기술문서

SSDT Hooking 탐지 기법

행정3 김범연

ccibomb@gmail.com

http://ccibomb.tistory.com

2009.9. 2.

○ 목차

1. SSDT

1-1. What is "SSDT" ?

SSDT 란 System Service Dispatch Table 의 약자로 System Service API 함수(Native API 라고도 부르며, 유저모드 애플리케이션이 Kernel Service를 받고자할 때 사용되는 함수를 의미)들의 주소를 담고 있는 테이블이다. 윈도우는 Win32, POSIX, OS/2 서브 시스템을 지원하는데, 이 서브 시스템들이 제공하는 서비스의 주소가 바로 SSDT라고 하는 커널 구조체에 의해 관리되는 것이다.

KeServiceDescriptorTable 은 커널이 제공하는 테이블로서 SSDT 의 주소를 가지고 있다. 총 4 개의 서비스 테이블로 구성되어 있고 각각의 서비스 테이블은 SDE(Service Descriptor Entry)라는 구조체로 이루어져 있다. SSDT 는 Ntoskrnl.exe 안에 구현된 윈도우의 핵심적인 시스템 서비스의 주소를 포함하고 있다.

```
typedef struct ServiceDescriptorEntry {
        PDWORD base
        PDWORD counter
        DWORD limit
        PBYTE arg
} SDE;

typedef struct ServiceDescriptorTable {
        SDE ntoskrnl;
        SDE win32k;
        SDE reserved[2];
} SDT;
```

그림 1. SDT 구조체

KeServiceDescriptorTableShadow 라고 하는 또 다른 테이블이 있다. 이는 커널 드라이버인 Win32k.sys 에 의해서 구현된 USER 와 GDI 서비스의 주소를 가지고 있다.

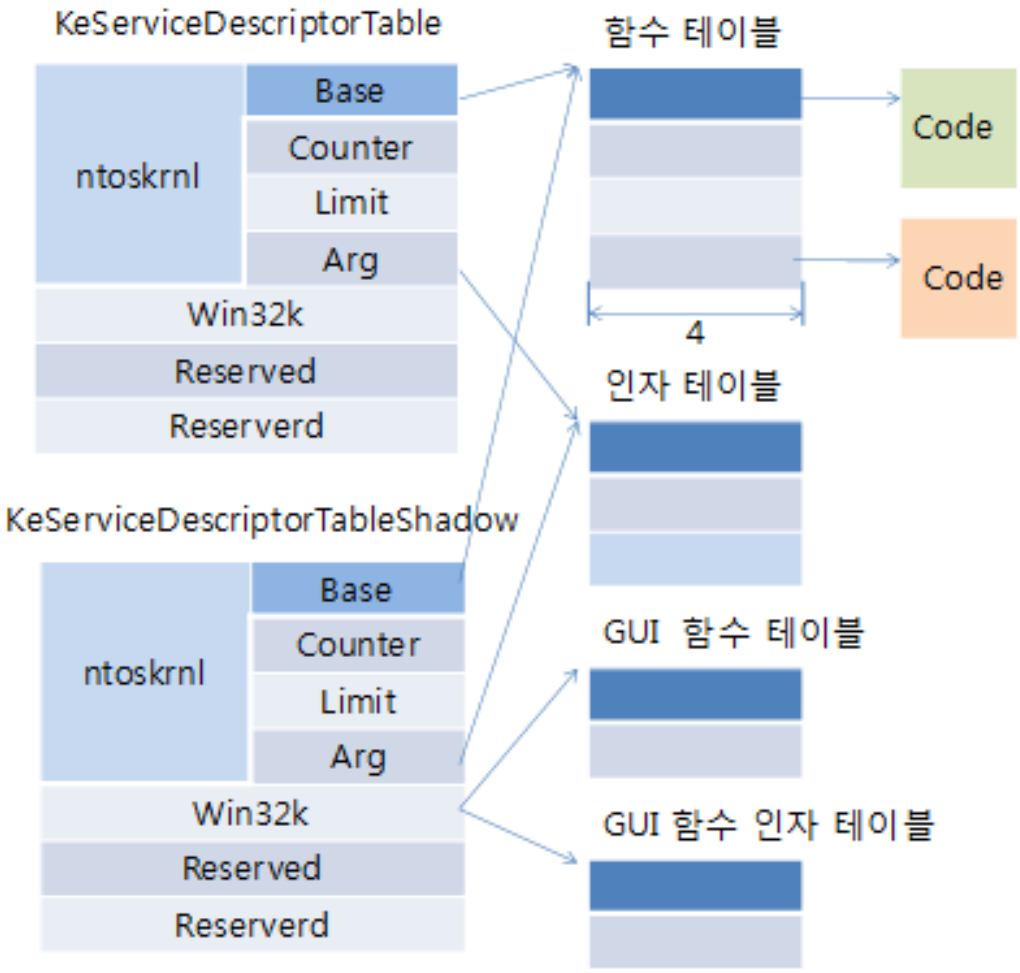


그림 2. SDT 구조

// KeServiceDescriptorTable : 일반적인 쓰레드가 사용
win32k GUI 관련 등록 X
reserved 는 비어있다.

KeServiceDescriptorTableShadow : GUI 쓰레드가 사용.
win32k 는 GUI 함수 관련

Field 별 역할 : base - 함수 테이블의 시작 주소 가리킴
counter - 각 함수가 몇 번 호출되었는지 기록
limit - 함수 테이블에 포함된 함수의 개수를 저장
arg - 인자크기를 저장하는 테이블 시작주소 가리킴

1-2. SSDT 살펴보기

1-2-1. Windbg 이용하여 직접 살펴보기

i. KeServiceDescriptorTable 을 살펴본다.

```
kd> dd KeServiceDescriptorTable
80552180  80501030 00000000 0000011c 805014a4
80552190  00000000 00000000 00000000 00000000
805521a0  00000000 00000000 00000000 00000000
805521b0  00000000 00000000 00000000 00000000
805521c0  00002710 bf80da45 00000000 00000000
805521d0  f8b6ba80 8223f1e0 82222a90 806dff40
805521e0  00000000 00000000 e1f39eb4 000000a9
805521f0  2a5acf74 01ca0b33 00000000 00000000
```

// 검은 상자가 ServiceDescriptor Table 이고 빨간 상자가 ServiceDescriptorEntry 이다. ServiceDescriptorEntry 는 상기 기술대로, 4 Bytes 씩 base, counter, limit, arg 필드로 이루어져 있다.

ii. Window native API 주소들을 기록한 함수 테이블을 가리키는 Base 의 내용을 보았다.

```
kd> dd 80501030
80501030  8059849a 805e5666 805e8ec4 805e5698
80501040  805e8efe 805e56ce 805e8f42 805e8f86
80501050  8060a5da 8060b84e 805e0a08 805e0660
80501060  805c9684 805c9634 8060ac00 805aa088
80501070  8060a218 8059c910 805a44da 805cb162
80501080  804fed04 805bce0e 8056abe6 805341dc
80501090  806038ea 805b0714 805e93fe 806187a2
805010a0  805ed8f0 80598b88 806189f6 8059843a
```

iii. Base 가 가리키고 있는 함수테이블 중 각각 4byte 의 값은 함수의 주소를 기록하고 있을 테니 각 함수의 주소 내용을 보았다.

```
kd> ln 8059849a
(8059849a)   nt!NtAcceptConnectPort   |  (80598b88)   nt!NtCompleteConnectPort
Exact matches:
    nt!NtAcceptConnectPort = <no type information>
kd> ln 805e5666
(805e5666)   nt!NtAccessCheck   |  (805e5698)   nt!NtAccessCheckByType
Exact matches:
    nt!NtAccessCheck = <no type information>
kd> ln 805e8ec4
(805e8ec4)   nt!NtAccessCheckAndAuditAlarm   |  (805e8efe)   nt!NtAccessCheckByTypeAndAuditAlarm
Exact matches:
    nt!NtAccessCheckAndAuditAlarm = <no type information>
```

// ln = list nearest symbols

ln [address] : 주소 값에 가장 가까운 심볼의 값

```
kd> u 805e8ec4
nt!NtAccessCheckAndAuditAlarm:
805e8ec4 8bff            mov     edi,edi
805e8ec6 55              push    ebp
805e8ec7 8bec            mov     ebp,esp
805e8ec9 33c0            xor     eax,eax
805e8ecb 50              push    eax
805e8ecc ff7530          push    dword ptr [ebp+30h]
805e8ecf ff752c          push    dword ptr [ebp+2Ch]
805e8ed2 ff7528          push    dword ptr [ebp+28h]
```

// 함수의 내용까지 알 수 있다.

1-2-2. Iceword 로 SSDT 살펴보기

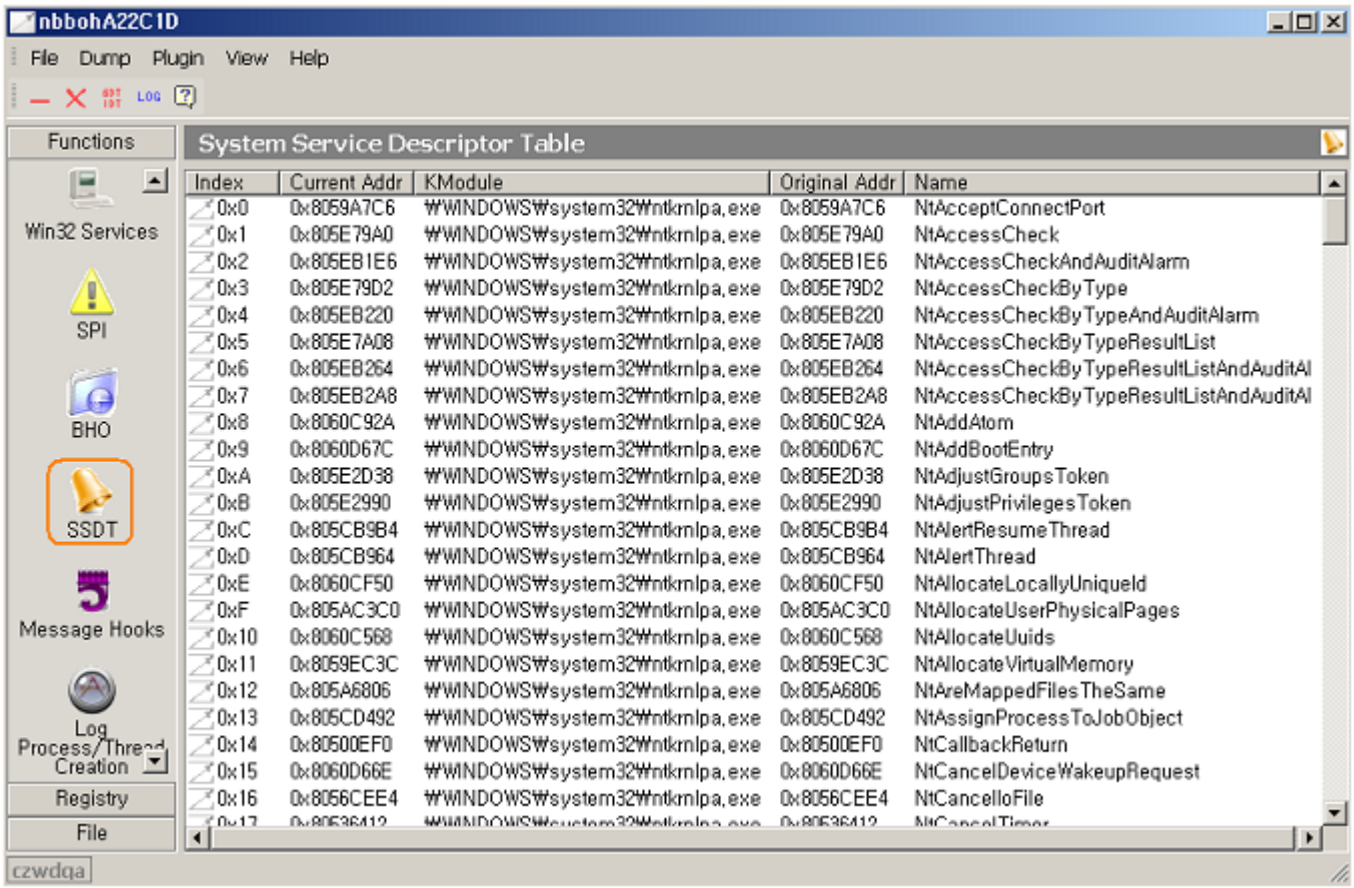


그림 3. Iceword로 살펴본 SSDT

2. SSDT Hooking

2-1. SSDT Hooking 이란?

윈도우 운영체제에서는 ZwQuerySystemInformation 함수를 이용해서 다양한 종류의 정보를 알아낼 수 있다. 작업관리자 (Taskmgr.exe)는 이를 이용해서 실행 중인 Process List 를 구한다.

루트킷이 SSDT 의 NtQuerySystemInformation 함수의 주소를 교체해서 자신의 함수가 먼저 호출되게 만들었다면, 그 루트킷 함수 내에서 원래의 NtQuerySystemInformation 함수를 호출해서 얻은 결과값을 변경시킬 수 있다.

SSDT 후킹은 윈도우즈 API 가 커널 모드에서 서비스를 받기 위해 필요한 SSDT(System Service Descriptor Table) 의 내용을 조작하는 커널 모드 후킹 방법 중에 하나이다. SSDT 는 프로세스에 독립적이고 커널 주소 공간에 전역적으로 올라와 있으므로, 특별한 조작을 가하지 않는다면 모든 프로세스가 같은 SSDT 를 가지고 있다. 커널모드에서 전역적으로 윈도우즈 서비스 함수를 가로챌 필요가 있을때 주로 SSDT 후킹을 많이 사용한다.

2-2. 실습

2-2-1. xcurelab.exe 를 실행하면 taskmgr 창에 나타난다.

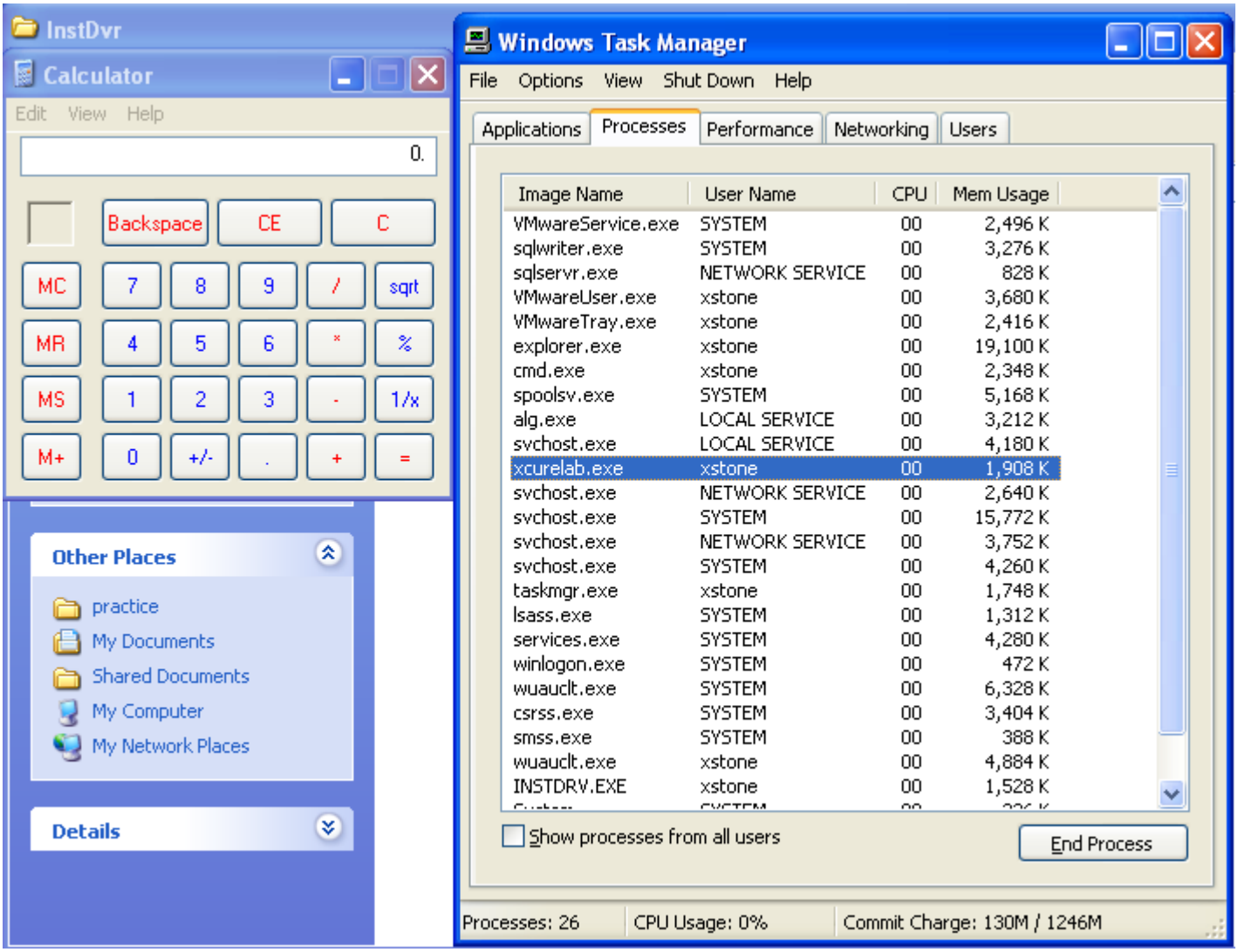


그림 4. 작업관리자(taskmgr) – xcurelab.exe 가 보인다.

2-2-2. KeServiceDescriptorTable 의 Base 가 가리키고 있는 함수테이블 중 QuerySystemInformation 함수를 살펴본다.

```
kd> dd poi(KeServiceDescriptorTable)+ad*4
805012e4  8060633e 806081c0 8060ba36 80607a78
805012f4  80616c8c 805acd2a 8056fe50 805c5d2e
80501304  80540018 8060975a 80570618 80570ba6
80501314  80599cd6 805a81ec 805c727e 8060c244
80501324  80609c18 8056be56 80637b94 806185f8
80501334  8061a7b2 80599356 8059a31e 80599d26
80501344  80599640 805bcda0 805968b4 80596be0
80501354  805bcbae 80603ccc 8051d12e 80616fda
```

// KeServiceDescriptorTable 의 첫 주소의 내용은 Base 가 가리키는 주소이기 때문에 poi(KeServiceDescriptorTable)는 함수테이블의 첫주소를 나타낸다.

QuerySystemInformation 함수는 ad 번째에 있기 때문에 ad*4(함수 하나의 크기는 4byte)를 해준다.

2-2-3. SSDT Hooking 을 하기 전까지는 정보가 변하지 않았다.

2-2-4. 함수를 가리키는 주소가 적혀있는 Table (NtQueryInformationSystem)의 값을 바꾸어보자.

(* 본 문서에서는 SSDT Hooking의 자세한 원리는 생략하고 미리 제작한 InstDrv(dev loader), hideprocess.sys(kernel 조작 코드)를 이용하였다. 자세한 내용은 이동수(alonglog@is119.jnu.ac.kr) 저 기술문서인 "SSDT HOOKING을 이용한 프로세스와 파일 숨기기"를 참고하기 바란다.)

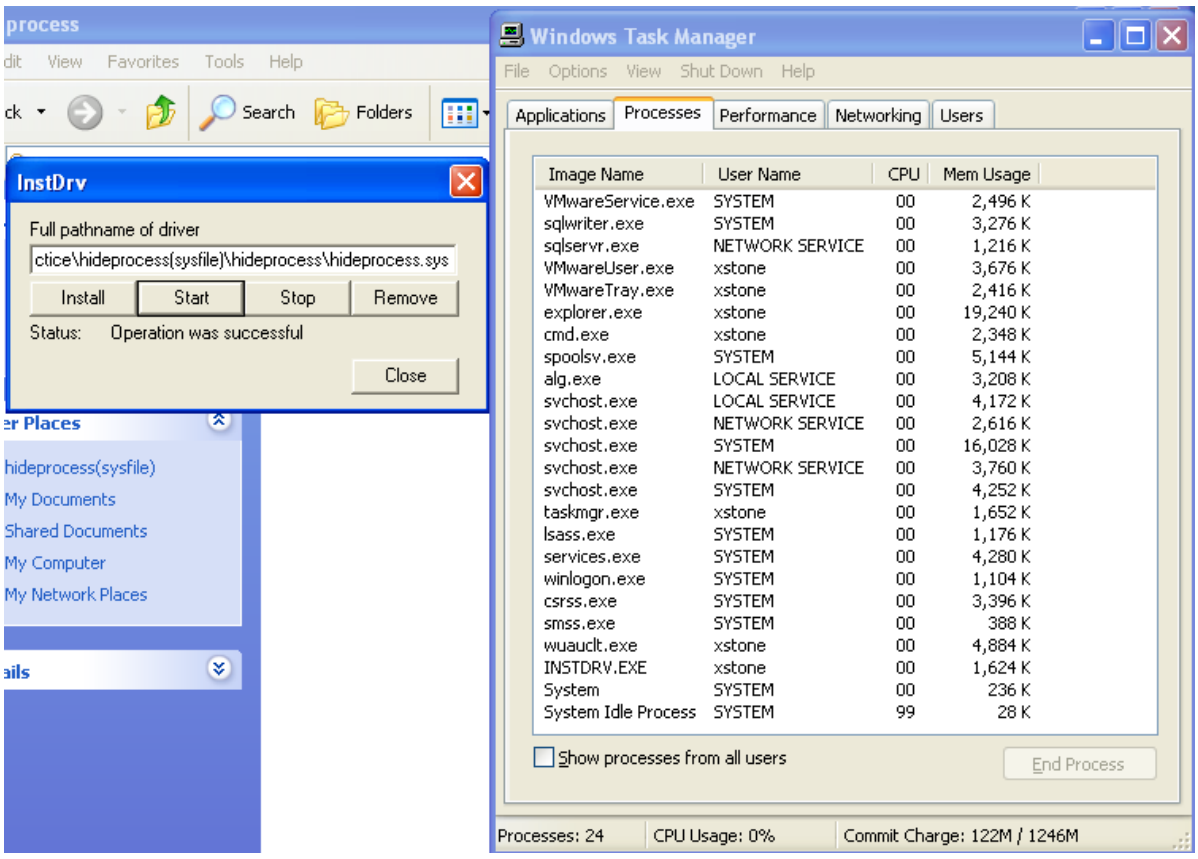


그림 5. 작업관리자(taskmgr) – xcurelab.exe 가 보이지 않는다.

// InstDrv (dev loader 역할) 를 통해서 커널을 통해 hideprocess.sys (코드 자체가 커널 영역에 있으며, 커널 영역을 조작하는 역할) 를 투여 후, start 를 누르니 xcurelab.exe 가 사라졌다.

hideprocess.sys 를 통해서 QueryInformationSystem 을 가리키는 함수주소 테이블을 다른 주소로 덮어씌워 다른 함수가 실행되도록 한다. 다른 함수는 xcurelab.exe 를 찾을 수 없도록 하는 내용의 시스템파일이다.

c.f) 최근 루트킷은 대부분 dev (kernel 서비스 확장) 형태로서, .exe 보다는 .sys, .dll 을 많이 사용한다.

2-2-5. SSDT Hooking 이 끝난 후이다. 함수 테이블 (QuerySystemInformation)의 내용이 변경되었는지 살펴보자.

```
kd> dd poi(KeServiceDescriptorTable)+ad*4
805012e4  f8cf7498 806081c0 8060ba36 80607a78
805012f4  80616c8c 805acd2a 8056fe50 805c5d2e
80501304  80540018 8060975a 80570618 80570ba6
80501314  80599cd6 805a81ec 805c727e 8060c244
80501324  80609c18 8056be56 80637b94 806185f8
80501334  8061a7b2 80599356 8059a31e 80599d26
80501344  80599640 805bcda0 805968b4 80596be0
80501354  805bcbae 80603ccc 8051d12e 80616fda
```

// QuerySytemInformation 이 가리키는 함수의 주소들이 달라졌다.

2-2-6. 변경된 함수의 주소 (이 중 첫 번째인 f8cf7498 에 한번 가보자)에 가서 내용을 한번 보자.

```
kd> u f8cf7498
*** ERROR: Module load completed but symbols could not be loaded for hideprocess.sys
hideprocess+0x498:
f8cf7498 8bff            mov     edi,edi
f8cf749a 55              push    ebp
f8cf749b 8bec            mov     ebp,esp
f8cf749d 53              push    ebx
f8cf749e 56              push    esi
f8cf749f ff7514          push    dword ptr [ebp+14h]
f8cf74a2 8b750c          mov     esi,dword ptr [ebp+0Ch]
f8cf74a5 ff7510          push    dword ptr [ebp+10h]
```

// f8cf7498 의 내용은 hideprocess.sys 에 있는 symbol 정보를 제대로 로딩하지 못한 모습이다.

3. SSDT Hooking 탐지 기법

* Volatility 를 통하여 SSDT Hooking 감지하기

3-1. mdd 로 ssdt hooking 된 상태의 memory 를 새롭게 dump 한다.

```
mdd -o c:₩hook.dd
```

// 물론 mdd.exe 와 hook.dd 의 위치는 해당 시스템이 아닌 usb 와 같이 다른 드라이브에서 실행하는 것이 좋다.

mdd : Mantech Memory DD (www.mantech.com/msma/mdd.asp)

3-2. volatility tool 내에 ssdt plugin 을 설치 후 메모리 내 존재하는 SDE 들을 모두 찾는다.

그림 6. win32k.sys와 ntoskrnl.exe 를 포함하여 출력한 모습

// 정상적인 함수들은 win32k.sys 와 ntoskrn1.exe 로 부터 불려와서 행해지는 모습이다.

3-3. 정상적인 SDE 인 ntoskrnl.exe 와 win32k.sys 를 제외하고 찾는다.

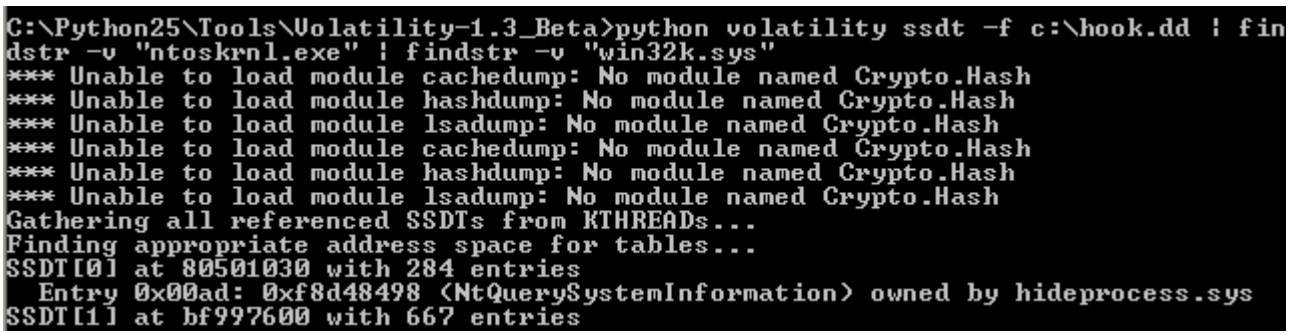
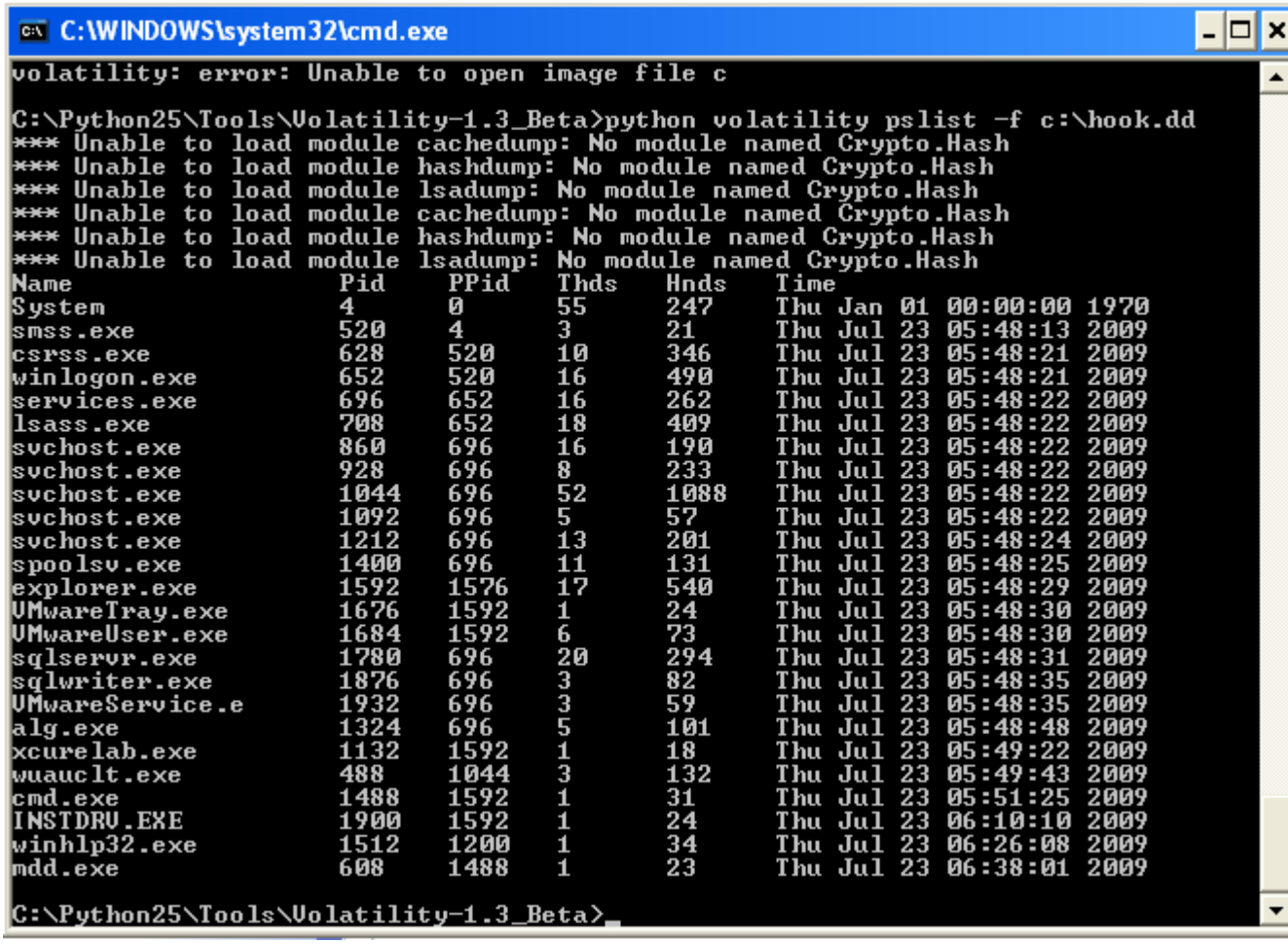


그림 7. win32k.sys와 ntoskrnl.exe 를 제외하고 출력한 모습

// findstr 은 grep 과 같은 역할을 하는 명령어로, 윈도우에서 지원해준다.
-v 옵션은 reverse 옵션으로, grep -v 와 동일하다.
정상적인 SDE 가 아닌 hideprocess.sys 에서 유래된 NtQuerySystemInformation 확인!

3-4. volatility 의 pslist 로도 SSDT Hooking 탐지가 가능하다.



// pslist 로도 xcurelab.exe 가 실행중임을 알 수 있다.
pslist 는 linkedlist 를 통해서 프로세스를 찾아내는 것이다.
SSDT Hooking 방법은 linkedlist 를 조작하는 것이 아니라 SSDT 의 함수 주소가 적혀있는 테이블을 조작하여 다른 함수를 실행하도록 하여 xcurelab.exe 를 탐지하지 못하도록 한다.

3-5. tasklist 로는 탐지되지 않는다. (xcurelab.exe)

```
C:\Python26\Tools\Volatility-1.3_Beta>tasklist

이미지 이름                     PID 세션 이름          세션#  메모리 사용
========================= ====== ================ ======== ============
System Idle Process               0                        0         28 K
System                            4                        0        296 K
smss.exe                        560                        0        436 K
csrss.exe                       620                        0      7,072 K
winlogon.exe                    644                        0      3,556 K
services.exe                    688                        0      4,672 K
lsass.exe                       700                        0      1,472 K
vmacthlp.exe                    852                        0      2,744 K
svchost.exe                     916                        0      3,652 K
svchost.exe                     984                        0      4,344 K
svchost.exe                    1096                        0     19,192 K
svchost.exe                    1156                        0      3,708 K
explorer.exe                   1396                        0     19,444 K
spoolsv.exe                    1452                        0      5,548 K
jusched.exe                    1560                        0      2,012 K
VMwareTray.exe                 1568                        0      3,520 K
VMwareUser.exe                 1576                        0      6,388 K
ctfmon.exe                     1584                        0      4,120 K
VMwareService.exe              1700                        0      4,236 K
svchost.exe                    2032                        0      4,028 K
conime.exe                     1876                        0      3,204 K
INSTDRV.EXE                    1028                        0      3,496 K
cmd.exe                         504                        0      3,052 K
tasklist.exe                    784                        0      4,560 K
wmiprvse.exe                    872                        0      5,900 K
```

// tasklist 는 linkedlist 를 통해 프로세스를 찾는데, Native API 를 이용한다.
따라서 SSDT Hooking 탐지가 되지 않는다.

3-6. SSDT Hooking 이 나쁜 쪽으로만 쓰이는 것이 아니다.
보안용으로 파일을 실행하기 전에 바이러스 검사 등을 할 때는 Hooking 방법을 통해서 해야하므로 꼭 나쁜 쪽으로 사용하는 것이 아니다. 그래서 검사를 통해 SSDT Hooking 이 발견하면 보안용인지 아닌지 알아보고 나서 검사하는 것이 중요하다.

4. 참고문헌

Greg Hoglund • Jamie Butler, "루트킷 : 윈도우 커널 조작의 미학" , 에이콘, July 2008

SSDT Hooking의 이해, http://blog.stsc.co.kr/, 2009.4.20.

루트킷-3: SDT 후킹의 창과 방패, http://www.alicerock.com/alice/, 2009.6.3.