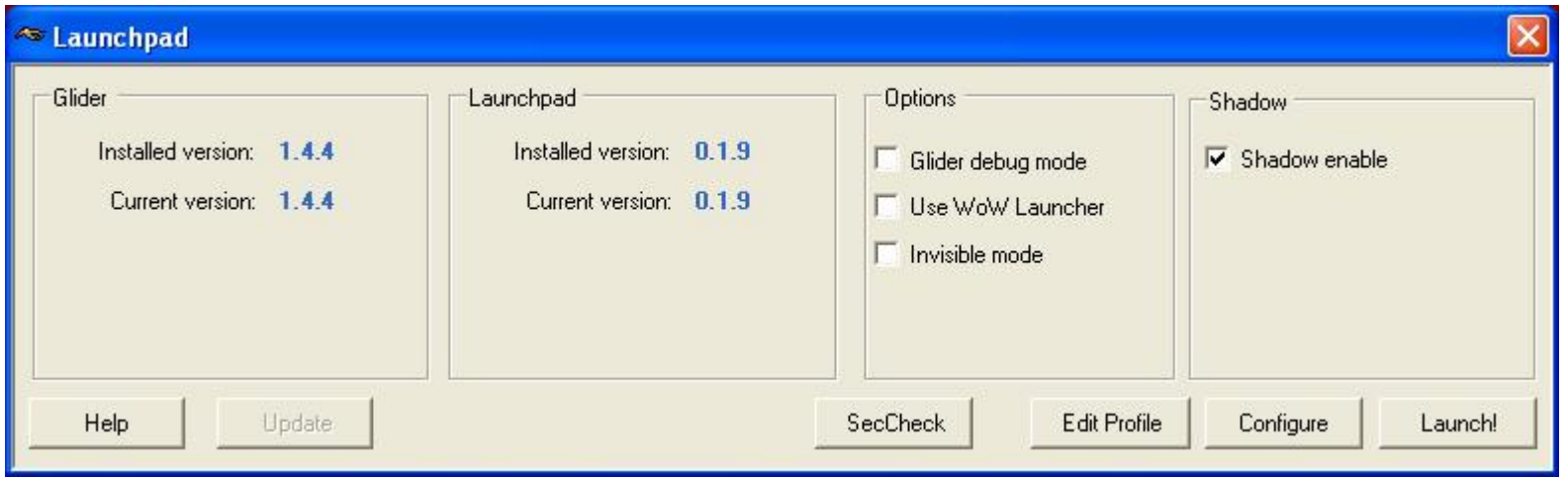
Overview

- Launchpad.exe
  - Renames Glider.exe to a random name (3-8 characters long)
  - Writes shadow kernel driver to disk with random name ".sys"
  - Loads kernel driver, starts it, then deletes the .sys file
  - Starts renamed Glider.exe /launchpad /handle=000001f4 /processid=00000418 /kill=shtokshtokshtok /resume
- shadow.sys (aka ppmevvyh.sys)
  - Hooks several system calls to hide itself and renamed glider.exe
  - Updates descriptor tables
  - Backs up current system states for replay to wow.exe
- Glider.exe (aka jxbvragk.exe)
  - Contains all of the game-playing logic
  - How does it query wow.exe process memory??

Launchpad.exe

- Shadow enable – use kernel driver
- Invisible mode – no glider gui



.NET Obfuscation – Makes it hard

```
.class public auto sealed ansi bc extends [mscorlib]System.Enum
{
  .field public specialname rtspecialname int32 value__
  .field public static literal value class bc a = int32(-1515870811)
  .field public static literal value class bc b = int32(-1515870811)
  .field public static literal value class bc c = int32(-1515870811)
  .field public static literal value class bc d = int32(-1515870811)
}

.class public auto ansi j extends [mscorlib]System.Object
{
  .field family static literal int32 a = int32(-1515870811)
  .field family static literal int32 b = int32(-1515870811)
  .field family static literal int32 c = int32(-1515870811)
  .field family static literal int32 d = int32(-1515870811)
  .field family static literal int32 e = int32(-1515870811)
  .field family static literal int32 f = int32(-1515870811)
  .field family static literal int32 g = int32(-1515870811)
  .field family static literal int32 h = int32(-1515870811)
  .field family static literal int32 i = int32(-1515870811)
  .field private static literal unsigned int32 j = unsigned int32(-1515870811)
  .field private static literal unsigned int32 k = unsigned int32(-1515870811)
  .field private static literal unsigned int32 l = unsigned int32(-1515870811)
  .field private static literal unsigned int32 m = unsigned int32(-1515870811)
  .field private static literal int32 n = int32(-1515870811)
  .field private static literal int32 o = int32(-1515870811)
  .field private static literal unsigned int32 p = unsigned int32(-1515870811)
  .field private static literal unsigned int32 q = unsigned int32(-1515870811)
```

Launchpad.exe – Reversed Functions

- 0x13FE0: Check for admin privileges
- 0x141D0: Ensure glider files are on an NTFS Volume
- 0x142F0: Warn user if running from desktop or root of drive
- 0x14952: Get WoW key from registry
  HKEY_LOCAL_MACHINE\SOFTWARE\Blizzard Entertainment\World of Warcraft\GamePath
- 0x14A20: Check if WoW is already running
- 0x14AA0: Report Warning (takes string)
- 0x14B50: Report Error (takes string) and exit launchpad
- 0x14B90: Initialize Launchpad.log with date and time or report error
- 0x14BF0: Backup Launchpad.log to Launchpad.lastRun.log
- 0x14CB0: Init Launchpad GUI (ctor for Forms, Groupbox, Label, etc...)

Launchpad.exe – More Reversed Functions

- 0x19DB3: Copy driver data to memory using GetManifestResourceStream
- 0x66F52: Create random string of 3-8 alpha characters using System.Random and StringBuilder::Append
- 0x19DF0: Create new shadow .sys file and copy binary data into it from memory
- 0x655C0: Phone Home (Perform DNS, open socket, write request, get responses from HTTP server)
- 0x65FD0: Install and start service (shadow.sys)
  - j::CreateService, j::StartService, j::OpenService, j::CloseServiceHandle
- 0x16530: Query all usernames and create restricted token

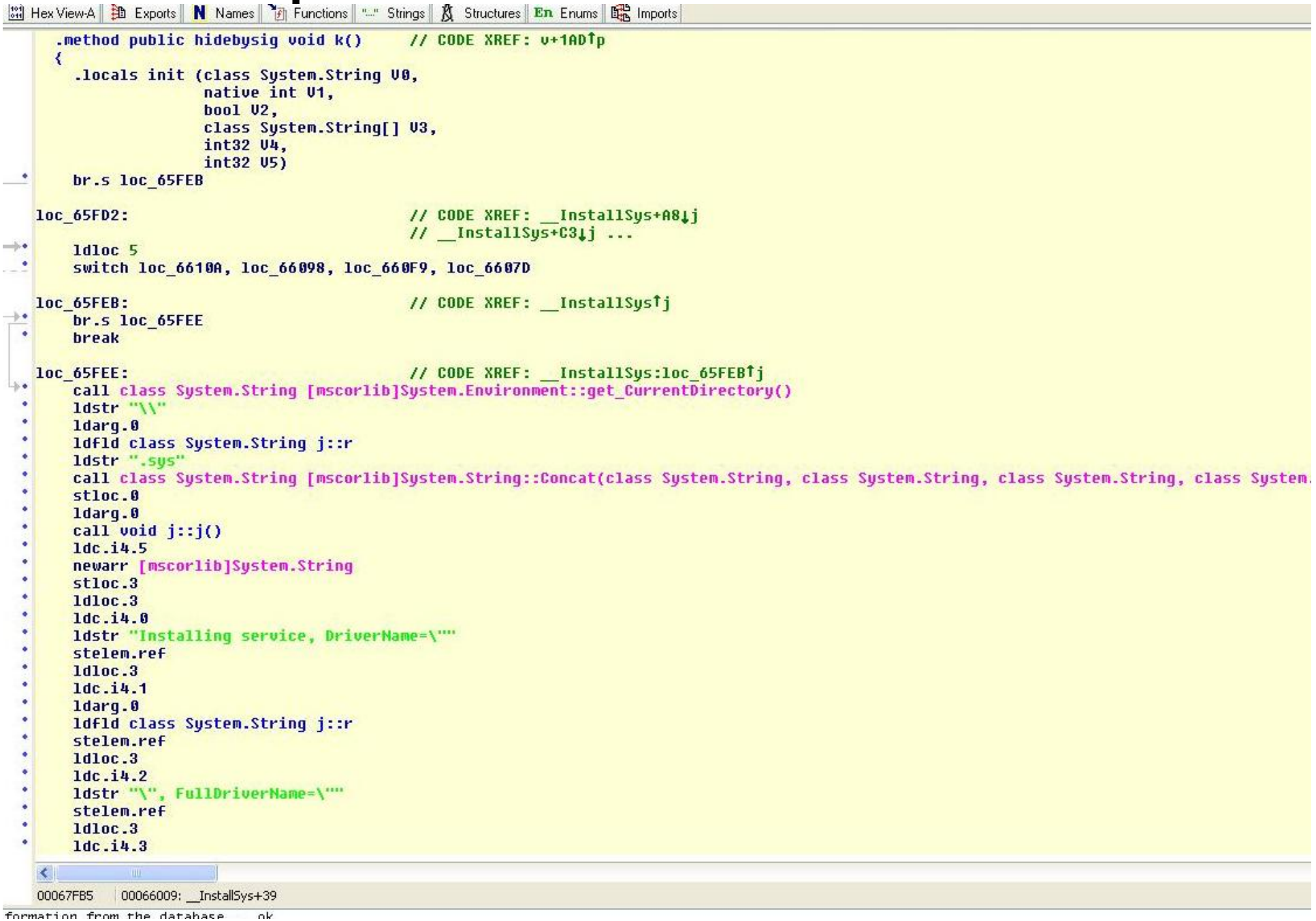
```
12:48:41 PM 0 = OZ\None
12:48:41 PM 1 = Everyone
12:48:41 PM 2 = BUILTIN\Administrators (not getting that one!)
12:48:41 PM 3 = BUILTIN\Users
12:48:41 PM 4 = NT AUTHORITY\INTERACTIVE
12:48:41 PM 5 = NT AUTHORITY\Authenticated Users
12:48:42 PM 6 = <unknown>
12:48:42 PM 7 = LOCAL
12:48:42 PM Created restricted token!
12:48:42 PM Fixing token owner/creator
12:48:42 PM Token owner: OZ\Dan
```

- 0x18752: Fix token owner/creator (ACL's, Sid, AccessToken, TokenInformationClass)
  - **This creates a token with many privileges disabled. Then, shadow.sys can replace the unwanted privilges with new privileges that glider needs to read memory, etc and assign this token info to the glider process. - could not validate this using process explorer from sysinternals. Showed same privileges for glider.exe as all other apps running.**

GetManifestResourceStream

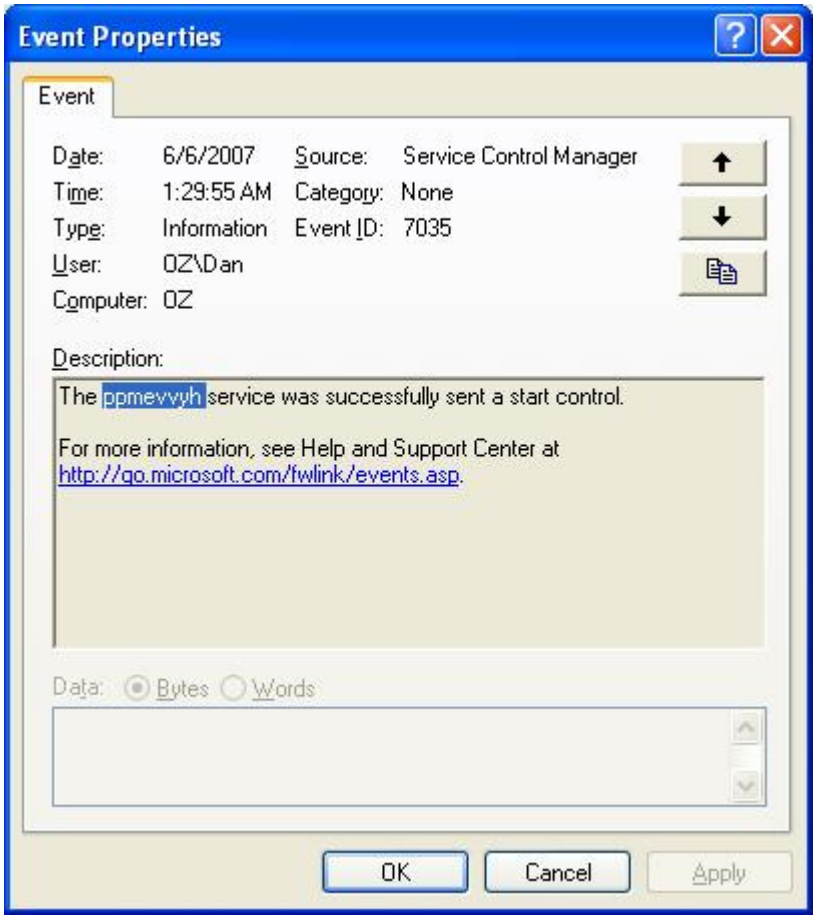
```
.method private static hidebysig unsigned int8[] d() // CODE XREF: __CreateShadow+58↓p
{
  .locals init (class [mscorlib]System.IO.Stream V0,
                unsigned int8[] V1)
  br.s loc_19DB3
  break

loc_19DB3:                                // CODE XREF: sub_19DB0↑j
  call class [mscorlib]System.Reflection.Assembly [mscorlib]System.Reflection.Assembly::GetExecutingAssembly()
  ldstr "Launchpad.Shadow.sys"
  callvirt class [mscorlib]System.IO.Stream [mscorlib]System.Reflection.Assembly::GetManifestResourceStream(class System.String)
  stloc.0
  ldloc.0
  callvirt int64 [mscorlib]System.IO.Stream::get_Length()
  conv.ovf.i
  newarr [mscorlib]System.Byte
  stloc.1
  ldloc.0
  ldloc.1
  ldc.i4.0
  ldloc.0
  callvirt int64 [mscorlib]System.IO.Stream::get_Length()
  conv.i4
  callvirt int32 [mscorlib]System.IO.Stream::Read(unsigned int8[], int32, int32)
  pop
  ldloc.0
  callvirt void [mscorlib]System.IO.Stream::Close()
  ldloc.1
  ret
}
```

Launchpad Installs Kernel "Service"

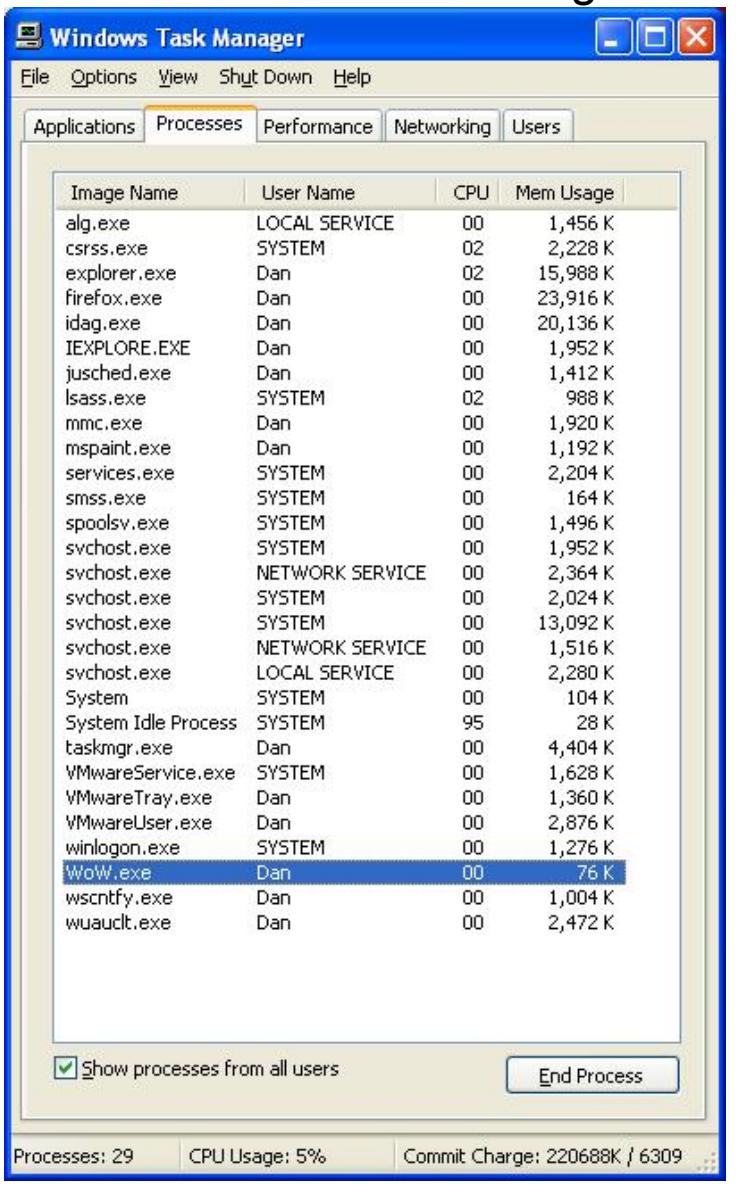


Shadow Kernel Driver Hides Glider

Windows Event Log



Windows Task Manager



EPROC Structure from "Rootkits"

"The assembly code for IoGetCurrentProcess goes to the offset 0x124 from the fs register. This is the pointer to the current ETHREAD. From the ETHREAD block, we follow the pointer in the KTHREAD structure to the EPROCESS block of the current process. We then traverse the doubly linked list of EPROCESS blocks until we locate the process we wish to hide.

One way to find a process is by its PID. The PID is located at an offset with the EPROCESS block that varies depending on the version of the operating system in which the rootkit is running. Here is where determining the operating system version will come into play."

EPROC Structure from "Rootkits"

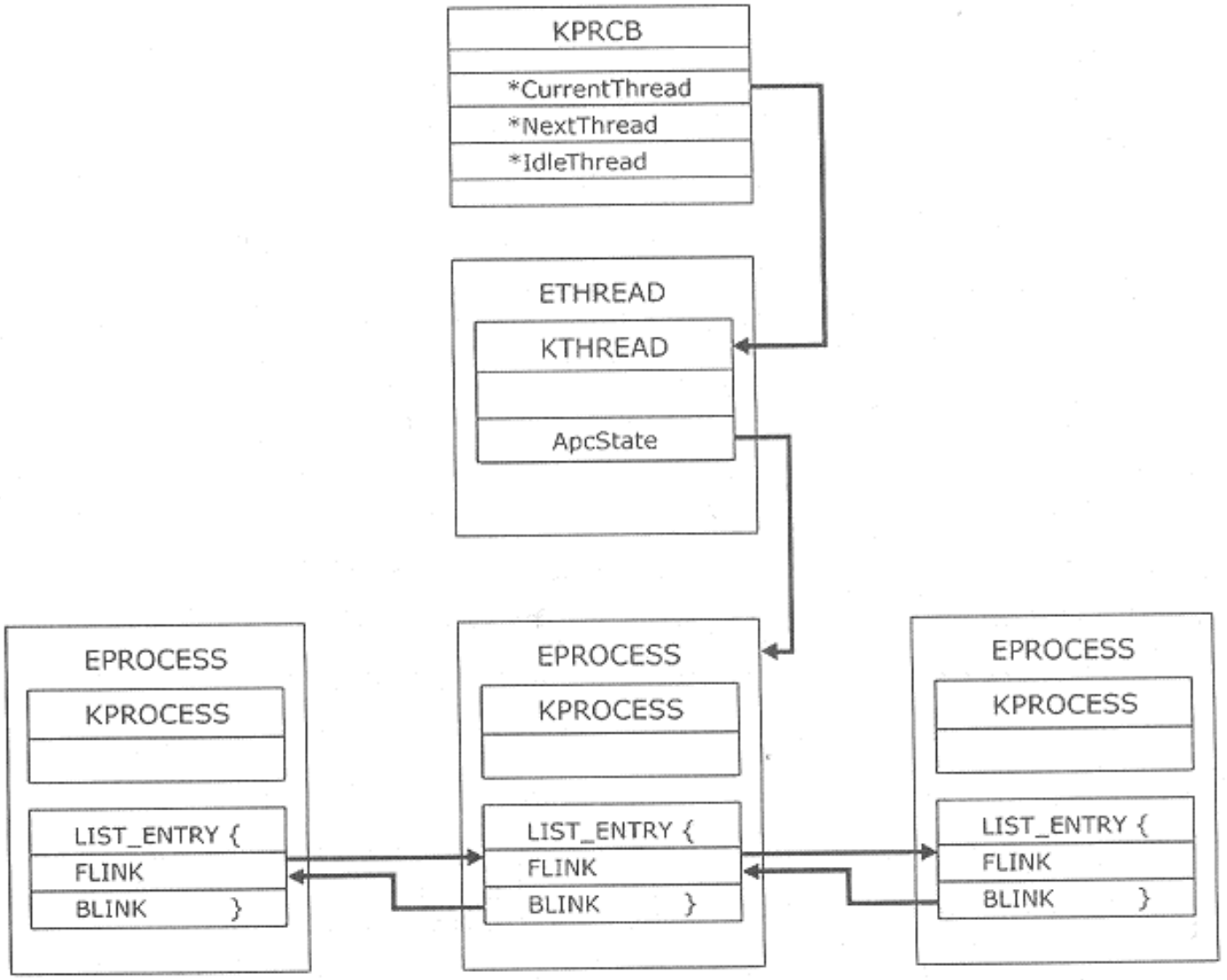


Figure 7–1 Path from KPRCB to the linked list of processes.

EPROC Structure from "Rootkits"

Table 7–1 Offsets to the PID and FLINK within the EPROCESS block.

	Windows NT	Windows 2000	Windows XP	Windows XP SP 2	Windows 2003
PID Offset	0x94	0x9C	0x84	0x84	, 0x84
FLINK Offset (to traverse the list of processes)	0x98	0xA0	0x88	0x88	0x88

Searching EPROC List

```
.text:000109BA                  push    esi
.text:000109BB                  push    edi
.text:000109BC                  mov     [ebp+var_8], 1
.text:000109C3                  call    ds:IoGetCurrentProcess
.text:000109C9                  mov     esi, eax
.text:000109CB                  mov     eax, dword_12B64
.text:000109D0                  mov     ebx, [eax+esi]
.text:000109D3                  push    ebx
.text:000109D4                  push    [ebp+arg_0]
.text:000109D7                  mov     [ebp+var_4], ebx
.text:000109DA                  push    offset aSearchingForD ; "Searching for: %d, StartPID = %d"
.text:000109DF                  call    nullsub_1
.text:000109E4                  push    esi
.text:000109E5                  push    ebx
.text:000109E6                  mov     edi, offset aLookingAtPidD ; "Looking at PID %d, EPROC = 0x%08x"
.text:000109EB                  push    edi
.text:000109EC                  call    nullsub_1
.text:000109F1                  mov     eax, ebx
.text:000109F3                  add     esp, 18h
.text:000109F6                  cmp     eax, [ebp+arg_0]
.text:000109F9                  jz      short loc_10A2C
.text:000109FB
.text:000109FB loc_109FB:                               ; CODE XREF: sub_109B2+78↓j
.text:000109FB                  cmp     [ebp+var_4], ebx
.text:000109FE                  jnz     short loc_10A06
.text:00010A00                  cmp     [ebp+var_8], 0
.text:00010A04                  jz      short loc_10A30
.text:00010A06
.text:00010A06 loc_10A06:                               ; CODE XREF: sub_109B2+4C↑j
.text:00010A06                  mov     eax, dword_12B68
.text:00010A0B                  mov     esi, [eax+esi]
.text:00010A0E                  and     [ebp+var_8], 0
.text:00010A12                  sub     esi, eax
.text:00010A14                  mov     eax, dword_12B64
.text:00010A19                  mov     ebx, [eax+esi]
.text:00010A1C                  push    esi
.text:00010A1D                  push    ebx
.text:00010A1E                  push    edi
.text:00010A1F                  call    nullsub_1
.text:00010A24                  add     esp, 0Ch
.text:00010A27                  cmp     ebx, [ebp+arg_0]
.text:00010A2A                  jnz     short loc_109FB
.text:00010A2C
.text:00010A2C loc_10A2C:                               ; CODE XREF: sub_109B2+47↑j
.text:00010A2C                  mov     eax, esi
.text:00010A2E                  jmp     short loc_10A3D
.text:00010A30 ; ---------------------------------------------------------------------------
.text:00010A30
.text:00010A30 loc_10A30:                               ; CODE XREF: sub_109B2+52↑j
.text:00010A30                  push    offset aLoopedAroundLi ; "Looped around list looking for process"...
.text:00010A35                  call    nullsub_1
.text:00010A3A                  pop     ecx
```

shadow.sys – Hooking System Calls

- **0x11538: Utility:** Hooks Descriptor Table (pass a string and ptr)
- 0x114BC: Hooks NtOpenProcess using 0x11538
- 0x1150E: Unhooks NtOpenProcess using 0x11538

- **0x110B2: Utility:** Hooks aW32kServiceTable (pass a string and ptr)
- 0x111A8: Hooks GetCursorPos using 0x110B2
- 0x111C1: Hooks GetForegroundWindow using 0x110B2
- 0x111DF: Hooks GetActiveWindow using 0x110B2
- 0x111FD: Hooks ThreeArg using 0x110B2
- 0x1121B: Hooks GetCursorInfo using 0x110B2
- 0x11239: Hooks OneArg using 0x110B2
- 0x11257: Hooks SetCursor using 0x110B2

0x11538 - HookDescriptorTable

```
.text:00011538 ; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||
.text:00011538
.text:00011538 ; Attributes: bp-based fr
.text:00011538
.text:00011538 HookDescriptorTable proc near           ; CODE XREF: sub_114BC+41↑p
.text:00011538                                         ; sub_1150E+17↑p ...
.text:00011538
.text:00011538 arg_0           = dword ptr  8
.text:00011538
.text:00011538                 mov     edi, edi
.text:0001153A                 push    ebp
.text:0001153B                 mov     ebp, esp
.text:0001153D                 mov     ecx, ds:KeServiceDescriptorTable
.text:00011543                 mov     ecx, [ecx]
.text:00011545                 push    esi
.text:00011546                 push    edi
.text:00011547                 mov     esi, eax
.text:00011549                 push    ebx
.text:0001154A                 shl     esi, 2
.text:0001154D                 mov     edi, [esi+ecx]
.text:00011550                 push    edi
.text:00011551                 push    eax
.text:00011552                 push    [ebp+arg_0]
.text:00011555                 push    offset aHookCallS04xFr ; "Hook call \"%s\" (%04X) from %08X -> %08X"...
.text:0001155A                 call    nullsub_1
.text:0001155F                 add     esp, 14h
.text:00011562                 test    ebx, ebx
.text:00011564                 jnz     short loc_11575
.text:00011566                 push    offset aWhoaNotPointin ; "Whoa, not pointing anything at NULL!  W"...
.text:0001156B                 call    nullsub_1
.text:00011570                 pop     ecx
.text:00011571                 xor     eax, eax
.text:00011573                 jmp     short end_HookDescriptorTable
.text:00011575 ; ---------------------------------------------------------------------------
.text:00011575
.text:00011575 loc_11575:                              ; CODE XREF: HookDescriptorTable+2C↑j
.text:00011575                 cli
.text:00011576                 mov     eax, cr0
.text:00011579                 and     eax, 0FFFEFFFFh
.text:0001157E                 mov     cr0, eax
.text:00011581                 mov     eax, ds:KeServiceDescriptorTable
.text:00011586                 mov     eax, [eax]
.text:00011588                 mov     [esi+eax], ebx
.text:0001158B                 mov     eax, cr0
.text:0001158E                 or      eax, 10000h
.text:00011593                 mov     cr0, eax
.text:00011596                 sti
.text:00011597                 mov     eax, edi
.text:00011599
.text:00011599 end_HookDescriptorTable:                ; CODE XREF: HookDescriptorTable+3B↑j
.text:00011599                 pop     edi
```

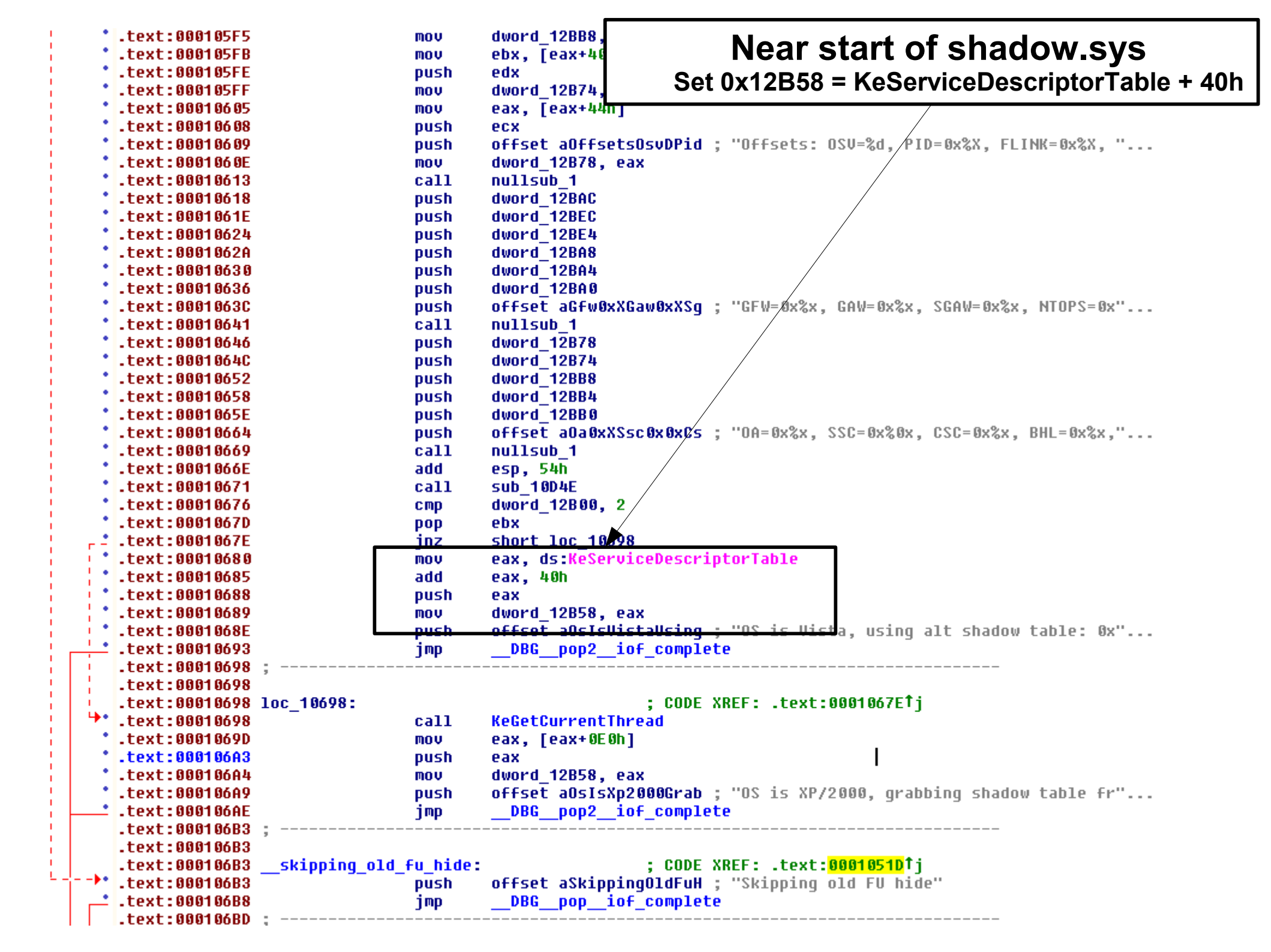
nullsub_1 "was" debug printf

Shadow.sys Uses CR0 "Trick"

Done exactly the way it's presented on p.67 of "Rootkits" book

End of 0x11538 - HookDescriptorTable

```
cli
mov     eax, cr0
and     eax, 0FFFEFFFFh
mov     cr0, eax
mov     eax, ds:KeServiceDescriptorTable
mov     eax, [eax]
mov     [esi+eax], ebx
mov     eax, cr0
or      eax, 10000h
mov     cr0, eax
sti
mov     eax, edi
```




```
.text:000105F5                  mov     dword_12BB8,
.text:000105FB                  mov     ebx, [eax+40
.text:000105FE                  push    edx
.text:000105FF                  mov     dword_12B74,
.text:00010605                  mov     eax, [eax+44h]
.text:00010608                  push    ecx
.text:00010609                  push    offset aOffsetsOsvDPid ; "Offsets: OSV=%d, PID=0x%X, FLINK=0x%X, "...
.text:0001060E                  mov     dword_12B78, eax
.text:00010613                  call    nullsub_1
.text:00010618                  push    dword_12BAC
.text:0001061E                  push    dword_12BEC
.text:00010624                  push    dword_12BE4
.text:0001062A                  push    dword_12BA8
.text:00010630                  push    dword_12BA4
.text:00010636                  push    dword_12BA0
.text:0001063C                  push    offset aGfw0xXGaw0xXSg ; "GFW=0x%x, GAW=0x%x, SGAW=0x%x, NTOPS=0x"...
.text:00010641                  call    nullsub_1
.text:00010646                  push    dword_12B78
.text:0001064C                  push    dword_12B74
.text:00010652                  push    dword_12BB8
.text:00010658                  push    dword_12BB4
.text:0001065E                  push    dword_12BB0
.text:00010664                  push    offset aOa0xXSsc0x0xCs ; "OA=0x%x, SSC=0x%0x, CSC=0x%x, BHL=0x%x,"...
.text:00010669                  call    nullsub_1
.text:0001066E                  add     esp, 54h
.text:00010671                  call    sub_10D4E
.text:00010676                  cmp     dword_12B00, 2
.text:0001067D                  pop     ebx
.text:0001067E                  jnz     short loc_10698
.text:00010680                  mov     eax, ds:KeServiceDescriptorTable
.text:00010685                  add     eax, 40h
.text:00010688                  push    eax
.text:00010689                  mov     dword_12B58, eax
.text:0001068E                  push    offset aOsIsVistaUsing ; "OS is Vista, using alt shadow table: 0x"...
.text:00010693                  jmp     __DBG__pop2__iof_complete
.text:00010698 ; ---------------------------------------------------------------------------
.text:00010698
.text:00010698 loc_10698:                               ; CODE XREF: .text:0001067E↑j
.text:00010698                  call    KeGetCurrentThread
.text:0001069D                  mov     eax, [eax+0E0h]
.text:000106A3                  push    eax
.text:000106A4                  mov     dword_12B58, eax
.text:000106A9                  push    offset aOsIsXp2000Grab ; "OS is XP/2000, grabbing shadow table fr"...
.text:000106AE                  jmp     __DBG__pop2__iof_complete
.text:000106B3 ; ---------------------------------------------------------------------------
.text:000106B3
.text:000106B3 __skipping_old_fu_hide:                  ; CODE XREF: .text:0001051D↑j
.text:000106B3                  push    offset aSkippingOldFuH ; "Skipping old FU hide"
.text:000106B8                  jmp     __DBG__pop__iof_complete
.text:000106BD ; ---------------------------------------------------------------------------
```

Hook Win32ServiceTable

```
.text:000110B2 ; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||
.text:000110B2
.text:000110B2 ; Attributes: bp-based frame
.text:000110B2
.text:000110B2 __HookTable     proc near               ; CODE XREF: __HookW32ServiceTable+80↓p
.text:000110B2                                         ; __HookW32ServiceTable+9E↓p ...
.text:000110B2
.text:000110B2 arg_0           = dword ptr  8
.text:000110B2 arg_4           = dword ptr  0Ch
.text:000110B2 arg_8           = dword ptr  10h
.text:000110B2
.text:000110B2                 mov     edi, edi
.text:000110B4                 push    ebp
.text:000110B5                 mov     ebp, esp
.text:000110B7                 mov     eax, [ebp+arg_4]
.text:000110BA                 mov     ecx, dword_12B58
.text:000110C0                 mov     ecx, [ecx+10h]
.text:000110C3                 push    ebx
.text:000110C4                 mov     ebx, [ebp+arg_8]
.text:000110C7                 push    esi
.text:000110C8                 push    edi
.text:000110C9                 mov     esi, eax
.text:000110CB                 push    ebx
.text:000110CC                 shl     esi, 2
.text:000110CF                 mov     edi, [esi+ecx]
.text:000110D2                 push    edi
.text:000110D3                 push    eax
.text:000110D4                 push    [ebp+arg_0]
.text:000110D7                 push    offset aHookCallS04xFr ; "Hook call \"%s\" (%04X) from %08X -> %08X"...
.text:000110DC                 call    nullsub_1
.text:000110E1                 add     esp, 14h
.text:000110E4                 test    ebx, ebx
.text:000110E6                 jnz     short loc_110F7
.text:000110E8                 push    offset aWhoaNotPointin ; "Whoa, not pointing anything at NULL!  W"...
.text:000110ED                 call    nullsub_1
.text:000110F2                 pop     ecx
.text:000110F3                 xor     eax, eax
.text:000110F5                 jmp     short loc_1111C
.text:000110F7 ; ---------------------------------------------------------------------------
.text:000110F7
.text:000110F7 loc_110F7:                              ; CODE XREF: __HookTable+34↑j
.text:000110F7                 cli
.text:000110F8                 mov     eax, cr0
.text:000110FB                 and     eax, 0FFFEFFFFh
.text:00011100                 mov     cr0, eax
.text:00011103                 mov     eax, dword_12B58
.text:00011108                 mov     eax, [eax+10h]
.text:0001110B                 mov     [esi+eax], ebx
.text:0001110E                 mov     eax, cr0
.text:00011111                 or      eax, 10000h
.text:00011116                 mov     cr0, eax
.text:00011119                 sti
.text:0001111A                 mov     eax, edi
.text:0001111C
```

Hook HwndList

```
.text:00010BC0 ; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||
.text:00010BC0
.text:00010BC0 ; Attributes: bp-based frame
.text:00010BC0
.text:00010BC0 __BuildHwndList proc near               ; CODE XREF: sub_10C2E+17↓p
.text:00010BC0                                         ; sub_10C58+30↓p
.text:00010BC0
.text:00010BC0 arg_0           = dword ptr  8
.text:00010BC0
.text:00010BC0                 mov     edi, edi
.text:00010BC2                 push    ebp
.text:00010BC3                 mov     ebp, esp
.text:00010BC5                 mov     ecx, dword_12B58
.text:00010BCB                 mov     ecx, [ecx+10h]
.text:00010BCE                 push    esi
.text:00010BCF                 push    edi
.text:00010BD0                 mov     esi, eax
.text:00010BD2                 push    ebx
.text:00010BD3                 shl     esi, 2
.text:00010BD6                 mov     edi, [esi+ecx]
.text:00010BD9                 push    edi
.text:00010BDA                 push    eax
.text:00010BDB                 push    [ebp+arg_0]
.text:00010BDE                 push    offset aHookCallS04xFr ; "Hook call \"%s\" (%04X) from %08X -> %08X"...
.text:00010BE3                 call    nullsub_1
.text:00010BE8                 add     esp, 14h
.text:00010BEB                 test    ebx, ebx
.text:00010BED                 jnz     short loc_10BFE
.text:00010BEF                 push    offset aWhoaNotPointin ; "Whoa, not pointing anything at NULL!  W"...
.text:00010BF4                 call    nullsub_1
.text:00010BF9                 pop     ecx
.text:00010BFA                 xor     eax, eax
.text:00010BFC                 jmp     short loc_10C23
.text:00010BFE ; ---------------------------------------------------------------------------
.text:00010BFE
.text:00010BFE loc_10BFE:                              ; CODE XREF: __BuildHwndList+2D↑j
.text:00010BFE                 cli
.text:00010BFF                 mov     eax, cr0
.text:00010C02                 and     eax, 0FFFEFFFFh
.text:00010C07                 mov     cr0, eax
.text:00010C0A                 mov     eax, dword_12B58
.text:00010C0F                 mov     eax, [eax+10h]
.text:00010C12                 mov     [esi+eax], ebx
.text:00010C15                 mov     eax, cr0
.text:00010C18                 or      eax, 10000h
.text:00010C1D                 mov     cr0, eax
.text:00010C20                 sti
.text:00010C21                 mov     eax, edi
.text:00010C23
.text:00010C23 loc_10C23:                              ; CODE XREF:   BuildHwndList+3C↑j
```

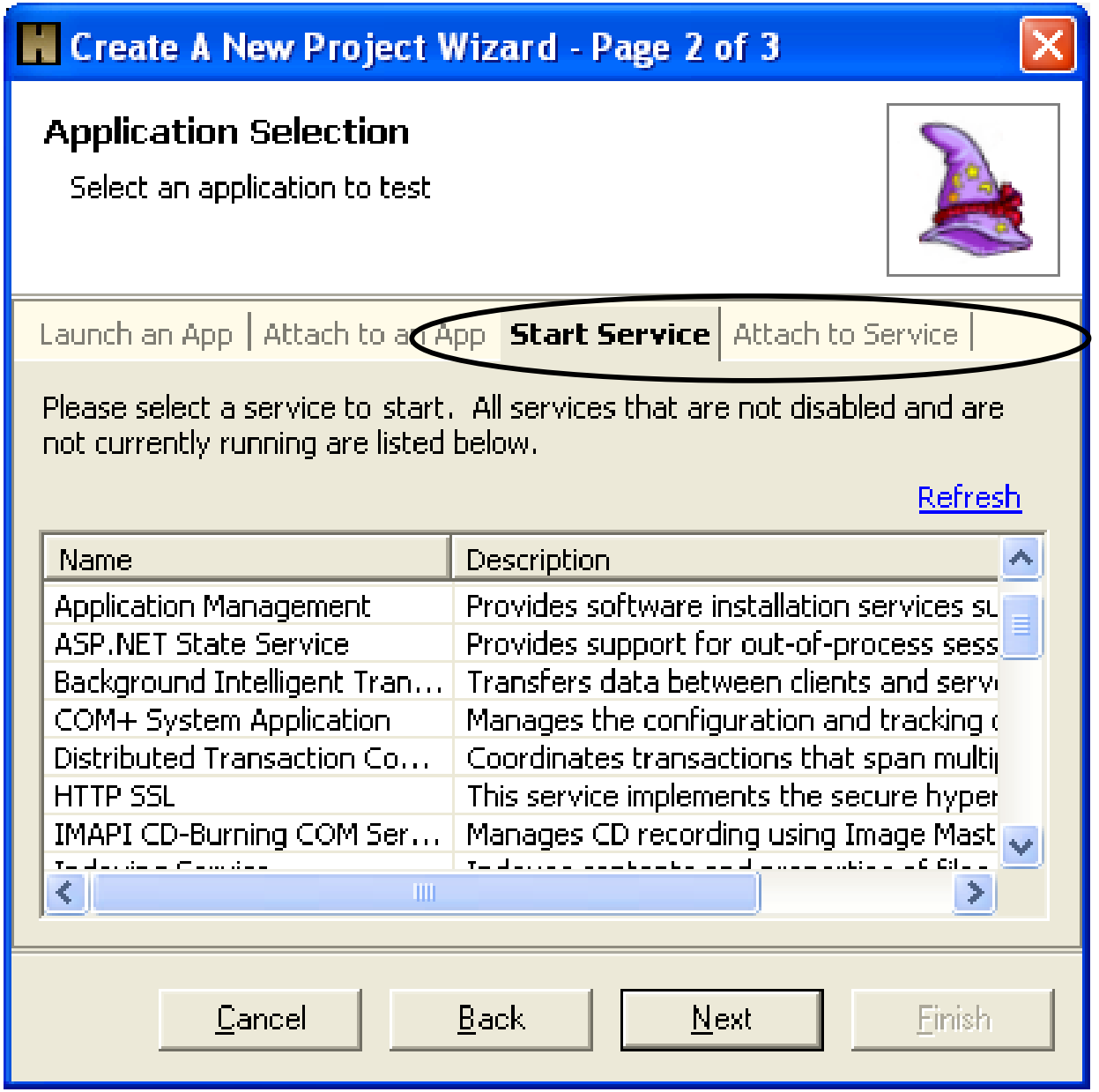
Example Hook: GetCursorInfo

Each hook just checks if it's WoW making the call

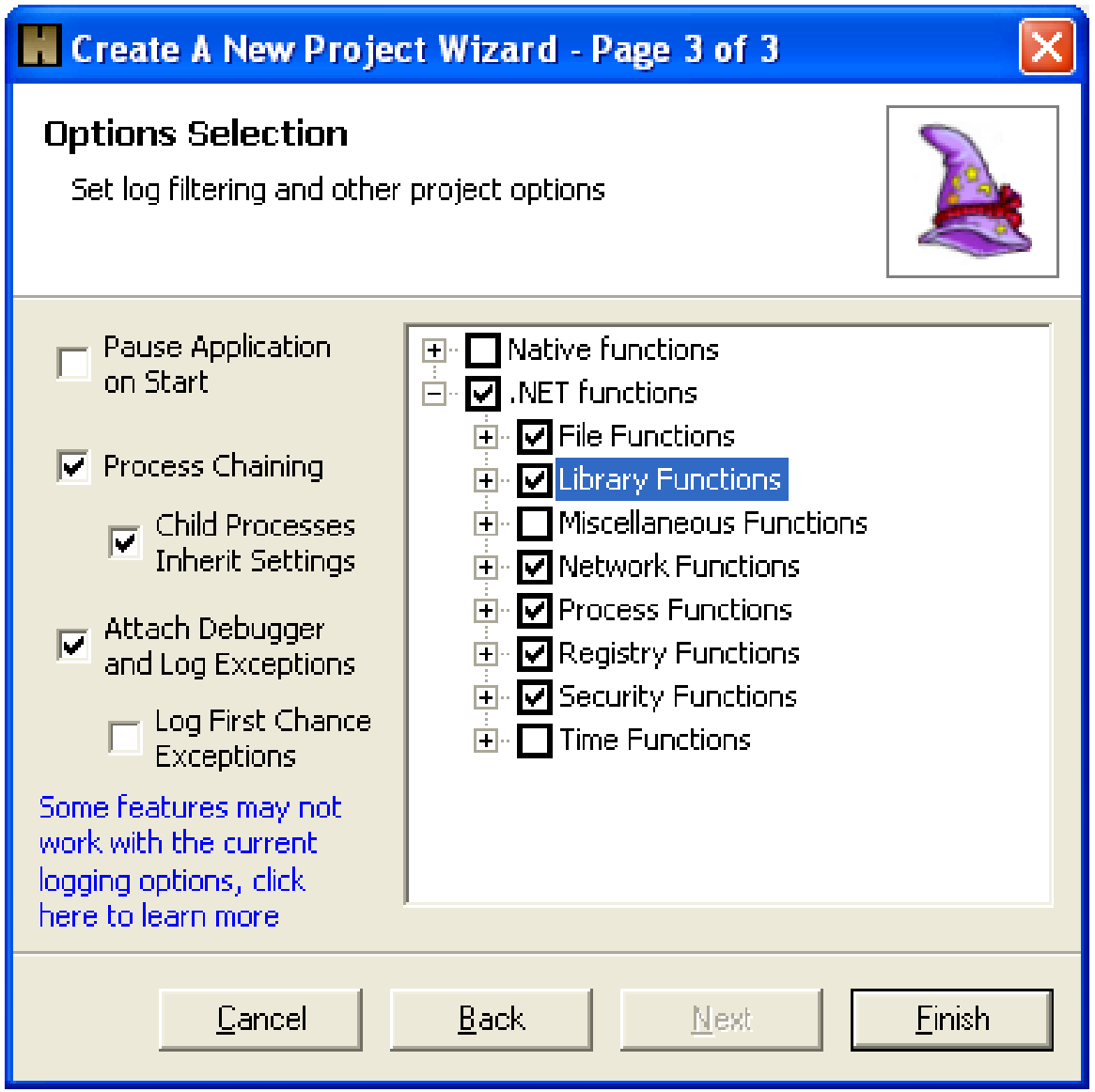
```
.text:00010F38
.text:00010F38 loc_10F38:                              ; DATA XREF: __HookW32ServiceTable+ED↓o
.text:00010F38                 mov     edi, edi
.text:00010F3A                 push    ebp
.text:00010F3B                 mov     ebp, esp
.text:00010F3D                 push    ebx
.text:00010F3E                 push    esi
.text:00010F3F                 mov     esi, [ebp+8]
.text:00010F42                 push    esi
.text:00010F43                 call    dword_12BCC
.text:00010F49                 mov     ebx, eax
.text:00010F4B                 test    ebx, ebx
.text:00010F4D                 jz      short loc_10FA4
.text:00010F4F                 push    edi
.text:00010F50                 call    ds:PsGetCurrentProcessId
.text:00010F56                 mov     edi, eax
.text:00010F58                 push    edi
.text:00010F59                 push    offset aGetcursorinfoC ; "GetCursorInfo called from process id %d"...
.text:00010F5E                 call    nullsub_1
.text:00010F63                 cmp     dword_12B30, 0
.text:00010F6A                 pop     ecx
.text:00010F6B                 pop     ecx
.text:00010F6C                 jz      short loc_10FA3
.text:00010F6E                 cmp     edi, WoW_pid
.text:00010F74                 jnz     short loc_10FA3
.text:00010F76                 mov     eax, dword_12B88
.text:00010F7B                 test    eax, eax
.text:00010F7D                 jle     short loc_10FA3
.text:00010F7F                 push    dword_12B8C
.text:00010F85                 push    eax
.text:00010F86                 push    offset aSendingBackBog ; "Sending back bogus cursor position: %d/"...
.text:00010F8B                 call    nullsub_1
.text:00010F90                 mov     eax, dword_12B88
.text:00010F95                 mov     [esi+0Ch], eax
.text:00010F98                 mov     eax, dword_12B8C
.text:00010F9D                 add     esp, 0Ch
.text:00010FA0                 mov     [esi+10h], eax
.text:00010FA3
.text:00010FA3 loc_10FA3:                              ; CODE XREF: .text:00010F6C↑j
```

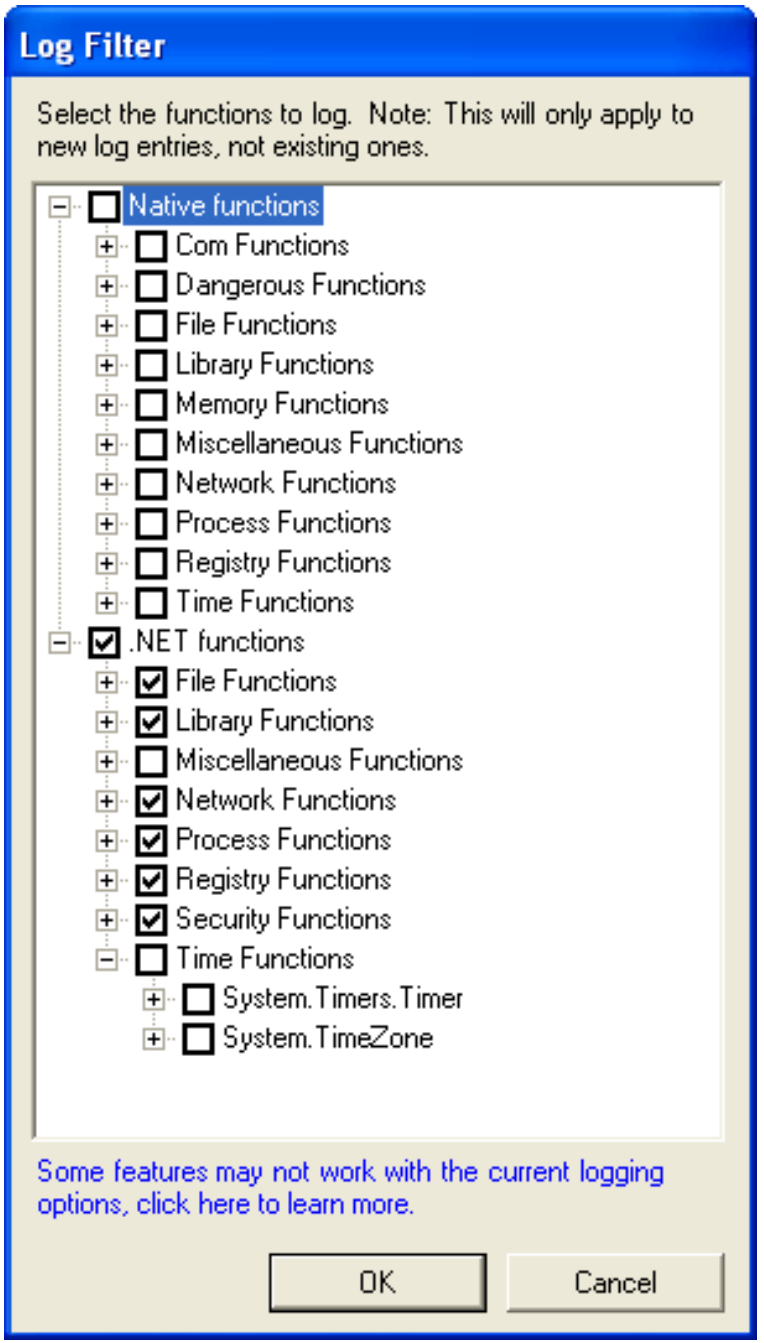
Is WoW asking??

Holodeck Cannot Find shadow.sys

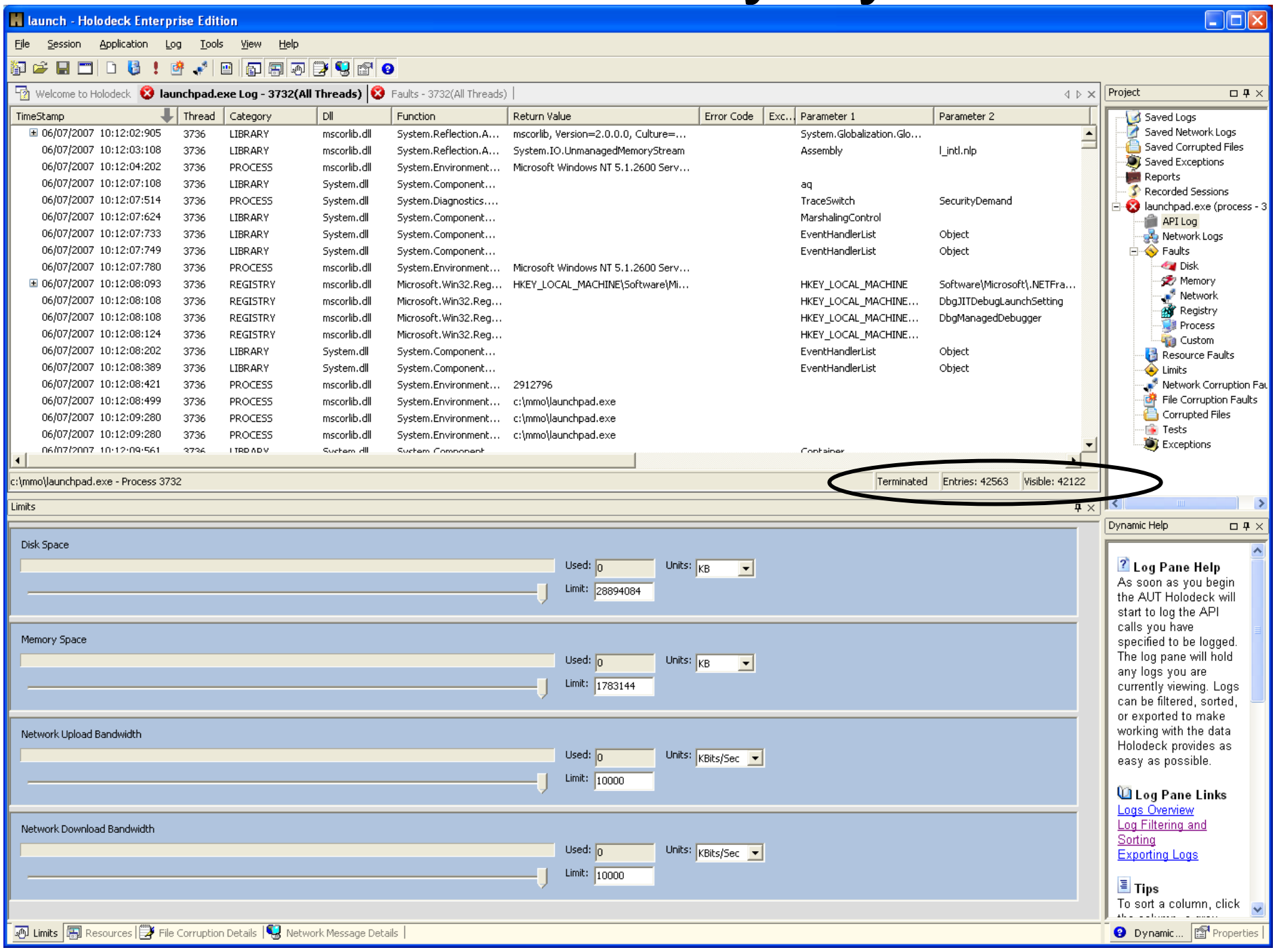


Holodeck Can Run Launchpad.exe

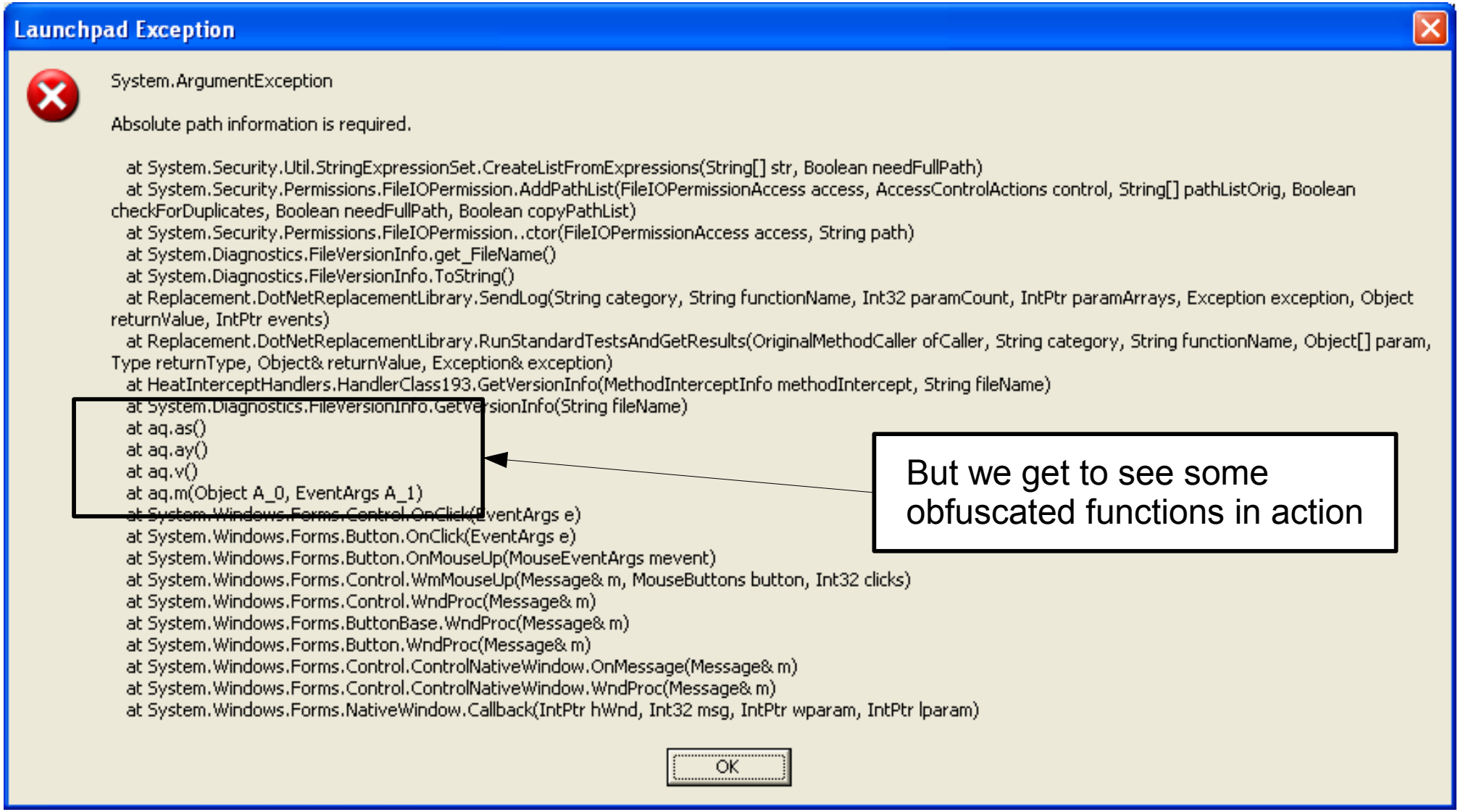




Holodeck – Too many System Calls



Holodeck – Exceptions and BSOD's



Further Strategies

- Add **printk** call to shadow.sys: nullsub_1 to see debug messages
- Edit launchpad.exe to use fixed shadow.sys on ever call rather than writing over it each time

Observations & Questions

- Mention of FU in shadow.sys could be FU_rootkit
- Shadow/Glider does not seem to hide disk files
- Blizzard wants to make sure glider is "running"?