Eagle Eye

샌드박스 시대의 종결, AI 에뮬레이터 시대의 도래

최원혁 : ㈜누리랩 대표/연구소장

Eagle Eye: 2008년 개봉한 영화로 AI 시스템 이글아이가 핸드폰, 현금지급기, 거리의 CCTV, 교통안내 LED사인보드, 신호등 등 우리 주변의 전자장치와 시스템을 모니터링 하여 사람들을 AI가 원하는 방향으로 행동하도록 조정하는데에서 착안

CodeEngn
www.CodeEngn.com
2021 CodeEngn Conference 17

최원혁

hanul93@gmail.com

이력

1993.07 : 악성코드 분석 시작
1995.10 : 키콤백신 개발 발표 (국내 4번째 백신)
1998.03 : ㈜하우리 창업 (악성코드 분석 및 바이로봇 개발 총괄 / CTO)
2009.03 : 동국대학교 국제정보대학원 정보보호학과 사이버포렌식전공 교수
2015.05 : ㈜누리랩 창업 (대표/연구소장)
2020.09 : 이화여자대학교 엘텍공과대학 소프트웨어학부 사이버보안 겸임교수
2021.03 : 고려대학교 인공지능사이버보안학과 겸임교수

활동

2004.08 : 민관합동조사단 1~4기 활동 (악성코드 분석 전문)
2009.01 : 국가정보원장 표창 수상
2014.03 : 사이버보안전문단 (미래창조과학부 임명)
2021.03 : 사이버작전사령부 자문 위원

집필

2017.09 : 파이썬으로 배우는 Anti-Virus 구조와 원리

⋮

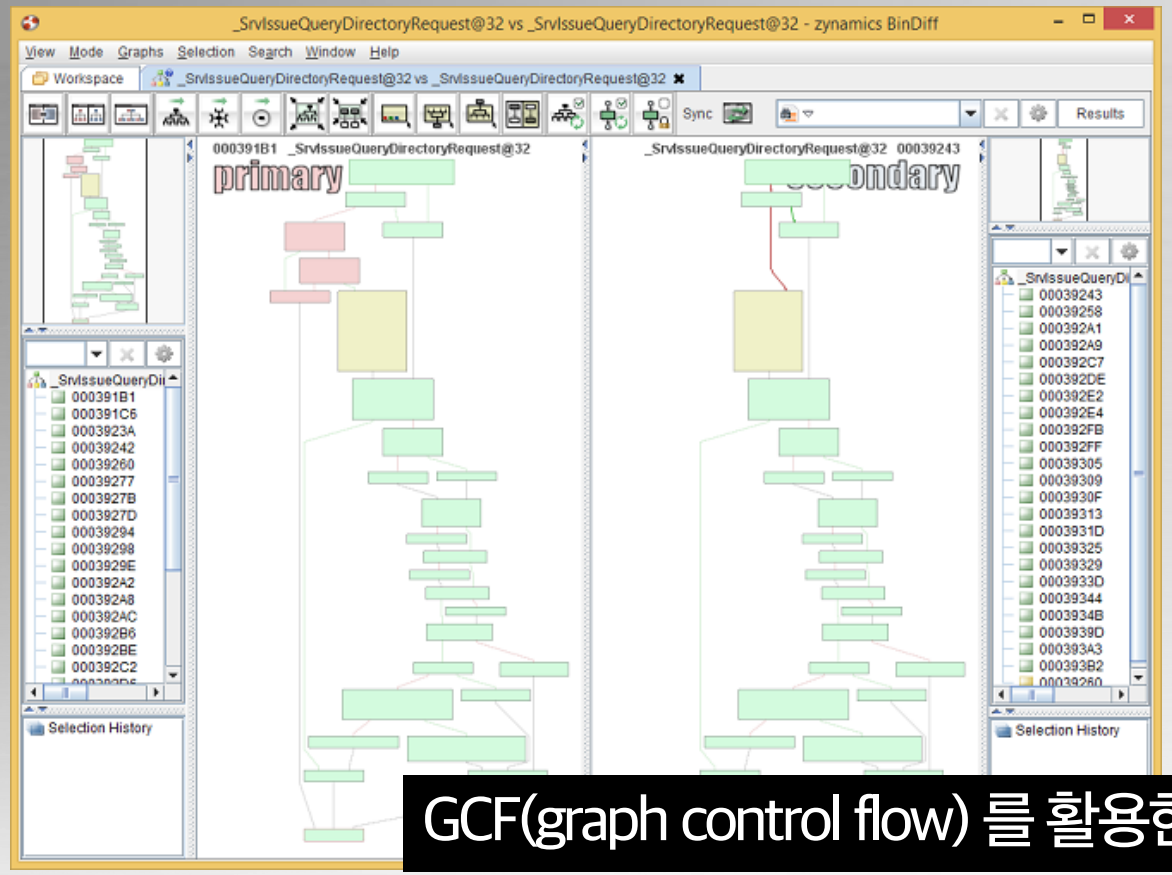
인터넷, 고객사를 통해 모은 악성코드 샘플들…

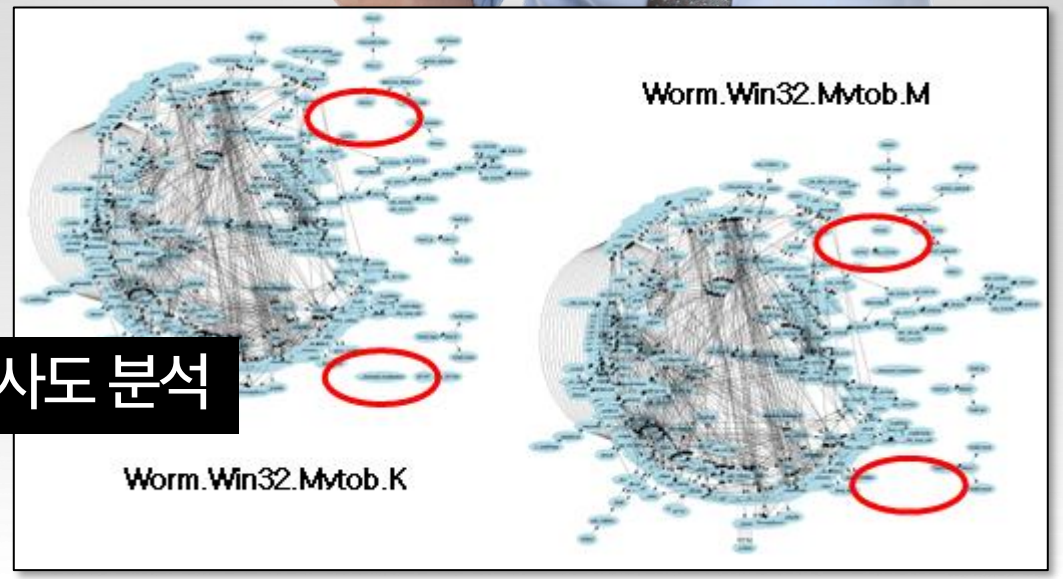
많아도 너~무 많아!!!

backdoor.msil.nanobot.aagj
backdoor.msil.nanobot.jjh
backdoor.msil.nanobot.lsp
backdoor.msil.nanobot.ohf
backdoor.msil.nanobot.oxg
backdoor.msil.nanobot.pax
backdoor.msil.nanobot.pdj
backdoor.msil.nanobot.pfl
backdoor.msil.nanobot.pqu
backdoor.msil.nanobot.pur
backdoor.msil.nanobot.pwp
backdoor.msil.nanobot.pyv
backdoor.msil.nanobot.rcg
backdoor.msil.nanobot.rcn
backdoor.msil.nanobot.rgw
backdoor.msil.nanobot.rpd
backdoor.msil.nanobot.rqw
backdoor.msil.nanobot.seb
backdoor.msil.nanobot.tgu
backdoor.msil.nanobot.tpv
backdoor.msil.nanobot.und
backdoor.msil.nanobot.upl
backdoor.msil.nanobot.uwc
backdoor.msil.nanobot.ved
backdoor.msil.nanobot.vex
backdoor.msil.nanobot.vht
backdoor.msil.nanobot.vlt
backdoor.msil.nanobot.vpa
backdoor.msil.nanobot.vqa
backdoor.msil.nanobot.vqk
backdoor.msil.nanobot.vww
backdoor.msil.nanobot.vxc
backdoor.msil.nanobot.vxv
backdoor.msil.nanobot.wbk
backdoor.msil.nanobot.wcl
backdoor.msil.nanobot.wcn
backdoor.msil.nanobot.wda
backdoor.msil.nanobot.wdm
nobot.wdz
ot.wee
t.wem
t.wfh
t.wfi
backdoor.msil.nanobot.xmg
backdoor.msil.nanobot.xnr
backdoor.msil.nanobot.xpo
backdoor.msil.nanobot.xpx
backdoor.msil.nanobot.xqm
backdoor.msil.nanobot.xrt
backdoor.msil.nanobot.xrz
backdoor.msil.nanobot.xsf
backdoor.msil.nanobot.xtd
backdoor.msil.nanobot.xvy
backdoor.msil.nanobot.xwe
backdoor.msil.nanobot.xwt
backdoor.msil.nanobot.xxl
backdoor.msil.nanobot.xxw
backdoor.msil.nanobot.xyo
backdoor.msil.nanobot.yae
backdoor.msil.nanobot.ybt
backdoor.msil.nanobot.yev
backdoor.msil.nanobot.ygi
backdoor.msil.nanobot.ygo
backdoor.msil.nanobot.ygz
backdoor.msil.nanobot.yhv
backdoor.msil.nanobot.yik
backdoor.msil.nanobot.yix
backdoor.msil.nanobot.yja
backdoor.msil.nanobot.yme
backdoor.msil.nanobot.ymf
backdoor.msil.nanobot.ymg
backdoor.msil.nanobot.ymh
backdoor.msil.nanobot.ymj
backdoor.msil.nanobot.ymm
backdoor.msil.nanobot.ymn
backdoor.msil.nanobot.ynz
backdoor.msil.nanobot.yow
backdoor.msil.nanobot.ytk
backdoor.msil.nanobot.yuw
backdoor.msil.nanobot.yvh
backdoor.msil.nanobot.yvq
backdoor.msil.nanobot.ywf
backdoor.msil.nanobot.yzs
backdoor.win32.agent.amb
backdoor.win32.agent.bxhj
backdoor.win32.androm.dds
backdoor.win32.nbdd.ofp
backdoor.win32.nbdd.ogd
backdoor.win32.nbdd.oib
backdoor.win32.nbdd.oiv
backdoor.win32.nbdd.ojm
backdoor.win32.nbdd.ojt
backdoor.win32.nbdd.vhq
backdoor.win32.nbdd.wef
backdoor.win32.nbdd.wja
backdoor.win32.nbdd.wjx
backdoor.win32.nbspy
backdoor.win32.ncx.b
backdoor.win32.netbus.20.d
backdoor.win32.netbus.160.a
backdoor.win32.netbus.170
backdoor.win32.netbus.ripper
backdoor.win32.netbus.toy
backdoor.win32.netdevil.14
backdoor.win32.nethief.10
backdoor.win32.nethief.104
backdoor.win32.nethief.am
backdoor.win32.netspy.10
backdoor.win32.netspy.10.a
backdoor.win32.netspy.10.b
backdoor.win32.netspy.20.j
backdoor.win32.netwiredrc.ash
backdoor.win32.netwiredrc.ato
backdoor.win32.netwiredrc.awz
backdoor.win32.netwiredrc.bcn
backdoor.win32.netwiredrc.bef
backdoor.win32.netwiredrc.bfi
backdoor.win32.netwiredrc.cgb
backdoor.win32.netwiredrc.cpf
backdoor.win32.netwiredrc.cvk
backdoor.win32.netwiredrc.cxf
backdoor.win32.netwiredrc.czg
backdoor.win32.netwiredrc.das
backdoor.win32.netwiredrc.djm
backdoor.win32.netwiredrc.dnc
backdoor.win32.netwiredrc.dpz
backdoor.win32.netwiredrc.dst
backdoor.win32.netwiredrc.dwf
backdoor.win32.netwiredrc.ebe
backdoor.win32.netwiredrc.edr
backdoor.win32.netwiredrc.edu
backdoor.win32.netwiredrc.ees
backdoor.win32.netwiredrc.eet
backdoor.win32.netwiredrc.efi
backdoor.win32.netwiredrc.efp
backdoor.win32.netwiredrc.egm
backdoor.win32.netwiredrc.egy
backdoor.win32.netwiredrc.ehg
backdoor.win32.netwiredrc.ehz
backdoor.win32.netwiredrc.eiu
backdoor.win32.netwiredrc.eix
backdoor.win32.netwiredrc.ejc
backdoor.win32.netwiredrc.ejq
backdoor.win32.netwiredrc.elf
backdoor.win32.netwiredrc.elv
backdoor.win32.netwiredrc.elz
backdoor.win32.netwiredrc.em
backdoor.win32.netwiredrc.qf
backdoor.win32.poison.ggrf
backdoor.win32.sdbot.awk
backdoor.win32.valvoline
backdoor.win32.vb.rm
backdoor.win32.winuoj.lps
email-worm.win32.netsky.ab
email-worm.win32.netsky.ghc
email-worm.win32.netsky.q
email-worm.win32.netsky.r
email-worm.win32.netsky.t
email-worm.win32.netsky.viy
email-worm.win32.netsky.y
email-worm.win32.runouce.b
exploit.win32.nuker.noname.b
heur_backdoor.msil.generic
heur_backdoor.win32.generic
heur_exploit.win32.generic
heur_trojan.msil.generic
heur_trojan.win32.bayrob.gen
heur_trojan.win32.daserf.gen
heur_trojan.win32.generic
heur_trojan-banker.win32.kora.gen
heur_trojan-downloader.win32.generic
heur_trojan-psw.win32.generic
net-worm.win32.bobic.q
p2p-worm.win32.niklas.w
packed.win32.krap.as
packed.win32.krap.b
packed.win32.krap.g
packed.win32.nsanti.a
packed.win32.nsanti.b
packed.win32.nsanti.r
packed.win32.pepatch.iu
packed.win32.polycrypt.b
packed.win32.polycrypt.h
rootkit.win32.lapka.an
rootkit.win32.necurs.iy
trojan.msil.agent.forb
trojan.msil.agent.fowx
trojan.msil.agent.fpar
trojan.msil.agent.fpcm
trojan.msil.agent.fpcy
trojan.msil.agent.fpdx
trojan.msil.agent.fpgc
trojan.msil.agent.fpvk
trojan.msil.crypt.emzr
trojan.msil.shopbot.bdk
trojan.win32.agent.ihpl
trojan.win32.agent.neyyyz
trojan.win32.agent.uxnz
trojan.win32.agentb.btqd
trojan.win32.agentb.btsk
trojan.win32.agentb.bttf
trojan.win32.agentb.burc
trojan.win32.agentb.busv
trojan.win32.agentb.bvcw
trojan.win32.agentb.bvgc
trojan.win32.agentb.bwbp
trojan.win32.agentb.bwhd
trojan.win32.agentb.bwnw
trojan.win32.agentb.bwvi
trojan.win32.agentb.bwye
trojan.win32.agentb.bxak
trojan.win32.agentb.bxbo
trojan.win32.agentb.bxcn
trojan.win32.agentb.bxcq
trojan.win32.agentb.bxct
trojan.win32.diskwriter.cu
trojan.win32.diss.susqi
trojan.win32.fleercivet.cah
trojan.win32.fsysna.eovm
trojan.win32.genome.yki
trojan.win32.hrup.a
trojan.win32.inject.aeffa
trojan.win32.inject.blcq
trojan.win32.inject.sbrr
trojan.win32.mucc.bqh
trojan.win32.mucc.bwb
trojan.win32.mucc.bxz
trojan.win32.mucc.byk
trojan.win32.mucc.byl
trojan.win32.mucc.bym
trojan.win32.mucc.byn
trojan.win32.mucc.byo
trojan.win32.mucc.byp
trojan.win32.mucc.byq
trojan.win32.mucc.byr
trojan.win32.mucc.byu
trojan.win32.neurevt.aaeb
trojan.win32.neurevt.aaeh
trojan.win32.neurevt.anc
trojan.win32.neurevt.cbk
trojan.win32.neurevt.dub
trojan.win32.neurevt.dvv
trojan.win32.neurevt.dvy
trojan.win32.neurevt.kk
trojan.win32.neurevt.vhr
trojan.win32.neurevt.way
trojan.win32.neurevt.wfh
trojan.win32.neurevt.wgr
trojan.win32.neurevt.xaq
trojan.win32.neurevt.xar
trojan.win32.neurevt.yto
trojan.win32.neurevt.yuu
trojan.win32.neurevt.yvd
trojan.win32.neurevt.yyn
trojan.win32.neurevt.yzs
trojan.win32.neurevt.yzu
trojan.win32.neurevt.zap
trojan.win32.neurevt.zbj

유사도의 필요성 대두

최대한 비슷한 유형끼리 묶어서 분석해야 시간을 절약하지



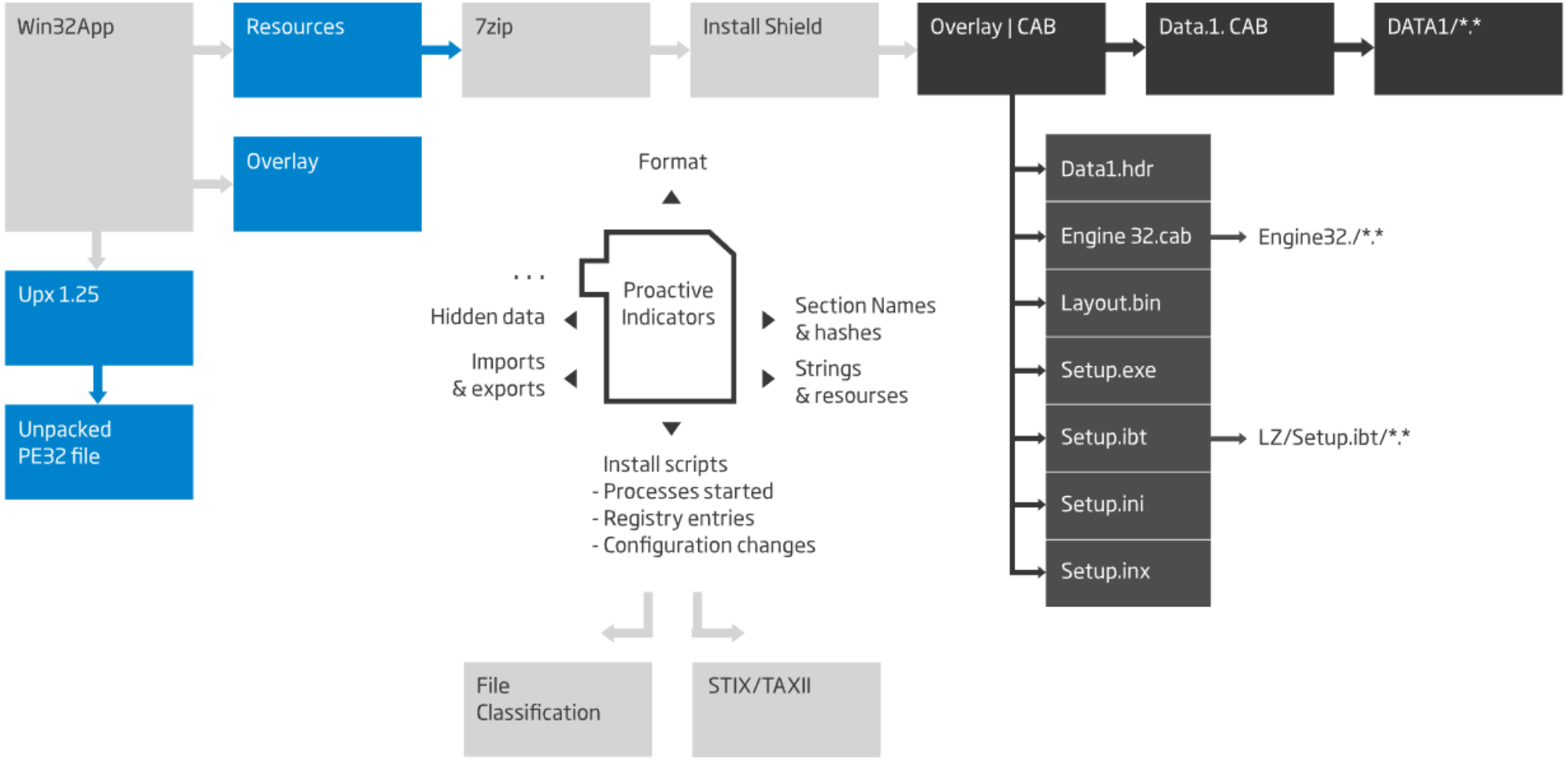


GCF(graph control flow) 를 활용한 유사도 분석

대다수 윈도우 악성코드들은 패킹되어 있다.

언팩을 해야 하는데…

이미지 출처 : https://www.reversinglabs.com/technology/active-file-decomposition



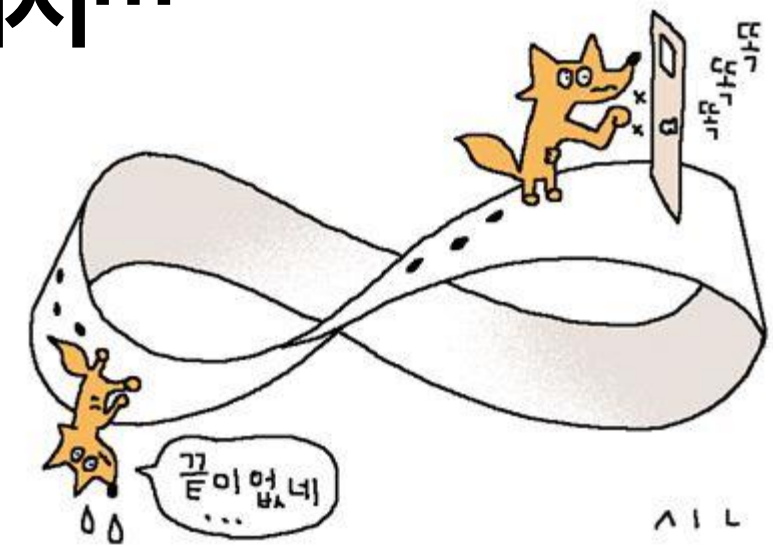
샌드박스를 사용하여 언팩을 진행하면 알려지지 않은 패킹도 처리할 수 있습니다.

↳ **하지만, 샌드박스 우회 악성코드도 있죠 !!**

↳ **그래서 이런 연구도 있습니다.**

↳ **그걸 또 우회하는 것도 있죠 !**

↳ **그래서…**



Cuckoo Sandbox를 회피하는 악성코드 탐지 방안 연구

정제성*, 김광조*

*한국과학기술원 전산학부

A Study on Detection of Evasive Malware in Cuckoo Sandbox

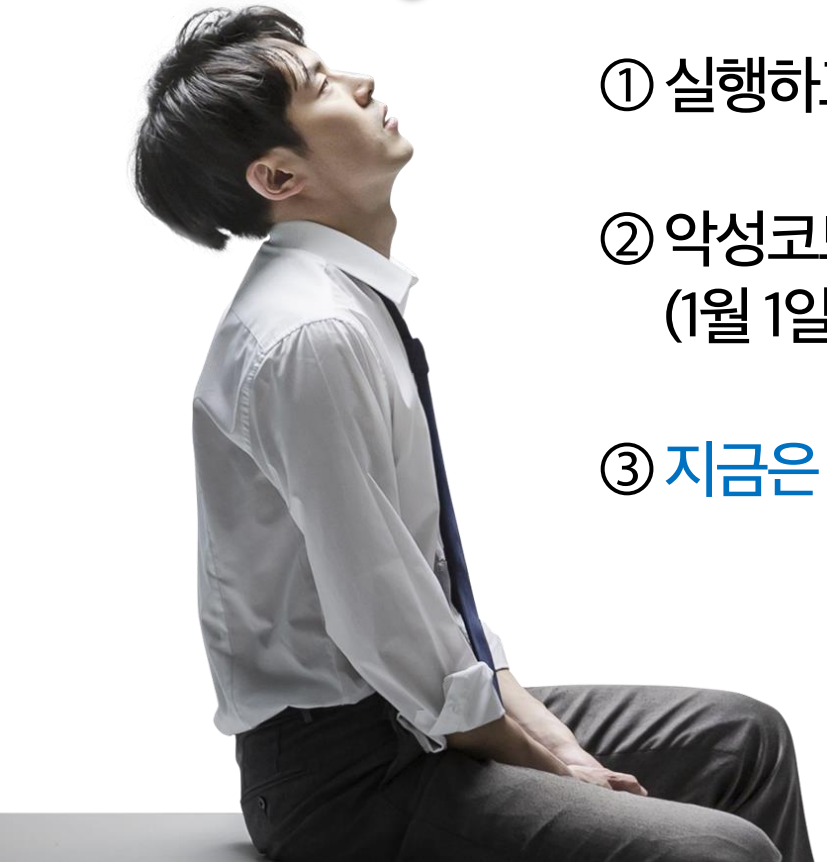
Je-Seong Jeong*, Kwangjo Kim*

*School of Computing, KAIST

요 약

악성코드는 정상적으로 시스템 사용을 못하게 하는 것으로서 컴퓨터 바이러스 부터 웜, 트로이 목마 등 그 종류도 다양하다. 이러한 악성코드를 분석하는 방법에는 정적분석과 동적분석이 있다. 이 둘은 각각의 장·단점이 있지만 요즘과 같이 악성코드의 수가 많은 시대에는 코드를 분석하는 정적분석 보다는 악성코드를 실행시켜 행위를 분석하는 동적분석이 시간적인 측면에서 봤을 시 좀 더 효율적이다. 그리고 요즘 동적분석으로 많이 사용하고 있는 것 중의 하나는 오픈소스 악성코드 자동 분석 시스템인 쿠루 샌드박스(Cuckoo Sandbox)이다. 하지만 요즘 악성코드는 분석 시스템을 우회하는 기능까지 탑재하여 분석을 어렵게 하고 있다. 따라서 본 논문에서는 이런 우회 기법을 분석 후 쿠루 샌드박스에 적용하여 탐지 기능을 향상시키는 방안을 제안하고자 한다.

악성코드를 무조건 실행할 수 있는 방법이 없을 까?



① 실행하고자 하는 악성코드가 윈도우용이든, 리눅스용이든 상관하고 싶지 않다.

② 악성코드가 특정 조건에만 동작하는 것도 무시하고 싶다.
(1월 1일에 파괴 동작하는 악성코드, 포르투갈 윈도우에서만 동작하는 악성코드 등)

③ 지금은 동작하지 않는 C&C라도 이 악성코드 분석은 계속 하길 원해…

④ 그냥 다 분석하고 싶어~~~~~~~~

에뮬레이터

기존 에뮬레이터… **운영체제를 에뮬레이션 할 수 없다.**

WINE은 리눅스에서 윈도우용 프로그램을 에뮬레이션한다.

즉, 윈도우 API를 적절하게 리눅스 API에 맵핑하면 된다.

아하 !! 그럼…
실행 파일 로더와
API를 적절하게 맵핑하면 되겠군.

세상은 넓고 나와 같은 생각을 하는 사람은 많다.

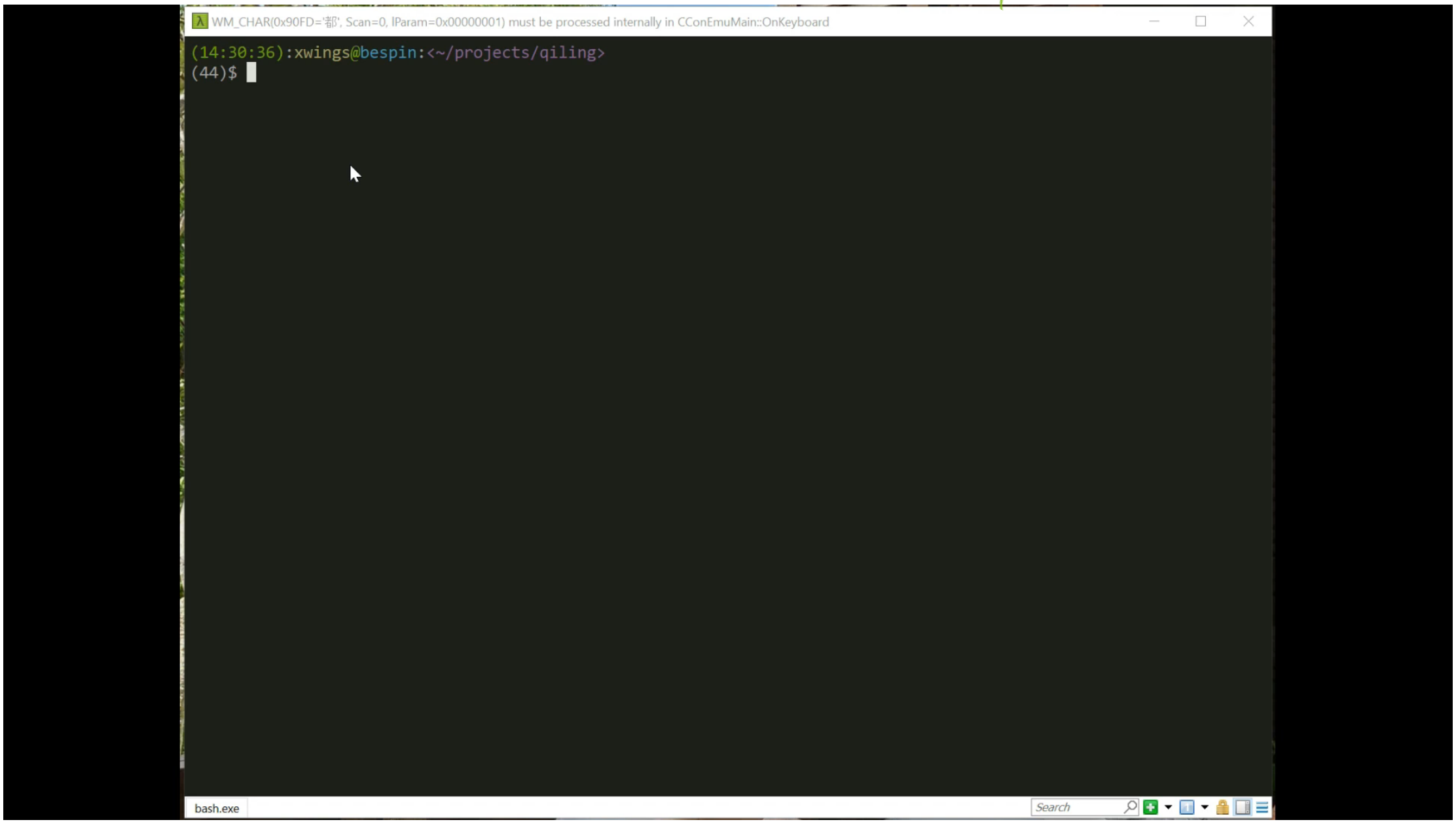
이미 이런 프로젝트가 존재합니다.

Qiling

Advanced Binary Emulation framework

Qiling
Advanced Binary Emulation framework

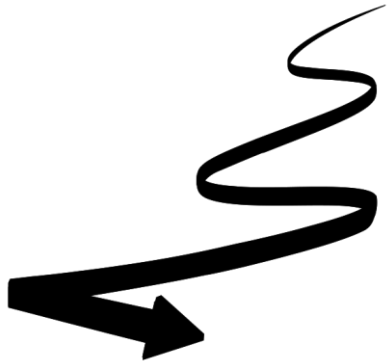
Wannacry Kill Switch 찾기



Qiling
Advanced Binary Emulation framework

앗 !!! 모든 윈도우 API를 에뮬레이션 못하네?

코딩 되지 않은 윈도우 API는 실행하지 못함



즉, 악성코드 실행을 위해 직접 필요한 API를 모두 코딩 해야 함

좋았어 !!! 내가 새로운걸 만들어 주겠어 !!

에뮬레이터

를 통해
다양한 운영체제 환경을 에뮬레이션 가능성 확인

API의 리턴 값은 하드 코딩된 값 만을 리턴

```
dllname = 'kernel32_dll'

# BOOL GetModuleInformation(
#   HANDLE       hProcess,
#   HMODULE      hModule,
#   LPMODULEINFO lpmodinfo,
#   DWORD        cb
# );
@winsdkapi(cc=STDCALL, dllname=dllname)
def hook_K32GetModuleInformation(self, address, params):
    # TODO
    return 0
```

특정일에 동작하는 악성코드의 경우,
날짜 API를 호출하여 얻은 결과를 통해 **악성행위**를 할지를 결정

하드 코딩된 API 리턴 값은 의미 없음

에뮬레이터

운영체제를 에뮬레이션 하고…

악성코드가 원하는 API 리턴 값을 잘 전달하며

최소한의 API 코딩으로 악성코드 행위를 추적할 수 있다면…

샌드박스로부터 **자유**를 외칠 수 있다.

에뮬레이터

분석 대상 프로그램 구동은 단순하게…

```python
# Author: Kei Choi (hanul93@gmail.com)
# NEmu 예제 프로그램

from nemu.nemu import NEmu

if __name__ == '__main__':
    ne = NEmu(['sample/abexcm1.exe'], verbose=True)  # NEmu 준비
    ne.run()  # 파일 실행
```

플러그인 방식으로 대상 파일 포맷 분석…

대상 파일에 맞는 운영체제 초기화

운영체제 메모리에 파일 로딩 후 실행

```python
    def run(self):  # 에뮬레이터를 실행
        # 파일 분석 (플러그인 방식으로)
        filename = self._run_args[0]

        plugins = [
            nefile_pe.NEmuFile(verbose=self._verbose),
        ]

        for plugin in plugins:
            if plugin.format(filename):
                break
        else:
            plugin = None

        # TODO 파일 포맷에 맞는 OS 부팅(?)
        if not plugin:
            return False
        else:
            plugin.os_init()

        # TODO 파일 실행
        plugin.run()
```

에뮬레이터

```python
    def run(self):  # 에뮬레이터를 실행
        # 파일 분석 (플러그인 방식으로)
        filename = self._run_args[0]

        plugins = [
            nefile_pe.NEmuFile(verbose=self._verbose),
        ]

        for plugin in plugins:
            if plugin.format(filename):
                break
        else:
            plugin = None

        # TODO 파일 포맷에 맞는 OS 부팅(?)
        if not plugin:
            return False
        else:
            plugin.os_init()

        # TODO 파일 실행
        plugin.run()
```

운영체제 초기화

```
C:\WINDOWS\system32\cmd.exe
(venv) D:\Project\NEmu>python example.py
================================================================================

                                    ..:rsr,
                                  :2;,;r2A@@5
                                 @2::s5A#@@@  @r
                                sd::riXA#@@ :@@@Gir;;AS9
                                Bs::sS3A#@2 @@#AhXirs$#;
                               iHr:r5d#@@@ .@#95sr;;rie
                               i,      ,@3 @@A2sr;;;r#5
                                :..:rll:    @@A5sr::r3@
                               @Hr;i2&@@@@     :rr;;;;;
                              S@r.:i2&@@@   @s       r
                              @2::ri2A@@# B@G2ir:...5i
                             :@r,r3X&#@@  @G5sr:..,:A
                            .@Ar;;rSB@@# H#2sr;,..,is
                            .         & ,@ASs;:..,:B
                                        ;rr;:,..,:.     TM


                          Windows(32bit) is Booting
                        Copyright 2015-2021 Nurilab Inc.

================================================================================
[+] Loding FileName: sample/abexcm1.exe
[+] File Image Base: 0x00400000
[+] File Image Size: 0x00006000
================================================================================
[+] 0x70000000, kernel32.dll:GetDriveTypeA
[+] 0x70000020, kernel32.dll:ExitProcess
[+] 0x70000040, user32.dll:MessageBoxA
================================================================================
00401000: 6a 00                         push    0
00401002: 68 00204000                   push    0x402000                        ; "abex' 1st crackme"
00401007: 68 12204000                   push    0x402012                        ; "Make me think your HD is a CD-Rom."
0040100c: 6a 00                         push    0
0040100e: e8 4e000000                   call    0x401061

; ============== S U B R O U T I N E =======================================
00401061: ff25 5c304000                 jmp     dword ptr [0x40305c]            ; user32.dll:MessageBoxA
; ---------------------------------------------------------------------------

00401013: 68 94204000                   push    0x402094                        ; "c:\"
00401018: e8 38000000                   call    0x401055

; ============== S U B R O U T I N E =======================================
00401055: ff25 50304000                 jmp     dword ptr [0x403050]            ; kernel32.dll:GetDriveTypeA
; ---------------------------------------------------------------------------

0040101d: 46                            inc     esi
0040101e: 48                            dec     eax
0040101f: eb 00                         jmp     0x401021
```

파일 가상 메모리에 로딩

윈도우 API를 가상 커널 메모리에 생성

프로그램 실행

Capstone 사용하여 에뮬레이션

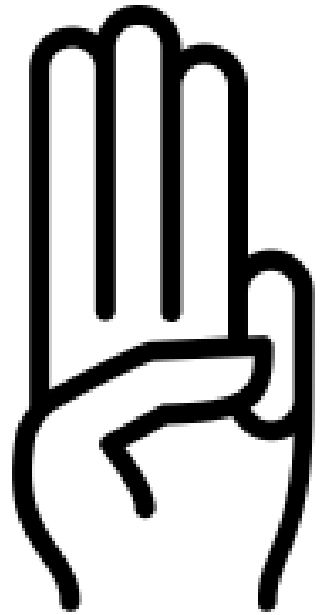
에뮬레이터

~~운영체제를 에뮬레이션 하고…~~ OK

악성코드가 원하는 API 리턴 값을 잘 전달하며

최소한의 API 코딩으로 악성코드 행위를 추적할 수 있다면…

샌드박스로부터 **자유**를 외칠 수 있다.



에뮬레이터 악성코드가 원하는 API 리턴 값

특정일에 동작하는 악성코드의 경우,
날짜 API를 호출하여 얻은 결과를 통해 **악성행위**를 할지를 결정

```
; =============== S U B R O U T I N E =======================================
00401055: ff25 50304000                   jmp     dword ptr [0x403050]        ; kernel32.dll:GetDriveTypeA
; ---------------------------------------------------------------------------

0040101d: 46                              inc     esi
0040101e: 48                              dec     eax
0040101f: eb 00                           jmp     0x401021
00401021: 46                              inc     esi
00401022: 46                              inc     esi
00401023: 48                              dec     eax
00401024: 3bc6                            cmp     eax, esi
00401026: 74 15                           je      0x40103d
00401028: 6a 00                           push    0
0040102a: 68 35204000                     push    0x402035                    ; "Error"
0040102f: 68 3b204000                     push    0x40203b                    ; "Nah... This is not a CD-ROM Drive!"
00401034: 6a 00                           push    0
00401036: e8 26000000                     call    0x401061
```

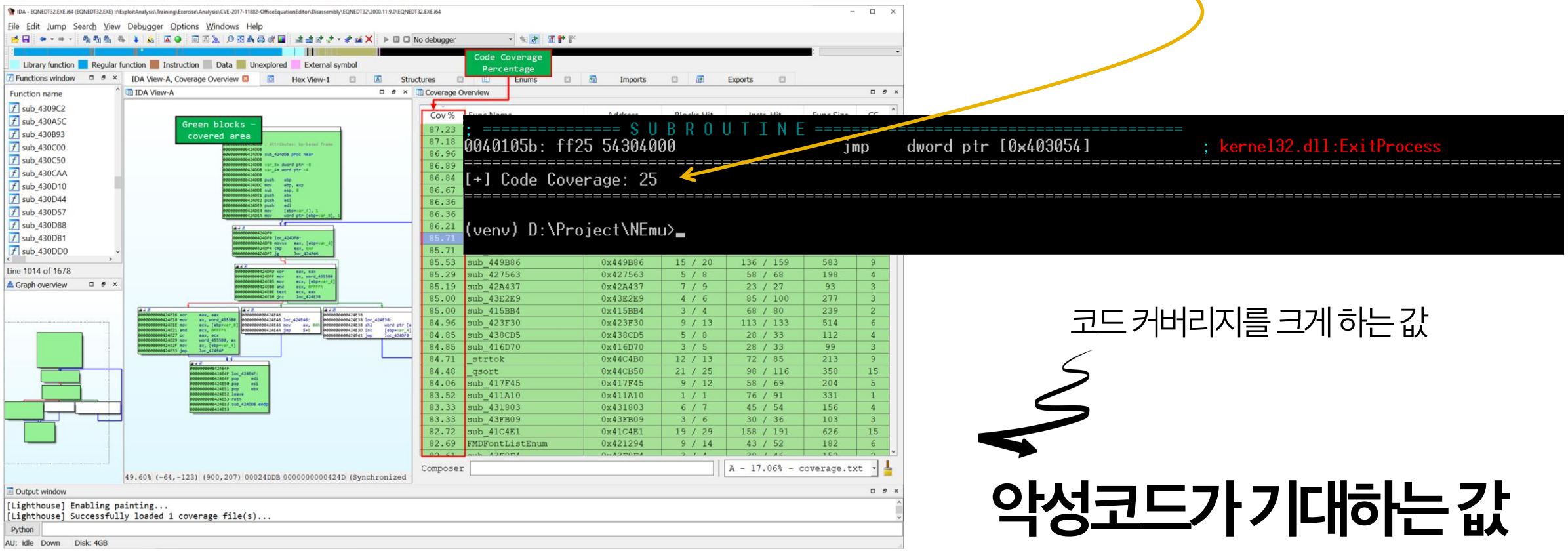
윈도우 API 호출 이후

cmp 중 eax와 비교 되는 값…

윈도우 API 기대 값

에뮬레이터 윈도우 API 기대 값에 따라 분기 발생

악성코드가 원하는 기대 값인가 아닌가는 **코드커버리지**로 측정



코드 커버리지를 크게 하는 값

악성코드가 기대하는 값

에뮬레이터

나는 크리스마스에...

나는 1월 1일에 하드디스크 파괴

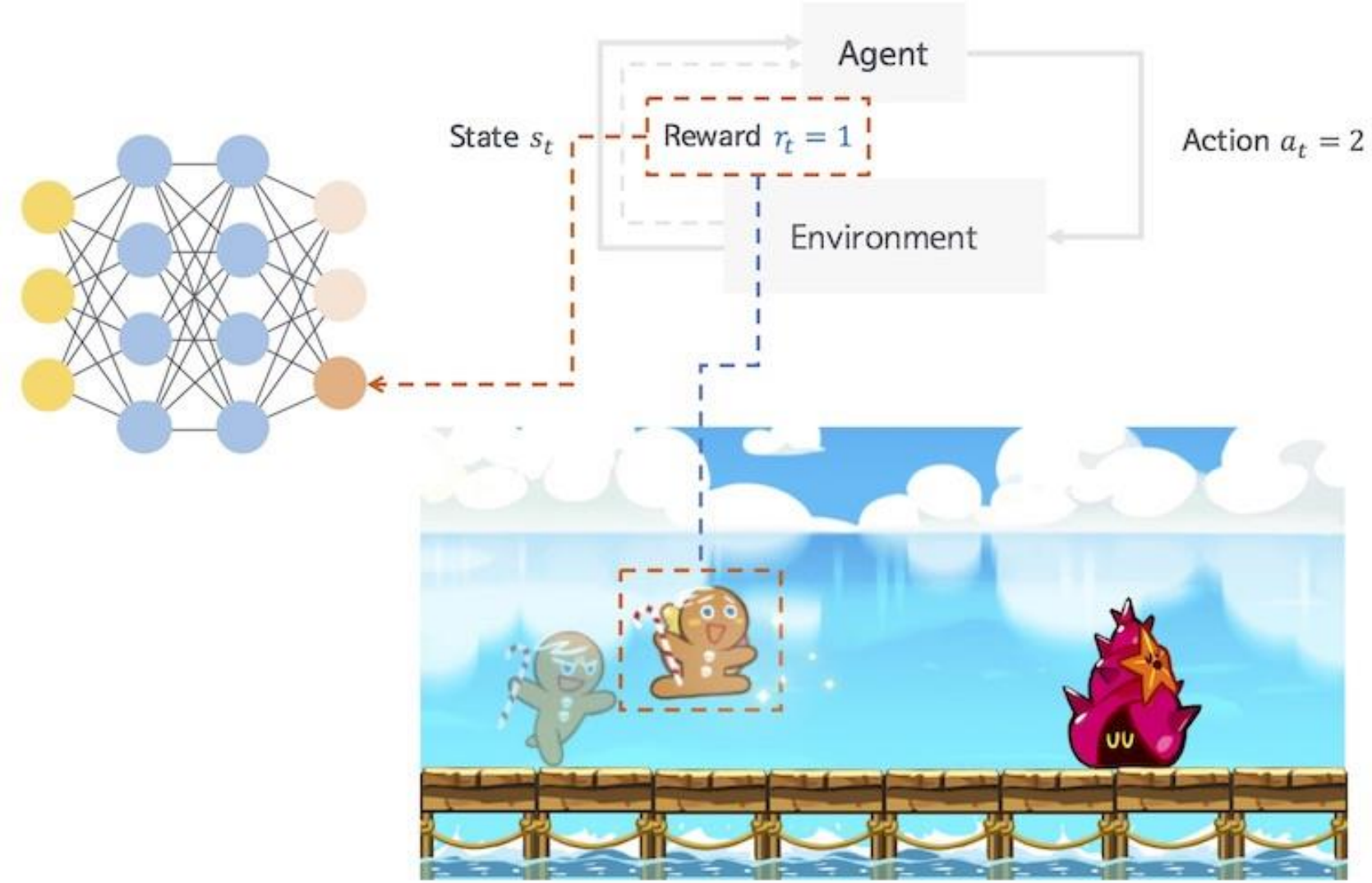
악성코드마다 기대 값이 다를 수 있음

지금까지 구동한 악성코드의 모든 API의 기대 값을 **데이터베이스화**

어떤 상황에서 **어떤 값**으로 **대응**하여 **코드커버리지**를 **크게**하는…

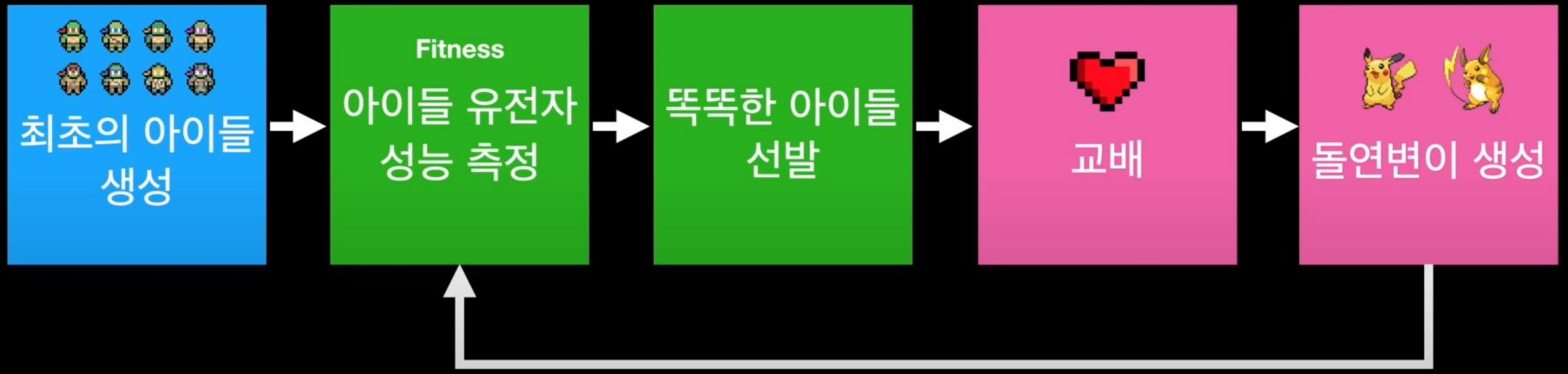
어떤 상황에서 어떤 값으로 대응하여 코드커버리지를 크게하는…

state Action Reward



에뮬레이터

Genetic Algorithm



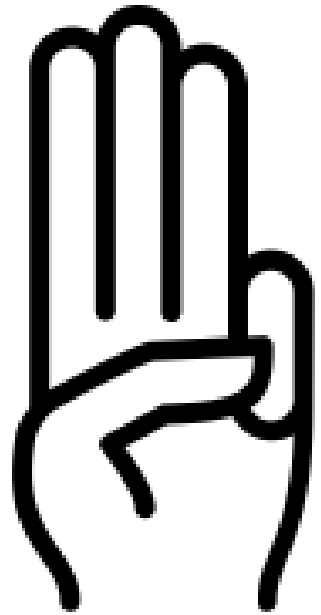
에뮬레이터

~~운영체제를 에뮬레이션 하고…~~ OK

~~악성코드가 원하는 API 리턴 값을 잘 전달하며~~ OK

최소한의 API 코딩으로 악성코드 행위를 추적할 수 있다면…

샌드박스로부터 **자유**를 외칠 수 있다.



에뮬레이터 최소한의 API 코딩으로 악성코드 행위를 추적할 수 있다!

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command And Control	Exfiltration	Impact
11 items	34 items	62 items	32 items	69 items	21 items	23 items	18 items	13 items	22 items	9 items	16 items
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Account Access Removal
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	Binary Padding	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Destruction
External Remote Services	Command-Line Interface	Account Manipulation	AppCert DLLs	BITS Jobs	Brute Force	Browser Bookmark Discovery	Component Object Model and Distributed COM	Clipboard Data	Connection Proxy	Data Encrypted	Data Encrypted for Impact
Hardware Additions	Compiled HTML File	AppCert DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	Domain Trust Discovery	Exploitation of Remote Services	Data from Information Repositories	Custom Command and Control Protocol	Data Transfer Size Limits	Defacement
Replication Through Removable Media	Component Object Model and Distributed COM	AppInit DLLs	Application Shimming	Clear Command History	Credentials from Web Browsers	File and Directory Discovery	Internal Spearphishing	Data from Local System	Custom Cryptographic Protocol	Exfiltration Over Alternative Protocol	Disk Content Wipe
Spearphishing Attachment	Control Panel Items	Application Shimming	Bypass User Account Control	CMSTP	Credentials in Files	Network Service Scanning	Logon Scripts	Data from Network Shared Drive	Data Encoding	Exfiltration Over Command and Control Channel	Disk Structure Wipe
Spearphishing Link	Dynamic Data Exchange	Authentication Package	DLL Search Order Hijacking	Code Signing	Credentials in Registry	Network Share Discovery	Pass the Hash	Data from Removable Media	Data Obfuscation	Exfiltration Over Other Network Medium	Endpoint Denial of Service
Spearphishing via Service	Execution through API	BITS Jobs	Dylib Hijacking	Compile After Delivery	Exploitation for Credential Access	Network Sniffing	Pass the Ticket	Data Staged	Domain Fronting	Exfiltration Over Physical Medium	Firmware Corruption
Supply Chain Compromise	Execution through Module Load	Bootkit	Elevated Execution with Prompt	Compiled HTML File	Forced Authentication	Password Policy Discovery	Remote Desktop Protocol	Email Collection	Domain Generation Algorithms	Scheduled Transfer	Inhibit System Recovery
Trusted Relationship	Exploitation for Client Execution	Browser Extensions	Emond	Component Firmware	Hooking	Peripheral Device Discovery	Remote File Copy	Input Capture	Fallback Channels		Network Denial of Service
Valid Accounts	Graphical User Interface	Change Default File Association	Exploitation for Privilege Escalation	Component Object Model Hijacking	Input Capture	Permission Groups Discovery	Remote Services	Man in the Browser	Multi-hop Proxy		Resource Hijacking
	InstallUtil	Component Firmware	Extra Window Memory Injection	Connection Proxy	Input Prompt	Process Discovery	Replication Through Removable Media	Screen Capture	Multi-Stage Channels		Runtime Data Manipulation
	Launchctl	Create Account	File System Permissions Weakness	Control Panel Items	Kerberoasting	Query Registry	Shared Webroot	Video Capture	Multiband Communication		Service Stop
	Local Job Scheduling	DLL Search Order Hijacking	Hooking	DCShadow	Keychain	Remote System Discovery	SSH Hijacking		Multilayer Encryption		Stored Data Manipulation
	LSASS Driver	Dylib Hijacking	Image File Execution Options Injection	Deobfuscate/Decode Files or Information	LLMNR/NBT-NS Poisoning and Relay	Security Software Discovery	Taint Shared Content		Port Knocking		System Shutdown/Reboot
	Mshta	Emond	Launch Daemon	Disabling Security Tools	Network Sniffing	Software Discovery	Third-party Software		Remote Access Tools		Transmitted Data Manipulation
	PowerShell	External Remote Services	New Service	DLL Search Order Hijacking	Password Filter DLL	System Information Discovery	Windows Admin Shares		Remote File Copy		
	Regsvcs/Regasm	File System Permissions Weakness	Parent PID Spoofing	DLL Side-Loading	Private Keys	System Network Configuration Discovery	Windows Remote Management		Standard Application Layer Protocol		
	Regsvr32	Hidden Files and Directories	Path Interception	Execution Guardrails	Securityd Memory	System Network Connections Discovery			Standard Cryptographic Protocol		
	Rundll32	Hooking	Plist Modification	Exploitation for Defense Evasion	Steal Web Session Cookie	System Owner/User Discovery			Standard Non-Application Layer Protocol		
	Scheduled Task	Hypervisor	Port Monitors	Extra Window Memory Injection	Two-Factor Authentication Interception	System Service Discovery			Uncommonly Used Port		
	Scripting	Image File Execution Options Injection	PowerShell Profile	File and Directory Permissions Modification		System Time Discovery			Web Service		
	Service Execution	Kernel Modules and Extensions	Process Injection	File Deletion		Virtualization/Sandbox Evasion					
	Signed Binary Proxy Execution	Launch Agent	Scheduled Task	File System Logical Offsets							
	Signed Script Proxy Execution	Launch Daemon	Service Registry Permissions Weakness	Gatekeeper Bypass							
	Source	Launchctl	Setuid and Setgid	Group Policy Modification							
	Space after Filename	LC_LOAD_DYLIB Addition	SID-History Injection	Hidden Files and Directories							
	Third-party Software	Local Job Scheduling	Startup Items	Hidden Users							
	Trap	Login Item	Sudo	Hidden Window							
	Trusted Developer Utilities	Logon Scripts	Sudo Caching	HISTCONTROL							
	User Execution			Image File Execution Options Injection							
	Windows Management			Indicator Blocking							

MITRE ATT&CK™

더 궁금한 점이 있으시면 언제든지 연락주세요

감사합니다

최원혁
whchoi@nurilab.com