아무도 모르는 Process

written by Kwak Taejin(bluewarz@devguru.co.kr)

많은 사람들이 스트레스 해소 방법 중의 하나로 PC 게임을 즐길 것이다. 필자 또한 PC 게임을 하면서 지친 머리를 잠시 쉬게 해준다. 때로는 더 스트레스를 받을 때도 있을 때도 있지만..

요새는 온라인 길거리 농구를 즐기고 있다. 농구를 좋아하는 필자로 써는 참으로 재미있고 잘 만들어진 게임이 아닐 수 없다.

어느 온라인 게임이든 Hack이라 불리 우는 프로그램 등을 이용하여 나쁜(?) 짓을 하는 유저들이 있다. 이 농구 게임 역시 현재 서비스 업체에서도 이 문제로 골머리를 썩더니 어느 순간부터는 hack 프로그램이 차단하는 코드를 삽입 하였다. (필자가 hack 프로그램을 사용하여 게임을 하진 않는다. ^^) 이후 hack 프로그램은 어느 정도 근절 된 것 같다.

- 차단은 System 상에 실행 되고 있는 Process를 열거해서 hack 프로그램으로 의심이 가는 프로그램이 존재 하면 게임을 실행 못하도록 하는 구조인 것 같다. -

마치 어린 아이처럼 하지 말라는 짓을 더 하고 싶은 것처럼 필자는 hack 프로그램을 차단하는 것을 무력화 해봐야 겠다는 생각이 들었다. (나쁜 의도에서가 아니라 게임도중 어떻게 하다 얘기가 나온 바람에 증명을 위해서 였다)

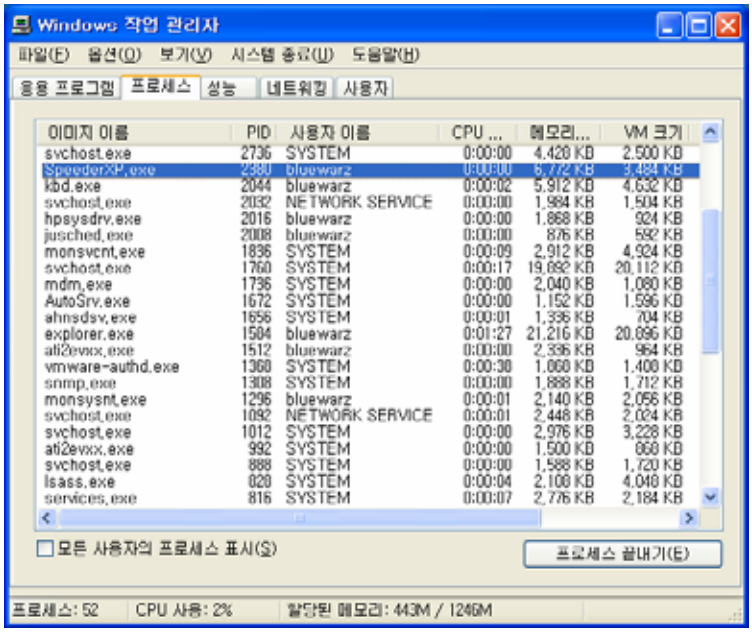
이 구현 방법이 다양한 곳에서 쓰일 수 있으니 필자와 같은 곳에는 사용 안 하길 바랄 뿐이다.

구현 방법 및 원리

우리가 흔히 사용 하고 있는 MS Window는 Multi-Tasking(여러 개의 프로그램의 실행하는 기능)을 지원하므로 여러 개의 프로그램 창을 띄어두고 필요에 따라 프로그램을 이동하며 사용을 많이 한다.

그러나 CPU는 system에 하나만이 존재 하기 때문에 (1이상이 존재 할 수도 있다) 어떤 시점에 실행 중인 Process(프로그램)는 우리가 실행 시켜둔 것 중의 하나 일 것이며, 특정 시간마다 순차적으로 돌아가며 모든 Process를 실행 시켜 줄 것이다. 우리에게 마치 동시에 실행 되는 것 처럼 보이지만 그렇지 않다는 것이다.

WINDOWS는 현재 실행 되고 있는 process를 List 형식으로 관리한다.
우리가 흔히 보는 작업 관리자를 실행 시켜보면 현재 동작하고 있는 모든 process를 볼 수 있을 것이다.
만약 작업 관리자에서 조차 특정 process가 보이지 않는다면 Hack 또는 특정 프로그램의 실행 되어 있는지 여부를 판단 하는 프로그램에 의해 발견 되지 않을 것이다.



그럼 우리는 WINDOWS 가 Process들을 정확이 어떻게 관리 하는 지를 알아야 하므로 WINDOWS의 Process 관리 매커니즘을 알아보도록 하자.

- Process 객체와 WINDOWS 관리 방법

사용자가 프로그램을 실행 시키면 WINDOWS는 Process라는 개념을 이용하여 프로그램을 표현한다. 그리고 Process를 이용하여 현재 실행 되어 있는 프로그램을 관리를 한다.
Process는 EPROCESS 라는 구조체로 되어 있으며 이 구조체에는 해당 Process에 대한 모든 정보들이 존재 한다.
아래는 XP에서 사용하는 EPROCESS에 대한 구조체 정보 이다.

```
kd> dt _EPROCESS
   +0x000 Pcb              : _KPROCESS
   +0x06c ProcessLock      : _EX_PUSH_LOCK
   +0x070 CreateTime       : _LARGE_INTEGER
```

```
+0x078 ExitTime             : _LARGE_INTEGER
+0x080 RundownProtect       : _EX_RUNDOWN_REF
+0x084 UniqueProcessId      : Ptr32 Void
+0x088 ActiveProcessLinks :   LIST_ENTRY
+0x090 QuotaUsage           : [3] Uint4B
+0x09c QuotaPeak            : [3] Uint4B
+0x0a8 CommitCharge         : Uint4B
+0x0ac PeakVirtualSize      : Uint4B
+0x0b0 VirtualSize          : Uint4B
+0x0b4 SessionProcessLinks : _LIST_ENTRY
+0x0bc DebugPort            : Ptr32 Void
+0x0c0 ExceptionPort        : Ptr32 Void
+0x0c4 ObjectTable          : Ptr32 _HANDLE_TABLE
+0x0c8 Token                : _EX_FAST_REF
+0x0cc WorkingSetLock       : _FAST_MUTEX
+0x0ec WorkingSetPage       : Uint4B
+0x0f0 AddressCreationLock : _FAST_MUTEX
+0x110 HyperSpaceLock       : Uint4B
+0x114 ForkInProgress       : Ptr32 _ETHREAD
+0x118 HardwareTrigger      : Uint4B
+0x11c VadRoot              : Ptr32 Void
+0x120 VadHint              : Ptr32 Void
+0x124 CloneRoot            : Ptr32 Void
+0x128 NumberOfPrivatePages : Uint4B
+0x12c NumberOfLockedPages : Uint4B
+0x130 Win32Process         : Ptr32 Void
+0x134 Job                  : Ptr32 _EJOB
+0x138 SectionObject        : Ptr32 Void
+0x13c SectionBaseAddress : Ptr32 Void
+0x140 QuotaBlock           : Ptr32 _EPROCESS_QUOTA_BLOCK
+0x144 WorkingSetWatch      : Ptr32 _PAGEFAULT_HISTORY
+0x148 Win32WindowStation : Ptr32 Void
+0x14c InheritedFromUniqueProcessId : Ptr32 Void
+0x150 LdtInformation       : Ptr32 Void
+0x154 VadFreeHint          : Ptr32 Void
+0x158 VdmObjects           : Ptr32 Void
```

```
+0x15c DeviceMap            : Ptr32 Void
+0x160 PhysicalVadList    : _LIST_ENTRY
+0x168 PageDirectoryPte : _HARDWARE_PTE
+0x168 Filler                : Uint8B
+0x170 Session               : Ptr32 Void
+0x174 ImageFileName       : [16] UChar
+0x184 JobLinks             : _LIST_ENTRY
+0x18c LockedPagesList    : Ptr32 Void
+0x190 ThreadListHead     : _LIST_ENTRY
+0x198 SecurityPort        : Ptr32 Void
+0x19c PaeTop               : Ptr32 Void
+0x1a0 ActiveThreads       : Uint4B
+0x1a4 GrantedAccess       : Uint4B
+0x1a8 DefaultHardErrorProcessing : Uint4B
+0x1ac LastThreadExitStatus : Int4B
+0x1b0 Peb                  : Ptr32 _PEB
+0x1b4 PrefetchTrace       : _EX_FAST_REF
+0x1b8 ReadOperationCount : _LARGE_INTEGER
+0x1c0 WriteOperationCount : _LARGE_INTEGER
+0x1c8 OtherOperationCount : _LARGE_INTEGER
+0x1d0 ReadTransferCount : _LARGE_INTEGER
+0x1d8 WriteTransferCount : _LARGE_INTEGER
+0x1e0 OtherTransferCount : _LARGE_INTEGER
+0x1e8 CommitChargeLimit : Uint4B
+0x1ec CommitChargePeak : Uint4B
+0x1f0 AweInfo               : Ptr32 Void
+0x1f4 SeAuditProcessCreationInfo : _SE_AUDIT_PROCESS_CREATION_INFO
+0x1f8 Vm                    : _MMSUPPORT
+0x238 LastFaultCount     : Uint4B
+0x23c ModifiedPageCount : Uint4B
+0x240 NumberOfVads        : Uint4B
+0x244 JobStatus            : Uint4B
+0x248 Flags                 : Uint4B
+0x248 CreateReported     : Pos 0, 1 Bit
+0x248 NoDebugInherit     : Pos 1, 1 Bit
+0x248 ProcessExiting     : Pos 2, 1 Bit
```

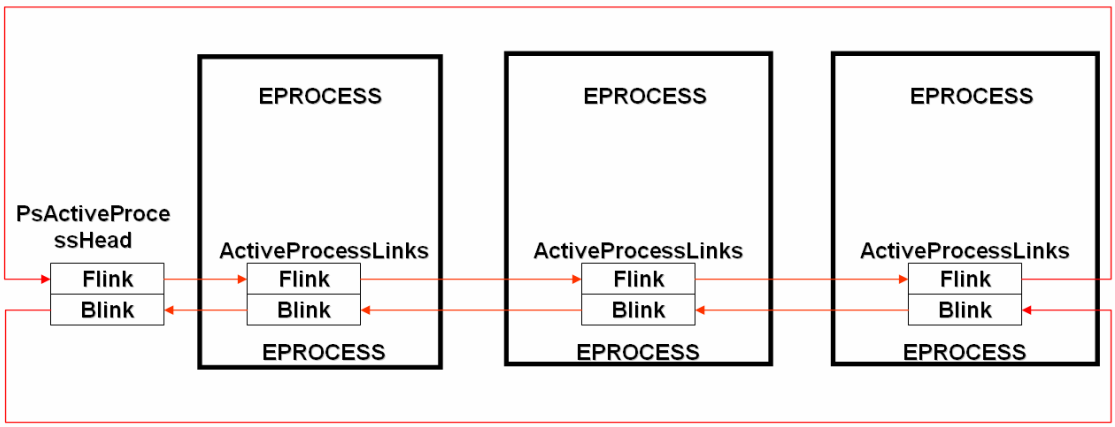
```
   +0x248 ProcessDelete      : Pos 3, 1 Bit
   +0x248 Wow64SplitPages    : Pos 4, 1 Bit
   +0x248 VmDeleted          : Pos 5, 1 Bit
   +0x248 OutswapEnabled     : Pos 6, 1 Bit
   +0x248 Outswapped         : Pos 7, 1 Bit
   +0x248 ForkFailed         : Pos 8, 1 Bit
   +0x248 HasPhysicalVad     : Pos 9, 1 Bit
   +0x248 AddressSpaceInitialized : Pos 10, 2 Bits
   +0x248 SetTimerResolution : Pos 12, 1 Bit
   +0x248 BreakOnTermination : Pos 13, 1 Bit
   +0x248 SessionCreationUnderway : Pos 14, 1 Bit
   +0x248 WriteWatch         : Pos 15, 1 Bit
   +0x248 ProcessInSession : Pos 16, 1 Bit
   +0x248 OverrideAddressSpace : Pos 17, 1 Bit
   +0x248 HasAddressSpace    : Pos 18, 1 Bit
   +0x248 LaunchPrefetched : Pos 19, 1 Bit
   +0x248 InjectInpageErrors : Pos 20, 1 Bit
   +0x248 Unused             : Pos 21, 11 Bits
   +0x24c ExitStatus         : Int4B
   +0x250 NextPageColor      : Uint2B
   +0x252 SubSystemMinorVersion : UChar
   +0x253 SubSystemMajorVersion : UChar
   +0x252 SubSystemVersion : Uint2B
   +0x254 PriorityClass      : UChar
   +0x255 WorkingSetAcquiredUnsafe : UChar
```

보는 봐와 같이 EPROCESS 구조체에는 많은 정보가 존재 한다. 그 중이 offset 0x88 을 보게 되면 **ActiveProcessLinks : _LIST_ENTRY** 가 있다.
이 field는 Process들 간의 연결 고리를 만들어주는 필드이면 구조체로 되어 있다.

```
kd> dt _LIST_ENTRY
   +0x000 Flink            : Ptr32 _LIST_ENTRY
   +0x004 Blink            : Ptr32 _LIST_ENTRY
```

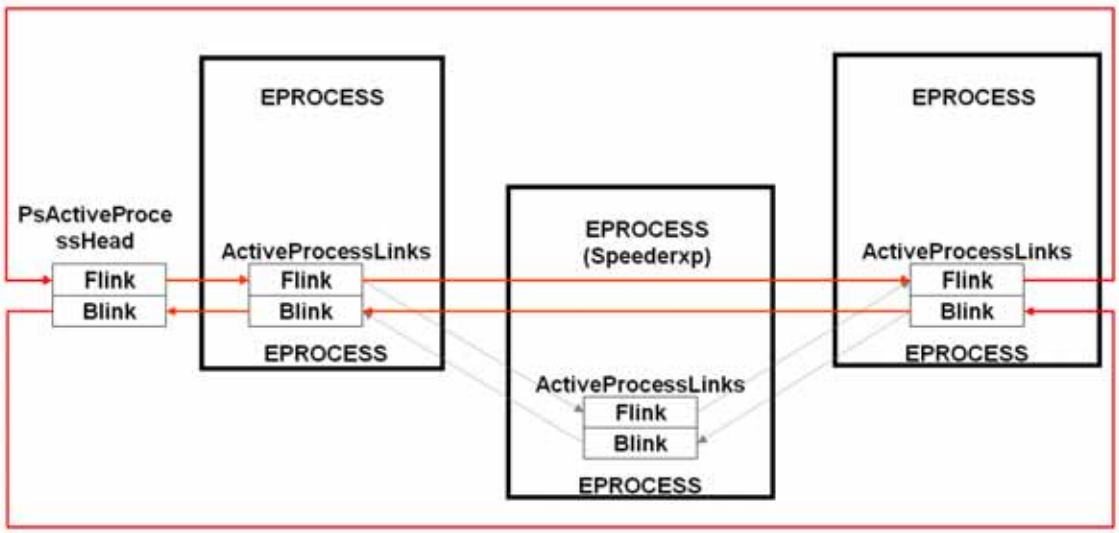
Flink 는 현재 Process 의 다음 EPROCESS 구조체를 가리킨다.

Blink 는 현재 Process 의 이전 EPROCESS 구조체를 가리킨다.



위와 같은 연결 고리가 존재 할 것이며(실제로 존재 한다) 이 리스트를 이용하여 Windows는 Process 관리 및 기타 작업 관리자 등이 process 목록 보이기를 원할 때 process 목록을 줄 것이다.

만약 이 연결 고리인 Linked list를 우리가 임의적으로 연결을 변경 한다면 어떻게 될 것인가?



위와 같이 우리가 숨기고 싶은 process SpeederXP의 앞뒤 Process의 Flink와 Blink를 조작 하여 SpeederXP Process의 연결 고리를 끊는다면, 어느 누구도 SpeederXP Process의 존재를 알지 못 할 것이다. (kernel mode에서 다른 technique을 이용한다면 알 수도 있다.)

적용시켜 보기

앞서 설명한 사실을 알았으니, 실습을 해보도록 하자.
우리는 kernel debug인 Soft-ICE를 이용하여 테스트를 해보도록 하겠다.

Step 1.

```
:proc
Process     KPEB      PID  Threads  Pri  User Time  Krnl Time  Status
 System     825B97B8    4       3C    8   00000000   0000025E  Ready
 smss       821B7908  2B4       3    B   00000001   00000001  Idle
 csrss      82116690  2E4       C    D   00000049   00000092  Ready
 ImUMgr     81D5C638  7F8       0    8   00000004   00000014  Deleting
 SpeederXP  816DA398  A7C       3    8   00000001   00000009  Ready
 Loader32   81DFECA0  1A4       3    8   00000012   00000041  Ready
 uedit32    81E0F120  C14       4    8   00000033   0000007F  Ready
 IEXPLORE   81F13020  CBC       E    8   00000019   00000054  Ready
*Idle       8055D0A0    0       2    0   00000000   0001B1F1  Running
```

Proc 명령을 이용하여 현재 SpeederXP의 존재 여부를 확인 한다.

Step 2.

```
:d 816da398+88
0010:816DA420 81DFED28  81D5C6C0  00000F28  0001004C      (.......(...L...
0010:816DA430 00000367  000010A8  000111DC  00000367      g...........g...
0010:816DA440 00000367  02B9B000  02A9F000  81DFED54      g...........T...
0010:816DA450 81E10CC4  00000000  E14491C8  E2BFA960      ..........D.`...
0010:816DA460 E2956D27  00000001  EE3EB97C  00000002      'm......|.>.....
0010:816DA470 00040001  00000000  816DA478  816DA478      ........x.m.x.m.
0010:816DA480 00000000  0001ACA9  00000001  EE120D04      ................
0010:816DA490 00000000  00040001  00000000  816DA49C      ..............m.
```

Speeder 의 KPEB 가 EPROCESS 구조체의 시작 주소이며, 이 주소를 기점으로 0x88 떨어진 곳이 ActiveProcessLinks 필드가 있다.
이 필드는 앞서 배운 것과 같이 Flink, Blink로 이전 Process와 다음 Process를 가르킨다.

```
:proc 81dfed28-88
Process     KPEB      PID  Threads  Pri  User Time  Krnl Time  Status
 Loader32   81DFECA0  1A4       3    8   00000025   00000077  Ready

:proc 81d5c6c0-88
Process     KPEB      PID  Threads  Pri  User Time  Krnl Time  Status
 ImUMgr     81D5C638  7F8       0    8   00000004   00000014  Deleting
```

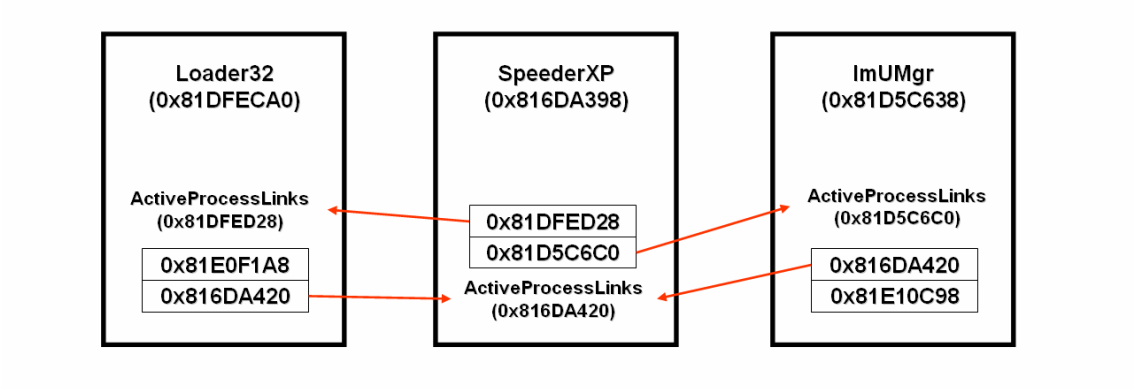
각각의 Flink와 Blink를 확인 해보면 위와 같은 Process가 보일 것이다.
그럼 왜 각 필드에 0x88 만큼을 빼준 것인가? 앞선 그림에서도 보았듯이 Flink와 Blink는 EPROCESS의 시작 주소를 가리키는 것이 아니라

EPROCESS의 ActiveProcessLinks를 가리키고 있기 때문이다.

SpeederXP process의 이전과 다음 EPROCESS의 ActiveProcessLinks 를 보면 서로 연결이 되어 있음을 확인 할 수 있다.

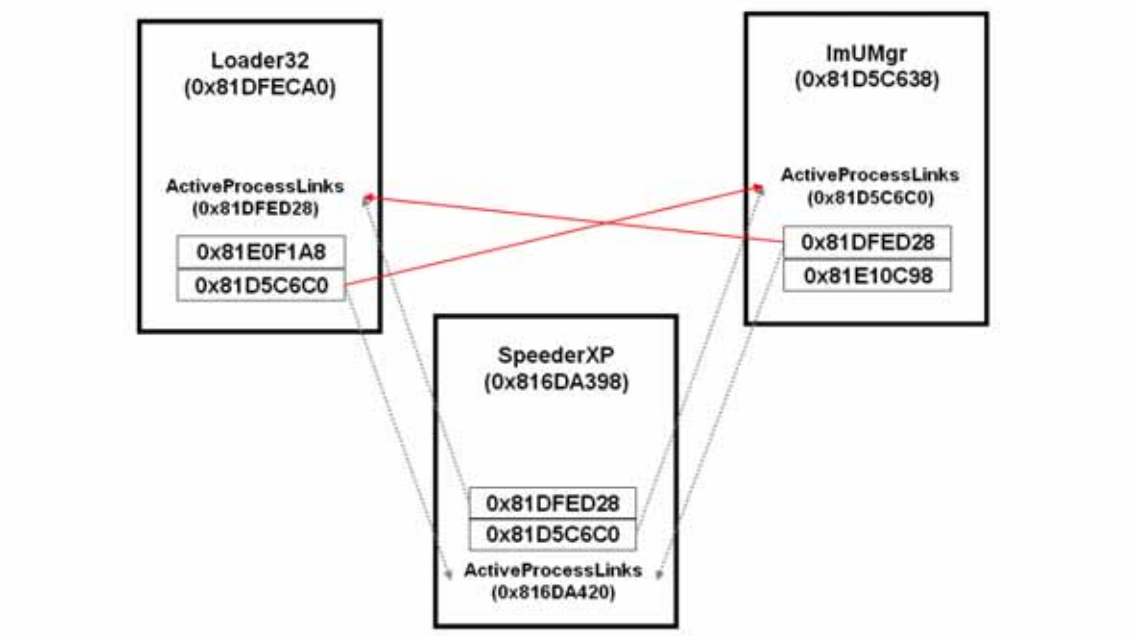
```
:d 81dfeca0+88
0010:81DFED28 81E0F1A8  816DA420  00001A68  0001785C      .... .m.h...Wx..
0010:81DFED38 000008E4  00001E20  0001C874  00000933      .... ...t...3...
0010:81DFED48 000008E4  0440B000  037AD000  81E0F1D4      ......@...z.....
0010:81DFED58 816DA44C  00000000  E14491C8  E1E0FC08      L.m.......D.....
0010:81DFED68 E1E8B582  00000001  EDD37D00  00000003      .........}......
0010:81DFED78 00040001  00000000  81DFED80  81DFED80      ................
0010:81DFED88 00000000  00019106  00000001  EDD37D04      ............}..
0010:81DFED98 00000003  00040001  00000000  81DFEDA4      ................
```

```
:d 81d5c638+88
0010:81D5C6C0 816DA420  81E10C98  00000000  00000000       .m............
0010:81D5C6D0 00000000  0000202E  0001AECC  000004B2      ..... ..........
0010:81D5C6E0 00000000  042E9000  033D9000  F8B99014      ..........=.....
0010:81D5C6F0 81E10CC4  00000000  E14491C8  00000000      ..........D.....
0010:81D5C700 E1392D23  00000001  EDFF8B80  00000001      #-9.............
0010:81D5C710 00040001  00000000  81D5C718  81D5C718      ................
0010:81D5C720 00000000  00012EF1  00000001  EDFF8C54      ............T...
0010:81D5C730 00000001  00040001  00000000  81D5C73C      ............<...
```



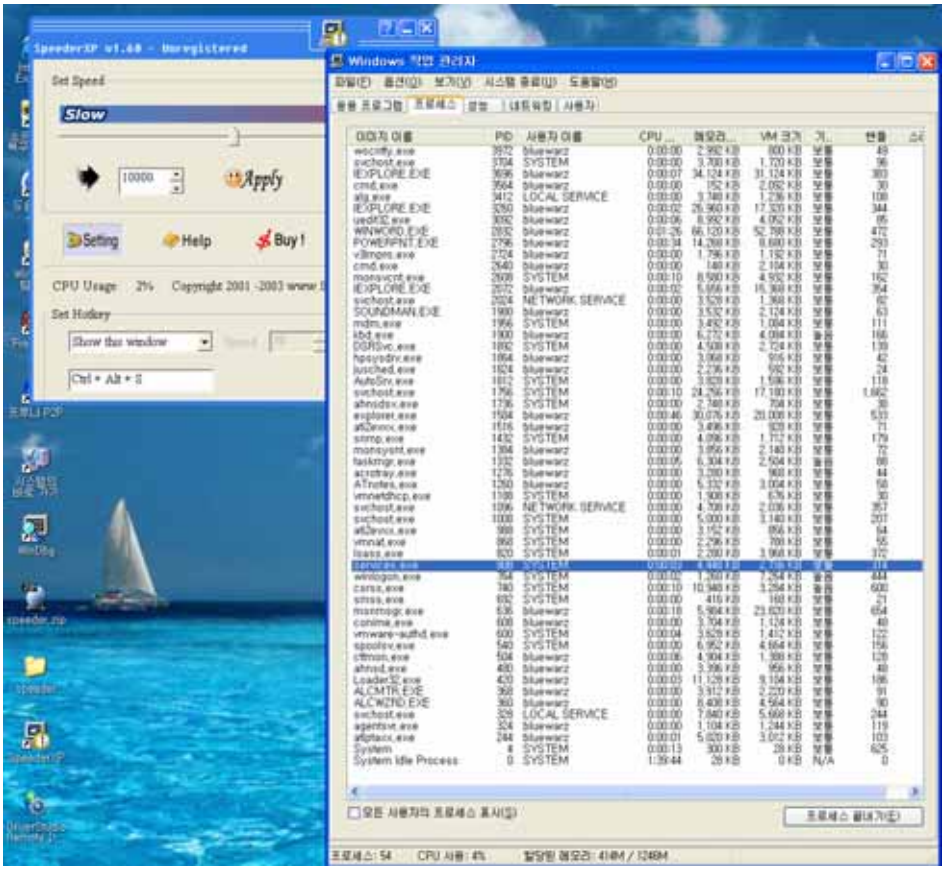
Step 3.

이런 상태에서 우리가 Loader32와 ImUMgr의 Flink와 Blink를 변경을 할 것이다.




```
:d 81dfed28
0010:81DFED28 81E0F1A8  81D5C6C0  00001A68  0001785C      ........h...Wx..
0010:81DFED38 000008E4  00001E20  0001C874  00000933      .... ...t...3...
0010:81DFED48 000008E4  0440B000  037AD000  81E0F1D4      ......@...z.....
0010:81DFED58 816DA44C  00000000  E14491C8  E1E0FC08      L.m.......D.....
0010:81DFED68 E1E8B581  00000001  EDD377FC  00000003      .........w......
0010:81DFED78 00040001  00000000  81DFED80  81DFED80      ................
0010:81DFED88 00000000  00019106  00000001  EDD37D04      ............}..
0010:81DFED98 00000003  00040001  00000000  81DFEDA4      ................
:d 81d5c6c0
0010:81D5C6C0 81DFED28  81E10C98  00000000  00000000      (...............
0010:81D5C6D0 00000000  0000202E  0001AECC  000004B2      ..... ..........
0010:81D5C6E0 00000000  042E9000  033D9000  F8B99014      ..........=.....
0010:81D5C6F0 81E10CC4  00000000  E14491C8  00000000      ..........D.....
0010:81D5C700 E1392D23  00000001  EDFF8B80  00000001      #-9.............
0010:81D5C710 00040001  00000000  81D5C718  81D5C718      ................
0010:81D5C720 00000000  00012EF1  00000001  EDFF8C54      ............T...
0010:81D5C730 00000001  00040001  00000000  81D5C73C      ............<...
```

Soft-ice를 이용하여 해당 offset를 수정 하면 된다.
이렇게 변경을 하게 되면SpeederXP의 Process를 감쪽같이 사라질 것이다.
Soft-Ice의 Proc 명령에 의해서도 나타나지 않을 것이며 작업 관리자에서 SpeederXP 를 감지 못하는 결과이다.



자. 이젠 필자가 좋아하는 온라인 농구 게임을 실행 시켜 봐야 겠다.
역시나 실행이 잘되며, 무적 basket man이 될 수 있을 것이다.

필자는 이 방식을 구현하면 하나 걱정스러운 점이 하나 있었다.
그건 Scheduling 매커니즘에서 임이로 Process list에서 제거한 Process는 CPU를 획듣할 기회를 얻지 못할지도 모른다는 생각 때문이었다.
그러나 다행이도 Scheduling 매커니즘은 Process를 기반으로 하는 것이 아니라 Thread 기반으로 하기 때문에 process list에서 제거가 되어진 Process도 아무 문제 없이 수행 되어지는 것 같다. (100% 로 문제가 없다는 말이 아니므로 필드에 나갈 제품은 이 방식을 채택 할 시 많은 테스트 등을 해야 할 것이다.)

마치며...
언제나 느끼는 것이지만 기본 원리를 이해 하고 있다면 가능한 것은 모두 구현 할 수 있는 능력을 키울 수 있는 것 같다.
Driver 혹은 System programming을 시작 하려는 개발자라면 windows의 구조를 좀더 자세히 이해하고 programming을 시작 하는 것이 후에 많은 도움

이 될 것이다.

본 컬럼에 나온 내용을 이용하여 나쁜 곳에 사용이 안되었으면 하는 바람이다. ^^

테스트 환경 및 Tools
OS : Windows XP SP2
Tools : Windbg 6.3
Soft-ICE 4.3.2

참고서적
Microsoft Windows Internals Fourth Edition. By Russinovich, Solomon
Undocumented Windows 2000 Secrets by Schreiber