+Win764x下做掉PatchGuard教程

b 于 2017-03-22 20:51:53 发布 7065 收藏 12

Win764x下做掉PatchGuard教程

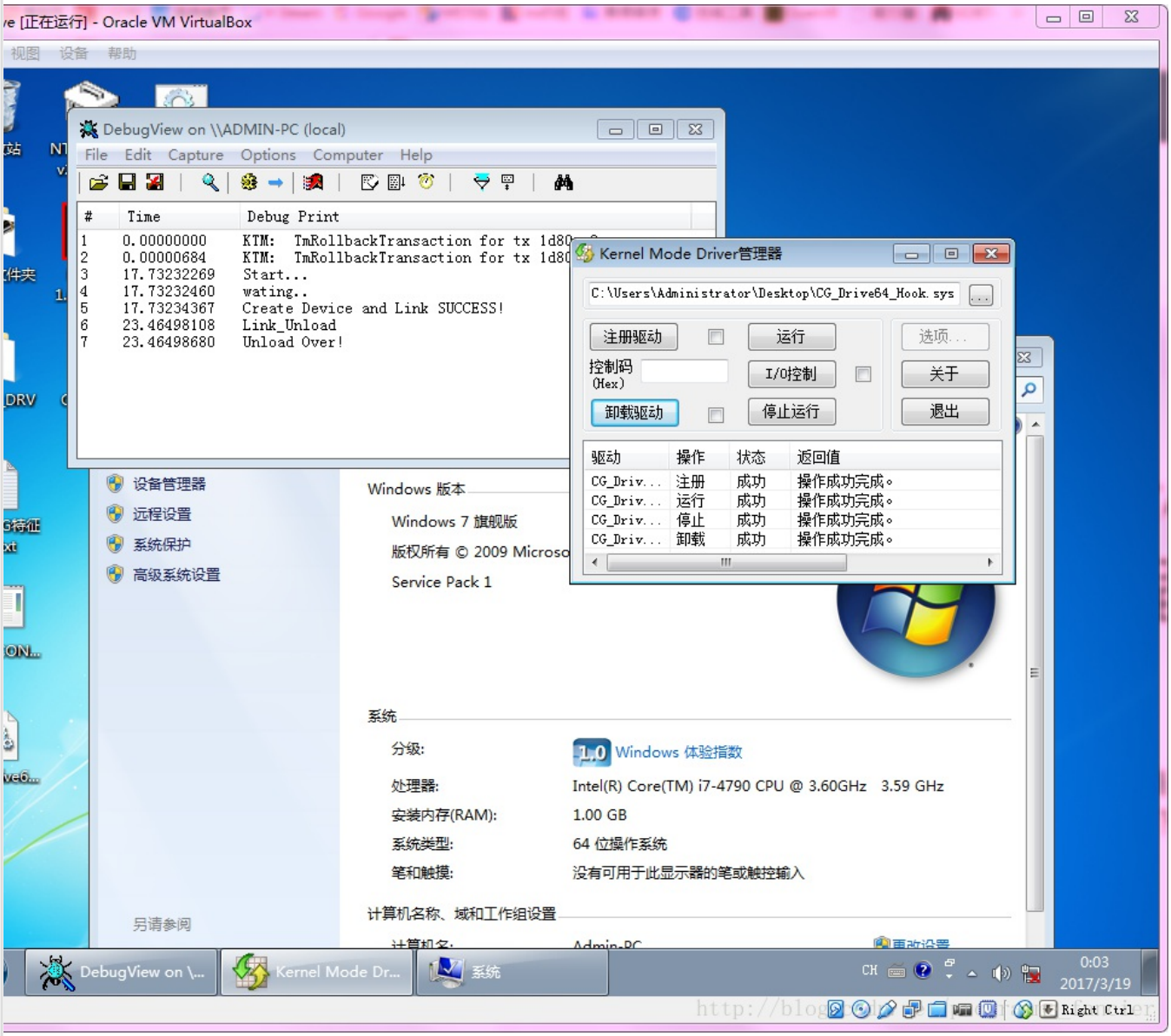
R3层干掉PG.我已经搞定很久了

直也没有发出来.

掉PG还有很多高深的办法,PG也不是那么难过掉的东西.我靠着一些资料埋头研究了半个月之久.成功的在win7 64位下干掉了PG.实现了

T HOOK以及绕驱动签名强制

都懂,没图你说个J8?上图



之前,有几点是必须说的

方法应该是来自Fyyre

根本不懂其原理!…只是做绕过PG而已.让我讲清楚这玩意到底怎么回事不太可能.我没那水平!

esla.Angela出了一份做掉PG的汇编教程,基本按照教程就没问题了!只不过需要自己实现,代码是汇编代码.不过问题是教程并不完整,少了最

.导致进入系统时蓝屏

yre的英文原文.再加上Tesla.Angela的中文教程,我才得以成功绕过PG.不过还有一份东西很重要.最后再说

文原文,我无法找到了.只找到了转载:http://www.m5home.com/bbs/forum.php?mod=viewthread&tid=6182&page=1

文教程我就不放了.免得造成什么不愉快,我是在看雪找到的.

不准备写教程的.因为我觉得这种技术以及出现很久了.而且过PG的方法太多,而我也是从别人那里学来的.这样的东西不值得写成博文!我认

种东西会相当小儿科

基友的诱拐下,还是决定写出来.他说不会过PG,和被PG拦在外面的人很多很多

我先黑一波

.0是渣渣 MFC是垃圾!**Java**就不应该被发明!!!

们开始吧…

本教程的开头我就说过，WIN64 有两个内核保护机制，KPP 和 DSE。KPP 阻

我们 PATCH 内核， DSE 拦截我们加载驱动。 当然 KPP 和 DSE 并不是不可战胜的，

IN7X64 出来不久，FYYRE 就发布了破解内核的工具，后来有个叫做 Feryno 的人

公开了源代码，接下来我结合 FYYRE 的文档 Feryno 的源码进行讲解。

实现突破 DSE 和 PatchGuard，必须修改两个文件，winload.exe 以及

oskrnl.exe。首先把 WINLOAD.EXE 扔到 IDA 里，看看要修改哪些地方

助手winload!

inload.exe is the Windows loader for Vista & Windows 7. Along with this, he makes some verification of digital signatures and

ecking to make sure the files have not been modified. If modification of ntoskrnl is detected, the result is winload refusing

boot Windows and launching a WinPE looking “Recovery Mode”.

ART I { additional }: new way for patch of winload.exe

unction ImgpValidateImageHash - signature we locate: 8B C3 49 8B 5B 20 49 8B 73 28 49 8B 7B 30 4D 8B – you may play with

is one to make him smaller. as for this

atching, use of dUP2… size of not a concern. First bytes replaced with xor eax, eax (STATUS_SUCCESS) .. all validations

ccessful.

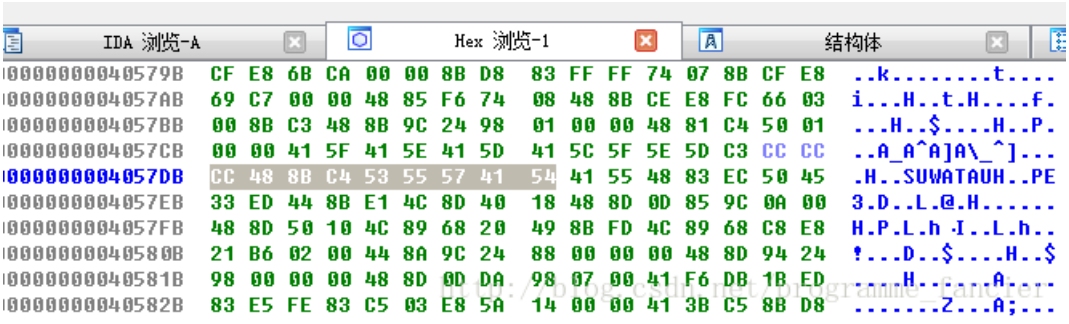
自行翻译吧…我就不献丑了= =

先把winload丢进IDA Pro

最开始,我是按照Fyyre给出来的特征码.不过后来我自己又修改了定位.所以这里就不用他的定位了

显示winload的定位

\xCC\x48\x8B\xC4\x53\x55\x57\x41\x54




```
.text:00000000004057DC arg_18          = qword ptr  20h
.text:00000000004057DC
.text:00000000004057DC                 mov     rax, rsp
.text:00000000004057DF                 push    rbx
.text:00000000004057E0                 push    rbp
.text:00000000004057E1                 push    rdi
.text:00000000004057E2                 push    r12
.text:00000000004057E4                 push    r13
.text:00000000004057E6                 sub     rsp, 50h
.text:00000000004057EA                 xor     r13d, r13d
.text:00000000004057ED                 mov     r12d, ecx
.text:00000000004057F0                 lea     r8, [rax+18h]
.text:00000000004057F4                 lea     rcx, dword_4AF480
.text:00000000004057FB                 lea     rdx, [rax+10h]
.text:00000000004057FF                 mov     [rax+20h], r13
.text:0000000000405803                 mov     rdi, r13
.text:0000000000405806                 mov     [rax-38h], r13
```


Starting from OslpMain, after loading the System registry hives(registry)… occurs a call to OslInitializeCodeIntegrity:

.text:00000000004016C3 call OslpLoadSystemHive

.text:00000000004016C3

.text:00000000004016C8 cmp eax, ebx

.text:00000000004016CA mov edi, eax

.text:00000000004016CC jl loc_401A08

.text:00000000004016CC

.text:00000000004016D2 mov ecx, ebp

.text:00000000004016D4 call OslInitializeCodeIntegrity <<– =(

48 8B C4 53

.text:00000000004057E8 mov rax, rsp

.text:00000000004057EB push rbx

.text:00000000004057EC push rbp

with: 0B0h, 01h, 0C3h, 090h … which produce:

mov al, 1

ret

nop

Fyyre的原文中说,需要修改此处四个字节.但是其实只需要修改3个字节即可!

要修改的是mov rax,rsp

也就是

48h,8Bh,C4h

mov rax, rsp

修改为

B0h,01h,C3h

mov al,1

ret

> 这么做的用途是跳过对 BlImgQueryCodeIntegrityBootOptions 的调用，据
> 我了解， 此函数 会判断 N NE TOSKRNL.EXE 的数字签名 有效性 （ 签名非法 的话就拒绝
> 加载 NTOSKRNL.EXE） ） 。所以这是一个难缠的家伙，直接跳过。

OK,winload搞定

> Save as winload.exe as osloader.exe (or whatever..) & correct PE checksum (LordPE and/or CFF_Explorer will do).
> Copy osloader.exe to \Windows\System32

至于文件校对值,我们后面再说!

接下来是ntoskrnl

首先是第一处地址:\x81\xEC\x58\x0F\x00\x00\x33\xFF\x39\x3D

```
000000014056D31A  00 00 00 00 00 00 90 90  90 90 90 90 90 90 90 90  ................
000000014056D32A  90 90 90 90 90 90 90 90  90 90 90 90 90 90 90 90  ................
000000014056D33A  90 90 90 90 90 90 44 89  44 24 18 89 54 24 10 89  ......D.D$..T$..
000000014056D34A  4C 24 08 53 55 56 57 41  54 41 55 41 56 41 57 48  L$.SUVWATAUAVAWH
000000014056D35A  81 EC 58 0F 00 00 33 FF  39 3D 88 21 D1 FF 74 07  ..X...3.9=.!..t.
000000014056D36A  B0 01 E9 68 2D 00 00 48  8D 1D 34 A5 C3 FF 48 8D  ...h-..H..4...H.
000000014056D37A  94 24 58 01 00 00 48 8B  CB E8 68 6C BF FF 48 3B  .$X...H...hl..H;
000000014056D38A  C7 0F 84 46 2D 00 00 48  8B 8C 24 58 01 00 00 E8  ...F-..H..$X....
000000014056D39A  BE 46 AF FF 48 3B C7 0F  84 30 2D 00 00 48 8B 94  .F..H;...0-..H..
000000014056D3AA  24 58 01 00 00 44 8B C3  48 8B C8 44 2B C2 E8 83  $X...D..H..D+...
```

```
NIT:000000014056D340                mov     [rsp+arg_10], r8d
NIT:000000014056D345                mov     [rsp+arg_8], edx
NIT:000000014056D349                mov     [rsp+arg_0], ecx
NIT:000000014056D34D                push    rbx
NIT:000000014056D34E                push    rbp
NIT:000000014056D34F                push    rsi
NIT:000000014056D350                push    rdi
NIT:000000014056D351                push    r12
NIT:000000014056D353                push    r13
NIT:000000014056D355                push    r14
NIT:000000014056D357                push    r15
NIT:000000014056D359                sub     rsp, 0F58h
NIT:000000014056D360                xor     edi, edi
NIT:000000014056D362                cmp     cs:InitSafeBootMode, edi
NIT:000000014056D368                jz      short loc_14056D371
NIT:000000014056D36A                mov     al, 1
NIT:000000014056D36C                jmp     loc_1405700D9
NIT:000000014056D371 ; ---------------------------------------------------------------------------
```


> Part III: Skip Initialization of PatchGuard - - (driver not required)
> As for this .txt, and PatchGuard… we are concerned with one function KiInitializePatchGuard(*1) which is called by KiFilterFiberContext.
> KiInitializePatchGuard is a very large function located in the INIT section of ntoskrnl, you can easily locate him via two calls from KiFilterFiberContext, by examination xrefs to exported dword InitSafeBootMode, searching for db 20h dup(90h) + db 044h … or 48 81 EC 58 0F 00 00 to name a few…
> PatchGuard does not initialize if we boot into safe mode. So to disable we just patch one conditional jxx

如果我们进入安全模式,PG则不会初始化.所以我们用nop填充掉

74 07

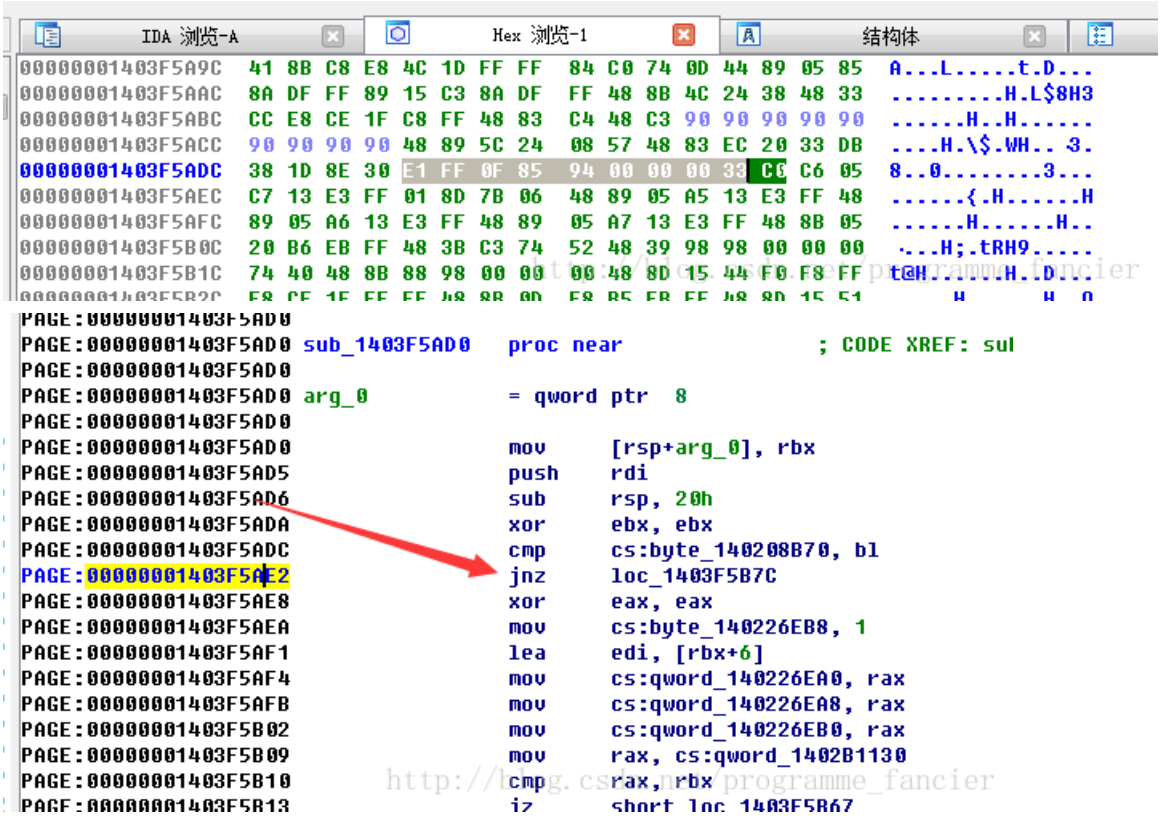
jz short loc_140561371

90 90

nop nop

这是ntoskrnl的第一处,接下来还有一处.是关于驱动签名强制的.在v1版本的英文原文中,我似乎没有找到这部分内容

其特征码为:\xE1\xFF\x0F\x85\x94\x00\x00\x00\x33\xC0



0Fh,85h,94h,00h,00h,00h

jnz loc_1403EAB0C

90 E9h,94h,00h,00h,00h

nop

jmp loc_1403EAB0C

> 这样做的目的是跳过“是否为 WINPE 模式”的判断，强行转入系统是 WINPE 模式的处理过程（loc_1403EAB0C）。因为如果是 WINPE 模式的话就不会校验驱动的数字签名。

OK,到此就已经完成了两份文件的修改

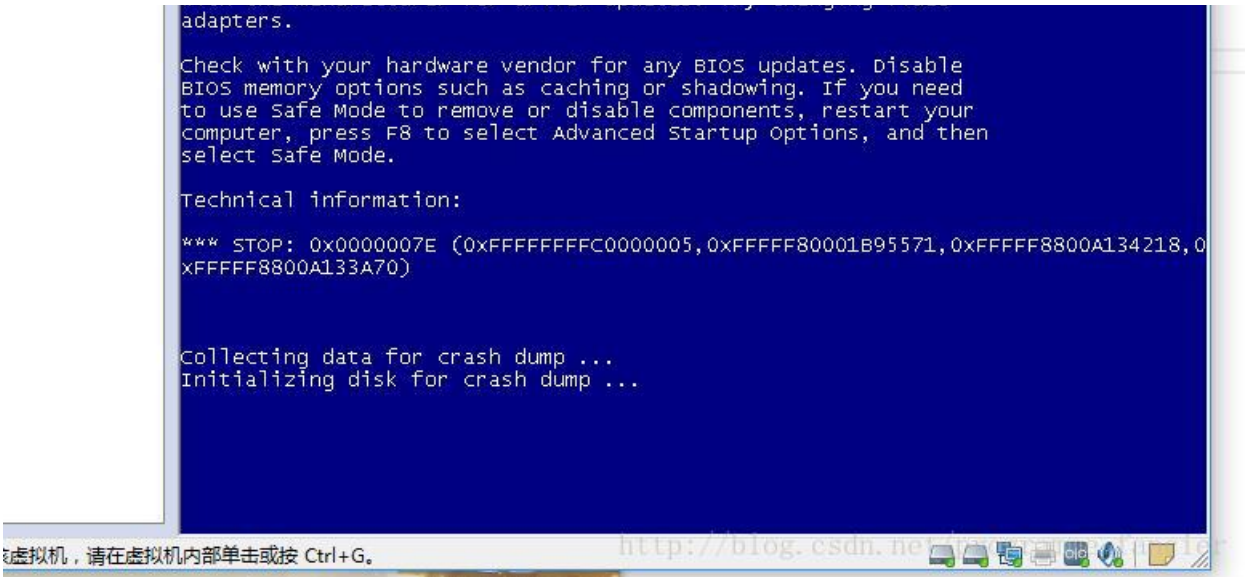
但是我们的任务还远远没有完成.当初我仅用了1,2天就搞定了这些,但实际上,真正搞定PG.我却花了半个月

至于文件校对我们最后说

先说说按照这样修改的结果

结果是在部分win64系统上完美的干掉了PG.驱动也不需要签名即可加载!

但仅仅是部分,随后我就遇到了麻烦



喜闻乐见

进入系统的时候蓝屏了

这是朋友返给我的截图 我从来不用VM.太他妈胖了…我用的是VirtualBox

当我第一次按照教程一步步来的时候,我成功的绕过了PG

那是在公司的时候,在公司的虚拟机上按照教程成功了.

但是当我回到家的时候,遇到了蓝屏…

无论我怎么校对,怎么**测试**.都没问题

也就是说,无论是V1版本的原文,还是中文的汇编过PG.都是不完整的!

我花了半个月研究这个问题.最终,我在网上找到一份过PG的补丁.其实

一直都在我硬盘里面…

那份补丁神奇的过掉了PG.于是我用了一个笨办法.二进制对比!

我捕捉了windows的日志,发现问题的来源.是ntoskrnl

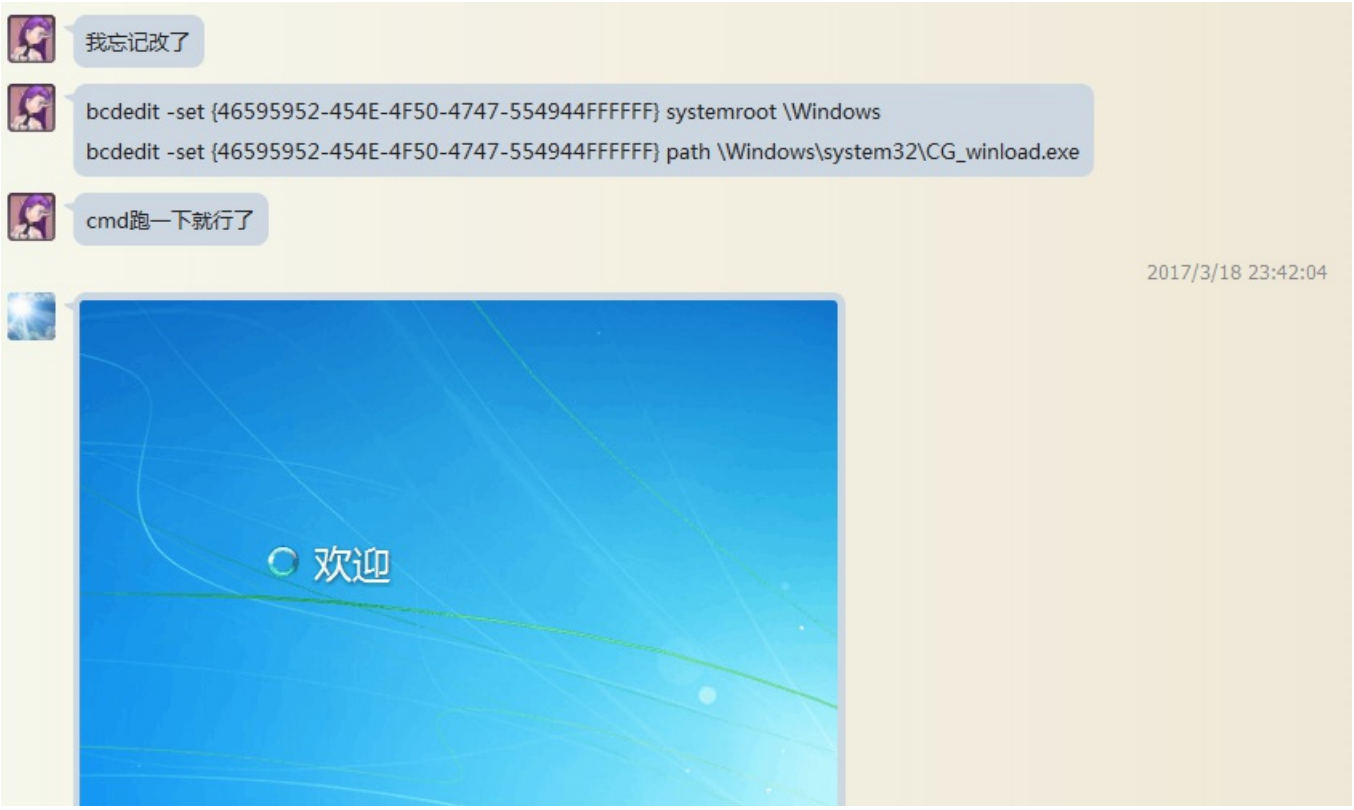
我继续深入下去,发现问题出在3个地方

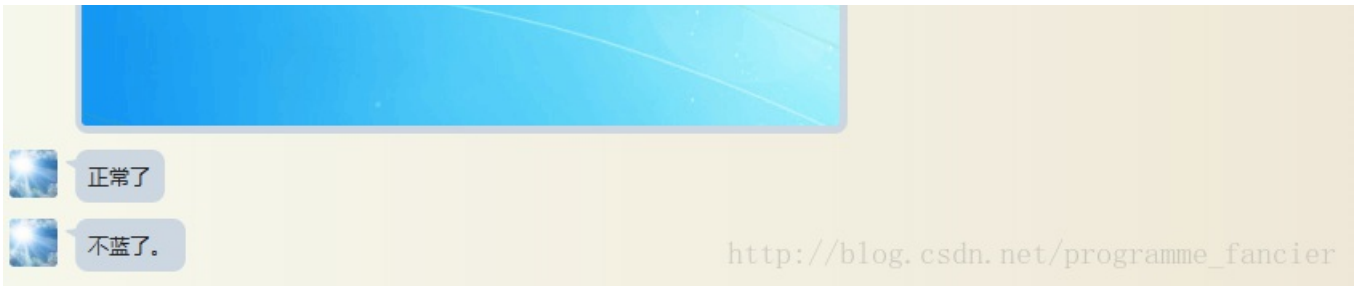
\xC1\xF8\x30

ntoskrnl中,会出现三次这段字节码

将30改成2F之后,完美过PG

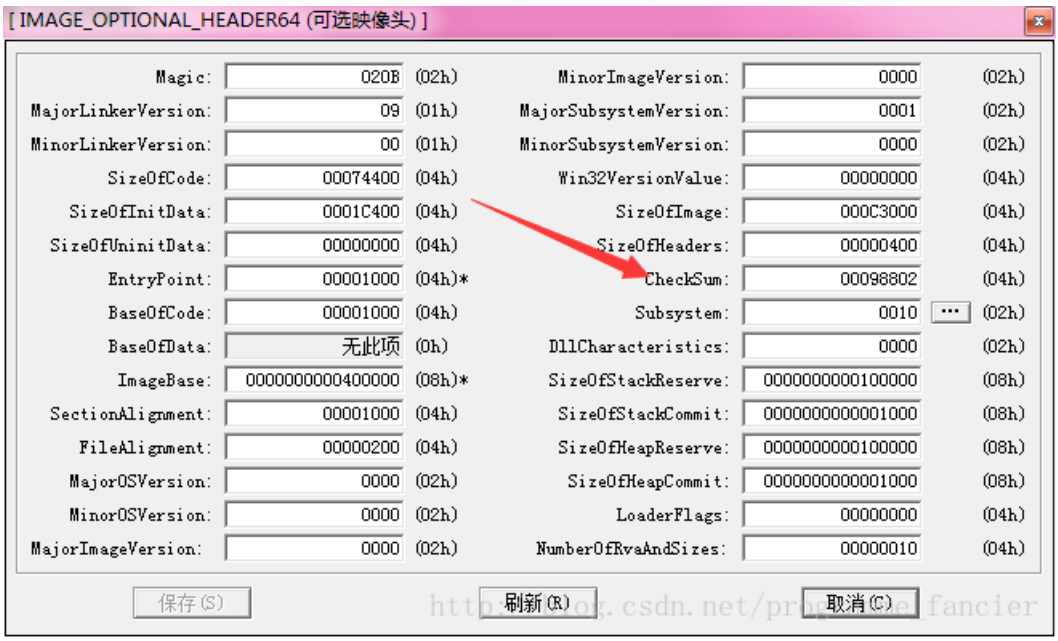
朋友返的图





OK,也就说.ntoskrnl要修改的地方并不止2处.而是3处!

接下来,我们的任务还没结束.因为windows会校对文件的checksum



上段代码,自行研究.

```
        DWORD Old, New;
        MapFileAndCheckSumA("CG_ntoskrnl.exe", &Old, &New);
        cout << "OLD" << Old << "NEW" << New << endl;
```

接下来才是最后一步,修改BCD启动项.我的选择是新建一份

这份是来自中文教程的

```
@ECHO OFF
ECHO.
ECHO Creating patched copies of winload, ntkrnlmp/ntoskrnl…
ECHO.
patch.exe
ECHO.
ECHO Creating BCD Entry…
ECHO.
set ENTRY_GUID={46595952-454E-4F50-4747-554944FFFFFF}
bcdedit -create %ENTRY_GUID% -d “KPP & DSE Disabled” -application OSLOADER
bcdedit -set %ENTRY_GUID% device partition=%SYSTEMDRIVE%
bcdedit -set %ENTRY_GUID% osdevice partition=%SYSTEMDRIVE%
bcdedit -set %ENTRY_GUID% systemroot \Windows
bcdedit -set %ENTRY_GUID% path \Windows\system32\freeload.exe
bcdedit -set %ENTRY_GUID% kernel goodkrnl.exe
bcdedit -set %ENTRY_GUID% recoveryenabled 0
bcdedit -set %ENTRY_GUID% nx OptOut
bcdedit -set %ENTRY_GUID% nointegritychecks 1
bcdedit -set %ENTRY_GUID% testsigning 1
bcdedit -displayorder %ENTRY_GUID% -addlast
bcdedit -timeout 5
bcdedit -default %ENTRY_GUID%
ECHO.
ECHO Setting PEAUTH service to manual… (avoid BSOD at login screen)
ECHO.
sc config peauth start= demand
ECHO.
ECHO Complete!
ECHO.
PAUSE
shutdown /r /t 0
```

OK,接下来就是C++实现的部分了.我会在代码上进行注释.很早以前写的,一直也没有整理.所以看起来很乱,有的也是瞎J8写的

```
// TEST_CON.cpp : 定义控制台应用程序的入口点。
//

#include "stdafx.h"//这好多都是不需要的.这本来是我用来写测试的CPP
#include "windows.h"
#include "winhttp.h"
#include <iostream>
#include <string>
#include "CGLIB.h"
#include <fstream>
#include <imagehlp.h>
#include <direct.h>
using namespace std;
#pragma comment(lib,"imagehlp")

void Function()
{
    fstream f("CG_winload.exe", ios::in | ios::binary | ios::out);//winload.打开文件,开始搜索特征码
    char p[] = "\xCC\x48\x8B\xC4\x53\x55\x57\x41\x54";
```

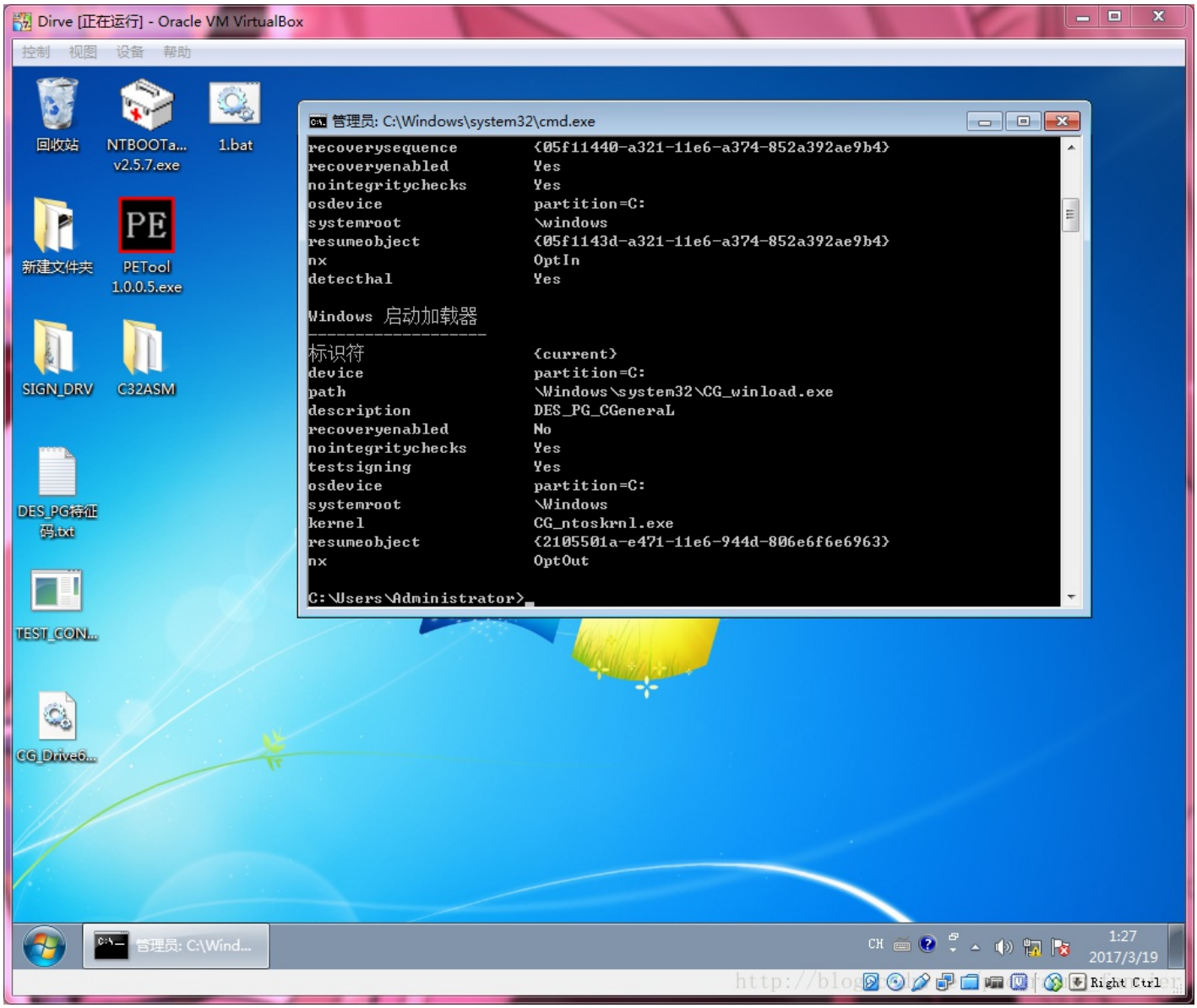
```
    string str((std::istreambuf_iterator<char>(f)), std::istreambuf_iterator<char>());
    int temp = str.find(p);//我选择直接读string里面去,然后用find来搜索,实现办法多的是.搜索结果的索引位置丢temp
    PIMAGE_DOS_HEADER  DosHeader;
    DosHeader = (PIMAGE_DOS_HEADER)str.c_str();//读PE
    int CheckSum = DosHeader->e_lfanew;
    CheckSum += 0x58;//读CheckSum.算偏移,为了后面做准备
    f.seekp(temp +1);//特征码偏移
    f.write("\xB0\x01\xC3", 3);//修改
    f.close();//关闭文件

    f.open("CG_winload.exe", ios::in | ios::binary | ios::out);//这里必须重新打开,不然无法获得正确的校对
    DWORD Old, New;
    MapFileAndCheckSumA("CG_winload.exe", &Old, &New);
    cout << "OLD" << Old << "NEW" << New << endl;
    f.seekp(CheckSum);
    char a[4] = { 0 };//这B玩意算出来的是十进制的
    memcpy(a, &New, 4);//所以我TM选择死亡.好吧,我用了memcpy
    f.write(a, 4);
    f.close();
}


int _tmain(int argc, _TCHAR* argv[])
{
    string File_Name1 = _getcwd(NULL, 0);
    string File_Name2 = File_Name1;
    File_Name1 += +"\\CG_winload.exe";
    File_Name2 += +"\\CG_ntoskrnl.exe";
    CopyFileA("C:\\Windows\\System32\\winload.exe", File_Name1.c_str(), 1);
    CopyFileA("C:\\Windows\\System32\\ntoskrnl.exe", File_Name2.c_str(), 1);//copy文件到运行目录并且改名

    Function();//先搞winload


    fstream f("CG_ntoskrnl.exe", ios::in | ios::binary | ios::out);
    char p[] = "\x81\xEC\x58\x0F\x00\x00\x33\xFF\x39\x3D";//第一个特征码
    char p2[] = "\xE1\xFF\x0F\x85\x94\x00\x00\x00\x33\xC0";//第二个特征码
    char p3[] = "\xC1\xF8\x30";//第三个特征码
    string str((std::istreambuf_iterator<char>(f)), std::istreambuf_iterator<char>());
    int index1 = str.find(p);
    int index2 = str.find(p2);
    int index3 = str.find(p3, 0);
    int index4 = str.find(p3, index3);
    int index5 = str.find(p3, index4);//笨办法,你们自己修改

    PIMAGE_DOS_HEADER  DosHeader;
    DosHeader = (PIMAGE_DOS_HEADER)str.c_str();
    int CheckSum = DosHeader->e_lfanew;
    CheckSum += 0x58;//老办法,读PE.算偏移
    f.seekp(index1 + sizeof(p)-1 + 4);
    f.write("\x90\x90", 2);
    f.seekp(index2 + 2);
    f.write("\x90\xE9\x94", 2);
    f.seekp(index3 + 2);
    f.write("\x2F", 1);
    f.seekp(index4 + 2);
    f.write("\x2F", 1);
    f.seekp(index5 + 2);
    f.write("\x2F", 1);
```

```
        f.close();//对一共5处进行修改 其中\xC1\xF8\x30有三处要修改为2F

        f.open("CG_ntoskrnl.exe", ios::in | ios::binary | ios::out);
        DWORD Old, New;
        MapFileAndCheckSumA("CG_ntoskrnl.exe", &Old, &New);
        cout << "OLD" << Old << "NEW" << New << endl;
        f.seekp(CheckSum);
        char a[4] = { 0 };
        memcpy(a, &New, 4);
        f.write(a, 4);
        f.close();
        cout << "OK" << endl;//重新改写校对

    //接下来是BCD部分
        string str_temp = "set CG={46595952-454E-4F50-4747-554944FFFFFF}&";
        str_temp += "bcdedit -create {46595952-454E-4F50-4747-554944FFFFFF} -d \"DES_PG_CGeneraL\" -applic
        str_temp += "bcdedit -set {46595952-454E-4F50-4747-554944FFFFFF} device partition=%SYSTEMDRIVE%&";
        str_temp += "bcdedit -set {46595952-454E-4F50-4747-554944FFFFFF} osdevice partition=%SYSTEMDRIVE%&
        str_temp += "bcdedit -set {46595952-454E-4F50-4747-554944FFFFFF} systemroot \\Windows&";
        str_temp += "bcdedit -set {46595952-454E-4F50-4747-554944FFFFFF} path \\Windows\\system32\\CG_winl
        str_temp += "bcdedit -set {46595952-454E-4F50-4747-554944FFFFFF} kernel CG_ntoskrnl.exe&";//此处改
        str_temp += "bcdedit -set {46595952-454E-4F50-4747-554944FFFFFF} recoveryenabled 0&";
        str_temp += "bcdedit -set {46595952-454E-4F50-4747-554944FFFFFF} nx OptOut&";
        str_temp += "bcdedit -set {46595952-454E-4F50-4747-554944FFFFFF} nointegritychecks 1&";
        str_temp += "bcdedit -set {46595952-454E-4F50-4747-554944FFFFFF} testsigning 1&";//测试驱动模式.至于
        str_temp += "bcdedit -displayorder {46595952-454E-4F50-4747-554944FFFFFF} -addlast&";//我选择添加到
        str_temp += "bcdedit -timeout 30&PAUSE&";//这句是废话
        system(str_temp.c_str());
        cout << "OK" << endl;

        CopyFileA(File_Name1.c_str(),"C:\\Windows\\System32\\CG_winload.exe" , 1);//复制文件
        CopyFileA(File_Name2.c_str(),"C:\\Windows\\System32\\CG_ntoskrnl.exe" , 1);//这两句要写到system前面

        getchar();
    }
```

OK.到此就搞定了

编译,运行.重启!

烦人的强行驱动签名和PG消失了!!!


Dirve [正在运行] - Oracle VM VirtualBox
控制 视图 设备 帮助
回收站
NTBOOTa... v2.5.7.exe
1.bat
新建文件夹
PETool 1.0.0.5.exe
SIGN_DRV
C32ASM
DES_PG特征码.txt
TEST_CON...
CG_Drive6...
管理员: C:\Windows\system32\cmd.exe
recoverysequence {05f11440-a321-11e6-a374-852a392ae9b4}
recoveryenabled Yes
nointegritychecks Yes
osdevice partition=C:
systemroot \windows
resumeobject {05f1143d-a321-11e6-a374-852a392ae9b4}
nx OptIn
detecthal Yes
Windows 启动加载器
标识符 {current}
device partition=C:
path \Windows\system32\CG_winload.exe
description DES_PG_CGeneraL
recoveryenabled No
nointegritychecks Yes
testsigning Yes
osdevice partition=C:
systemroot \Windows
kernel CG_ntoskrnl.exe
resumeobject {2105501a-e471-11e6-944d-806e6f6e6963}
nx OptOut
C:\Users\Administrator>
管理员: C:\Wind...
CH
1:27
2017/3/19
Right Ctrl