Effective Yara

디지털포렌식트랙 김태룡

0. 과제

- 목표 : 바이러스 267개 모두 탐지하는 Yara 룰 작성하기.
- 주어진 샘플들을 모두 탐지할 수 있는 창의적인 Yara룰을 작성하라.
- 제출 Email : kjkwak12@gmail.com

1. Yara

1-1) 소개

- 악성코드의 특성과 행위에 포함된 패턴을 이용하여 악성 파일을 분류하는데 사용되는 툴이다.
- VirusTotal에서 제작하였으며, VirusTotal, FireEye 등에서 사용 중이다.
- 간단한 Rule을 작성하여 패턴에 맞는 악성코드를 판단할 수 있다.
- Rule은 단순 Text String, Binary값뿐만 아니라 Entry Point, File Offset 등의 특정이 가능하다.
- 하지만 동일한 Rule을 계속해서 사용 할 경우, 변종 악성코드 탐지 능력이 떨어진다.
- Yara Docs : https://yara.readthedocs.io/

1-2) Rule

test.yar ①

```
rule samplerule {                                                    ②
    meta:                                                            ③
        author = "FSICEAT"
        type = "APT"
        filetype = "Win32 EXE"
        version = "1.0"
        date = "2017-03-07"
        MD5 = 9394078671922de6b5cd194e3581ec46
        description = "Rule to detect EniumEscape"

    strings:                                                         ④
        $str = "DJU!*JE&!M@UNQ@" full word ascci wide nocase
        $delimiter1 = ":FZ:" fullword ascii wide
        $enc1 = {8A 08 80 F9 ?? 7C ?? 80 F9 ?? 7F 0? B2 DB 2A ?? 8? ??}
        $re_string = /([0-9]{1,3}₩.){3}[0-9]{1,3}/

    condition:                                                       ⑤
        unit16(0) == 0x5A4D and (2 of ($delimiter*) or (2 of ($filename*)) or $enc1) or any of ($str*)
        $enc1 or $str or $restring
}
```

① Rule 파일

- Rule은 텍스트 파일 형태로 저장/관리할 수 있다.

- .yar 확장자를 사용하여 Yara Rule파일을 구분한다.

② Rule Identifier

- rule 으로 시작한다.
- 첫 문자는 숫자가 될 수 없다.
- 대소문자 구분 128자 이내로 작성해야 한다.
- Yara Rule에서 사용되는 예약단어를 Rule Identifier로 설정할 수 없다.

Yara Rule Keywords						
all	and	any	ascii	at	condition	contains
entrypoint	FALSE	filesize	fullword	for	global	in
import	include	int8	int16	int32	int8be	int16be
int32be	matches	meta	nocase	not	or	of
private	rule	strings	them	TRUE	unit8	unit16
unit32	unit8be	unit16be	unit32be	wide		

[표 1-1] Yara Rule 예약 단어

③ meta

- Rule 작성자가 남겨 놓는 정보.
- Author(작성자), Date(작성날짜), Description(설명), MD5(샘플Hash)를 남겨 주어야 한다.

④ strings

- 각 변수는 $로 시작한다.
- 변수 명은 알파벳과 언더 바 _ 를 사용할 수 있다.
- 변수 명은 알아보기 쉽게 작성하는 것이 좋다.
- 문자열 혹은 16진수 형태로 정의할 수 있다.
- 문자열 사용 시, 쌍 따옴 표 " " 를 사용한다.
- 16진수 형태일 경우, 중괄호 { } 를 사용한다.
- 16진수 형태의 문자열은 띄어쓰기 하지 않아도 인식한다.
- 16진수 형태에 물음표 ? 대괄호 [-] 파이프라인 | 을 통해 유연한 패턴을 생성할 수 있다.
  EX) 중간에 임의의 1자리가 들어가는 상황 : {E2 34 3D C8 A4 FB}
  EX) 중간에 임의의 2자리가 들어가는 상황 : {E2 34 3? C8 A4 FB}
  EX) 중간에 4자리에서 6자리의 임의의 값이 들어가는 상황 : {E2 34 [4-6] A4 FB}
  EX) 중간에 62 B4 혹은 56이 들어가는 상황 : {E2 34 (62 B4 | 56) C8 A4 FB}
- 10진수는 16진수 문자열에 사용할 수 없다.
- 정규표현식 사용 시 슬래시 / / 를 사용한다.

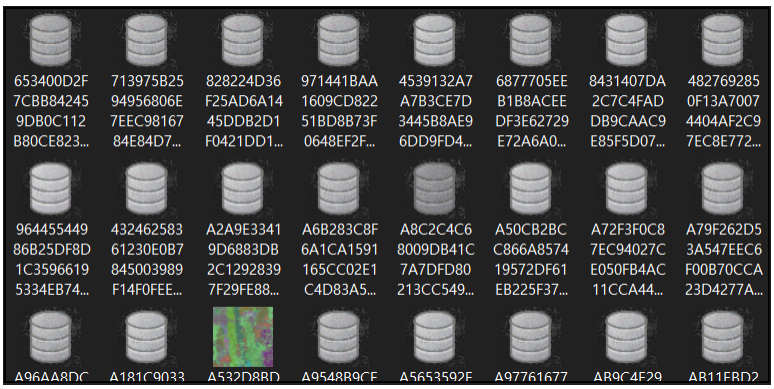
⑤ condition

- and, or, not 조건을 걸 수 있다.
- any of them, all of them, 1 of them, 3 of them 조건을 걸 수 있다.
- 작성 후, VirusTotal에 올렸을 때, 경고 메시지가 나온다면 그 부분은 제거하는 것이 좋다.

2. 패턴 분석

2-1) 외부적 패턴

- 멘토님께서 공유하신 267개의 악성 프로그램을 살펴보니, 대부분 데이터베이스 아이콘 형태를 취하고 있었으며, 0~9, A~F까지 16진수로 이루어진 파일 명을 가지고 있었다.



[그림 2-1] 대부분 데이터베이스 아이콘인 모습

- 파일 크기 또한 가장 큰 것은 562KB, 가장 작은 것이 118KB로, 1MB를 넘지 않았다.

파일 크기	개수
184KB 이상	13개
179KB	147개
178KB	53개
173KB	44개
154KB 이하	10개

[표 2-1] 파일 크기별 개수

- 만든 날짜는 모두 2020년 07월 13일 오후 1시 44분으로 동일하였으나, 수정 날짜는 달랐다. 압축 해제 작업이 일어남에 따라 만든 날짜가 수정 날짜보다 뒤쳐진 것을 추측할 수 있다.

수정 일자	개수
오전 9시 이전	3개
오전 9시	183개
오전 10시	52개
오전 11시	15개
오후	14개

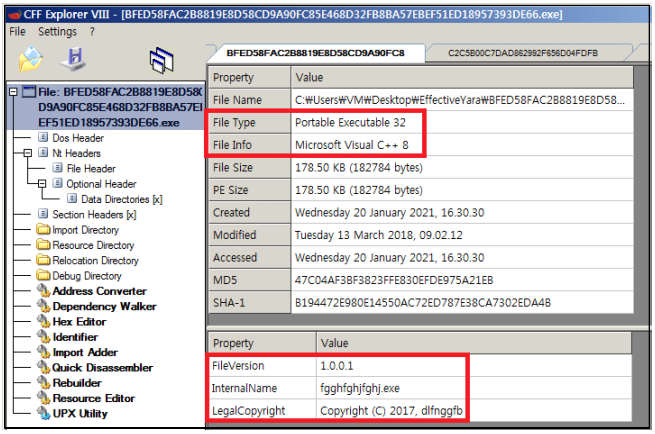
[표 2-2] 수정일자

- 현재 PC시간이 KST로 설정되어있음을 감안하여 한국 근무시간(오전9시 ~ 오후 6시)과 동일하므로, 국내에서 만들어진 샘플로 추정되며, 수정일자의 대부분이 오전 9시부터 오후 4시 50분 이전에, 12시 이후부터 2시 사이시간은 비워져 있음을 보아, 이 샘플을 생성한 사람은 개인이 아닌 특정 조직에서 근무하고 있을 가능성이 높아 보였다.
- 2017년도 수정 된 것이 2개, 2018년도에 수정 된 것이 265개였다.

2-2) 내부적 패턴

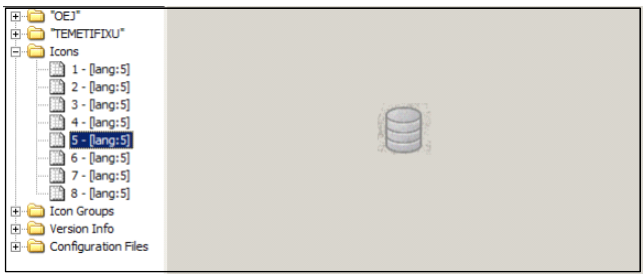
- CFF Explorer를 이용하여 기본 정보를 살펴보니, 대부분 아래와 같은 특징을 가지고 있었다.
  ① 32비트 PE 파일

② Microsoft Visual C++ 8 사용
③ 파일 버전 1.0.0.1
④ ?gghfghjfghj.exe 의 InternalName (?는 알파벳 소문자)
⑤ Copyright (C) 2017, [8-10]의 LegalCopyright (8~10글자는 알파벳 소문자)



[그림 2-2] CFF Explorer로 살펴본 파일 기본 정보

⑥ 섹션 테이블을 확인 해 보니, text, rdata, data, gfids, rsrc, reloc 테이블만 사용하고 있었다.
⑦ Import Directory에서는 KERNEL32.dll, USER32.dll을 모두 공통적으로 사용하고 있었으며, 대부분 ADVAPI32.dll과 SHELL32.dll을 포함하고 있었다.
⑧ Rich 헤더가 확인되었기 때문에 VisualStudio를 통해 컴파일 되었음을 알 수 있다.
⑨ 사용된 리소스를 확인 해 보면 모두 8개 이상의 아이콘을 포함하였으며, 그 중 데이터베이스 모양 아이콘이 가장 많았다.



[그림 2-3] 데이터베이스 아이콘이 확인된 모습

2-3) 패턴 중요도 매기기

- 알아낸 패턴을 모두 적용한다면, 분명 267개의 바이러스 샘플을 모두 걸러낼 수 있을 것이다.
- 하지만 최소한의 룰을 이용하여 267개를 탐지해야하는 조건이 걸려 있다.
- 또한 2차 프로젝트 기간 때 필자의 실수로 프로그램 정확도가 낮아 졌을 때 들은 꾸중이 있다.

> **미탐이 있을지언정 오탐이 일어나서는 안 된다.**
>
> [김종민 멘토님]

- 따라서 찾아낸 패턴 중, 오탐 가능성이 가장 낮은 패턴만을 룰에 적용시키기로 하였다.

패턴	오탐 가능성[1]
0 ~ 9, A ~ F 16진수로 이루어진 64자리 파일 이름	★★★★☆
exe 확장자	★★★★★
100KB 이상, 600KB 이하 파일 크기	★★★★☆
만든 날짜 2020년 07월 13일 오후 1시 44분	★☆☆☆☆
오전 9시 ~ 12시 사이 집중된 수정 시각	★★★☆☆
2017년 이상 2018년 이하 수정 시각	★★★★☆
32비트 PE 파일	★★★★★
Microsoft Visual C++ 8 컴파일러 사용	★★★☆☆
파일 버전 10.0.0.1	★★★★☆
?gghfghjfghj.exe 의 InternalName	★☆☆☆☆
Copyright (C) 2017, [8-10]의 LegalCopyright	★☆☆☆☆
text, rdata, data, gfids, rsrc, reloc 테이블만 사용	★★★☆☆
KERNEL32.dll, USER32.dll, ADVAPI32.dll, SHELL32.dll 사용	★★★★☆
Rich 헤더 사용	★★★★☆
데이터베이스 모양 아이콘 포함	★☆☆☆☆

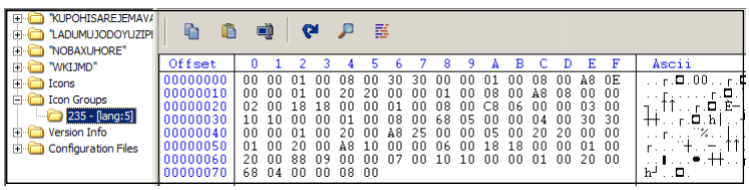
[표 2-3] 찾아낸 패턴과 그에 대한 오탐 가능성

- 64자리 해시로 이루어진 파일 이름의 경우, 바이러스 샘플 다운로드 사이트에서 받는 바이러스 대부분이 64자리 해시 네이밍을 가지기에 오탐 가능성을 높게 잡았다.
- 만든 날짜가 2020년 07월 13일 오후 1시 44분인 경우는 오탐 가능성은 굉장히 낮지만, 앞으로 해당 바이러스의 변종에 변종, 혹은 버전 2가 나왔을 경우 탐지하지 못하므로 제외하기로 하였다.

3. Yara 룰 작성

3-1) 아이콘 포함 여부 탐지

- 가장 오탐 가능성이 낮고, 가시적인 항목인 아이콘에 대한 탐지 룰을 우선 작성하기로 하였다.
- CFF Explorer로 아이콘 파일 하나하나 다운로드 받아 적용 시키려니 탐지율이 너무 낮았으며, 267개 각 항목마다 최소 아이콘이 8개 이상 포함되어있었기에 시간이 너무 오래 걸렸다.
- 아이콘 파일을 등록하는 것이 비효율적이었기에, IconGroup을 이용하기로 하였다.



[그림 3-1] 아이콘 그룹

- 아이콘 그룹에 나타난 HEX 값을 Yara 룰에 적용시켰더니, 264개의 악성프로그램이 탐지되었다.
- 탐지되지 않은 3가지 악성코드에 대해 살펴보니, IconGroup이 살짝 다른 2개와 완전히 박살난 프로그램[2] 1개를 발견할 수 있었다.
- IconGroup이 살짝 달랐던 두 파일은 ? 기호를 사용하여 성공적으로 탐지할 수 있었다.

1) 다른 바이러스 샘플 파일과 비교했을 때의 오탐 가능성.
2) 5E8EF8049F64AE60995ED8CDB3A023EB7C4207081B9709707505127007A9139E.exe

IconGroup.yar

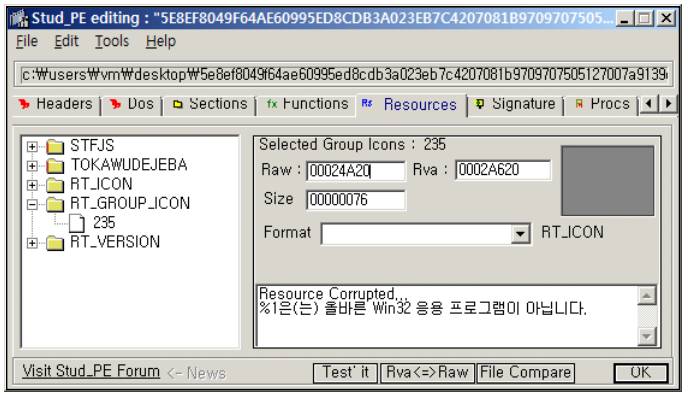
```
rule iconGroup {
    meta:
        author = "GoldBigDragon"
        type = "Malware"
        filetype = "Win32 EXE"
        version = "1.0"
        date = "2021-01-21"
        description = "Rule to detect various malware"

    strings:
        $iconGroup = {0000010008003030000001000800A80E00000?0020200000001000800A8080000}

    condition:
        all of them
}
```

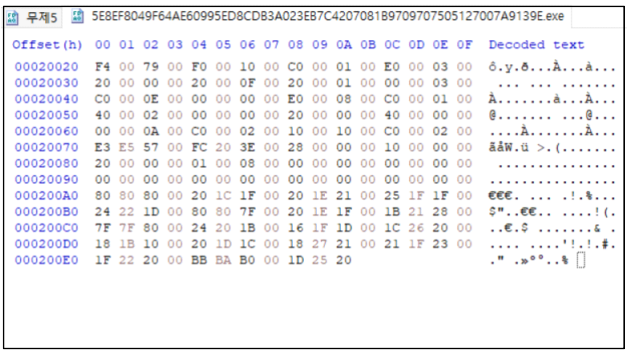
탐지 항목 : 266개

- 다행히 3일 전 최원영 멘토님의 PE파일 과제를 수행하였기에 박살난 파일을 분석할 수 있었다.
- .reloc 테이블의 물리적 위치가 존재하지 않는 주소로 잡혀 있었으며, 이를 임시로 .rsrc 테이블의 물리적 위치와 동일한 곳으로 잡아 주었더니 실행 시 오류가 일어나지는 않았다.
- 고쳐진 파일을 Stud PE으로 확인하니, GROUP_ICON의 RAW가 0x00024A20임을 알게 되었다.



[그림 3-2] GROUP_ICON의 물리 위치를 찾은 모습

- 하지만 HxD를 통해 파일을 확인 해 보면, 0x000200E0으로 끝나는 모습을 확인 할 수 있었다.



[그림 3-2] 0x000200E0가 최종 주소인 모습

- 원래 Icon을 포함한 악성코드였으나, 누군가 프로그램을 수정 했다고 볼 수 있는 정황이며, 아이콘 탐색 룰 한 줄로 과제를 끝낼 수 없도록 하기 위한 멘토님의 조치일 가능성이 커 보였다.

- GROUP ICON을 사용할 수 없기에 오탐율이 낮은 다른 요소를 적용 해 보기로 하였다.
- LegalCopyright의 경우, 컴파일 년도에 따라 앞부분이 달라질 수도 있으며, 뒤의 임의 문자가 8 ~ 10글자 사이라는 점에서 항상 12글자에 앞부분만 바뀌는 InternalName을 룰에 적용시키는 것이 더욱 합리적이라 판단되어 InternalName을 룰에 추가시키기로 하였다.

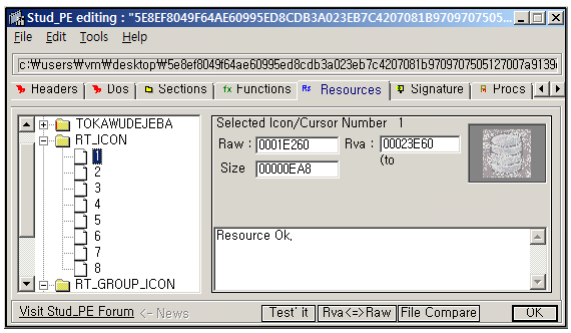
InternalName.yar

```
import "pe"
rule internalName {
    meta:
        author = "GoldBigDragon"
        type = "Malware"
        filetype = "Win32 EXE"
        version = "1.0"
        date = "2021-01-21"
        description = "Rule to detect various malware"

    condition:
        pe.version_info["InternalName"] contains "gghfghjfghj"
}
```

탐지 항목 : 177개

- 하지만 .reloc영역이 날아가면서 InternalName은커녕 LegalCopyright 정보도 획득 할 수 없었다.
- 이에 Stud PE로 계속해서 훑어보던 찰나, ICON GROUP 영역은 잘려나갔지만, 데이터베이스 모양 아이콘 1, 2, 3이 온전히 남아있는 모습을 발견하였다.



[그림 3-3] 4번부터 8번까지는 잘려있다.

IconAndIconGroup.yar

```
rule iconAndIconGroup {
    meta:
        author = "GoldBigDragon"
        type = "Malware"
        filetype = "Win32 EXE"
        version = "1.0"
        date = "2021-01-21"
        description = "Rule to detect various malware"

    strings:
        $iconGroup = {00000100080030300000001000800A80E00000?00202000000100080 0A8080000}
        $icon = {EFEFEFEFEF6B6BEF6D6D6D6DEFEF6DEFEF6D6DEF6D56EFEFEFEFEFEFEF6B746B}

    condition:
        1 of them
}
```

탐지 항목 : 267개

3-2) 오탐 가능성 확인

- 룰에 대한 오탐/미탐 가능성을 확인하기 위해 다양한 모집단을 이용 해 보았다.
- 임의 멀웨어 100가지를 다운로드 받은 다음, 앞서 작성한 Yara rule을 적용 시켜 보았다.

멀웨어 구분	결과	멀웨어 구분	결과
Android.PegasusB	미인식	Android.Skygofree	미인식
Android.Spy.49_iBanking_Feb2014	미인식	Android.VikingHorde	미인식
AndroRat_6Dec2013	미인식	AntiExe.A	미인식
Artemis	미인식	Backdoor.MSIL.Tyupkin	미인식
BAT.Drop	미인식	BAT.Pot.A	미인식
BAT.Skul	미인식	BlackEnergy2.1	미인식
Brain.A	미인식	Careto_Feb2014	미인식
Cascade.1701.W	미인식	Catapillar.E	미인식
Civil_War.282	미인식	Coll.CozyBear	미인식
Coll.DarkHydrus	미인식	CryptoLocker_10Sep2013	미인식
CryptoLocker_20Nov2013	미인식	CryptoLocker_22Jan2014	미인식
Dino	미인식	DOS.Yesmile	미인식
Dropper.Taleret	미인식	Duqu2	미인식
Dyre	미인식	EquationGroup	미인식
EquationGroup.DoubleFantasy	미인식	EquationGroup.EquationDrug	미인식
EquationGroup.EquationLaser	미인식	EquationGroup.Fanny	미인식
EquationGroup.GrayFish	미인식	EquationGroup.GROK	미인식
EquationGroup.TripleFantasy	미인식	FancyBear.GermanParliament	미인식
Form.A	미인식	Friday_the_13th.408	미인식
Friday_the_13th.416.A	미인식	Friday_the_13th.416.B	미인식
Friday_the_13th.540.A	미인식	Green_Caterpillar.1575.A	미인식
IllusionBot_May2007	미인식	INTC.A	미인식
JS.JScript.A	미인식	JS.Lame	미인식
Jumper.B	미인식	Junkie	미인식
Kampana.A	미인식	Kelihos	미인식
Keylogger.Ardamax	미인식	KRBanker	미인식
Linux.Chapros.A	미인식	Linux.Encoder.1	미인식
Linux.Mirai.B	미인식	Linux.Snoopy.A	미인식
Linux.Snoopy.B	미인식	Linux.Snoopy.C	미인식
Linux.Wirenet	미인식	Michelangelo	미인식
Net-Worm.Win32.Kido	미인식	Neurevt.1.7.0.1	미인식
Nitlove	미인식	Nivdort	미인식
NYB.B	미인식	OSX.Backdoor.iWorm	미인식
OSX.HellRaiser	미인식	OSX.JacksBot	미인식
OSX.Lazarus	미인식	OSX.MacSecurity	미인식
OSX.OceanLotus	미인식	OSX.Wirenet	미인식
OSX.XAgent	미인식	Parity_Boot.B	미인식
Phoenix.2000	미인식	PlugX	미인식
PotaoExpress	미인식	Poweliks	미인식
Proteus	미인식	Quax.A	미인식
Ransomware.Cerber	미인식	Ransomware.Cryptowall	미인식
Ransomware.Jigsaw	미인식	Ransomware.Locky	미인식
Ransomware.Mamba	미인식	Ransomware.Matsnu	미인식
Ransomware.Petrwrap	미인식	Ransomware.Petya	미인식
Ransomware.Radamant	미인식	Ransomware.Rex	미인식
Ransomware.Satana	미인식	Ransomware.TeslaCrypt	미인식
Ransomware.Thanos	미인식	Ransomware.Unnamed_0	미인식
Ransomware.Vipasana	미인식	Ransomware.WannaCry	미인식
Ransomware.Wannacry_Plus	미인식	Rombertik	미인식
Rustock	미인식	Win32Dircrypt.Trojan.Ransom.ABZ	미인식

[표 3-1] 멀웨어에 대한 Yara rule 적용 결과

- 멀웨어에 대한 결과가 모두 미인식으로 나와서 다행이었지만, 혹시나 멀웨어 한정으로 잘 적용되는 룰 일까봐 멀웨어보다 용량이 큰 일반 EXE 파일 40건에 대한 실험을 해 보았다.

파일명	크기	확장자	결과	파일명	크기	확장자	결과
jdk-8u121-windows-x64	195MB	exe	미인식	eclipse-inst-win64	44.7MB	exe	미인식
CDSpace8Setup	2.68MB	exe	미인식	HxDPortableSetup	3.21MB	exe	미인식
nox_setup_v5.0.0.1_full_intl	277MB	exe	미인식	InteliJ idealC-2020.3.1	617MB	exe	미인식
jd-gui	8.48MB	exe	미인식	JANDI	70.8MB	exe	미인식
SandboxieInstall	5.93MB	exe	미인식	KakaoTalk_Setup	61.2MB	exe	미인식
VMware-player-12.5.0-4352439	74.6MB	exe	미인식	Nox_setup_v7.0.0.8_full_intl	413MB	exe	미인식
paint.net.4.0.11.install	6.74MB	exe	미인식	npp.7.8.8.Installer	3.59MB	exe	미인식
oCam_v465.0_ko	8.79MB	exe	미인식	Spyder_64bit_full	169MB	exe	미인식
DaVinci_Resolve_15.2.3_Windows	0.98GB	exe	미인식	SteamSetup	1.50MB	exe	미인식
PotPlayerSetup	19.3MB	exe	미인식	Ultra iso	4.32MB	exe	미인식
SamsungUniversalPrintDriver3	25.3MB	exe	미인식	VirtualBox-6.1.16-140961-Win	103MB	exe	미인식
ALSong336	13.2MB	exe	미인식	VSCodeUserSetup-x64-1.31.0	46.6MB	exe	미인식
audacity-win-2.1.2	252MB	exe	미인식	Wireshark-win64-3.4.2	58.6MB	exe	미인식
cacaoencoder_setup_1.82	14.5MB	exe	미인식	yara64	2.08MB	exe	미인식
GitHubSetup	663KB	exe	미인식	yarac64	2.04MB	exe	미인식
BANDIZIP-SETUP-KR	4.70MB	exe	미인식	Setup	1.97MB	exe	미인식
AccessData FTK Imager 3.4.2	41.2MB	exe	미인식	dotnetfx	22.4MB	exe	미인식
HashTab_v6.0.0.34_Setup	1.11MB	exe	미인식	visualvm	217KB	exe	미인식
ChromeSetup	1.25MB	exe	미인식	nbexec64	205KB	exe	미인식
DaumCleaner64	2.55MB	exe	미인식	DiscordSetup	65.6MB	exe	미인식

[표 3-2] 일반 프로그램에 대한 Yara rule 적용 결과

4. 결론

- 오직 '과제'만을 위해 오탐을 무시하고 짧고 창의적인 룰을 설정한다면, [표 2-3]에 쓰인 대로 만든 날짜가 2020년 07월 13일 오후 1시 44분인 파일만을 짚어내는 룰을 설정하면 되었다.
- 하지만 동일 악성코드에 대한 버전 업데이트, 변종의 출현, 다른 멀웨어와 섞였을 때 등 실질적인 위협 요소까지 생각한다면 가장 짧은 룰이 가장 좋은 것만은 아님을 알 수 있었다.