

# 6주차 스터디

파일 카빙, 메모리 포렌식

# 파일 카빙 (File Carving)

- 저장 매체가 포맷되었거나 손상된 파일들을 추출, 복구하는 작업<sup>12</sup>
- 일반적으로 포맷마다 고유한 header/footer 시그니처 및 데이터 구조 기반으로 카빙한다.
- 파일을 삭제한다고 디스크에 파일이 삭제되지 않는다.
  - FAT32 는 디렉토리 엔트리의 첫 바이트가 0xE5 로 설정되며, NTFS 는 MFT 엔트리의 플래그 값이 파일인 경우 0x00 , 폴더인 경우 0x02 로 설정된다.
- 포맷을 했더라도 경우에 따라서 일부 복구가 가능하다. (ex: 비활당 영역)
  - 그래서 CCleaner와 같은 삭제 도구를 보면 ?-pass 라는 표현을 쓰는데, 몇 번 덮어쓰기를 할 것인지 선택하는 옵션이다.<sup>345</sup>

---

<sup>1</sup> 시나리오 기반 이미지 개발을 통한 파일 카빙 도구 검증 방안 연구

<sup>2</sup> [https://forensics.wiki/file\\_carving/](https://forensics.wiki/file_carving/)

<sup>3</sup> [https://en.wikipedia.org/wiki/Data\\_erosure](https://en.wikipedia.org/wiki/Data_erosure)

<sup>4</sup> All about Physical Data Recovery

<sup>5</sup> 나무위키 - 소거 프로그램

# bulk\_extractor

- 윈도우 관련 아티팩트, 도메인 주소, 압축 파일, 신용카드 번호 외에도 다양한 데이터들을 카빙해준다.<sup>1</sup>
- 파일 시스템 구조를 분석하지 않고, 처음부터 끝까지 스캔하여 탐지되는 모든 데이터를 저장한다.<sup>2</sup>
- 병렬로 처리하기 때문에 보유한 자원들이 놀지 않는다.<sup>3</sup>
  - 1.6 버전에서 대부분 싱글 쓰레드로 동작했었고 병목 현상이 존재하여 다시 설계하여 만들었다고 한다.
- 이전 버전보다 더 많은 파일들을 카빙할 수 있다고 한다.<sup>4</sup>

THE GEORGE WASHINGTON UNIVERSITY  
WASHINGTON, DC

BE1 vs. BE2:  
BE2 is finding a lot of stuff that BE1 missed

Size ✓  
Compile-time (relevant for development) ✓  
Runtime ✓  
Analysis

file	BE16	BE2.0 Beta 4
alerts.txt	62	19
domain.txt	72,027	76,800
email.txt	8,757	8,751
ether.txt	5	1
ether_histogram_1.txt	n/a	0
exif.txt	232	235
facebook.txt	n/a	0
ip.txt	4	4,444
jpeg_carved.txt	43	1,767
json.txt	4	958
kml.txt	0	2
ntfsusn_carved.txt	2	1
rfc822.txt	4,240	4,219
tcp.txt	n/a	56
tcp_histogram.txt	n/a	0
telephone.txt	767	760
unzip_carved.txt	41	n/a
url.txt	108,352	112,754
winpe.txt	10,740	10,592
winpe_carved.txt	4	10,573
winprefetch.txt	124	0
zip.txt	5,196	10,193

1,724 additional JPEGs carved

10,569 windows executables carved!

<sup>1</sup> [https://github.com/simsong/bulk\\_extractor/wiki](https://github.com/simsong/bulk_extractor/wiki)

<sup>2</sup> <https://youtu.be/odvDTGA7rYI?t=1060> - Forensic Carving of Network Packets with bulk\_extractor and tcpflow

<sup>3</sup> <https://youtu.be/RHCzrmvul4Q?t=270> - A case study of updating bulk\_extractor 1.6 to 2.0

<sup>4</sup> <https://youtu.be/RHCzrmvul4Q?t=820> - A case study of updating bulk\_extractor 1.6 to 2.0

<sup>5</sup> [https://github.com/simsong/bulk\\_extractor/blob/main/doc/ROADMAP\\_2.0.md](https://github.com/simsong/bulk_extractor/blob/main/doc/ROADMAP_2.0.md)

<sup>6</sup> [https://github.com/simsong/bulk\\_extractor/releases/tag/v2.1.0](https://github.com/simsong/bulk_extractor/releases/tag/v2.1.0)

# bulk\_extractor

- 윈도우 버전은 2024. 02. 08 기준 구버전인 1.5.0 과 2.0 을 사용할 수 있다.<sup>1</sup>
  - 1.5.3 - 2014. 08. 29 릴리즈, 2.0 - 2022. 02. 13 릴리즈, 2.1 - 2024. 01. 22 릴리즈<sup>2</sup>

## Windows Users

- Version 1.5.0 graphical installer with Windows GUI:  
[https://digitalcorpora.s3.amazonaws.com/downloads/bulk\\_extractor/bulk\\_extractor-1.5.0-windowsinstaller.exe](https://digitalcorpora.s3.amazonaws.com/downloads/bulk_extractor/bulk_extractor-1.5.0-windowsinstaller.exe)
- Version 2.0 command-line EXE: [https://digitalcorpora.s3.amazonaws.com/downloads/bulk\\_extractor/bulk\\_extractor-2.0.0-windows.zip](https://digitalcorpora.s3.amazonaws.com/downloads/bulk_extractor/bulk_extractor-2.0.0-windows.zip)

- 버전업할 때마다 exe 파일을 배포해주지 않기 때문에 현재 stable하며, 최신 버전인 2.1 을 사용하기 위해서는 리눅스 환경에서 빌드를 진행해야 한다.

Bulk\_extractor version 2.1 is the first stable version of bulk\_extractor version 2 that is recommended for general use. I scanner that caused bulk\_extractor to hang on open-ended regular expressions such as [a-z]\*@company.com specific

<sup>1</sup> [https://github.com/simsong/bulk\\_extractor/wiki/Installing-bulk\\_extractor#windows-users](https://github.com/simsong/bulk_extractor/wiki/Installing-bulk_extractor#windows-users)

<sup>2</sup> [https://github.com/simsong/bulk\\_extractor/releases](https://github.com/simsong/bulk_extractor/releases)

# bulk\_extractor

- 빌드하기 전에 필요한 패키지들을 설치한다. - `etc` 폴더에서 환경에 맞는 bash 스크립트 실행<sup>1</sup>

```
→ bulk_extractor-2.1.0 cd etc
→ etc ls
CONFIGURE_AMAZON_LINUX.bash    CONFIGURE_FEDORA36_win64.bash    CONFIGURE_UBUNTU22_win64.bash
CONFIGURE_CENTOS7.bash          CONFIGURE_MACOS.bash          makefile_builder.py
CONFIGURE_DEBIAN9.bash          CONFIGURE_PENTO06.2.8.bash  paths.bash
CONFIGURE_FEDORA34.bash         CONFIGURE_UBUNTU20LTS.bash   whats-missing-from-dist.py
CONFIGURE_FEDORA34_win64.bash   CONFIGURE_UBUNTU20_win64.bash
CONFIGURE_FEDORA36.bash          CONFIGURE_UBUNTU22LTS.bash
+ etc ./CONFIGURE_UBUNTU22LTS.bash|
```

- `wget https://github.com/simsong/bulk_extractor/releases/download/v2.1.0/bulk_extractor-`

`2.1.0.tar.gz`<sup>2</sup> → manual 참고하여 설치<sup>2</sup>

2. Download the latest version of *bulk\_extractor*. It can be obtained from [http://digitalcorpora.org/downloads/bulk\\_extractor/](http://digitalcorpora.org/downloads/bulk_extractor/). The file is called `bulk_extractor-x.y.z.tar.gz`, where x.y.z is the latest version.

3. Un-tar and un-zip the file. In the newly created `bulk_extractor-x.y` directory, run the following commands:

- `./configure`
- `make`
- `sudo make install`

<sup>1</sup> [https://github.com/simsong/bulk\\_extractor/wiki/Installing-bulk\\_extractor#linux-and-os-x-users](https://github.com/simsong/bulk_extractor/wiki/Installing-bulk_extractor#linux-and-os-x-users)

<sup>2</sup> [https://github.com/simsong/bulk\\_extractor/releases/tag/v2.1.0](https://github.com/simsong/bulk_extractor/releases/tag/v2.1.0)

<sup>3</sup> [http://digitalcorpora.org/downloads/bulk\\_extractor/BEUsersManual.pdf](http://digitalcorpora.org/downloads/bulk_extractor/BEUsersManual.pdf)

# bulk\_extractor

- 설치가 완료되었다면 /usr/local/bin에 bulk\_extractor 파일이 생성되어 사용할 수 있다.

```
→ ~ bulk_extractor
imagefile not provided
bulk_extractor version 2.1.0: A high-performance flexible digital forensics program.
Usage:
    bulk_extractor [OPTION...] image_name
```

- x 는 disable, -e 는 enable이며, 기본으로 활성화된 스캐너들이 많이 있다.
  - 아래 사진은 기본 설정에서 facebook 스캐너를 비활성화, hiberfile 스캐너를 활성화했다.
- 자세한 사용 방법과 스캐너 종류는 bulk\_extractor --help bulk\_extractor --info\_scanners 참고<sup>1</sup>

```
→ ~ bulk_extractor /mnt/c/Users/hyuunnnn/Desktop/240103.E01 -x facebook -e hiberfile -o /mnt/c/Users/hyuunnnn/Desktop/bulk2_1
opening /mnt/c/Users/hyuunnnn/Desktop/240103.E01

bulk_extractor version: 2.1.0
Input file: "/mnt/c/Users/hyuunnnn/Desktop/240103.E01"
Output directory: "/mnt/c/Users/hyuunnnn/Desktop/bulk2_1"
Disk Size: 21474836480
Scanners: aes base64 elf evtx exif find gzip hiberfile httplogs json kml_carved msxml net ntfsindx ntfslogfile ntfsfmt ntfsusn pdf rar sqlite utmp vcard_carved windirs winlnk winpe winprefetch zip accts email gps
Threads: 20
going multi-threaded...( 20 )
bulk_extractor      Thu Feb  8 19:59:36 2024
```

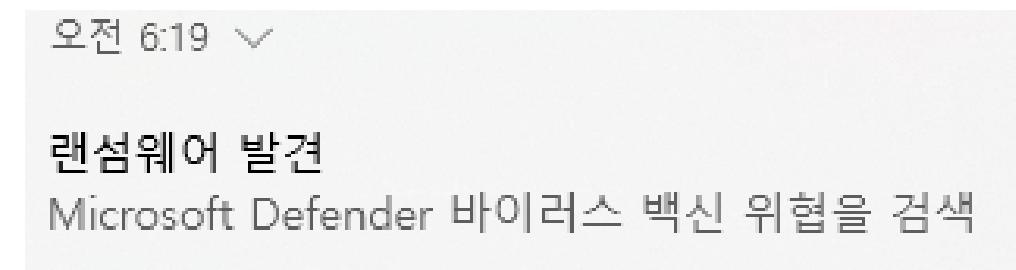
<sup>1</sup> [https://github.com/simsong/bulk\\_extractor/wiki](https://github.com/simsong/bulk_extractor/wiki)

# bulk\_extractor

- 1.5.0, 2.0, 2.1 버전 별로 카빙한 파일 개수와 용량 차이가 있음을 확인할 수 있다.

bulk1_6 속성	bulk2_0 속성	bulk2_1 속성
일반	일반	일반
bulk1_6	bulk2_0	bulk2_1
종류: 파일 폴더	종류: 파일 폴더	종류: 파일 폴더
위치: C:\Users\hyuunnnn\Desktop	위치: C:\Users\hyuunnnn\Desktop	위치: C:\Users\hyuunnnn\Desktop
크기: 1.53GB (1,653,114,382 바이트)	크기: 6.94GB (7,462,613,559 바이트)	크기: 6.94GB (7,462,678,301 바이트)
디스크 할당 크기: 1.53GB (1,653,428,224 바이트)	디스크 할당 크기: 6.98GB (7,497,748,480 바이트)	디스크 할당 크기: 6.98GB (7,497,252,864 바이트)
내용: 파일 354, 폴더 12	내용: 파일 48,261, 폴더 66	내용: 파일 47,783, 폴더 64

- 또한 2.x 버전에서 랜섬웨어 알림을 확인할 수 있었다. → KEEPER CTF 이미지( E01 ) 파일 사용



# bulk\_extractor

- windirs.txt에서 \*.pf(프리패치 파일)로 필터링하여 랜섬웨어 파일과 FTK Imager, WinRAR 등 실행 증거 확인 가능

3456090112	0IAJDNGXORFO.EXE-D7D393C8.pf	<fileobject src='mft'> <atime_fn>2024-01-03T08:18:11Z<
3455918080	ACCESSDATA_FTK_IMAGER_4.7.1.E-2320A33B.pf	<fileobject src='mft'> <atime_fn>2024-01-03T08:20:12Z<
3455926272	ACCESSDATA_FTK_IMAGER_4.7.1.E-E18FB623.pf	<fileobject src='mft'> <atime_fn>2024-01-03T08:20:13Z<
3459501056	APPLICATIONFRAMEHOST.EXE-8CE9A1EE.pf	<fileobject src='mft'> <atime_fn>2024-01-03T07:54:25Z<
3456245760	AUDIODG.EXE-AB22E9A6.pf	<fileobject src='mft'> <atime_fn>2024-01-03T07:36:25Z<

- winlnk.txt에서 lnk 파일 분석 가능

```
<lnk><atime>2024-01-03T08:04:29Z</atime><ctime>2024-01-03T07:58:22Z</ctime><local_base_path>C:\Users\134User\134Downloads\13411\277\371_회\260\350\272\316.rar</local_base_path><w
<lnk><atime>2024-01-03T08:08:40Z</atime><ctime>2021-08-05T22:41:46Z</ctime><local_base_path>C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe</local_base_path><
<lnk><atime>2024-01-03T08:15:03Z</atime><ctime>2024-01-03T07:48:29Z</ctime><local_base_path>C:\Users\134User\134Downloads\007</local_base_path><wtime>2024-01-03T08:08:40Z</wtime></lnk
<lnk><atime>2024-01-03T08:15:03Z</atime><ctime>2024-01-03T07:48:29Z</ctime><local_base_path>C:\Users\134User\134Downloads</local_base_path><wtime>2024-01-03T08:08:40Z</wtime></lnk>
<lnk><atime>2024-01-03T08:15:03Z</atime><ctime>2024-01-03T08:00:29Z</ctime><local_base_path>C:\Users\134User\134Downloads\134x86_x86_64\276\306\330\305\330\302\367\300\314.pdf</local_base_path><wtime>2024-01-03T08:18:23Z</wtime>
<lnk><atime>2024-01-03T08:20:25Z</atime><ctime>2022-01-19T00:02:40Z</ctime><local_base_path>C:\Program Files\134AccessData\134FTK Imager\FTK Imager.exe</local_base_path><wtime>2024-01-03T08:20:25Z</wtime>
```

# bulk\_extractor

- winpe\_carved.txt에서 PE 구조<sup>1</sup>의 파일들을 카빙한 결과 확인 가능
  - Microsoft Defender에서 카빙한 파일 중에 랜섬웨어 존재 확인 가능

The screenshot shows the Microsoft Windows Defender Event Viewer interface. The title bar reads "Microsoft-Windows-Windows Defender%4Operational" with "이벤트 수: 3,766". The main pane displays a table of events with columns: 수준 (Level), 날짜 및 시간 (Date and Time), 원본 (Source), 이벤트 ID (Event ID), and 작업 범주 (Task Category). Three events are listed:

수준	날짜 및 시간	원본	이벤트 ID	작업 범주
경고	2024-02-08 오후 8:05:32	Windows Defender	1116	없음
정보	2024-02-08 오후 8:05:32	Windows Defender	2010	없음
경고	2024-02-08 오후 8:05:32	Windows Defender	1116	없음

Below the table, a specific event is expanded: "이벤트 1116, Windows Defender". The "자세히" (Details) tab is selected. The details pane contains the following information:

Microsoft Defender 바이러스 백신이(가) 맬웨어 또는 기타 사용자 등의 없이 설치된 소프트웨어를 검색했습니다.  
자세한 내용은 다음을 참조하십시오.  
<https://go.microsoft.com/fwlink/?linkid=37020&name=Ransom:MSIL/Ryzerlo.A&threatid=2147707683&enterprise=0>

이름: Ransom:MSIL/Ryzerlo.A  
ID: 2147707683  
심각도: 심각  
범주: 랜섬웨어  
경로: [file:///C:/Users/hyuunnnn/Desktop/bulk2\\_1/winpe\\_carved\016\12287537152.winpe](file:///C:/Users/hyuunnnn/Desktop/bulk2_1/winpe_carved\016\12287537152.winpe); [file:///C:/Users/hyuunnnn/Desktop/bulk2\\_1/winpe\\_carved\016\12362633216.winpe](file:///C:/Users/hyuunnnn/Desktop/bulk2_1/winpe_carved\016\12362633216.winpe)

<sup>1</sup> [https://ko.wikipedia.org/wiki/PE\\_포맷](https://ko.wikipedia.org/wiki/PE_포맷)

# bulk\_extractor

n=8	ecdsa-sha2-nistp256-cert-v01@openssh.com
n=8	ecdsa-sha2-nistp384-cert-v01@openssh.com
n=8	ecdsa-sha2-nistp521-cert-v01@openssh.com
n=8	hy00un_@naver.com
n=8	rsa-sha2-256-cert-v01@openssh.com
n=8	rsa-sha2-512-cert-v01@openssh.com

- email\_histogram.txt에서 로그인할 때 사용했던 이메일 확인 가능

- json.txt에서 11월\_회계부.rar을 WinRAR 프로그램으로 열었던 기록할 수 있었다.
    - 데이터의 형태를 보면 ActivitiesCache.db인 것 같다.
    - 한글이 깨진다면 데이터 → 텍스트/CSV에서 → 65001: 유니코드(UTF-8) → 로드

# bulk\_extractor

- url\_searches.txt 에서 실제로 검색했던 디펜더 비활성화 도구와 WinRAR을 확인할 수 있었다.

n=1	search?q=windows%20defender%20disabl
n=1	search?q=windows%20defender%20disable
n=1	search?q=windows%20defender%20disable%20
n=1	search?q=windows%20defender%20disable%20g
n=1	search?q=windows%20defender%20disable%20git
n=1	search?q=windows%20defender%20disable%20gith
n=1	search?q=windows%20defender%20disable%20githu
n=1	search?q=windows%20defender%20disable%20github
n=1	search?q=windows%20defender%20disable%20github&cp=0&client=desktop-gws-wiz-on-1
n=63	search?q=windows+defender+disable+github
n=1	search?q=winr
n=1	search?q=winra
n=61	search?q=winrar
n=1	search?q=winrar&cp=0&client=desktop-gws-wiz-on-focus-serp&xssi=t&gs_pcrt=3&hl=ko8

- url.txt 에서 WinRAR 을 softsonic 사이트에서 받았음을 확인할 수 있다.
  - 문제 시나리오에서 CVE-2023-38831 취약점 사용 - softsonic 에 있는 최신 버전( 6.22 )은 취약함

# bulk\_extractor

- record를 기준으로 카빙<sup>1</sup>해주는데 대부분의 분석 도구는 MFT 파일 하나만 요구하기 때문에 합쳐준다.

The screenshot shows a Windows-style file explorer interface. The path is displayed as: 바탕 화면 > bulk2\_1 > ntfsmft\_carved > 000. A search bar at the top right contains the text "000 검색". Below the path, there are standard file operations icons: copy, move, delete, and refresh. To the right of these are sorting and viewing options: 정렬 (Sort) and 보기 (View). A "세부 정보" (Detailed Information) button is also present. The main area is a table listing files:

이름	수정한 날짜	유형	크기
21072207872.mft	2024-02-08 오후 8:06	MFT 파일	19KB
21072183296.mft	2024-02-08 오후 8:06	MFT 파일	16KB
20871913472.mft	2024-02-08 오후 8:06	MFT 파일	4KB
3456106496.mft	2024-02-08 오후 8:01	MFT 파일	13,548KB
3439329280.mft	2024-02-08 오후 8:01	MFT 파일	16,384KB
3422552064.mft	2024-02-08 오후 8:01	MFT 파일	16,384KB
3405774848.mft	2024-02-08 오후 8:01	MFT 파일	16,384KB
3388997632.mft	2024-02-08 오후 8:01	MFT 파일	16,384KB
3372220416.mft	2024-02-08 오후 8:01	MFT 파일	16,384KB
3355443200.mft	2024-02-08 오후 8:01	MFT 파일	16,384KB
3343933440.mft	2024-02-08 오후 8:01	MFT 파일	11,240KB
3343908864.mft	2024-02-08 오후 8:01	MFT 파일	16KB
3338058752.mft_corrupted	2024-02-08 오후 8:01	MFT_CORRUPTED...	1KB

<sup>1</sup> [https://www.kazamiya.net/en/bulk\\_extractor-record-based-carving](https://www.kazamiya.net/en/bulk_extractor-record-based-carving)

# bulk\_extractor

```
import os
with open("MFT", "wb") as f:
    for i in os.listdir():
        if i.endswith(".mft"):
            f.write(open(i, "rb").read())
```

```
C:\Users\hyuunnnn\Desktop\bulk2_1\ntfsmft_carved\000>python
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec  4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import os;
>>> with open("MFT", "wb") as f:
...     for i in os.listdir():
...         if i.endswith(".mft"):
...             f.write(open(i, "rb").read())
```

- MFTECmd, MFT Explorer와 같은 MFT 분석 도구를 사용하여 분석 가능

```
C:\Users\hyuunnnn\Desktop\MFTECmd>MFTECmd.exe -f C:\Users\hyuunnnn\Desktop\bulk2_1\ntfsmft_carved\000\MFT --csv .
MFTECmd version 1.2.2.1

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd

Command line: -f C:\Users\hyuunnnn\Desktop\bulk2_1\ntfsmft_carved\000\MFT --csv .
```

# bulk\_extractor

- MFTExplorer를 사용하여 카빙한 MFT 파일을 정상적으로 분석할 수 있었다.

The screenshot shows the MFT Explorer interface version 2.0.0.0. On the left, there is a tree view of the file system structure. The main area displays a table of files with columns: Name, Image Icon, Name, Parent Path, Is Dir, Is Deleted, SI\_Created On, FN\_Created On, and SI\_Mod. Below the table, a file browser window titled '000' is open, showing a list of files with columns: 이름 (Name), 수정한 날짜 (Last Modified Date), 유형 (Type), and 크기 (Size). The table data is as follows:

Name	Image Icon	Name	Parent Path	Is Dir	Is Deleted	SI_Created On	FN_Created On	SI_Mod
No image data	abc	x86_x64_아직저 차이.pdf.locked	.#Users##User##Downloads	<input type="checkbox"/>	<input type="checkbox"/>	2024-01-03 08:00:29.7427288		2024-0
		winrar-x32-622.exe.locked	.#Users##User##Downloads	<input type="checkbox"/>	<input type="checkbox"/>	2024-01-03 08:03:45.1976262		2024-0
		KEEPER 기술문서 최종발표.pptx.locked	.#Users##User##Downloads	<input type="checkbox"/>	<input type="checkbox"/>	2024-01-03 08:08:39.8219995		2024-0
		disable-defender (2).exe.locked	.#Users##User##Downloads	<input type="checkbox"/>	<input type="checkbox"/>	2024-01-03 07:55:38.5362956		2024-0
		desktop.ini	.#Users##User##Downloads	<input type="checkbox"/>	<input type="checkbox"/>	2024-01-03 07:49:34.8664030		2024-0
		cors(키퍼발표).pptx.locked	.#Users##User##Downloads	<input type="checkbox"/>	<input type="checkbox"/>	2024-01-03 08:00:22.2179580		2024-0
		AccessData_FTK_Imager_4.7.1.exe.locked	.#Users##User##Downloads	<input type="checkbox"/>	<input type="checkbox"/>	2024-01-03 08:19:47.3990431		2024-0
		최종발표.pdf.locked	.#Users##User##Downloads	<input type="checkbox"/>	<input type="checkbox"/>	2024-01-03 08:07:58.8689273		2024-0
		발표자료_합본.pdf.locked	.#Users##User##Downloads	<input type="checkbox"/>	<input type="checkbox"/>	2024-01-03 08:00:37.4429952		2024-0
		2023년 9월 회계부.png.locked	.#Users##User##Downloads	<input type="checkbox"/>	<input type="checkbox"/>	2024-01-03 07:58:54.5776395		2024-0
		2023년 8월 회계부.png.locked	.#Users##User##Downloads	<input type="checkbox"/>	<input type="checkbox"/>	2024-01-03 07:59:49.7024817		2024-0
		2023년 5월 회계부.png.locked	.#Users##User##Downloads	<input type="checkbox"/>	<input type="checkbox"/>	2024-01-03 07:59:55.6241457		2024-0

The bottom navigation bar includes buttons for 새롭 만들기 (New), 파일 (File), 편집 (Edit), 삭제 (Delete), 정렬 (Sort), 보기 (View), 세부 정보 (Detailed Information), and a search bar labeled '000 검색'.

# bulk\_extractor

- evtx 는 일부 파일만 카빙된 것으로 보인다.
  - 존재하는 모든 파일을 카빙해주는 것은 불가능에 가깝다.

99744645129974464512_valid_header_1chunks_1actu...	2024-02-08 오후 8:04	이벤트 로그	68KB
89728901128972890112_valid_header_1chunks_1actu...	2024-02-08 오후 8:03	이벤트 로그	68KB
84266926088426692608_17chunks_1242records.evtx	2024-02-08 오후 8:03	이벤트 로그	1,284KB
84253491208425349120_1chunks_114records.evtx	2024-02-08 오후 8:03	이벤트 로그	196KB
84243660808424366080_2chunks_149records.evtx	2024-02-08 오후 8:03	이벤트 로그	132KB
84165836808416583680_valid_header_1chunks_1actu...	2024-02-08 오후 8:03	이벤트 로그	68KB
532033536532033536_9chunks_979records 이벤트 수: 979			
수준	날짜 및 시간 ^	원본	이벤트 ID   작업 범주
정보	2024-01-03 오후 5:08:58	Shell-Core	62144 (62132)
정보	2024-01-03 오후 5:09:01	Shell-Core	62144 (62132)
정보	2024-01-03 오후 5:09:01	Shell-Core	62144 (62132)
이벤트 62144, Shell-Core			
일반	자세히		
패키지의 설치 상태를 Microsoft.MicrosoftOfficeHub_8wekyb3d8bbwe에서 '완료'로 업데이트하는 중입니다(HRESULT 0).			

# bulk\_extractor

- 복구가 불가능한 파일 시스템의 경우 위와 같은 카빙 도구를 사용하여 분석할 수도 있겠다.
- 그러나 테스트로 사용한 이미지 파일은 문제가 없는 파일이기 때문에 도구의 유용성 확인이 어렵다.
  - FTK Imager로 모든 파일들을 볼 수 있으며, 손상되지 않은 파일이기 때문이다.
- 메모리 분석에서 MFT, 레지스트리, evttx, sqlite 파일 등을 추출할 때 사용에 문제가 있을 정도로 많이 손상되어 있다.

---

<sup>1</sup> <https://hyuunnn.github.io/2023/01/27/forensic-recover/>

메모리 덤프에서 네트워크 통신에 관한 더 자세한 정보를 알아내는 좋은 방법은 `bulk_extractor`를 사용해 메모리 덤프의 내용에서 패킷 캡쳐 파일(.pcap)을 추출할 수 있는지 확인하는 것이다.

`Volatility`의 `netscan` 플러그인 결과에 나타나지 않았던 침해된 시스템과 다른 시스템 사이에 4번의 '통신'이 있었음을 알 수 있다.

이는 `Volatility` 플러그인이 네트워크 연결을 감지하지 못했다는 것을 의미하지 않는다.

`Volatility` 플러그인은 윈도우에서 유지되는 연결 정보를 찾기 때문에 `bulk_extractor`가 네트워크 패킷을 찾고 파싱하는 것과 다르다.

"이러한 발견은 데이터에 대한 포괄적인 시야를 제공하기 위해 서로 다른 여러 도구를 실행하는 것이 가치 있음을 보여준다."

[Windows 환경에서 침해 시스템 분석하기 - p150, 151, Eng](#)

# 참고 자료

- [https://forensics.wiki/bulk\\_extractor/](https://forensics.wiki/bulk_extractor/)
- <https://warroom.rsmus.com/find-sensitive-data-with-bulk-extractor/>
- [docs, wiki, User's Manual, Worked Examples, DFRWS 2012 bulk\\_extractor tutorial](#)
- Extract Sensitive Information from Drives Using Bulk Extractor
- Forensic Carving of Network Packets with bulk\_extractor and tcpflow - Slides
- Sharpening Your Tools: Updating bulk\_extractor for the 2020s
- <https://volatility-labs.blogspot.com/2015/01/incorporating-disk-forensics-with.html>
- <https://www.stark4n6.com/2020/12/magnet-weekly-ctf-week-10-warrens.html>
- <https://www.petermstewart.net/magnet-weekly-ctf-week-11/>

# foremost<sup>1 2</sup>

- apt-get install foremost
- foremost 보다 scalpel 을 권장하고 있다.<sup>2</sup>

```
→ ~ foremost -h
foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus.
$ foremost [-v|-V|-h|-T|-Q|-q|-a|-w-d] [-t <type>] [-s <blocks>] [-k <size>]
      [-b <size>] [-c <file>] [-o <dir>] [-i <file>]
```

## Spinoffs

Foremost served as the basis for [Golden G. Richard III's Scalpel](#), a significantly faster program to also recover deleted files. It has also inspired [tcpextract](#), a program for extracting file from network traffic.

Foremost's authors have recommended that practitioners use [Scalpel](#) instead of Foremost.

---

<sup>1</sup> [https://en.wikipedia.org/wiki/Foremost\\_\(software\)](https://en.wikipedia.org/wiki/Foremost_(software))

<sup>2</sup> <https://forensics.wiki/foremost/>

<sup>3</sup> [Recover Permanently Deleted Files Using Foremost](#)

# scalpel<sup>1</sup>

- foremost 를 기반으로 만든 데이터 카빙 도구
- 마지막 버전( 2.0 )의 윈도우 전용 실행 파일( exe )을 제공한다. - Scalpel-2.0

```
C:\Users\hyuunnnn\Desktop\Scalpel-2.0-master>scalpel.exe
Scalpel version 2.0
Written by Golden G. Richard III and Lodovico Marziale.
ERROR: Couldn't open configuration file:
C:\Users\hyuunnnn\Desktop\Scalpel-2.0-master\scalpel.conf -- No such file or directory
Scalpel was unable to read a needed file and will abort.
```

- foremost 에 비해 좋은 퍼포먼스를 제공한다고 한다.<sup>2</sup>

Scalpel 1.5	1h33m10s
Foremost 0.69	6h21m54s

**Table 6.** Carving results for 8GB raw drive (unknown source, no partition table) on P2-350. Carving parameters: 10MB GIF, 10MB JPG, 10MB AVI, 10MB MOV, 10MB MPG, 100K BMP, 5MB DOC, 50MB PST/OST, 50K HTML, 5MB PDF, 200K WAV, 1MB RealAudio, 10MB ZIP. ~52,000 files carved.

Scalpel 1.5	2h40m39s
Foremost 0.69	9h50m31s

**Table 7.** Carving results for 40GB NTFS (from a UNO laboratory) on P2-350. Carving parameters:

<sup>1</sup> <https://forensics.wiki/scalpel/>

<sup>2</sup> Scalpel: A Frugal, High Performance File Carver

# PhotoRec<sup>1</sup>

PhotoRec ignores the file system and goes after the underlying data, so it will still work even if your media's file system has been severely damaged or reformatted.

- 파일 시스템을 무시하고 데이터를 복구하기 때문에 손상되거나 포맷된 경우에도 작동한다.<sup>1</sup>
- 복구 능력만 따지면 `scalpel` 보다 뛰어나며, 다양한 파일 타입을 제공한다고 한다.<sup>2</sup>
- 그러나 `foremost`, `scalpel`, `PhotoRec` 을 테스트하는 논문의 결과는 무조건 그렇진 않은 것 같다.<sup>3</sup>
  - 하지만 해당 리포트는 2011년에 작성되었고, 현재까지 유일하게 업데이트되고 있는 도구이기 때문에 지금은 어떤 결과가 나올지 모른다.
- 사용 방법: [A File's Life - File Deletion and Recovery](#)

---

<sup>1</sup> <https://en.wikipedia.org/wiki/PhotoRec>

<sup>2</sup> <https://news.ycombinator.com/item?id=23059344>

<sup>3</sup> <https://core.ac.uk/download/pdf/36708564.pdf>

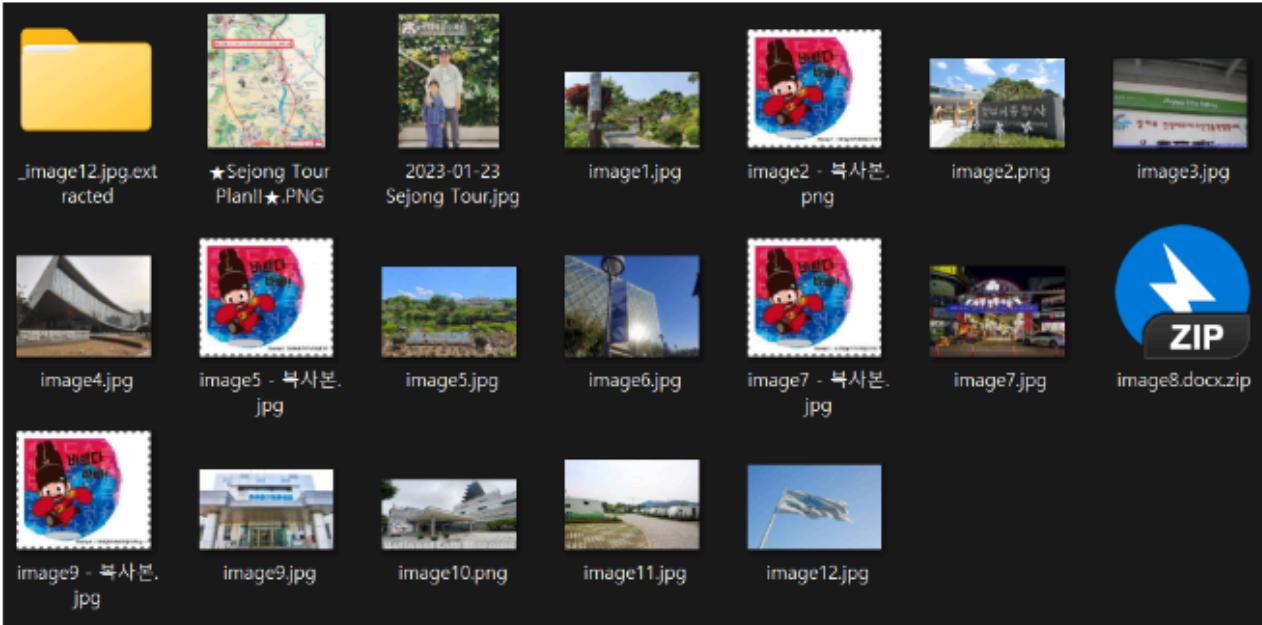
# binwalk

- 시그니처 기반으로 파일 내부에 있는 또 다른 파일들을 추출할 수 있다.  
→ 파일 안에 숨겨둔 파일들을 찾을 수 있다.  
(CTF에서도 활용 가능<sup>1</sup>)
- 도구의 원래 목적은 펌웨어 이미지 분석에 사용되는 도구이다.
  - 하지만 펌웨어도 결국 파일 안에 존재하는 부트로더, 커널 이미지 등을 추출하는 것<sup>12</sup>
- apt-get install binwalk로 설치 가능

<sup>1</sup> <https://mandu-mandu.tistory.com/472>

<sup>2</sup> <https://lifs.hallym.ac.kr/blog/2019/08/19/File-System-Analysis-of-LG-Forensic-Image-with-Binwalk.html>

<sup>3</sup> <https://sergioprado.blog/reverse-engineering-router-firmware-with-binwalk/>



```
mandu@mandu-VirtualBox:/media/sf_share/media$ binwalk image2.png
DECIMAL      HEXADECIMAL      DESCRIPTION
----          -----          -----
0            0x0              PNG image, 600 x 400, 8-bit/color RGBA, non-interlaced
91           0x5B             Zlib compressed data, compressed
546443       0x8568B          JPEG image data, JIFIF standard 1.01
546473       0x856A9          TIFF image data, big-endian, offset of first image directory: 8
```

binwalk 명령어를 사용해서 Sejong Tour Plan!!.png 이미지에 표시된 이미지에 해당되는 파일 image2.png, image5.jpg, image7.jpg, image9.jpg에서 end시그니쳐 뒤에 붙어있는 다른 이미지 파일을 분리해낼 수 있다.

# magika<sup>1</sup>

- AI 기반 파일 탐지 도구라고 한다.
- 약 1MB에 불과하며 단일 CPU에서도 밀리초 내에 파일을 식별한다.
- 백만개 이상의 파일, 100개 이상의 탐지에서 99% 이상의 정확도를 달성했다고 한다.

```
$ magika -r examples/
examples/README.md: Markdown document (text)
examples/bmp.bmp: BMP image data (image)
examples/code.asm: Assembly (code)
examples/code.py: Python source (code)
examples/doc.docx: Microsoft Word 2007+ document (document)
examples/doc.ini: INI configuration file (text)
examples/elf64.elf: ELF executable (executable)
examples/flac.flac: FLAC audio bitstream data (audio)
examples/java.class: Java compiled bytecode (executable)
examples/jpg.jpg: JPEG image data (image)
examples/pdf.pdf: PDF document (document)
examples/pe32.exe: PE executable (executable)
examples/png.png: PNG image data (image)
examples/tar.tar: POSIX tar archive (archive)
examples/webm.webm: WebM data (video)
```

---

<sup>1</sup> Magika: AI powered fast and efficient file type identification

# EVTXtract

- 리눅스에 기본으로 설치되어 있는 `python2.7` 활용 - `pip2 install evtxtract`

```
hyuunnn@hyuunnn:/mnt/e/forensic_study/memprocfs/winevt/Logs$ evtxtract ffffbe8e758b2a90-Security.evtx > result.xml
INFO:evtxtract:no matching templates for record at offset: 0x1b1200
INFO:evtxtract:no matching templates for record at offset: 0x1bb040
INFO:evtxtract:no matching templates for record at offset: 0x1bb290
```

## UsnJrnICarver, RcrdCarver, MftCarver, yarp ( yarp-carver, yarp-memcarver )

## SQLITE 파일 복구 - undark, sqlite-dissect, sqbrite, fqlite, SQLite-Deleted-Records-Parser

- 그 외에도 적절한 Carver, Recover 도구를 찾아서 사용하거나 만들어야 하는 상황이 발생할 수도 있다.

# 메모리 포렌식

- 휘발성 메모리<sup>1</sup>인 RAM에 존재하는 다양한 정보들을 분석하는 기법
- 메모리 포렌식을 수행하기 위해 메모리 내의 데이터 일부 또는 전체를 저장하는 **메모리 덤프 과정**이 필요하다.<sup>3</sup>
- 메모리 포렌식 분석을 통해 네트워크 정보, 실행 중이거나 숨겨진 프로세스 정보, 키보드 입력 정보 외에도 지금까지 배웠던 포렌식 분석이 일부 가능하다.
  - 일부만 가능한 이유는 휘발성 데이터이기 때문에 추출할 수 있는 윈도우 아티팩트 정보가 한정적이다.
  - 악성 파일을 저장하지 않고 메모리에 바로 로드되어 실행되는 **fileless**의 경우 메모리 포렌식이 필요할 수 있다.<sup>45</sup>

---

<sup>1</sup> [https://ko.wikipedia.org/wiki/휘발성\\_메모리](https://ko.wikipedia.org/wiki/휘발성_메모리)

<sup>2</sup> [https://en.wikipedia.org/wiki/Memory\\_forensics](https://en.wikipedia.org/wiki/Memory_forensics)

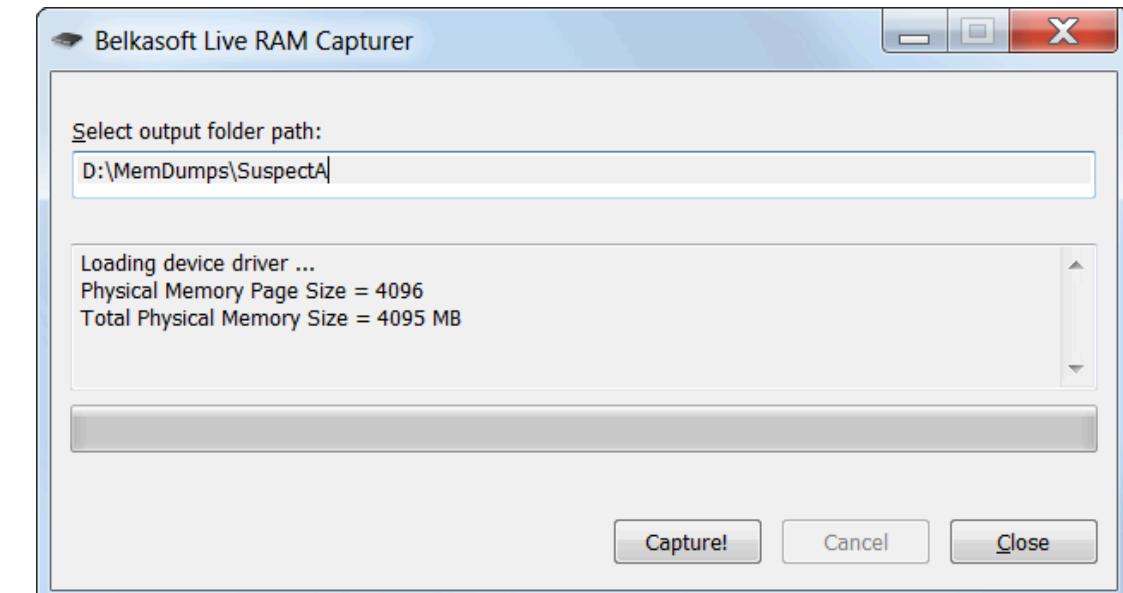
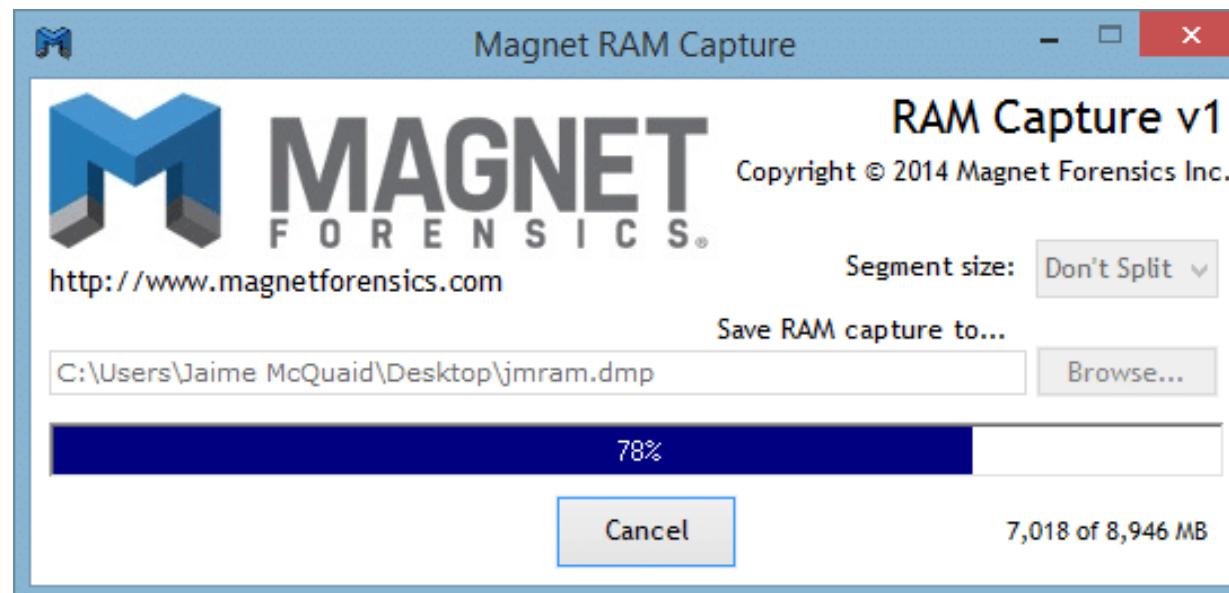
<sup>3</sup> 메모리 덤프 기반 포렌식을 활용한 개인 식별정보 노출 분석

<sup>4</sup> Powershell을 이용한 파일리스 공격

<sup>5</sup> 파일리스(Fileless) 사이버공격의 분류 모델

# 메모리 포렌식

- 덤프 도구
  - WinPmem, MAGNET RAM Capture<sup>1</sup>, Belkasoft RAM Capturer
  - avml, dumpit-linux



<sup>1</sup> <https://www.magnetforensics.com/blog/acquiring-memory-with-magnet-ram-capture/>

# hiberfil.sys

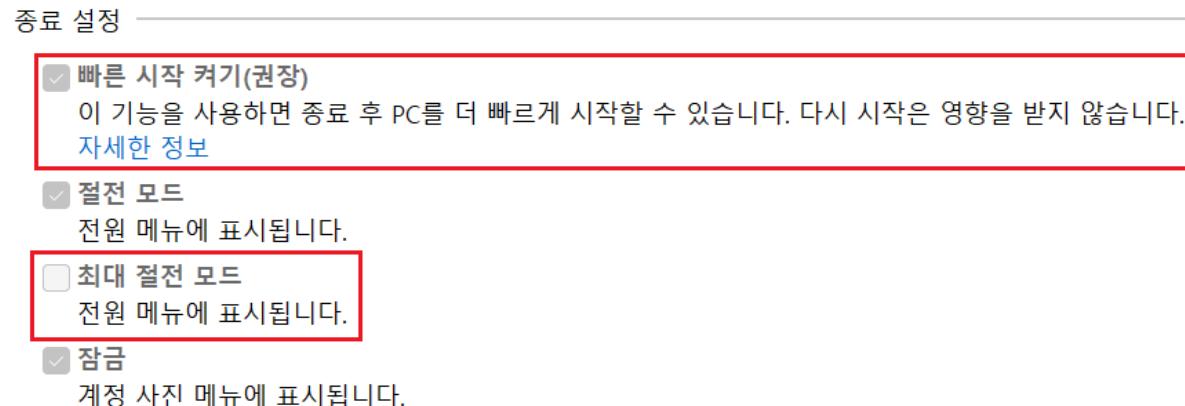
- 윈도우에서 제공하는 전원 및 절전 옵션 선택에 따라 휘발성 메모리 데이터를 `hiberfil.sys`에 저장<sup>1</sup>
- 빠른 시작 켜기 옵션이 활성화되어 있을 때 시스템 종료 시 윈도우 커널 및 드라이버의 사용 영역을 `hiberfil.sys`에 저장 후 종료
  - 빠른 시작 켜기 : 부팅 시간의 감소를 위한 기능
- 최대 절전 모드 옵션이 활성화되어 있을 때 최대 절전 모드에 진입하기 전의 휘발성 메모리 데이터를 그대로 `hiberfil.sys`에 저장
- 하이브리드 절전 모드는 절전 모드, 최대 절전 모드의 기능을 결합한 모드
  - 역시 `hiberfil.sys`에 휘발성 메모리 데이터가 저장된다.
- **Hibernation Recon**, **Volatility**, **strings** 명령어 등 활용 가능

---

<sup>1</sup> Windows 10 내의 `hiberfil.sys` 파일에 대한 포렌식 활용 방안

# hiberfil.sys

- 제어판 → 전원 옵션 → 전원 단추 작동 설정 → 빠른 시작 켜기



〈Table 1〉 Comparison of memory storage state and default activation by options

옵션		hiberfil.sys 파일로의 메모리 데이터 저장 여부	저장되는 메모리 영역	기본 활성화 여부
시스템 종료	빠른 시작 ON	O	커널	O
	빠른 시작 OFF	X	-	X
다시 시작		X	-	O
절전 모드		X	-	X
최대 절전 모드		O	커널 + 사용자	O
하이브리드 절전 모드		O	커널	O

<sup>1</sup> Windows 10 내의 hiberfil.sys 파일에 대한 포렌식 활용 방안

# pagefile.sys<sup>1</sup> & swapfile.sys<sup>23</sup>

- RAM이 가득찼을 때 일부 내용을 `pagefile.sys`라는 가상 메모리 파일에 옮겨 실제 메모리 공간을 확보한다.
- `pagefile.sys`은 RAM 보다 읽는 속도가 느리다. (파일 형태로 저장되어 있기 때문이다.)  
→ 자주 액세스하지 않는 데이터들을 보관한다.
- `swapfile.sys`는 페이지 파일과 유사하지만 프로그램을 더 빠르게 실행하기 위한 용도라고 한다.
  - 유의미한 데이터가 존재할 수 있다. ([strings](#), [bstrings](#), [page\\_brute](#), [glogg](#) 등 문자열 분석 도구 활용)

 hiberfil.sys	2024-02-13 오후 4:27	시스템 파일	6,597,908KB
 pagefile.sys	2024-02-13 오후 4:27	시스템 파일	16,777,216KB
 swapfile.sys	2024-02-13 오후 4:27	시스템 파일	16,384KB

<sup>1</sup> <https://forensafe.com/blogs/pagefile.html>

<sup>2</sup> [https://forensafe.com/blogs/swap\\_file\\_urls.html](https://forensafe.com/blogs/swap_file_urls.html)

<sup>3</sup> <http://forensic-proof.com/archives/3994>

# pagefile.sys & swapfile.sys

- strings를 사용하여 문자들을 추출한 결과 혹은 파일 자체를 glogg, grep 등을 사용하여 데이터 분석

## Examine Strings in the Malicious svchost Process

- You can extract both ASCII and Unicode strings from the process memory space with the following command line.

```
strings M:\name\svchost.exe-2884\minidump\minidump.dmp > %USERPROFILE%\Desktop\svchost-2884.txt
```

- Then, you can find a suspicious URL by applying one of the filters we mentioned before. This should be the malicious URL.

The screenshot shows the command-line interface of the strings tool. In the 'Text:' field, the regular expression '^s\*https?:\/\/' is entered and highlighted with a red box. Below it, a message says '84 matches found.' A list of matches is displayed, each consisting of a red dot followed by a process ID and a URL. The first four matches are highlighted with a red box:

- 9973 https://www.monferriina.com/web/ssl1q2w/index.php
- 9974 https://www.monferriina.com/web/ssl1q2w/index.php
- 9984 https://www.monferriina.com/web/ssl1q2w/index.php
- 9986 https://www.monferriina.com/web/ssl1q2w/index.php

A small number '124' is visible in the bottom right corner of the list area.

<sup>1</sup> [https://www.slideshare.net/IIJ\\_PR/super-easy-memory-forensics](https://www.slideshare.net/IIJ_PR/super-easy-memory-forensics)

# Volatility3

- Volatility는 메모리에 존재하는 다양한 정보들을 추출하기 위해 사용되는 도구
- 2.6.1 이후로 업데이트가 종료되었으며, 이전에 개발된 플러그인들이 3 버전으로 포팅되고 있다.
- 특히 3 버전에서 속도가 크게 향상되었다고 한다.<sup>1</sup>

The image shows two GitHub repository pages side-by-side: 'volatility' and 'volatility3'.  
The top section shows the 'volatility' repository. It has 11 branches and 10 tags. A recent commit by 'iMHLv2' is shown: 'Merge pull request #765 from volatilityfoundation/win10\_19041' with commit hash 'a438e76' and a timestamp of '4 years ago'. This commit is highlighted with a red box.  
The bottom section shows the 'volatility3' repository. It has 30 branches and 10 tags. A recent commit by 'ikelos' is shown: 'Merge pull request #754 from volatilityfoundation/feature/cli-config...' with commit hash '6d34e43' and a timestamp of '4 days ago'. This commit is also highlighted with a red box.  
Both repositories are public, have 'Watch' counts (307 for volatility, 57 for volatility3), and green 'Code' buttons.

<sup>1</sup> <https://volatility3.readthedocs.io/en/latest/vol2to3.html>

# Volatility3

```
$ git clone https://github.com/volatilityfoundation/volatility3
$ cd volatility3
$ pip3 install -r requirements.txt
$ python3 vol.py -h # 사용할 수 있는 argument 옵션, 플러그인 리스트 확인 가능
```

```
→ volatility3 git:(develop) ls
API_CHANGES.md  MANIFEST.in  doc          requirements-minimal.txt  test      volatility3
CITATION.cff    README.md    mypy.ini       requirements.txt        vol.py    volshell.py
LICENSE.txt     development  requirements-dev.txt  setup.py           vol.spec  volshell.spec
→ volatility3 git:(develop) python3 vol.py
Volatility 3 Framework 2.5.2
usage: volatility [-h] [-c CONFIG] [--parallelism [{processes,threads,off}]] [-e EXTEND] [-p PLUGIN_DIRS]
                  [-s SYMBOL_DIRS] [-v] [-l LOG] [-o OUTPUT_DIR] [-q] [-r RENDERER] [-f FILE] [--write-config]
                  [--save-config SAVE_CONFIG] [--clear-cache] [--cache-path CACHE_PATH] [--offline]
                  [--single-location SINGLE_LOCATION] [--stackers [STACKERS ...]]
                  [--single-swap-locations [SINGLE_SWAP_LOCATIONS ...]]
                  plugin ...
volatility: error: Please select a plugin to run
```

<sup>1</sup> [https://youtu.be/rwTWZ7Q5i\\_w](https://youtu.be/rwTWZ7Q5i_w)

<sup>2</sup> <https://youtu.be/v9oFztyRkbA>

<sup>3</sup> <https://cpuu.postype.com/post/9993241>

# Volatility3

- The Stolen Szechuan Sauce 케이스의 Desktop Memory 파일 사용

```
python3 vol.py -f <path to memory image> plugin_name plugin_option
```

```
→ volatility3 git:(develop) python3 vol.py -f /mnt/e/forensic_study/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem windows.pslist.PsList
Volatility 3 Framework 2.5.2
Progress: 38.28          Scanning memory_layer using BytesScanner
```

```
→ volatility3 git:(develop) python3 vol.py -f /mnt/e/forensic_study/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem windows.pslist.PsList
Volatility 3 Framework 2.5.2
Progress: 100.00          PDB scanning finished
PID      PPID     ImageFileName   Offset(V)    Threads Handles SessionId    Wow64   CreateTime        ExitTime       File output
4         0         System        0xbe8e71087040 151      -      N/A    False   2020-09-19 01:24:07.000000      N/A    Disabled
92        4         Registry      0xbe8e710a6080 4       -      N/A    False   2020-09-19 01:24:04.000000      N/A    Disabled
312        4         smss.exe     0xbe8e71d6d040 2       -      N/A    False   2020-09-19 01:24:07.000000      N/A    Disabled
424        416        csrss.exe    0xbe8e74467140 10      -      0     False   2020-09-19 01:24:08.000000      N/A    Disabled
500        416        wininit.exe  0xbe8e74519080 1       -      0     False   2020-09-19 01:24:08.000000      N/A    Disabled
```

프로세스 리스트를 보여주는 windows.pslist.PsList 플러그인 사용 결과

TMI: 매년 [플러그인 콘테스트](#)가 상금과 함께 진행되고 있다.

<sup>1</sup> <https://volatility3.readthedocs.io/en/latest/getting-started-windows-tutorial.html>

# Volatility3

- 프로파일을 만들어서 사용하자. - 덤프 파일에 대한 config 정보 저장
  - 명령어를 실행할 때마다 스캐닝하는 과정으로 인해 대기 시간이 존재한다.
  - `--save-config <json filename>` : 명령어 종류 상관 없이 실행하면 json 파일이 생성된다.
  - `-c <json filename>` : 프로파일을 사용하여 스캐닝 과정을 생략하여 빠른 분석 가능

```
→ volatility3 git:(develop) python3 vol.py --save-config config.json -f /mnt/e/forensic_study/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem windows.pslist.PsList
Volatility 3 Framework 2.5.2
Progress: 100.00          PDB scanning finished
PID      PPID     ImageFileName  Offset(V)      Threads Handles SessionId      Wow64   CreateTime   ExitTime   File output
```

```
→ volatility3 git:(develop) python3 vol.py -c config.json -f /mnt/e/forensic_study/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem windows.pslist.PsList
Volatility 3 Framework 2.5.2
Progress: 100.00          PDB scanning finished
PID      PPID     ImageFileName  Offset(V)      Threads Handles SessionId      Wow64   CreateTime   ExitTime   File output
```

<sup>1</sup> <https://cpuu.postype.com/post/9993241>

# Volatility3

- config.json 파일 내부에는 운영체제 버전, 비트( 32bit or 64bit ) 등 덤프 파일에 대한 정보들이 저장되어 있다.

```
→ volatility3 git:(develop) cat config.json
{
    "dump": false,
    "kernel.layer_name.class": "volatility3.framework.layers.intel.WindowsIntel32e",
    "kernel.layer_name.kernel_virtual_offset": 272684833390592,
    "kernel.layer_name.memory_layer.class": "volatility3.framework.layers.physical.FileLayer",
    "kernel.layer_name.memory_layer.location": "file:///mnt/e/forensic_study/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem",
    "kernel.layer_name.page_map_offset": 1757184,
    "kernel.layer_name.swap_layers": true,
    "kernel.layer_name.swap_layers.number_of_elements": 0,
    "kernel.offset": 272684833390592,
    "kernel.symbol_table_name.class": "volatility3.framework.symbols.windows.WindowsKernelIntermedSymbols",
    "kernel.symbol_table_name.isf_url": "file:///home/hyuumnn/volatility3/volatility3/symbols/windows/ntkrnlmp.pdb/81BC5C377C525081645F9958F209C527-1.json.xz",
    "kernel.symbol_table_name.symbol_mask": 0,
    "physical": false,
    "pid": []
}
```

# Volatility3

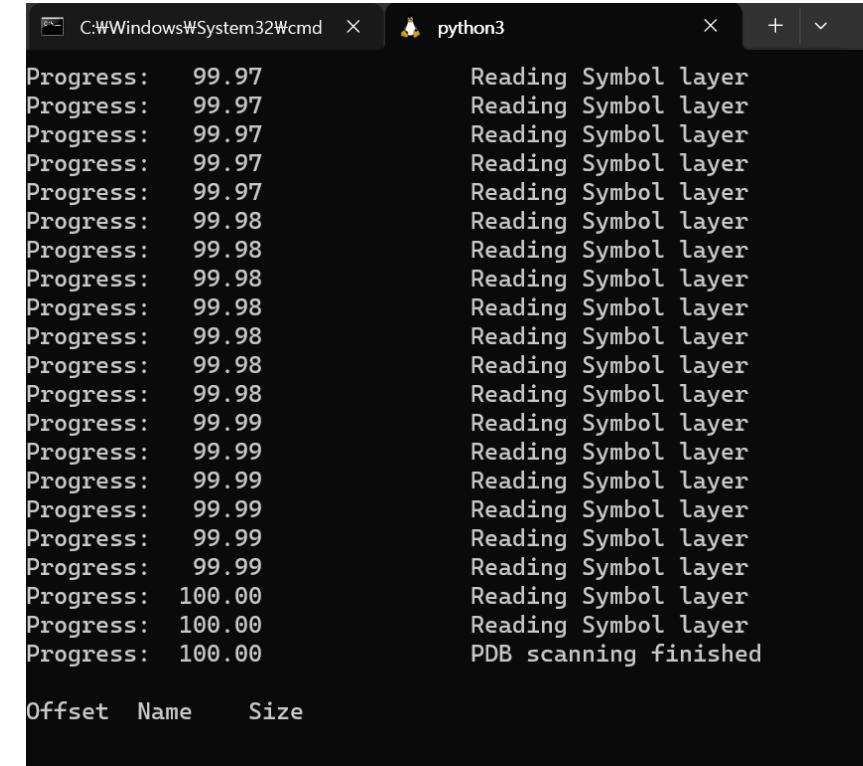
- windows.pstree.PsTree
- windows.cmdline.CmdLine
- windows.filescan.FileScan
- windows.dumpfiles.DumpFiles
- windows.netscan.NetScan
- windows.netstat.NetStat
- windows.mftscan.MFTScan
- windows.memmap.Memmap
- timeliner.Timeliner
- ...

---

<sup>1</sup> <https://volatility3.readthedocs.io/en/latest/volatility3.plugins.html>

# volatility 2.6.1

- volatility3이 정상적으로 동작하지 않는다면?
  - 어쩔 수 없이 2.6.1 을 사용해야 한다.
  - 비교적 최근에 프로젝트가 시작되었기 때문에 아직 자잘한 버그가 있다. - 오른쪽 사진처럼 프로그램이 끝나지 않는다.
- 아직 3 버전으로 옮겨지지 않은 플러그인을 사용해야 한다면 역시 2.6.1 버전을 사용해야 한다.
- volatility3은 profile을 자동으로 찾아주며, 필요한 경우 온라인에서 자동으로 다운로드하여 분석한다.<sup>1</sup>
- 그러나 2 버전은 지원하지 않는 profile인 경우 직접 만들어야 한다.<sup>2</sup>



A screenshot of a Windows command prompt window titled 'cmd' and a Python terminal window titled 'python3'. Both windows show the output of the volatility command. The cmd window shows progress messages like 'Progress: 99.97' followed by 'Reading Symbol layer' repeated many times, and finally 'PDB scanning finished'. The python3 window shows a similar sequence of messages.

```
Progress: 99.97          Reading Symbol layer
Progress: 99.98          Reading Symbol layer
Progress: 99.99          Reading Symbol layer
Progress: 100.00         Reading Symbol layer
Progress: 100.00         Reading Symbol layer
Progress: 100.00         Reading Symbol layer
PDB scanning finished
```

Offset	Name	Size
--------	------	------

<sup>1</sup> <https://cpuu.postype.com/post/9993241>

<sup>2</sup> <https://cpuu.postype.com/post/665132>

# volatility 2.6.1

```
$ git clone https://github.com/volatilityfoundation/volatility
$ cd volatility
$ apt-get install python-pip # volatility는 python 2버전 사용
$ python2 setup.py build
$ sudo python2 setup.py install
$ pip2 install distorm3==3.4.1 # python2 마지막으로 지원하는 버전
$ pip2 install pycryptodome
$ pip2 install openpyxl==2.1.2 # (Optional) xlsx 파일 생성할 때 사용
$ python2 vol.py -h # 사용할 수 있는 argument 옵션, 플러그인 리스트 확인 가능
```

```
→ volatility git:(master) python2 vol.py
Volatility Foundation Volatility Framework 2.6.1
ERROR    : volatility.debug     : You must specify something to do (try -h)
```

- [volatility-binaries](#)에서 2.6.1, 3 파일을 받아서 사용해도 된다.

# volatility 2.6.1

- `imageinfo` 명령어를 사용하여 수동으로 profile 정보 획득

```
→ volatility git:(master) python2 vol.py -f /mnt/e/forensic_study/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO    : volatility.debug    : Determining profile based on KDBG search ...

        Suggested Profile(s) : Win10x64_19041
                                AS Layer1 : SkipDuplicatesAMD64PagedMemory (Kernel AS)
                                AS Layer2 : FileAddressSpace (/mnt/e/forensic_study/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem)
                                PAE type : No PAE
                                DTB : 0x1ad002L
                                KDBG : 0xf80163614b20L
        Number of Processors : 2
Image Type (Service Pack) : 0
                            KPCR for CPU 0 : 0xfffff801617f9000L
                            KPCR for CPU 1 : 0xfffffa481957c0000L
                            KUSER_SHARED_DATA : 0xfffff780000000000L
        Image date and time : 2020-09-19 05:10:39 UTC+0000
Image local date and time : 2020-09-18 22:10:39 -0700
```

- `pslist` , `pstree` , `psxview` , `cmdline` , `memdump` , `filescan` , `dumpfiles` 등 명령어를 사용하여 분석

<sup>1</sup> <https://blog.onfvp.com/post/volatility-cheatsheet/>

# volatility 2.6.1

```
→ volatility git:(master) python2 vol.py -f /mnt/e/forensic_study/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem --profile Win10x64_19041 pslist  
Volatility Foundation Volatility Framework 2.6.1
```

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start	Exit
---									
0xfffffbe8e71087040	System	4	0	151	0	-----	0	2020-09-19 01:24:07 UTC+0000	
0xfffffbe8e710a6080	Registry	92	4	4	0	-----	0	2020-09-19 01:24:04 UTC+0000	
0xfffffbe8e71d6d040	smss.exe	312	4	2	0	-----	0	2020-09-19 01:24:07 UTC+0000	

```
→ volatility git:(master) x python2 vol.py -f /mnt/e/forensic_study/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem --profile Win10x64_19041 cmdline  
Volatility Foundation Volatility Framework 2.6.1
```

```
*****  
System pid: 4  
*****  
Registry pid: 92  
*****  
smss.exe pid: 312  
*****  
csrss.exe pid: 424  
Command line : 朝鲜语  
*****  
wininit.exe pid: 500  
*****  
services.exe pid: 616  
Command line : C:\Windows\system32\services.exe  
*****  
lsass.exe pid: 656  
Command line : C:\Windows\system32\lsass.exe  
*****  
svchost.exe pid: 764  
Command line : C:\Windows\system32\svchost.exe -k DcomLaunch -p
```

# volatility 2.6.1

- filescan : 메모리에 존재하는 FILE\_OBJECT 구조체 탐색

```
→ volatility git:(master) python2 vol.py -f /mnt/e/forensic_study/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem --profile Win10x64_19041 filescan >> filescan.txt
Volatility Foundation Volatility Framework 2.6.1
→ volatility git:(master) x cat filescan.txt
Offset(P)      #Ptr  #Hnd Access Name
_____|_____|_____|_____|_____
0x0000be8e710aed40  32781    1  RW-r-- \Device\HarddiskVolume3\Windows\System32\winevt\Logs\Microsoft-Windows-Kernel-WHEA%4Operatio
nal.evtx
0x0000be8e71a65940   19     0  RW-rwd \Device\HarddiskVolume3\$MftMirr
0x0000be8e71a65d90   15     0  R--r-d \Device\HarddiskVolume3\Windows\System32\drivers\vmusbmouse.sys
0x0000be8e71cab070    3     0  RW-rwd \Device\HarddiskVolume3\$Extend:$I30:$INDEX_ALLOCATION
0x0000be8e71cab350    3     0  RW-r-- \Device\HarddiskVolume3\$Extend\$RmMetadata\$Repair
0x0000be8e71cab4c0   24     0  RW-rwd \Device\HarddiskVolume3\$Extend\$UsnJrnl:$J:$DATA
0x0000be8e71cab630  32769    1  RWDrwd \Device\clfs\Device\HarddiskVolume3\$Extend\$RmMetadata\$Txfl0a\$Txfl0a
```

- dumpfiles : filescan 의 offset 주소를 통해 매핑되어 있는 파일 추출

```
→ volatility git:(master) x python2 vol.py -f /mnt/e/forensic_study/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem --profile Win10x64_19041 dumpfiles -Q 0x0000be8e71a65940 -D .
Volatility Foundation Volatility Framework 2.6.1
```

# volatility 2.6.1

- netscan : 네트워크 관련 정보 분석

```
→ volatility git:(master) x python2 vol.py -f /mnt/e/forensic_study/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem --profile Win10x64_19041 netscan >> netscan.txt
Volatility Foundation Volatility Framework 2.6.1
→ volatility git:(master) x cat netscan.txt
Offset(P)      Proto    Local Address          Foreign Address      State        Pid  Owner           Created
0xbe8e71cc0be0  TCPv4    0.0.0.0:3389        0.0.0.0:0            LISTENING   520  svchost.exe    2020-09-19
01:24:09 UTC+0000
0xbe8e71cf52f0  TCPv4    0.0.0.0:49666       0.0.0.0:0            LISTENING   988  svchost.exe    2020-09-19
01:24:09 UTC+0000
0xbe8e71cf52f0  TCPv6    ::: 49666          :::: 0              LISTENING   988  svchost.exe    2020-09-19
01:24:09 UTC+0000
```

- timeliner : volatility 플러그인들의 아티팩트를 시간 순서로 분석 가능

```
→ volatility git:(master) x python2 vol.py -f /mnt/d/Forensic_Challenge/dfirmadness/the-stolen-szechuan-sauce/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem --profile Win10x64_19041 timeliner --output-file=result.csv
Volatility Foundation Volatility Framework 2.6.1
Outputting to: result.csv
```

```
→ volatility git:(master) x python2 vol.py -f /mnt/d/Forensic_Challenge/dfirmadness/the-stolen-szechuan-sauce/DESKTOP-SDN1RPT-memory/DESKTOP-SDN1RPT.mem --profile Win10x64_19041 timeliner --output-file=result.xlsx --output=xlsx
Volatility Foundation Volatility Framework 2.6.1
Outputting to: result.xlsx
```

# volatility 2.6.1

- timeliner 플러그인 출력 결과
  - 프로세스 사용 기록, 레지스트리 접근 기록, 장치 기록( SYMLINK ) 등 확인 가능

Column1	Column2	Column3	Column4
-	[PE DEBUG]	gdi32full.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
2020-09-19 01:24:08	[DLL LOADTIME (dll)]	gdi32full.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
2011-02-09 21:34:19	[PE HEADER (dll)]	kernel.appcore.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
-	[PE DEBUG]	kernel.appcore.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
2020-09-19 01:24:08	[DLL LOADTIME (dll)]	kernel.appcore.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
2040-03-22 02:34:05	[PE HEADER (dll)]	user32.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
-	[PE DEBUG]	user32.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
2020-09-19 01:24:08	[DLL LOADTIME (dll)]	user32.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
2070-07-18 00:37:56	[PE HEADER (dll)]	ucrtbase.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
-	[PE DEBUG]	ucrtbase.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
2020-09-19 01:24:08	[DLL LOADTIME (dll)]	ucrtbase.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
2055-06-07 02:27:41	[PE HEADER (dll)]	msvcrt.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
-	[PE DEBUG]	msvcrt.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
2020-09-19 01:24:08	[DLL LOADTIME (dll)]	msvcrt.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
2093-09-23 02:17:46	[PE HEADER (dll)]	sechost.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
-	[PE DEBUG]	sechost.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
2020-09-19 01:24:08	[DLL LOADTIME (dll)]	sechost.dll	Process: svchost.exe/PID: 904/PPID: 616/Process POffset: 0x5e06f2c0/
2020-09-19 05:08:59	[PROCESS]	RuntimeBroker.	PID: 7852/PPID: 764/POffset: 0x0cd66080

- Investigating Malware Using Memory Forensics - A Practical Approach

# MemProcFS

- 물리적인 메모리를 가상 파일 시스템에서 파일 형태로 볼 수 있게 해주는 도구
- Multi-threading + C + intelligent parsing → Super Fast!<sup>2</sup>

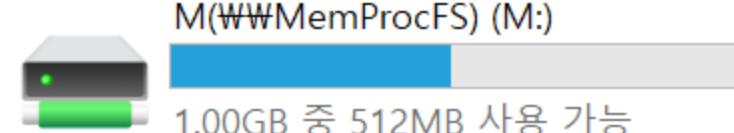
```
C:\Users\hyuunnnn\Desktop\MemProcFS_files_and_binaries_v5.8.25-win_x64-20240207>MemProcFS.exe -f E:\forensic_study\DESKTOP-SDN1RPT-memory\DESKTOP-SDN1RPT.mem -forensic 1
Initialized 64-bit Windows 10.0.19041

===== MemProcFS =====
- Author: Ulf Frisk - pcileech@frizk.net
- Info: https://github.com/ufrisk/MemProcFS
- Discord: https://discord.gg/BCmfBhDPXX
- License: GNU Affero General Public License v3.0

MemProcFS is free open source software. If you find it useful please
become a sponsor at: https://github.com/sponsors/ufrisk Thank You :)

- Version: 5.8.25 (Windows)
- Mount Point: M:\
- Tag: 19041_621678cf
- Operating System: Windows 10.0.19041 (X64)
=====
```

## ▼ 네트워크 위치



<sup>1</sup> [https://www.slideshare.net/IIJ\\_PR/super-easy-memory-forensics](https://www.slideshare.net/IIJ_PR/super-easy-memory-forensics)

<sup>2</sup> <https://youtu.be/mca3rLsHuTA?t=950>

# MemProcFS

✓ Think of MemProcFS as being *roughly* equivalent to an aggregation of the output from numerous Volatility plugins, organized in a hierarchical structure to facilitate easy analysis – **with one command**

- 하나의 명령어로 Volatility 플러그인들의 결과를 집계한 것과 거의 동일하다고 생각하면 된다!
  - 두 도구를 적절히 활용하여 효율적인 분석을 진행하면 되겠다. (도구의 단점을 다른 도구가 커버한다.)
  - Volatility 를 사용하여 \*.evtx, \*.pf, \*.lnk, 레지스트리 하이브 파일 외에도 다양한 파일들을 추출한다고 했을 때 filescan → dumpfiles 명령어 노가다를 해야한다.
    - MemProcFS 는 메모리에서 추출할 수 있는 파일들을 쉽게 가져올 수 있다. ( CTRL+C → CTRL+V )

<sup>1</sup> <https://youtu.be/hjWVUrf7Obk?t=74>

# MemProcFS

- dokany 삭제, 재설치 관련 이슈
  - MemProcFS에서 인식을 못하는 경우가 있을 때 재설치를 해야하는데, 삭제가 안되는 이슈가 있다.<sup>123</sup>
  - HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet001\Services\dokan2 삭제,  
sc delete dokan2 - dokan2 서비스 삭제  
프로그램 추가/제거에서 dokan 검색 후 삭제 → 재부팅 등의 삽질을 하다보면 삭제가 된다...

---

<sup>1</sup> <https://github.com/dokan-dev/dokany/issues/1200>

<sup>2</sup> <https://github.com/dokan-dev/dokany/issues/1201>

# MemProcFS

내 PC > M(\\MemProcFS) (M:) > forensic > csv



- 메모리 덤프에서 얻을 수 있는 다양한 정보들을 CSV 파일로 확인 가능

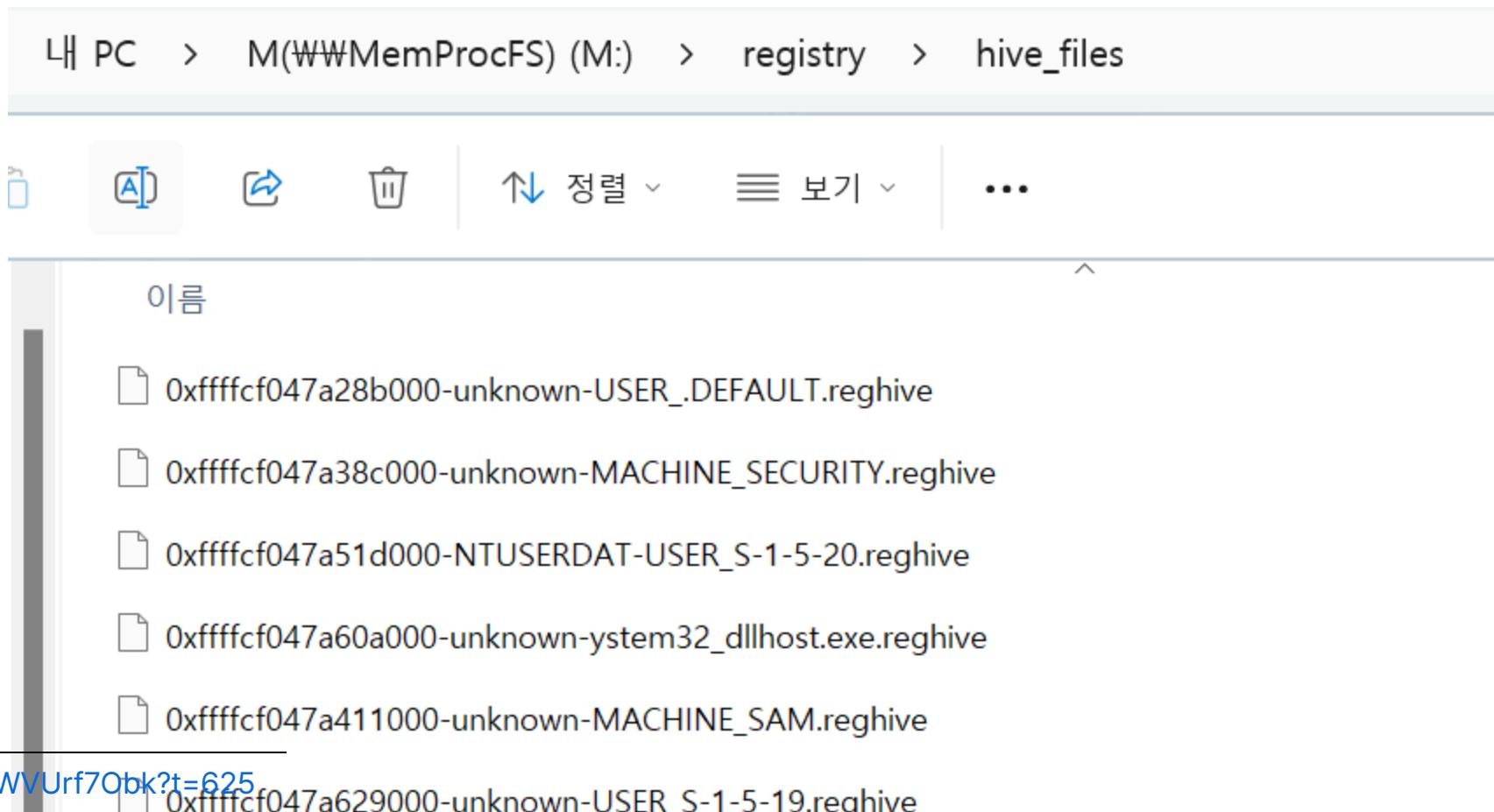
- Excel, Timeline Explorer 등의 도구를 활용하여 분석하면 되겠다.

- timeline\_all.csv 파일의 경우 네트워크, NTFS, 레지스트리 등 추출된 모든 아티팩트들이 타임라인으로 저장되어 있다. (해당 파일 외에도 timeline\_ntfs.csv, findevil.csv 등 많이 있다.)

timeline_all.csv									
Drag a column header here to group by that column									
Line	Tag	Time	Type	Action	PID	Value32	Value64	Text	
387		2020-09-19 05:10:10	NTFS	CRE	0	0x1be	0x3466f800	\1\Users\ricksanchez\AppData\Local\Pac	
388		2020-09-19 05:10:10	NTFS	MOD	0	0x0	0x2ff51c00	\1\Users\ricksanchez\AppData\Local\Pac	
389		2020-09-19 05:10:09	Net	CRE	0	0x0	0xfffffbe8e...	TCPv0 CLOSED ***:0	
390		2020-09-19 05:10:09	Net	CRE	0	0x0	0xfffffbe8e...	TCPv4 CLOSED 10.42.85.115:50991	
391		2020-09-19 05:10:09	REG	MOD	0	0x0	0x0	??\C:\Users\ricksanchez\ntuser.dat\Win	
392		2020-09-19 05:10:09	REG	MOD	0	0x0	0x0	\HAM\AUI\ppleae38af2e007f4358a809ac99a	
393		2020-09-19 05:10:09	NTFS	RD	0	0x0	0x74b0ac00	\1\Users\ricksanchez\AppData\Local\Pac	
394		2020-09-19 05:10:09	NTFS	RD	0	0x5f000	0x4f34f000	\1\Windows\System32\CapabilityAccessMa	
395		2020-09-19 05:10:09	THREAD	CRE	67...	0x19a4	0xfffffbe8e...	TID: 6564	
396		2020-09-19 05:10:09	REG	MOD	0	0x0	0x0	HKLM\SOFTWARE\Microsoft\Windows\Window	
397		2020-09-19 05:10:09	NTFS	MOD	0	0x0	0x3f007c00	\1\Windows\Prefetch\BACKGROUNDTRANSFER	

# MemProcFS

- MemProcFS 의 분석 결과가 아닌 타 도구를 사용하고 싶다면 Registry Explorer, RECmd와 같은 분석 도구를 활용하면 되겠다.<sup>1</sup> (메모리에서 추출하여 손상된 하이브이기 때문에 일부 데이터 분석 가능)



<sup>1</sup> <https://youtu.be/hjWVUrf7Obk?t=625>

# MemProcFS

- 그러나 테스트 결과 대부분의 하이브 파일들을 인식하지 못했다.

Message Date	Message Type ▲	Message
=	=	RBC
2024-02-13 09:02:06	Error	An error occurred loading '0xfffffcf047a411000-unknown-MACHINE_SAM.reghive'. See previous message...
2024-02-13 09:02:07	Error	An error occurred loading '0xfffffcf047a38c000-unknown-MACHINE_SECURITY.reghive'. See previous me...
2024-02-13 09:02:07	Error	"M:\registry\hive_files\0xfffffcf047a411000-unknown-MACHINE_SAM.reghive" is not a Registry hive (b...
2024-02-13 09:02:07	Error	"M:\registry\hive_files\0xfffffcf047a38c000-unknown-MACHINE_SECURITY.reghive" is not a Registry hi...
2024-02-13 09:02:07	Error	An error occurred loading '0xfffffcf047a51d000-NTUSERDAT-USER_S-1-5-20.reghive'. See previous mess...
2024-02-13 09:02:07	Error	An error occurred loading '0xfffffcf0482bb0000-ntuserdat-rs_ricksanchez_ntuser.dat.reghive'. See previo...
2024-02-13 09:02:20	Error	An error occurred loading '0xfffffcf047a38c000-unknown-MACHINE_SECURITY.reghive'. See previous me...
2024-02-13 09:02:20	Error	"M:\registry\hive_files\0xfffffcf047a38c000-unknown-MACHINE_SECURITY.reghive" is not a Registry hi...

- MemProcFS 의 분석 결과 파일들을 활용하거나 추출한 파일에 리커버 도구를 사용하여 분석<sup>1</sup>
  - 구조를 따라가면서 수동으로 분석하는 방법도 있다.
  - 메모리 분석의 한계 (실무에서 메모리 분석은 대부분 보조 역할을 한다.)

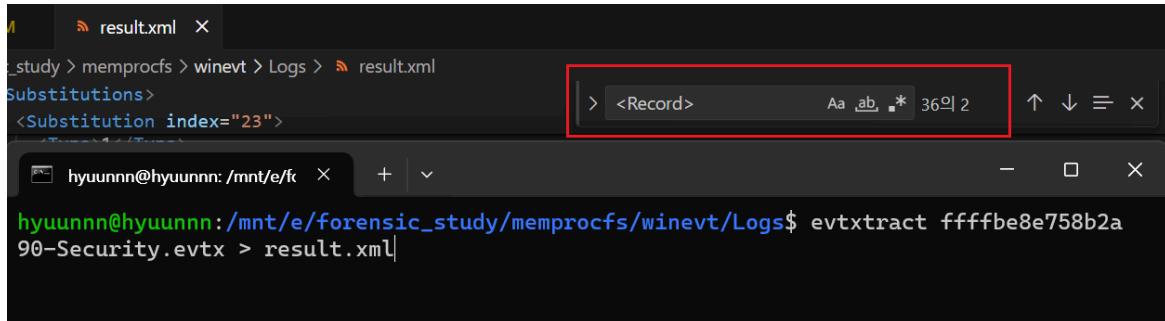
<sup>1</sup> <https://hyuunnn.github.io/2023/01/27/forensic-recover/>

# MemProcFS

- evtx 파일도 분석 도구( EvtxECmd 등)를 포함하여 대부분 인식하지 못했다.

A	B	C	D	E	F	G	H	I	J	K	L
RecordNu	EventRecc	TimeCreat	EventId	Level	Provider	Channel	ProcessId	ThreadId	Computer	ChunkNur	Userld
1	1	41:19.0	8001	Info	Microsoft-Microsoft-		772	1012	WIN-2IH1	0	S-1-5-18
2	2	46:01.7	8004	Info	Microsoft-Microsoft-		772	776	WIN-2IH1	0	S-1-5-18
3	3	46:01.8	8005	Info	Microsoft-Microsoft-		772	944	WIN-2IH1	0	S-1-5-18
4	4	46:31.8	8001	Info	Microsoft-Microsoft-		768	1016	DESKTOP-	0	S-1-5-18
5	5	58:35.0	8004	Info	Microsoft-Microsoft-		768	772	DESKTOP-	0	S-1-5-18
6	6	58:35.0	8005	Info	Microsoft-Microsoft-		768	5488	DESKTOP-	0	S-1-5-18
7	7	08:58.8	8001	Info	Microsoft-Microsoft-		764	2996	DESKTOP-	0	S-1-5-18

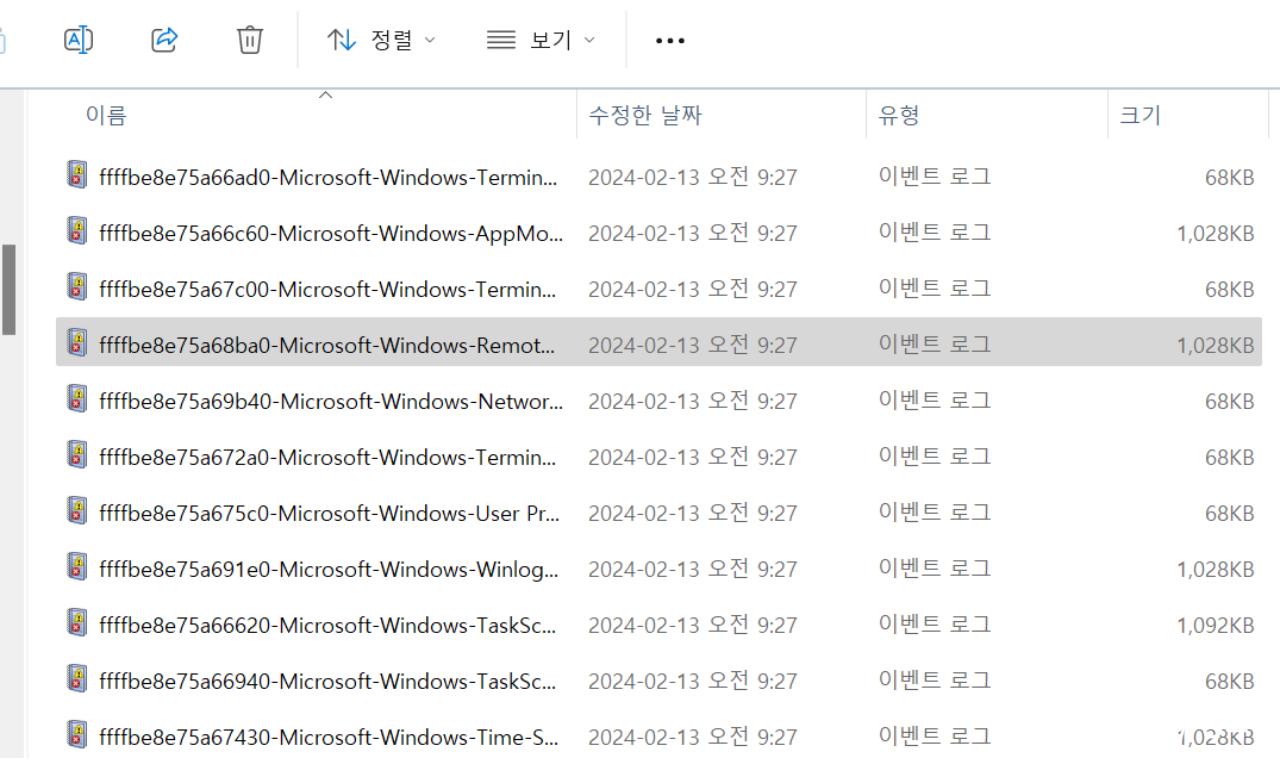
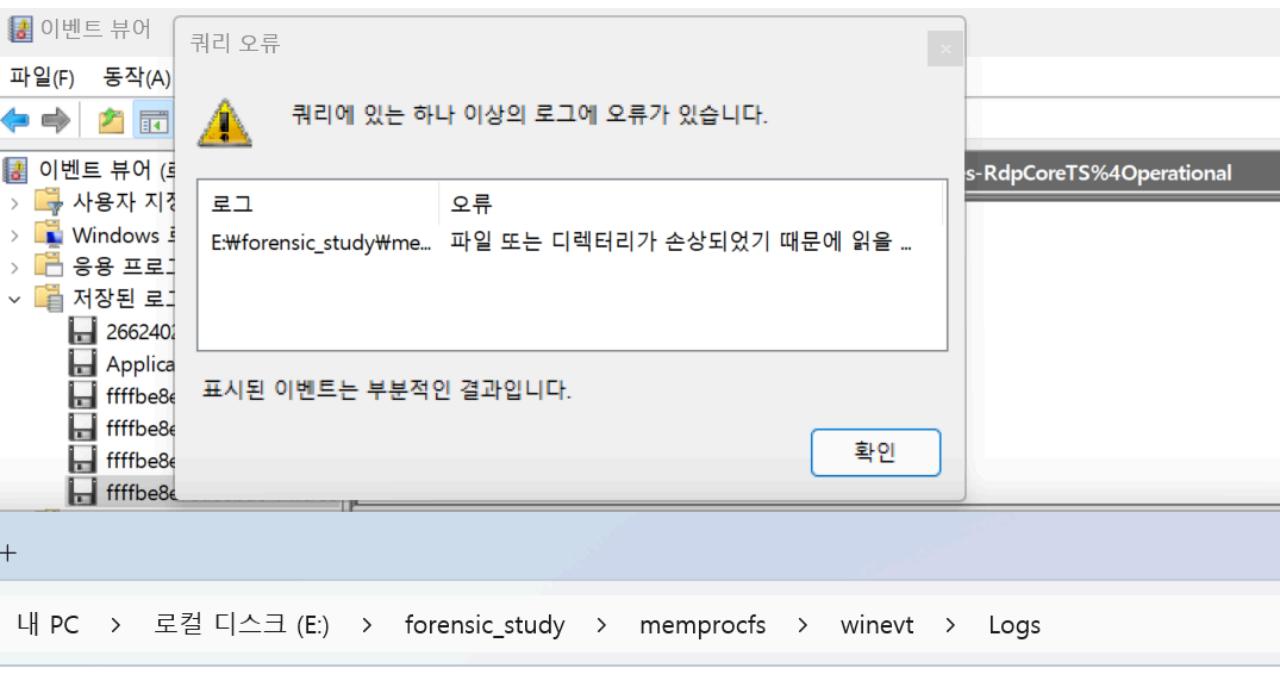
EvtxECmd - 전체 파일 중에서 7개 확인 가능



```
result.xml x
study > memprocfs > winevt > Logs > result.xml
Substitutions>
<Substitution index="23">
hyuunnn@hyuunnn:/mnt/e/forensic_study/memprocfs/winevt/Logs$ evtextract fffffbe8e758b2a
90-Security.evtx > result.xml
```

evtx 파일 하나에서 36개 이벤트 확인 가능

파일 하나에서 이미 7개보다 많은 로그 확보



# MemProcFS

- name 폴더에 들어가면 프로세스별 상호작용하는 파일들과 memmap, minidump 등 다양한 결과 확인 및 유의미한 분석 가능
  - 메모리에서 실행 중인 악성코드를 빠르게 찾을 수도 있겠다.  
→ Windows Defender, Virustotal 활용

이름	수정한 날짜	유형	크기
FTK Imager.exe	2024-02-13 오전 7:43	응용 프로그램	22,022KB
GDI32.dll	2024-02-13 오전 7:43	응용 프로그램 확장	148KB
gdi32full.dll	2024-02-13 오전 7:43	응용 프로그램 확장	1,027KB

# ETC

- <https://github.com/evild3ad/MemProcFS-Analyzer>
- <https://github.com/LDO-CERT/orochi>
- [redline](#)
- AXIOM
  - <https://www.magnetforensics.com/resources/magnet-axiom-2-0-memory-analysis/>
  - <https://blog.system32.kr/74>
- The Art of Memory Forensics - 번역본
  - Volatility 핵심 개발팀이 쓴 책이다.
- awesome-memory-forensics
- Extracting Windows command line details from physical memory

# 참고자료

메모리 덤프 기반 포렌식을 활용한 개인 식별정보 노출 분석

<https://cpuu.postype.com/post/9993241>

<https://cpuu.postype.com/post/11807930>

<https://bitofhex.com/2018/04/29/volatility-and-tor/>

<https://bitofhex.com/2018/05/10/memory-forensics-tor-part-two/> - bulk\_extractor 활용 사례

# 과제

- [WDF 디지털 포렌식 챌린지](#)
  - 주어진 이미지 파일을 분석하고 시나리오에서 요구하는 내용 찾아보기
- [Forensic CTF: Baud.. James Baud..](#)
  - [CTF Questions](#)에서 요구하는 정답 찾아보기
- 지금까지 배운 윈도우 아티팩트 분석, 카빙 등 다양한 방법으로 접근해보기