015A6
                                                eax, ØFFFFFFFFh
                                      CMP
.text:004015A9
                                      setnz
                                                al
.text:004015AC
                                      test
                                                al, al
.text:004015AE
                                                short loc 40156A
                                      inz
.text:004015B0
.text:004015B0 loc_4015B0:
                                                                    ; CODE XREF: _main:loc_4015901j
                                               dword ptr [esp], offset aFlag ; "flag
__Z6printfPKcz ; printf(char const*,...)
dword ptr [esp], offset aC ; "如
__Z6printfPKcz ; printf(char const*,...)
dword ptr [esp], offset aCtf_nuptsast_C ; "群号在ctf.nuptsast.com的to 16级新
Z60r0ffEPKcz : printf(char const*)
.text:004015B0
                                      mov
.text:004015B7
                                      call
.text:004015BC
                                      mov
.text:004015C3
                                      call
.text:004015C8
                                      MOV
                                                _____Z6printfPKcz ; printf(char const*,...)
dword ptr [esp], offset asc_41008F ; "很期
.text:004015CF
                                      call
.text:004015D4
.text:004015DB
                                      mov
                                                  Z6printfPKcz ; printf(char const*,...)
                                      call
.text:004015E0
                                                getchar
                                      call
.text:004015E5
                                      call
                                                 getchar
.text:004015EA
                                      mov
                                                eax, Ø
.text:004015EF
                                      leave
                                      Potn
```

```
看到如下结果,同样将v5~v11的结果用R改为字符串得到flag
      main();
   printf("请输入flag: ");
   v5 = 'galf';
   v6 = 'leW{';
                                                      I
   v7 = 'emoc';
v8 = '_oT_';
   v9 = 'W_ER';
   v10 = 'dlro';
   v11 = '}!';
   v12 = 0;
  vhile ( scanf("%s", v4) != -1 && strcmp(v4, (const char *)&v5) )
printf("flag错误。再试试? \n");
printf("flag正确。\n");
printf("如果是南邮16级新生并且感觉自己喜欢逆向的话记得加群\n");
printf("都号在ctf.nuptsast.com的to 16级新生页面里\n");
printf("很具待遇见喜欢re的新生23333\n");
optchar():
   getchar();
   getchar();
   return 0;
}
```

4.实验吧 Just Click

下载文件用exeinfo这款软件查看发现程序用C#撰写



打开软件发现需要点击相应的数字才能发现flag

III MainWindow		_		×
What do	you want to input?			
[button1			
E	button2			
Ε	button3			
[button4			
	https://blog.cs	dn.net/	Charle	sGodX

因为是用C#写的所以我们考虑用Reflector软件将其打开



找到MainWindow发现类似主函数的东西,分析发现需要按顺序点击8次就能出现flag int[] **numArray** = new int[] {0, 1, 3, 4, 2, 1, 2, 3, 4};

按这个顺序点击即出现flag。

5.南邮CTF py交易

链接: https://pan.baidu.com/s/1o8fVxkl密码: kd37

下载文件发现是pyc格式,我们直接在网上找在线反编译python的网站: https://tool.lu/pyc/

反编译后发现是这样的

```
#!/usr/bin/env python
# visit http://tool.lu/pyc/ for more information
import base64
def encode(message):
    5 = ''
    for i in message:
         x = ord(i) \wedge 32
         \mathbf{x} = \mathbf{x} + \mathbf{16}
         s += chr(x)
    return base64.b64encode(s)
correct = 'X1NkVmtUI1MgXWBZXCFeKY+AaXNt'
flag = ''
print 'Input flag:'
flag = raw_input()
if encode(flag) == correct:
    print 'correct'
else:
    print 'wrong'
```

分析算法: 首先输入一段字符串, 进入encode函数之后与字符串correct进行比较

encode函数就是将输入的字符串中每个字符ascii都与32进行异或运算,然后每个在加上16得到新的字符串,最 后再将这个字符

串进行base64加密。

所以我们只需将"XINkVmtUI1MgXWBZXCFeKY+AaXNt"进行base64解密,再将每个字符ascii码都减16,接着与 32异或即可得

到flag

python代码如下:

```
import base64
correct ='XlNkVmtUI1MgXWBZXCFeKY+AaXNt'
s = base64.b64decode(correct)
flag =''
for i in s:
i = chr((ord(i)-16)^32)
flag += i
print flag
```

运行即可得到flag: nctf{d3c0mpil1n9_PyC}

6.Jarvis OJ :FindKey

下载文件发现是一个名字比较长的东西(大多数题目后缀名都比较长)



用一款叫做斯托夫文件格式分析器分析一下这个软件的类型

可能性	类型说明	扩展名
100.0%	Python optimized code	.PYO
	nttps://biog.csun.net/tna	LIESGOUY

发现是python写的,将其后缀名改为.pyc然后放入在线反编译网站里得到如下

```
import sys
```

```
lookup = [
196,153, 149,206, 17,221, 10, 217, 167, 18, 36, 135, 103, 61, 111, 31, 92, 152, 21, 228, 105, 191, 173, 41,
pwda = [188, 155, 11, 58, 251, 208, 204, 202, 150, 120, 206, 237, 114, 92, 126, 6, 42]
pwdb = [53, 222, 230, 35, 67, 248, 226, 216, 17, 209, 32, 2, 181, 200, 171, 60, 108]
flag = raw_input('Input your Key:').strip()
if len(flag) != 17:
    print 'Wrong Key!!'
    sys.exit(1)
flag = flag[::-1]
for i in range(0, len(flag)):
    if ord(flag[i]) + pwda[i] & 255 != lookup[i + pwdb[i]]:
        print 'Wrong Key!!'
        sys.exit(1)
print 'Congratulations!!'
```

下面写个脚本满足输出flag的条件就ok了

```
import sys
lookup = [
196,153, 149,206, 17,221, 10, 217, 167, 18, 36, 135, 103, 61, 111, 31, 92, 152, 21, 228, 105, 191, 173, 41,
pwda = [188, 155, 11, 58, 251, 208, 204, 202, 150, 120, 206, 237, 114, 92, 126, 6, 42]
pwdb = [53, 222, 230, 35, 67, 248, 226, 216, 17, 209, 32, 2, 181, 200, 171, 60, 108]
flag = ''
for i in range(0,17): //这里就是要满足wrong key的条件才能得到正确的flag
    flag+=chr(lookup[i + pwdb[i]]-pwda[i] & 255)
flag=flag[::-1]
print flag
```

运行一下就得到flag了

PCTF{PyC_Cr4ck3r} son.net/CharlesGodX

7.Jarvis OJ : stheasy

拿到题目下载了一个很复杂的文件,我们先放入斯托夫文件格式分析器分析,发现是ELF文件:

可能性	类型说明	扩展名
50.2%	ELF Executable and Linkable format (Linux)	
49.8%	ELF Executable and Linkable format (generic)	.0
	https://blog_csdp_pei	t/CharlesGodX
	100ps.//brog. osun. no	ty chur resolut

我们用IDA将其打开,很容易找到关键函数位置:

	.text:080486A0 main	proc ne	ar : DATA XREF: start+17îo
•	.text:080486A0	push	ebp
•	.text:080486A1	mov	ebp. esp
•	.text:080486A3	and	esp, ØFFFFFFØh
•	.text:080486A6	push	ebx
•	.text:080486A7	sub	esp. 11Ch
•	.text:080486AD	lea	ebx, [esp+10h]
•	.text:080486B1	mov	dword ptr [esp], offset format ; "Input flag:"
•	.text:080486B8	call	printf
•	.text:080486BD	mov	dword ptr [esp+4], 100h ; n
•	.text:080486C5	mov	[esp], ebx ; s
•	.text:080486C8	call	sub_80485A0
•	.text:080486CD	mov	[esp], ebx ; s
•	.text:080486D0	call	sub_8048630
•	.text:080486D5	test	al, al
2	.text:080486D7	jnz	short loc_80486F8
•	.text:080486D9	mov	dword ptr [esp], offset s ; "Flag is wrong."
•	.text:080486E0	call	_puts
•	.text:080486E5	add	esp, 11Ch
•	.text:080486EB	xor	eax, eax
•	.text:080486ED	рор	ebx
•	.text:080486EE	mov	esp, ebp
•	.text:080486F0	рор	ebp
•	.text:080486F1	retn	
	.text:080486F1 ;		
•	.text:080486F2	align 8	
	.text:080486F8		
	.text:080486F8 loc_80486F8:		; CODE XREF: main+37îj
	.text:080486F8	mov	dword ptr [esp], offset aFlagIsRight_ ; "Flag is right."
	.text:080486FF	call	_puts
	.text:08048704	add	esp, 11Ch
	.text:0804870A	xor	eax, eax
	.text:0804870C	рор	ebx
	.text:0804870D	mov	esp, ebp
	.text:0804870F	рор	ebp https://hlog.ogdp.pot
	.text:08048710	retn	nttps://biog.csdn.net
	tout.00060740 main	anda	

按下F5编译一下,观察到如下函数:

---- --- --~---1 int __cdecl main() 2 (3 int result; // eax@2 int v1; // [sp+10h] [bp-110h]@1 4 5 printf("Input flag:"); sub_80485A0(&v1, 0x100u); 1 6 7 ı. 1 8 if (sub_8048630(&v1)) 9 -{ 10 puts("Flag is right."); result = 0; 111 12 } else 13 14 - { puts("Flag is wrong."); 115 16 result = 0; 17 > 18 return result; 19

有一个sub_8048630函数决定了Flag的对错,所以我们只需要研究一下它:

```
1
                       IDA VIEW A
                                            L 4:
 1 int __cdecl sub_8048630(char *s)
 2 {
    size_t v1; // eax@2
 3
 4
    int v3; // edx@5
 5
 б
    if ( 5 )
 7
    {
      v1 = strlen(s);
 8
 9
      if ( U1 )
10
      {
        if ( v1 == 29 )
11
12
        {
          V3 = 0;
13
          while ( s[v3] == a[(b[v3] / 3u - 2)] )
14
15
          {
16
            if ( ++03 == 29 )
              return 1;
17
18
          }
19
        }
20
      }
21
    }
22
    return 0;
23 }
```

这里我为了便于观察重新命名了a,b函数,我们双击a和b查找一下他们具体的值,将a这两排选中用shift + E快 捷键选择第四个选项,用数组表示a如下,b同理:

 .data:08049AE0; char a[] .data:08049AE0; a .data:08049AE1; char a[] .data:08049AE1; char b[] .data:08049B15; char b[] .data:08049B15; b .data:08049B16; data:08049B16 .data:08049B16; data:08049B17 	db 6Ch ; DATA XREF: sub_8048630+531r sA db 'k2j9Gh}AgfYAds-a6QV1Hk5ER_T[cuLbU7n0m32eX{CHt8SZo]U',0 db 48h ; DATA XREF: sub_8048630:loc_80486681r db 5Dh ;] db 48h db 5Dh ; ;	 C unsigned char array (hex) C unsigned char array (decimal) initialized <u>C</u> variable raw bytes
- data: 06 04 90 16 - data: 08 04 98 19 - data: 08 04 98 10 - data: 08 04 98 18 - data: 08 04 98 10 - data: 08 04 98 10 - data: 08 04 98 11	uu 241, ,	Save data to clipboard Preview [unsigned char ida_chars[] =
 data: 88,94981F data: 88,94982 data: 88,94982 data: 88,94982 data: 88,949823 data: 88,949823 data: 88,949824 data: 88,949825 data: 88,949825 data: 88,949826 data: 88,949826 data: 88,949826 	db 1Eh db 69h jb 7Eh jb 33h jb 15h db 72h jb 33h jb 15h db 33h jb 15h db 33h jb 15h db 33h jb 34h jb 25h jb 25h jb 25h jb 33h jb 24h jb 25h jb 25h <td< th=""><th>{ Ox6C, 0x6E, 0x32, 0x6A, 0x39, 0x47, 0x68, 0x7D, 0x41, 0x67, Ox66, 0x59, 0x34, 0x64, 0x73, 0x2D, 0x61, 0x36, 0x51, 0x57, 0x31, 0x23, 0x6B, 0x35, 0x45, 0x52, 0x5F, 0x54, 0x5B, 0x63, 0x76, 0x4C, 0x52, 0x55, 0x4F, 0x6D, 0x33, 0x5A, 0x55, 0x58, 0x7B, 0x43, 0x4D, 0x74, 0x38, 0x53, 0x5A, 0x6F, 0x5D, 0x55, 0x00 }; https://blog.csdn.net/CharlesGodD</th></td<>	{ Ox6C, 0x6E, 0x32, 0x6A, 0x39, 0x47, 0x68, 0x7D, 0x41, 0x67, Ox66, 0x59, 0x34, 0x64, 0x73, 0x2D, 0x61, 0x36, 0x51, 0x57, 0x31, 0x23, 0x6B, 0x35, 0x45, 0x52, 0x5F, 0x54, 0x5B, 0x63, 0x76, 0x4C, 0x52, 0x55, 0x4F, 0x6D, 0x33, 0x5A, 0x55, 0x58, 0x7B, 0x43, 0x4D, 0x74, 0x38, 0x53, 0x5A, 0x6F, 0x5D, 0x55, 0x00 }; https://blog.csdn.net/CharlesGodD

研究完算法之后就可以写脚本了:

```
a = [
 0x48, 0x5D, 0x8D, 0x24, 0x84, 0x27, 0x99, 0x9F, 0x54, 0x18,
 0x1E, 0x69, 0x7E, 0x33, 0x15, 0x72, 0x8D, 0x33, 0x24, 0x63,
 0x21, 0x54, 0x0C, 0x78, 0x78, 0x78, 0x78, 0x78, 0x1B
    1
b = [
 0x6C, 0x6B, 0x32, 0x6A, 0x39, 0x47, 0x68, 0x7D, 0x41, 0x67,
 0x66, 0x59, 0x34, 0x64, 0x73, 0x2D, 0x61, 0x36, 0x51, 0x57,
 0x31, 0x23, 0x6B, 0x35, 0x45, 0x52, 0x5F, 0x54, 0x5B, 0x63,
 0x76, 0x4C, 0x62, 0x56, 0x37, 0x6E, 0x4F, 0x6D, 0x33, 0x5A,
 0x65, 0x58, 0x7B, 0x43, 0x4D, 0x74, 0x38, 0x53, 0x5A, 0x6F,
 0x5D, 0x55, 0x00
     1
flag = ''
c = []
for i in range(0,len(a)):
   c.append(a[i]/3-2)
                       //append() 方法用于在列表末尾添加新的对象
                         //将数据转换为整形,不转换会出错
   c[i] = int(c[i])
for j in range(0,len(a)):
   flag += chr(b[c[j]])
print(flag)
```

最后运行得到Flag:



0x02:总结

上面仅仅是一些入门的题目,如果是新手的话先把这些题目弄懂,弄透。熟悉各种工具的使用,不断的总结, 逆向最重要的是分析,要自己多去分析。



<u>创作打卡挑战赛</u> 赢取流量/现金/CSDN周边激励大奖