逆向-360逆向writeup

原创

Nightsay 于 2015-04-09 11:49:49 发布 2016 收藏 2

分类专栏： 逆向分析 文章标签： 逆向

本文链接：https://blog.csdn.net/Nightsay/article/details/44957863

Crackme逆向分析

逆向第一题，很简单，分分钟能搞定。题目大致是说找一个key，提示success就行。

Peid查一下没有任何壳，是vc写的程序，里面没有知名的算法。开始分析

错误的时候是这个样子

首先这一点就能给我们以下几点提示：

1此程序是vc写的，那么得到用户输入肯定是用到getdigitemtext系列函数，通过对函数下断即可以得到用户的输入

2.比较字符串，提示错误的字符串为done，查找字符串

3.失败与成功都会弹出对话框，那么同理也可也对messagebox系列函数下断

4.……其他 的方法暂且不说

直接getdigitemtext下断发现断不下，嗯嗯，后来查看程序的时候发现这个程序并不是调用了这个函数，而是他的同类型函数GetWindowTextA,下断点之后，截断了用户的输入来到这个函数开头：

```
0042BAA3  |. 8B4D 10        mov ecx,[arg.3]
0042BAA6  |.  6AFF          push -0x1
0042BAA8  |.  E8846BFFFF    call CrackMe.00422631
0042BAAD  |.  EB0B          jmp XCrackMe.0042BABA
0042BAAF  |> 8B45 10        mov eax,[arg.3]
0042BAB2  |. FF30           push dword ptrds:[eax]
0042BAB4  |. 56             push esi
0042BAB5  |.  E8E4EBFFFF    call CrackMe.0042A69E
0042BABA  |> 5F             pop edi
0042BABB  |. 5E             pop esi
0042BABC  |. 5D             pop ebp
0042BABD  \.  C20C00        retn 0xC
，嗯嗯，直接单步掉，过几个跳转之后来到算法的地方：
00401420   .  6AFF          push -0x1
00401422   .  68D0904300    push CrackMe.004390D0                ;  SE 处理程序安装
00401427   . 64:A1 0000000>mov eax,dword ptr fs:[0]              ;  关键代码
0040142D   . 50             push eax
0040142E   . 64:8925 00000>mov dword ptr fs:[0],esp
00401435   . 83EC 0C        sub esp,0xC                          ;  保护现场
00401438   . 56             push esi
00401439   . 57             push edi
0040143A   . 8BF9           mov edi,ecx
```

```
0040143C   .  6A01         push 0x1
0040143E   .  E8FD3E0200   call CrackMe.00425340                  ;  读字符串给edx
00401443   .  6A00         push 0x0
00401445   . 8D4F 5C       lea ecx,dword ptrds:[edi+0x5C]
00401448   .  E895110200   call CrackMe.004225E2                  ;  读输入字符串给eax
0040144D   . 51            push ecx
0040144E   . 8BF0          mov esi,eax
00401450   . 8BCC          mov ecx,esp
00401452   . 896424 14     mov dword ptrss:[esp+0x14],esp
00401456   .  68CCA04400   push CrackMe.0044A0CC                  ;  strawberry
0040145B   .  E8C00D0200   call CrackMe.00422220
00401460   .  E81BFFFFFF   call CrackMe.00401380                  ;  读入一系列key做运算
00401465   . 83C4 04       add esp,0x4
00401468   . 8D4C24 0C     lea ecx,dword ptrss:[esp+0xC]
0040146C   . 50            push eax
0040146D   .  E8AE0D0200   call CrackMe.00422220
00401472   . 8D4C24 0C     lea ecx,dword ptrss:[esp+0xC]
00401476   . C74424 1C 000>mov dword ptr ss:[esp+0x1C],0x0
0040147E   .  E8CF120200   call CrackMe.00422752
00401483   .  6A00         push 0x0
00401485   . 8D4C24 10     lea ecx,dword ptrss:[esp+0x10]
00401489   .  E854110200   call CrackMe.004225E2
0040148E   . 50            push eax
0040148F   . 8D4C24 0C     lea ecx,dword ptrss:[esp+0xC]
00401493   . C640 09 4B    mov byte ptrds:[eax+0x9],0x4B
00401497   .  E8840D0200   call CrackMe.00422220
0040149C   . 8D4C24 08     lea ecx,dword ptrss:[esp+0x8]
004014A0   . C64424 1C 01  mov byte ptrss:[esp+0x1C],0x1
004014A5   .  E8A8120200   call CrackMe.00422752
004014AA   .  6A00         push 0x0
004014AC   . 8D4C24 0C     lea ecx,dword ptrss:[esp+0xC]
004014B0   .  E82D110200   call CrackMe.004225E2
004014B5   >  8A10          mov dl,byte ptr ds:[eax]               ;  判断成功的代码
004014B7   .  8ACA          mov cl,dl                              ;  查看比较就知道是上面出现过的key
004014B9   .  3A16          cmp dl,byte ptr ds:[esi]
004014BB   .  75 1C         jnz XCrackMe.004014D9
004014BD   .  84C9          test cl,cl
004014BF   .  74 14         je XCrackMe.004014D5
004014C1   .  8A50 01       mov dl,byte ptr ds:[eax+0x1]
004014C4   .  8ACA          mov cl,dl
004014C6   .  3A56 01       cmp dl,byte ptr ds:[esi+0x1]
004014C9   .  75 0E         jnz XCrackMe.004014D9
004014CB   .  83C0 02       add eax,0x2
004014CE   .  83C6 02       add esi,0x2
004014D1   .  84C9          test cl,cl
004014D3   .^ 75 E0         jnz XCrackMe.004014B5
004014D5   >  33C0          xor eax,eax
004014D7   .  EB 05         jmp XCrackMe.004014DE
004014D9   >  1BC0          sbb eax,eax
004014DB   .  83D8 FF       sbb eax,-0x1
004014DE   >  85C0          test eax,eax
004014E0   .  6A 00         push 0x0
004014E2   .  6A 00         push 0x0
004014E4   .  75 07         jnz XCrackMe.004014ED
004014E6   .  68 E0A04400   push CrackMe.0044A0E0                  ;  success
004014EB   .  EB 05         jmp XCrackMe.004014F2                  ;  破解完成
004014ED   >  68 D8A04400   push CrackMe.0044A0D8                  ;  Done
```

破解过程直接看注释吧，比较累，所以写的比较简单，没有仔细分析算法。
嗯嗯，破到这个地方，看看堆栈，发现暂停堆栈分析也能秒破这个程序

```
0018F830   00422761  CrackMe.00422761
0018F834   01D3D8E8  ASCII "K$q*a_+@Xt"
0018F838   01D3D848  ASCII "123456"
0018F83C   00000000
0018F840   0018FE6C
0018F844   0043B890  CrackMe.0043B890
0018F848   01D3D8E8  ASCII "K$q*a_+@Xt"
0018F84C   01D3D898  ASCII "tX@+_a*q$K"
```

分析结束，大牛勿喷！

Reverseme逆向分析

题目只记得大致意思，是说有一个文件.db被这个程序加密了，要想得到key就要么逆向这个加密程序的算法，然后自己写解密程序解密，要么用这个软件修复里面的bug运行两次就可以得到key（坑爹的提示，反正我没有运行两次就整出来了）
同样peid查出无壳，没有加过壳，运行这个程序：

嗯嗯，根据提示，肯定是里面某个函数传参出了问题。
OD载入，刚入口就有这个messagebox，一开始参数出现了错误。

```
0040103D>/$  68 10304000   push ReverseM.00403010
00401042      6A 01        push 0x1
00401044  |.  E873010000   call ReverseM.004011BC
00401049  |. 83F8 01       cmp eax,0x1
0040104C  |.  7414         je XReverseM.00401062
0040104E  |.  6A00         push 0x0                                 ; /Style =MB_OK|MB_APPLMODAL
00401050  |.  681D344000   push ReverseM.0040341D                   ; |Title = "提示"
00401055  |.  6810344000   push ReverseM.00403410                   ; |Text = "输入参数错误"
0040105A  |.  6A00         push 0x0                                 ; |hOwner =NULL
0040105C  |.  E847010000   call<jmp.&user32.MessageBoxA>           ; \MessageBoxA
```

跟进Reverse,.004011BC,发现这个函数实际上是在读这个软件的路径，但是没有成功。（算法分析过，结果最近手残OD清空过一次，算法没了，有兴趣的可以自己下去分析，）用IDA分析的算法差不多是下面这个样子：

将读出的路径写到00403010去，这里管不管都行，，因为程序的大致思路是在下面的readfile读出来，然后在write回去。加上pass.db把参数push0x1改为 push 0x0（这个时候会读reverseme.exe）。
单步到这个地方：

```
00401089  |.  6A00         push 0x0                                 ;/pFileSizeHigh = NULL
0040108B  |. FF35 00304000 push dword ptr ds:[0x403000]             ; |hFile = 0000005C
00401091  |.  E8EE000000   call<jmp.&kernel32.GetFileSize>         ; \GetFileSize
00401096  |.  A304304000   mov dword ptrds:[0x403004],eax
0040109B  |.  6A04         push 0x4                                 ; /Protect = PAGE_READWRITE
0040109D  |.  6800100000   push 0x1000                              ; |AllocationType= MEM_COMMIT
004010A2      FF35 FA2F4000 push dword ptrds:[0x402FFA]
004010A8  |.  6A00         push 0x0                                 ; |Address = NULL
004010AA  |.  E8E1000000   call<jmp.&kernel32.VirtualAlloc>         ; \VirtualAlloc
```

发现virtualalloc在分配内存的时候分配了0字节（关于这个api不懂的可以百度或者msdn），往上面瞅瞅，刚好有GetFileSize，将参数传递给他，然后绕过这个bug
然后哈，文件就可以正常的读进来了

（这个地方其实是已经得到的正确的key，貌似之前读进来的是一堆asc值好像，记不太清了）同理，绕过下面的bug，然后key就是刚刚那个了。
关于程序入口处第一个messagebox，是要修改跳转的，如果不修改跳转，而把参数0x1改为0x0,发现读进来的是

然后messagebox是没有异常存在了，但是下面那个分配内存的地方还是会出错，然后修改一下，在createfile和readfile之后文件夹中多了这么一个文件

Winhex分析之后实际上就是这个.exe的代码

熟悉的PE结构，然后下面写之后变成这样

哈，分析上面的之后加密算法就可以找出来了。然后找出来之后同样可以对.db解密。
至此，分析完毕，大牛勿喷！

inject

第三个程序：inject：
依然没有壳，想到病毒，第一反应是PE结构，然后也神马都不懂的就winhex了一下，结果啥子都不晓得。管他，运行一下，尼玛啊，啥都没有。看看资源管理器，并没有inject的进程。估计是被玩坏了的样子。
OD载入一下，没的啥思路，看了看字符串，嗯？

提示啊，来到这个函数地方
开始没搞懂这个是神马杰宝，代码并没有分析完全，然后分析了一下代码弄成这个样子：

```
0043E005   .  6A00         push 0x0
0043E007   .  685BE04300   push Inject.0043E05B                 ;  Key?NO!
0043E00C   .  682BE04300   push Inject.0043E02B                 ;  A02B8DE7F2BCD4ED
0043E011   .  6A 00        push 0x0
0043E013   .  E8F909A775   call kernel32.75EAEA11
0043E018   .  6A00         push 0x0
0043E01A      E8            db E8
0043E01B      47            db 47                              ;  CHAR 'G'
0043E01C      62            db 62                              ;  CHAR 'b'
0043E01D      08            db 08
0043E01E      75            db 75
```

下面的OD还是没法分析，不过同样是call kernel32.75EAEA11（机器码一样）
在call的时候又一个A02B……的字符串，不过不是key，然后程序因为被病毒感染，所以IAT应该受到破坏了，一运行程序就自己蹦掉
了。

然后想想看还有木有其他办法：
程序从开头直接跳到假码，然后就直接崩了，所以直接把jmp给nop掉
发现了一些很有特征的函数，GetVersion,GetCommandLine……，擦，c程序的入口啊，尼玛跑下来，估计然后查一下字符串，发现了另外一个

坑爹啊，这货就是！！！，但是是在sendmessagebox后面，估计会弹出来个什么框框吧。
最坑爹的事情来了，尼玛nop掉之后我还在程序里面想怎么让他弹出个框，想了好久，每次还是会让这个程序崩掉，后来觉得每次改一遍nop麻烦，就直接先把nop保存下来，然后无聊点了一下保存的文件，靠，直接弹框啊

没道理啊，嗯，要去吃饭了，没怎么弄明白为什么保存之后程序可以正常运行了
，看这个样子估计是程序里面有反调试吧，在OD环境下不会正常运行。
分析完毕，大牛勿喷！！！

第4题题目忘记了

很遗憾当时第4题因为在web上被卡住了未能做出来。第4道题转自我的队友：

下午做了一下，这一题共有两个文件

那个exe打开后如下图

看到那个图片了么，还有一串像key的数字，我猜那串数字一定不是key，而且也不是隐写的题，要不然怎么放逆向这边了呢，所以咱还是要分析这个程序，经过几番尝试，知道源文件编辑框中输入上边这张图，数据文件是你要加入的数据，用txt保存，目的文件是将以上两个文件组合成第三个文件，所以这个目的文件也要是bmp文件。
点按钮“开始隐藏”，会弹出“隐藏信息完毕！”的对话框，可这怎么得到key呢，我们只有从图片中提取key。
用IDA看看，

发现不但有“隐藏信息完毕！”，还有“提取信息完毕！”，找到引用该字符串的代码，在401DD2处，向上翻，

有两个CreateFile，然后又两个ReadFile,又有一个WriteFile,发现CreateFile中的Filename是空的，应该就是那两个特别小的编辑框了，用PE-Explorer将编辑框放大。

就可以输入文件名了，可是怎么到达这处代码呢，看“开始隐藏”,
这部分功能代码与提取信息的那部分代码相似。

具体就不分析了，因为无关紧要，直接改代码，使它跳转到提取信息的代码处

保存运行：

这时候2.txt中就有key了，“B0ACD1BDA3FD1CD6”。