

메모리 분석 방안



JK Kim

@pr0neer

forensic-proof.com

proneer@gmail.com

1. 메모리 이해
2. 메모리 획득
3. 메모리 분석
4. 챌린지

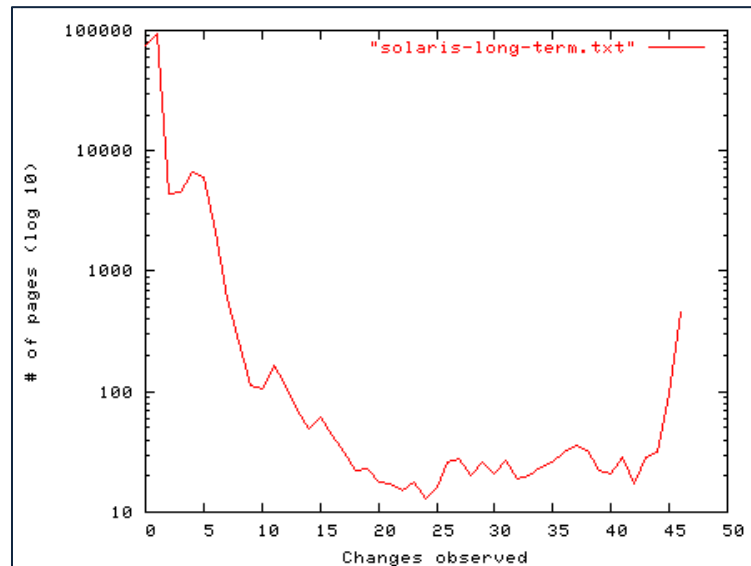
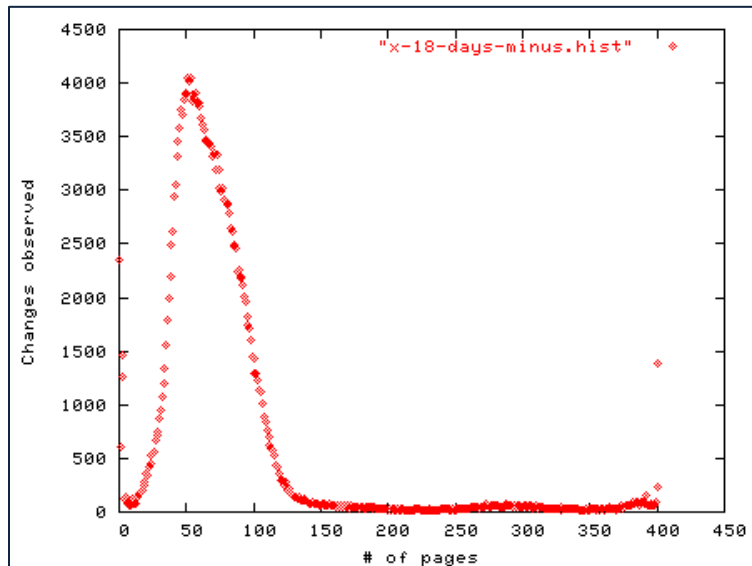
메모리 이해

메모리 포렌식 목적

- 프로세스(악성코드, 서비스 등)의 행위 탐지
- 네트워크 연결 정보
- 사용자 행위
- 복호화, 언패킹, 디코딩된 데이터
- 패스워드와 암호 키 획득

메모리 지속성

- Red Hat 6.1과 Solaris 8 테스트 (<http://www.porcupine.org/forensics/forensic-discovery/chapter8.html>)

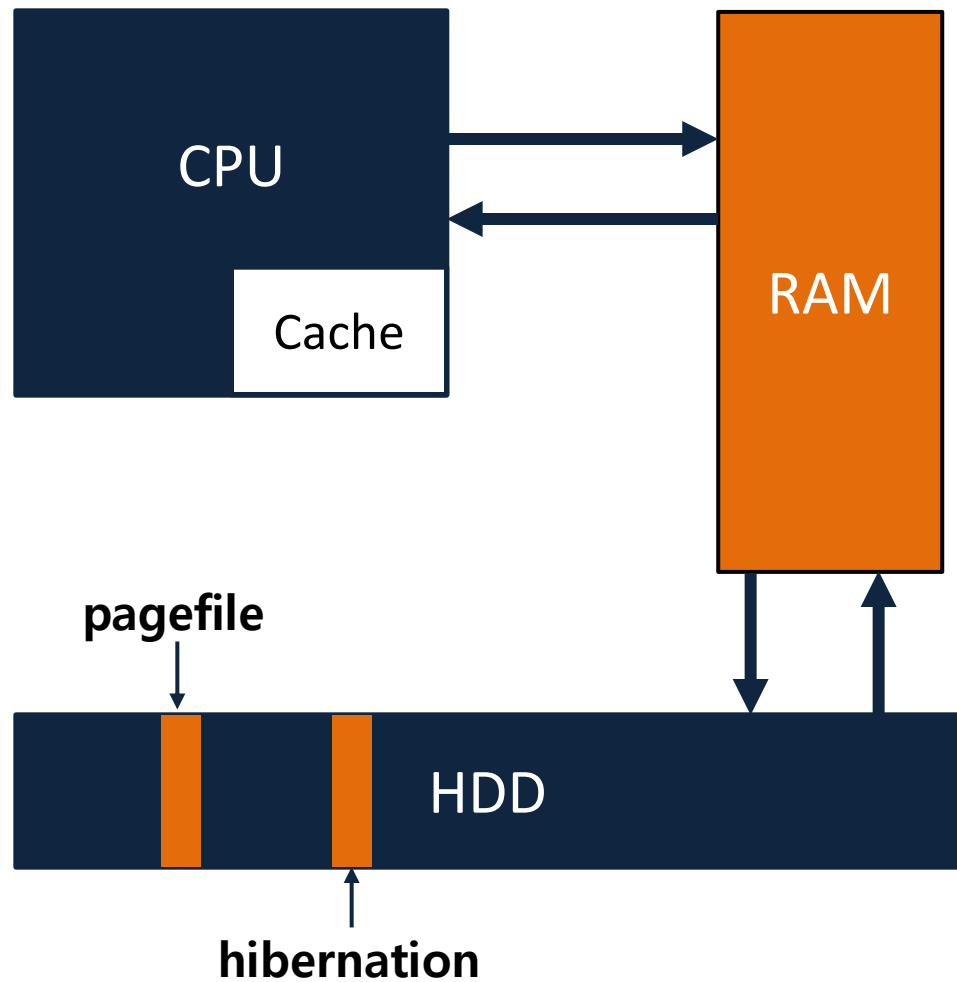


- 402시간(16.75일) 동안 메모리 관찰
 - 일당 65,000 건의 요청 처리시스템
 - 메모리 페이지 해시를 통해 탐지
 - 2,350/256,000 페이지 그대로 유지
- 41.2일 동안 메모리 페이지 관찰
 - 전체 메모리 페이지의 86% 변화 없음
 - 매일 0.4%만 변화

메모리 이해

메모리 포렌식 대상

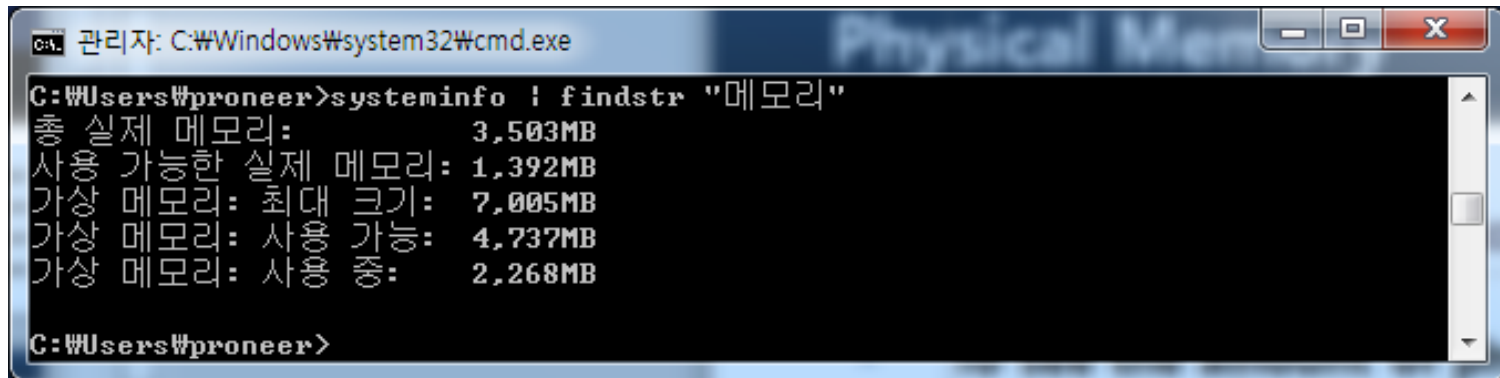
- 물리메모리
- 페이지 파일
- 하이버네이션 파일



메모리 이해

물리메모리 소개

- 물리메모리 확인 방법



A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.exe'. The command prompt shows the command 'systeminfo ! findstr "메모리"' and its output. The output lists physical and virtual memory statistics in MB. The prompt is 'C:\Users\proneer>'.

```
C:\Windows\system32\cmd.exe
C:\Users\proneer>systeminfo ! findstr "메모리"
총 실제 메모리: 3,503MB
사용 가능한 실제 메모리: 1,392MB
가상 메모리: 최대 크기: 7,005MB
가상 메모리: 사용 가능: 4,737MB
가상 메모리: 사용 중: 2,268MB

C:\Users\proneer>
```

- 시스템 등록정보를 통해서도 확인 가능

물리 주소 확장 (PAE, Physical Address Extension)

- 운영체제 버전, 하드웨어 플랫폼, 구성 방식에 따라 접근 가능한 메모리 크기

Version	Limit for 32-bit Hardware	Limit for 64-bit Hardware
Windows XP Home, Mediacenter	4 GB	N/A
Windows XP Professional	4 GB	128 GB
Windows Server 2003 Standard	4 GB	32 GB
Windows Server 2003 R2 Enterprise, Datacenter	64 GB	1 TB
Windows Vista Business, Enterprise, Ultimate	4 GB	128 GB
Windows Server 2008 Standard, Web	4 GB	32 GB
Windows Server 2008 Enterprise, Datacenter	4 GB	2 TB
Windows 7 Home Premium	4 GB	16 GB
Windows 7 Pro, Enterprise, Ultimate	4 GB	192 GB
Windows Server 2008 R2 Standard	N/A	32 GB
Windows Server 2008 R2 Enterprise	N/A	2 TB

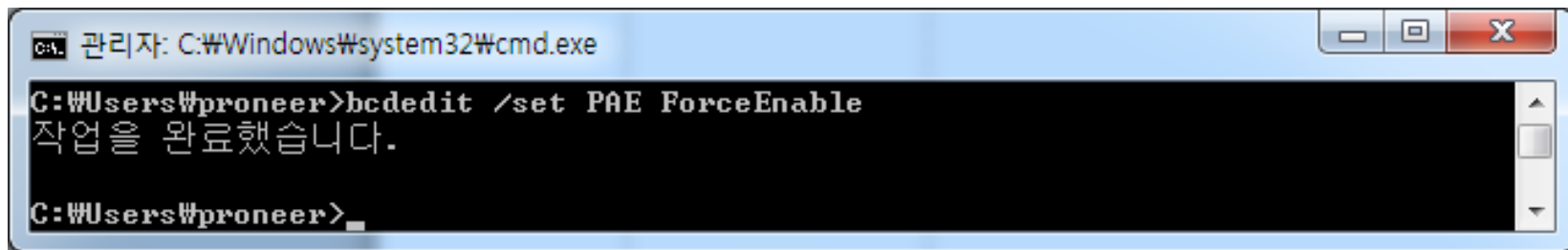
물리 주소 확장 (PAE, Physical Address Extension)

■ 물리 주소 확장이란?

- 물리적 주소 지정 비트를 32비트에서 36비트로 확장 (4GB → 64GB)
- 접근 가능한 물리 주소 공간 증가

■ 윈도우 Vista 이상에서 PAE 설정

- bcdedit (BCD, Boot Configuration Data)

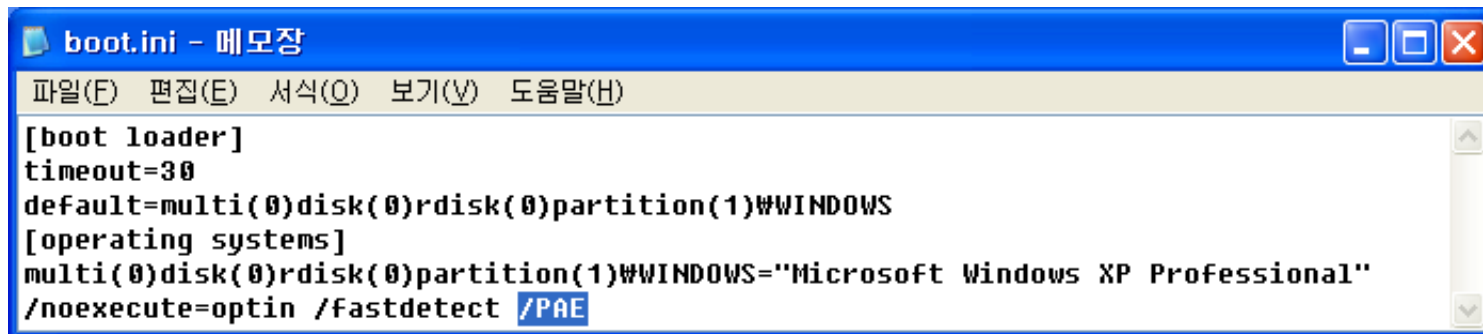


```
관리자: C:\Windows\system32\cmd.exe
C:\Users\proneer>bcdedit /set PAE ForceEnable
작업을 완료했습니다.
C:\Users\proneer>
```

메모리 이해

물리 주소 확장 (PAE, Physical Address Extension)

- 원도우 2003 이전에서 PAE 설정



The screenshot shows a Notepad window titled "boot.ini - 메모장". The menu bar includes "파일(F)", "편집(E)", "서식(O)", "보기(V)", and "도움말(H)". The text content is as follows:

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional"
/noexecute=optin /fastdetect /PAE
```

The "/PAE" switch at the end of the last line is highlighted with a blue selection box.

- %SystemDrive%\boot.ini 파일에 /PAE 스위치 추가
- 확장된 주소 공간 접근을 위해 AEW(Address Windowing Extensions) API 사용
- PAE로 확장된 공간은 보통 램 디스크(RAM Disk)로 활용

주소 윈도우 확장 (AWE, Address Windowing Extensions)

▪ 확장된 메모리 접근 API

- 물리 주소 확장(PAE)에 의해 확장된 메모리를 접근하기 위한 API (winbase.h)

▪ Lock Pages in Memory

- 사용자 응용프로그램의 경우, 가상 메모리로 데이터를 페이징하지 않도록 "Lock Pages in Memory" 가 설정되어야 함

AWE Routine	Description
VirtualAlloc()	Reserves a region in the linear address space of the calling process
VirtualAllocEx()	Reserves a region in the linear address space of the calling process
AllocateUserPhysicalPages()	Allocate pages of physical memory to be mapped to linear memory
MapUserPhysicalPages()	Map allocated pages of physical memory to linear memory
MapUserPhysicalPagesScatter()	Map allocated pages of physical memory to linear memory
FreeUserPhysicalPages()	Release physical memory allocated for use by AWE

데이터 실행 방지 (DEP, Data Execution Prevention)

■ 데이터 실행 방지

- 스택, 데이터 세그먼트, 힙과 같은 메모리 페이지를 실행 불가능하도록 설정
- 버퍼 오버플로우 등의 공격 방지

하드웨어 강제 DEP 설정 비트	CPU 제조사	설명
NX	AMD	No-eXecute page-protection
XD	Intel	eXecution Disable bit

■ 하드웨어 강제 (Hardware-enforced)

- CPU의 NX/XD 비트로 설정하며 OS와 사용자 애플리케이션 모두에서 사용 가능
 - ✓ NX/XD 비트를 지원하는 CPU에서만 동작
- PAE가 동작할 때만 설정 가능(NX/XD 비트를 사용할 경우 자동으로 PAE로 부팅)

■ 소프트웨어 강제 (Software-enforced)

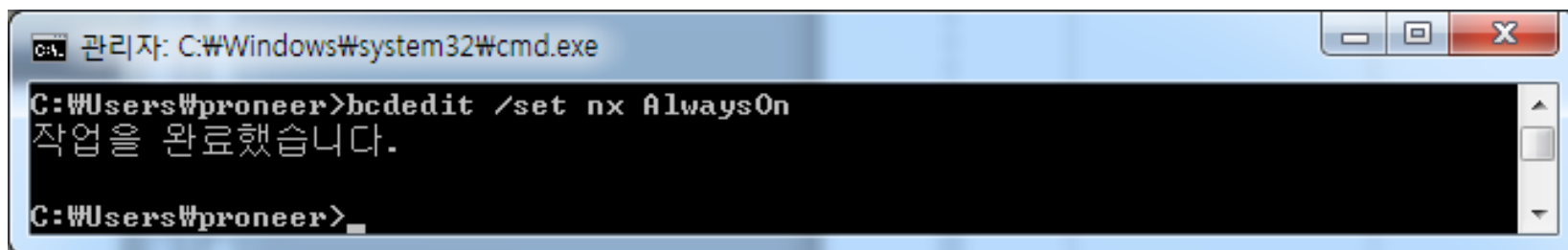
- 비주얼 스튜디오의 /SafeSEH 링커 옵션(SEH, Structured Exception Handler)

메모리 이해

데이터 실행 방지 (DEP, Data Execution Prevention)

■ 윈도우 Vista/7, 서버 2008에서 하드웨어 강제 DEP 설정

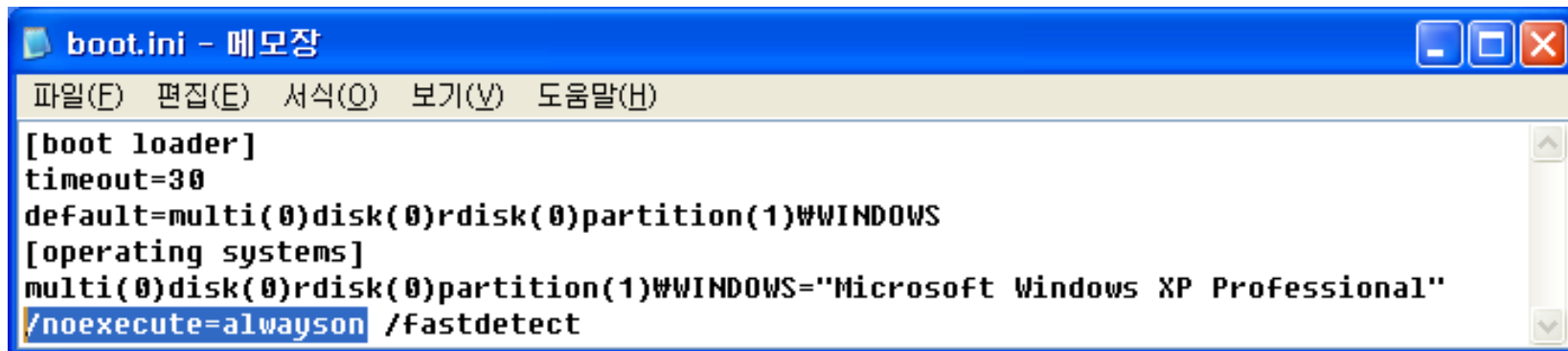
- bcdedit /set nx AlwaysOn



```
관리자: C:\Windows\system32\cmd.exe
C:\Users\proneer>bcdedit /set nx AlwaysOn
작업을 완료했습니다.
C:\Users\proneer>_
```

■ 윈도우 서버 2003 이하에서 하드웨어 강제 DEP 설정

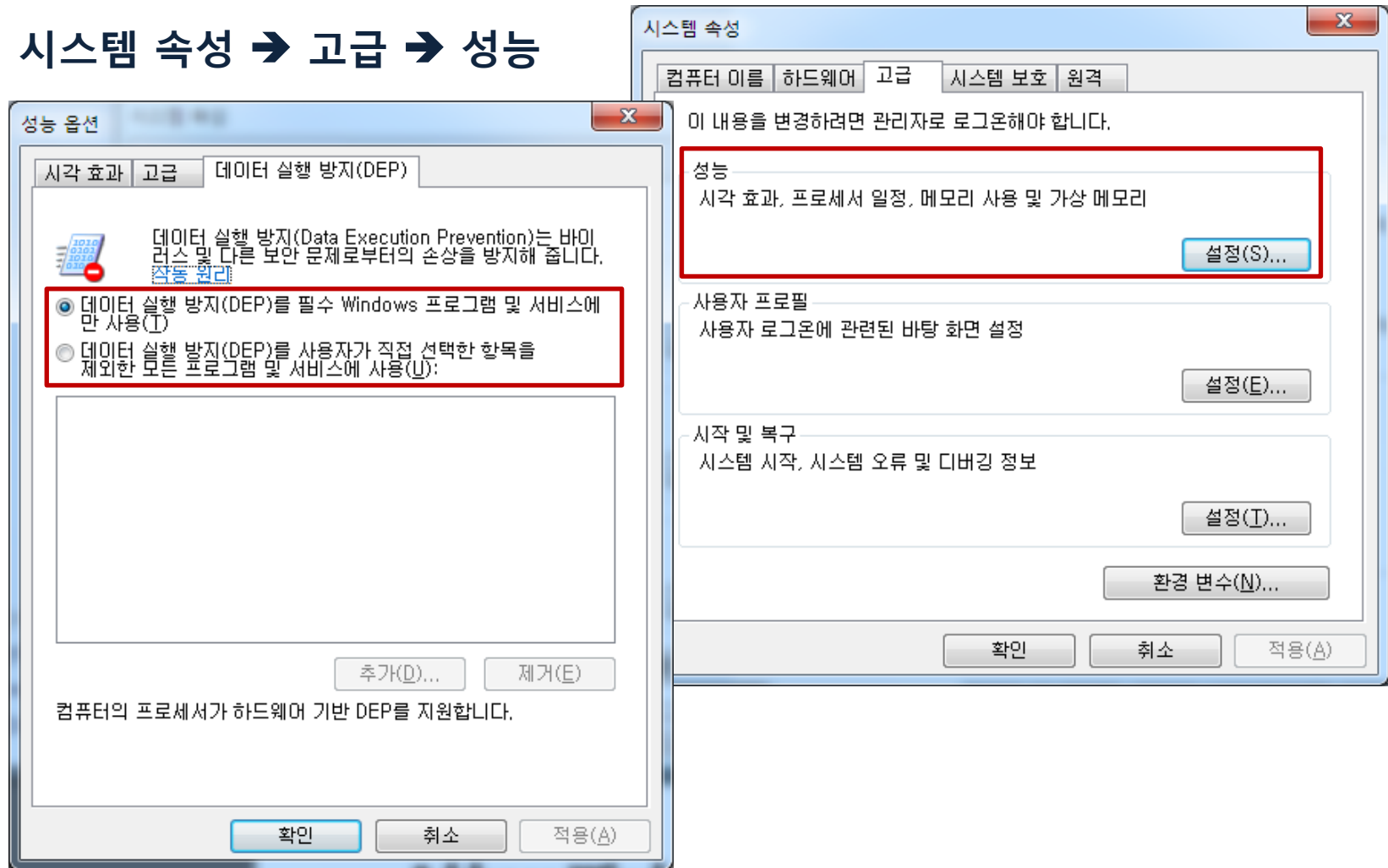
- %SystemDrive%\boot.ini 파일의 /noexecute 스위치 추가



```
boot.ini - 메모장
파일(F)  편집(E)  서식(O)  보기(V)  도움말(H)
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional"
/noexecute=always on /fastdetect
```

데이터 실행 방지 (DEP, Data Execution Prevention)

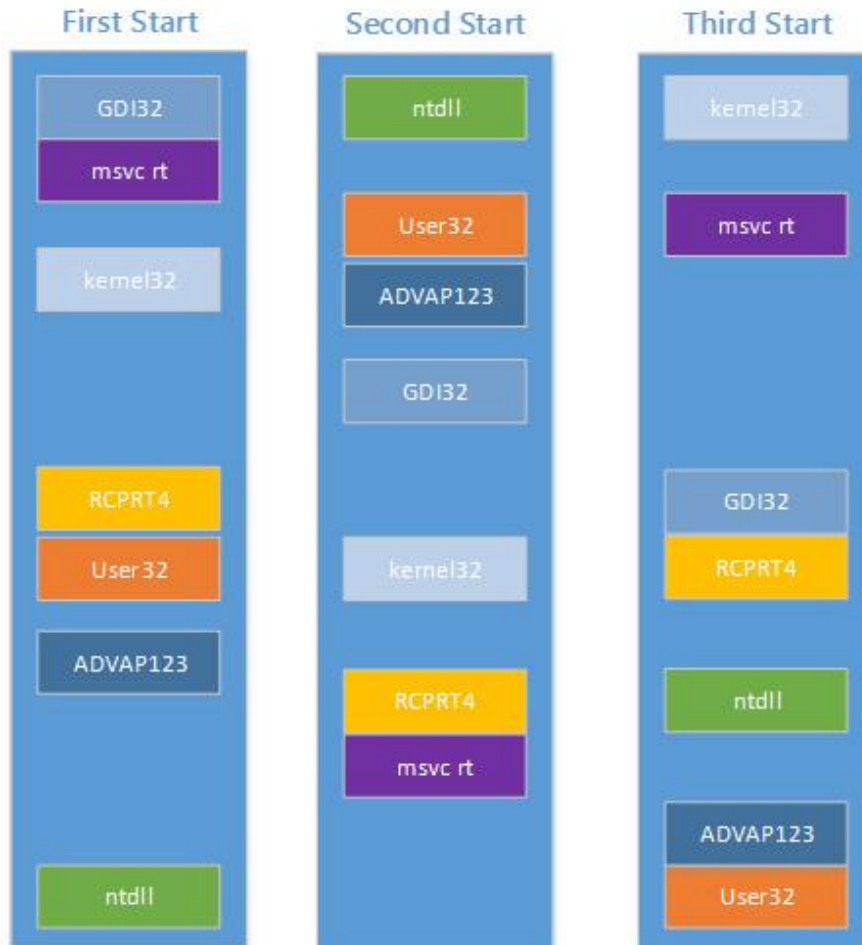
- 시스템 속성 → 고급 → 성능



메모리 이해

주소 공간 배치 랜덤화 (ASLR, Address Space Layout Randomization)

- 실행 오브젝트의 가상 주소 공간을 매 실행마다 랜덤하게...

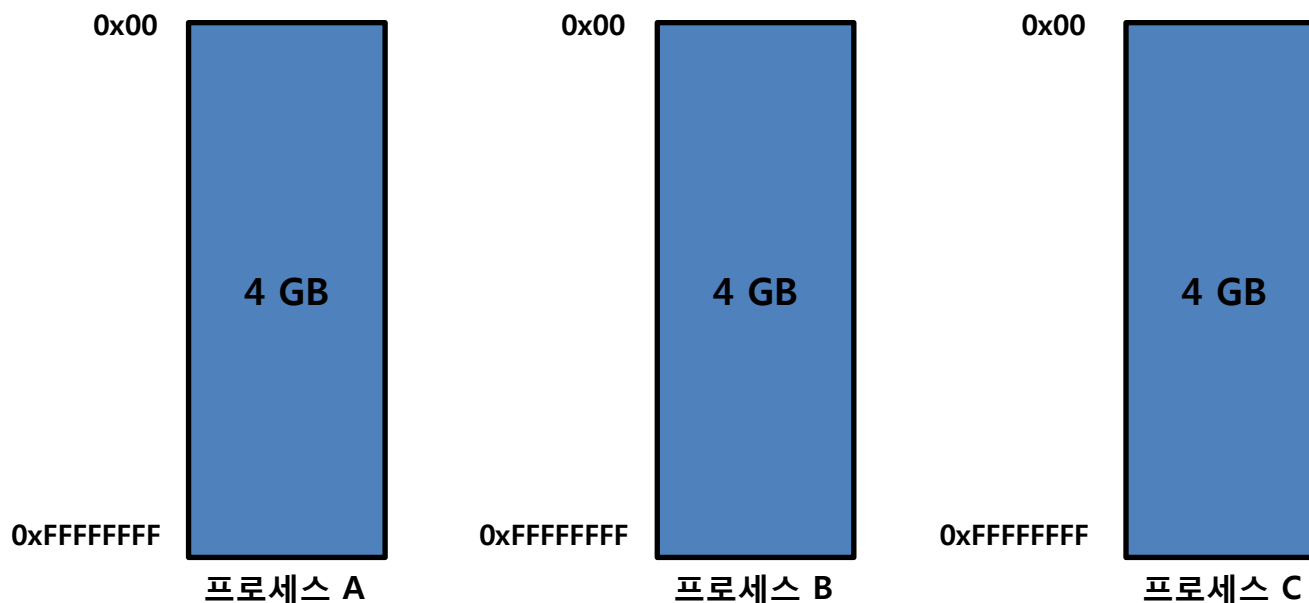


<http://technet.microsoft.com/en-us/library/dn283963.aspx>

메모리 이해

가상 메모리 소개

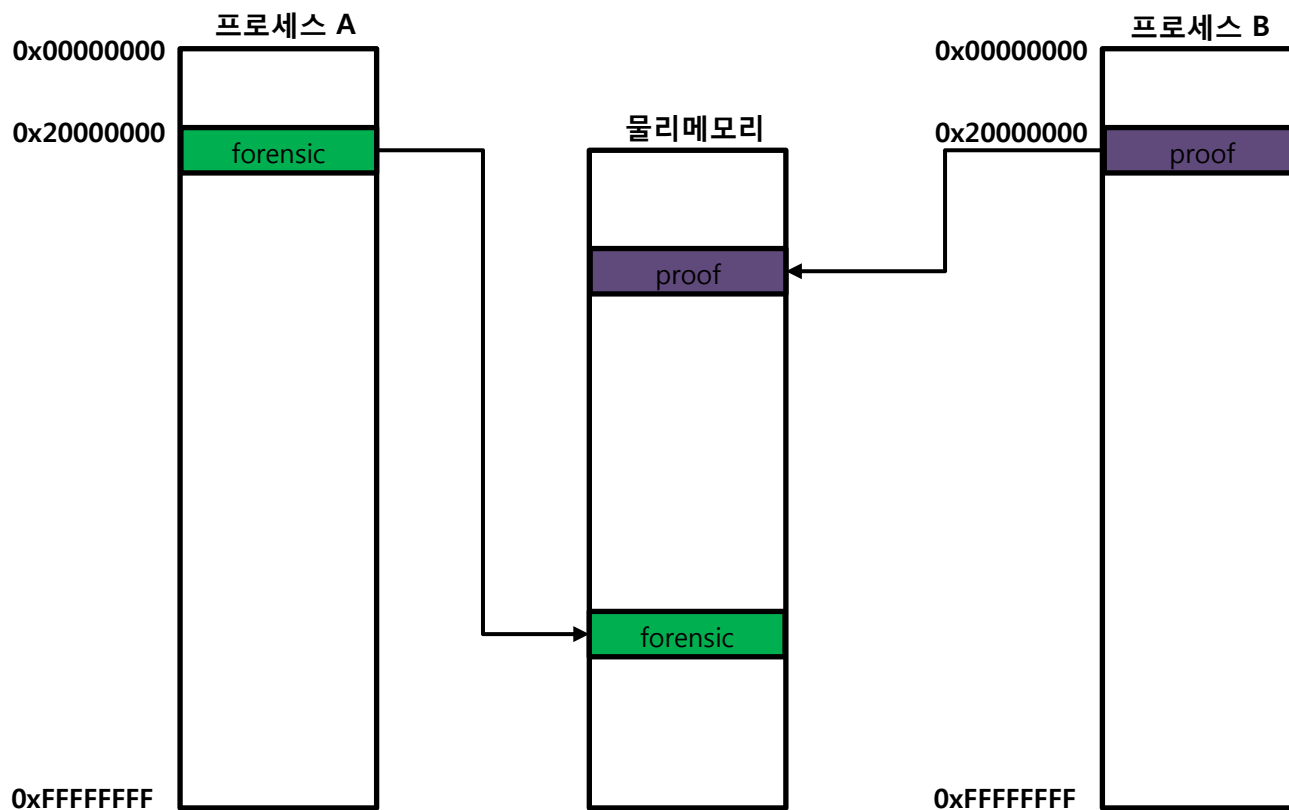
- Win32 시스템의 모든 프로세스는 4 GB의 연속된 주소 공간 사용
- 물리메모리 용량이 4GB인 시스템에서 다수의 프로세스가 동작하는 이유는?
- 가상 메모리 기법으로 대용량 메모리를 사용하는 프로그램 수행 가능



메모리 이해

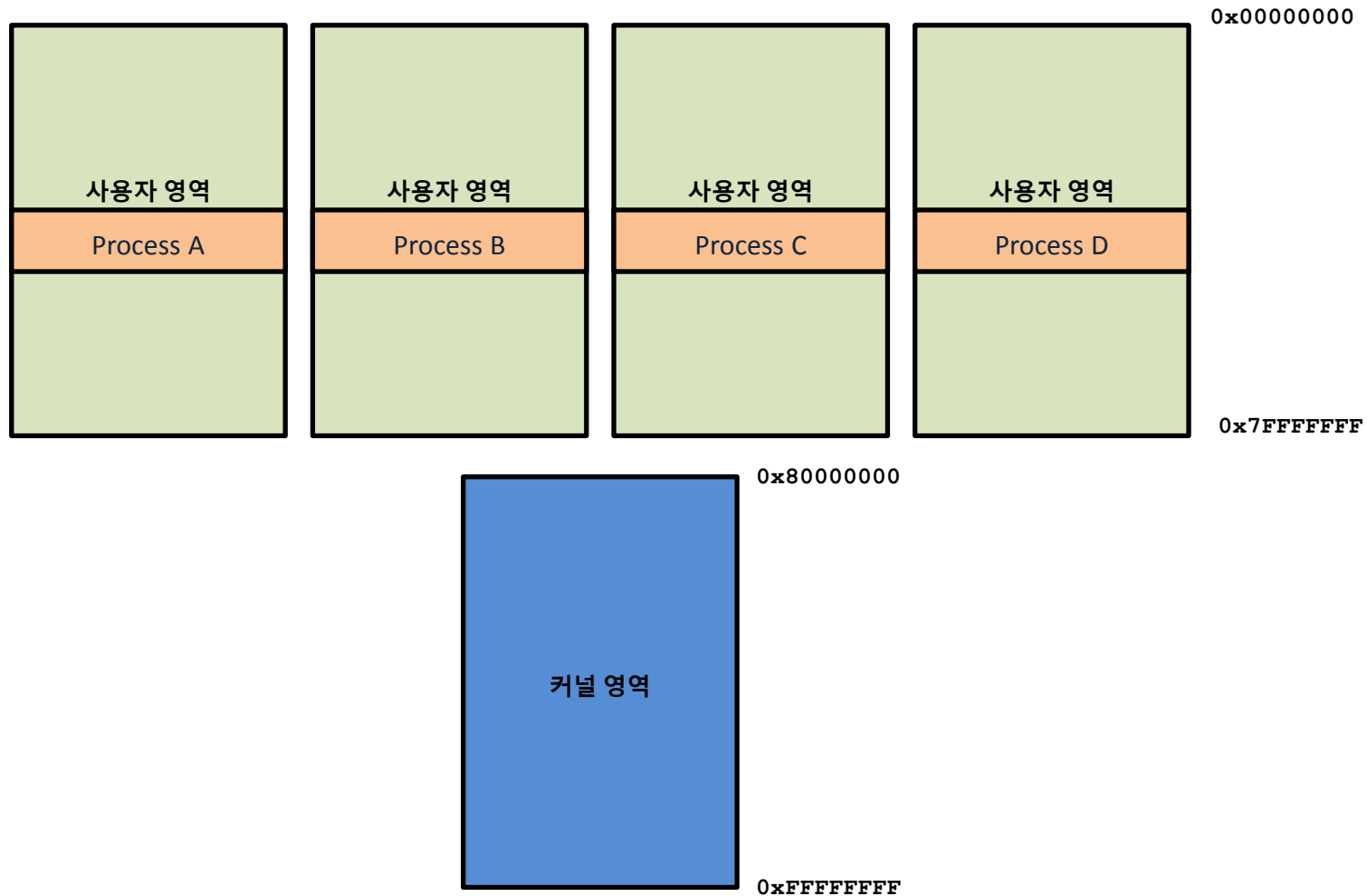
x86 주소 공간

- 모든 프로세스는 독자적인 사용자 영역 사용
- 프로세스에서 가상 주소가 같더라도 실제 물리 주소는 차이가 남



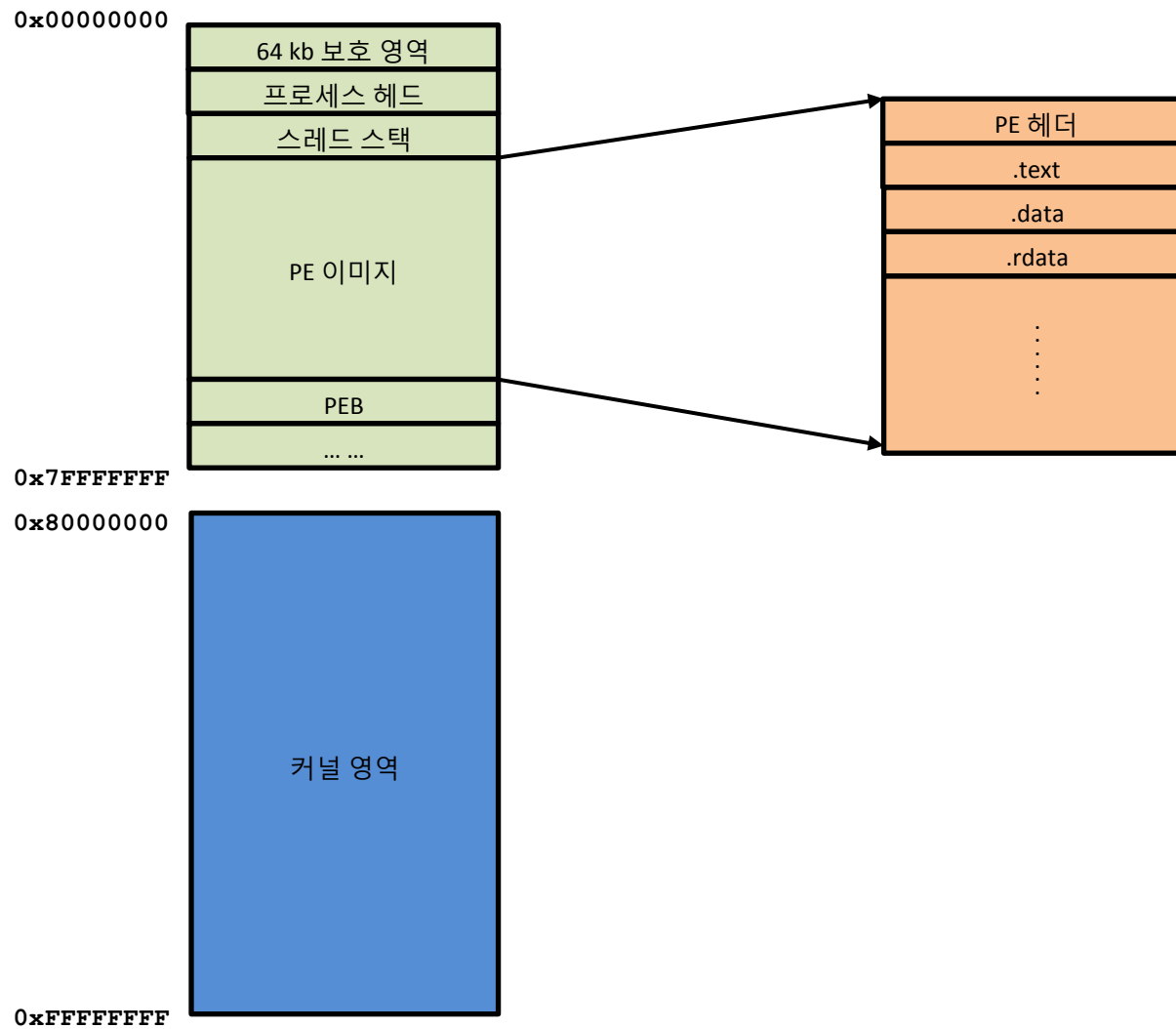
메모리 이해

x86 주소 공간



메모리 이해

x86 주소 공간



가상 메모리 주소 공간

- Win32 시스템의 모든 프로세스는 4 GB의 주소 공간 사용

- 사용자 영역 (0x00000000 – 0x7FFFFFFF)
- 커널 영역 (0x80000000 – 0xFFFFFFFF)

- 사용자 영역 확장

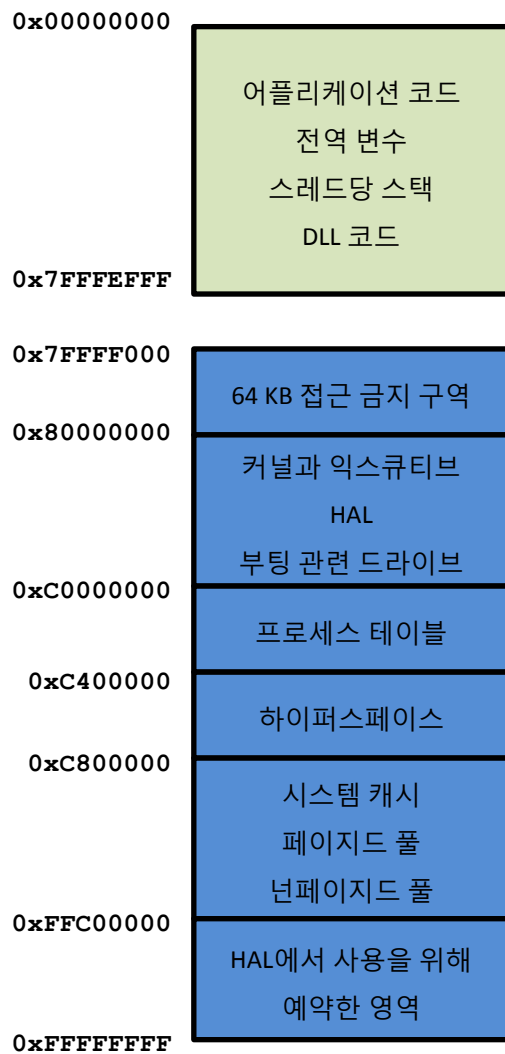
- 윈도우 Vista 이상
 - ✓ bcdedit /set increaseuserva 3072

```
C:\#>bcdedit /set increaseuserva 3072
작업을 완료했습니다.
```

- 윈도우 XP, 2003
 - ✓ boot.ini 파일에 /3GB 스위치 추가

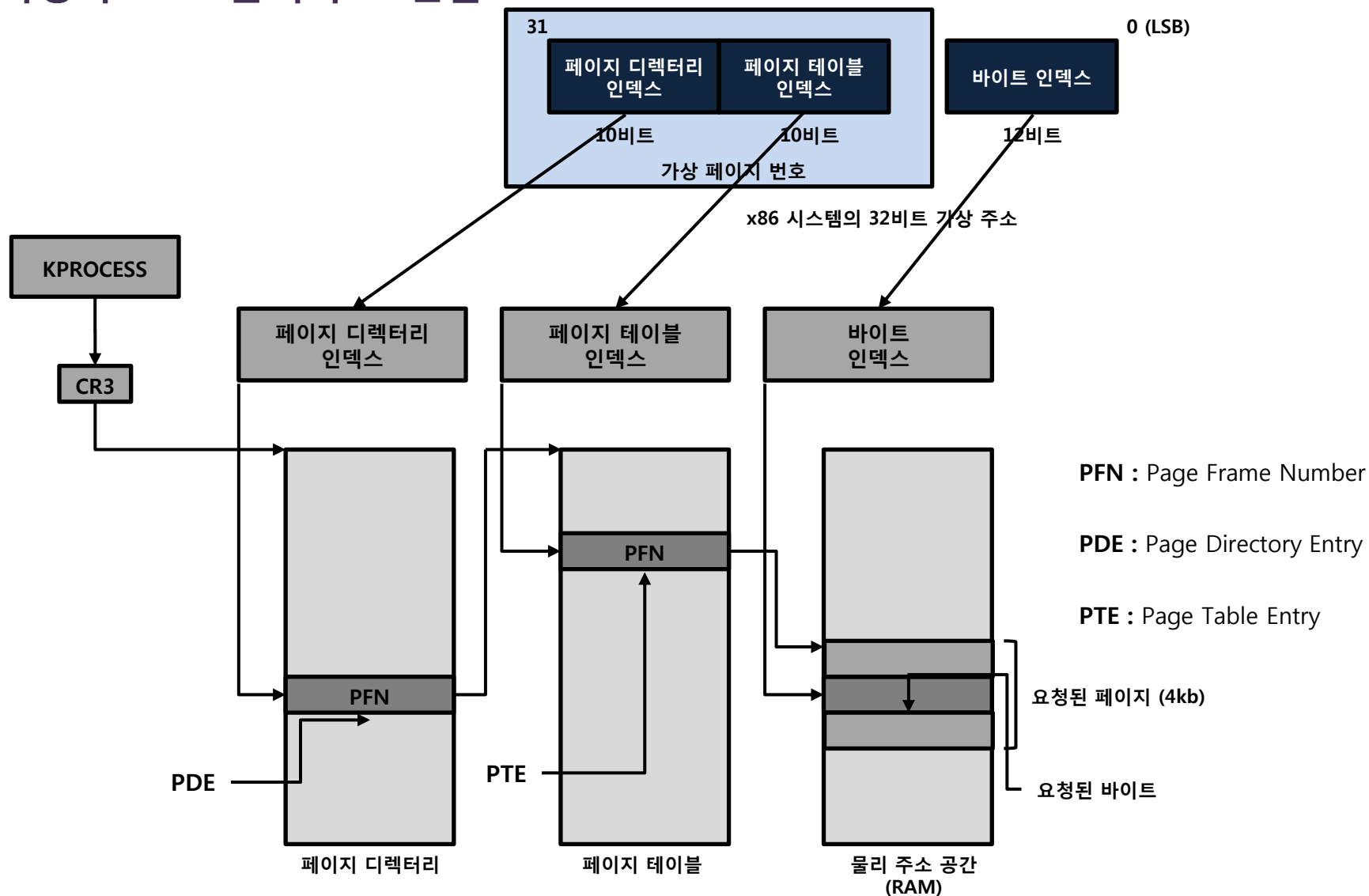
메모리 이해

x86 주소 공간



메모리 이해

가상 주소 → 물리 주소 변환



프로세스 구조

- EPROCESS(Executive Process, 실행 프로세스) : 프로세스 커널 오브젝트
- 윈도우에서 프로세스란 EPROCESS 구조체를 의미
- 주요 EPROCESS 항목

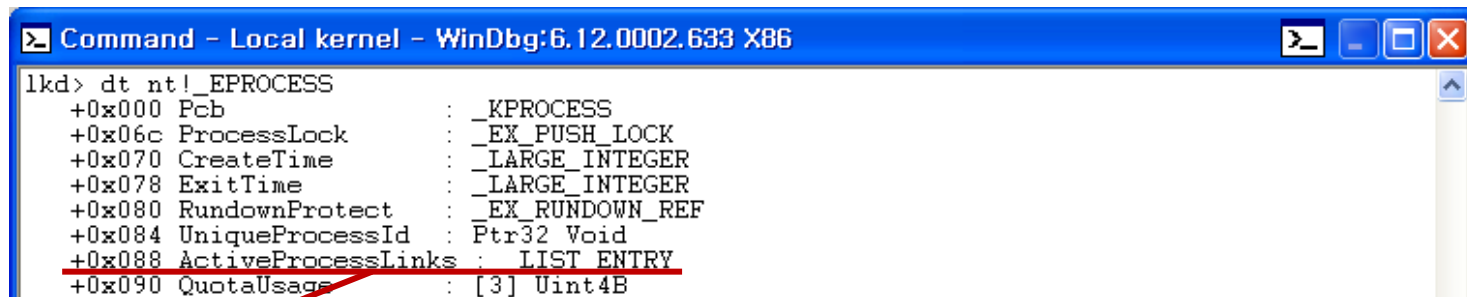
항목	데이터 타입	설명
PCB (Process Control Block)	_KPROCESS	DISPATCHER_HEADER, 디렉터리 테이블 주소, KTHREAD 목록, 우선 순위, 커널/유저 CPU 시간 등
CreateTime	_LARGE_INTEGER	64비트 윈도우 시간 형식의 프로세스 시작 시간
ExitTime	_LARGE_INTEGER	64비트 윈도우 시간 형식의 프로세스 종료 시간
UniqueProcessId	Ptr32 Void	프로세스 ID
ActiveProcessLinks	_LIST_ENTRY	ActiveProcess List를 구성하는 이중 링크드 리스트
ObjectTable	Ptr32_HANDLE_TABLE	오브젝트 핸들 테이블의 위치를 가르키는 포인터
WorkingSetPage	Uint4B	프로세스 WorkingSetPage
Peb (Process Environment Block)	Ptr32_PEB	프로세스 실행에 필요한 정보, 실행파일/DLL 경로, 베이스 어드레스, 모듈 리스트, 힙/스택 정보 등

메모리 이해

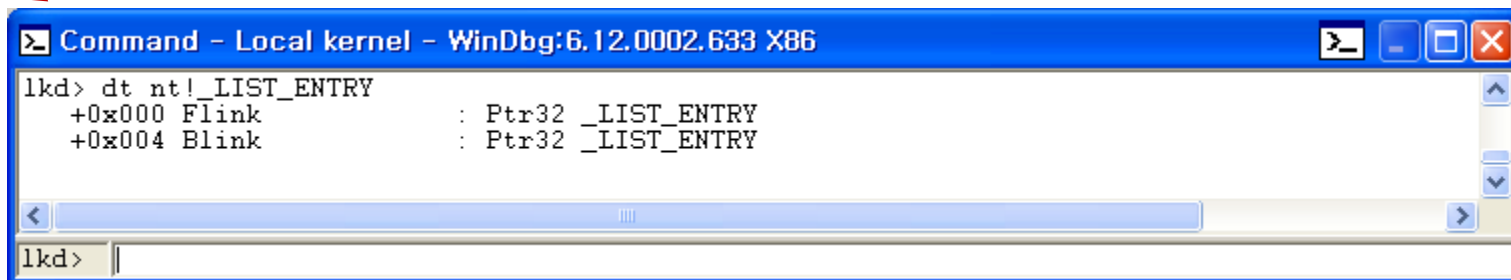
프로세스 구조

■ ActiveProcessLinks

- 환형 이중 링크드 리스트 형태로 구성
- 시스템은 ActiveProcessList를 이용해 프로세스 목록 획득



```
Command - Local kernel - WinDbg:6.12.0002.633 X86
lkd> dt nt!_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
```



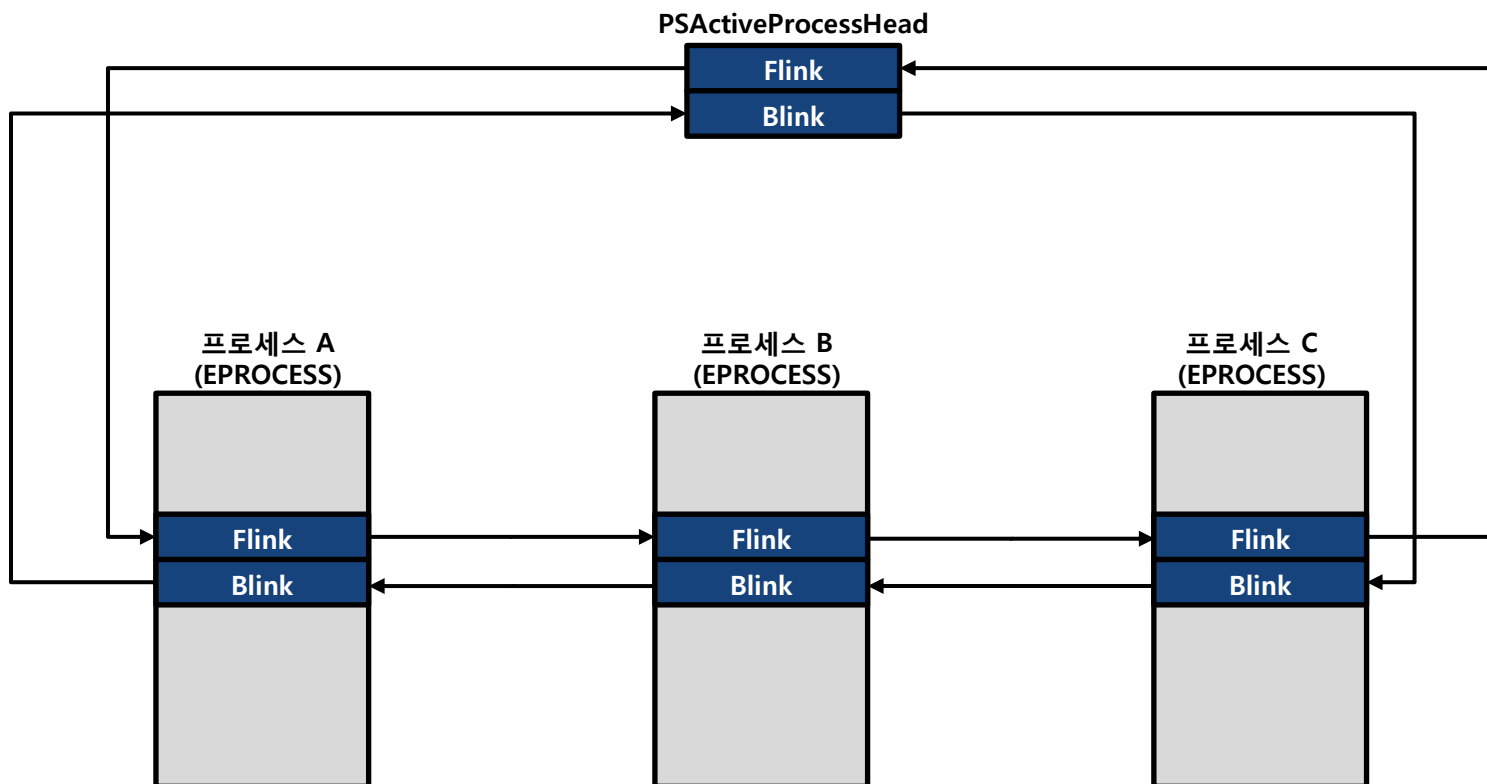
```
Command - Local kernel - WinDbg:6.12.0002.633 X86
lkd> dt nt!_LIST_ENTRY
+0x000 Flink : Ptr32 _LIST_ENTRY
+0x004 Blink : Ptr32 _LIST_ENTRY

lkd> |
```


프로세스 구조

■ ActiveProcessLinks

- PActiveProcessHead 커널 객체를 통해 전체 프로세스 접근

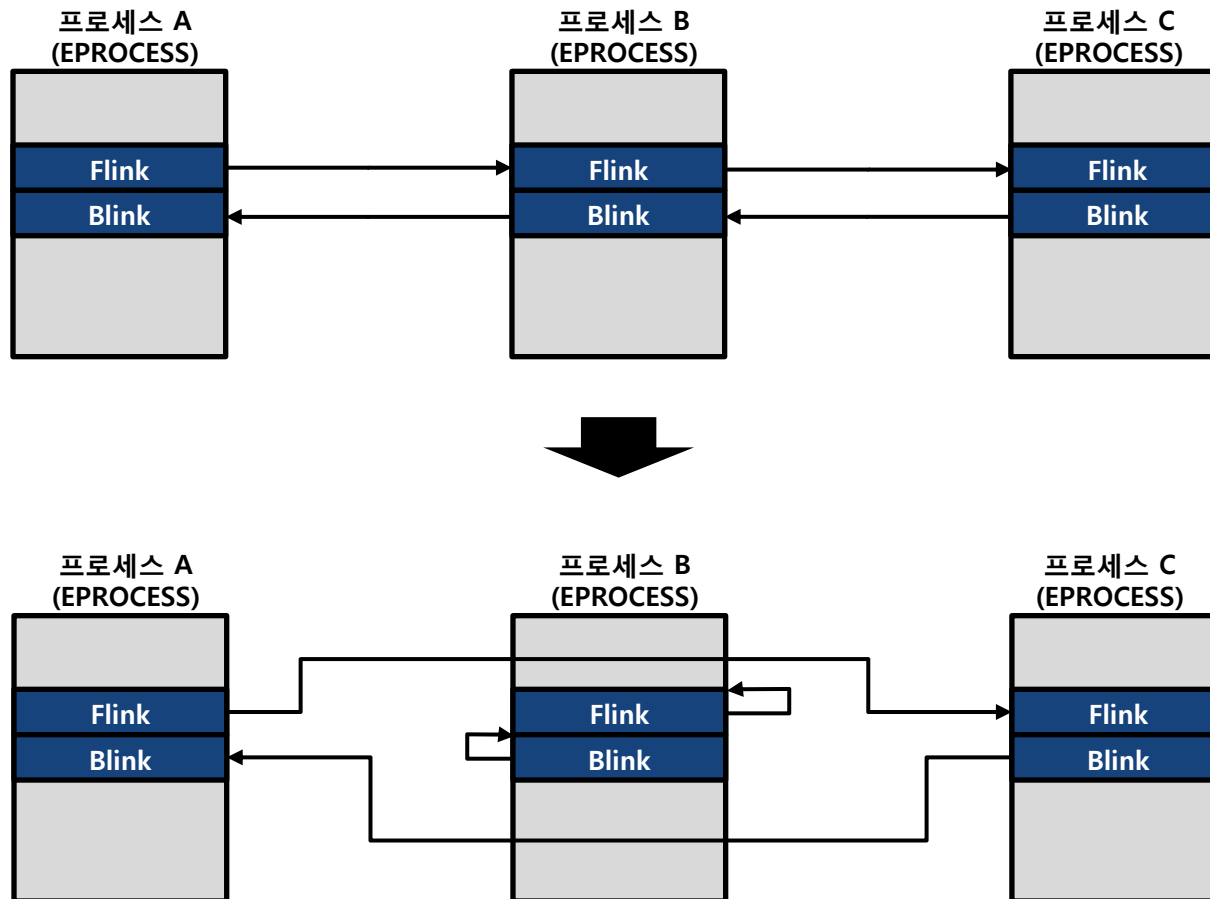


프로세스 은닉(루트킷) 기법

- DLL Injection
- IAT (Imported Address Table) Hooking
- IDT (Interrupt Descriptor Table) Hooking
- Inline Hooking
- SSDT (System Service Dispatch Table) Hooking
- **DKOM (Direct Kernel Object Manipulation)**

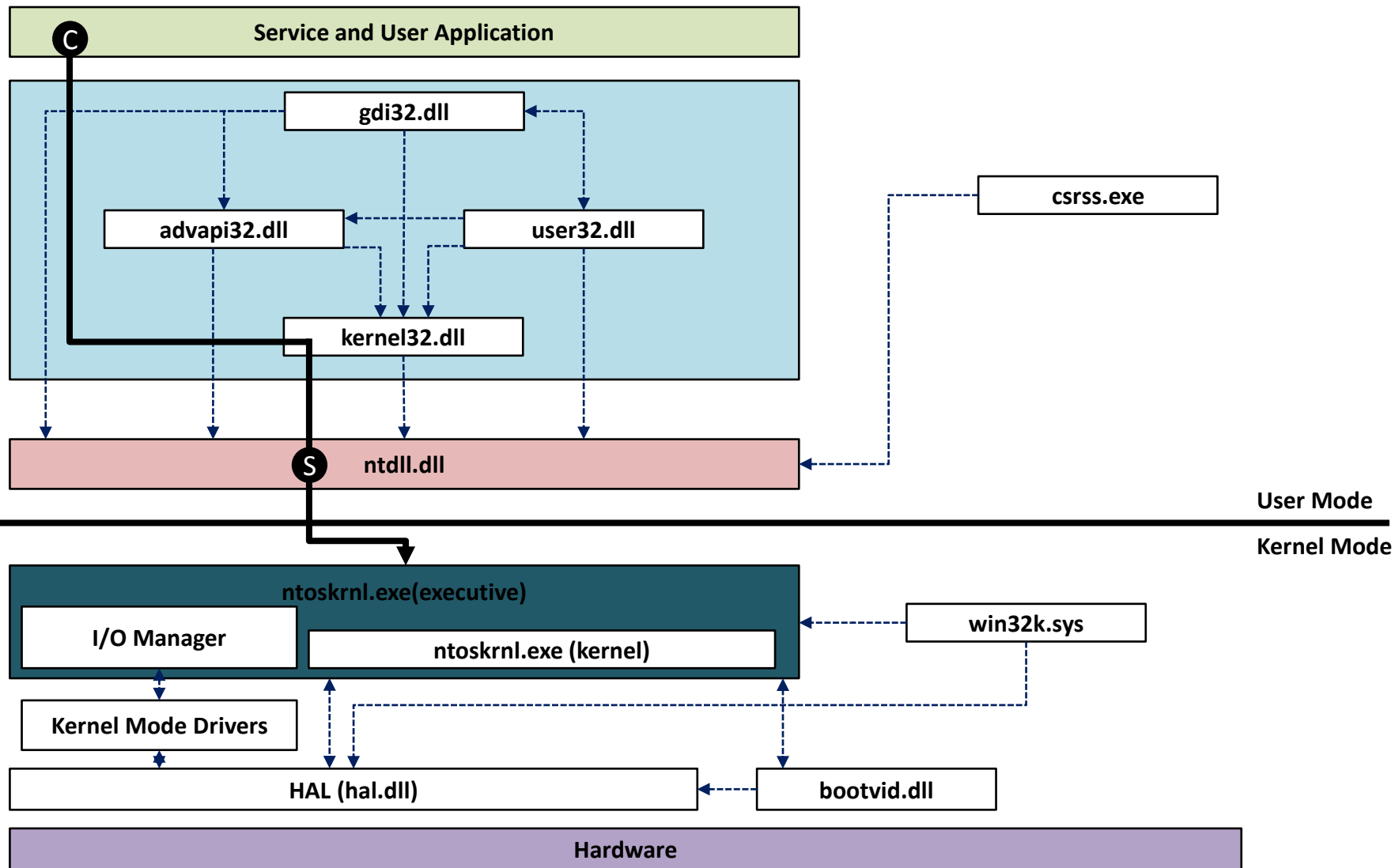
프로세스 은닉(루트킷) 기법

- DKOM (Direct Kernel Object Manipulation)



메모리 이해

사용자 모드 vs. 커널 모드



메모리 획득

메모리 획득

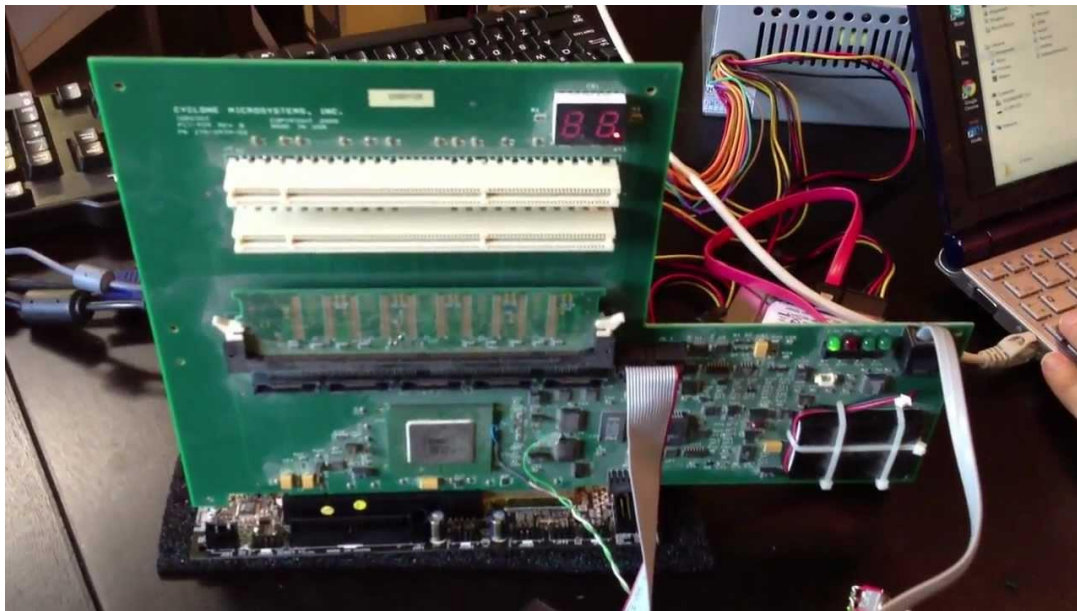
메모리 획득 방식

1. 하드웨어를 이용한 덤프
2. 소프트웨어를 이용한 덤프
3. 크래시 덤프
4. 절전모드 덤프
5. 콜드부트 덤프
6. 가상화 시스템 덤프

하드웨어를 이용한 덤프

■ PCI 장치를 이용한 덤프 – Tribble

- “A Hardware-Based Memory Acquisition Procedure for Digital Investigations”
- PCI 장치를 이용해 물리 메모리를 외부 저장장치로 덤프
- 추가적인 하드웨어/소프트웨어의 설치 없이 무결성을 최대한 보장
- 단, 사전에 미리 설치되어 있어야 함 ➔ 조직의 규정으로 마련되어야....



하드웨어를 이용한 덤프

■ FireWire(IEEE 1394)를 이용한 메모리 덤프 – FireWire Attack

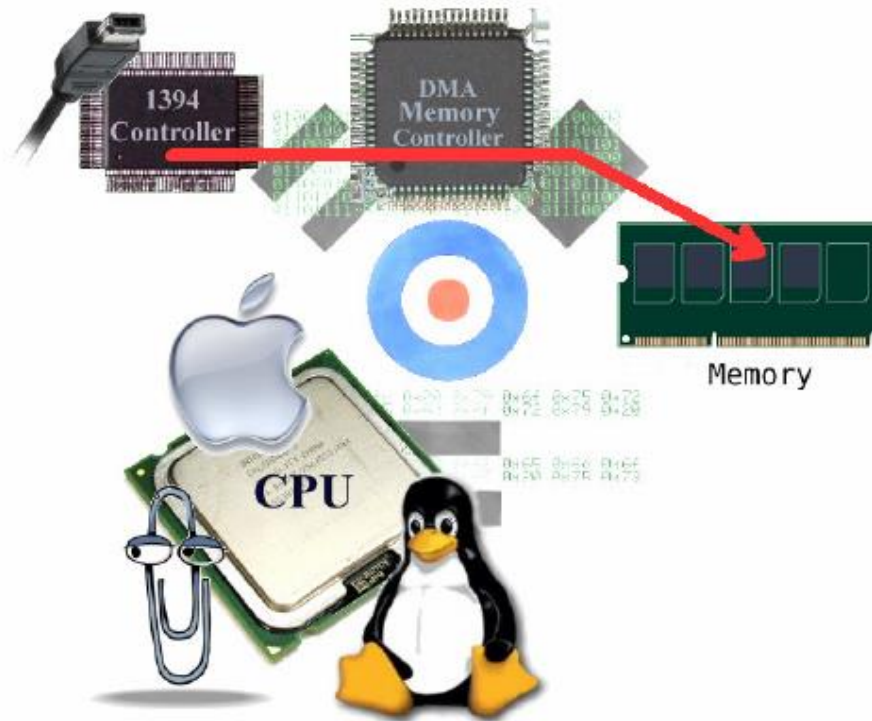
- FireWire 인터페이스를 이용해 DMA에 접근하여 메모리 덤프
- 윈도우, 리눅스, 맥 OS에서 가능

• 장점

- ✓ 악성 프로그램에 영향을 받지 않음
- ✓ 빠른 메모리 덤프
- ✓ 무결성 훼손 최소화

• 단점

- ✓ 간혹 시스템 크래시 발생
- ✓ 안정성 문제로 현재 연구로만 진행



소프트웨어를 이용한 덤프

▪ DD (Disk Dump)

- 유닉스용 도구 (윈도우용 버전도 존재)
- 접근 가능한 데이터를 바이트 혹은 블록 단위로 복사

• CDRom 이미지 생성

```
dd if=c:\\?\\Device\\CdRom0 of=c:\\temp\\disc1.iso bs=1M
```

• USB 이미지 생성

```
dd if=\\?\\Device\\Harddisk1\\Partition0 of=c:\\temp\\usb2.img bs=1M --size --progress
```

• 완전 삭제

```
dd if=/dev/random of=/dev/sda
```

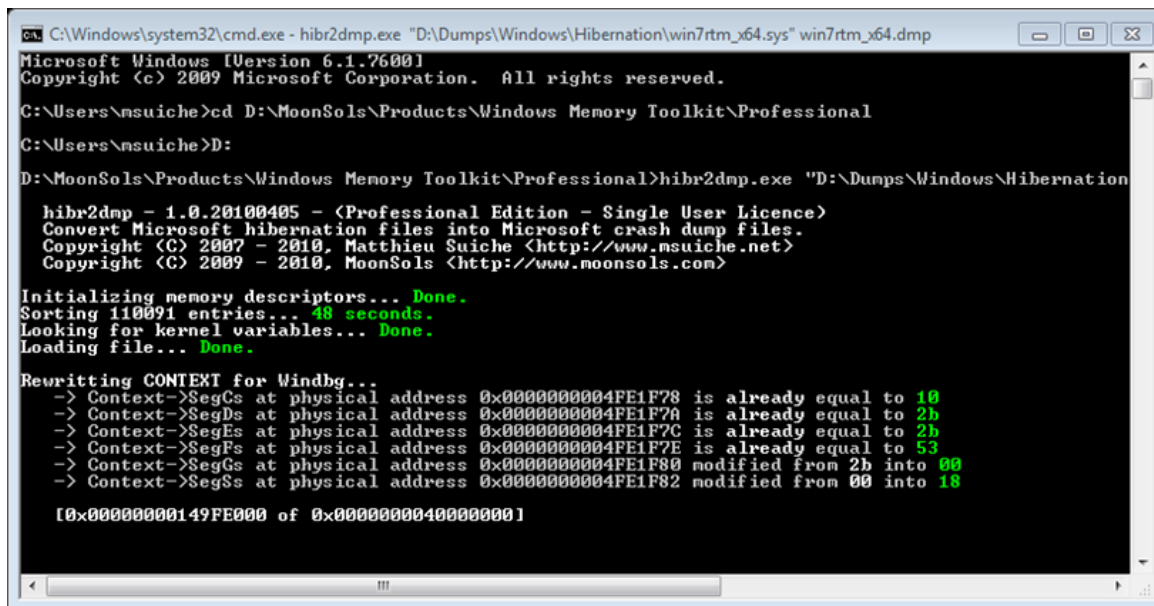
```
dd if=/dev/zero of=path/to/file bs=512 count=1 conv=notrunc
```

• 기타 DD 스타일 도구

- ✓ dcfldd, fdu dd, mdd 등

소프트웨어를 이용한 덤프

- **Win(32|64)dd/DumpIt** – <http://www.moonsols.com/>
 - Matthieu Suiche가 개발한 후 현재는 MoonSols Windows Memory Toolkit에 포함
 - 기능이 제한된 커뮤니티 버전은 무료
 - 프로페셔널 버전은 사용자당 500EUR (약 75만원)
 - 윈도우 XP부터 윈도우 7까지 덤프 지원 (윈도우 8은 아직....)



```
C:\Windows\system32\cmd.exe - hibr2dmp.exe "D:\Dumps\Windows\Hibernation\win7rtm_x64.sys" win7rtm_x64.dmp
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\msuiche>cd D:\MoonSols\Products\Windows Memory Toolkit\Professional
C:\Users\msuiche>D:
D:\MoonSols\Products\Windows Memory Toolkit\Professional>hibr2dmp.exe "D:\Dumps\Windows\Hibernation
hibr2dmp - 1.0.20100405 - <Professional Edition - Single User Licence>
Convert Microsoft hibernation files into Microsoft crash dump files.
Copyright (C) 2007 - 2010, Matthieu Suiche <http://www.msuiche.net>
Copyright (C) 2009 - 2010, MoonSols <http://www.moonsols.com>

Initializing memory descriptors... Done.
Sorting 110091 entries... 48 seconds.
Looking for kernel variables... Done.
Loading file... Done.

Rewriting CONTEXT for Windbg...
-> Context->SegCs at physical address 0x000000004FE1F78 is already equal to 10
-> Context->SegDs at physical address 0x000000004FE1F7A is already equal to 2b
-> Context->SegEs at physical address 0x000000004FE1F7C is already equal to 2b
-> Context->SegFs at physical address 0x000000004FE1F7E is already equal to 53
-> Context->SegGs at physical address 0x000000004FE1F80 modified from 2b into 00
-> Context->SegSs at physical address 0x000000004FE1F82 modified from 00 into 18

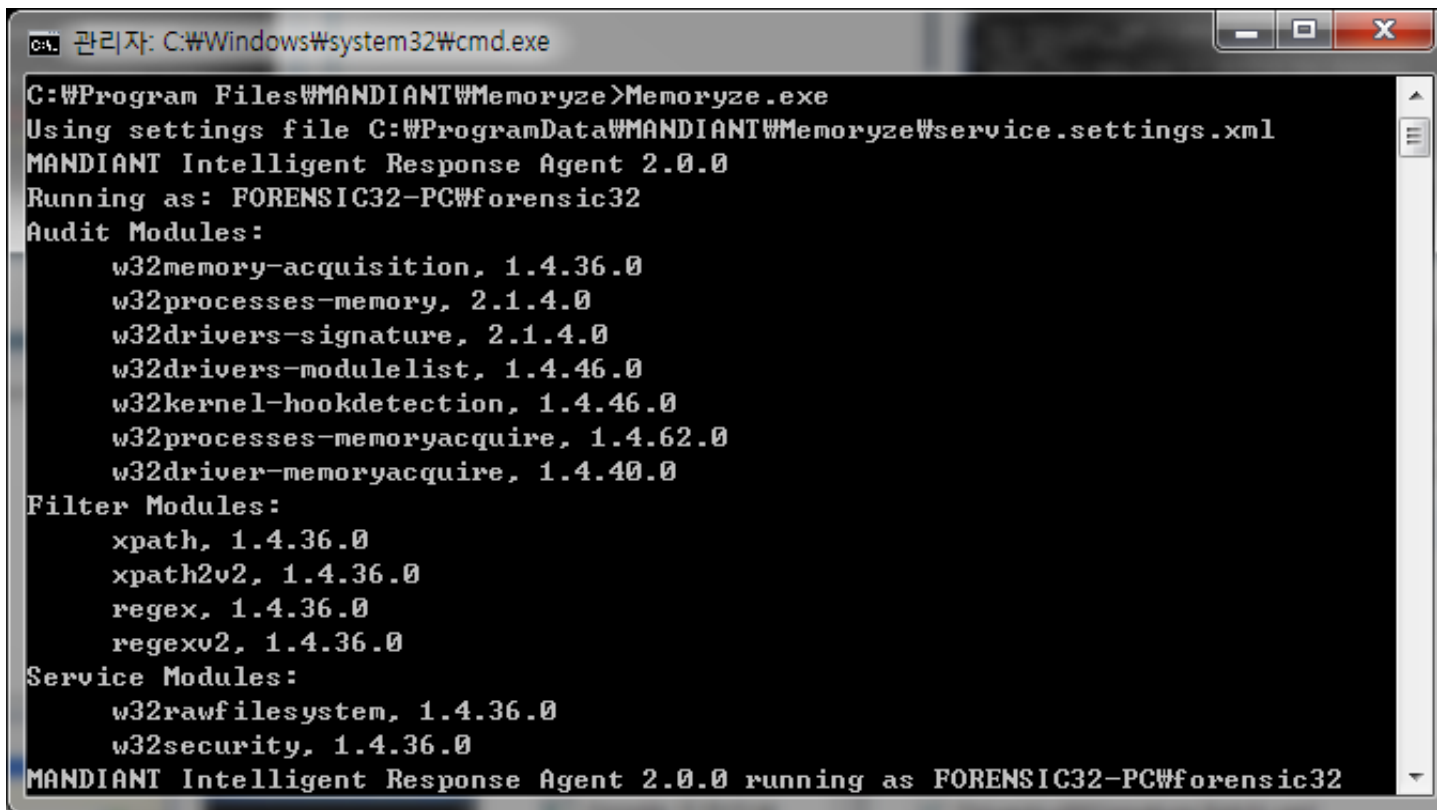
[0x00000000149FE000 of 0x0000000040000000]
```

메모리 획득

소프트웨어를 이용한 덤프

■ Memorize™

- 맨디언트에서 개발한 무료 메모리 이미징/분석 도구
- 윈도우 2000 SP4부터 윈도우 8(2012)까지 지원

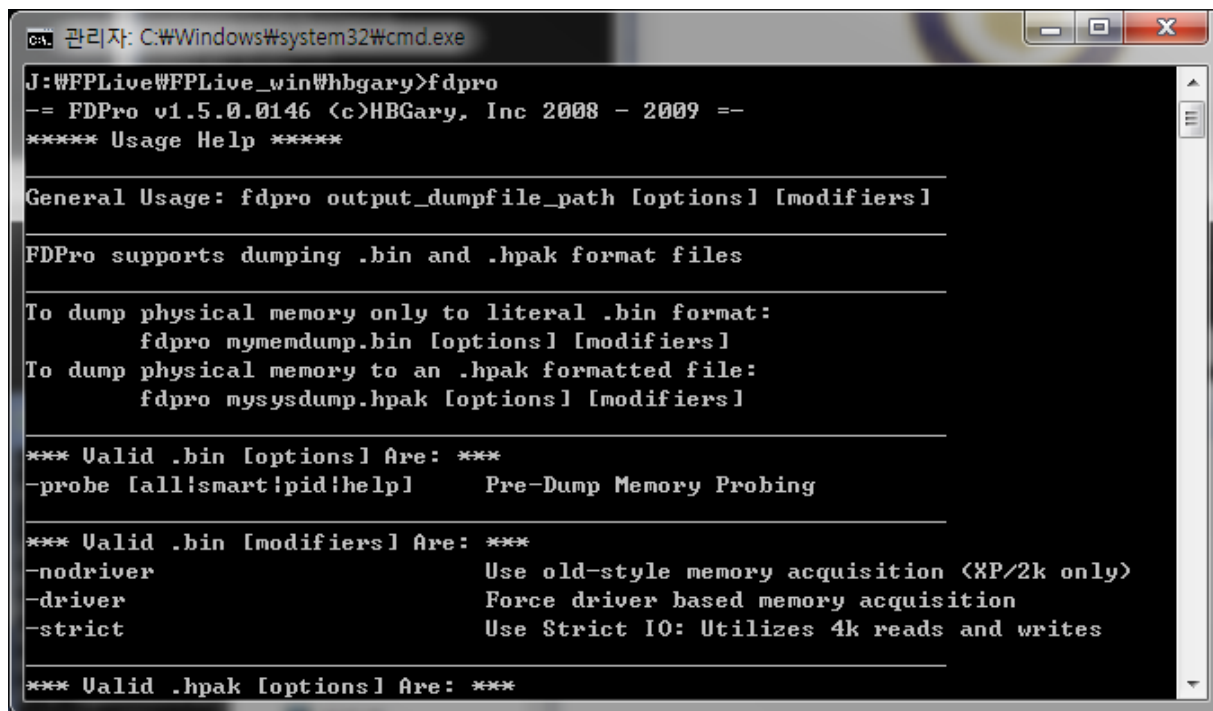


```
C:\Program Files\MANDIANT\Memoryze>Memoryze.exe
Using settings file C:\ProgramData\MANDIANT\Memoryze\service.settings.xml
MANDIANT Intelligent Response Agent 2.0.0
Running as: FORENSIC32-PC\forensic32
Audit Modules:
    w32memory-acquisition, 1.4.36.0
    w32processes-memory, 2.1.4.0
    w32drivers-signature, 2.1.4.0
    w32drivers-modulelist, 1.4.46.0
    w32kernel-hookdetection, 1.4.46.0
    w32processes-memoryacquire, 1.4.62.0
    w32driver-memoryacquire, 1.4.40.0
Filter Modules:
    xpath, 1.4.36.0
    xpath2v2, 1.4.36.0
    regex, 1.4.36.0
    regexv2, 1.4.36.0
Service Modules:
    w32rawfilesystem, 1.4.36.0
    w32security, 1.4.36.0
MANDIANT Intelligent Response Agent 2.0.0 running as FORENSIC32-PC\forensic32
```

소프트웨어를 이용한 덤프

■ FD(FastDump)Pro

- HBGary에서 개발한 상용 메모리 덤프 도구로 Responder 제품의 번들
- 무료 버전은 64비트와 일부 운영체제 미지원
- 매우 빠른 이미징 속도와 대용량 메모리도 가능



```
관리자: C:\Windows\system32\cmd.exe
J:\WPFLive\WPFLive_win\hbgary>fdpro
== FDPro v1.5.0.0146 (c)HBGary, Inc 2008 - 2009 ==
***** Usage Help *****

General Usage: fdpro output_dumpfile_path [options] [modifiers]

FDPro supports dumping .bin and .hpak format files

To dump physical memory only to literal .bin format:
    fdpro mymemdump.bin [options] [modifiers]
To dump physical memory to an .hpak formatted file:
    fdpro mysysdump.hpak [options] [modifiers]

*** Valid .bin [options] Are: ***
-probe [all|smart|pid|help]      Pre-Dump Memory Probing













*** Valid .bin [modifiers] Are: ***
-nodriver                        Use old-style memory acquisition (XP/2k only)
-driver                          Force driver based memory acquisition
-strict                          Use Strict IO: Utilizes 4k reads and writes

*** Valid .hpak [options] Are: ***
```

소프트웨어를 이용한 덤프

■ Winen

- 엔케이스(EnCase) 상용 버전에 포함된 메모리 덤프 도구
- RAW 형식이 아닌 .E01 형식으로 덤프
- FTK Imager, EnCase를 이용해 다른 형식으로 변환 가능
- 배치 파일을 이용해 실행 가능

	linen	2014-01-13 오후...	파일	2,638KB
	linen.1.gz	2014-01-13 오후...	GZ 파일	2KB
	linen.txt	2014-01-13 오후...	텍스트 문서	3KB
	register.htm	2013-11-17 오후...	HTM 파일	4KB
	ReleaseNotes.chm	2014-01-13 오후...	컴파일된 HTML ...	7,806KB
	ServletCab.cab	2014-01-13 오후...	캐비닛 파일	11,983KB
	winacq.exe	2014-01-13 오후...	응용 프로그램	406KB
	winacq.txt	2014-01-13 오후...	텍스트 문서	3KB
	winen.exe	2014-01-13 오후...	응용 프로그램	409KB
	winen.txt	2014-01-13 오후...	텍스트 문서	2KB
	winen64.exe	2014-01-13 오후...	응용 프로그램	1,420KB
	winen64_sys	2014-04-21 오전...	시스템 파일	73KB

크래시 덤프

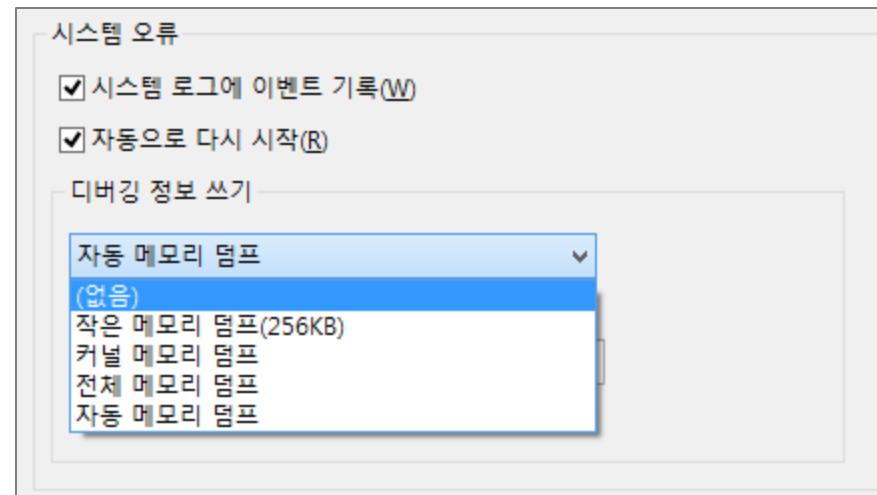
▪ 시스템 크래시 특징

- BSOD(Blue Screen of Death) 발생 시 자동 생성
- WinDbg, Kernel Memory Space Analyzer등을 통해 디버깅 가능
- 물리메모리에 가장 최소한의 영향을 미치는 방법
- 윈도우만 지원하며 수동 크래시를 발생시키려면 시스템 재부팅 필요

크래시 덤프

■ 시스템 크래시 설정

- [시스템] → [고급] → [시작 및 복구] → [설정]
- 크래시 덤프 종류
 - ✓ 커널 메모리 덤프 : 윈도우 2003/7 기본 설정
 - ✓ 작은 메모리 덤프 : 윈도우 2000 Pro/XP/Vista 기본 설정
 - ✓ 전체 메모리 덤프
- 윈도우 7 – 전체 메모리 덤프가 사라짐
- 윈도우 8 – 자동 메모리 덤프 추가



크래시 덤프

■ 레지스트리 설정

• HKLM\SYSTEM\CurrentControlSet\Control\CrashControl\CrashDumpEnabled

- ✓ 0 : 없음
- ✓ 1 : 전체 메모리 덤프
- ✓ 2 : 커널 메모리 덤프
- ✓ 3 : 작은 메모리 덤프
- ✓ 7 : 자동 메모리 덤프 (윈도우 8만)

• 크래시 덤프 수동 생성

- ✓ PS/2 Keyboard : HKLM\System\CurrentControlSet\Services\i8042prt\Parameter
- ✓ USB Keyboard : HKLM\System\CurrentControlSet\Services\kbdhid\Parameter
- ✓ CrashOnCtrlScroll을 새로 만들고 DWORD 값을 0x01로 설정
- ✓ CTRL(오른쪽) + SCROLL + SCROLL 키 조합으로 강제 크래시 덤프

절전모드 덤프

■ 절전모드 특징

- 절전 모드(Hibernation)로 진입 시 메모리를 압축하여 C:\hiberfil.sys 파일로 저장
- 부팅 과정에서 NTLDR에 의해 메모리로 로드 된 후 이전 상태로 복귀
- 노트북과 같은 휴대용 시스템에서는 절전모드가 기본적으로 설정
- 조사 시 강제로 절전모드 설정 후 실행
- 추가적인 프로그램이나 장비 불필요 → 비교적 순수한 메모리 덤프 가능
- 전체 메모리 영역의 덤프가 아닌 사용 중인 영역만 덤프

메모리 획득

콜드 부트

- 종료된 시스템 메모리를 차갑게 유지시켜 메모리 손실을 줄이는 방법



http://www.youtube.com/results?search_query=memory+cold+boot

메모리 획득

콜드 부트

- 콜드 부트로 스마트폰 패스워드 획득



가상화 시스템 덤프

- 대부분의 가상화 시스템은 활성 메모리를 별도의 파일로 관리
- 최근 흔적을 은폐하고자 가상시스템을 이용한 사건이 다수 발생
- [일시 중지] 상태에서 메모리 복사

이름	수정한 날짜	유형	크기
Windows 7 (eng).nvram	2013-11-19 오전...	VMware virtual m...	9KB
Windows 7 (eng).vmdk	2012-09-02 오후...	VMware virtual di...	6,114,560KB
Windows 7 (eng).vmsd	2013-11-19 오전...	VMware snapshot...	1KB
Windows 7 (eng).vmx	2013-11-19 오전...	VMware virtual m...	3KB
Windows 7 (eng).vmxf	2012-01-04 오후...	VMwre.VMTeam...	2KB
Windows 7 (eng)-000001.vmdk	2013-04-26 오후...	VMware virtual di...	290,816KB
Windows 7 (eng)-000002.vmdk	2013-11-19 오전...	VMware virtual di...	510,976KB
Windows 7 (eng)-05998c81.vmem	2013-11-18 오전...	VMEM 파일	1,048,576KB
Windows 7 (eng)-05998c81.vms	2013-11-19 오전...	VMware suspend...	164,116KB
Windows 7 (eng)-Snapshot1.vmem	2012-09-02 오후...	VMEM 파일	1,048,576KB
Windows 7 (eng)-Snapshot1.vmsn	2012-09-02 오후...	VMware virtual m...	169,268KB
Windows 7 (eng)-Snapshot2.vmem	2013-04-26 오후...	VMEM 파일	1,048,576KB
Windows 7 (eng)-Snapshot2.vmsn	2013-04-26 오후...	VMware virtual m...	167,102KB

실전 메모리 덤프

▪ 최근 메모리 덤프 방식

- 하드웨어 방식은 미리 설치되어 있어야 하거나 안정성 문제
- 절전 모드, 크래시 덤프는 제한된 환경에서만 동작
- 따라서, 주로 소프트웨어 방식을 이용해 메모리 덤프

▪ 고려 사항

- 덤프한 메모리는 외장저장장치에 저장
- 외장저장장치 인터페이스에 따라 수집 속도 차이
- 침해사고의 경우 현장 상황에 따라 D:W 볼륨을 이용하는 것도 고려!!!

→ 실습

- Memorize를 이용해 시스템 메모리 덤프하기

```
<?xml version="1.0" encoding="utf-8"?>
<script xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" chaining="implicit">
  <commands>
    <command xsi:type="ExecuteModuleCommand">
      <module name="w32memory-acquisition" version="1.3.22.2" />
      <config xsi:type="ParameterListModuleConfig">
        <parameters>
          </parameters>
        </config>
      </command>
    </commands>
  </script>
```

```
$> memoryze.exe -o [output path] -script [xml path] -encoding none -allowmultiple
```

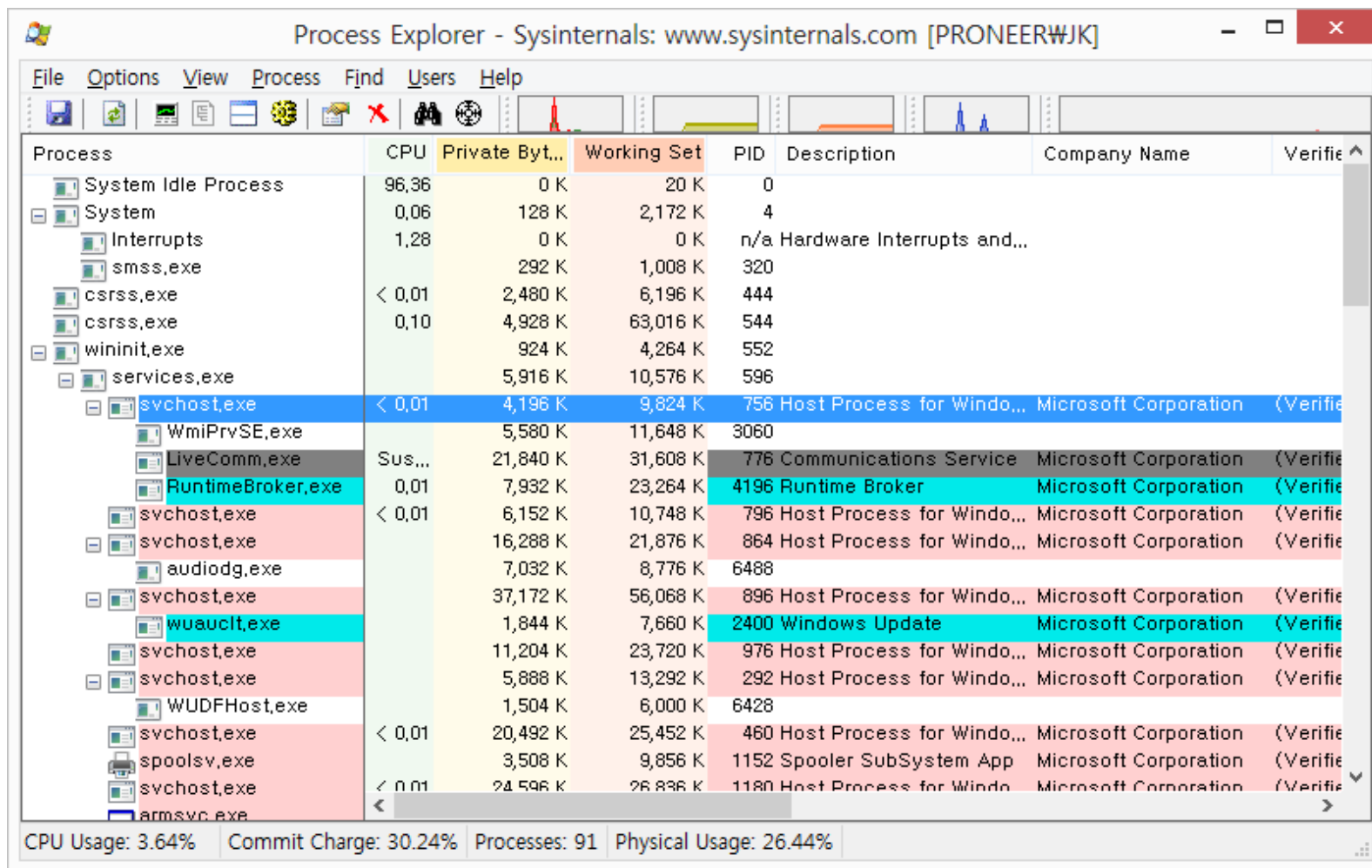
```
$> MemoryDD.bat -output [output path]
```

메모리 분석

메모리 분석

프로세스 구조 파악

- Process Explorer BY Sysinternals



Process Explorer - Sysinternals: www.sysinternals.com [PRONEERWJK]

File Options View Process Find Users Help

Process	CPU	Private Byt...	Working Set	PID	Description	Company Name	Verifie ^
System Idle Process	96,36	0 K	20 K	0			
System	0,06	128 K	2,172 K	4			
Interrupts	1,28	0 K	0 K	n/a	Hardware Interrupts and...		
smss.exe		292 K	1,008 K	320			
csrss.exe	< 0,01	2,480 K	6,196 K	444			
csrss.exe	0,10	4,928 K	63,016 K	544			
wininit.exe		924 K	4,264 K	552			
services.exe		5,916 K	10,576 K	596			
svchost.exe	< 0,01	4,196 K	9,824 K	756	Host Process for Windo...	Microsoft Corporation	(Verifie
WmiPrivSE.exe		5,580 K	11,648 K	3060			
LiveComm.exe	Sus...	21,840 K	31,608 K	776	Communications Service	Microsoft Corporation	(Verifie
RuntimeBroker.exe	0,01	7,932 K	23,264 K	4196	Runtime Broker	Microsoft Corporation	(Verifie
svchost.exe	< 0,01	6,152 K	10,748 K	796	Host Process for Windo...	Microsoft Corporation	(Verifie
svchost.exe		16,288 K	21,876 K	864	Host Process for Windo...	Microsoft Corporation	(Verifie
audiodg.exe		7,032 K	8,776 K	6488			
svchost.exe		37,172 K	56,068 K	896	Host Process for Windo...	Microsoft Corporation	(Verifie
wuauclt.exe		1,844 K	7,660 K	2400	Windows Update	Microsoft Corporation	(Verifie
svchost.exe		11,204 K	23,720 K	976	Host Process for Windo...	Microsoft Corporation	(Verifie
svchost.exe		5,888 K	13,292 K	292	Host Process for Windo...	Microsoft Corporation	(Verifie
WUDFHost.exe		1,504 K	6,000 K	6428			
svchost.exe	< 0,01	20,492 K	25,452 K	460	Host Process for Windo...	Microsoft Corporation	(Verifie
spoolsv.exe		3,508 K	9,856 K	1152	Spooler SubSystem App	Microsoft Corporation	(Verifie
svchost.exe	< 0,01	24,596 K	26,836 K	1180	Host Process for Windo...	Microsoft Corporation	(Verifie
armsvc.exe							

CPU Usage: 3.64% Commit Charge: 30.24% Processes: 91 Physical Usage: 26.44%

메모리 분석

초기 메모리 분석 방법

- **문자열 추출**

- 특정 패턴의 문자열 검색
- 이메일, 계정, 비밀번호, 메신저 대화 등

- **파일 카빙**

- 그래픽 이미지, HTML, 레지스트리 하이브 구조, 이벤트 로그 레코드 등

오브젝트 검색

- 물리메모리 상의 오브젝트를 찾기 위한 방법
 - 리스트 워킹 (List-Walking)
 - 패턴 매칭 (Pattern Matching)

오브젝트 검색

▪ 리스트 워킹

• EPROCESS 프로세스 이름을 이용한 프로세스 탐색 기법

- ✓ EPROCESS 구조체 내부의 프로세스 이름을 검색하여 주요 프로세스 확인 (system, smss 등)
- ✓ 이름이 확인되면 해당 위치를 기준으로 앞 뒤에 EPROCESS 구조가 존재하는지 검증
- ✓ EPROCESS 구조가 존재한다면 ActiveProcessLinks를 이용해 프로세스 탐색

• KPCR(Kernel Processor Control Region)을 이용한 프로세스 탐색 기법

- ✓ KPCR은 XP, 2003, Vista에서 가상 주소 0xFFDFF000 위치, 확장 버전인 KPCRB는 0xFFDFF120
- ✓ KPCR에는 EPROCESS의 KTHREAD 구조체 포인터 값이 저장됨
- ✓ 결국, KTHREAD를 이용해 EPROCESS의 위치 확인 가능

• DKOM과 같은 프로세스 은닉 기법 탐지 불가능!!

오브젝트 검색

▪ 패턴 매칭

- 프로세스 구조체의 패턴을 이용해 메모리 영역 전체 검색
- 은닉 프로세스라도 동일한 프로세스 구조체를 가짐
- 매칭 기준
 - ✓ 프로세스와 스레드는 오브젝트로 존재하며, 모든 오브젝트는 OBJECT_HEADER를 포함
 - ✓ 프로세스와 스레드는 동기화가 필요하므로 하부 구조체로 DISPATCHER_HEADER를 포함
 - ✓ 프로세스와 스레드는 중요 오브젝트이므로 page pool이 아닌 non-paged pool에 기록 (POOL_HEADER 포함)

물리메모리 분석 도구

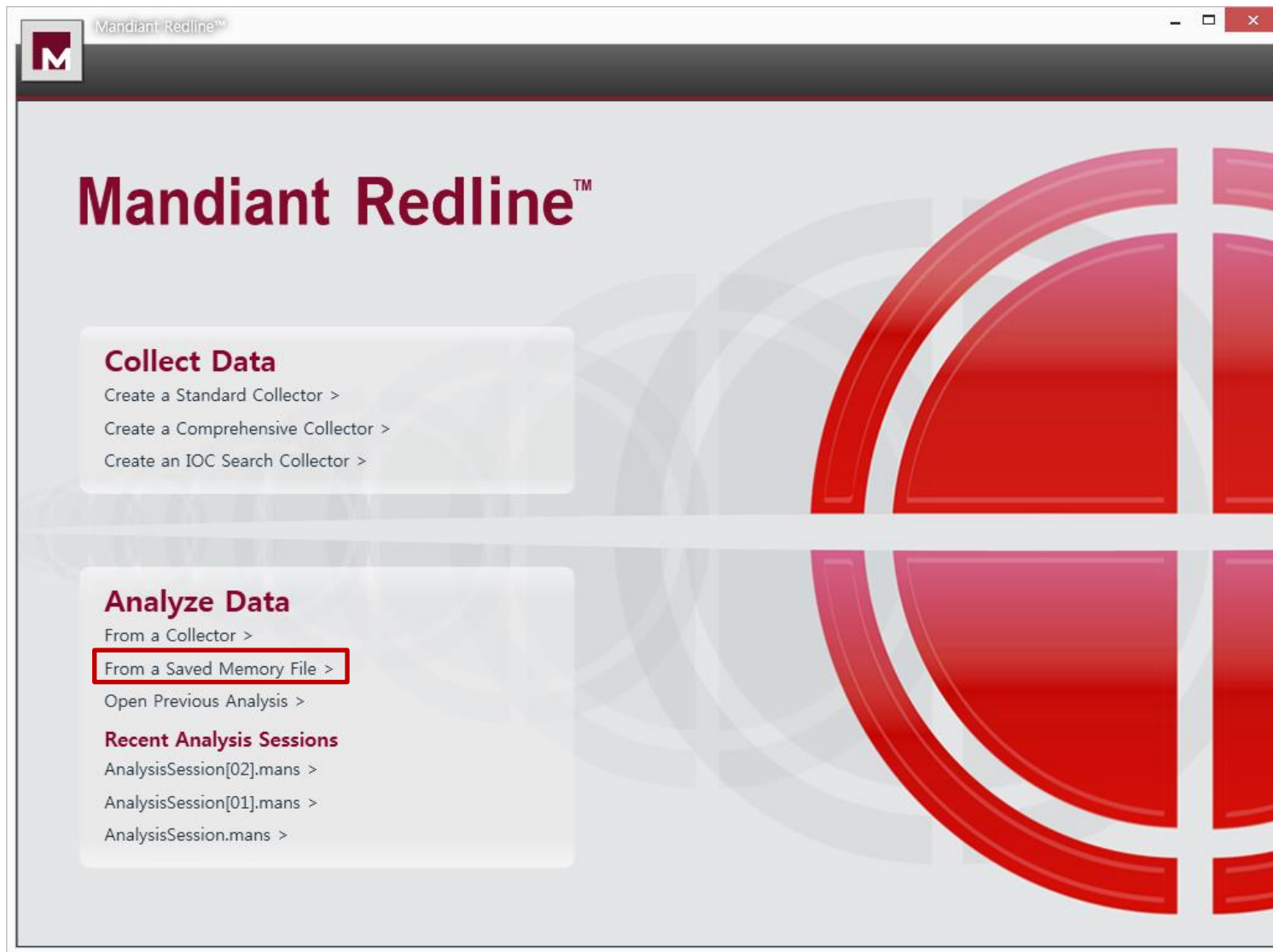
이름	인터페이스	플랫폼	제조사	라이선스
Redline™	GUI	Windows	Mandiant	Freeware
Volatility	CLI	Anywhere	Volatile Systems	Opensource
Responder Pro	GUI	Windows	HBGary	Commercial
Second Look® Linux Memory Analysis	CLI	Linux	Raytheon Pikewerks	Commercial
Volafox	CLI	Mac OS	n0fate	Opensource
Volafunx	CLI	FreeBSD	n0fate	Opensource

물리메모리 분석 도구 비교


이름	XP	2003	Vista	2008	2008R2	7	2012	2012R2	8, 8.1
Redline™	32(SP2)	32(SP2) 64(SP2)	32	32 64	64	32 64	-	-	32 64
Volatility	32(SP2,3) 64(SP1,2)	32(SP0,1,2) 64(SP1,2)	32(SP0,1,2) 64(SP0,1,2)	32(SP1,2) 64(SP1,2)	64(SP0,1)	32(SP0,1) 64(SP0,1)	64	64	32 64
Responder Pro	32 64	32 64	32 64	32 64	32 64	32 64	32 64		32 64

이름	Linux	Mac OSX
Redline™	-	-
Volatility	32(Kernel 2.6.11 – 3.5) 64(Kernel 2.6.11 – 3.5)	32(10.5.x, 10.6.x, 10.7.x, 10.8.x, 10.9.x) 64(10.7.x, 10.8.x, 10.9.x)
Responder Pro	-	-

Redline™



Redline™



Start your Analysis Session

Instructions

Select the memory image captured. Redline will analyze this image to aid in navigation and to assist in the identification of potential issues. Redline can also search for Indicators of Compromise (described below) at this time. If you wish to search for Indicators of Compromise (IOCs) at a later time, this option can be found under the IOC reports tab.

Indicators of Compromise:

Indicators of Compromise are forensic artifacts left behind by an intrusion. An IOC file describes these artifacts using the OpenIOC format. When configuring an audit, Redline verifies that the correct data exists, or will be acquired, so that these artifacts can be identified.

Note that IOC sets may be run against an audit after collection and analysis. Redline will issue a warning if an IOC

Configuration

Location of Saved Memory Image: [Open Containing Folder](#)

C:\Temp\Audits\FORENSIC32-PC\20130806153712\memory.11333f11.img

Browse...

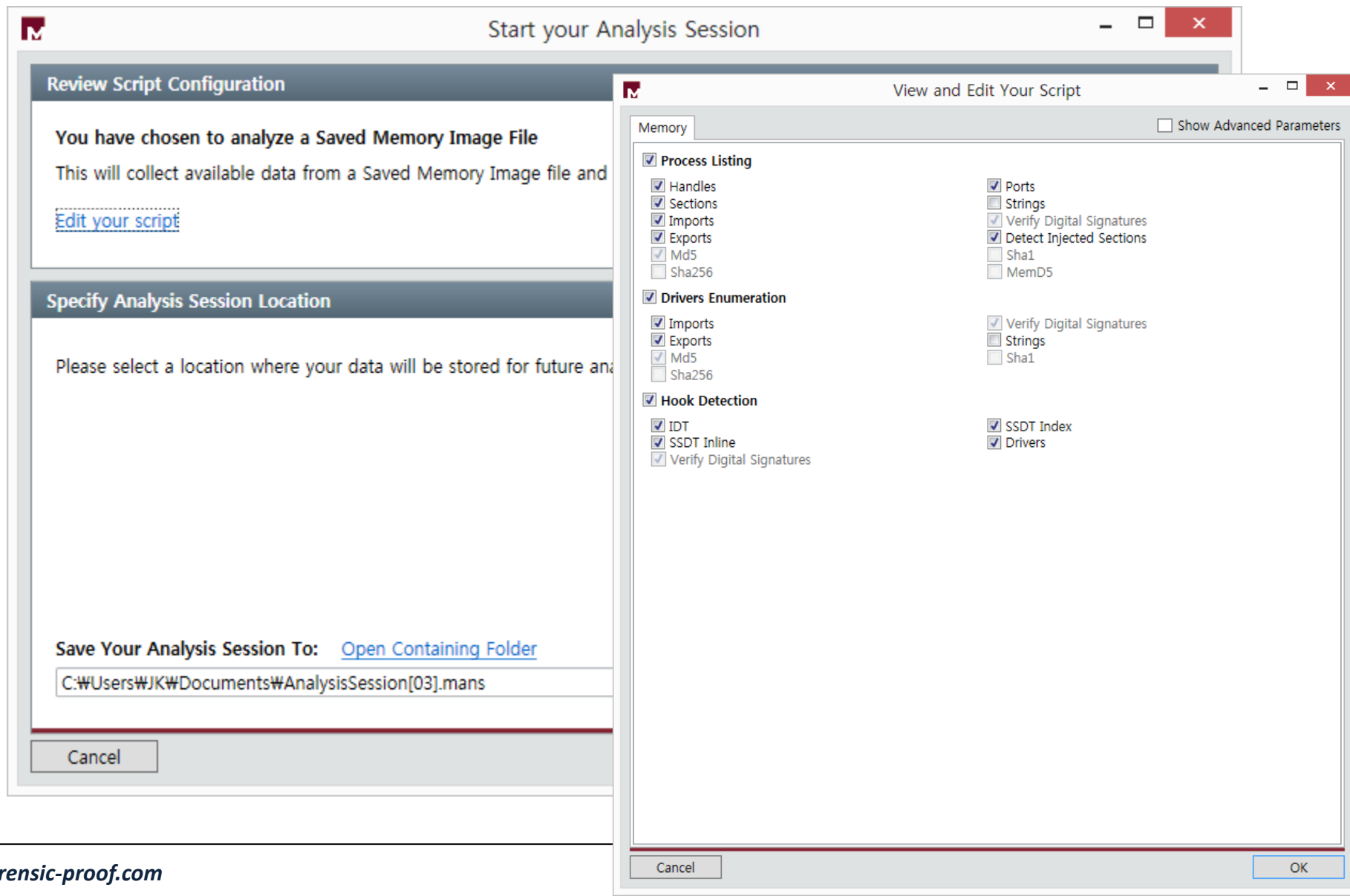
☐ Indicators of Compromise Location: [Open Folder](#)

Choose a directory containing your Indicators of Compromise

Browse...

Cancel

Next



Redline™

Mandiant Redline™

Home ▶ Host ▶ Timeline

Analysis Data

- Processes
 - Handles
 - Memory Sections
 - Strings
 - Ports
 - Hierarchical Processes
- Driver Modules
- Device Tree
- Hooks
- Timeline**
- Tags and Comments
- Acquisition History

Timeline Configuration

☐ Show All Deselect All ☐

Files:

- ☒ Created
- ☐ Accessed
- ☒ Modified
- ☒ Changed
- ☒ FilenameCreated
- ☐ FilenameAccessed
- ☒ FilenameModified
- ☒ FilenameChanged
- ☒ PEInfo/Exports/ExportsTimestamp
- ☒ PEInfo/PETimestamp

Processes:

- ☒ StartTime

Registry:

- ☒ Modified

Event Logs:

- ☒ GenTime
- ☐ WriteTime

Tasks:

- ☒ NextRunTime
- ☒ MostRecentRunTime
- ☒ CreationDate
- ☐ Trigger/Begin
- ☐ Trigger/End

Timestamp	Field	Summary
2013-08-06 02:25:38Z	Process/StartTime	Name: lsmd.exe (672) Path:
2013-08-06 02:25:38Z	Process/StartTime	Name: winlogon.exe (680) Path:
2013-08-06 02:25:38Z	Process/StartTime	Name: services.exe (636) Path:
2013-08-06 02:25:38Z	Process/StartTime	Name: lsass.exe (644) Path:
2013-08-06 02:25:38Z	Process/StartTime	Name: svchost.exe (800) Path:
2013-08-06 03:50:21Z	Process/StartTime	Name: System (4) Path:
2013-08-06 03:50:21Z	Process/StartTime	Name: smss.exe (372) Path:
2013-08-06 03:50:23Z	Process/StartTime	Name: csrss.exe (512) Path: C:\Win
2013-08-06 03:50:24Z	Process/StartTime	Name: wininit.exe (572) Path:
2013-08-06 03:50:24Z	Process/StartTime	Name: csrss.exe (564) Path: C:\Win
2013-08-06 03:50:24Z	Process/StartTime	Name: services.exe (624) Path: C:\Win
2013-08-06 03:50:25Z	Process/StartTime	Name: lsmd.exe (676) Path: C:\Win
2013-08-06 03:50:25Z	Process/StartTime	Name: lsass.exe (648) Path: C:\Win
2013-08-06 03:50:25Z	Process/StartTime	Name: winlogon.exe (684) Path:
2013-08-06 03:50:30Z	Process/StartTime	Name: svchost.exe (1060) Path: C:\Win
2013-08-06 03:50:30Z	Process/StartTime	Name: svchost.exe (872) Path: C:\Win
2013-08-06 03:50:30Z	Process/StartTime	Name: svchost.exe (964) Path: C:\Win
2013-08-06 03:50:30Z	Process/StartTime	Name: svchost.exe (1016) Path: C:\Win
2013-08-06 03:50:30Z	Process/StartTime	Name: svchost.exe (1092) Path: C:\Win

Host IOC Reports Not Collected

TimeCrunches™ 0 TimeWrinkles™ 0 Processes

84 Items

Volatility 준비

- **Volatility**

- <https://github.com/volatilityfoundation/volatility>

- **Volatility Profiles**

- <https://github.com/volatilityfoundation/profiles>

- **Memory Forensics Cheat Sheet v1.2**

- <http://forensicmethods.com/memory-forensics-cheat-sheet-v1-2>

- **Volatility Cheat Sheet v2.3**

- <https://code.google.com/p/volatility/downloads/list>



Volatility 사용법

- 이미지 프로파일 확인

```
$> vol.py -f [image] imageinfo
```

- 플러그인 확인

```
$> vol.py --info
```

- 플러그인 옵션

```
$> vol.py [plugin] --help
```

- 외부 플러그인 로드

```
$> vol.py --plugins=[path] [plugin]
```

Volatility 사용법

- 프로세스 목록

```
$> vol.py -f [image] --profile=[profile] pslist
```

```
$> vol.py -f [image] --profile=[profile] psscan
```

```
$> vol.py -f [image] --profile=[profile] psxview
```

```
$> vol.py -f [image] --profile=[profile] pstree
```

Volatility 사용법

- 프로세스 관련 정보

```
$> vol.py -f [image] --profile=[profile] dlllist
```

```
$> vol.py -f [image] --profile=[profile] vadinfo
```

```
$> vol.py -f [image] --profile=[profile] handles
```

```
$> vol.py -f [image] --profile=[profile] privs
```

```
$> vol.py -f [image] --profile=[profile] threads
```

Volatility 사용법

- PE 관련 정보 추출

```
$> vol.py -f [image] --profile=[profile] -D [dir] moddump
```

```
$> vol.py -f [image] --profile=[profile] -D [dir] procexedump
```

```
$> vol.py -f [image] --profile=[profile] -D [dir] procmemdump
```

```
$> vol.py -f [image] --profile=[profile] -D [dir] dlldump
```

Volatility 사용법

- 인젝션 코드 확인

```
$> vol.py -f [image] --profile=[profile] -D [dir] malfind
```

```
$> vol.py -f [image] --profile=[profile] ldrmodules
```

```
$> vol.py -f [image] --profile=[profile] impscan
```


Volatility 사용법

- 네트워크 관련 정보

```
$> vol.py -f [image] --profile=[XP/2003 profile] connections
```

```
$> vol.py -f [image] --profile=[XP/2003 profile] sockets
```

```
$> vol.py -f [image] --profile=[XP/2003 profile] connscan
```

```
$> vol.py -f [image] --profile=[Vista/2008/7 profile] netscan
```

Volatility 사용법

- 로그 확인

```
$> vol.py -f [image] --profile=[profile] cmdscan
```

```
$> vol.py -f [image] --profile=[profile] consoles
```

```
$> vol.py -f [image] --profile=[profile] svcscan
```

- SID 확인

```
$> vol.py -f [image] --profile=[profile] getsids
```

- 환경변수

```
$> vol.py -f [image] --profile=[profile] envvars
```

Volatility 사용법

- 커널 메모리

```
$> vol.py -f [image] --profile=[profile] modules
```

```
$> vol.py -f [image] --profile=[profile] modscan
```

```
$> vol.py -f [image] --profile=[profile] timers
```

```
$> vol.py -f [image] --profile=[profile] callbacks
```

```
$> vol.py -f [image] --profile=[profile] ssdt
```

```
$> vol.py -f [image] --profile=[profile] idt
```

```
$> vol.py -f [image] --profile=[profile] gdt
```

```
$> vol.py -f [image] --profile=[profile] devicetree
```

Volatility 사용법

- 커널 오브젝트

```
$> vol.py -f [image] --profile=[profile] driverscan
```

```
$> vol.py -f [image] --profile=[profile] mutantscan
```

```
$> vol.py -f [image] --profile=[profile] filescan
```

```
$> vol.py -f [image] --profile=[profile] symlinkscan
```

Volatility 사용법

- 레지스트리 관련

```
$> vol.py -f [image] --profile=[profile] hivelist
```

```
$> vol.py -f [image] --profile=[profile] printkey
```

```
$> vol.py -f [image] --profile=[profile] userassist
```

```
$> vol.py -f [image] --profile=[profile] shellbags
```

```
$> vol.py -f [image] --profile=[profile] shimcache
```

Volatility 사용법

- 패스워드 관련

```
$> vol.py -f [image] --profile=[profile] lsadump
```

```
$> vol.py -f [image] --profile=[profile] hashdump
```

```
$> vol.py -f [image] --profile=[profile] dumptcerts
```

Volatility 사용법

- YARA 관련

```
$> vol.py -f [image] --profile=[profile] yarascan  
-yara-file=/path/to/rules/yar
```

```
$> vol.py -f [image] --profile=[profile] yarascan -D [dump file]  
--yara-rules="simpleStringToFind"
```

챌린지

- #1 Honeynet 2010 Forensic Challenge – Banking Troubles
- #2 DC3 2013 301 – Windows Memory Image Analysis
- #3 SANS DFIR Challenge – APT Attack Analysis

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

- **The Challenge;** http://www.honeynet.org/challenges/2010_3_banking_troubles

Company X has contacted you to perform forensics work on a recent incident that occurred. **One of their employees had received an email from a fellow co-worker that pointed to a PDF file.** Upon opening, the employee did not seem to notice anything, however **recently they have had unusual activity in their bank account.** Company X was able to obtain a memory image of the employee's virtual machine upon suspected infection. **Company X wishes you to analyze the virtual memory and report on any suspected activities found.** Questions can be found below to help in the formal report for the investigation.

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

■ Questions;

1. List the processes that were running on the victim's machine. Which process was most likely responsible for the initial exploit? (2pts)
2. List the sockets that were open on the victim's machine during infection. Are there any suspicious processes that have sockets open? (4pts)
3. List any suspicious URLs that may be in the suspected process's memory. (2pts)
4. Are there any other processes that contain URLs that may point to banking troubles? If so, what are these processes and what are the URLs? (4pts)
5. Were there any files that were able to be extracted from the initial process? How were these files extracted? (6pts)
6. If there was a file extracted from the initial process, what techniques did it use to perform the exploit? (8pts)
7. List suspicious files that were loaded by any processes on the victim's machine. From this information, what was a possible payload of the initial exploit be that would be affecting the victim's bank account? (2pts)
8. If any suspicious files can be extracted from an injected process, do any anti-virus products pick up the suspicious executable? What is the general result from anti-virus products? (6pts)
9. Are there any related registry entries associated with the payload? (4pts)
10. What technique was used in the initial exploit to inject code in to the other processes? (6pts)

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

■ 프로파일 확인

```
F:\TEMP\volatility-read-only>vol.py -f Bob.vmem imageinfo
Volatile Systems Volatility Framework 2.3_beta
Determining profile based on KDBG search...

Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated with WinXPSP2x86)
AS Layer1 : IA32PagedMemoryPae (Kernel AS)
AS Layer2 : FileAddressSpace (F:\TEMP\volatility-read-only\Bob.vmem)
PAE type : PAE
DTB : 0x319000L
KDBG : 0x80544ce0L
Number of Processors : 1
Image Type (Service Pack) : 2
KPCR for CPU 0 : 0xffdff000L
KUSER_SHARED_DATA : 0xffdf0000L
Image date and time : 2010-02-27 20:12:38 UTC+0000
Image local date and time : 2010-02-27 15:12:38 -0500
```

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

■ 서비스팩 확인

```
F:\TEMP\volatility-read-only>vol.py -f Bob.vmem dlllist | findstr "xpsp2"
Volatile Systems Volatility Framework 2.3_beta
0x01530000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x2 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x5 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
0x20000000      0x2c5000      0x1 C:\WINDOWS\system32\xpsp2res.dll
```

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

1. List the processes that were running on the victim's machine.
Which process was most likely responsible for the initial exploit? (2pts)

```
F:\TEMP\volatility-read-only>vol.py -f Bob.vmem --profile=WinXPSP2x86 psxview
Volatile Systems Volatility Framework 2.3_beta
Offset(P)  Name                      PID pslist psscan thrdproc pspcid csrss session deskthrd
-----
0x02268020 firefox.exe                888 True  True  True    True   True  True   True
0x022d6b88 alg.exe                 2024 True  True  True    True   True  True   True
0x024ea020 svchost.exe                1040 True  True  True    True   True  True   True
0x01edd790 explorer.exe             1756 True  True  True    True   True  True   True
0x02456da0 services.exe             688 True  True  True    True   True  True   True
0x01fe55f0 svchost.exe                1244 True  True  True    True   True  True   True
0x023018b0 vmtoolsd.exe            1628 True  True  True    True   True  True   True
0x022618c8 AcroRd32.exe             1752 True  True  True    True   True  True   True
0x024e1da0 svchost.exe                948 True  True  True    True   True  True   True
0x01fea020 svchost.exe                1100 True  True  True    True   True  True   True
0x02409640 svchost.exe                1384 True  True  True    True   True  True   True
0x02329da0 lsass.exe                 700 True  True  True    True   True  True   True
0x0241a020 wuauc.lt.exe               232 True  True  True    True   True  True   True
0x022cd5c8 VMwareUser.exe        1116 True  True  True    True   True  True   True
... ..
```

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

1. List the processes that were running on the victim's machine.
Which process was most likely responsible for the initial exploit? (2pts)

```
F:\TEMP\volatility-read-only>vol.py -f Bob.vmem --profile=WinXPSP2x86 pslist
Volatile Systems Volatility Framework 2.3_beta
```

Offset (V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start
0x823c8830	System	4	0	58	573	-----	0	
0x81f04228	smss.exe	548	4	3	21	-----	0	2010-02-26 03:34:02 UTC+0000
0x822eeda0	csrss.exe	612	548	12	423	0	0	2010-02-26 03:34:04 UTC+0000
0x81e5b2e8	winlogon.exe	644	548	21	521	0	0	2010-02-26 03:34:04 UTC+0000
0x82256da0	services.exe	688	644	16	293	0	0	2010-02-26 03:34:05 UTC+0000
0x82129da0	lsass.exe	700	644	22	416	0	0	2010-02-26 03:34:06 UTC+0000
0x81d3f020	vmacthlp.exe	852	688	1	35	0	0	2010-02-26 03:34:06 UTC+0000
0x82266870	svchost.exe	880	688	28	340	0	0	2010-02-26 03:34:07 UTC+0000
0x822e1da0	svchost.exe	948	688	10	276	0	0	2010-02-26 03:34:07 UTC+0000
0x822ea020	svchost.exe	1040	688	83	1515	0	0	2010-02-26 03:34:07 UTC+0000
...	...							
0x81celaf8	msiexec.exe	452	244	0	-----	0	0	2010-02-26 03:46:07 UTC+0000
0x81c80c78	wuauclt.exe	440	1040	8	188	0	0	2010-02-27 19:48:49 UTC+0000
0x8221a020	wuauclt.exe	232	1040	4	136	0	0	2010-02-27 19:49:11 UTC+0000
0x82068020	firefox.exe	888	1756	9	172	0	0	2010-02-27 20:11:53 UTC+0000
0x820618c8	AcroRd32.exe	1752	888	8	184	0	0	2010-02-27 20:12:23 UTC+0000
0x82209640	svchost.exe	1384	688	9	101	0	0	2010-02-27 20:12:36 UTC+0000

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

1. List the processes that were running on the victim's machine.

Which process was most likely responsible for the initial exploit? (2pts)

```
F:\TEMP\volatility-read-only>vol.py -f Bob.vmem --profile=WinXPSP2x86 pstree
```

```
Volatile Systems Volatility Framework 2.3_beta
```

Name	Pid	PPid	Thds	Hnds	Time
0x81cdd790:explorer.exe	1756	1660	14	345	2010-02-26 03:34:38 UTC+0000
. 0x82068020:firefox.exe	888	1756	9	172	2010-02-27 20:11:53 UTC+0000
.. 0x820618c8:AcroRd32.exe	1752	888	8	184	2010-02-27 20:12:23 UTC+0000
0x823c8830:System	4	0	58	573	1970-01-01 00:00:00 UTC+0000
. 0x81f04228:smss.exe	548	4	3	21	2010-02-26 03:34:02 UTC+0000
.. 0x81e5b2e8:winlogon.exe	644	548	21	521	2010-02-26 03:34:04 UTC+0000
... 0x82256da0:services.exe	688	644	16	293	2010-02-26 03:34:05 UTC+0000
.... 0x822ea020:svchost.exe	1040	688	83	1515	2010-02-26 03:34:07 UTC+0000
..... 0x81cee5f8:wscntfy.exe	1132	1040	1	38	2010-02-26 03:34:40 UTC+0000
..... 0x8221a020:wuaclt.exe	232	1040	4	136	2010-02-27 19:49:11 UTC+0000
.... 0x81de55f0:svchost.exe	1244	688	19	239	2010-02-26 03:34:08 UTC+0000
.... 0x81dde568:spoolsv.exe	1460	688	11	129	2010-02-26 03:34:10 UTC+0000
.... 0x82209640:svchost.exe	1384	688	9	101	2010-02-27 20:12:36 UTC+0000
.... 0x820d6b88:alg.exe	2024	688	7	130	2010-02-26 03:34:35 UTC+0000
.... 0x82266870:svchost.exe	880	688	28	340	2010-02-26 03:34:07 UTC+0000
.... 0x82333620:msiexec.exe	244	688	5	181	2010-02-26 03:46:06 UTC+0000
... ..					

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

2. List the sockets that were open on the victim's machine during infection.
Are there any suspicious processes that have sockets open? (4pts)

```
F:\TEMP\volatility-read-only>vol.py -f Bob.vmem --profile=WinXPSP2x86 connections
Volatile Systems Volatility Framework 2.3_beta
```

Offset (V)	Local Address	Remote Address	Pid	
0x81c6a9f0	192.168.0.176:1176	212.150.164.203:80	888	(Israel)
0x82123008	192.168.0.176:1184	193.104.22.71:80	880	(Iran)
0x81cd4270	192.168.0.176:2869	192.168.0.1:30379	1244	
0x81e41108	127.0.0.1:1168	127.0.0.1:1169	888	
0x8206ac58	127.0.0.1:1169	127.0.0.1:1168	888	
0x82108890	192.168.0.176:1178	212.150.164.203:80	1752	(Israel)
0x82210440	192.168.0.176:1185	193.104.22.71:80	880	(Iran)
0x8207ac58	192.168.0.176:1171	66.249.90.104:80	888	(US, Google Inc.)
0x81cef808	192.168.0.176:2869	192.168.0.1:30380	4	
0x81cc57c0	192.168.0.176:1189	192.168.0.1:9393	1244	
0x8205a448	192.168.0.176:1172	66.249.91.104:80	888	(US, Google Inc.)

880(svchost.exe), 888(firefox.exe), 1752(AcroRd32.exe)

<http://wq.apnic.net/apnic-bin/whois.pl>

http://www.ip-adress.com/ip_tracer/

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

2. List the sockets that were open on the victim's machine during infection.
Are there any suspicious processes that have sockets open? (4pts)

```
F:\TEMP\volatility-read-only>vol.py -f Bob.vmem --profile=WinXPSP2x86 sockets
Volatile Systems Volatility Framework 2.3_beta
```

Offset (V)	PID	Port	Proto	Protocol	Address	Create Time
0x81cf2998	1040	68	17	UDP	192.168.0.176	2010-02-27 20:12:35 UTC+0000
0x81c833a0	880	1185	6	TCP	0.0.0.0	2010-02-27 20:12:36 UTC+0000
0x82210c40	1244	1189	6	TCP	0.0.0.0	2010-02-27 20:12:37 UTC+0000
0x820ac218	1040	1181	17	UDP	192.168.0.176	2010-02-27 20:12:35 UTC+0000
0x81d0fe98	880	30301	6	TCP	0.0.0.0	2010-02-27 20:12:36 UTC+0000
0x81c96b98	1752	1178	6	TCP	0.0.0.0	2010-02-27 20:12:32 UTC+0000
0x81d1a8b8	1752	1177	17	UDP	127.0.0.1	2010-02-27 20:12:32 UTC+0000
0x820c37d0	1244	2869	6	TCP	0.0.0.0	2010-02-27 20:12:37 UTC+0000
0x81cc72b0	1040	1182	17	UDP	127.0.0.1	2010-02-27 20:12:35 UTC+0000
0x81cbd320	1040	1186	17	UDP	127.0.0.1	2010-02-27 20:12:36 UTC+0000
0x82061740	888	1176	6	TCP	0.0.0.0	2010-02-27 20:12:28 UTC+0000
0x81cde008	880	1184	6	TCP	0.0.0.0	2010-02-27 20:12:36 UTC+0000
...	...					

880(svchost.exe), 888(firefox.exe), 1752(AcroRd32.exe)

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

3. List any suspicious URLs that may be in the suspected process's memory. (2pts)

1. 880 (svchost.exe)
2. 888 (firefox.exe)
3. 1040 (svchost.exe)
4. 1244 (svchost.exe)
5. 1752 (AcroRd32.exe)

```
$> vol.py -f Bob.vmem --profile=WinXPSP2x86 memdump -p [PID] -D [TARGET]  
$> vol.py -f Bob.vmem --profile=WinXPSP2x86 yarascan -Y [rule] -p [PID] -D [TARGET]
```

```
$> strings [dumpfile] | findstr http:// | sort > [output file]  
$> strings [dumpfile] | findstr https:// | sort > [output file]
```

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

3. List any suspicious URLs that may be in the suspected process's memory. (2pts)

880(svchost.exe)

```
http://193.104.22.71/~produkt/69825439870/73846525#N
http://193.104.22.71/~produkt/983745213424/34650798253
http://193.104.22.71/~produkt/9j856f_4m9y8urb.php
https://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
https://banki....ng.*.de/cgi/ueberweisu
```

888(firefox.exe)

```
http://search-network-plus.com/cache/PDF.php?st=Internet%20Explorer%206.0
https://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
```

1040(svchost.exe)

```
https://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
```

1244(svchost.exe)

```
https://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
```

1752(AcroRd32.exe)

```
https://onlineeast#.bankofamerica.com/cgi-bin/ias/*/GotoWelcome
http://search-network-plus.com/load.php?a=a&st=Internet Explorer 6.0&e=2
```

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

4. Are there any other processes that contain URLs that may point to banking troubles? If so, **what are these processes and what are the URLs?** (4pts)

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

5. Were there any files that were able to be extracted from the initial process? **How were these files extracted?** (6pts)

```
$> vol.py -f Bob.vmem --profile=WinXPSP2x86 memdump -p 1752
```

- **FOREMOST**
- **RMF (Recover My Files)**
 - ✓ **EXE : 227**
 - ✓ **GIF : 7**
 - ✓ **HTML : 16**
 - ✓ **ICO : 1**
 - ✓ **Index.dat : 1**
 - ✓ **PDF : 2**
 - ✓ **PNG : 10**
 - ✓ **JPEG : 1**

#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

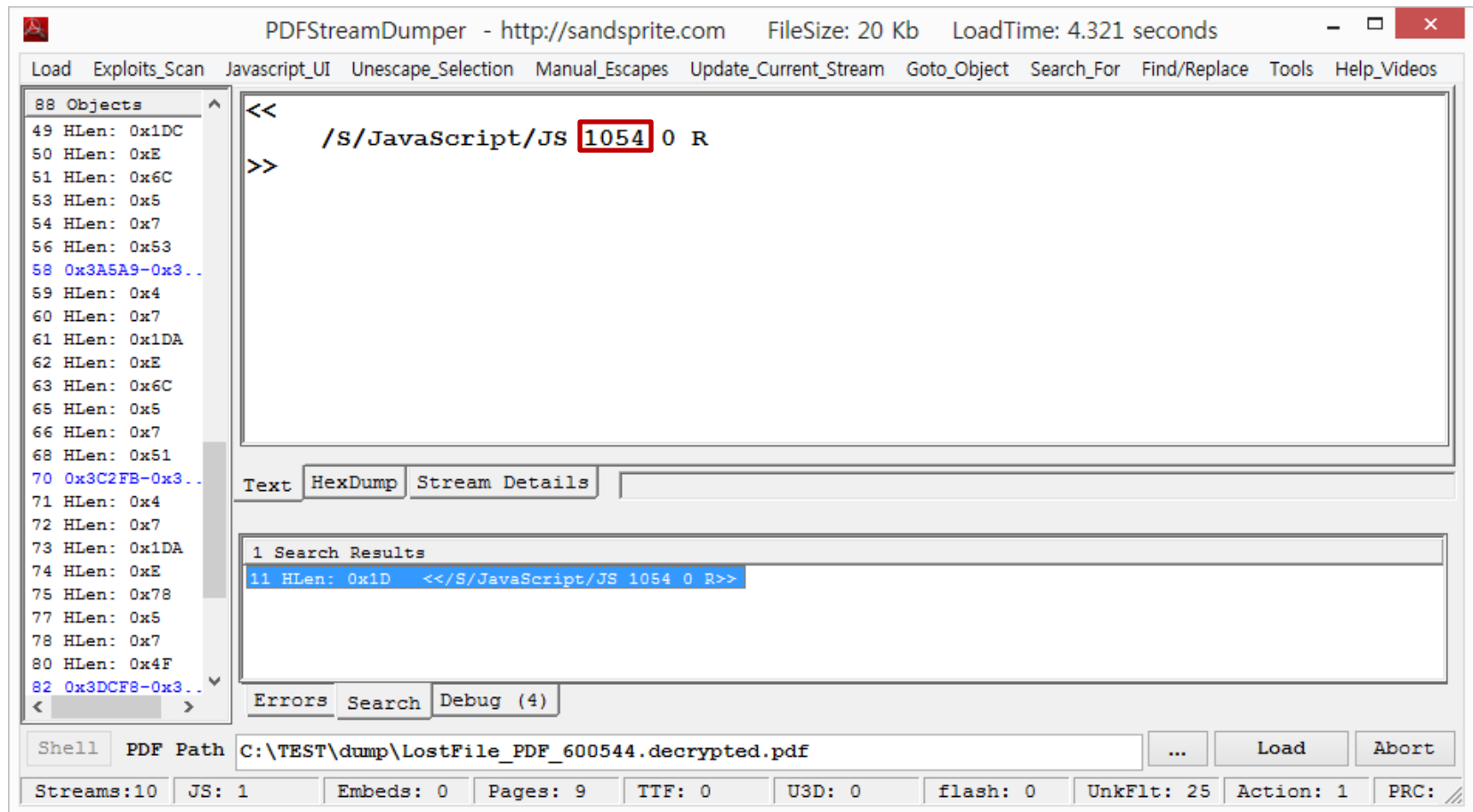
5. Were there any files that were able to be extracted from the initial process? How were these files extracted? (6pts)

- index.dat

Access Time (UTC+0)	URL	User	Filename	HTTP Header
2010-02-07 19:50:35.442	http://www.oldversion.com/download/firefox1502.exe	administrator	firefox1502[1].exe	HTTP/1.1 200 OK ETag: "13c95c-4e0830-452be78c0ae80" Content-Length: 5113904 Keep-Alive: timeout=5; max=97 Content-Type: application/x-msdownload
2010-02-07 20:12:34.100	http://search-network-plus.com/load.php?a=a&st=Internet%20Explorer%206.0&e=2	administrator	file[1].exe	HTTP/1.1 200 OK X-Powered-By: PHP/5.2.12 Pragma: no-cache Content-Transfer-Encoding: binary Content-Disposition: attachment; filename=file.exe; Content-Encoding: gzip Keep-Alive: timeout=1; max=100 Transfer-Encoding: chunked Content-Type: application/x-download Content-Language: ru
2010-02-07 20:12:34.100	http://search-network-plus.com/load.php?a=a&st=Internet Explorer 6.0&e=2		file[2].exe	HTTP/1.1 200 OK Content-Length: 110080 Content-Type: application/x-download Content-Disposition: attachment; filename=file.exe

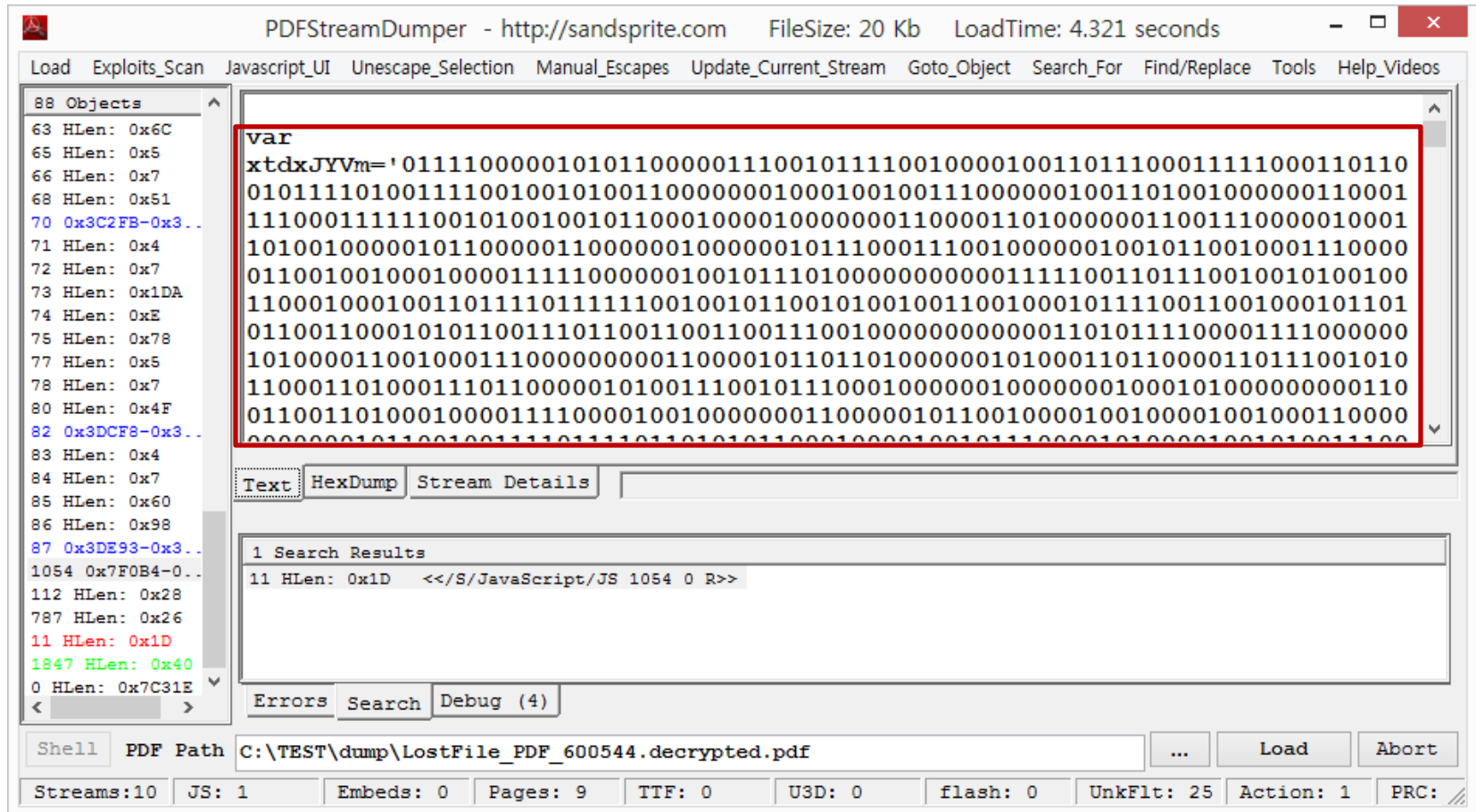
#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

6. If there was a file extracted from the initial process, what techniques did it use to perform the exploit? (8pts)



#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

6. If there was a file extracted from the initial process, what techniques did it use to perform the exploit? (8pts)



#1 – Honeynet 2010 Forensic Challenge – Banking Troubles

7. List suspicious files that were loaded by any processes on the victim's machine. From this information, what was a possible payload of the initial exploit be that would be affecting the victim's bank account? (2pts)

```
$> vol.py -f Bob.vmem --profile=WinXPSP2x86 filescan | findstr ".exe"
```

- \Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\Y9UHCP2P\file[1].exe
- \Device\HarddiskVolume1\DOCUME~1\ADMINI~1\LOCALS~1\Temp\e.exe
- \Device\HarddiskVolume1\WINDOWS\system32\sdra64.exe (Zeus Bot)
 - \Device\HarddiskVolume1\WINDOWS\system32\lowsec\local.ds
 - \Device\HarddiskVolume1\WINDOWS\system32\lowsec\user.ds.ill
 - \Device\HarddiskVolume1\WINDOWS\system32\lowsec\user.ds
 - <http://www.malwarehelp.org/find-and-remove-zeus-zbot-banking-trojan-2009.html>
 - <http://support.kaspersky.com/2020?el=88446>

#2 – DC3 2013 301 – Windows Memory Image Analysis

- **The Challenge;**

An employee of a large corporation is **accused of computer misuse**. Coworkers tipped the system administrator that the suspect **plays games and sends Instant Messages all day** from their computer. The system administrator monitors the employee's system to **find that the user is using a Virtual Machine**. When the employee goes on break, they suspend their system and the system administrator is able to **collect the virtual memory file (.vmem)**. You are called in to investigate and report your findings back to the company. Before leaving, you verify that the company had a computer use policy that prohibits such actions and that the suspect be the user agreement form, which allows all data of theirs to be monitored and/or seized for any reason at any time.

#2 – DC3 2013 301 – Windows Memory Image Analysis

▪ Questions;

1. What was the LOCAL date and time the image was captured?
(Example: Tuesday, January 01, 2034 01:59:59)
2. What Operating System and service pack was running at the time of capture?
3. What processes were running and what were their Pid's?
(List in numerical order by PID)
4. Were there any processes running relevant to the accusation and if so, what were they?
5. Were there any active windows open and if so, which ones were open?
6. Provide a screen shot of the active windows.

#3 – SANS DFIRCon – APT Malware & Memory Challenge

- The Challenge;



#3 – SANS DFIRCon – APT Malware & Memory Challenge

▪ Questions;

1. What is the Process ID of the rogue process on the system?
2. Determine the name or the rogue file that is found in the process (PID) that contains the rogue process found in the above question.
3. How is the malware achieving persistence on the system?
4. What is the filename of the file that is hiding the presence of the malware on the system?
5. What is the name of the ISP that hosts the network where the malware is communicating with?

