메모리 포렌식

MaJ3stY

saiwnsgud@gmail.com

http://maj3sty.tistory.com

Rather be dead than cool.

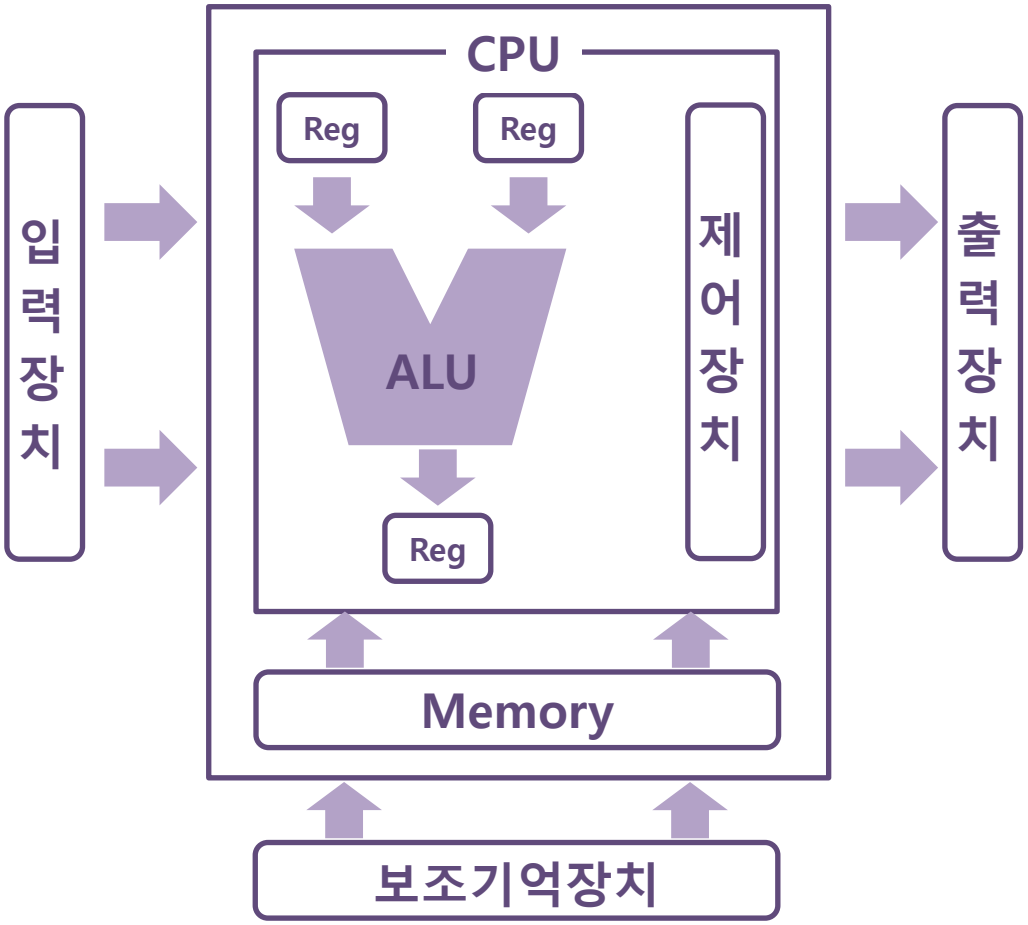
목 차

- Memory?!
- Memory Forensics?!
- Memory acquisition?!
- Memory Analysis?!
- Anti Memory Forensics?!
- Memory Forensics Challenge!

Memory?!

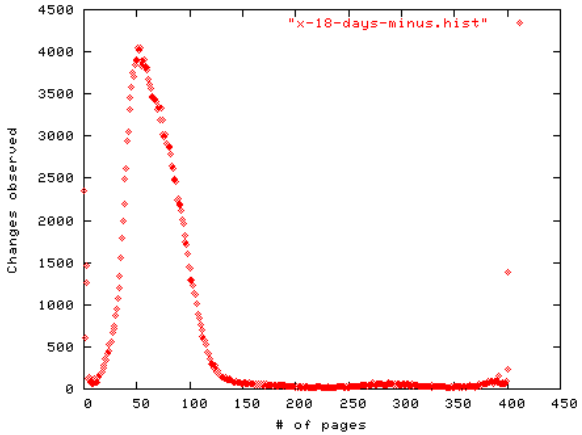
1. 메모리의 이해

- 컴퓨터의 기본 구조
  - ✓ 폰노이만 구조

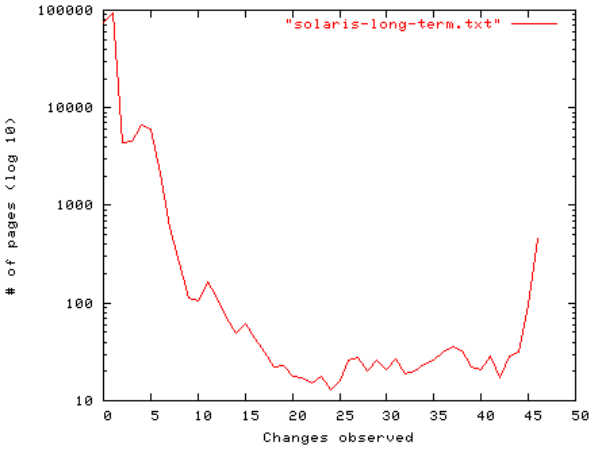


- 물리메모리에서의 데이터 지속성
  - ✓ Red Hat 6.1 and Solaris 8 Test

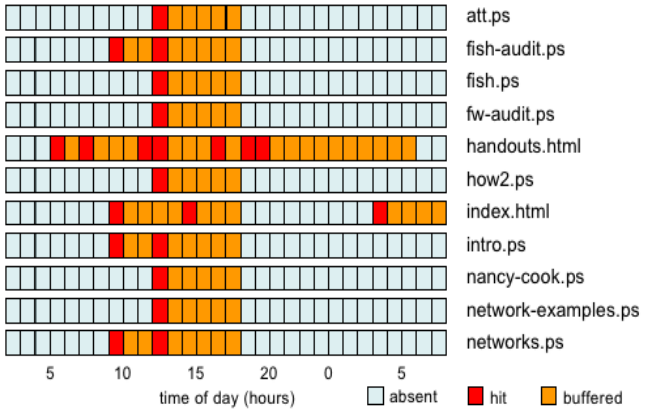
출처 : http://www.porcupine.org/forensics/forensic-discovery/chapter8.html



- 16.75일 동안 관찰한 결과
- 하루 당 65,000건 처리
- 메모리 페이지 해시 통계
- 2,350/256,000 페이지 유지

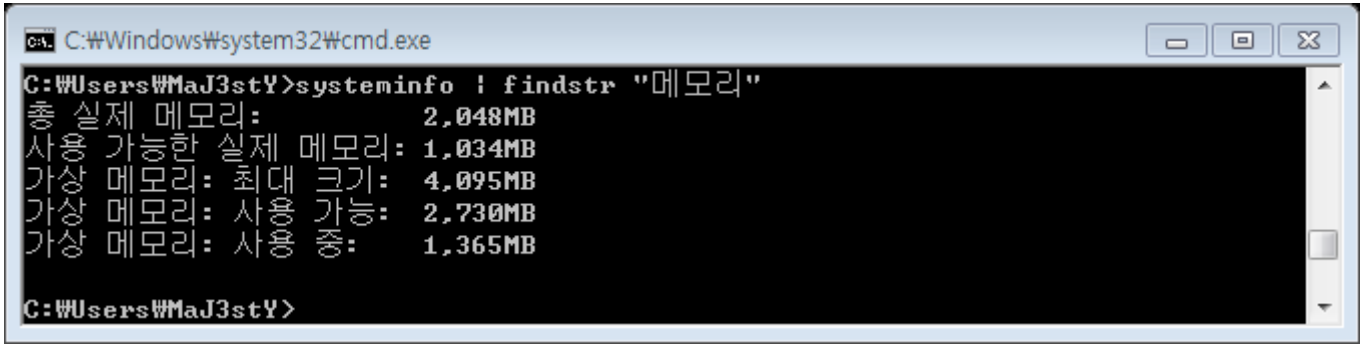


- 41.2일 동안 관찰한 결과
- 약 86% 페이지 유지
- 매일 평균 0.4% 정도 변화



- 파일 별 메모리 잔여 시간 측정

- 물리메모리
  - ✓ 실제 하드웨어 형태의 메모리



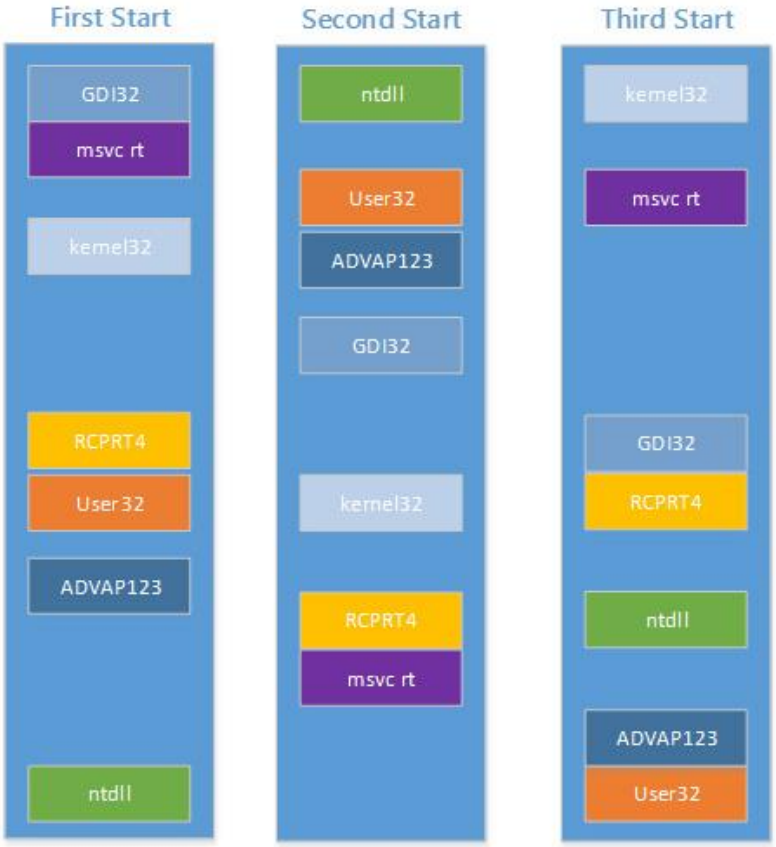
- 물리 주소 확장(Physical Address Extension, PAE)
  - ✓ 접근 가능한 물리주소의 bit를 증가시키는 기술(32bit -> 36bit)
  - ✓ 운영체제와 CPU에서 지원하는 기술
  - ✓ 대부분의 운영체제에서 지원

- 주소 윈도잉 확장(Address Windowing Extensions, AEW)
  - ✓ 사용자 영역 2GB -> 3GB
  - ✓ Lock Pages in Memory(winbase.h)

AWE API	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

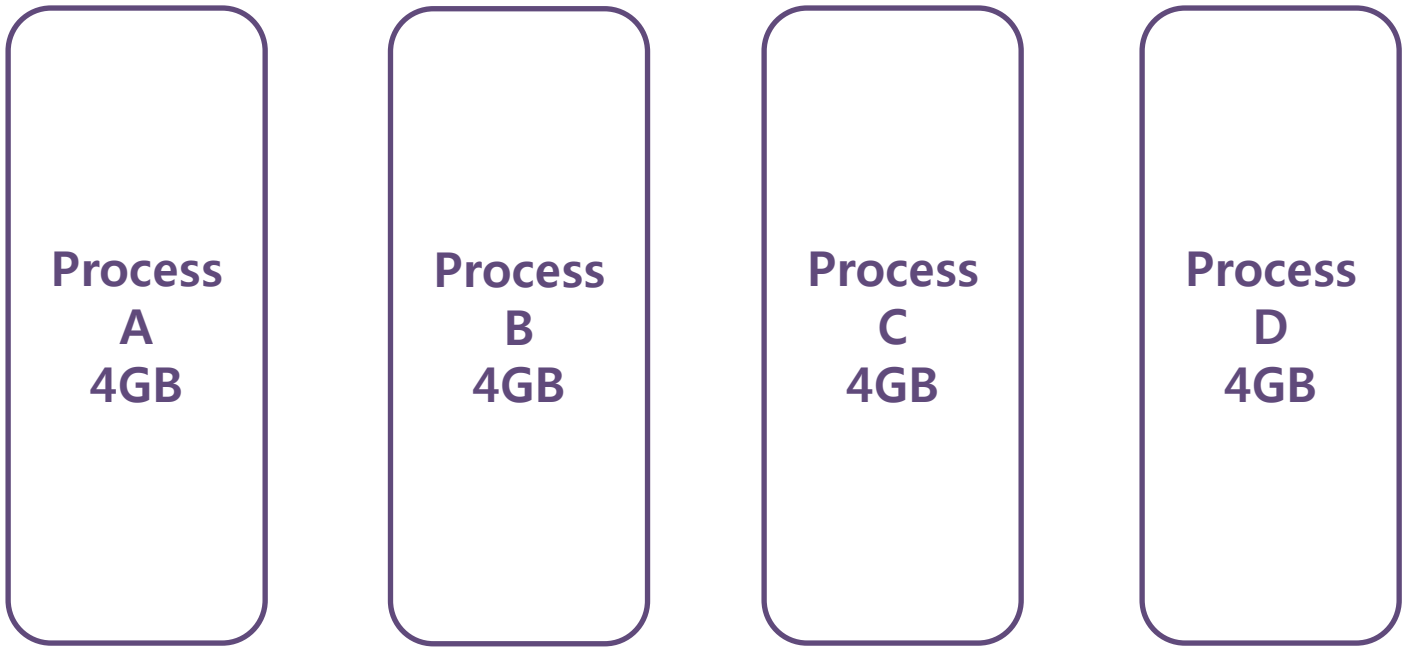
- **데이터 실행 방지(DEP, Data Execution Prevention)**
  - ✓ **데이터 실행 방지**
    - 스택, 힙 등과 같은 메모리 공간에서 코드가 실행되는 것이 불가능하도록 설정
    - Buffer Overflow 등의 공격 방지
  - ✓ **하드웨어 강제적 데이터 실행 방지**
    - 지원하지 않는 CPU 존재
    - CPU의 NX/XD 비트 설정
  - ✓ **소프트웨어 강제적 데이터 실행 방지**
    - 컴파일 시 /SafeSEH 옵션으로 적용

- 주소 공간 랜덤 배치(Address Space Layout Randomization, ASLR)
  - ✓ 실행되는 오브젝트들의 위치를 랜덤화

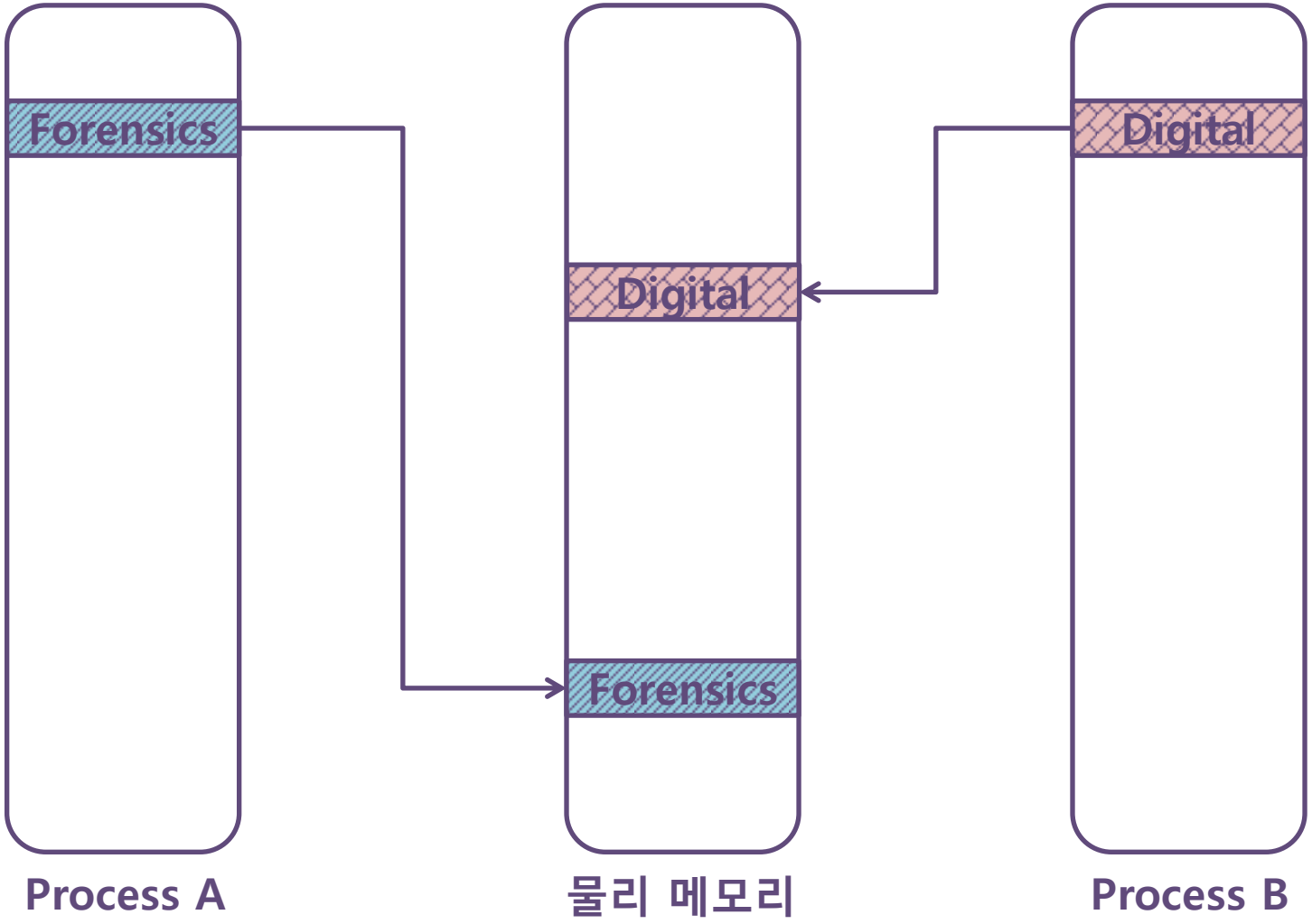


출처 : http://technet.microsoft.com/en-us/library/dn283963.aspx

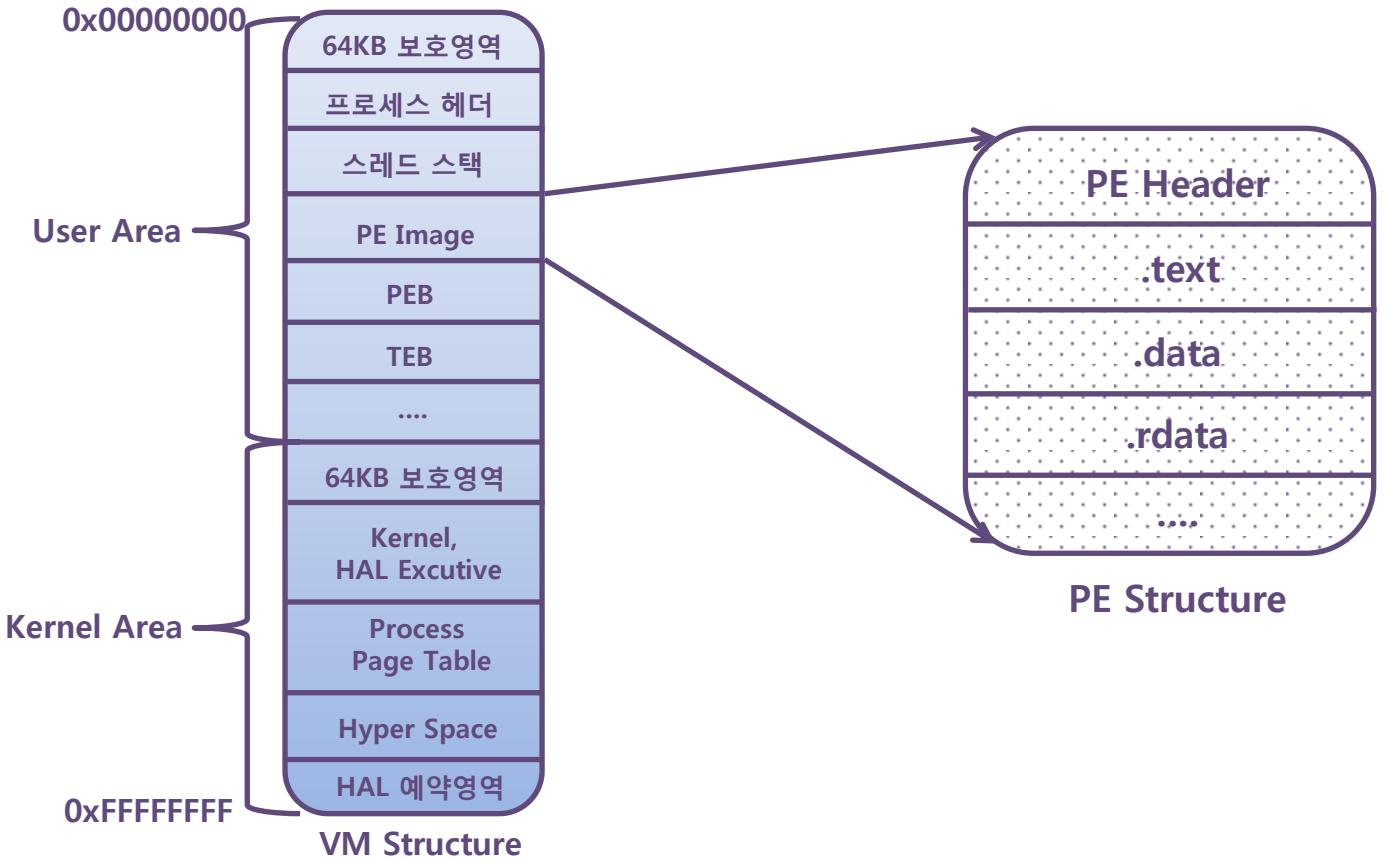
- 가상 메모리
  - ✓ 물리메모리의 부족함을 보충하기 위한 가상의 메모리
  - ✓ 각 프로세스는 자신만의 메모리로 가상 메모리 4GB를 소유
  - ✓ 가상 메모리 기법으로 대용량 프로세스의 실행 가능



- 가상/물리 메모리의 동작

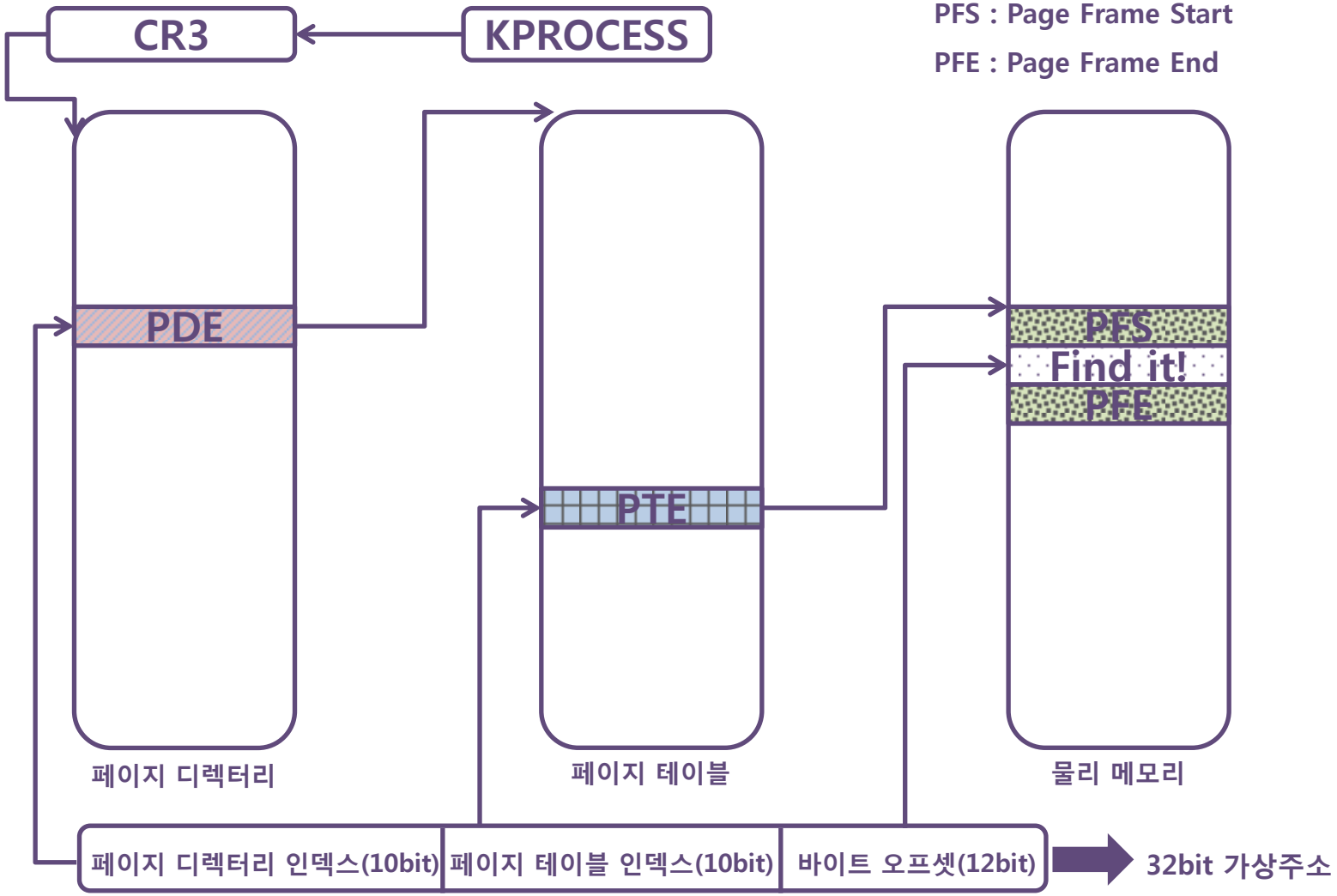

Forensics
Digital
Digital
Forensics
Process A
물리 메모리
Process B

- 가상 메모리 주소(x86)
  - ✓ User Area(0x00000000 ~ 0x7FFFFFFF)
  - ✓ Kernel Area(0x80000000 ~ 0xFFFFFFFF)



- 가상 주소 -> 물리 주소 변환

PDE : Page Directory Entry
PTE : Page Table Entry
PFS : Page Frame Start
PFE : Page Frame End



Memory Forensics?!

1. 메모리 포렌식의 중요성

- 메모리 포렌식
  - ✓ 분석 대상 컴퓨터의 메모리에서 정보 또는 증거로서 가치가 있는 데이터를 찾기 위한 분석 행위
  - ✓ 메모리 포렌식은 현재도 계속 연구 중인 분야 중 하나

- 메모리 포렌식이 활성화 된 이유
  - ✓ 악성코드 분석
  - ✓ 사용자의 프라이버시 강화
  - ✓ 안티 포렌식 기술의 발전

- 메모리에서 얻을 수 있는 것
  - ✓ 프로세스 실행 이미지
  - ✓ 네트워크 연결 정보
  - ✓ 사용자 행위
  - ✓ 복호화, 언패킹, 디코딩된 데이터
  - ✓ 암/복호화에 필요한 데이터
  - ✓ 기타...

Memory acquisition?!

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

- **Tribble – PCI 장치를 이용한 덤프**
  - ✓ 추가적인 하드웨어/소프트웨어 설치 없이 덤프
  - ✓ 무결성을 최대한 보장
  - ✓ 미리 설치되어 있어야 한다는 단점이 존재

참고 : A Hardware-Based Memory Acquisition Procedure for Digital Investigations

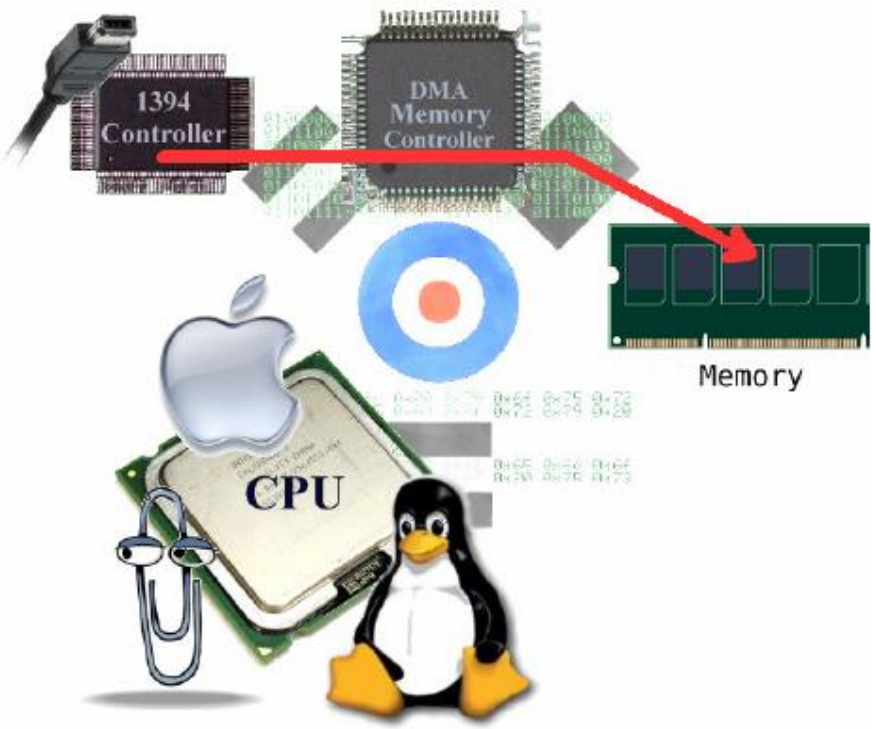
- **FireWire Attack – FireWire(IEEE 1394)를 이용한 덤프**
  - ✓ **FireWire Interface를 이용하여 DMA에 접근 해 메모리 덤프**
  - ✓ **Windows, Linux, Mac OS X 가능**

✓ **장점**

- 악성 프로그램의 영향 X
- 빠른 속도의 메모리 덤프
- 데이터 무결성 위반 최소화

✓ **단점**

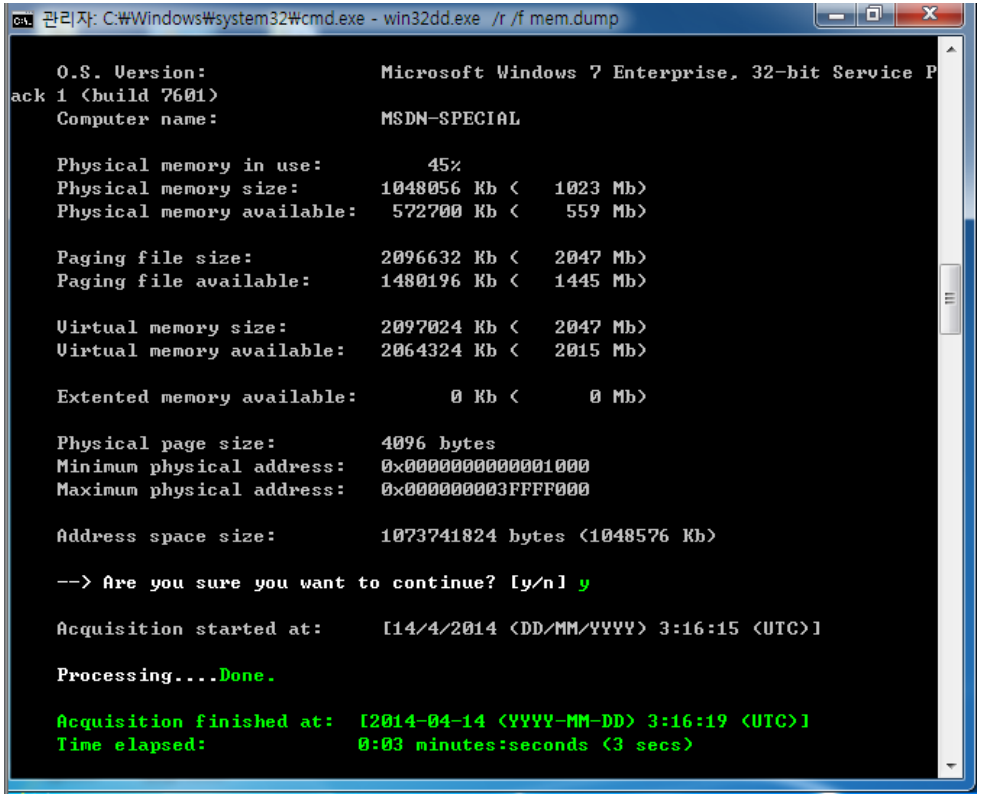
- 간혹 크래시 발생
- 접근할 수 있는 주소 제한



참고 : A Hardware-Based Memory Acquisition Procedure for Digital Investigations: A Forensic Approach

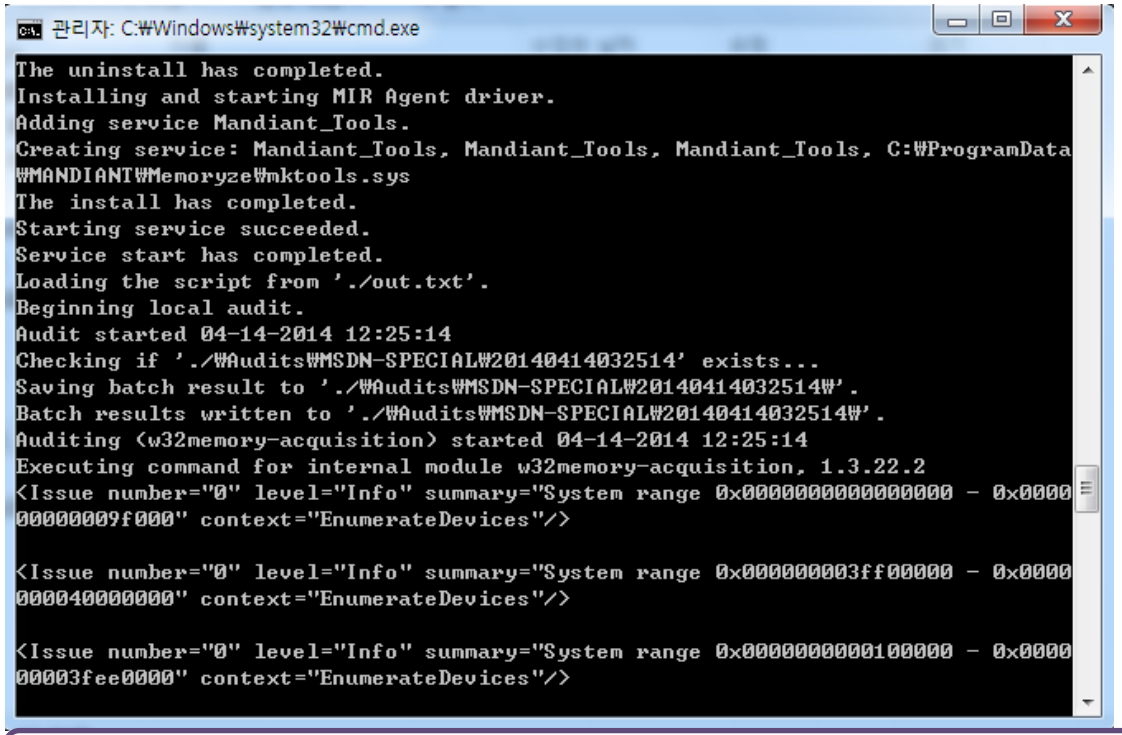
Memory acquisition?! – 소프트웨어

- **Win32/64dd**
  - ✓ Matthiew Suiche가 개발, 현재는 MoonSols window Memory Toolkit에 포함
  - ✓ 유/무료 버전으로 나뉨
  - ✓ 가격은 약 75만원
  - ✓ 윈도우 XP ~ 7까지 지원



Win32dd.exe /r /f <filename>

- **Memorize**
  - ✓ Mandiant 회사에서 개발한 무료 메모리 이미징 도구
  - ✓ Windows 2000 SP4 ~ 8, 32/64bit 모두 지원

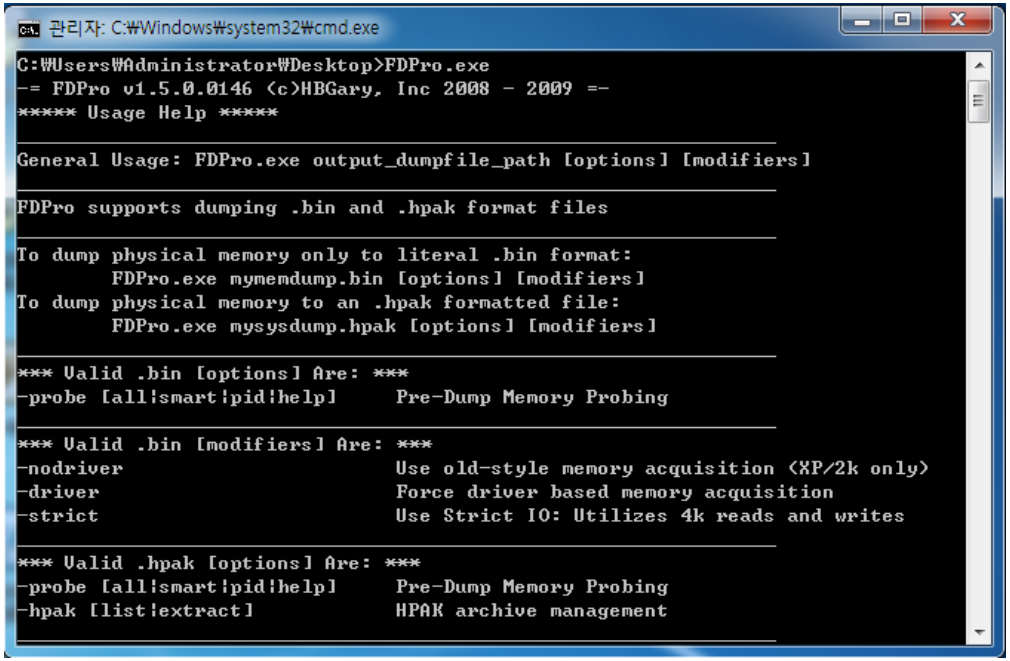


Memorize.exe –o <output path> -script <xml path> -encoding none -allowmultiple

MemoryDD.bat –output <output path>

Memory acquisition?! – 소프트웨어

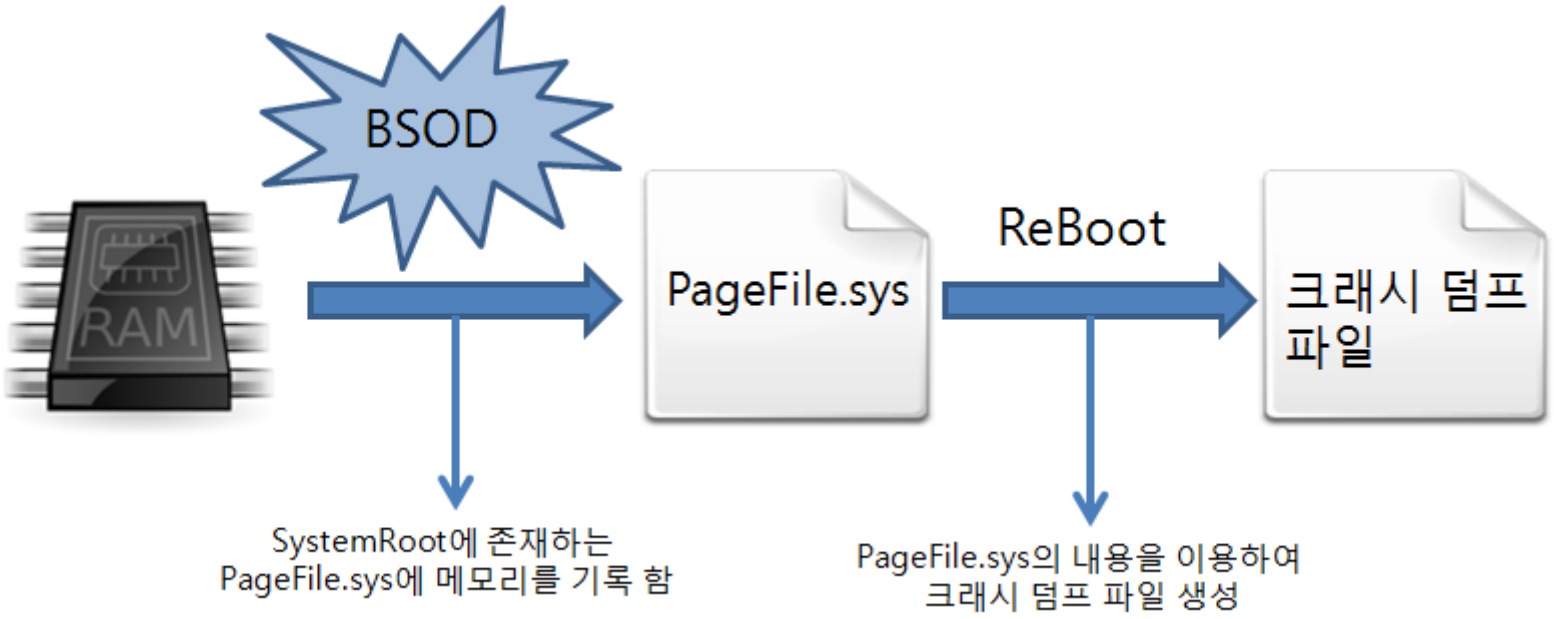
- **FastDump Pro**
  - ✓ HBGary에서 개발한 상용 메모리 덤프 도구
  - ✓ 무료 버전이 존재하지만 64bit와 일부 운영체제 지원 X
  - ✓ 이미징 속도가 매우 빠르고 대용량 메모리 이미징도 지원



FDPro.exe <dump file path> -probe all

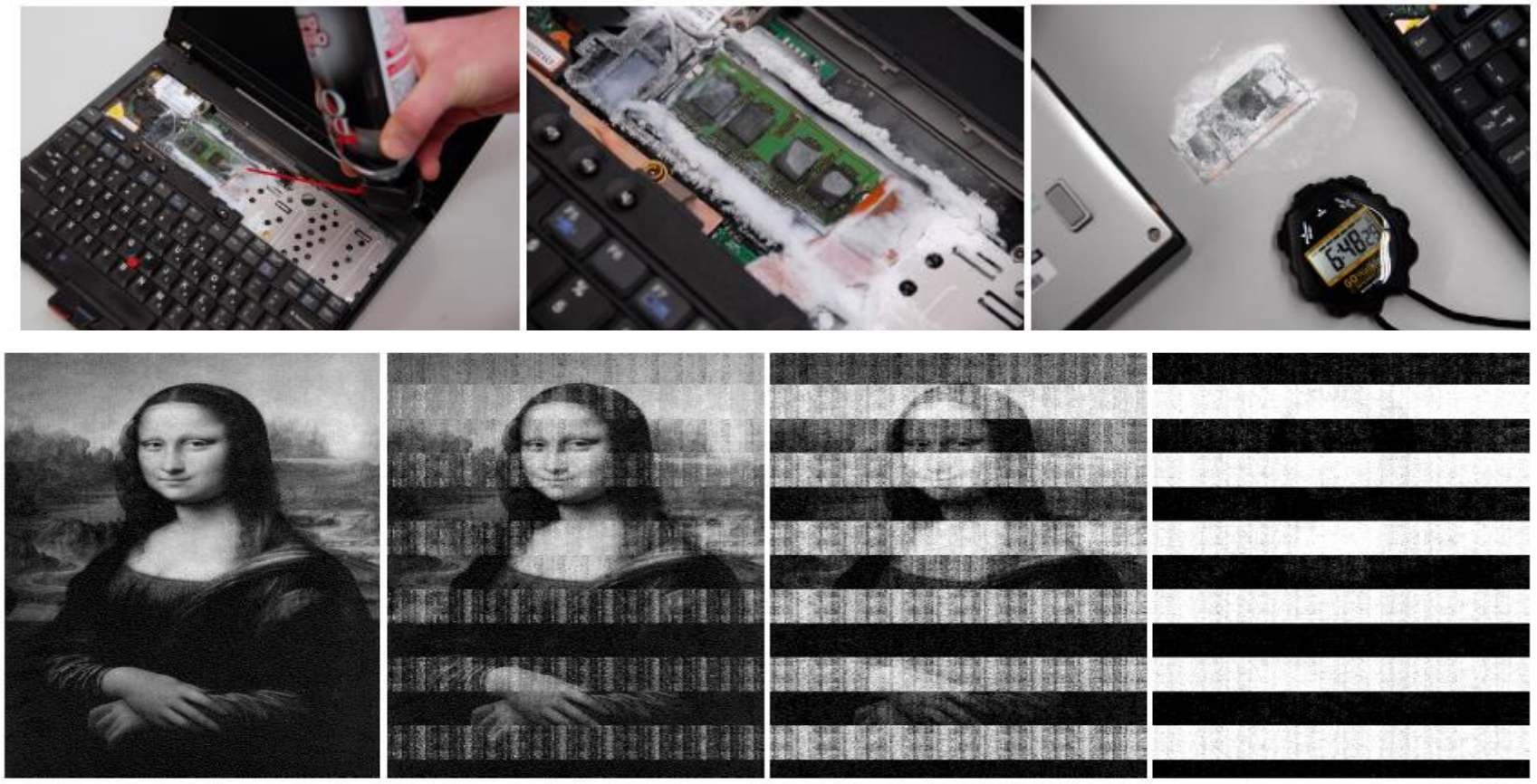
- 크래시 덤프
  - ✓ Windows 운영체제에서 프로세스가 오류나 에러를 일으킬 때 생성되는 에러해결용 메모리 덤프
  - ✓ 리눅스와 비교한다면 Core Dump
  - ✓ 작은 메모리 덤프, 커널 메모리 덤프, 전체 메모리 덤프
  - ✓ BSOD(Blue Screen Of Death) 발생 시 자동으로 생성
  - ✓ Windbg, Kernel Memory Space Analyzer 등을 통해 디버깅
  - ✓ 물리메모리에 영향을 최소화 할 수 있는 덤프 방법

- 크래시 덤프 생성
  - ✓ "[시스템] -> [고급] -> [시작 및 복구] -> [설정]" 에서 설정
  - ✓ Windows 7에서는 전체 메모리 덤프가 사라짐
  - ✓ Windows 8에서는 자동 메모리 덤프 항목이 생김
  - ✓ 수동 크래시를 발생시키려면 시스템 재부팅 필요



- **크래시 덤프 설정**
  - ✓ **HKLM₩SYSTEM₩CurrentControlSet₩Control₩CrashControl₩CrashDumpEnabled**
    - 0 : None
    - 1 : Complete memory dump
    - 2 : Kernel memory dump
    - 3 : Small memory dump
  - ✓ **수동 크래시**
    - 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

- 콜드부트 덤프
  - ✓ 종료된 시스템 메모리를 차갑게 하여 메모리 데이터의 손실 시간을 최대한으로 확보하는 방법



출처 : Lest We Remember: Cold Boot Attacks on Encryption Keys

- 가상화 시스템 덤프
  - ✓ 가상화 시스템은 시작, 일시 정지, 중지가 가능
  - ✓ 가상화 시스템 일시 정지 시 현재 상태의 메모리가 이미지 파일로 생성
  - ✓ Hyper Visor, Cloud System, Vmware, VirtualBox, …

- 절전 덤프
  - ✓ 절전 모드로 진입 시 C:₩hiberfil.sys 파일로 메모리 상태가 저장
  - ✓ NTLDR에 의해 부팅 과정에서 메모리로 로드 되면 이전 상태를 복원
  - ✓ 노트북과 같은 환경은 기본 활성화
  - ✓ 조사 시 강제로 절전모드를 실행 하여 메모리 이미지 획득
  - ✓ 추가적인 하드웨어/소프트웨어 불필요
  - ✓ 단, 전체 메모리가 아닌 현재 사용 중인 영역만 저장

- 메모리 덤프 추세
  - ✓ 아직까지 하드웨어 덤프는 고려사항이 많아 실무적으로 적용하기 힘듦
  - ✓ 소프트웨어가 대세
  - ✓ 크래시 덤프, 절전 모드 덤프는 제한 된 환경에서만 가능

- 메모리 덤프 고려/주의사항
  - ✓ 가급적 메모리 덤프 이미지는 외장장치에 저장
  - ✓ 외장저장장치 인터페이스의 속도 차이 파악

- 실습 #1
  - ✓ 메모리 덤프 해보기!

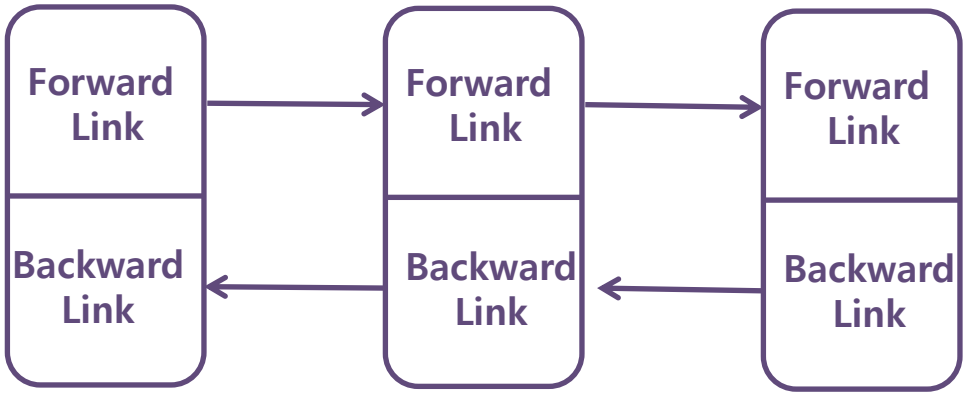
Memory Analysis?!

1. 메모리 분석 방법
2. Redline
3. Volatility

- **초기 메모리 분석**
  - ✓ **문자열 검색**
    - 비밀번호
    - 문서의 특정 단어
    - 파일 이름
    - 기타...
  - ✓ **파일 카빙**
    - 그림 파일
    - HTML 파일
    - 레지스트리
    - 기타...

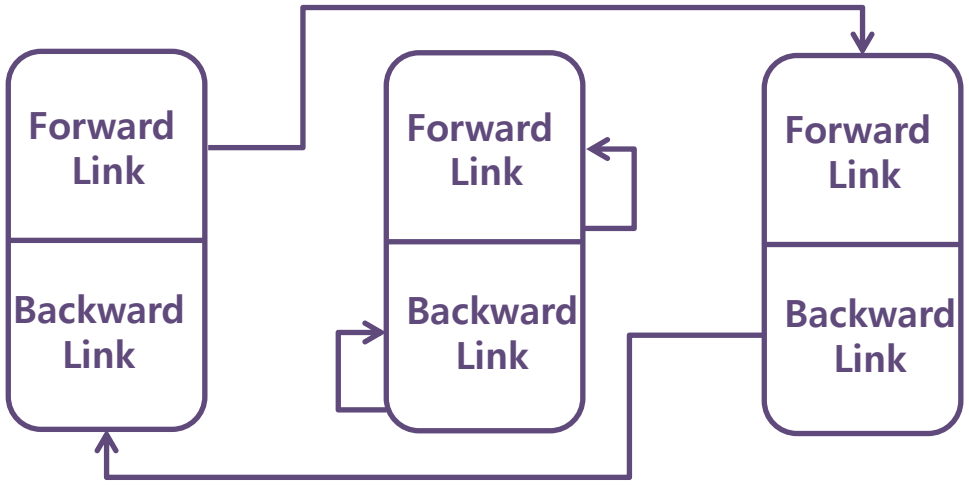
- 메모리 분석 방법
  - ✓ 물리 메모리 내부의 오브젝트 찾는 방법
    - 리스트 워킹(List Walking)
    - 패턴 매칭(Pattern Matching)
    - PspCidTable Walking
    - Process Handle Table Link
    - ETHREAD::ThreadsProcess Pointing
  - ✓ 실시간 메모리 분석
    - VMI(Virtual Machine Introspection)

- **리스트 워킹(List Walking)**
  - ✓ **EPROCESS 구조체를 이용한 방법**
    - EPORCESS 구조체 내부의 ActiveProcessLinks 멤버의 값을 이용하여 프로세스 구조를 파악



  - ✓ **KPCR(Kernel Processor Control Region)을 이용한 방법**
    - KPCR은 운영체제 별로 가상 메모리에서의 위치(XP, 2003-0xFFDFF000, KPCRB-0xFFDFF120)가 고정되어 있음
    - KPCR에는 ERPOCESS의 KTHREAD 구조체의 주소 값이 저장 되어 있음
    - KTHREAD를 이용해 EPROCESS의 위치 확인 가능
  - ✓ **위 두 방법 모두 DKOM 방식이 적용 된 프로세스 은닉 탐지 불가!**

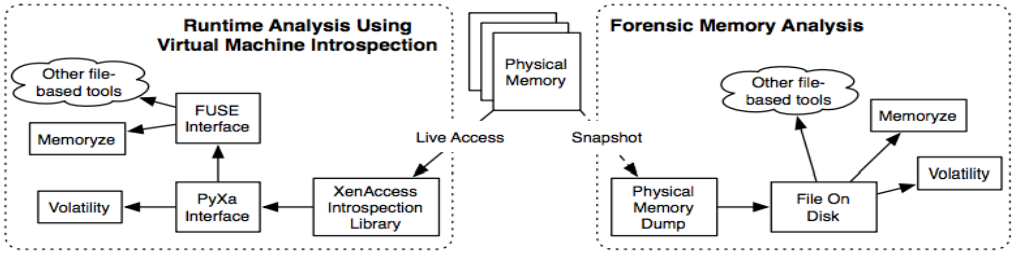
- DKOM(Direct Kernel Object Manipulation)
  - ✓ 전체 Process 구조에서 프로세스의 오브젝트를 구조에서 제외시켜 프로세스 탐지가 되지 않도록 하는 은닉 기법



- **패턴 매칭(Pattern Matching)**
  - ✓ **리스트 워킹 방식의 한계(프로세스 은닉 탐지)를 해결하기 위한 방법**
  - ✓ **프로세스의 구조체는 모두 동일하다는 아이디어**
  - ✓ **프로세스의 구조체를 메모리 전체에서 검색**
  - ✓ **프로세스의 구조체 조건**
    - 프로세스와 스레드는 오브젝트로 존재, 모든 오브젝트는 OBJECT_HEADER를 포함
    - 프로세스와 스레드는 동기화가 필요, 하부 구조체로 DISPATCHER_HEADER를 포함
    - 프로세스와 스레드는 중요 오브젝트, Page Pool이 아닌 Non-Paged Pool이므로 POOL_HEADER를 포함

참고 : Searching for processes and threads in Microsoft Windows Memory dumps

- VMI(Virtual Machine Introspection)
  - ✓ 실시간으로 메모리를 모니터링하며 분석하는 기법
  - ✓ XenLib를 통해 GuestOS의 메모리를 HostOS로 넘겨주어 실시간으로 분석
  - ✓ Cloud 서버의 메모리 또는 대용량 메모리 분석 가능 기대
  - ✓ 가상 허니넷, 동적 악성코드 분석에서 사용 중

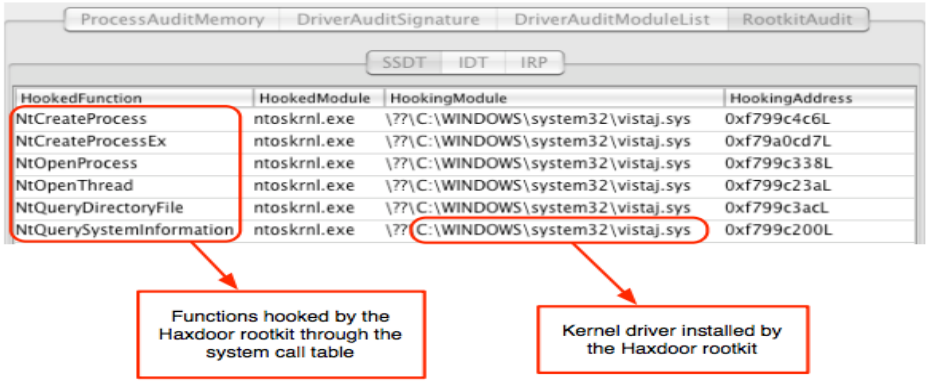



```
PyXa-Volatility# python volatility connections -x 1
Local Address             Remote Address            Pid
xx.xx.xx.xx:1066          208.73.210.121:80         1748

PyXa-Volatility# python volatility pslist -x 1
Name                 Pid    PPid   Thds   Hnds   Time
System               4      0      49     238    Thu Jan 01 00:00:00 1970
smss.exe             436    4      3               51:26 2009
csrss.exe            484    436    10              51:27 2009
winlogon.exe         508    436    16              51:27 2009
services.exe         552    508    15              51:28 2009
lsass.exe            564    508    20     326      Wed Apr 29 18:51:28 2009
svchost.exe          716    552    16     173      Wed Apr 29 18:51:28 2009
svchost.exe          772    552    10     227      Wed Apr 29 18:51:28 2009
svchost.exe          832    552    65     1342     Wed Apr 29 18:51:28 2009
svchost.exe          892    552    6      76       Wed Apr 29 18:51:28 2009
svchost.exe          98                            29 18:51:28 2009
spoolsv.exe          12                            29 18:51:29 2009
alg.exe              17                            29 18:51:38 2009
explorer.exe         19                            29 18:51:45 2009
ctfmon.exe           2024   1916   1      69       Wed Apr 29 18:51:45 2009
iexplore.exe         724    1916   7      359      Wed Apr 29 15:01:20 2009
SYS_INF03.exe        1408   1424   4      185      Wed Apr 29 15:08:52 2009
SYS_INF03.exe        1748   1660   5      174      Wed Apr 29 15:11:41 2009
```

PID for the malicious click fraud traffic

Process information associated with the PID found above

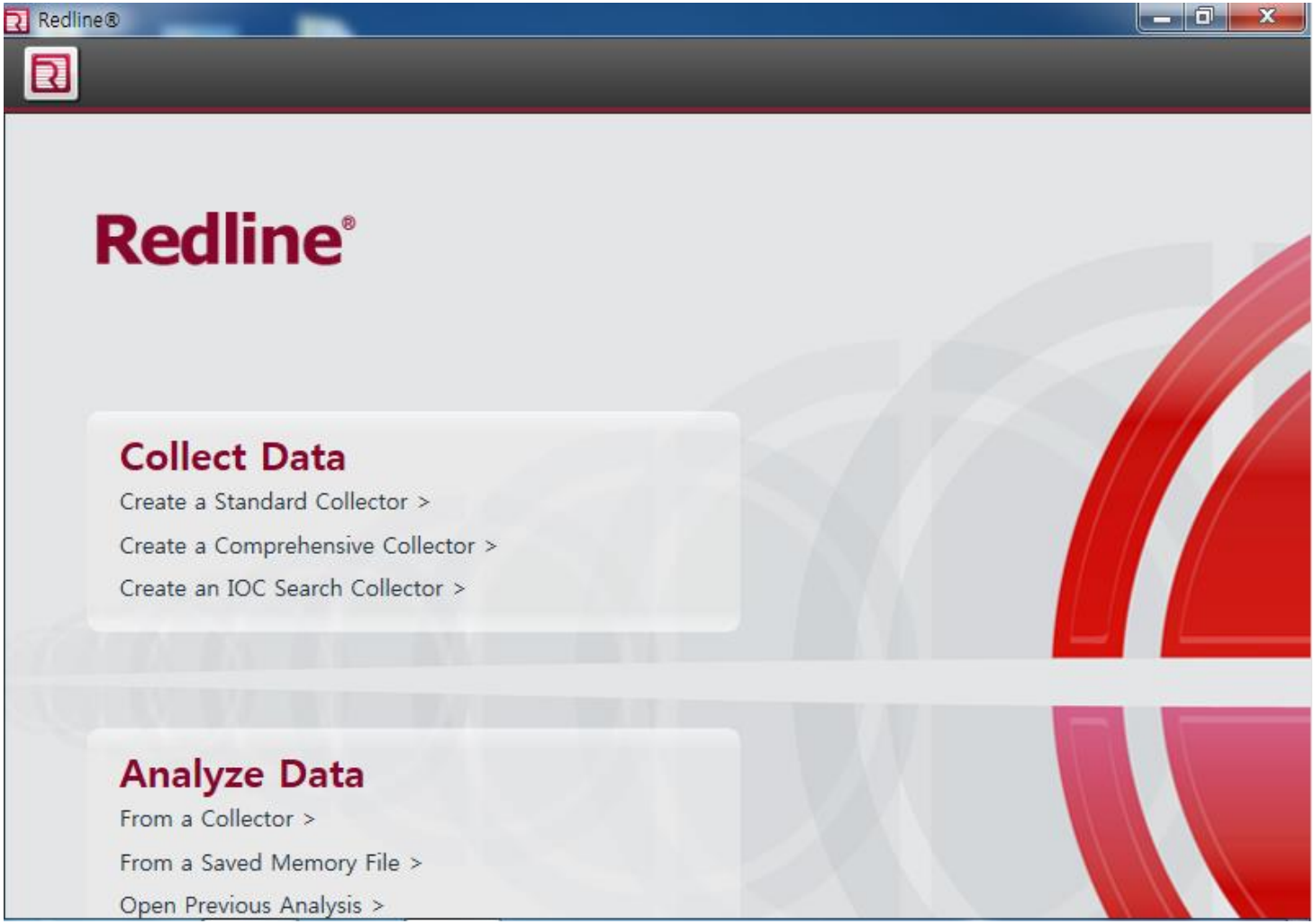


참고 : Leveraging Forensic Tools for Virtual Machine Introspection

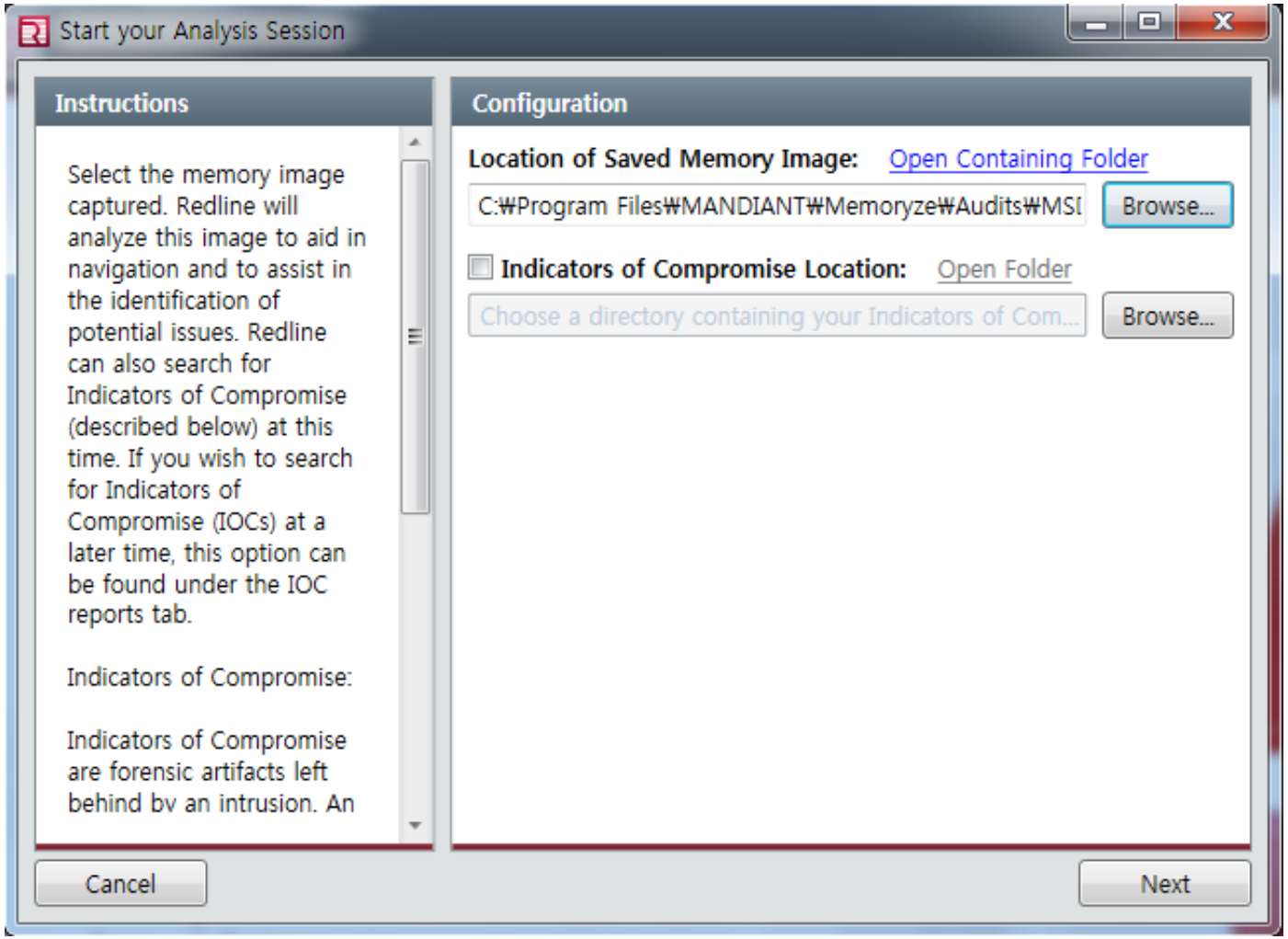
- 분석 도구

이름	인터페이스	플랫폼	제조사	라이선스
Redline	GUI	Windows	Mandiant	Freeware
Volatility	CLI/GUI	All	Volatile Systems	Opensource
Responder Pro	GUI	Windows	HBGray	Commercial
Second Lock® Linux Memory Analysis	CLI	Linux	Raytheon Pikewerks	Commercial
Volafox	CLI	Mac OS	N0fate	Opensource
Volafunx	CLI	FreeBSD	N0fate	Opensource
ReKall	CLI	All	Google	Opensource

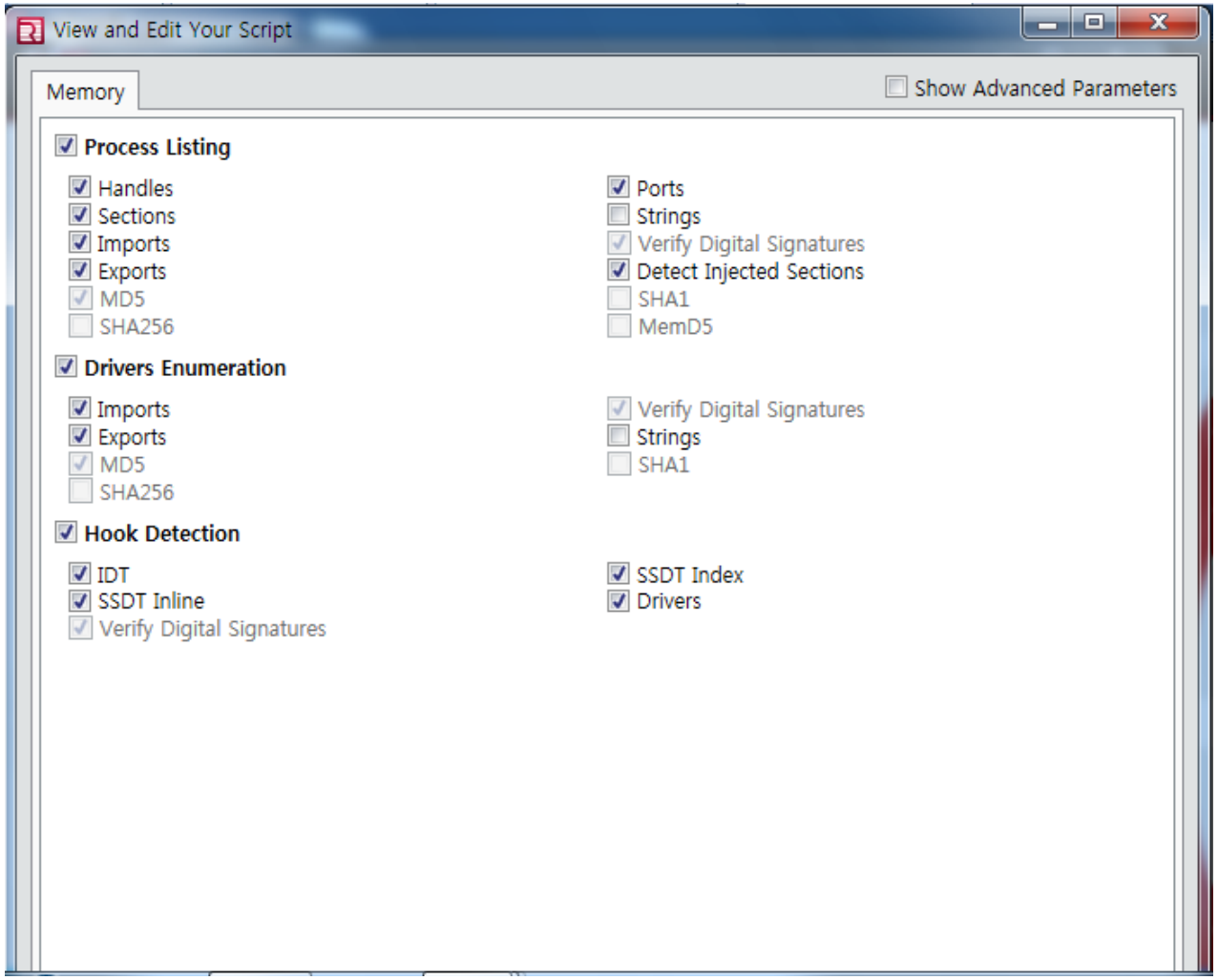
- Redline


Redline®
Redline®
Collect Data
Create a Standard Collector >
Create a Comprehensive Collector >
Create an IOC Search Collector >
Analyze Data
From a Collector >
From a Saved Memory File >
Open Previous Analysis >

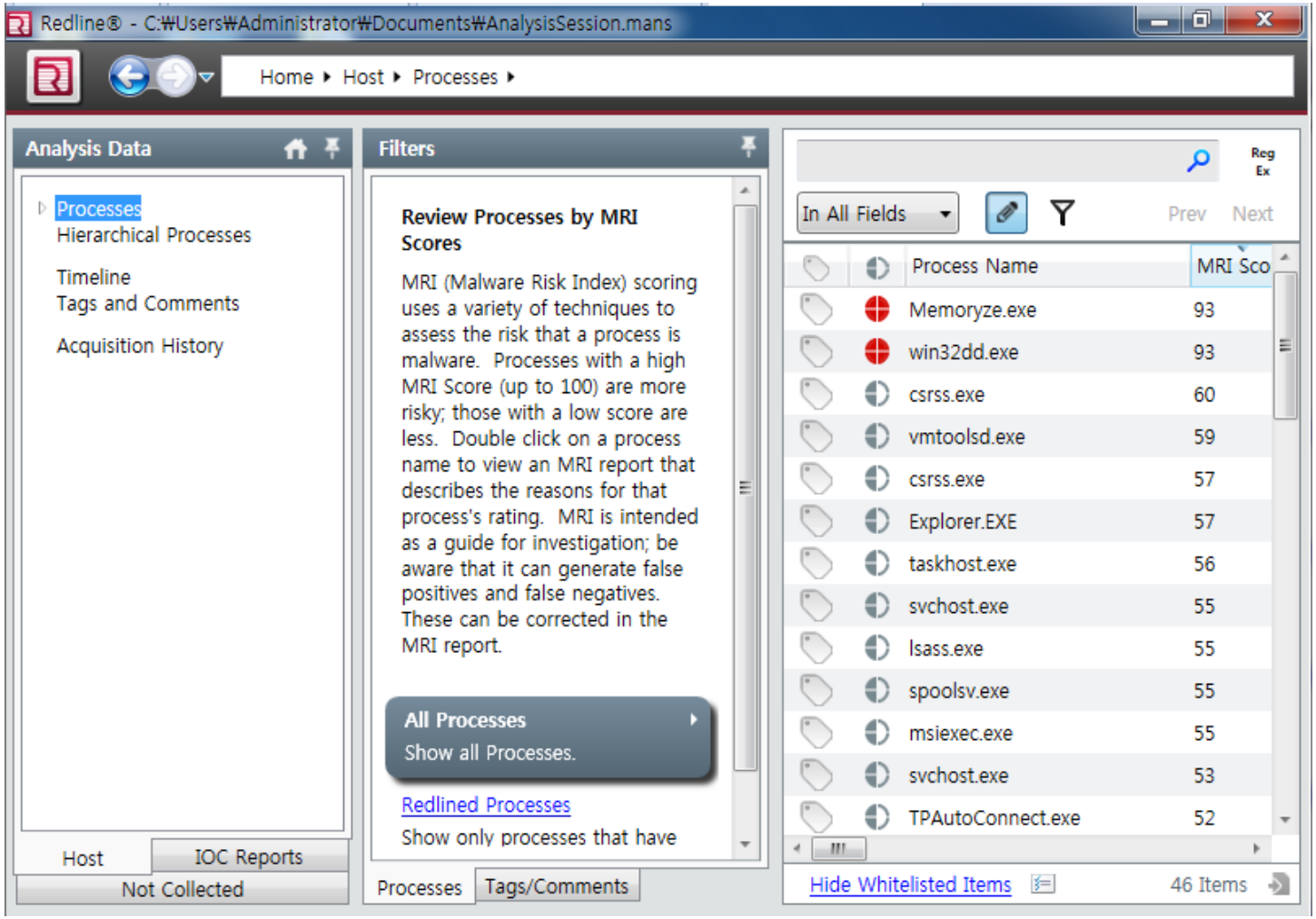
- Redline


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
Configuration
Location of Saved Memory Image:
Open Containing Folder
C:₩Program Files₩MANDIANT₩Memoryze₩Audits₩MS[
Browse...
Indicators of Compromise Location:
Open Folder
Choose a directory containing your Indicators of Com...
Browse...
Cancel
Next

- Redline


View and Edit Your Script
Memory
Show Advanced Parameters
Process Listing
Handles
Sections
Imports
Exports
MD5
SHA256
Ports
Strings
Verify Digital Signatures
Detect Injected Sections
SHA1
MemD5
Drivers Enumeration
Imports
Exports
MD5
SHA256
Verify Digital Signatures
Strings
SHA1
Hook Detection
IDT
SSDT Inline
Verify Digital Signatures
SSDT Index
Drivers

- Redline


Redline® - C:₩Users₩Administrator₩Documents₩AnalysisSession.mans
Home ▸ Host ▸ Processes ▸
Analysis Data
Processes
Hierarchical Processes
Timeline
Tags and Comments
Acquisition History
Host
IOC Reports
Not Collected
Filters
Review Processes by MRI Scores
MRI (Malware Risk Index) scoring uses a variety of techniques to assess the risk that a process is malware. Processes with a high MRI Score (up to 100) are more risky; those with a low score are less. Double click on a process name to view an MRI report that describes the reasons for that process's rating. MRI is intended as a guide for investigation; be aware that it can generate false positives and false negatives. These can be corrected in the MRI report.
All Processes
Show all Processes.
Redlined Processes
Show only processes that have
Processes
Tags/Comments
Reg Ex
In All Fields
Prev
Next
Process Name
MRI Sco
Memoryze.exe 93
win32dd.exe 93
csrss.exe 60
vmtoolsd.exe 59
csrss.exe 57
Explorer.EXE 57
taskhost.exe 56
svchost.exe 55
lsass.exe 55
spoolsv.exe 55
msiexec.exe 55
svchost.exe 53
TPAutoConnect.exe 52
Hide Whitelisted Items
46 Items

- **Volatility 준비사항**
  - ✓ **Tool 설치**
    - https://github.com/volatilityfoundation/volatility
  - ✓ **Volatility Plugin List 파악**
    - https://github.com/volatilityfoundation/volatility/wiki
  - ✓ **Memory Forensics Cheat Sheet Paper 참고**
    - https://blogs.sans.org/computer-forensics/files/2012/04/Memory-Forensics-Cheat-Sheet-v1_2.pdf
  - ✓ **Python 설치**
    - https://www.python.org/download/releases/2.7.3
  - ✓ **PyCrypto 설치**
    - http://www.voidspace.org.uk/python/modules.shtml#pycrypto
  - ✓ **pyPIL 설치**
    - http://www.pythonware.com/products/pil/
  - ✓ **Distorm3 설치**
    - https://pypi.python.org/pypi/distorm3

- **Volatility 사용법**
  - ✓ **이미지 프로파일 확인**
    - $> vol.py –f <image path> imageinfo
  - ✓ **플러그인 목록 확인**
    - $> vol.py <--info|-h>
  - ✓ **플러그인 옵션 확인**
    - $> vol.py <plugin name> --help
  - ✓ **외부 플러그인 로드**
    - $> vol.py –plugins=<plugin path> <plugin name>

- **Volatility 사용법**

  ✓ **프로세스 목록 확인**

  - $> vol.py –f <image path> --profile=<profile> pslist
  - $> vol.py –f <image path> --profile=<profile> psscan
  - $> vol.py –f <image path> --profile=<profile> pstree
  - $> vol.py –f <image path> --profile=<profile> pxsview
  - $> vol.py –f <image path> --profile=<profile> psdispscan

  ✓ **프로세스 관련 정보 확인**

  - $> vol.py –f <image path> --profile=<profile> dlllist
  - $> vol.py –f <image path> --profile=<profile> vadinfo
  - $> vol.py –f <image path> --profile=<profile> handles
  - $> vol.py –f <image path> --profile=<profile> privs
  - $> vol.py –f <image path> --profile=<profile> threads

- **Volatility 사용법**

  ✓ **네트워크 정보 확인**

  - $> vol.py –f <image path> --profile=<XP|2003 profile> connections
  - $> vol.py –f <image path> --profile=<XP|2003 profile> sockets
  - $> vol.py –f <image path> --profile=<XP|2003 profile> connscan
  - $> vol.py –f <image path> --profile=<Vista|2008|7 profile> netscan

  ✓ **PE 관련 정보 추출**

  - $> vol.py –f <image path> --profile=<profile> -D [dir path] moddump
  - $> vol.py –f <image path> --profile=<profile> -D [dir path] procexedump
  - $> vol.py –f <image path> --profile=<profile> -D [dir path] procmemdump
  - $> vol.py –f <image path> --profile=<profile> -D [dir path] dlldump

- **Volatility 사용법**

✓ **인젝션 행위 탐지**

- $> vol.py –f <image path> --profile=<profile> -D [dir] malfind
- $> vol.py –f <image path> --profile=<profile> ldrmodules
- $> vol.py –f <image path> --profile=<profile> impscan

✓ **기타 정보 추출**

- $> vol.py –f <image path> --profile=<profile> cmdscan
- $> vol.py –f <image path> --profile=<profile> consoles
- $> vol.py –f <image path> --profile=<profile> svcscan

✓ **SID 확인**

- $> vol.py –f <image path> --profile=<profile> getsids

✓ **환경변수 확인**

- $> vol.py –f <image path> --profile=<profile> envars

- **Volatility 사용법**
  - ✓ **커널 관련 정보 확인**
    - $> vol.py –f <image path> --profile=<profile> modules
    - $> vol.py –f <image path> --profile=<profile> modscan
    - $> vol.py –f <image path> --profile=<profile> timers
    - $> vol.py –f <image path> --profile=<profile> callbacks
    - $> vol.py –f <image path> --profile=<profile> ssdt
    - $> vol.py –f <image path> --profile=<profile> idt
    - $> vol.py –f <image path> --profile=<profile> gdt
    - $> vol.py –f <image path> --profile=<profile> devicetree

- **Volatility 사용법**
  - ✓ **커널 오브젝트 확인**
    - $> vol.py –f <image path> --profile=<profile> driverscan
    - $> vol.py –f <image path> --profile=<profile> mutantscan
    - $> vol.py –f <image path> --profile=<profile> filescan
    - $> vol.py –f <image path> --profile=<profile> symlinkscan

- **Volatility 사용법**

  ✓ **레지스트리 정보 확인**

  - $> vol.py –f <image path> --profile=<profile> hivelist
  - $> vol.py –f <image path> --profile=<profile> printkey
  - $> vol.py –f <image path> --profile=<profile> userassit
  - $> vol.py –f <image path> --profile=<profile> shellbags
  - $> vol.py –f <image path> --profile=<profile> shimcache

  ✓ **패스워드 관련 정보 확인**

  - $> vol.py –f <image path> --profile=<profile> lsadump
  - $> vol.py –f <image path> --profile=<profile> hashdump
  - $> vol.py –f <image path> --profile=<profile> dumpcerts

- 실습 #1
  - ✓ 사용자 계정 추출
    - 문자열 추출
    - Volatility Plugin을 이용하여 추출

- 실습 #2
  - ✓ 악성코드 언패킹
    - 악성코드의 본래 코드를 획득해보자!!
    - Volatility Plugin을 이용하여 추출

Anti Memory Forensics?!

1. Anti Memory Dumping
2. ADD(Attention Deficit Disorder)

- **Anti Memory Dumping**
  - ✓ 메모리 덤프 도구의 실행을 방해하여 메모리 덤프가 수행되지 못하도록 하는 방법
  - ✓ 악성코드, 패커, 백신, 보안 프로그램에서 많이 사용
  - ✓ 종류
    - Nanomites
    - Stolen Bytes(code Splicing)
    - Self-Unmapping
    - 기타..
  - ✓ 하드웨어 메모리 덤프 도구 사용 또는 프로세스 무력화
  - ✓ 커널 레벨에서의 덤프
    - Belkasoft Live RAM Capturer

- ADD(Attention Deficit Disorder)
  - ✓ Shmoocon 2014에서 발표 된 안티 메모리 포렌식 기법
  - ✓ 허위 오브젝트 구조체를 만들어 메모리에 삽입하는 방법
  - ✓ 현재까지 프로세스, 네트워크, 파일만 구현
  - ✓ EPROCES 구조체, 문자열 검색 등으로 안티 포렌식 행위 파악 가능

출처 : https://code.google.com/p/attention-deficit-disorder/

- Memory Clearing or Encryption
  - ✓ 메모리의 데이터를 보호하는 목적으로 연구 된 기술
  - ✓ 메모리의 데이터를 강제적으로 지우거나 암호화 함
  - ✓ 대표적인 예, TrueCrypt
  - ✓ 암호화는 현재 연구가 진행되고 있으나 실제로 적용 된 바는 없음

- 실습 #1
  - ✓ ADD 기법이 적용 된 메모리 이미지 분석하기!
  - ✓ 답 제출 : https://www.surveymonkey.com/s/Q2738W6
  - ✓ Hint : 문자열 검색

Memory Forensics Challenge!

1. Honeynet 2010 Forensic Challenge 5
2. SANS DFIR Challenge
3. Nuit du hack 2011 100
4. DC3 2013 303

- **챌린지 #1**
  - ✓ **Honeynet Forensic Challenge - Banking Troubles**
  - ✓ **문제 요약**
    - X사는 자신들의 회사에서 발생한 최근 사건에 대해 우리에게 의뢰를 하였다. 그 사건은 은행 계좌에서 비정상적인 활동들이 발견 된 사건이었다.
    - 포렌식 작업을 수행하던 중 우리는 직원 중 하나가 PDF 파일이 첨부된 메일을 동료 직원으로부터 전달 받는 것을 확인 할 수 있었다.
    - X사는 악성코드 감염이 의심되는 직원의 컴퓨터 메모리 이미지를 분석 해 의심되는 행동을 보고 싶어한다.
  - ✓ **답 제출 : https://www.surveymonkey.com/s/QQTMFLY**

출처 : https://www.honeynet.org/challenges/2010_3_banking_troubles

- 챌린지 #2
  - ✓ SANS DFIR Challenge – APT Attack Analysis
  - ✓ 답 제출 : https://www.surveymonkey.com/s/QX7ZK5D

- 챌린지 #3
  - ✓ Nuit du hack 2011 100 – 서버로 전송 된 데이터를 찾아라.

- **챌린지 #4**

  ✓ **DC3 Windows Memory Image Analysis.**

  - 한 회사는 직원들의 컴퓨터를 모니터링 중 한 직원의 컴퓨터에서 게임 인스턴스 메시지가 계속해서 발생하는걸 발견하였다.
  - 회사는 직원을 근무태만으로 고발하였고, 감사팀은 해당 직원의 컴퓨터를 압수하여 분석하기 시작했다.
  - 직원 컴퓨터에 관한 보고서를 써야 하는데, 압수 당시의 상황을 파악하지 못하여 난감해 하고 있다.

  ✓ **답 제출 : https://www.surveymonkey.com/s/WWKN7JX**