

201716905 김강민

악성코드 분석

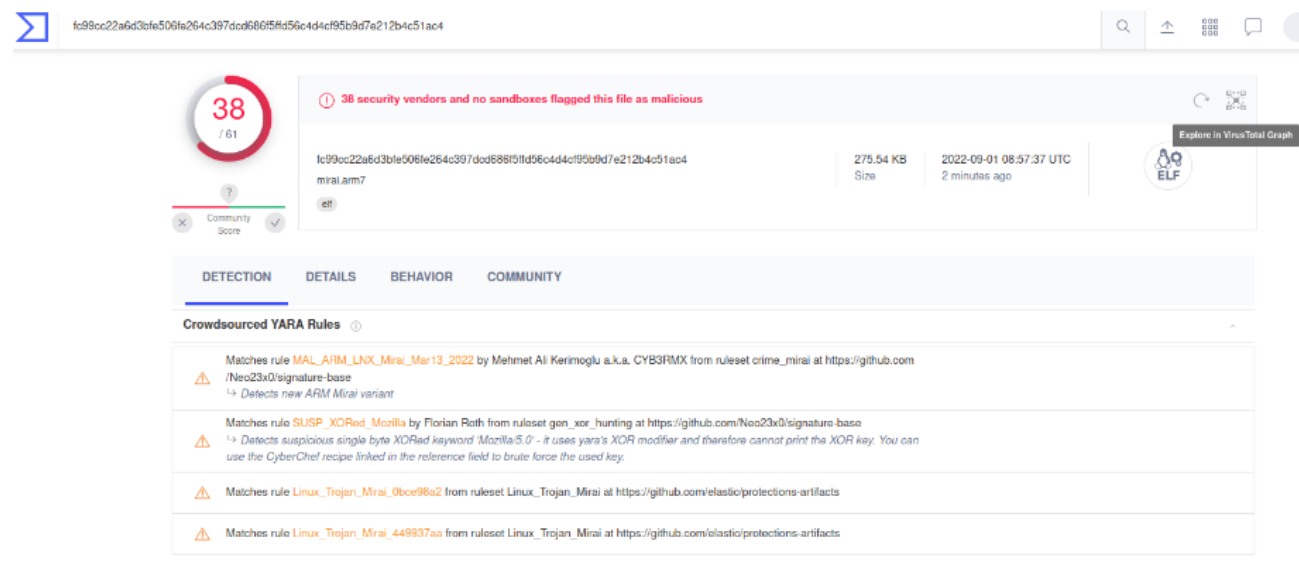
IT 정보공학과 BCG LAB 201716905 김강민

· 정적 악성코드 분석

1. 백신 바이너리 스캐닝 이용하기
2. PE header 정보 조사하기
3. 문자열 추출하여 분석하기
4. 파일 난독화 여부 확인하기
5. Hash value 비교하기
6. Yara를 통해 분류하기

1. 백신 바이너리 스캐닝

- 백신 웹사이트/검색엔진을 통해 시그니처 검색
- 백신 스캐너의 공개 API를 이용하여 질의도 가능 (python)
- 백신 스캐너에 감지되지 않았다고, 안전한 것은 아님



2. PE header 정보 조사하기

(PE file : Portable Executable의 약자로 윈도우 실행 파일)

- signature : 파일 유형 식별자 => 파일 유형 파악 가능
- import : Import하고 있는 dll, functions 정보 확인 가능 => 다양한 정보 확인 가능
- export: 다른 프로그램에서 Import하는 함수 목록 => DLL의 성격 파악 및 참고하는 프로그램 확인 가능
- section: 관련된 메모리끼리 분류한 공간 정보 => 수상한 section 확인 및 packing 여부 확인
- Time-stamp: 코드 생성 시간 등 타임 라인 확인 가능
- PE resource: 아이콘, 메뉴 같은 리소스

2-1. Signature 분석하기

- hex editor, 파일 식별 도구, 파이썬 등을 사용하여 확인 가능
- 4D 5A로 시작하는 파일은 윈도우 실행파일(.exe)
- https://en.wikipedia.org/wiki/List_of_file_signatures 에서 다양한 signature 확인 가능

```
root@ubuntu:/home/hacker/Desktop# file cron.sh
cron.sh: POSIX shell script, ASCII text executable
```

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....ÿÿ..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	10	01	00	00
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..°..'.í!..Lí!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....\$......

2-2. Import 및 functions 확인 (의존성 확인)

- 프로그램은 API를 이용하며, 동적 라이브러리 링크(DLL) 파일이 필요
- load한 DLL list, API list를 확인 가능
- LoadLibrary()와 같은 API를 통해 DLL을 호출하여, GetProcAddress() API를 통해 실행시킨 경우 해당 PE import table에서 확인 불가
- Import가 거의 없는 파일은 Packing한 binary일 가능성이 높음

library (7)	flag (0)	bound (0)	type (1)	functions (31)	description
shell32.dll	-	-	implicit	1	Windows Shell Common DLL
kernel32.dll	-	-	implicit	9	Windows NT BASE API Client DLL
msvcrt.dll	-	-	implicit	15	Windows NT CRT DLL
advapi32.dll	-	-	implicit	3	Advanced Windows 32 Base API
api-ms-win-core...	-	-	implicit	1	n/a
api-ms-win-core...	-	-	implicit	1	n/a
api-ms-win-core...	-	-	implicit	1	n/a

pestudio 9.40 - Malware Initial Assessment - www.winitor.com [c:\windows#winsxs#wow64_microsoft-windows-calc_31bf3856ad364e08_x-ww6495966b40ccf0a03b910c2-73#calc.exe]

file settings about

functions (31)	flag (4)	ordinal (0)	library (7)
ShellExecuteW	x	-	shell32.dll
TerminateProcess	x	-	kernel32.dll
GetCurrentProcessId	x	-	kernel32.dll
GetCurrentThreadId	x	-	kernel32.dll
SetUnhandledExceptionFilter		-	kernel32.dll
GetCurrentProcess		-	kernel32.dll
UnhandledExceptionFilter		-	kernel32.dll
GetSystemTimeAsFileTime		-	kernel32.dll
GetTickCount		-	kernel32.dll
QueryPerformanceCounter		-	kernel32.dll
_amsg_exit		-	msvcrt.dll
_p_fmode		-	msvcrt.dll
_setusermatherr		-	msvcrt.dll
_initterm		-	msvcrt.dll
_wcmdln		-	msvcrt.dll
void __cdecl terminate(void)		-	msvcrt.dll
_controlfp		-	msvcrt.dll
_exit		-	msvcrt.dll
exit		-	msvcrt.dll
p_commode		-	msvcrt.dll

2.3 Export 조사

- 다른 프로그램에서 해당 프로그램의 함수를 사용하는 경우 export에 기록이 남음
- 실행 파일보다는 대부분 dll에서 존재
- export 목록을 보면 어떤 dll인지 유추 할 수도 있음

pestudio 9.40 - Malware Initial Assessment - www.winitor.com [c:\riot games\riot client\wivoxsdk.dll]

file settings about

Index	name (493)	flag (0)	location	duplicate (1...	ordinal (0)	gap (0)	forwarded (0)	entry-point
1	destroy_evt	-	.text:0x28...	-	-	-	-	-
2	destroy_req	-	.text:0x2900	-	-	-	-	-
3	destroy_resp	-	.text:0x2930	-	-	-	-	-
4	flite_lang_list	-	.reloc:0xA...	-	-	-	-	-
5	flite_lang_list_length	-	.reloc:0xA...	-	-	-	-	-
6	flite_voice_list	-	.reloc:0xA...	-	-	-	-	-
7	vx_account_create	-	.text:0x2960	-	-	-	-	-
8	vx_account_free	-	.text:0x2990	-	-	-	-	-
9	vx_alloc_spurs_jobqueue_ha...	-	.text:0xA670	-	-	-	-	-
10	vx_allocate	-	.text:0x2480	-	-	-	-	-
11	vx_allocate_aligned	-	.text:0x24...	-	-	-	-	-
12	vx_apply_font_to_file	-	.text:0x29C0	-	-	-	-	-
13	vx_apply_font_to_file_return...	-	.text:0x2A00	x	-	-	-	-
14	vx_apply_font_to_vxz_file_re...	-	.text:0x2A00	x	-	-	-	-
15	vx_auto_accept_rule_create	-	.text:0x2A20	-	-	-	-	-
16	vx_auto_accept_rule_free	-	.text:0x2A50	-	-	-	-	-
17	vx_auto_accept_rules_create	-	.text:0x2A80	x	-	-	-	-
18	vx_auto_accept_rules_free	-	.text:0x2A80	-	-	-	-	-
19	vx_block_rule_create	-	.text:0x2AE0	x	-	-	-	-
20	vx_block_rule_free	-	.text:0x2B10	-	-	-	-	-
21	vx_block_rules_create	-	.text:0x2A80	x	-	-	-	-

2.4 Section 조사

- PE 파일의 실제 내용은 section으로 분류
- 일반적이지 않은 section 명에 대해서는 추가 분석이 필요
- UPX packing 같은 경우 UPX0, UPX1 section이 존재
- 일반적으로 virtual-size와 raw-size가 동일하거나 유사한데, raw-size < virtual-size가 크게 차이나는 경우 패킹 (압축 해제로 인한 size up)

섹션	설명
.text (CODE)	실행 코드가 기록된 부분
.data (DATA)	읽기/쓰기 데이터와 전역 변수
.rdata	읽기 전용 데이터 (Import, Export 정보 포함)
.idata	Import 테이블 존재 시 포함, 없을 시 rdata 섹션에 Import 정보 저장
.edata	Export 테이블 존재 시 포함, 없을 시 rdata 섹션에 Export 정보 저장
.rsrc	아이콘, 대화창, 문자열 등 리소스

필드	설명
names	섹션명
virtual-size	메모리에 로딩할 때의 섹션의 크기
virtual-address	섹션의 프로세스 내 offset (상대 위치)
raw-size	디스크에 존재할 때의 크기
raw-data	파일에서 해당 섹션의 offset (상대 위치)
entry-point	코드 실행 위치 (상대적 가상 주소) => 일반적으로 .text 내의 값

2.5. Time-stamp 조사

- PE header는 바이너리 컴파일 될 때의 정보 포함
- 코드 생성 시간 등 타임라인을 확인 가능
- 공격자가 time-stamp를 수정하여 방해할 수도 있음

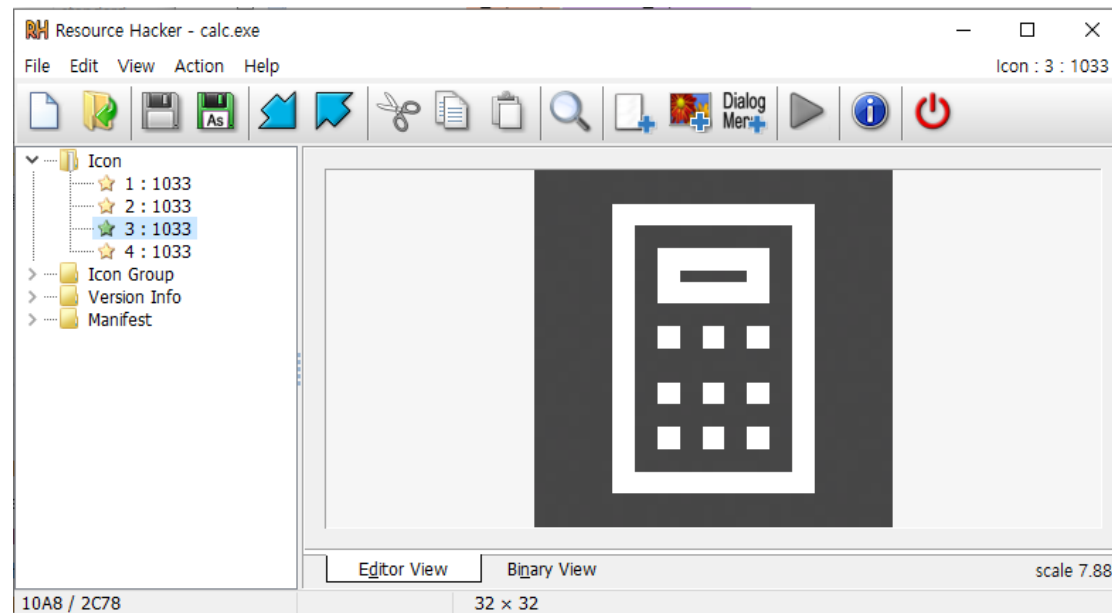
pestudio 9.40 - Malware Initial Assessment - www.winitor.com [c:\windows\winsxs\wow64_microsoft-windows-calc_31bf3856ad364e35_10.0.19041.1_none_6a03b9

file settings about

property	value	detail
characteristics	0x0102	
dynamic-link-library	0x0000	false
32-bit words support	0x0100	true
file-can-be-executed	0x0002	true
system-image	0x0000	false
large-address-aware	0x0000	false
debug-stripped	0x0000	false
line-stripped-from-file	0x0000	false
local-symbols-stripped-from-file	0x0000	false
relocation-stripped	0x0000	false
uniprocessor	0x0000	false
bytes-of-machine-words-reversed-Low	0x0000	false
bytes-of-machine-words-reversed-Hi	0x0000	false
media-run-from-swap	0x0000	false
network-run-from-swap	0x0000	false
general		
compiler-stamp	0xAA9563FE	Thu Sep 09 00:59:42 2060 UTC
size-of-optional-header	0x00E0	224 bytes
signature	0x00004550	PE00
machine	0x014C	Intel-386
sections	0x0005	5
pointer-symbol-table	0x00000000	0x00000000
number-of-symbols	0x00000000	0x00000000

2.6 PE resource 조사

- 아이콘, 메뉴, 문자열 같은 resource를 .rsrc section에 저장
- 공격자는 때때로 추가 바이너리, decoy 문서, 설정 데이터를 rsrc 저장하므로 힌트가 될 수 있음
- 근원지, 회사명, 프로그램 제작자 세부 정보, 저작권 등 정보 노출 가능



3. 문자열 추출하기

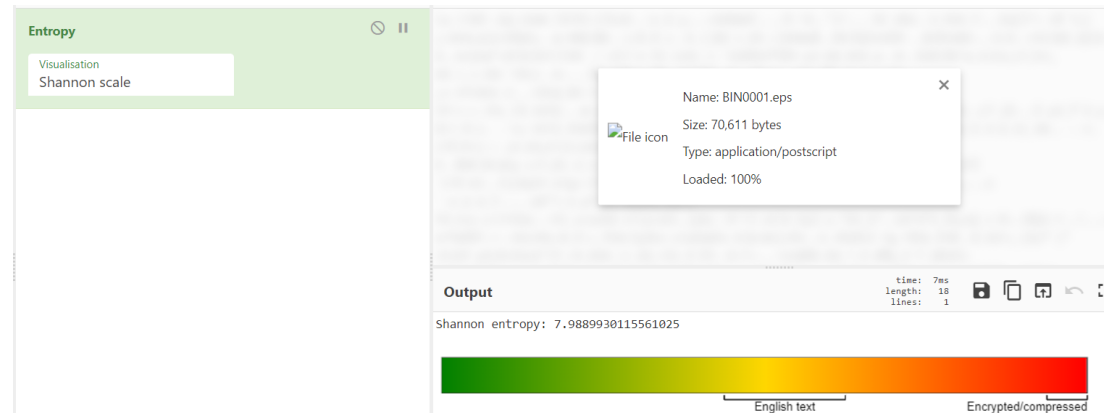
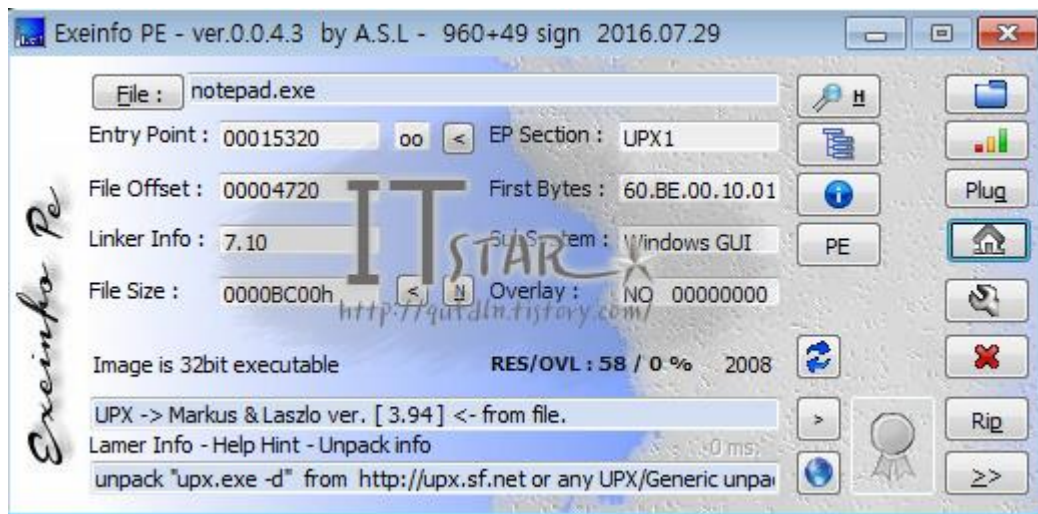
- 리눅스에서는 `string` 유틸리티를 통해 확인 가능
- 난독화된 문자열은 `string` 같은 유틸리티로는 확인 불가
- FLOSS는 은닉한 문자열 찾도록 도와주는 프로그램

encoding (1)	size (bytes)	location	flag (0)	hint (2)	value (14)
ascii	9	0x00000179	-	file	ida64.exe
ascii	39	0x000001F6	-	file	C:\Program Files\IDA Demo 8.0\ida64.exe
ascii	3	0x00000026	-	-	?g;
ascii	3	0x0000005E	-	-	+00
ascii	4	0x00000064	-	-	/C:\
ascii	8	0x00000089	-	-	PROGRA~1
ascii	10	0x00000115	-	-	IDADEM~1.0
ascii	4	0x00000174	-	-	UTm
ascii	15	0x000002FF	-	-	desktop-677hog0
ascii	6	0x00000311	-	-	WLmqFM
ascii	6	0x00000331	-	-	WLmqFM
ascii	7	0x0000035B	-	-	1SPSU(L
ascii	4	0x00000399	-	-	1SPS
ascii	3	0x000003A6	-	-	H@.

“FLOSS”

4. 파일 난독화 파악

- 난독화가 되어있으면 일반적인 방법으로 분석 어려움 (분석, 리버싱 등 회피)
- packer : 실행파일의 내용을 압축해 난독화 하는 프로그램
- cryptor: 실행파일을 암호화 하는 프로그램
- 프로그램을 이용하거나 특정 패턴 확인, 엔트로피 확인 등으로 유추 가능



5. Hash 값 분석

- binary, fuzzy, Import, section hash 와 같은 hash 값을 기존에 공개된 악성코드의 hash 값과 비교
- fuzzy hash: 파일 전체에 대한 hash value를 계산하지 않고, 일정 크기 단위로 구분하여 각 단위 블록에 대한 hash value를 만드는 방식
(파일 무결성 검증, 파일 유사도 파악 가능)
- fuzzy hash는 ssdeep 패키지를 통해 유사도를 확인 가능

```
root@ubuntu:/home/hacker/mirai/debug# ssdeep -pb *
/home/hacker/mirai/debug/bins: Is a directory
/home/hacker/mirai/debug/mitmAP: Is a directory
attack2.txt matches attack3.txt (48)
attack2.txt matches attack.txt (71)

attack3.txt matches attack2.txt (48)

attack.txt matches attack2.txt (71)
```

pestudio 9.40 - Malware Initial Assessment - www.winitor.com [c:\windows\winsxs\wow64_microsoft-windows-calc_31bf3856ad364e35_10.0.19041.1_none_6a03b910ee7a4073#calc.exe]

property	value	value	value	value	value
general					
name	.text	.data	.idata	.rsrc	.reloc
md5	0427DB371A56AEF4156D...	44C08E1F5E2CA5F40CDA...	78A7577BC986598BF4829...	28E2D87DC9A4C14CDB8...	3CE043D67FA16741A7CB...
entropy	5.690	0.240	4.064	2.812	4.699
file-ratio (96.08%)	15.69 %	1.96 %	5.88 %	70.59 %	1.96 %
raw-address	0x00000400	0x00001400	0x00001600	0x00001C00	0x00006400
raw-size (25088 bytes)	0x00001000 (4096 bytes)	0x00000200 (512 bytes)	0x00000600 (1536 bytes)	0x00004800 (18432 bytes)	0x00000200 (512 bytes)
virtual-address	0x00001000	0x00002000	0x00003000	0x00004000	0x00009000
virtual-size (24532 bytes)	0x00000F2C (3884 bytes)	0x0000039C (924 bytes)	0x000004A8 (1192 bytes)	0x00004708 (18184 bytes)	0x0000015C (348 bytes)
entry-point	0x00001800				

pestudio 9.40 - Malware Initial Assessment - www.winitor.com [c:\windows\winsxs\wow64_microsoft-windows-calc_31bf3856ad364e35_10.0.19041.1_none_6a03b910ee7a4073#cal

property	value
md5	961E0938E1F666FD38602AD90A5F480F
sha1	3574FC3A80D80146A7067A478DB209E452757950
sha256	B183BD6414C5123465075D76D2413C999D569492F8543AC8C296908487458DF2
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 88 00 00 00 00 00 00 40 00 00 00 00 ...
first-bytes-text	M Z @
file-size	26112 bytes
entropy	3.924
imphash	15911A8CE2E8CODE1F88E3B926F8458E

6. Signature 기반 탐지도구 사용 (YARA)

- binary에 나타나는 고유 문자열/바이너리 구분자를 기준으로 악성코드 분류
- YARA rule을 작성하여 분류 가능
- `yara64.exe -r <yara_rule_name> <file>`

```
rule embedded_office_document
{
  meta:
    description = "Detects embedded office document"

  strings:
    $mz = { 4D 5A }
    $a = { D0 CF 11 E0 A1 B1 1A E1 }

  condition:
    ($mz at 0) and $a in (1024..filesize)
}
```

```
C:\Users\kise\>C:\Users\kise\Desktop\Files\yara64.exe -r C:\Users\kise\Desktop\Files\shell.bin
IsWow64Process_10764760 C:\Users\kise\Desktop\Files\shell.bin
CreateFileMappingA_23f9cd0a C:\Users\kise\Desktop\Files\shell.bin
UnmapViewOfFile_257ca71e C:\Users\kise\Desktop\Files\shell.bin
VirtualFree_300f2f0b C:\Users\kise\Desktop\Files\shell.bin
VirtualAllocEx_3f9287ae C:\Users\kise\Desktop\Files\shell.bin
RtlCreateUserThread_40a438c8 C:\Users\kise\Desktop\Files\shell.bin
GetSystemTime_40dc4f36 C:\Users\kise\Desktop\Files\shell.bin
Thread32Next_42f34ed0 C:\Users\kise\Desktop\Files\shell.bin
Thread32First_4a04d8b7 C:\Users\kise\Desktop\Files\shell.bin
CreateFileA_4fdaf6da C:\Users\kise\Desktop\Files\shell.bin
OpenProcess_50b695ee C:\Users\kise\Desktop\Files\shell.bin
GetCurrentProcess_51e2f352 C:\Users\kise\Desktop\Files\shell.bin
ExitProcess_56a2b5f0 C:\Users\kise\Desktop\Files\shell.bin
```

· 동적 악성코드 분석

- 폐쇄망에서 악성코드를 돌려 행동을 관찰

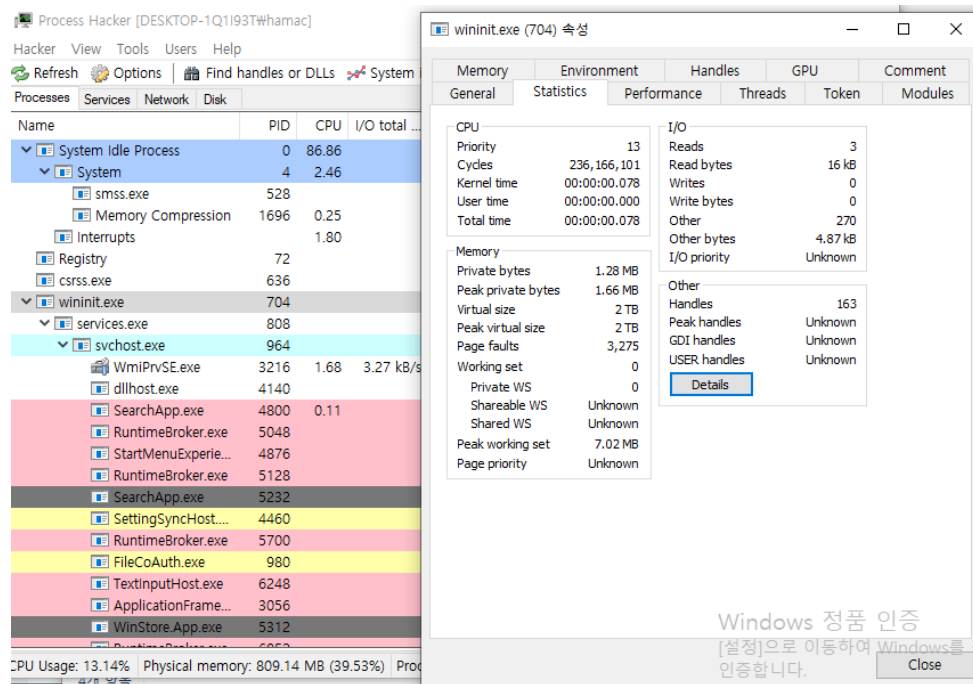
1. 프로세스 모니터링: 악성코드 실행되는 동안 생성한 결과의 속성 검사
2. 파일 시스템 모니터링: 악성코드가 실행되는 동안 파일 시스템을 실시간 모니터링
3. 레지스트리 모니터링: 접근/수정된 registry Key나 악성코드가 읽거나 작성한 registry data를 모니터링
4. 네트워크 모니터링: 악성코드가 실행되는 동안 발생한 라이브 traffic 모니터링

1. 동적 악성코드 분석 도구

1. Process hacker
2. Process monitor
3. Noriben
4. Wireshark
5. INetSim

1.1 Process hacker

- 시스템 리소스를 모니터링하는 도구
- 실행 중인 프로세스 조사 및 속성 확인
- 서비스, 네트워크 접속, 디스크 활동 등 파악 가능

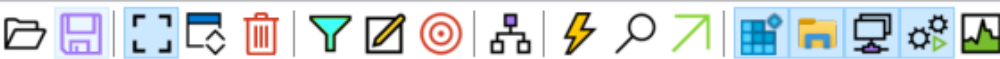


1.2 Process monitor

- 현재 실행중인 프로세스의 정보 확인 가능
- 시스템, 레지스트리, 프로세스/스레드 활동의 실시간 시스템 상호작용 모니터링

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help



Time...	Process Name	PID	Operation	Path	Result	Detail
오후 4:...	Explorer.EXE	4396	ReadFile	C:\Windows\System32\TaskFl...	SUCCESS	Offset: 1,482,752...
오후 4:...	MsmpegEng.exe	2424	CreateFile	C:\Windows\System32\oleacc...	SUCCESS	Desired Access...
오후 4:...	MsmpegEng.exe	2424	FileSystemC...	C:\Windows\System32\oleacc...	OPLOCK HANDL...	Control: FSCTL...
오후 4:...	MsmpegEng.exe	2424	FileSystemC...	C:\Windows\System32\oleacc...	SUCCESS	Control: 0x902eb...
오후 4:...	MsmpegEng.exe	2424	CloseFile	C:\Windows\System32\oleacc...	SUCCESS	
오후 4:...	Explorer.EXE	4396	ReadFile	C:\Windows\System32\TaskFl...	SUCCESS	Offset: 1,466,368...
오후 4:...	MsmpegEng.exe	2424	CreateFile	C:\Windows\System32\oleacc...	SUCCESS	Desired Access...
오후 4:...	MsmpegEng.exe	2424	FileSystemC...	C:\Windows\System32\oleacc...	OPLOCK HANDL...	Control: FSCTL...
오후 4:...	MsmpegEng.exe	2424	FileSystemC...	C:\Windows\System32\oleacc...	SUCCESS	Control: 0x902eb...
오후 4:...	MsmpegEng.exe	2424	CloseFile	C:\Windows\System32\oleacc...	SUCCESS	
오후 4:...	svchost.exe	1932	ReadFile	C:\Windows\System32\StateR...	SUCCESS	Offset: 690,688, ...
오후 4:...	Explorer.EXE	4396	ReadFile	C:\Windows\System32\TaskFl...	SUCCESS	Offset: 1,416,192...
오후 4:...	svchost.exe	1932	ReadFile	C:\Windows\System32\StateR...	SUCCESS	Offset: 678,400, ...
오후 4:...	Explorer.EXE	4396	ReadFile	C:\Windows\System32\TaskFl...	SUCCESS	Offset: 1,399,608...
오후 4:...	svchost.exe	1932	ReadFile	C:\Windows\System32\StateR...	SUCCESS	Offset: 644,096, ...
오후 4:...	Explorer.EXE	4396	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Name
오후 4:...	Explorer.EXE	4396	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: HandleT...
오후 4:...	Explorer.EXE	4396	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: HandleT...
오후 4:...	Explorer.EXE	4396	RegOpenKey	HKCU\Software\Classes\WU...	NAME NOT FOU...	Desired Access...
오후 4:...	Explorer.EXE	4396	RegOpenKey	HKCR\WU\Users\hamac\OneDr...	NAME NOT FOU...	Desired Access...
오후 4:...	Explorer.EXE	4396	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Name
오후 4:...	Explorer.EXE	4396	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: HandleT...
오후 4:...	Explorer.EXE	4396	RegQueryKey	HKCU\Software\Classes	SUCCESS	Query: Name
오후 4:...	Explorer.EXE	4396	RegOpenKey	HKCU\Software\Classes\WU...	NAME NOT FOU...	Desired Access...
오후 4:...	Explorer.EXE	4396	RegOpenKey	HKCR\WU\Users\hamac\OneDr...	NAME NOT FOU...	Desired Access...

1.3 Noriben

- 악성코드의 런타임 지표 수집, 분석, 리포트 하는 파이썬 스크립트
- 악성코드와 관련된 event에 집중할 수 있도록 돕는 사전 필터 존재
- Noriben을 종료하면 txt와 csv 형태로 결과 저장
- txt: 카테고리 기반 분석, csv: 타임라인 기반 분석

```
C:\Users\hamac\OneDrive\바탕 화면\secure\dynamic_analysis\ProcessMonitor>Noriben.py
[+] Python module "requests" not found. Internet functionality is disabled.
[+] This is acceptable if you do not wish to upload data to VirusTotal.

--===[ Noriben v1.8.7
[!] Filter file ProcmonConfiguration.PMC not found. Continuing without filters.
[*] Using procmon EXE: procmon.exe
[*] Procmon session saved to: Noriben_06_Sep_22__17_51_816396.pml
[*] Launching Procmon ...
[*] Procmon is running. Run your executable now.
[*] When runtime is complete, press CTRL+C to stop logging.
```

 Noriben_06_Sep_22__17_51_816396.csv		2022-09-06 오후 5:52	Microsoft Excel ...
 Noriben_06_Sep_22__17_51_816396.pml		2022-09-06 오후 5:52	ProcMon Log File
 Noriben_06_Sep_22__17_51_816396.txt		2022-09-06 오후 5:52	텍스트 문서
 Noriben_06_Sep_22__17_51_816396_ti...		2022-09-06 오후 5:52	Microsoft Excel ...

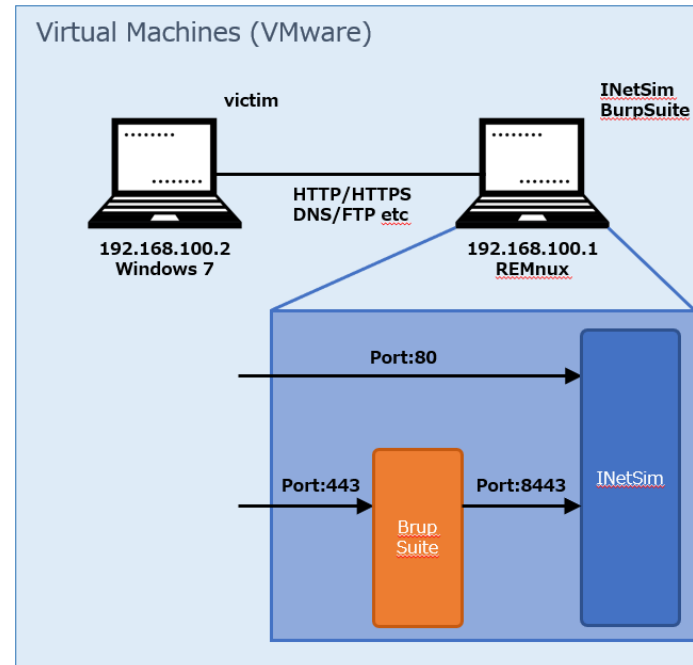
1.4 Wireshark

- 네트워크 traffic을 캡처 하는 packet sniffer
- windows, linux에서 모두 사용 가능

Source	Destination	Protocol	Length	Info
10.200.8.4	10.200.8.6	TCP	54	[TCP Port numbers reuse
10.200.8.6	10.200.8.4	TCP	60	80 → 15410 [SYN, ACK] S
10.200.8.6	10.200.8.4	TCP	60	80 → 15411 [SYN, ACK] S
10.200.8.6	10.200.8.4	TCP	60	80 → 15412 [SYN, ACK] S
10.200.8.6	10.200.8.4	TCP	60	80 → 15413 [SYN, ACK] S
10.200.8.6	10.200.8.4	TCP	60	80 → 15414 [SYN, ACK] S
10.200.8.6	10.200.8.4	TCP	60	80 → 15415 [SYN, ACK] S
10.200.8.4	10.200.8.6	TCP	54	15410 → 80 [RST] Seq=1
10.200.8.4	10.200.8.6	TCP	54	15411 → 80 [RST] Seq=1
10.200.8.4	10.200.8.6	TCP	54	15412 → 80 [RST] Seq=1
10.200.8.4	10.200.8.6	TCP	54	[TCP Port numbers reuse
10.200.8.4	10.200.8.6	TCP	54	15413 → 80 [RST] Seq=1
10.200.8.4	10.200.8.6	TCP	54	15414 → 80 [RST] Seq=1
10.200.8.4	10.200.8.6	TCP	54	15415 → 80 [RST] Seq=1
10.200.8.4	10.200.8.6	TCP	54	[TCP Port numbers reuse
10.200.8.6	10.200.8.4	TCP	60	80 → 15416 [SYN, ACK] S
10.200.8.6	10.200.8.4	TCP	60	80 → 15417 [SYN, ACK] S

1.5 INetSim

- C&C server 접근을 확인하고 싶을 때, 직접 접속하지 않은 상태로 확인 가능 (fake net을 구축해 악성코드의 C2 서버 연결 등을 확인 가능)
- 악성코드가 작동하는데 필요한 모든 서비스 제공
- Windows VM => Http/Https 요청 => Linux VM의 INetSim 응답 => 악성코드는 C&C server와 통신하는 것으로 착각



2. 동적 악성코드 분석 단계

1. clean shot으로 복원: 가상 머신을 초기 스냅샷 상태로 복원
2. 모니터링/동적 분석 도구 실행: 악성코드 샘플 실행 전에 모니터링 도구 실행
3. 악성코드 샘플 실행: 관리자 권한으로 악성코드 샘플 실행
4. 모니터링 도구 종료: 악성코드 바이너리 일정 시간 동안 실행한 후 종료
5. 결과 분석: 데이터/리포트 수집하고 분석

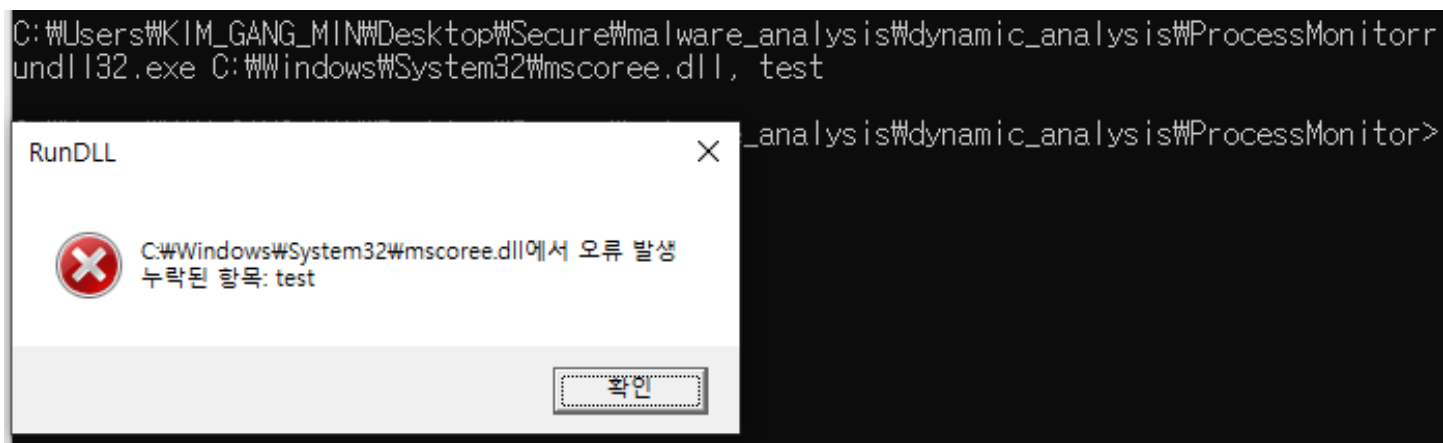
3. 동적 링크 라이브러리(DLL) 분석

- 분석 난이도 상승, 지속성 등을 위해 악성 행위를 DLL로 만드는 경우 존재
- 실행 파일로 구성된 악성코드는 직접 실행해서 테스트 가능하지만, DLL은 실행이 불가능
- rundll32.exe라는 프로그램을 통해 DLL을 load하여, 함수 호출 가능

```
rundll32.exe <DLL Path>, <Export func> <arg 1> <arg 2> <...>
```

3.1 export가 없는 DLL 분석

- export가 없는 경우 => 다른 프로그램에서 해당 DLL의 함수를 호출하지 않음 => DLL 내부에서 실행
- DLLMain 함수는 일반적인 함수의 main 함수 역할을 하며, 해당 함수에 악의적 기능을 구현
- rundll32.exe라는 프로그램을 통해 DLL을 load하여, 함수 호출 가능
- export가 없는 경우에도 export할 함수를 호출하지 않으면 실행 x => 에러가 발생해도 dll 자체는 메모리에 load(DLLMain은 작동됨)



3.2 export가 있는 DLL 분석

- DLL의 entry point(DLLMain)에 함수가 구현되지 않으면, 메모리에 로드 되더라도 어떤 행위도 하지 않음
(이런 경우는 export되는 함수들을 직접 분석할 필요가 있음)
- export 함수가 갖는 매개변수와 그 종류를 파악 필요 (코드 분석)
- 일부 DLL은 특정 프로세스에서만 작동 (rundll32.exe 이용 불가) => RemoteDLL 같은 도구로 DLL 삽입하여 분석)

1. 다운로더 (Downloader)

- 다른 악성코드 컴포넌트를 다운로드
- `UrlDownloadToFile()`: 디스크에서 파일 다운로드
- `ShellExecute()`, `WinExec()`, `CreateProcess()` : 다운로드 받은 컴포넌트 실행

2. 드로퍼 (Dropper)

- 악성코드 컴포넌트를 추출해 디스크에 저장
- 다운로더: 외부에서 파일 다운로드, 드로퍼: 자기 자신 데이터를 통해 새로운 파일 생성
- `FileResource()`, `LoadResource()`, `LockResource()`, `SizeOfResource()`

3. 키로거 (Keylogger)

- 키 입력을 가로채서 기록하도록 설계된 프로그램
- `GetAsyncKeyState()`: 어떤 key가 눌려졌는지 확인
- `SetWindowsHookEx()`: Hook 함수를 등록해 키보드 이벤트를 모니터링

4. 이동식 미디어를 통한 악성코드 복제

- 이동식 미디어 감염 시켜 악성 프로그램 유포
- Autorun 취약점을 통해 자동적으로 감염 (Vista 이전 버전)
- GetLogicalDriveStringsA(): 유효한 드라이브 상세 정보 획득
- GetDriveType(): 드라이브가 이동식 미디어인지 확인
- CopyFileA(): 악성코드가 자신을 미디어에 복사
- setFileAttributesA(): FILE_ATTRIBUTE_HIDDEN 인자를 사용해 파일 숨김

5. 악성코드 명령 및 제어(C&C)

- 감염된 시스템과 통신 및 제어
- C2 server를 통해 명령 받기, 추가 컴포넌트 다운로드, 정보 유출 등
- 과거에는 IRC, 현재에는 HTTP/HTTPS 사용 (방화벽/네트워크 기반 탐지 우회 => 정상 트래픽으로 위장)
- 일부는 P2P, DNS 터널링과 같은 프로토콜 사용

5.1 HTTP 명령 및 제어

- InternetOpen(): 인터넷 통신 초기화 (GetComputerName()을 통해 호스트 이름 확인)
- InternetOpenUrl() : URL과 연결
- InternetReadFile(): 웹 페이지의 내용 추출 (div 태그 내에 암호화된 악성코드 추출 등)
- UrlDownloadToFile(): URL에서 실행할 파일 다운로드
- CreateProcess(): 다운 받은 파일 프로세스로 실행
- InternetConnect(), HttpOpenRequest(), HttpSendRequest(), InternetReadFile() 같은 API도 사용
- URL을 소셜 네트워크, 정상 사이트, 클라우드 스토리지 등을 사용하면 탐지 어렵고 네트워크 기반 보안 우회 가능

5.2 유저 정의 명령 및 제어

- 유저 정의 프로토콜, 비표준 포트 => 트래픽 은닉
- WSStart(): 윈도우 소켓 시스템 초기화
- GetHostByName(): C2 domain 이름 주소 획득 가능
- inet_ntoa() : IP주소를 ASCII 코드로 변환
- Connect() : IP주소와 연결
- CreateThread(): 새로운 thread 실행

• 악성코드 지속성

- 악성코드가 지속적으로 실행될 수 있게 하는 기술
- 재부팅, 업데이트를 하더라도 지속적으로 실행
- 재부팅을 통한 재부팅으로 인해 권한 상승