另类方法攻破程序壁垒---充分利用程序验证的漏洞

作者:cntrump

工具:VB Decompiler Lite, OllyDbg

由于正在学习软件汉化相关的知识,遇到了需要汉化非标准资源的情况,于是在网上一搜,就找到了汉化新世纪网站,下载了 xxplus,该软件使用 VB 编译.该程序虽然已经停止更新了,但是功能依然够用,目前依然是汉化非标的首选辅助工具之一.

安装后先看程序说明,如果不注册,将会有好几个功能不能使用.对于这种情况,一般有两种解决方案,一是制作出注册机一劳永逸,二是通过爆破解除程序的功能限制.

试用注册一下程序,提示需要重启来验证注册码正确性,这种情况的话,要么是真的重启验证,要么是程序虚晃一枪,用来代替注册错误提示的障眼法.为了验证是哪一种情况,我们用 VB Decompiler 载入程序进行辅助分析,找到注册按钮处理事件的伪代码如下:

```
Private Sub OK_Click() '430C40
…略去无关代码
…
   loc_00430D76: mov var_6C, 0040E474h ; "Plase input user name!"
…
   loc_00430D9B: call [00401120h] ; MsgBox(arg_1, arg_2, arg_3, arg_4, arg_5)
…
   loc_00430DB3: call [00401070h] ; RegSetValueEx(%x1, %x2, %x3, %x4, %x5, %x6, %x7)
…中间略去了很多代码
   loc_00430E16: mov var_6C, 0040E4D8h ; "To check the register code, you need restart this software."
   loc_00430E1D: mov var_74, 00000008h
   loc_00430E24: call [0040130Ch] ; RegSetValueEx(%x1, %x2, %x3, %x4, %x5, %x6, %x7)
  …
   loc_00430E3B: call [00401120h] ; MsgBox(arg_1, arg_2, arg_3, arg_4, arg_5)
…
   loc_00430E53: call [00401070h] ; RegSetValueEx(%x1, %x2, %x3, %x4, %x5, %x6, %x7)
…
   loc_00430E6F: call [004012B4h] ; RegSetValueEx(%x1, %x2, %x3, %x4, %x5, %x6, %x7)
…
   loc_00430E88: call [0040112Ch] ; RegSetValueEx(%x1, %x2, %x3, %x4, %x5, %x6, %x7)
…
…
   loc_00430EA8: call [004010C8h] ; RegSetValueEx(%x1, %x2, %x3, %x4, %x5, %x6, %x7)
   loc_00430EAE: lea ecx, var_24
   loc_00430EB1: call [00401388h] ; RegSetValueEx(%x1, %x2, %x3, %x4, %x5, %x6, %x7)
…
…省略了后面的代码
```

很明显,程序只是把用户和注册码保存到注册表中然后提示重启程序进行验证就完事了.这样进行注册验证的过程就应该在主窗口的 from_load 事件中了,因为这个程序是使用自然编译的,不是 P-Code,跟踪起来就和普通的程序没有区别了,但是由于我不懂 VB,在跟踪的时候很快就迷失在了相似的指令中,VB 的函数比 API 要难记…

就在我准备要给作者汇钱要个注册码的时候,突然看到了程序的关于对话框,看到了红色的 Unregister version,直觉告诉我还有一线希望!在 VB Decompiler 中查看关于窗体:

```
VERSION 5.00
Begin VB.Form Form2
....省略无关部分
    Begin Label aRegInfo
       Caption = "Unregister version"
       ForeColor = &HFF&
       Left = 120
       Top = 1965
       Width = 3405
       Height = 255
    End
...
...

Attribute VB_Name = "关于对话框"
```

可以知道程序原始的文字就是 Unregister version,那么这个窗口的 form_load 过程中肯定存在注册验证代码,用 VB Decompiler 查看 from_load 代码如下:

```
Private Sub Form_Load() '423E90
...
...
  loc_00423ED0: xor ebx, ebx                ;ebx 清零,ebx=0
  loc_00423ED2: cmp [00439134h], bx         ; [00439134h]是一个全局变量,和 0 比较
  loc_00423ED9: mov var_18, ebx
  loc_00423EDC: mov var_1C, ebx
  loc_00423EDF: jz 00423F85h                ;如果[00439134h]为零就跳
  loc_00423EE5: mov edx, [esi]
  loc_00423EE7: push esi
  loc_00423EE8: call [edx+00000304h]
  loc_00423EEE: push eax
  loc_00423EEF: lea eax, var_1C
  loc_00423EF2: push eax
  loc_00423EF3: call [00401118h] ; Set (object)
  loc_00423EF9: mov ecx, [00439130h] ;
  loc_00423EFF: mov edi, eax
  loc_00423F01: push 0040DC0Ch ; "Register to:"      ;如果上面的跳转实现了,这里就不会执行
  loc_00423F06: push ecx
...
  loc_00423FBF: retn 0004h
```

很明显,窗体在加载的时候,会对一个全局变量进行判断,如果不是零,那么就显示 Register to:也就是注册给 xx 了,这说明,程序进行了注册验证之后会把结果保存到这个全局变量,成功为非零,失败为零,程序会在运行过程中判断这个全局变量,以确定当前运行模式是注册模式还是试用模式,按照这个思路,我们只要找出程序中对这个全局变量进行判断的地方,然后进行

修改,让程序以注册模式运行,那么我们的目的就达到了.

现在关键问题来了,如何找出全部的判断位置呢?如果有源码,我们可以直接搜索变量的出现的位置,但是现在想看源码是不可能的.别忘了我们还有神器 VB Decompiler,用它就可以,VB Decompiler 有一个可以说是很鸡肋的功能---字符串查找,听起来不错吧?查找字符串,很正常呀,OD 都有专门的字符串搜索插件,怎么能说这个功能是鸡肋呢?嗯,虽然表面看起来是这样的,但是它的功能和 OD 的字符串搜索插件完全不一样, VB Decompiler 的字符串查找功能是在代码窗口中查找文本,而不是找程序中的字符串!所以说是鸡肋一点也不过份,不过有一句怎么说来着?金子在需要它的人眼里才会发光!现在这个功能可是派上了大用场,能轻易解决我们现在的问题---查找所有 00439134h 变量出现的位置!

由于 VB Decompiler 反汇编代码的时候,是从当前选择的地址一直往下反汇编,直到代码段全部反汇编完为止,这也是为什么在反汇编时很慢的原因.既然 VB Decompiler 有这样的特点,那么肯定要为我们所用,只要从程序流程最开始的地方进行反汇编,那么整个程序的流程就会全部被反汇编在代码窗口中了!然后再利用它的字符串搜索功能,搜索 00439134h 这个字符串,不就能找到所有使用该变量的位置了吗?哈哈,我都要给我自己下跪了~~

我们从程序主窗口的 form_load 开始反汇编,然后搜索 00439134h:
查找到的结果如下:
之前一直没有用到 OD,其实 OD 的用处只有一个,那就是修改程序代码~~
第 1 处:

```
loc_0041675C: cmp word ptr [00439134h], 0000h
loc_00416764: jz 4167D4h                    ;nop 这句
```

第 2 处:

```
loc_00416DFC: cmp word ptr [00439134h], 0000h
loc_00416E04: jz 416E74h                    ;nop 这句
```

第 3 处:

```
loc_00417E83: cmp word ptr [00439134h], 0000h
loc_00417E8B: jz 00417F17h                  ;nop 这句
```

第 4 处:

```
loc_0041A62C: cmp [00439134h], di
loc_0041A633: jnz 0041A6F5h                 ;改为 jmp
```

第 5 处:

```
loc_0041AEF0: cmp [00439134h], bx
loc_0041AEF7: jnz 0041AFB9h                 ;改为 jmp
```

第 6 处:

```
loc_0041B71F: cmp [00439134h], bx
loc_0041B726: jnz 0041B7E8h                 ;改为 jmp
```

第 7 处:

```
loc_00423ED2: cmp [00439134h], bx
loc_00423ED9: mov var_18, ebx
loc_00423EDC: mov var_1C, ebx
loc_00423EDF: jz 00423F85h                  ;nop 这句
```

OK,现在保存文件,运行看看效果,发现程序的关于对话框已经变为 Register to:了,再看看试用版中不能使用的功能,也全部能使用了,至此,大功告成!用 dUP2 做一个补丁吧,下次再安装的时候直接打上补丁就行了.

如果去跟踪验证过程,那么将是和作者肉搏,但是程序只用一个全局变量来作为注册标志,

我们就可以智取,相比之下成本要小多了.
附一个补丁截图,由于是国产软件,补丁就不发了.