

# 7주차 스터디

6주차 문제 풀이, 악성코드 유사도 분석

# 6주차 문제 풀이

- [WDF 디지털 포렌식 챌린지](#)
  - 주어진 이미지 파일을 분석하고 시나리오에서 요구하는 내용 찾아보기
- [Forensic CTF: Baud.. James Baud..](#)
  - [CTF Questions](#)에서 요구하는 정답 찾아보기

# WDF 디지털 포렌식 챌린지

최고수 분석관은 디지털 포렌식 경력 7년차의 팀장이다. 그는 며칠 전 성범죄 수사전담반의 나 강력 수사관부터 디지털 포렌식 분석 의뢰를 받았다.

나강력 수사관으로부터 확인된 내용은 다음과 같다.

- 최근 불법 음란물 온라인 판매에 대한 단속을 강화하고 있는 가운데, 인터넷 커뮤니티 D 사이트로부터 수사의뢰를 받음
- 아이피 추적 결과 (주)투스타커피에서 음란물 판매 관련 글을 게시한 것으로 확인되어, 압수수색을 실시함
- 압수수색 결과, 문제의 글을 게시한 날짜(2020년 10월 21일)에는 전직 직원이었던 ‘홍길동’만 사무실에 근무하였던 것으로 확인됨
- 홍길동이 사용하였던 노트북은 회사 기밀 유출 건과 관련하여 민간 포렌식 업체에서 분석을 수행한 바 있으며, 포렌식 분석 이후에는 해당 노트북을 중고시장에 매각하여 그 행방을 알 수 없음
- 다만 이전에 분석을 담당하였던 민간 포렌식 업체에 분석 결과물이 남아 있음을 확인하고, 추가 압수수색을 실시하여 당시 분석 결과물 중에 하나인 가상머신에 관한 컨테이너 파일을 확보 함

최고수 팀장은 위 가상머신 컨테이너 파일에 대한 분석을 디지털 포렌식 경력 2년차인 김신참 분석관에게 배정하였다.

# WDF 디지털 포렌식 챌린지

김신참 분석관은 나강력 수사관과 혐의자 홍길동에 대한 조사 상황을 확인하였고, 다음과 같은 사실을 확인하였다.

- 홍길동은 현재 인터넷 도박 혐의로 구속되어 수감중인 상태이며, 관련 건으로 수사중인 상태임
- 홍길동은 과거 (주)투스타커피에 직원으로 근무한 바 있었으며, 경쟁회사에 재취업하면서 회사 기밀을 유출하여 형사처벌을 받은 사실 있음
- 홍길동은 평소 IT쪽에 관심이 많아 가상머신을 공부한 적이 있으나, 주로 퇴근 후에 IT 공부 차원에서 사용하였을 뿐, 근무시간에 가상머신을 사용한 적은 없다고 주장함
- 홍길동은 문제가 된 음란물을 인터넷 커뮤니티에 게시한 사실이 전혀 없다고 주장함

참고 : 고양이 그림 --> 아동 음란물, 토끼 그림 --> 성인음란물

나강력 수사관의 분석요구사항은 다음과 같다.

- 가상머신의 실제 사용자 확인
- 2020년 10월 21일 전후의 혐의자의 컴퓨터 사용 이력 분석
- 아동 음란물 등 음란물 유포, 판매 등 흔적 확인
- 기타 혐의자와 관련된 특이정황

# Forensic CTF: Baud.. James Baud..

**SCENARIO:** You're on deck to investigate the high profile hack of a celebrity. Your client provided two screenshots of pop-up message boxes he saw on his system, after which he noticed several vital files were deleted from his system.

1. Whose computer is this evidence from?
2. Who is the other actor?
3. What email service are they using (include TLD)?
4. What makes this email service difficult to analyze?
5. What is the email address of the user?
6. What email address does he correspond with?
7. What type of file is the payload?
8. What is the first Google search the user made about the other individual?
9. What is the second Google search the user made about the other individual?
10. What is the third Google search the user made about the other individual?
11. What IP address was used by the attacker for C2?
12. What is the exact name of the payload?
13. What is the first time the user logged into their email (MM/DD/YYYY H:MM:SS AM/PM)?
14. What is the mail server name used to send these messages?
15. What is the UTC time of the initial email (as stated in the email header)?
16. What is the email subject of the first threatening email sent by the user?
17. What insult does the other individual use in his response?

# Forensic CTF: Baud.. James Baud..

- todo

# Forensic CTF: Baud.. James Baud..

- todo

# 정적 분석, 동적 분석

- 정적 분석<sup>1</sup>: 실행 없이 소프트웨어를 분석하는 것
  - 소스코드 기반<sup>2</sup>: code smell<sup>3</sup>이나 문제가 될 수 있는 소스코드를 탐지
  - 바이너리 기반: 바이너리에 존재하는 어셈블리 코드를 분석
  - [IDA Pro](#), [Binary Ninja](#), [Cutter](#), [Ghidra](#), [Decompiler Explorer](#) 외에도 [bytecode-viewer](#), [jd-gui](#), [jadx](#), [dnSpy](#), [CyberChef](#), [SSView](#) 등 분석 타겟, 용도에 따라서 사용하는 도구가 다르다.
- 동적 분석<sup>4</sup>: 실제 또는 가상 환경에서 프로그램을 실행하여 행위 분석
  - 자동 분석: [cuckoo sandbox](#), [CAPEv2](#), [hybrid-analysis](#), [intezer](#) 등 - [malware-tools](#) 참고
  - 수동 분석: [IDA Pro](#), [x64dbg](#), [ollydbg](#), [Immunity Debugger](#), [Sysinternals](#), [SysmonTools](#) 등
- 개발, 보안 관점에 따라서 해석이 조금 다를 수 있지만 기본적인 개념은 같다.

---

<sup>1</sup> [https://en.wikipedia.org/wiki/Static\\_program\\_analysis](https://en.wikipedia.org/wiki/Static_program_analysis)

<sup>2</sup> [https://owasp.org/www-community/controls/Static\\_Code\\_Analysis](https://owasp.org/www-community/controls/Static_Code_Analysis)

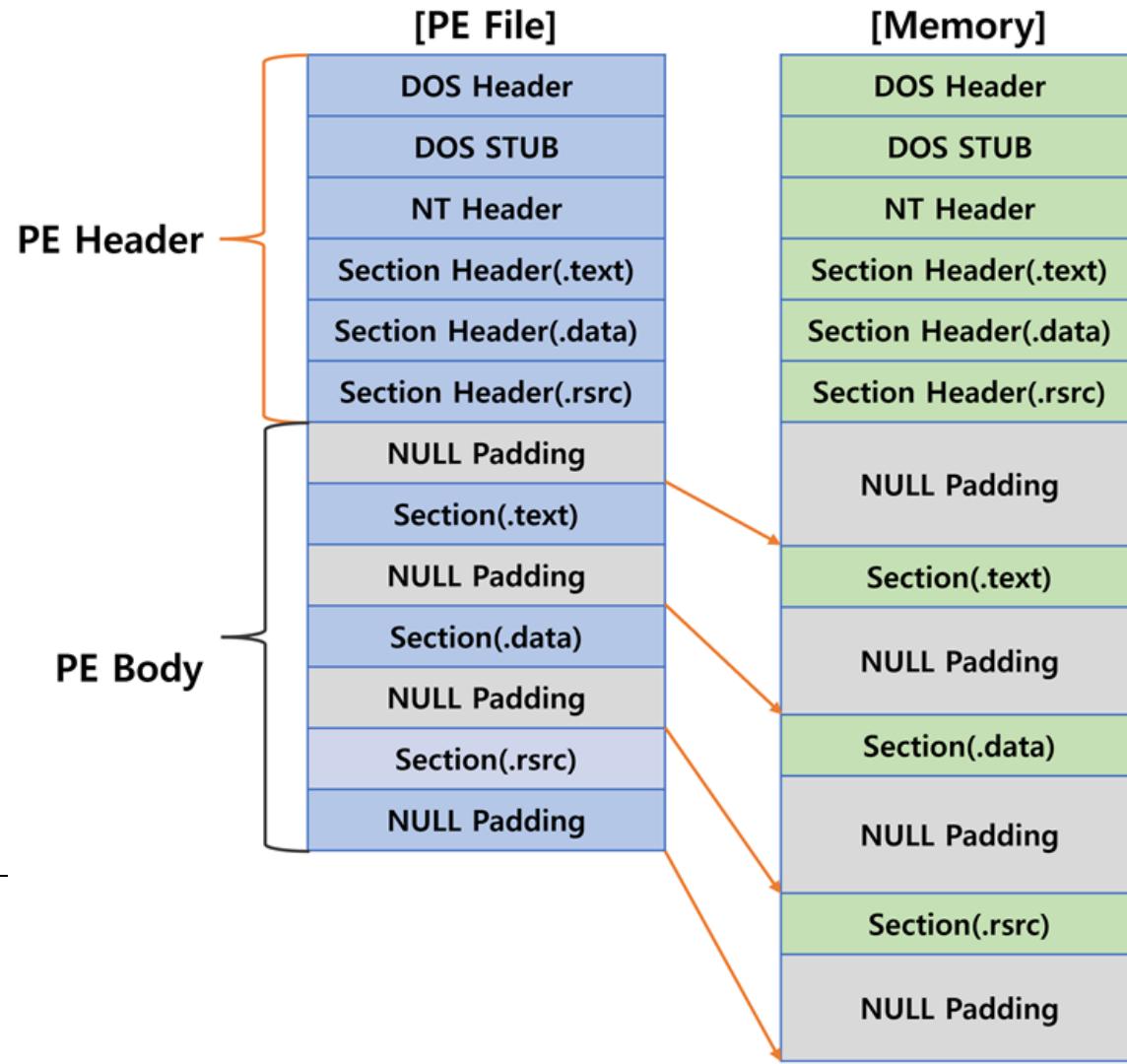
<sup>3</sup> [https://en.wikipedia.org/wiki/Code\\_smell](https://en.wikipedia.org/wiki/Code_smell)

<sup>4</sup> [https://en.wikipedia.org/wiki/Dynamic\\_program\\_analysis](https://en.wikipedia.org/wiki/Dynamic_program_analysis)

<sup>5</sup> <https://www.vmray.com/glossary/dynamic-analysis/>

# PE (Portable Executable)<sup>1</sup>

- 윈도우 운영체제에서 사용되는 실행 파일, DLL 등을 위한 파일 형식<sup>2</sup>
- 파일의 어느 청크가 메모리 어디 부분에 적재되어야 하는지, 프로그램 코드 중 어느 부분에서 프로그램 실행을 시작해야 하는지 등을 정의하고 있다.<sup>4</sup>
- .text 섹션에 있는 프로그램 코드로 리버싱을 진행 한다.
- IAT<sup>2</sup>를 분석하여 어떤 API가 사용되는지 확인하는 것도 의미있는 분석이 될 수 있다.



<sup>1</sup> <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format>

<sup>2</sup> [https://ko.wikipedia.org/wiki/PE\\_포맷](https://ko.wikipedia.org/wiki/PE_포맷)

<sup>3</sup> <https://hwanstory.kr/@kim-hwan/posts/Windows-Portable-Executable-File-Format>

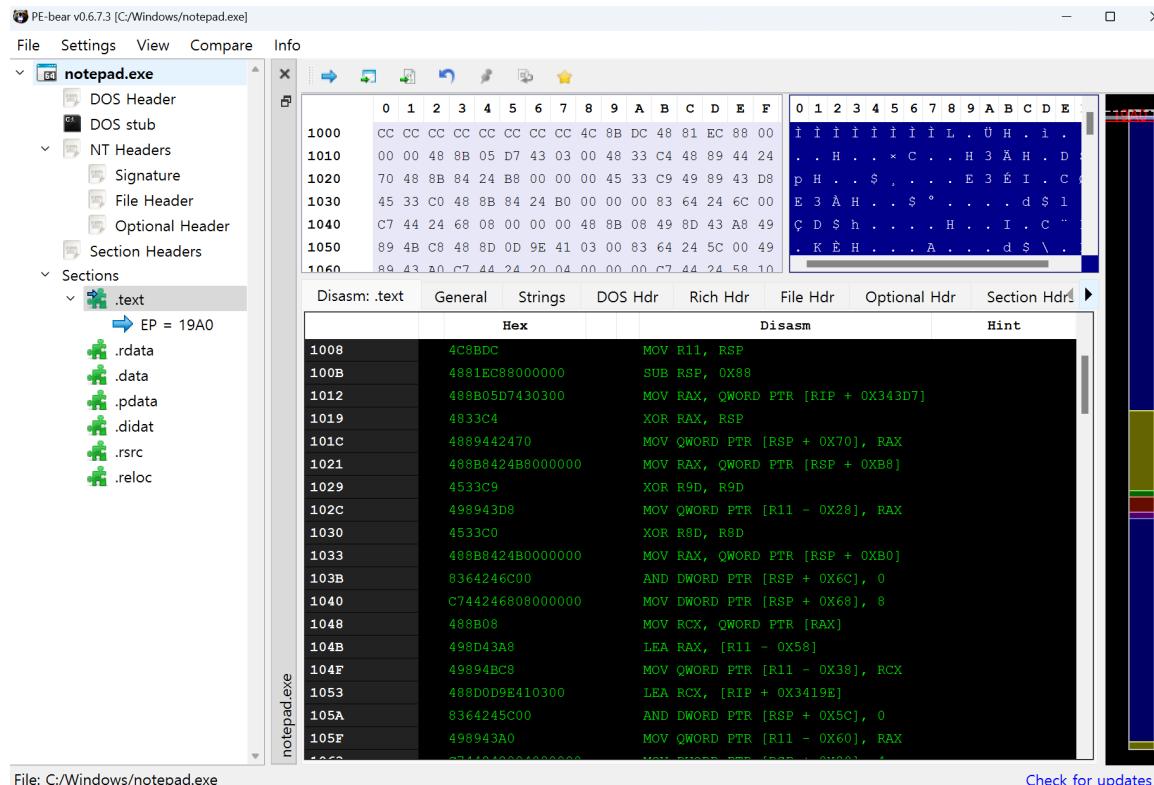
<sup>4</sup> 멀웨어 데이터 과학 - p2

# **PE (Portable Executable)**

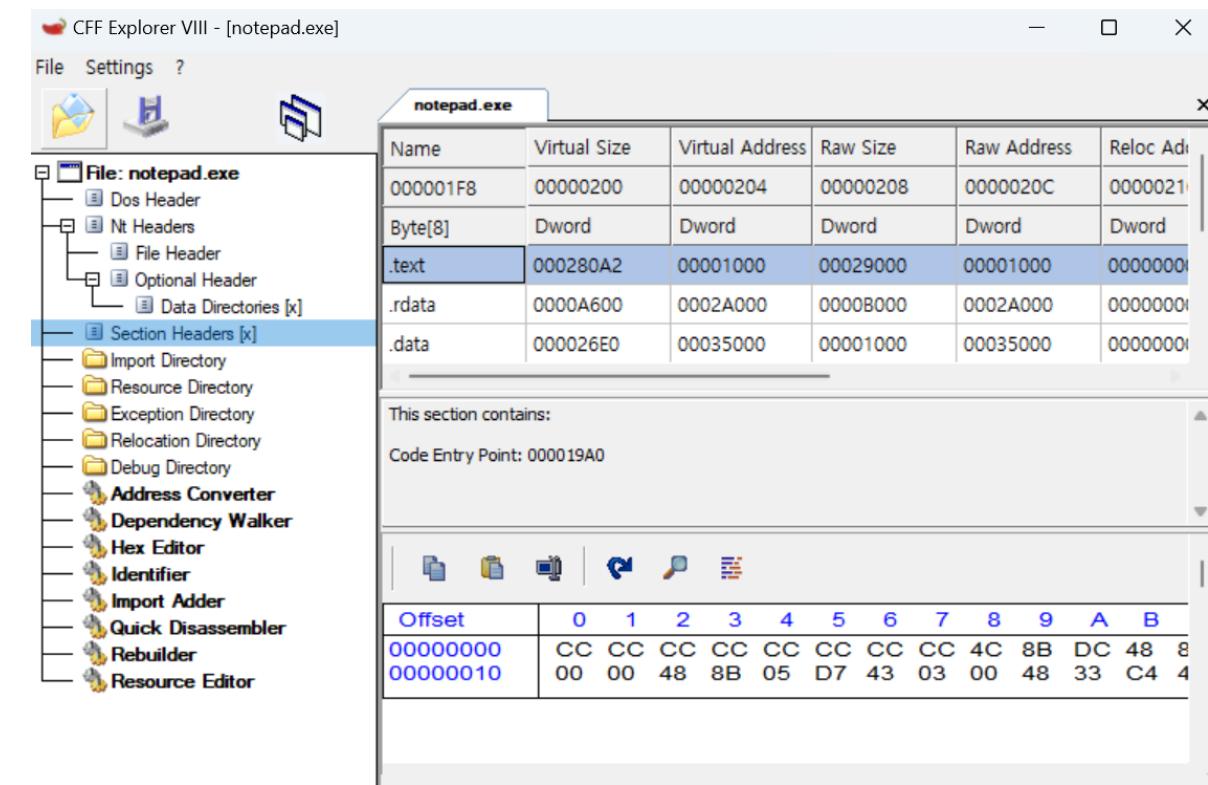
<sup>1</sup> <https://github.com/corkami/pics/blob/master/binary/pe101/pe101ko.png>

# PE (Portable Executable)

- PE-bear, CFF Explorer, PEView, FileInsight-plugins, PEStudio

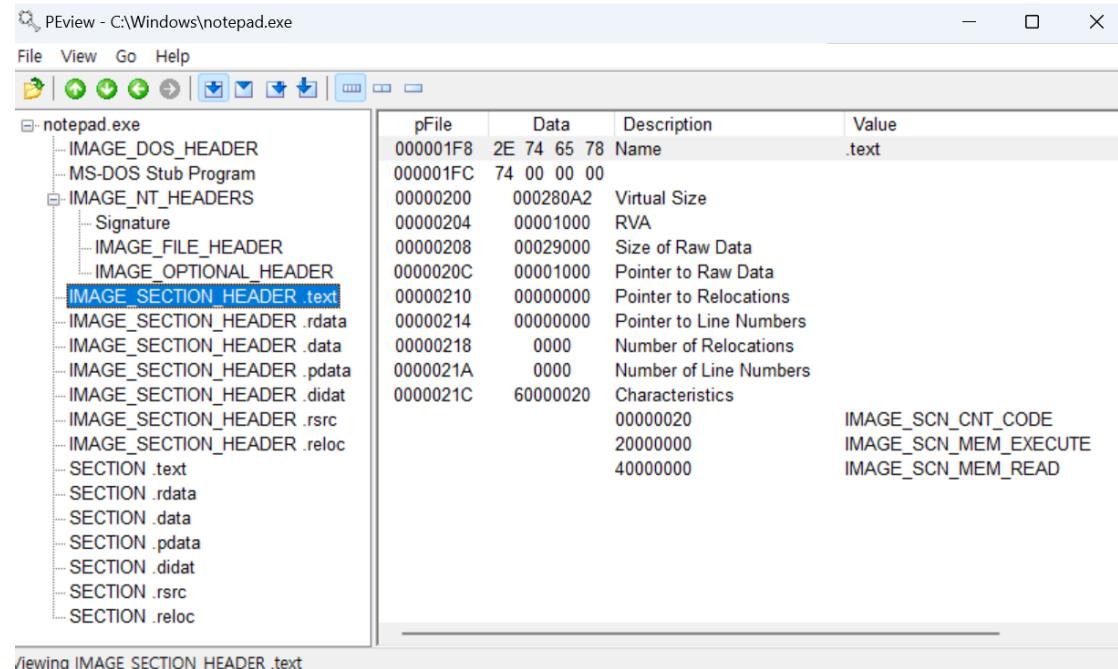


PE-bear

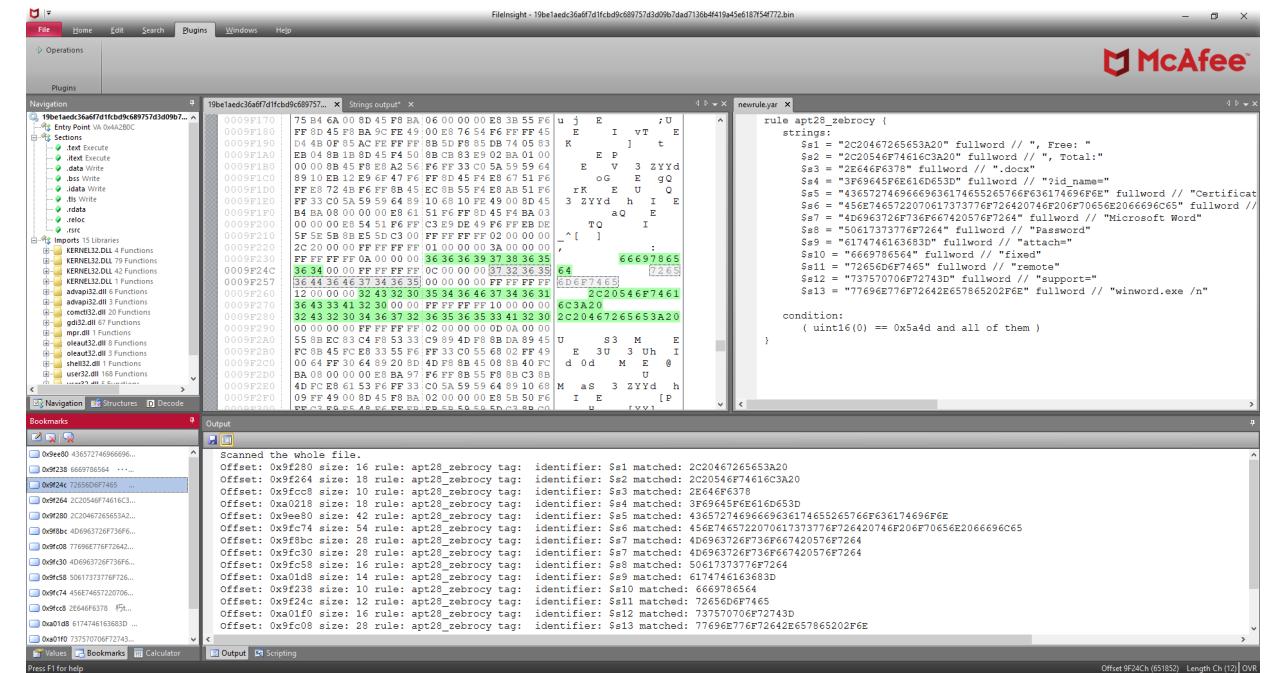


CFF Explorer

# PE (Portable Executable)



PEView



FileInsight-plugins

# pefile

```
pip install pefile
```

```
import pefile
pe = pefile.PE('sample.exe')
for section in pe.sections:
    print(section.Name, hex(section.VirtualAddress),
          hex(section.Misc_VirtualSize), section.SizeOfRawData)
```

```
import pefile
for entry in pe.DIRECTORY_ENTRY_IMPORT:
    print(entry.dll)
    for imp in entry.imports:
        print('\t', hex(imp.address), imp.name)
```

- PE 파일의 section들의 데이터를 출력하는 코드, 사용하는 API들을 출력하는 코드

<sup>1</sup> <https://pefile.readthedocs.io/en/latest/usage/UsageExamples.html#iterating-through-the-sections>

<sup>2</sup> <https://pefile.readthedocs.io/en/latest/usage/UsageExamples.html#listing-the-imported-symbols>

# 패커 (Packer), 프로텍터 (Protector)

- 패커: 실행 파일을 압축한 후 실행될 때 메모리에 스스로 압축을 풀어서 원본 코드를 다시 만드는 방법
  - 리버싱을 어렵게 한다는 점에서 프로텍터와의 구분이 모호한 경우도 있다.
  - 대부분 악의적인 목적으로 사용된다고 한다.<sup>1</sup>
- 프로텍터: 프로그램 변조, 리버싱 등을 방지하기 위한 방법
  - 일반적으로 패킹, 난독화 기법<sup>3</sup>들이 모두 포함되어 있다.<sup>1</sup>
- Detect It Easy, Exeinfo PE, PEID 등의 도구를 사용하여 패커, 프로텍터 적용 유무를 확인할 수 있다.
- UniExtract2, VMUnpacker 등 언패커를 사용하는 방법도 있지만 실패한다면 수동으로 메모리 덤프를 통해 추출해야 한다.

---

<sup>1</sup> <https://www.malwarebytes.com/blog/news/2017/03/explained-packer-crypter-and-protector>

<sup>2</sup> [https://en.wikipedia.org/wiki/Executable\\_compression](https://en.wikipedia.org/wiki/Executable_compression)

<sup>3</sup> [https://en.wikipedia.org/wiki/Obfuscation\\_\(software\)](https://en.wikipedia.org/wiki/Obfuscation_(software))

# 패커 (Packer), 프로텍터 (Protector)

- Themida Protector를 사용하는 카카오톡

The screenshot shows the 'Detect It Easy v3.09 [Windows 10 Version 2009] (x86\_64)' interface. The file path is 'C:\Program Files (x86)\Kakao\KakaoTalk\KakaoTalk.exe'. The file type is 'PE32' (selected in dropdown) and the size is '24.85 MiB'. The search mode is set to '자동적 인' (Automatic). The analysis results for the PE32 section are as follows:

분석 항목	결과	상세
운영 시스템	Windows(Vista)[I386, 32비트, GUI]	S ?
링커	Microsoft Linker(14.33.31629)	S ?
컴파일러	Microsoft Visual C/C++(19.33.31630)[C++]	S ?
언어	C/C++	S ?
도구	Visual Studio(2022 version 17.3)	S ?
서명 도구	Windows Authenticode(2.0)[PKCS #7]	S ?
프로텍터	Themida/Winlicense(3.XX)	S ?
오버레이	Binary	
인증서	WinAuth(2.0)[PKCS #7]	S ?

<sup>1</sup> <https://hummingbird.tistory.com/6468>

# IOC(Indicator Of Compromise, 침해지표)<sup>1 2 3</sup>

- 침해사고 분석을 통해 얻어낸 정보들을 의미한다.
  - 악성 파일의 해시값, IP 주소, 악성 URL 및 도메인 이름 등

[IOC]

MD5

A25ACC6C420A1BB0FDC9456B4834C1B4  
393CBA61A23BF8159053E352ABDD1A76

C2

hxxp://hmcks.realma.r-e[.]kr/gl/ee.txt

95.187.5.53	IPv4	IP Watchlist
95.187.70.15	IPv4	IP Watchlist
95.187.99.232	IPv4	IP Watchlist
95.87.25.221	IPv4	IP Watchlist
AB4ABEE83FFD526F1975EB48DBDBC812	md5	File Hash Watchlist
E95C318D1B1906D57471BB524FFF128356C160132D4	SHA256	File Hash Watchlist
3FDF856B6FBCB23E7C3372A3F53CE26C0FE6DE77	SHA1	File Hash Watchlist
5D29DFE2EA9CA8DA3FF7A14FB20C5E86	md5	File Hash Watchlist
8F4FC2E10B6EC15A01E0AF24529040DD	md5	File Hash Watchlist
B48DC6ABCD3AEFF8618350CCBDC6B09A	md5	File Hash Watchlist
3CEEE0BE85D24D911B9C02714817774C	md5	File Hash Watchlist
C:\Windows\System32\NPf.SYS	filepath	File Hash Watchlist
C:\Windows\System32\NETPLG.LOG	filepath	File Hash Watchlist
C:\Windows\System32\CATROOT2\WEDBCHK.LOG	filepath	File Hash Watchlist
C:\Windows\System32\MIMEFILTER.XML	filepath	File Hash Watchlist
C:\WORK\CATROOT2	filepath	File Hash Watchlist
C:\Windows\System32\CATROOT2\{12CD0A1D-filepath	filepath	File Hash Watchlist
C:\Windows\System32\CATROOT2\{A750E6C3-filepath	filepath	File Hash Watchlist

<sup>1</sup> <https://www.trendmicro.com/vinfo/us/security/definition/indicators-of-compromise>

<sup>2</sup> <https://www.crowdstrike.com/cybersecurity-101/indicators-of-compromise/>

<sup>3</sup> [https://en.wikipedia.org/wiki/Indicator\\_of\\_compromise](https://en.wikipedia.org/wiki/Indicator_of_compromise)

<sup>4</sup> <https://asec.ahnlab.com/ko/50275/>

<sup>5</sup> <https://www.cisa.gov/news-events/alerts/2017/06/13/hidden-cobra-north-koreas-ddos-botnet-infrastructure> - TA-17-164A\_csv.csv

# 악성코드 스캔

- [Virustotal](#)에서 확인하기
  - 무작정 Virustotal에 업로드하는 것은 좋지 않다.
    - [Virustotal Intelligence](#)<sup>1,2</sup>에서 누구나 샘플을 다운로드할 수 있기 때문
  - [OpenHashTab](#), [HashMyFiles](#), [Quickhash-GUI](#) 등 해시값(MD5, SHA1, ...) 생성 도구를 사용하여 해시값을 검색해보고, 업로드가 안된 파일이면 경우에 따라서 업로드 유무 판단
  - [PEStudio](#)가 해시값으로 Virustotal에 조회해서 보여준다.

engine (0/70)	score (0/70)	date (dd.mm.yyyy)	age (days)
ALYac	-	04.12.2023	72
APEX	-	28.11.2023	78
AVG	-	04.12.2023	72
Acronis	-	28.08.2023	170
AhnLab-V3	-	04.12.2023	72
Alibaba	-	27.05.2019	1724

<sup>1</sup> <https://www.virustotal.com/gui/intelligence-overview>

<sup>2</sup> <https://docs.virustotal.com/docs/virustotal-intelligence-introduction>

# 악성코드 스캔

- Microsoft Defender 활용하기
  - 이전에 포렌식 문제에서 랜섬웨어 추출과 동시에 Defender가 잡아줬던 것처럼 탐지율이 준수하다.
  - 그러나 Defender가 잡았다는 것은 백신 서버에 파일이 저장된 것이기 때문에 악성코드 공유 문제 발생
- ClamAV<sup>1,2</sup> 활용하기
  - 시그니처 데이터베이스를 활용하여 악성코드 패턴을 탐지한다.

```
C:\Users\hyuunnnn\Desktop\clamav-1.3.0.win.x64>freshclam.exe
ClamAV update process started at Thu Feb 15 06:31:10 2024
daily.cvd database is up-to-date (version: 27185, sigs: 2053392, f-level: 90, builder: raynman)
main.cvd database is up-to-date (version: 62, sigs: 6647427, f-level: 90, builder: sigmgr)
bytecode.cvd database is up-to-date (version: 334, sigs: 91, f-level: 90, builder: anvilleg)
```

- 오픈소스이며 PC 내부에서 동작하기 때문에 외부로 업로드되지 않는다.
- 상용엔진에 비해 많이 부족한 것은 사실이다. ClamAV는 단순히 스캐너일 뿐이다.<sup>3</sup>

```
C:\Users\hyuunnnn\Desktop\clamav-1.3.0.win.x64>clamscan.exe "C:\Users\hyuunnnn\Desktop\test\test\240103, P3"
Loading: 14s, ETA: 0s [=====] 8.69M/8.69M sigs
Compiling: 3s, ETA: 0s [=====] 41/41 tasks
```

<sup>1</sup> <https://docs.clamav.net>

<sup>2</sup> <https://youtu.be/9gQXBUJ0sHE>

<sup>3</sup> <https://blog.clamav.net/2011/03/top-5-misconceptions-about-clamav.html>

# 문자열 검사

- 문자열을 추출하여 어떤 기능을 하는지 유추 가능
  - `strings`: 바이너리 파일에서 ascii, unicode 문자를 추출한다.

```
$ strings 052596A8380EDEEE8E211B504F44E133
```

```
...
File [%s] : sent %d bytes !
IP address not found
Socket error : %d
S^127.0.0.1
Login failure
Login Success!
Command is [%s]
Server found
No Server !
communication is started !
Disconnected from Controller !
```

<sup>1</sup> [Trojan/Win32.Phandoor.R174803](#)

<sup>2</sup> [악성코드 중심의 침해대응 매트릭스 모델링 - 2019 인텔리전스 보고서](#), S 트랜스폼

## FLOSS STACK STRINGS (3)

```
ROOT\CIMV2  
DeviceId  
SELECT * FROM Win32_PnPEntity
```

## FLOSS TIGHT STRINGS (0)

## FLOSS DECODED STRINGS (31)

```
[\@HTP(LPH  
AXeA  
paAXeA  
pbA8  
Kern  
el32.dll  
GetS  
tartupInfoA  
Heap  
Create  
Exit  
Process  
GetP  
rocessHeap
```

# 문자열 검사

- **floss:** strings 기능에 더하여 난독화된 문자열을 탐지하고 복호화 루틴을 찾아서 에뮬레이팅하는 과정을 수행한다.  
→ 탐지율이 높진 않은 것 같다. (그래도 의미있는 정보를 얻을 가능성도 있기 때문에 밑져야 본전 마인드로 사용하면 되겠다.)

The screenshot shows a NetworkMiner interface. On the left, under 'Network Communication', there are sections for 'IP Traffic' (listing three TCP connections) and 'Memory Pattern IPs' (listing three IP addresses). On the right, a sidebar titled 'FLOSS STACK STRINGS (6)' lists the following strings:  
113.160.112.125  
213.210.194.59  
196.45.177.52  
199.26.11.18  
S0VA?0VAK  
EDf3

9394078671922de6b5cd194e3581ec46<sup>1</sup>

<sup>1</sup> [HiddenCobra\\_BANKSHOT - AdobeARM.exe](#)

# 문자열 검사

- **CAPA**: 실행 파일의 기능, 행위를 탐지한다. → 특정 API 사용 여부나 payload 등을 탐지하여 어떤 행위가 예상되는지 알려준다. (어디에 초점을 두고 분석해야 할지 가이드를 잡아준다.)

CAPABILITY	NAMESPACE
check for time delay via GetTickCount	anti-analysis/anti-debugging/debugger-detection
check for time delay via QueryPerformanceCounter	anti-analysis/anti-debugging/debugger-detection
contain obfuscated stackstrings	anti-analysis/obfuscation/string/stackstring
receive data (5 matches)	communication
send data (5 matches)	communication
connect to URL	communication/http/client
create HTTP request	communication/http/client
get socket status	communication/socket
initialize Winsock library	communication/socket
set socket configuration	communication/socket
receive data on socket (5 matches)	communication/socket/receive
send data on socket (5 matches)	communication/socket/send
connect TCP socket	communication/socket/tcp
create TCP socket	communication/socket/tcp
create UDP socket (5 matches)	communication/socket/udp/send
act as TCP client	communication/tcp/client
contain a resource (.rsrc) section	executable/pe/section/rsrc
extract resource via kernel32 functions	executable/resource
contain an embedded PE file	executable/subfile/pe
get file size	host-interaction/file-system/meta
move file	host-interaction/file-system/move
read file	host-interaction/file-system/read
resolve DNS (5 matches)	host-interaction/network/dns/resolve
get networking interfaces	host-interaction/network/interface
create service	host-interaction/service/create
start service	host-interaction/service/start
create thread (3 matches)	host-interaction/thread/create
terminate thread	host-interaction/thread/terminate
link function at runtime	linking/runtime-linking
linked against ZLIB	linking/static/zlib
persist via Windows service	persistence/service

<sup>1</sup> <https://youtu.be/cbmMstmsq9c?t=319>

# 문자열 검사

```
self delete
namespace anti-analysis/anti-forensic/self-deletion
author michael.hunhoff@mandiant.com, @mr-tz
scope function
att&ck Defense Evasion::Indicator Removal::File Deletion [T1070.004]
mbc Defense Evasion::Self Deletion::COMSPEC Environment Variable [F0007.001]
function @ 0x4027B0
and:
or:
match: host-interaction/process/create @ 0x4028AF
or:
api: CreateProcess @ 0x402914
or:
regex: /del\s*/S/
- "@echo off\r\n:n:D1\r\n\ndel /a %1\r\nif exist %1 goto D1\r\n\ndel /a %0" @ 0x4028B5

receive data (3 matches)
namespace communication
author william.ballenthin@mandiant.com
scope function
mbc Command and Control::C2 Communication::Receive Data [B0030.002]
description all known techniques for receiving data from a potential C2 server
function @ 0x401B4D
or:
match: receive data on socket @ 0x401B4D
or:
api: recv @ 0x401B7D
api: recv @ 0x401B7D
```

```
    strcpy(Destination, aEchoOffD1DelA1);
    v1 = strlen(Destination);
    WriteFile(FileA, Destination, v1, &NumWritten);
    CloseHandle(FileA);
    memset(&StartupInfo, 0, sizeof(StartupInfo));
    StartupInfo.dwFlags = 1;
    StartupInfo.wShowWindow = 0;
    CreateProcessA(0, CommandLine, 0, 0, 0,
                  0, 0, 0, 0, &Process, &Thread);
}
return 1;
}
000028B5 sub_4027B0:49 (4028B5)
aEchoOffD1DelA1 db '@echo off',0Dh,0Ah ; DATA XREF
db ':D1',0Dh,0Ah
db 'del /a %1',0Dh,0Ah
db 'if exist %1 goto D1',0Dh,0Ah
db 'del /a %0',0Dh,0Ah
align 4
```

7FE80CEE04003FED91C02E3A372F4B01<sup>1</sup>

<sup>1</sup> Backdoor.Win32.Joanap - report, wikipedia

# PDB (Program Data Base)

- PDB path는 디버그 모드로 빌드할 때 생성된다. 이때 프로젝트명은 무엇인지, 제작자는 누구인지 등 악성코드에 대한 정보를 얻을 수 있다.
- KEEPER CTF IR-1 문제의 랜섬웨어 파일에 PDB path를 통해 제작자와 랜섬웨어 이름 확인 가능
  - CAPA를 사용한 결과이며, `strings` 명령어를 통해서도 확인할 수 있다.

```
contains PDB path
namespace executable/pe/pdb
author moritz.raabe@mandiant.com
scope file
regex: /:\\.*/.pdb/
- "C:\\Users\\hyuunnnn\\Downloads\\hidden-tear-master\\hidden-tear-master\\hidden-tear\\hidden-tear\\obj\\Debug\\hidden-tear.pdb" @ file+0x1AAD8
```

```
lSystem.Resources.ResourceReader, mscorelib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089#System.Resources.RuntimeResourceSet
PADPAPD
RSDS[
C:\Users\hyuunnnn\Downloads\hidden-tear-master\hidden-tear-master\hidden-tear\hidden-tear\obj\Debug\hidden-tear.pdb
_CorExeMain
mscoree.dll
```

<sup>1</sup> Visual Studio 디버거에서 기호 파일(.pdb) 및 소스 파일 지정(C#, C++, Visual Basic, F#)

<sup>2</sup> <https://www.mandiant.com/resources/blog/definitive-dossier-of-devilish-debug-details-part-one-pdb-paths-malware>

# PDB (Program Data Base)

## 라이플(Rifle)

위협그룹을 프로파일링 하기 위한 시작점이 된 악성코드 샘플의 PDB<sup>\*</sup> 경로에 다음과 같이 rifle이라는 문자열이 포함되어 있었다.

\* 비주얼스튜디오를 이용한 프로그램 개발시 디버깅에 필요한 프로그램데이터베이스

---

E:\Data\My Projects\Troy Source Code\tcvp1st\rifle\Release\rifle.pdb

---

위협그룹이나 침해사고 분석보고서를 공개할 때 코드명을 사용하는 것을 자주 접할 수 있다. 군사작전 등에서 기밀유출에 대비하거나 작전명에 의미를 부여해 임무와 지령을 보다 명확하게 하기 위해 코드명을 사용하는 것처럼, 사이버테러(전쟁)와 관련된 분석과정에도 이런 고유한 코드명을 부여한다. 사이버 공간의 주요 침해사고들도 이와 비슷하게 하나의 침해사고를 오퍼레이션(작전), 일련의 연속된 침해사고들을 지정해 캠페인으로 부른다.

---

<sup>1</sup> 악성코드 중심의 침해대응 매트릭스 모델링 - 2019 인텔리전스 보고서, p5

# Rich Header<sup>1 2 3</sup>

- Microsoft 툴체인을 사용할 때 빌드 환경에 대한 정보를 담고 있다. (컴파일러, 링커 정보 등)
  - 공식적으로 문서화되지 않은 영역이다.
- 쉽게 조작이 가능한 데이터이므로 참고만 해야한다. ( PDB 정보나 컴파일 시간 정보 등도 마찬가지)
  - 평창 올림픽에 장애를 일으켰던 OlympicDestroyer 악성코드와 북한 배후의 그룹인 BlueNoroff 가 만들었던 악성코드의 Rich Header 정보가 동일하여 북한 소행이 아닌지 의심했던 적이 있다.<sup>2</sup>
  - 그러나 북한 배후의 공격이 아닌 것으로 확인되었다.<sup>4</sup>

---

<sup>1</sup> Rich Headers: leveraging this mysterious artifact of the PE format

<sup>2</sup> The devil's in the Rich header

<sup>3</sup> <https://github.com/RichHeaderResearch/RichPE>

<sup>4</sup> 평창동계올림픽 해킹, 어떻게 시작됐나

```

def get_imphash(self):
    impstrs = []
    exts = ["ocx", "sys", "dll"]
    if not hasattr(self, "DIRECTORY_ENTRY_IMPORT"):
        return ""
    for entry in self.DIRECTORY_ENTRY_IMPORT:
        if isinstance(entry.dll, bytes):
            libname = entry.dll.decode().lower()
        else:
            libname = entry.dll.lower()
        parts = libname.rsplit(".", 1)
        if len(parts) > 1 and parts[1] in exts:
            libname = parts[0]
        entry_dll_lower = entry.dll.lower()
        for imp in entry.imports:
            funcname = None
            if not imp.name:
                funcname = ordlookup.ordLookup(
                    entry_dll_lower, imp.ordinal, make_name=True
                )
            if not funcname:
                raise PEFormatError(
                    f"Unable to look up ordinal {entry.dll}:{imp.ordinal:04x}"
                )
            else:
                funcname = imp.name
            if not funcname:
                continue
            if isinstance(funcname, bytes):
                funcname = funcname.decode()
            impstrs.append("%s.%s" % (libname.lower(), funcname.lower()))
    return md5(",".join(impstrs).encode()).hexdigest()

```

# imphash (import hashing)

- 실행 파일에 있는 라이브러리/임포트 함수 이름과 특유의 순서를 바탕으로 해시값을 생성하는 방법<sup>1</sup>
- IAT에서 라이브러리 이름과 API 이름을 리스트에 append하고 마지막에 , 를 기준으로 join 한 문자의 조합을 md5로 변환하는 방식으로 구현하고 있다.
- [virustotal](#), [pefile](#)<sup>2</sup> 라이브러리 등으로 확인 가능

<sup>1</sup> 악성코드 분석 시작하기 - p91

<sup>2</sup> [pefile - get\\_imphash](#)

# ssdeep<sup>4</sup>

- Fuzzy hashing (퍼지 해싱)<sup>3</sup>: 유사하지만 완전히 같지 않은 데이터를 탐지하기 위한 기술  
사소한 차이에도 아예 다른 값을 만드는 암호화 해시 함수와는 대조적이다.<sup>1</sup>
  - CTPH (Context Triggered Piecewise Hashing)<sup>12</sup>: 여러 조각으로 분할하여 조각에 대한 해시를 생성한 후 문자열로 구성하는 방법 → ssdeep은 퍼지 해싱 기술을 사용한 도구이며, CTPH 기법을 사용했다.
- 같은 유형의 악성코드가 계속 나오고 있고, 해시값으로 인한 탐지를 피하기 위해 수정하는 경우도 있다고 한다. → 워너크라이 랜섬웨어의 경우 변종 샘플이 약 12000개였다고 한다.<sup>3</sup>

```
sudo apt-get install ssdeep
```

```
hyuunnn@hyuunnn:/mnt/c/Users/hyuunnn/Desktop/ransom$ ssdeep *
ssdeep,1.1--blocksize:hash:hash,filename
3072:yM+lmsOLAIrRuw+mqv9j1MWLQWMTmmsolNIrRuw+mqv9j1MWLQA:x+lDAA4TmDAN,"/mnt/c/Users/hyuunnn/Desktop/ransom/0IAjDNgX0rFO.exe"
3072:yM+lmsOLAIrRuw+mqv9j1MWLQWMTmmsolNIrRuw+mqv9j1MWLQA:x+lDAA4TmDAN,"/mnt/c/Users/hyuunnn/Desktop/ransom/LqozVnjz0VrN.exe"
3072:yM+lmsOLAIrRuw+mqv9j1MWLQmMTmmsolNIrRuw+mqv9j1MWLQA:z+lDAACtM DAN,"/mnt/c/Users/hyuunnn/Desktop/ransom/gviBFQRFUYKg.exe"
3072:yM+lmsOLAIrRuw+mqv9j1MWLQrMTmmsolNIrRuw+mqv9j1MWLQA:x+lDAARTmDAN,"/mnt/c/Users/hyuunnn/Desktop/ransom/ifi2UtAXWPKT.exe"
3072:yM+lmsOLAIrRuw+mqv9j1MWLQmMTmmsolNIrRuw+mqv9j1MWLQA:F+lDAAQtM DAN,"/mnt/c/Users/hyuunnn/Desktop/ransom/p0cZproCkop9.exe"
hyuunnn@hyuunnn:/mnt/c/Users/hyuunnn/Desktop/ransom$ md5sum *
2d8717519cc57e95a877ce354cd383 0IAjDNgX0rFO.exe
2d8717519cc57e95a877ce3b54cd383 LqozVnjz0VrN.exe
03d710b320022ba456f78763ebf4f568 gviBFQRFUYKg.exe
```

<sup>1</sup> Fuzzy hashing

<sup>2</sup> Malware Detection Context Triggered Piecewise Hashes (CTPH)

<sup>3</sup> Playbook of the Week: Discovering Unknown Malware Using SSDeep

<sup>4</sup> How to Identify Malware Similarities with Fuzzy Hashing

# ssdeep

```
hyuunnn@hyuunnn:/mnt/c/Users/hyuunnn/Desktop/ransom$ ssdeep -pb *
0IAjDNgX0rF0.exe matches LqozVnjz0VrN.exe (100)
0IAjDNgX0rF0.exe matches gviBFQRfUYKg.exe (97)
0IAjDNgX0rF0.exe matches ifi2UtAXWPKT.exe (99)
0IAjDNgX0rF0.exe matches p0cZproCkop9.exe (97)

LqozVnjz0VrN.exe matches 0IAjDNgX0rF0.exe (100)
LqozVnjz0VrN.exe matches gviBFQRfUYKg.exe (97)
LqozVnjz0VrN.exe matches ifi2UtAXWPKT.exe (99)
LqozVnjz0VrN.exe matches p0cZproCkop9.exe (97)

gviBFQRfUYKg.exe matches 0IAjDNgX0rF0.exe (97)
gviBFQRfUYKg.exe matches LqozVnjz0VrN.exe (97)
gviBFQRfUYKg.exe matches ifi2UtAXWPKT.exe (97)
gviBFQRfUYKg.exe matches p0cZproCkop9.exe (97)

ifi2UtAXWPKT.exe matches 0IAjDNgX0rF0.exe (99)
ifi2UtAXWPKT.exe matches LqozVnjz0VrN.exe (99)
ifi2UtAXWPKT.exe matches gviBFQRfUYKg.exe (97)
ifi2UtAXWPKT.exe matches p0cZproCkop9.exe (97)

p0cZproCkop9.exe matches 0IAjDNgX0rF0.exe (97)
p0cZproCkop9.exe matches LqozVnjz0VrN.exe (97)
p0cZproCkop9.exe matches gviBFQRfUYKg.exe (97)
p0cZproCkop9.exe matches ifi2UtAXWPKT.exe (97)
```

# TLSH (Trend Micro Locality Sensitive Hash)

- ssdeep의 CTPH와 다른 구현 방식인 LSH (Locality Sensitive Hash)<sup>123</sup> 기법 사용

```
wget https://github.com/trendmicro/tlsh/archive/master.zip -O master.zip  
unzip master.zip  
cd tlsh-master
```

```
hyuunnnn@hyuunnnn:/mnt/c/Users/hyuunnnn/tlsh-master/bin$ ./tlsh -c /mnt/c/Users/hyuunnnn/Desktop/ransom/0IAjDNgXOrF0.exe -r /mnt/c/Users/hyuunnnn/Desktop/ransom -details  
eval /mnt/c/Users/hyuunnnn/Desktop/ransom/0IAjDNgXOrF0.exe T1B62414F1A740C465D8A796B9883BDA7A423A24DDC6C490D3D92FF0B7D723474027C9B  
eval /mnt/c/Users/hyuunnnn/Desktop/ransom/LqozVnjzOVrN.exe T1B62414F1A740C465D8A796B9883BDA7A423A24DDC6C490D3D92FF0B7D723474027C9B  
eval /mnt/c/Users/hyuunnnn/Desktop/ransom/gviBFQRFUYKg.exe T1F42414F1A740C465D8A796B9883BDA7A423A24DDC6C490D3D92FF0B7D723474027C9B  
eval /mnt/c/Users/hyuunnnn/Desktop/ransom/ifi2UtAXWPKT.exe T13C2414F1A740C465D8A796B9883BDA7A423A24DDC6C490D3D92FF0B7D723474027C9B  
eval /mnt/c/Users/hyuunnnn/Desktop/ransom/p0cZproCkop9.exe T1832414F1A740C465D8A796B9883BDA7A423A24DDC6C490D3D92FF0B7D723474027C9B  
eval /mnt/c/Users/hyuunnnn/Desktop/ransom/0IAjDNgXOrF0.exe T1B62414F1A740C465D8A796B9883BDA7A423A24DDC6C490D3D92FF0B7D723474027C9B  
/mnt/c/Users/hyuunnnn/Desktop/ransom/0IAjDNgXOrF0.exe /mnt/c/Users/hyuunnnn/Desktop/ransom/0IAjDNgXOrF0.exe 0  
/mnt/c/Users/hyuunnnn/Desktop/ransom/0IAjDNgXOrF0.exe /mnt/c/Users/hyuunnnn/Desktop/ransom/LqozVnjzOVrN.exe 0  
/mnt/c/Users/hyuunnnn/Desktop/ransom/0IAjDNgXOrF0.exe /mnt/c/Users/hyuunnnn/Desktop/ransom/gviBFQRFUYKg.exe 1  
/mnt/c/Users/hyuunnnn/Desktop/ransom/0IAjDNgXOrF0.exe /mnt/c/Users/hyuunnnn/Desktop/ransom/ifi2UtAXWPKT.exe 1  
/mnt/c/Users/hyuunnnn/Desktop/ransom/0IAjDNgXOrF0.exe /mnt/c/Users/hyuunnnn/Desktop/ransom/p0cZproCkop9.exe 1
```

<sup>1</sup> [https://en.wikipedia.org/wiki/Locality-sensitive\\_hashing](https://en.wikipedia.org/wiki/Locality-sensitive_hashing)

<sup>2</sup> <https://haandol.github.io/2019/05/28/lsh-minhash-explained.html>

<sup>3</sup> <https://www.pinecone.io/learn/series/faiss/locality-sensitive-hashing/>

# impfuzzy<sup>1</sup>

- import API 정보를 사용한다는 imphash와 유사도 탐지에 사용되는 퍼지 해싱 기술을 합친 방법

```
>>> import pyimpfuzzy
>>> hash1 = pyimpfuzzy.get_impfuzzy("0IAjDNgXOrF0.exe")
>>> hash2 = pyimpfuzzy.get_impfuzzy("gviBFQRfUYKg.exe")
>>> hash3 = pyimpfuzzy.get_impfuzzy("ifi2UtAXWPKT.exe")
>>> hash4 = pyimpfuzzy.get_impfuzzy("p0cZproCkop9.exe")
>>> hash1, hash2, hash3, hash4
('3:rGsLdAIEK:tf', '3:rGsLdAIEK:tf', '3:rGsLdAIEK:tf', '3:rGsLdAIEK:tf')
>>> pyimpfuzzy.hash_compare(hash1, hash2)
100
>>> pyimpfuzzy.hash_compare(hash3, hash4)
100
>>> pyimpfuzzy.hash_compare(hash1, hash4)
100
>>> pyimpfuzzy.hash_compare(hash2, hash3)
100
```

<sup>1</sup> Classifying Malware using Import API and Fuzzy Hashing – impfuzzy –

<sup>2</sup> Malware Clustering using impfuzzy and Network Analysis - impfuzzy for Neo4j -

# 관련 자료

[Forensic Malware Analysis: The Value of Fuzzy Hashing Algorithms in Identifying Similarities](#)

[A Ransomware Detection Method Using Fuzzy Hashing for Mitigating the Risk of Occlusion of Information Systems](#)

[Fuzzy-Import Hashing: A Malware Analysis Approach](#)

[Nation-State Threat Actor Attribution Using Fuzzy Hashing](#)

[FUZZY HASHES](#)

	Talos Group	Dell Secure W	Other Name 1	Other Name 2	Other Name 3	Other Name 4	Other Name 5	Other Name 6	Other Name 7	Other Name 8	Rep. of Korea	MITRE AT&CK	Operation 1	Operation 2	Op
Chollima	Group 77	Hastati Group	121,BeagleBoyzBureau121,BeagleBoyz	Unit 121,Unit121	Whois Hacking Team,WHOis Team	NewRomanic Cyber Army Team	ZINC,APT-C-26,UNC2970,UNC577,UNC4736	Appleworm,NICKEL GLADSTONE,COVELLITE,ATK3	Hidden Cobra,Diamond Sleet,Black Artemis,	Nickel Academy,LolZarus		G0032	Troy	Blockbuster	D
hollima	Group 123	ScarCruft	APT37	Red Eyes	Reaper	Venus 121(금성121)	THALLIUM				G0067	Reaper	Erebus	G	
lima			DEV-0530,DEV0530	DarkSeoul,Dark Seoul	PLUTONIUM	Guardian of Peace	GOP	Onyx Sleet	Storm-0530	H0lyGh0st	Andariel	G0138	Dark Hotel	DesertWolf	V
llima														Campaign	

위협 인텔리전스 팀은 악성코드 분석에서 파악한 식별자를 사용해 공격을 분류하고 알려진 위협과 구분 짓는다.  
악성코드 분석을 공격 배후에 누가(경쟁자, 국가 지원 공격 그룹 등) 있는지에 대한 정보를 얻는 데 도움을 준다.

## 악성코드 분석 시작하기 - p33

			APT38										G0082		
hollima			APT38	ElectricFish	BlueNoroff	TA444 (Proofpoint)	<a href="#">COPERNICIUM</a> TAG-71 (Microsoft)				G0082			Far Eastern I	
o														Honeybee	

<sup>1</sup> APT Groups and Operations

<sup>2</sup> [https://en.wikipedia.org/wiki/List\\_of\\_hacker\\_groups](https://en.wikipedia.org/wiki/List_of_hacker_groups)

# 유사도 분석을 왜 할까?

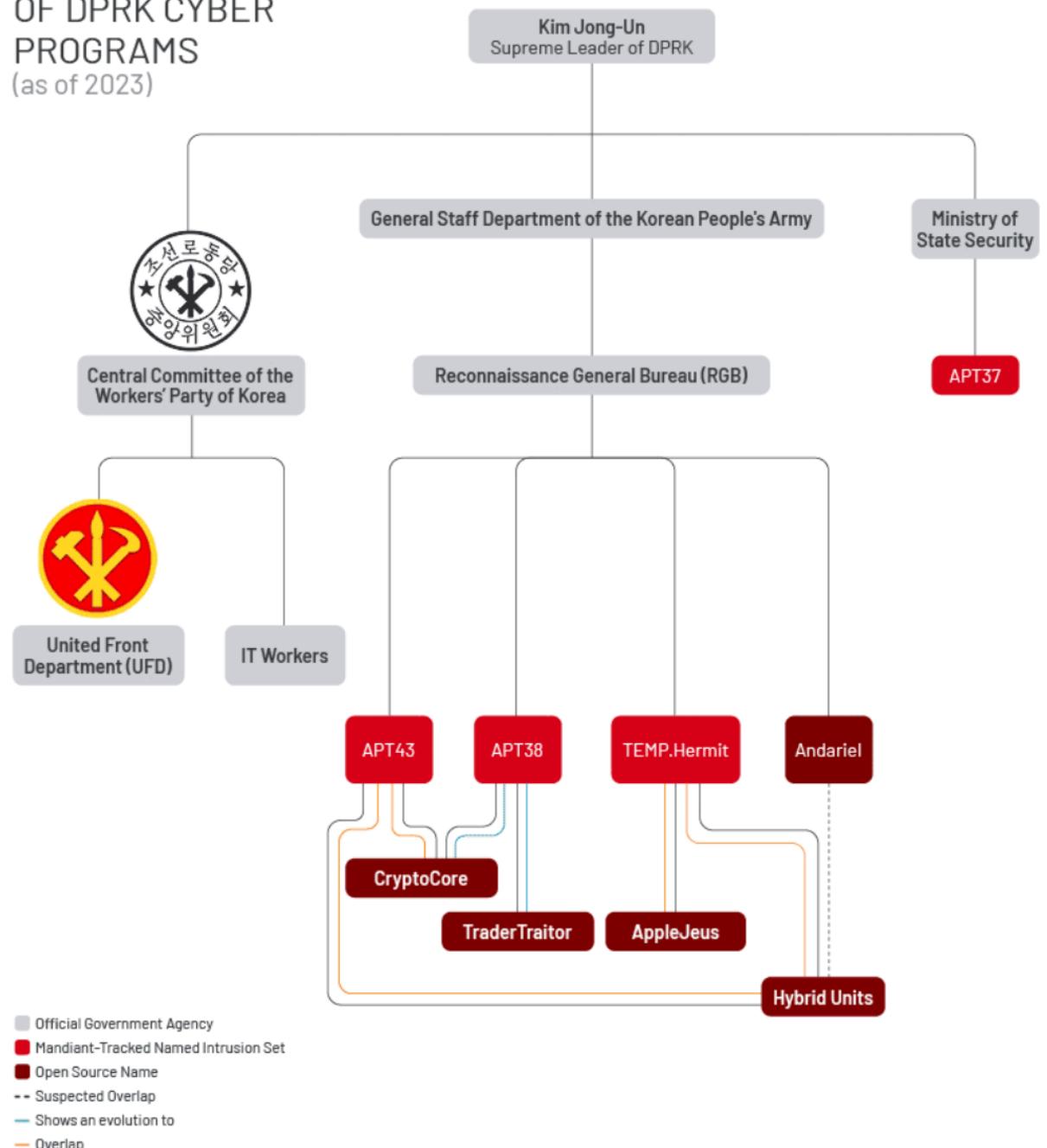
Table 1: CISA and Joint CISA Publications

Publication Date	Title	Description
February 9, 2023	<a href="#">#StopRansomware: Ransomware Attacks on Critical Infrastructure Fund DPRK Malicious Cyber Activities</a>	The NSA, FBI, CISA, Department of Health and Human Services, the Republic of Korea (ROK) National Intelligence Service, and the ROK Defense Security Agency issued a joint Cybersecurity Advisory to highlight ongoing ransomware activity against Healthcare and Public Health Sector organizations and other critical infrastructure sector entities.
July 6, 2022	<a href="#">Joint FBI-CISA-Treasury CSA: North Korean State-Sponsored Cyber Actors Use Maui Ransomware to Target the Healthcare and Public Health Sector</a>	The FBI, CISA, and the Department of the Treasury issued a joint Cybersecurity Advisory to provide information on Maui ransomware, which has been used by North Korean state-sponsored cyber actors since at least May 2021 to target Healthcare and Public Health (HPH) Sector organizations.

<sup>1</sup> <https://www.cisa.gov/topics/cyber-threats-and-advisories/advanced-persistent-threats/north-korea>

<sup>2</sup> <https://www.mandiant.com/resources/blog/north-korea-cyber-structure-alignment-2023>

## ASSESSED STRUCTURE OF DPRK CYBER PROGRAMS (as of 2023)



# 유사도 분석을 왜 할까?

북한



WinRAR 취약점(CVE-2023-38831) 분석  
보고서

2024.01.26



2024년 보안 위협 전망 및 2023년 주요 보안  
이슈 회고  
2024.01.02



ESRC 주간 Email 위협 통계 (11월 넷째주)  
2023.11.28



'금성121' APT 조직, 국내 정치사회적 이슈를  
악용한 공격 진행중!  
2023.09.19



## 자산 관리 프로그램을 악용한 공격 정황 포착 (Andariel 그룹)

Posted By song.th , 2023년 11월 10일

ASEC 분석팀은 Lazarus 그룹과 협력 관계이거나 하위 조직으로 알려진 Andariel 위협 그룹이 최근 특정 자산 관리 프로그램을 이용한 공격을 통해 악성코드를 유포하고 있는 정황을 확인하였다. Andariel 그룹은 최초 침투 과정에서 주로 스파이 피싱 공격이나 워터링 툴 공격 그리고 공급망 공격을 이용하며, 이외에도 악성코드 설치 과정에서 중앙 관리 솔루션을 악용하는 사례도 존재한다. 최근에는 Log4shell 및 InnoRox Agent 등 여러 프로그램에 대한 취약점을 이용하여 국내 다양한 기업군에 공격을 해오고 있다. [1] 이번에 확인된 공격은 국내의 또 다른 자산 관리 프로그램이 사용되었으며, 이외에도...



김수키(Kimsuky)조직의 'Mail Online Security' 프로그램 위장 공격 주의!

2023.06.02



한미(韓美) 합동 보안권고문 : 북한 김수키(Kimsuky) 조직의 싱크탱크, 핵계, 미디어 대…  
2023.06.02



북한 사이버 공격의 현주소와 그 대응 방법  
2023.05.26



2023.05.23



## Lazarus 조직의 Operation Dream Magic

Posted By securityresponseteam , 2023년 10월 13일

Lazarus 조직은 국가가 배후인 것으로 알려진 해킹 조직으로 금전적인 이득, 자료 탈취 등의 목적으로 전세계를 대상으로 꾸준히 해킹하고 있습니다. Lazarus 조직의 이니세이프 취약점을 악용한 워터링 툴을 간단하게 정리하면 언론사의 특정 기사에 악성 링크 삽입, 해당 기사를 클릭하는 기업, 기관이 해킹 대상, 국내 취약한 홈페이지를 C2로 악용 그리고 제한된 범위의 해킹을 위해서 IP 필터링 등을 사용

- 국내 뿐만 아니라 외국에서도 악성코드를 분석하고 리포트를 작성하여 공유하고 있다.

<sup>1</sup> <https://blog.alyac.co.kr/search/ 북한>

<sup>2</sup> <https://asec.ahnlab.com/ko/>

# 유사도 분석을 왜 할까?

## Hacks, Thefts, and Total Amounts Stolen

This Repo	Value Stolen	Incidents		Chainalysis	Value Stolen	Incidents		TRM	Value Stolen	Incidents
2023	\$649,889,146	20		2023	\$1,000,000,000	20		2023	\$600,000,000	?
2022	\$767,638,000	9		2022	\$1,650,000,000	15		2022	\$850,000,000	?
2021	\$317,050,000	13		2021	\$428,800,000	9		2021	\$250,000,000	?
2020	\$307,726,000	8		2020	\$300,000,000	5		2020	\$290,000,000	?
2019	\$191,794,000	9		2019	\$271,000,000	9		2019	\$200,000,000	?
2018	\$430,265,000	15		2018	\$522,000,000	10		2018	\$400,000,000	?
2017	\$109,490,000	6		2017	\$29,000,000	4		2017	\$100,000,000	?
2016	0	0		2016	\$1,500,000	1		2016	0	?
Total:	\$2,773,852,146	80		Total:	\$4,202,300,000	73		Total:	\$2,690,000,000	0

<sup>1</sup> <https://github.com/tayvano/lazarus-bluenoroff-research>

# yara

- 구글이 인수한 `virustotal`에서 만든 유사도 탐지 도구
- 텍스트 또는 패턴을 기반으로 악성코드를 식별하고 분류하기 위해 만들었다고 한다.

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        threat_level = 3
        in_the_wild = true

    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"

    condition:
        $a or $b or $c
}
```

- `strings`에서 제작한 rule을 기반으로 `condition` 영역에 탐지 조건을 정의한다.

---

<sup>1</sup> <https://virustotal.github.io/yara/>

# yara

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        threat_level = 3
        in_the_wild = true

    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"

    condition:
        $a or $b or $c
}
```

- `meta` : rule에 대한 설명, 언제 만들었는지, 제작자 이름 등을 적는 공간이다.
- `strings` : 탐지해야 하는 데이터를 정의하는 공간이다.
- `condition` : `strings`에서 정의한 rule들의 탐지 조건을 정의하는 공간이다.

# yara

Rule 옵션	설명
wide	2바이트로 인코딩된 문자열을 탐지할 때 사용 "Borland" wide → B\x00o\x00r\x00l\x00a\x00n\x00d\x00
ascii	ascii 문자를 탐지 → 보통 wide를 사용할 때 ascii도 같이 사용한다.
nocase	대소문자 구분 없이 탐지 허용 "foobar" nocase → Foobar, FOOBAR, fOoBaR 모두 탐지
fullword	탐지하려는 문자열 사이에 다른 문자열이 포함되어 있는 경우 탐지 X "domain" fullword → mydomain.com 탐지 X, my-domain.com, domain.com 탐지 O

- \$a = "test" nocase ascii wide 처럼 복수의 옵션 사용 가능
  - 대소문자 구분 X, UTF-16 문자열 탐지

# yara

- 유연한 rule 탐지를 위해 와일드카드 기능을 제공한다.
- 표시 방법은 아래와 같이 ? 을 사용한다.

```
rule WildcardExample
{
    strings:
        $hex_string = { E2 34 ?? C8 A? FB }

    condition:
        $hex_string
}
```

- ex:) push eax = 0x50 , push edx = 0x52 , push ebx = 0x53  
→ \$a = { 5? } : push eax , push edx , push ebx 등 모두 탐지 가능

---

<sup>1</sup> YARA Rules for Malware Detection

# yara

- 왜 유사도 분석 분야에서 사용되는걸까?
  - [Virustotal Intelligence \(VTI\)](#)라는 시스템에서 yara rule을 활용하여 `virustotal`에 업로드된 파일들을 실시간으로 탐지할 수 있다.
    - 이전에 만들었던 rule에 탐지된 파일이 새로운 악성코드인지, 재사용되는 코드는 없는지 등 분석<sup>1</sup>
  - 분석 리포트<sup>2</sup>, 블로그, Twitter, Github 등에서 rule이 활발하게 공유되고 있다.
    - yara 관련 도구, rule 관련 정보는 [awesome-yara](#)에서 찾을 수 있다.

## 심화과정

### [YARA-Style-Guide](#), YARA Performance Guidelines

---

<sup>1</sup> <https://intezer.com/blog/threat-hunting/turning-open-source-against-malware/>

<sup>2</sup> HIDDEN COBRA – North Korean Remote Administration Tool: FALLCHILL

# yara

```
xor    ebx, edx
and   ebx, 7F8h
shl   ebx, 14h
shr   edx, 8
or    edx, ebx
lea    ebx, [eax+eax]
xor   ebx, eax
shl   ebx, 4
xor   ebx, eax
mov   ebp, eax
and   ebx, 0FFFFFF80h
shl   ebp, 7
xor   ebx, ebp
shl   ebx, 11h
shr   eax, 8
or    eax, ebx
```

```
xor    edi, edx
and   edi, 7F8h
shl   edi, 14h
shr   edx, 8
or    edx, edi
lea    edi, [eax+eax]
xor   edi, eax
and   cl, al
shl   edi, 4
xor   edi, eax
xor   cl, bl
mov   ebx, eax
and   edi, 0FFFFFF80h
shl   ebx, 7
xor   edi, ebx
shl   edi, 11h
shr   eax, 8
or    eax, edi
```

```
xor    ebx, edx
and   ebx, 7F8h
shl   ebx, 14h
shr   edx, 8
or    edx, ebx
lea    ebx, [eax+eax]
xor   ebx, eax
shl   ebx, 4
xor   ebx, eax
mov   ebp, eax
and   ebx, 0FFFFFF80h
shl   ebp, 7
xor   ebx, ebp
shl   ebx, 11h
shr   eax, 8
inc   esi
or    eax, ebx
```

F90662273DB92AA8DE0ABED37767B911

EE778BE503FDA770EE2F40E51EDFD595

AC3C5383432F8AA6A462F86B1EC00919

레지스터 값이 다르거나 어셈블리 코드가 추가 되어있는 샘플도 존재  
이러한 변종까지 잡기 위해서는 와일드 카드 기능 필요

<sup>1</sup> 악성코드 중심의 침해대응 매트릭스 모델링 - 2019 인텔리전스 보고서, p26

# yara

```
xor    ebx, edx
and   ebx, 7F8h
shl   ebx, 14h
shr   edx, 8
or    edx, ebx
lea    ebx, [eax+eax]
xor   ebx, eax
shl   ebx, 4
xor   ebx, eax
mov   ebp, eax
and   ebx, 0FFFFFF80h
shl   ebp, 7
xor   ebx, ebp
shl   ebx, 11h
shr   eax, 8
or    eax, ebx
```

```
xor    edi, edx
and   edi, 7F8h
shl   edi, 14h
shr   edx, 8
or    edx, edi
lea    edi, [eax+eax]
xor   edi, eax
and   cl, al
shl   edi, 4
xor   edi, eax
xor   cl, bl
mov   ebx, eax
and   edi, 0FFFFFF80h
shl   ebx, 7
xor   edi, ebx
shl   edi, 11h
shr   eax, 8
or    eax, edi
```

```
xor    ebx, edx
and   ebx, 7F8h
shl   ebx, 14h
shr   edx, 8
or    edx, ebx
lea    ebx, [eax+eax]
xor   ebx, eax
shl   ebx, 4
xor   ebx, eax
mov   ebp, eax
and   ebx, 0FFFFFF80h
shl   ebp, 7
xor   ebx, ebp
shl   ebx, 11h
shr   eax, 8
inc   esi
or    eax, ebx
```

F90662273DB92AA8DE0ABED37767B911

EE778BE503FDA770EE2F40E51EDFD595

AC3C5383432F8AA6A462F86B1EC00919

레지스터 값이 다르거나 어셈블리 코드가 추가 되어있는 샘플도 존재  
이러한 변종까지 잡기 위해서는 와일드 카드 기능 필요

<sup>1</sup> 악성코드 중심의 침해대응 매트릭스 모델링 - 2019 인텔리전스 보고서, p26

# yara

32 DA

32 D8

32 D9

xor bl, dl

xor bl, al

xor bl, cl

81 E7 F8 07 00 00

C1 E7 14

C1 EA 08

and edi, 7F8h

shl edi, 14h

shr edx, 8

83 E7 80

C1 E3 07

and edi, 0FFFFFF80h

shl ebx, 7

32 DA

32 D8

32 D9

xor bl, dl

xor bl, al

xor bl, cl

81 E3 F8 07 00 00

C1 E3 14

C1 EA 08

and ebx, 7F8h

shl ebx, 14h

shr edx, 8

83 E3 80

C1 E5 07

and ebx, 0FFFFFF80h

shl ebp, 7

F90662273DB92AA8DE0ABED37767B911

EE778BE503FDA770EE2F40E51EDFD595

<sup>1</sup> 악성코드 중심의 침해대응 매트릭스 모델링 - 2019 인텔리전스 보고서, p26

# yara

Job status	Finished
Rules	rule XOR_transform : XOR { meta: tool = "https://github.com/hy0oun/Hyara" version = "1.4" date = "2018-09-07" MD5 = "F3D5..." }
Creation time	9 7, 2018, 9:25 오후
Start time	9 7, 2018, 11:58 오후
Finish time	9 8, 2018, 3:22 오전
Scanned data	107.7 TB
Scanning speed	9.0 GB/s
Matches	<a href="#">21 Download hashes</a>

Job status	Finished
Rules	rule XOR_transform_wildcard : XOR { meta: tool = "https://github.com/hy0oun/Hyara" version = "1.4" date = "2018-09-07" MD..." }
Creation time	9 7, 2018, 9:24 오후
Start time	9 7, 2018, 11:58 오후
Finish time	9 8, 2018, 3:22 오전
Scanned data	107.7 TB
Scanning speed	9.0 GB/s
Matches	<a href="#">58 Download hashes</a>

[Start new job](#)

와일드카드를  
사용하지 않은 rule

와일드카드를  
사용한 rule

- Magniber , Hermes , GandCrab 랜섬웨어가 유사하다는 사례가 있다.<sup>1</sup>



Kay Kyoung-ju Kwak

@kjkwak12

...

#Matrix, #Magniber, #Hermes, #GlobeImposter, #GandCrab use exactly same icons, similar loader and decryption algorithm.

Below is yararule to detect some of icons

[pastebin.com/fXjgs39A](http://pastebin.com/fXjgs39A)

Any feedback welcome.

[DeepL로 번역](#) ⚙

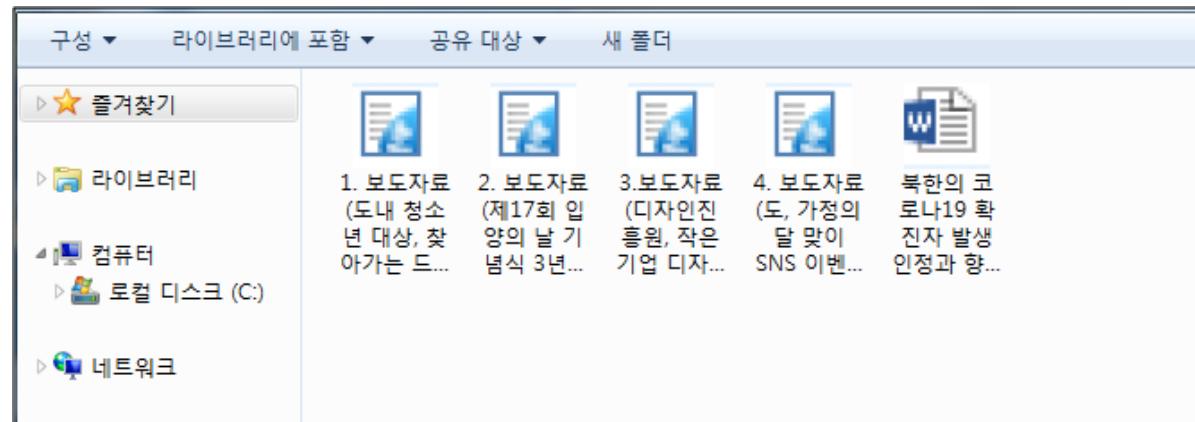
[게시물 번역하기](#)

0AA2B07C743DF99246A1A0C068...	0AC758D0E65C3B3E13B1D5EF475...	0ADD59B2AA11F5C7863DC0A368...	0AEDA9B5709BD21460CEA52353...
0AFC1D09E8D8812847DEFE212B8...	0B1AB78C01D5705956DE9E9E97F...	0B3E2C54516553A2B2F92D4055A...	0B50731FAED2020C1987C93A8B2...
0BAC7387B68DEBD6E2322985352...	0BD060C28955F9AA73848881EA1...	0BF814FDEEC0A91222806888413...	0C8DDF3FFB34FD8D7406B0822E...
0CA17828E917C357A9878528692...	0C786148924948B68AC8ABA8C7...	0CD3AAC4948F1638F5BFEEA482C...	0CD581E4ADD1F26B06406A8AF...
0CDFB347DA5883435D28E28D79...	0D03D8FD85122C7E54CF598ECD...	0D61B1A97C8AD401CCD9719087...	0D282F3A041173F0CE9E7F31D51...
0DA17F7A6103078FDCB826AF66...	0DC5421DA605C917D073C6EC81...	0DE0E085728370A6B298EA53E3EB...	0E260122E91CC995086FF54508...
0E69866174BC8C0F0D92838F872...	0E592792068872A36868A1C1C36...	0EC7E0B859489257D43A21021E8A...	0E1F26F35DDD50AAE83F412777...
0FOF02DDF2DF8DE6258954E972...	0F3FB53DA8381A58CAA526808FF...	0F4A699E555E0DF97388945321D...	0F88FE21AA99E58E744E84183362...
0FD1133BE84A4CCCCB8CA401416...	0FE477DAE2491C08888951BCA68...	0FF7CF52F315D36759E144D3FC9...	1A4E57D98C8C383929E1E6D0E45...
1A3146D80871AF6CD637898F1E...	1A9509A4957DC0812FCAF3F2F5...	1AC09D5884FCDF039FE850A7E4...	1ACC82D681BFF883076FC2C88FB...
1AD0EAA20942A28F45071B9828E...	1AE2EBA46C40BAE6A5C4A6CCE77...	1AE9155C07066428C226035ABC9...	1AE861F902C86D058AA838F87D...
182F5DBBE559FCB3A6A524D13...	1B69C44E43F2605843F36518ABC...	1B891E68CD236885F069A84E1116...	1B14476E4357B8CAEFB274627FA...
1B1879472ABCBD0529877094C...	1B8252A673E8BBF2940DEE370DB...	1BBB7B990F20DAC43CA39C09CA...	1BD8EAE600383CE9313ABFE46F...
1B843E1937911586435F690886C...	1C8F59222F28F9AAC9E95C9F6DF9...	1C988AB20F0E63E98CB510F1107...	1C608FA84C83E178CFEC8744C4...
1C84F1C1C9276604A0DD723069...	1C1498A4F6658D4798A7800CD2...	1C4845840A4E2984C78D463C0C...	1CA50EABD77D880298A3862148...
1CDC8C94F6758116002A5A650D...	1CF32C65E212F0A109776E545AF...	1D2EE78D9F2DA29CD5E17A299A...	1D642C94A970F883C13B0DADD7...
1DC9C6BA3387CEBD48558F3A5F...	1E82184C8C39D14F98EB258075D...	1E505795291F2AE647CC97D88A8...	1EEDFDA42538E04FCE341AC5F98...
1F28435840B895AC441FFBD9E75...	1F6F7C27892902D1E9E4BE1853F0...	1F064C3F670388536720E988832...	1F735C5822B971F721D9A445CF7...
1F851DF370A007E253C21DC87D...	1O2AF91983763048208868C3A80E...	1O2B67F98AB0A0B0C8652781A4F01...	1O2C7600DD73900896BCEBF836189...
2A3005868379E087298548D5E4C...	2AA29ECC2C9D71E6D3305E508...	2A531E626DFF9E09474CC46FAB5...	2AA162135E4E69E3F1C80C6C22E...
2A46C86BA487A8902D0D78E449...	2B0A006F6A8DED40B6B51C06857...	2B2DACC55DD8BA2EEFC585F8ED...	2B92C2F5E7364F1EF4B51005A671...
2B2F4A2C37A24FFB0C8C1F1BC55...	2B10E2B86D83F6CFA58FF9F987...	2B92C2F5E7364F1EF4B51005A671...	2B9467DD63F3D79339A77EB879...

<sup>1</sup> <https://twitter.com/kjkwak12/status/980708057037467648>

# 정상파일 위장 - 아이콘, 확장자 수정

이름	원본 크기	압축 크기	압축률	종류	수정한 날짜
readme.txt	95	96	-1%	텍스트 문서	2023-06-15 오전 9:38
개인정보유출내역.zip	82,320	81,675	1%	ZIP 파일	2023-06-15 오전 10:07
이름	원본 크기	압축 크기	압축률	종류	수정한 날짜
개인정보유출내역.hwp	...	299,520	81,624	73% 응용 프로그램	2023-06-15 오전 10:02



<sup>1</sup> <https://asec.ahnlab.com/ko/54473/> 기금 채용해주시며 언제나 성실하게 일합니다

<sup>2</sup> <https://asec.ahnlab.com/ko/34383/>

<sup>3</sup> <https://zdnet.co.kr/view/?no=20231124180240>

<sup>4</sup> Evaluation of Image Similarity Algorithms for Malware Fake-Icon Detection

# yara

- todo

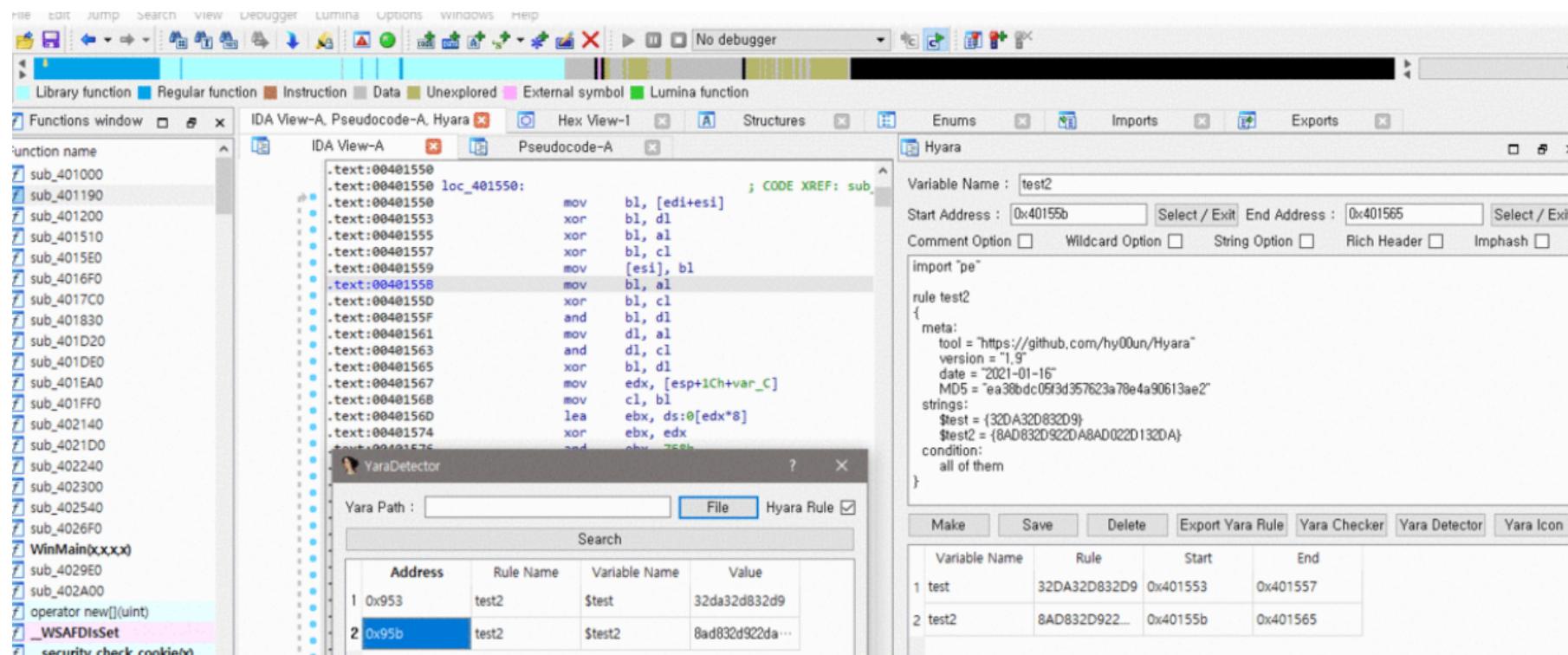
---

<sup>1</sup> <https://ahnlabasec.tistory.com/1124>

<sup>2</sup> 국내를 타깃으로 하는 위협그룹 프로파일링 - 2017 사이버 위협 인텔리전스 보고서

# Hyara

- yara rule 생성에 도움을 주는 플러그인 - IDA Pro , Binary Ninja , Cutter , Ghidra 에서 사용 가능
  - rule에 추가하고 싶은 어셈블리의 시작과 끝 주소를 지정하여 빠르게 생성 가능
  - 그 외에도 rule을 검증하기 위한 YaraDetector , YaraChecker 등 분석 도구 안에서 yara와 관련된 모든 작업을 해결할 수 있다.



# Hyara

- git clone <https://github.com/hyuunnn/Hyara> 또는 Code → Download ZIP 으로 파일 다운로드
- hyara\_lib 폴더 + Hyara\_IDA.py 파일을 IDA Pro 설치 경로에 있는 plugins 폴더에 넣기
- 현재 YaraIcon 동작 이슈로 인해 python 3.9.13 , pillow 8.3.2 설치 필요
  - C:\Users\유저명\AppData\Local\Programs\Python\Python39\pip.exe install -r requirements.txt
- IDA Pro 실행 후 CTRL + SHIFT + Y 단축키로 실행 가능

Variable Name	Rule	Start	End
test	0000000000000000	0x0000000000000000	0x0000000000000000
test2	0000000000000000	0x0000000000000000	0x0000000000000000

# yarGen

- todo

---

<sup>1</sup> [How to post-process YARA rules generated by yarGen](#)

# Loki

- todo

---

<sup>1</sup> [How to Write Simple but Sound Yara Rules](#)

<sup>2</sup> [How to Write Simple but Sound Yara Rules – Part 2](#)

<sup>3</sup> [How to Write Simple but Sound Yara Rules – Part 3](#)

# THOR Lite

- todo

---

<sup>1</sup> THOR Lite Introduction and Demo

<sup>2</sup> How to scan systems with THOR lite scanner during compromise assessment and incident response

<sup>3</sup> THOR- Lite Exchange Server 2019 Scan for ProxyShell ProxyToken Exploitation

<sup>4</sup> <https://www.nextron-systems.com/compare-our-scanners>

# Sigma

- todo

---

<sup>1</sup> Hack.lu 2017 Sigma - Generic Signatures for Log Events by Thomas Patzke

<sup>2</sup> Hunting for Hackers with Sigma Rules

<sup>3</sup> Sigma - Generic Signatures for SIEM Systems by Florian Roth

# iocextract

- todo

# bindiff

- todo

---

<sup>1</sup> Binary Comparisons for Patch Differencing - BinDiff Tutorial

<sup>2</sup> Identifying Code Reuse in Ransomware with Ghidra and BinDiff

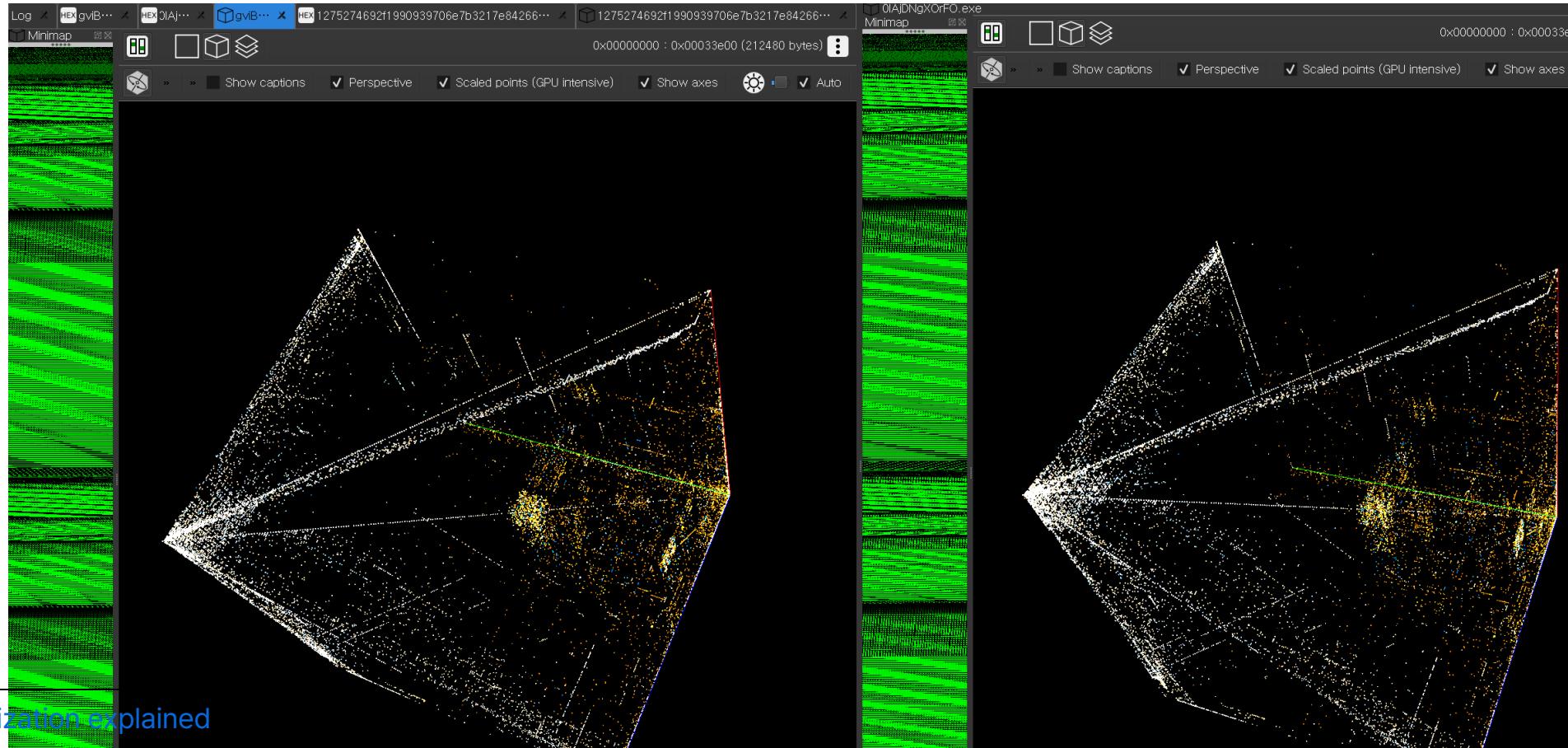
# diaphora

- todo

# pigaios

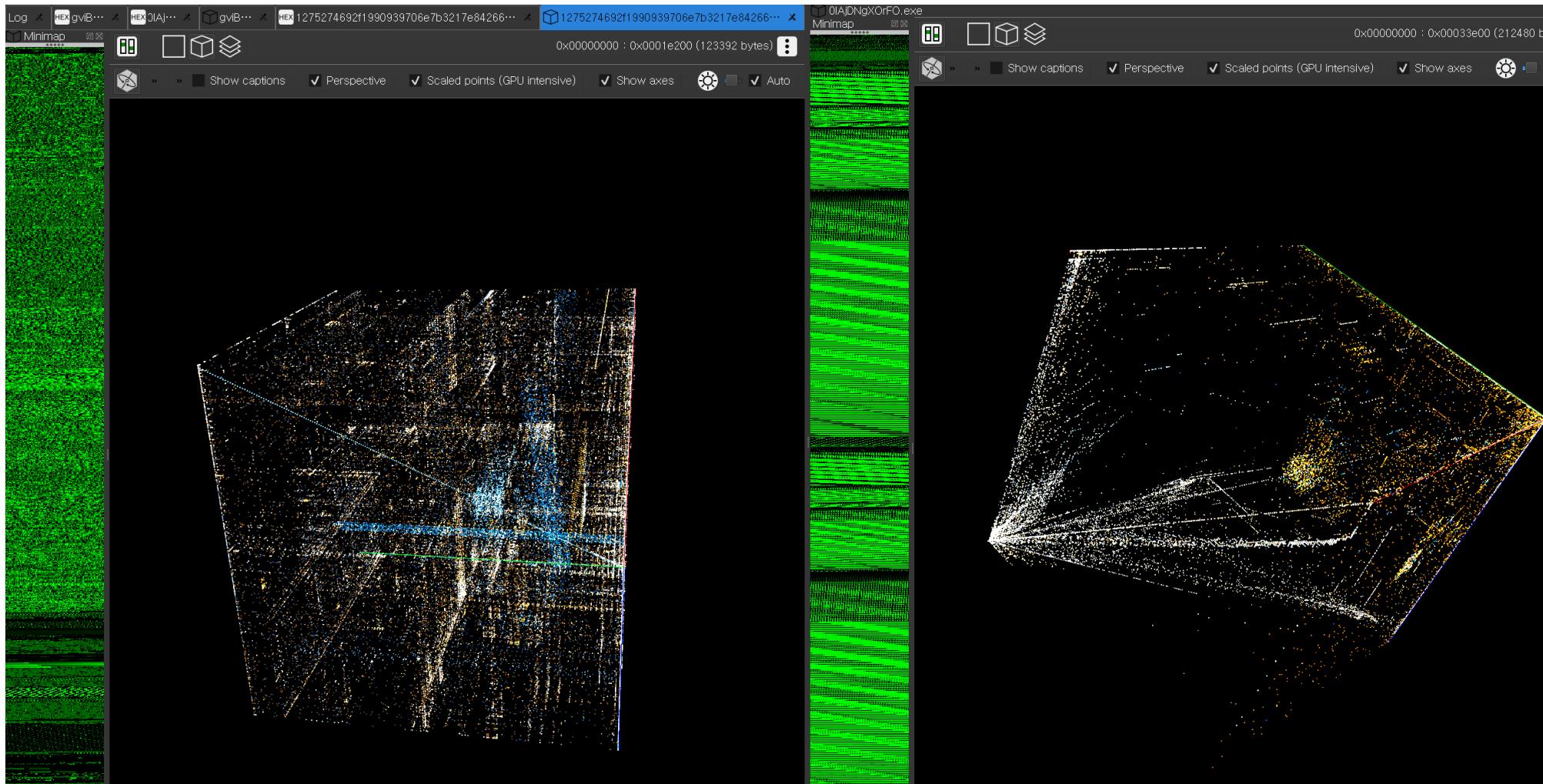
- todo

- binary visualization tool 중에서 신박하다고 생각한 프로젝트
- 미니맵을 보면 두 파일이 거의 유사한 것을 볼 수 있으며, 해당 데이터를 3차원으로 보여준다.



<sup>1</sup> Binary visualization explained

- 다른 파일의 결과를 보면 미니맵과 출력 결과가 다른 것을 확인할 수 있다.



# ETC

- [Cantordust - Blog](#)
- [binocle](#)
- [binvis](#)
- [binary\\_viewer](#)
- [Visualizing ELF binaries - Binary Visualization](#)
- ...

- GTFOBins, LOLBAS

시스템에 있는 기본 도구를 사용하는 공격자들을 'living off the land'라고 한다.

대상 운영체제에 포함된 도구에 대해 철저하고 정통한 지식을 갖고 있으면  
공격자는 자신의 도구를 모두 가지고 다닐 필요 없이 '가벼운 여행'을 할 수 있다.

net.exe 명령어의 사용을 통해 시스템에 사용자 계정이 추가됐다는 점은  
공격자가 획득한 접근 수준과 사용한 방법을 알려준다.

사용자 프로파일의 shellbag 아티팩트를 조사한 결과  
사용자를 추가하기 위해 제어판에 액세스한 적이 있음을 알게 되면  
터미널 서비스를 통해 GUI 셸과 상호작용할 수 있는 셸 기반 액세스 권한이 있음을 알 수 있다.

[Windows 환경에서 침해 시스템 분석하기 - p160, Eng](#)

# 악성코드 샘플

[awesome-malware-analysis](#), [theZoo](#), malware samples & writeup - 1 2 3 4 5 6 7 8 9,

[Ultimate-RAT-Collection](#), [malware-traffic-analysis](#), [hybrid-analysis](#), [virusshare](#), [vx-underground](#)

[malware-study](#), [malware-tools](#), [Reverse Engineering tools](#), [Reverse-Engineering](#),  
[reverseengineering-reading-list](#)

[exploitation-course](#), [레드팀 플레이북](#), [Win32\\_Offensive\\_Cheatsheet](#)

[linux-re-101](#), [osx-re-101](#), [RE-iOS-Apps](#), [AndroidAppRE](#)

# 참고자료

멀웨어 데이터 과학

악성코드 분석 시작하기

국내를 타깃으로 하는 위협그룹 프로파일링 - 2017 사이버 위협 인텔리전스 보고서

악성코드 중심의 침해대응 매트릭스 모델링 - 2019 인텔리전스 보고서