# lab4-report

## PB15111662 李双利

本次实验是用ollydbg对crackme.exe程序进行反汇编,通过反汇编后的代码找到程序中登录认证的入口,并且修改部分汇编代码从而绕过认证,并且保存输出修改后的程序,从而破解crackme.exe。

此次我的实验尝试过程如下:

## • 程序入口追踪

首先,这个程序是选择register后会弹出一个界面输入Name和Serial,然后进行判断是否匹配,所以想到在汇编代码中追踪该输入框的调用地址,在ollydbg中用 ctrl + N 调出name选项的窗口,通过这个窗口可以看到程序中所有调用函数的名称,并且双击可以跳转到函数所在地址:

004031A4 .idata	Import	&USER32.LoadBitmapA
004031A8 .idata	Import	&USER32.SetFocus
004031AC .idata	Import	&USER32.MessageBoxA
004031B0 .idata	Import	&USER32.PostQuitMessage
004031B4 .idata	Import	&USER32.WinHelpA

#### 跳转到相关地址:

```
0040134D | _ $
                                PUSH
                                                                                    Type = MB_OK|MB_ICONEXCLAMATION|MB_DEFBUTTON1|MB_APPLMOT
               68 <u>29214000</u>
68 <u>34214000</u>
                                PUSH OFFSET 00402129
0040134F
                                                                                   Caption = "Good work!"

Text = "Great work. mate! Now tru the next CrackMe!"
                                      OFFSET 00402134
00401354
                                PUSH
                                PUSH DWORD PTR SS:[EBP+8]

CALL <JMP.&USER32.MessageB
                                                                                   hOwner => [ARG.EBP+8]
                FF75 08
0040135C
                E8 D9000000
                                                                                  USER32. Mes
00401361
                ۲٦
00401362 
                                                                                  Type = MB_0K
USER32.Messag
               6A 00
00401364
                E8 AD000000
                                      <JMP.&USER32.MessageBeep>
                                      30
OFFSET 00402160
00401369
                6A 30
                                                                                  Type = MB_OK[MB_ICONEXCLAMATION|MB_DEFBUTTON1|MB_APPLMOT
                68 60214000
                                                                                   Caption = "No luck!"
Text = "No luck there, mate!"
0040136B
                                PUSH
00401370
                68 69214000
                                      OFFSET 00402169
                                PUSH
                                      DWORD PTR SS:[EBP+8]

<JMP.&USER32.MessageBoxA>
00401375
                                                                                   hOwner => [ARG.EBP+8]
00401378
                E8 BD000000
                                                                                  USER32.MessageBoxA
0040137D L
```

可以看到在 MessageBoxA 附近有字符串 Good work! 和 No luck! ,而这两个恰好就是我们在登录时候验证通过或者失败的输出提示信息。所以显然此处的两个函数是经过匹配比较之后分别跳到的两个地址。考虑继续往前追踪。

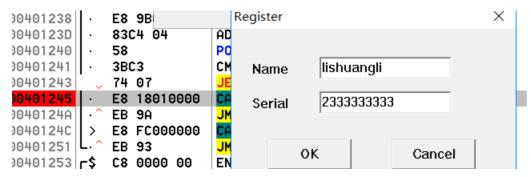
## • 匹配入口追踪

在上面其中一个函数的入口处在 Local call 选择 Go to Call from....,如下图:

Stack [0019FF80]=0 (current registers)
Imm=0



然后在跳转到的地方设一个断点,执行程序,进行如下输入:



这时可以看到输入的Name和Serial字符串的内容,进行了如下图所示的处理过程:

```
00401228 | -
             68 8E214000 | PUSH OFFSET 0040218E
                                                                   ASCII "LISHUANGLI"
0040122D ·
            E8 4C010000
                          CALL 0040137E
00401232
            50
                          PUSH EAX
00401233
                          PUSH OFFSET 0040217E
                                                                   ASCII "2333333333"
            68 7E214000
00401238
            E8 9B010000
                          CALL 004013D8
0040123D
            83C4 04
                          ADD ESP, 4
00401240
            58
                          POP EAX
00401241 .
             3BC3
                          CMP EAX, EBX
          V 74 07
00401243
                          JE SHORT 0040124C
90401245
            E8 18010000
                          CALL 00401362
         ·^ EB 9A
0040124A
                          JMP SHORT 004011E6
0040124C | > E8 FC000000
00401251 | > EB 93
                          CALL 0040134D
                          JMP SHORT 004011E6
```

从上图可以看出,输入的两个字符串分别push之后然后分别调用两个函数进行处理,返回之后将内容放入 EAX和EBX两个寄存器中,随后又进行了EAX和EBX寄存器的比较,如果相等的话,会调用 0040134D 处的函数,也就是会提示 Great work! 的函数,否则会跳转到提示错误的函数。

有了以上信息之后,就可以判断问题的关键在于IE处的判断语句,如果能够做如下修改:

```
;JE SHORT 0040124C 修改为下面的指令。即JE修改为JMP
JMP SHORT 0040124C
```

这样就绕过了匹配的判断,无条件跳转到匹配成功的函数处!修改之后保存导出程序即可。

(修改后的程序命名为PB15111662\_CRACKME.exe)

## 实验总结

此次实验实现了用ollydbg工具对原程序进行反汇编,从而通过代码结构分析确定程序的核心部分,找到判断验证的函数以及语句,进行简单的修改即可绕过验证,实现了exe程序的简单破解。