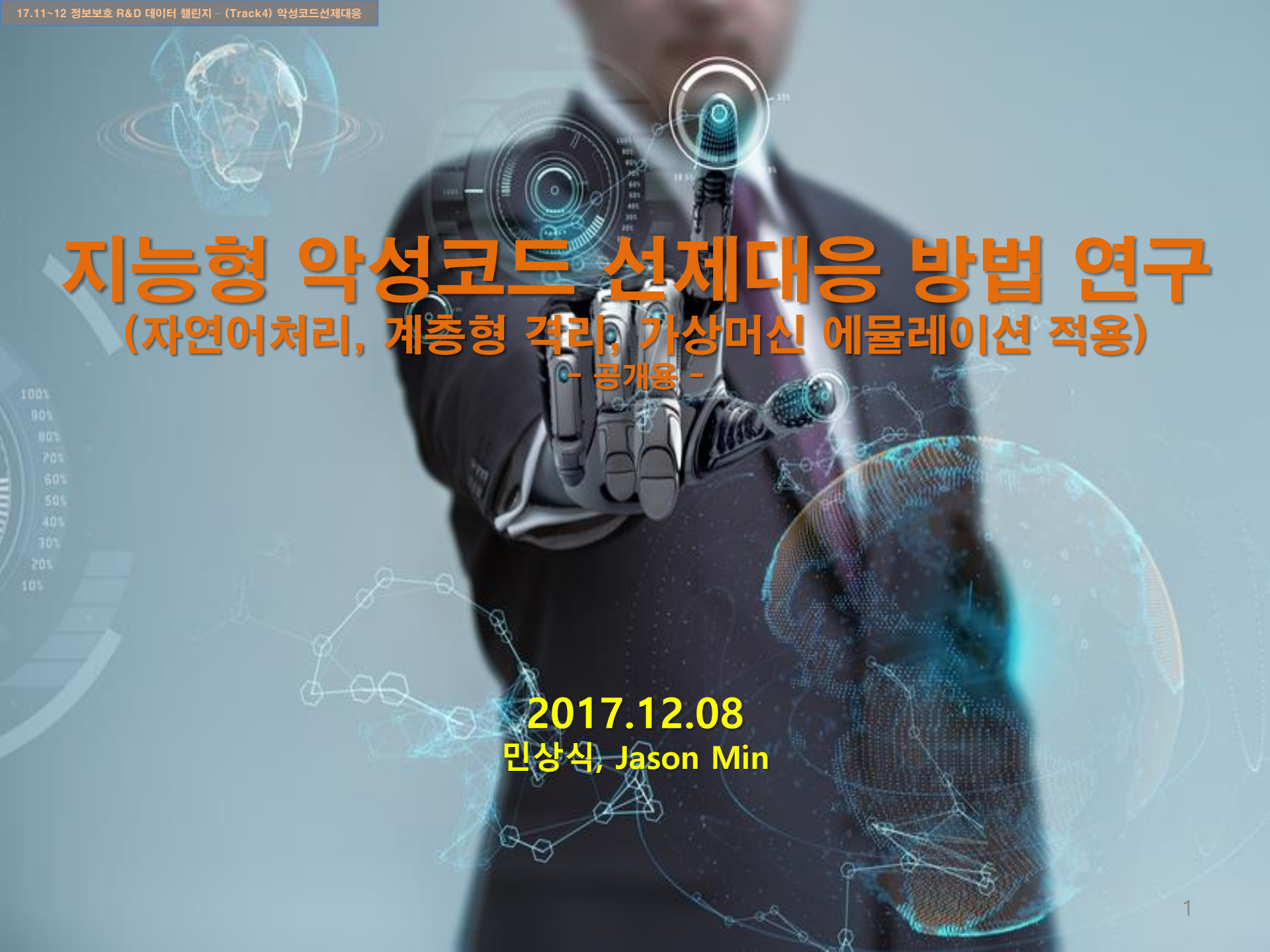


지능형 악성코드 선제대응 방법 연구

(자연어처리, 계층형 격리, 가상머신 에뮬레이션 적용)

– 공개용 –

2017.12.08

민상식, Jason Min

A. 아이디어 구상(개요)

(기준) 300개 지능형 악성코드 분석

↓

(피해보정) 현 지능형 악성코드 통계 조사를 통한 대표성 보정

↓

(성능보정) 악성행위, 피해영향도, 대응실효성 등 종합분석

↓

알고리즘 설계

↓

알고리즘 구현
(모든 코드는 구현하여 검증, Part3 악성코드와 비교 등)

최초설계

제공된 악성코드를 통한 행위 기본 분석 (자

1단계 : 정적 – PE 특징 추출
2단계 A : 정적 – 자동패킹해제 후 디컴파일(역어셈블) &
or IDA, HexRay 최신버전 확인 및 왁보
2단계 B : 동적 - 가상환경에서 동적 데이터 추출(샌드박스 분석툴
=> 통계분석(방어필요포인트 파악)
기타
- 엔트로피 분석 등
- 가상환경 확인여부
- 인터넷 연결여부
- 인터넷 연결시 로케이션 등 이용자원 확인여부

제공된 악성코드를 통한 행위 기본 분석, Paper, 전문가 분석자료(BlackHat 등) 등
악성코드 Parent Process(정상) ID/Name의 통계적 분석(Fileless, Exploit)을 통한
악성행위 여부 판단
(피해 : 악성코드 동작환경 정의)
보호대상 :
무결성 : 일반파일, OS(레지스트리 등), 메모리,
*** 서비스 형태 등록**
기밀성 : 네트워크(유출) 등

정상/악성 분류가 어려운 지능형 악성파일(추정)은 기본적으로 예비격리 시킨 상태에서 수행

- 수행 시 가상환경 인것 처럼 시뮬레이션이 가능한 **예비격리 환경**을 제공하는 등

*** 지능형 악성코드의 Anti-탐지 기능을 역 이용**

B. 아이디어 구상(상세 정의)

2차설계

I. **지능형 악성코드 검토**
  A. 기본분석(제시된 300개 악성코드)
    a. 정적분석 – PE 추출 후 검토
      ① **PEStudioX Command Mode**
    b. 정적분석 – 역분석 (BASIC)
    c. 정적분석 – 역분석 (난독화, 패킹 해제)
    d. 동적분석 – 가상머신
    e. 동적분석 – 리얼머신 (가상머신과 비교를 위해 리얼머신 모니터링 방법론 개발 필요)
  B. 기본분석 (최근 지능형악성코드, 기능, 가상머신동적분석 회피기술, 피해 등)
    a. 정보삭제(랜섬웨어 등)
    b. 시스템(운영체제 및 자료 삭제) 파괴
    c. 정보유출(비트코인, 중요정보 등) 악성코드
    d. 자원유출(비트코인 마이닝)

  C. 상세분석
    a. 확보된 정보의 분석(통계적 기법 및 머신러닝 등) – NLP 분석 기본정리

II. **지능형 악성코드 선제대응 전략 수립**
  A. 목적 : 지능형악성코드 선제대응
  B. 선제대응 검토결과
    a. (기존) 자동분석을 위한 **패킹연구, 가상머신 동적분석** 등의 기존 방법론은 일반 악성코드 방어에는 적합할 수 있으나
    b. (한계) 최신 방어수단을 확인 후 회피기능이 적용된 **지능형 악성코드의 방어에는 현실적인 어려움**이 있음
    c. (결론) 악성행위 직전까지 판단 불가한 지능형 악성코드 선제대응을 위해 "질병관리센터 격리체계"를 수용한 방법론 적용
  C. 구현방향
    a. 유포지 (인터넷/이메일 서버 영역) – 본 챌린지의 방어영역은 아닌것으로 판단하여 제외
    b. 파일 다운로드/감염/수신 & 실행 (이용자 PC)
      ① **(서버) 정보 통합서버 구축**
      ② **(PC) PC(클라이언트) 측 방어 방안**
        - **기존 유전체분석방법(DNA)에서 실제 코드이해를 위한 자연어처리 (NLP)적용 제안**
        - **공중보건관리체계를 사이버 보안체계에 접목 - 질병관리센터(CDC) 격리체계 적용 – 격리레벨분석**
        - **리얽머신 모니터링 기술 설계 (커널 후킹 구현)**

III. **악성코드 분석 지연을 위한 지능형·고도화 기술(Offensive Research) 제안**
  A. 정적분석/동적분석이 불가하도록 엔트로피 임의 증가 기술 제안

1. 지능형 악성코드 분석

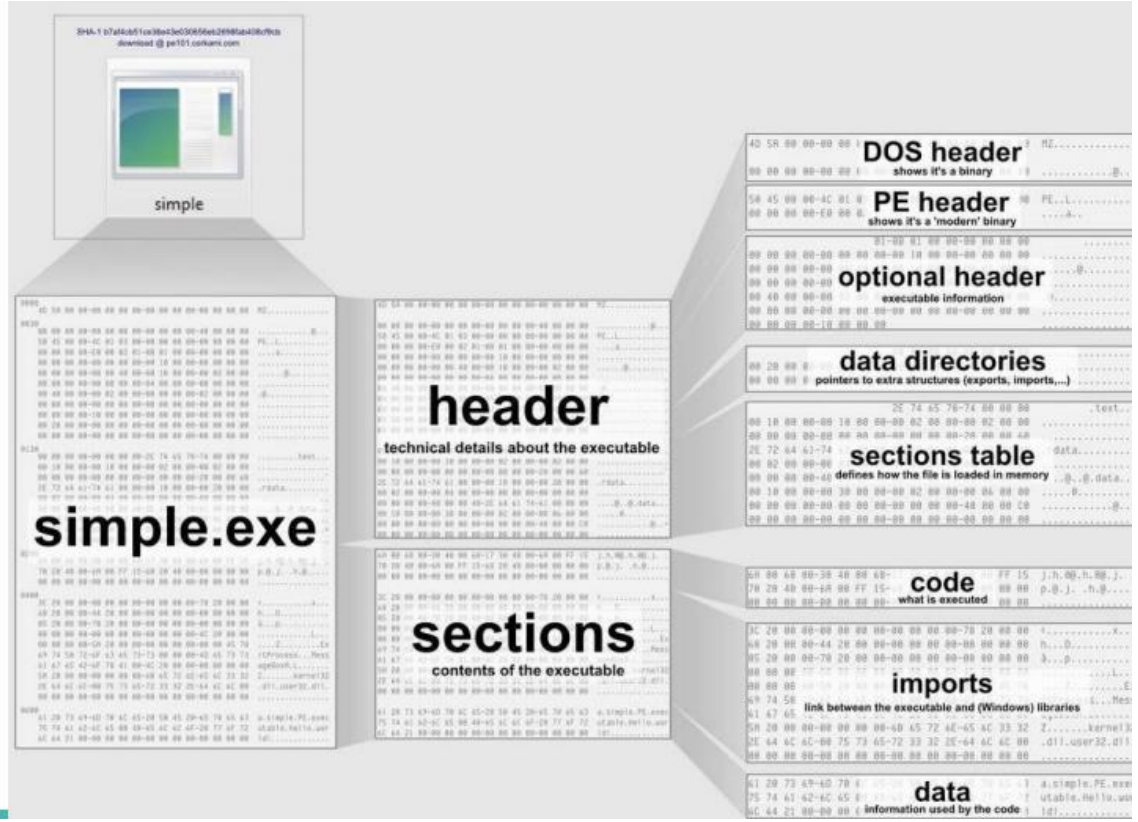
(300->295->200)

MAL

Machine Learning for Malware Analysis

Andrew Davis
Data Scientist
CYLANCE

Portable Executable (PE) Format



MAL

Feature Engineering - Static Analysis

- What kinds of features can we extract for PE files?
- Objective: extract features from the EXE without executing anything
- PE-Specific features
  - Information about the structure of the PE file
- Strings
  - Print off all human-readable strings from the binary
- ★ Entropy features — Unpacking
  - Extract information about the predictability of byte sequences
  - Compressed/encrypted data is high entropy
- ★ Disassembly features — Disassembly
  - Get an idea of what kind of code the sample will execute

MAL

AntiVM - Dynamic Analysis

FireEye VM 자동분석 Env

휴먼 인터랙션

SetWinodwsHookExA

파일 기반의 샌드박스는 인간 사용자(Human User) 없이 물리적 시스템을 에뮬레이션하여 동작합니다. 공격자들은 이러한 중요한 차이점을 악용하여, 마우스 클릭, 대화상자에 대한 지능적인 응답 등과 같은 인간 사용자의 개입을 탐지할 때까지 잠복해 있는 악성코드를 작성합니다. 이 섹션에서는 이러한 확인사항들을 더 상세하게 설명합니다.

마우스 클릭

2012년 12월에 분석된 트로이 목마 바이러스의 일종인 UpClicker는 마우스 클릭을 통해서 사람의 활동을 탐지하는, 최초의 악성코드 샘플 중 하나입니다.[2] (이와 유사하지만 더 간단한 기법이 몇 달 더 일찍 출현했습니다.[3]) UpClicker는 샌드박스를 속이기 위해서 마우스 왼쪽 버튼을 클릭한 것을 탐지한 후에만 악성 CnC 서버를 통해서 통신을 수립합니다. UpClicker는 지능형 지속적 위협(APT) 공격과 관련된 원격 접속 툴(RAT)인 Poison Ivy를 둘러싸고 있는 래퍼입니다.[4]

그림 1은 UpCliker 코드 정보가 매개변수로 0Eh를 사용하여 *SetWinodwsHookExA*를 호출하는 것을 보여줍니다. 이러한 설정은 Windows에서 *WH_MOUSE_LL* 연결절차(hook procedure)를 사용하여 낮은 레벨의 마우스 입력을 감시하게 합니다.[5]

```
add     esp, 8
push    0               ; dwThreadId
push    0               ; lpModuleName
call    ds:GetModuleHandleA
push    eax             ; hmod
push    offset fn       ; lpfn
push    0Eh             ; idHook ; WH_MOUSE_LL
call    ds:SetWindowsHookExA
mov     esi, ds:GetMessageA
push    0               ; wMsgFilterMax
```

UpClicker. BaneChant

"Trojan.APT.BaneChant: In-Memory Trojan That Observes for Multiple Mouse Clicks

설정

샌드박스가 보호 대상인 물리적 컴퓨터를 최대한 모방하려고 시도하는 이러한 가상 환경은 정의된 매개변수 세트로 설정됩니다. 사이버 공격자는 이러한 설정을 확인하여 회피하는 방법을 찾아내었습니다.

잠복기(Sleep Call)

파일 기반의 샌드박스는 수많은 샘플을 조사하기 때문에, 보통 파일들을 몇 분 동안 모니터한 후에 의심스러운 행동이 나타나지 않으면 다음 파일로 이동합니다.

이러한 절차는 악성코드 작성자에게 샌드박스를 통과할 때까지 기다리는 간단한 회피 전략을 제공합니다. 악성코드는 휴면 호출을 연장함으로써 모니터 과정이 끝날 때까지 의심스러운 행동을 자제합니다.

2013년 2월에 발견된[7] Trojan Nap은 이러한 접근방법을 사용합니다. 이 트로이 목마 바이러스는 Microsoft와 Kaspersky가 2011년에 해체한 Kelihos Botnet과 연계되어 있습니다.[8]

Trojan Nap

0x0927C0의 타임아웃 매개변수 값(600,000밀리초, 즉 10분)

이 보고서는 다음의 카테고리에 대한 샌드박스 회피 기법을 상세하게 설명합니다.

- **휴먼 인터랙션**—마우스 클릭 및 대화상자
- **설정 관련**—휴면 호출, 시간 트리거 및 프로세스 은닉
- **환경 관련**—버전, 내장된 iframe 및 DLL 로더
- **VM웨어 관련**—시스템-서비스 리스트, 고유한 파일 및 VMX 포트

지능형 악성코드 – 샌드박스 회피

https://www.fireeye.kr/content/dam/fireeye-www/regional/ko_KR/current-threats/pdfs/fireeye-hot-kni

MAL

특정시간 동작(Time Trigger: 시한 폭탄)

휴면 API 호출은 때때로 타임 트리거와 함께 사용하여 정해진 날짜와 시간 후에만 악성코드를 실행하므로, 그 시간 전에 파일을 모니터하는 샌드박스는 비정상적인 행동을 탐지하지 못합니다.

이에 적합한 사례는 2013년 3월에 한국에서 발생한 대규모의 데이터 파괴 공격에 사용된 Hastati라는 이름의 트로이 목마 바이러스입니다.[9] Hastati는 Windows의 SystemTime 을 참조하여 해당 지역의 현재 날짜와 시간을 결정하는 *GetLocalTime()* API 방법을 사용합니다. 가상 머신 샌드박스가 그 특정한 시간에 파일을 모니터하지 않는 경우, 악성코드는 탐지되지 않고 감시망을 벗어나 잠입할 수 있습니다.

Hastati 악성코드

프로세스 숨기기

파일 분석 위주의 샌드박스는 운영체제에서 발생하는 모든 프로세스를 모니터하여 의심스러운 악성 활동을 탐지합니다. 많은 샌드박스는 Microsoft가 제공하는 *PsSetCreateProcessNotifyRoutine*이라고 하는 커널 루틴을 사용하여 이러한 활동을 탐지하도록 설정됩니다. 이 루틴을 사용하면 Windows 프로세스가 생성 또는 종료될 때 하드웨어 드라이버가 호출할 소프트웨어 루틴의 리스트를 작성 또는 변경할 수 있습니다. 파일 기반의 샌드박스는 이 정보를 사용하여 시스템 활동을 추적하고 중요한 정보를 보호할 수 있습니다.

PsSetCreateProcessNotifyRoutine

환경

이론적으로, 샌드박스에서 실행되는 악성코드는 물리적 컴퓨터에서 동일한 방법으로 실행해야 합니다. 실제로, 대부분의 샌드박스는 탐지형으로서 공격자들은 이러한 가상 환경을 확인하는 기능을 악성코드에 포함시킬 수 있습니다. 이 섹션에서는 이러한 몇 가지 확인사항에 대해 더 상세하게 설명합니다.

버전 확인

많은 악성 파일은 어플리케이션이나 운영체제의 특정한 버전에서만 실행되도록 설정됩니다. 이처럼 공격자가 자체적으로 부과한 제한은 반드시 샌드박스를 특별히 회피하기 위한 것만은 아니며, 예를 들어 많은 공격자들이 어플리케이션의 특정한 버전에만 있는 결함을 악용하려고 시도합니다.

그러나 결과는 종종 동일합니다. 모든 샌드박스에는 사전 정의된 설정이 있습니다. 특정한 설정에 운영체제와 어플리케이션의 특별한 조합이 없는 경우, 일부 악성코드는 실행되지 않고 탐지를 회피합니다.

MAL

GIF와 Flash 파일에 들어 있는 내장 iframe

악성코드는 종종 무해한 것처럼 보이는 파일을 사용하여 방어 시스템을 통과하고 악성 페이로드를 다운로드합니다. 통상적인 접근방법은 GIF 사진 또는 Acrobat Flash와 같은 다른 비실행 파일에 들어 있는 iframe HTML 엘리먼트를 숨기는 것입니다.

이러한 파일들은 독자적으로 실행되지 않으므로 샌드박스에서 의심스러운 행동을 드러내지 않습니다. 그 대신에, 이러한 파일들은 데이터를 숨기며, 이 데이터는 침해된 물리적 컴퓨터에서 그 데이터를 기다리고 있는 별도의 파일에 의해 언로크가 되고 실행됩니다.

DLL 로더 확인

보통, 동적 연결 라이브러리(DLL) 파일을 실행하는 프로세스에는 run32dll.exe를 사용하거나 DLL을 로드하는 것이 포함됩니다. 어떤 악성코드는 DLL을 실행할 특정한 로더를 필요로 하는 다른 프로세스를 사용합니다. 필요한 로더가 없는 경우에는 DLL이 실행되지 않고 샌드박스에 의해 탐지되지 않습니다.

그림 16은 로더의 해시를 계산하여 필요한 로더가 맞는지 알아내는 악성코드를 보여줍니다.

VM웨어 회피 기법

이 보고서에서 지금까지 요약 설명한 샌드박스 회피 기법은 특히 지능형 악성코드와 APT에 사용됩니다. 그러나 저희의 원격 측정 데이터에 의하면 몇 가지 오래된 회피 기법들도 악성코드 작성자에게 계속 유용하다는 것이 증명되고 있습니다.[11] 인기있는 가상 머신 툴인 VM웨어는 독특한 설정 때문에 특히 탐지하기 쉽습니다.

시스템-서비스 리스트

어떤 악성코드들은 VM웨어를 사용하여 생성한 샌드박스가 있는지 탐지하기 위해 vmicheatbeat, vmci, vmdebug, vmmouse, vmscis, VMTools, vmware, vmx86, vmhgfs, vmxnet와 같은 VM웨어에 고유한 서비스를 확인합니다.

Virus Bulletin. "Techniques for Evading Automated Analysis(자동화 분석을 회피하는 기법)." 2013년 2월.

지능형 악성코드 – 샌드박스 회피

https://www.fireeye.kr/content/dam/fireeye-www/regional/ko_KR/current-threats/pdfs/fireeye-hot-kni

MAL

독특한 파일(Unique File)

VM웨어를 사용하는 샌드박스에서 악성코드가 실행되고 있다는 다른 명백한 증거는 VM웨어에서만 존재하는 파일들을 확인하는것입니다. 그림 18은 GetFileAttributeA() 함수를 사용하여 VM웨어 마우스 드라이버를 확인하는 악성코드를 보여줍니다.

```
vmmouse.sys
```

VMX 통신 포트

다른 명백한 인디케이터는 VM웨어가 가상 머신과 통신하기 위해 사용하는 VMX 포트입니다. 이 포트가 있는 경우, 악성코드는 “죽은 척” 하고 있어 탐지를 회피합니다. 그림 19는 이 포트를 확인하는 악성코드를 보여줍니다.

AntiVM - Dynamic Analysis

Detecting Malware and Sandbox Evasion Techniques

Author: Dilshan Keragala, dkeragala@att.net

Advisor: Chris Walker, C|CISO, CISSP, GSEC, GCUX, GCWN, GWEB

Accepted: January 16, 2016

Qemu, CWSandbox, Anubis, VMWare Fusion, VMWare Workstation and Virtual Box, Zen

vmusrvc.exe, vboxtray.exe, vmtoolsd.exe, df5serv.exe, and vboxservice.exe

Once the malware detects a single core, it stops executing.

Unpacking malware before analysis
malware communicates with remote servers on a network such as a download and C&C servers
Malware Sandbox analysis has three important properties: observability, containment, and efficiency

Singh, Sudeep. (2015). Breaking the sandbox. Exploit-db.Com. Retrieved from https://www.exploit-db.com/docs/34591.pdf

Anti VM techniques

- **Process Names** vmusrvc.exe, vboxtray.exe, vmtoolsd.exe, df5serv.exe, vboxservice.exe and so on
- **Registry Artifacts** HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Disk\Enum
- **Module Names**
- **Backdoor Detection**
- **Long Opcode Instructions**
- **Number of Cores**
- **Data structures**
- **Device Information**
- **File System Artifacts**
- **Network Adapter MAC Address**
- **Sensitive Instructions**

```
#include <windows.h>
#include <stdio.h>
#include <TlHelp32.h>

/*
Author: Sudeep Singh
*/

int main(int argc, char **argv)
{
HANDLE psnap;
HMODULE hModule;
MODULEENTRY32 me;
me.dwSize = sizeof(MODULEENTRY32);

psnap = CreateToolhelp32Snapshot(TH32CS SNAPMODULE, 0);

if(!Module32First(psnap, &me))
{
    printf("There was an error in retrieving the module information\n");
    exit(0);
}

while(Module32Next(psnap, &me))
{
    if(strcmp(me.szModule, "kernel32.dll") != 0)
    {
        if(strcmp(me.szModule, "ntdll.dll") != 0)
        {
            hModule = GetModuleHandle(me.szModule);
            if(FreeLibrary(hModule) != 0)
            {
                printf("successfully unloaded injected module, %s\n",
me.szModule);
            }
        }
    }
}

return 0;
}
```

https://gist.github.com/c0d3inj3cT/c68a203c2c1224df55b3.

http://www.crestandiisp.com/wp-content/uploads/2

MAL

SandGrox: Detecting and bypassing sandboxes

PwC Threat and Vulnerability Management

April 2017

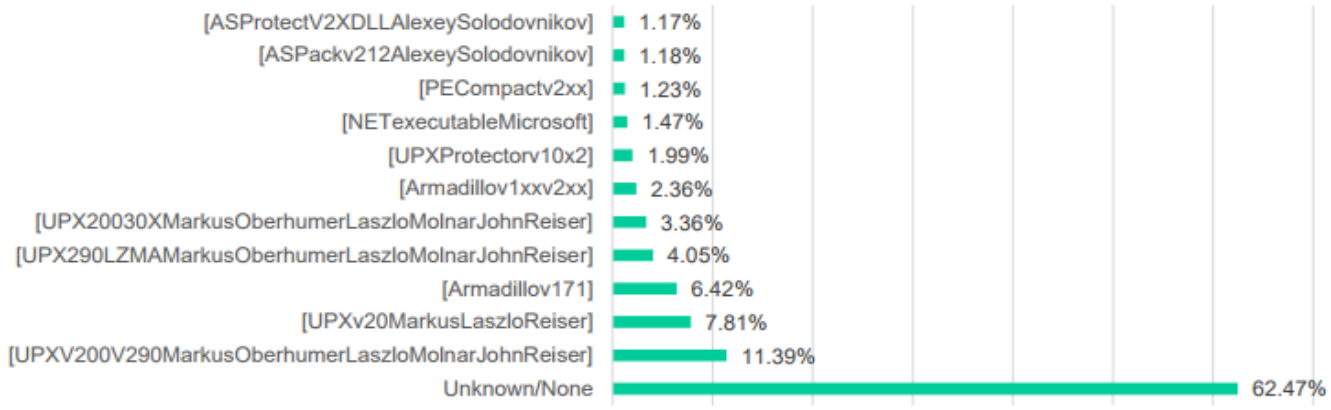
- Virtualisation (VMs, isolated, & micro-virtualisation)
- AV emulators
- Hardware-based
- Browser-based
- etc

- Pafish
- BlackHat '13: The Sandbox Roulette
- BlackHat '14: One Packer to Rule Them All
- BlackHat '16: AVLeak
- VMBuster
- InviZzzible
- EvilBunny (and lots of others!)
- ...etc

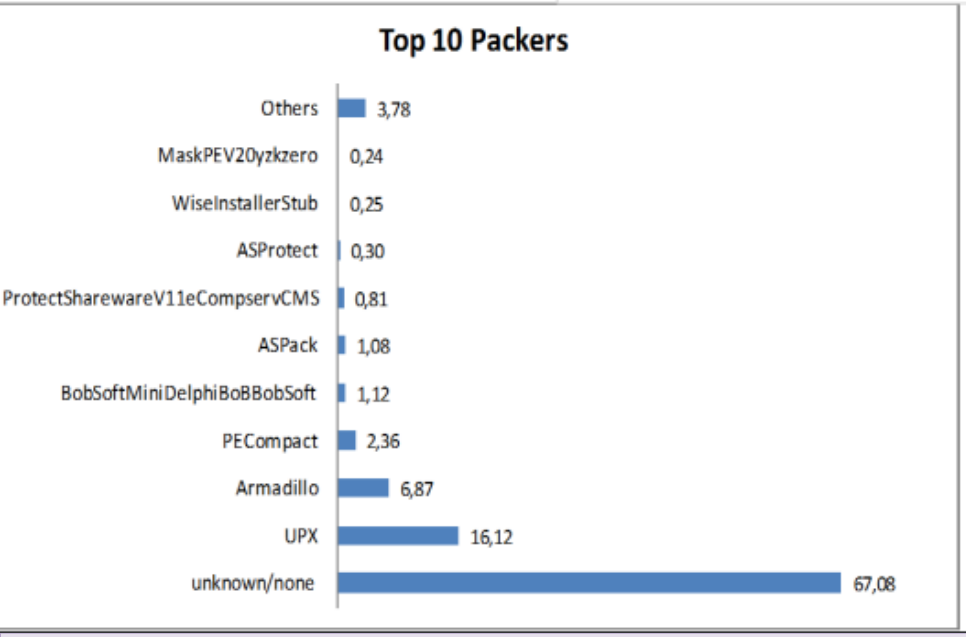
- 111 functions
  - *Sleep: 8*
  - *Memory: 5*
  - *Hardware: 14*
  - *Assembly: 33*
  - *Environment: 23*
  - *Filesystem: 12*
  - *Network: 10*
  - *Bypasses: 6*

MAL

Top Packers (updated)



2012 results

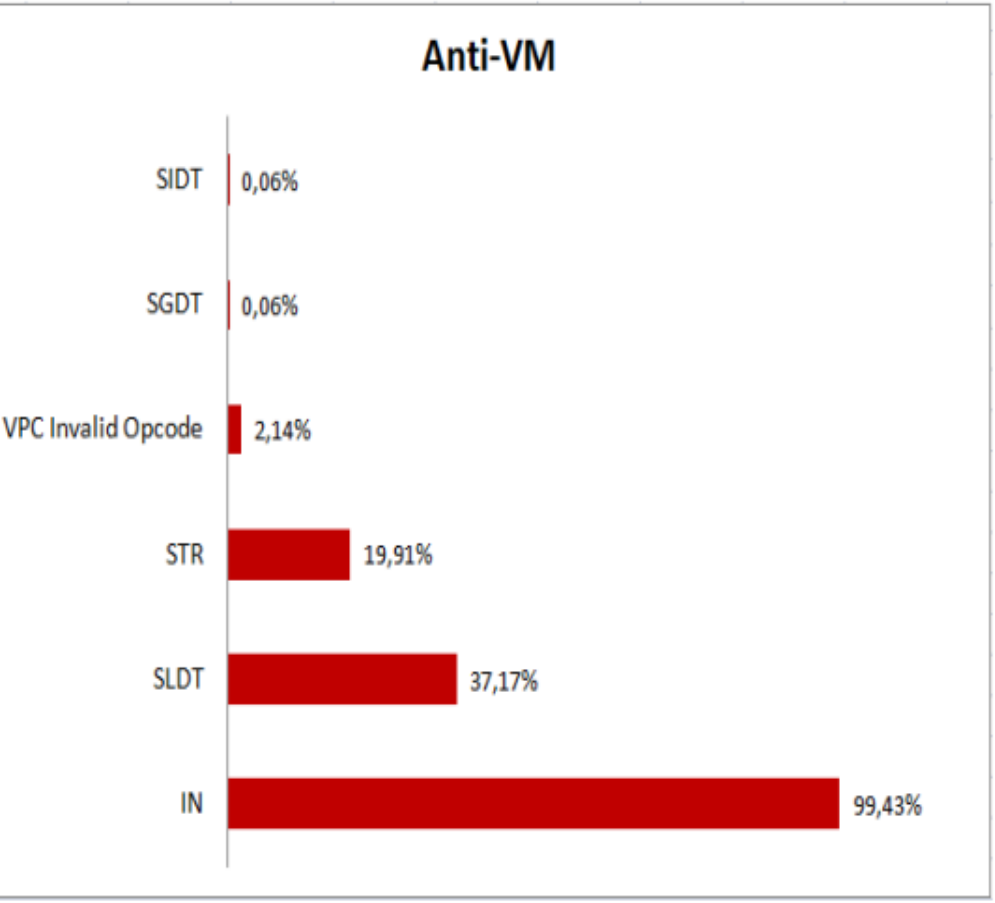


2014 results

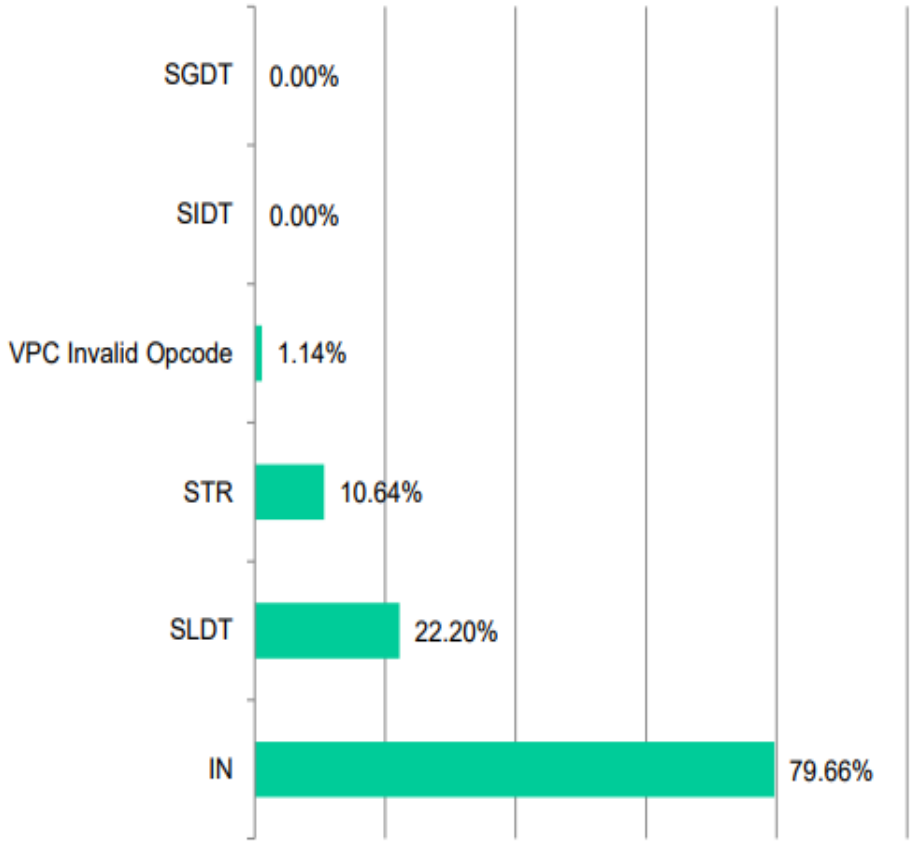
(검토결과) 언패킹처리 : 최소 40% 이상 악성코드 분석과 관련됨

MAL

Anti-VM (updated)



2012 results



2014 results

(검토결과) VM 자동분석기능으로 동적행위분석 어려움

2. 악성코드 분석 지연을 위한 지능형 고도화 기술 (제안)

악성코드 분석 지연을 위한 지능형·고도화 기술 (Offensive Research) 제안

1. **정적분석 방어** : String, Code 등 실제 실행되지만 결과에 영향을 주지 않는 Dummy 코드 추가, 다중패킹, **자체 VM내장**

2. **동적분석 방어** : **Layered integrity Check, Randomized Activation algorithm** 적용을 통한 역분석 방지, 자체 VM내장

3. **리버싱, VM방어** : 블록체인 마이닝과 같은 방식, 암호키를 찾는 알고리즘 적용
   - 악성코드 내 암호키 없음 – 정적분석 불가
   - C&C 서버 내 암호키 없음
   - **일정시간 후 실행가능성 100% 보장** (가상머신 우회 가능)

Time-lock puzzles and timed-release Crypto

Ronald L. Rivest*, Adi Shamir**, and David A. Wagner***

Revised March 10, 1996

4. C&C를 일반 유명게시판 or SNS로 지정하고, **한글로 인코딩된 암호키**와 Exploit Code를 주기적으로 확인하여 악성코드가 원하는 정보를 수신하는 경우 작동

"시간 잠금 퍼즐 및 시간 제한적 암호화" (96)

Time-lock puzzles and timed-release Crypto

Ronald L. Rivest*, Adi Shamir**, and David A. Wagner***

Revised March 10, 1996

*MIT Laboratory for Computer Science
545 Technology Square, Cambridge, Mass. 02139

**Weizmann Institute of Science
Applied Mathematics Department
Rehovot, Israel

***Computer Science Department
U.C. Berkeley
Berkeley, California 94720

{rivest,shamir}@theory.lcs.mit.edu, daw@cs.berkeley.edu

1 Introduction

Our motivation is the notion of "timed-release crypto," where the goal is to encrypt a message so that it can not be decrypted by anyone, not even the sender, until a pre-determined amount of time has passed. The goal is to "send information into the future." This problem was first discussed by Timothy May [6].

What are the applications of "timed-release crypto"? Here are a few possibilities (some due to May):

- A bidder in an auction wants to seal his bid so that it can only be opened after the bidding period is closed.
- A homeowner wants to give his mortgage holder a series of encrypted mortgage payments. These might be encrypted digital cash with different decryption dates, so that one payment becomes decryptable (and thus usable by the bank) at the beginning of each successive month.
- An individual wants to encrypt his diaries so that they are only decryptable after fifty years.
- A key-escrow scheme can be based on timed-release crypto, so that the government can get the message keys, but only after a fixed period (say one year).

There are presumably many other applications.

There are two natural approaches to implementing timed-release crypto:

- Use "time-lock puzzles"–computational problems that can not be solved without running a computer continuously for at least a certain amount of time.
- Use trusted agents who promise not to reveal certain information until a specified date.

Using trusted agents has the obvious problem of ensuring that the agents are trustworthy; secret-sharing approaches can be used to alleviate this concern. Using time-lock puzzles has the problem that the CPU time required to solve a problem can depend on the amount and nature of the hardware used to solve the problem, as well as the parallelizability of the computational problem being solved.

In this note we explore both approaches. (We note that Tim May has suggested an approach based on the use of trusted agents.)

2 Time-lock puzzles

We first explore an approach based on computational complexity: we study the problem of creating computational puzzles, called "time-lock puzzles," that require a precise amount of time to solve. The solution to the puzzle reveals a key that can be used to decrypt the encrypted information. This approach has the obvious problem of trying to make "CPU time" and "real time" agree as closely as possible, but is nonetheless interesting.

The major difficulty to be overcome, as noted above, is that those with more computational resources might be able to solve the time-lock puzzle more quickly, by using large parallel computers, for example. Our goal is thus to design time-lock puzzles that, to the greatest extent possible, are "instrinsically sequential" in nature, and can not be solved substantially faster with large investments in hardware. In particular, we want our puzzles to have the property that putting computers to work together in parallel doesn't speed up finding the solution. (Solving the puzzle should be like having a baby: two women can't have a baby in 4.5 months.) We propose an approach to building puzzles that appears to be intrisically sequential in the desired manner.

Of course, our approach yields puzzles with a solution time that is only *approximately* controllable, since different computers work at different speeds. For example, the underlying technology may be different: gallium arsenide gates are faster than silicon gates. If precise timing of the information release is essential, an approach based on the use of trusted agents is preferable.

We also note that with our approach, the puzzle doesn't automatically become solvable at a given time; rather, a computer needs work continuously on the puzzle until it is solved. A ten-year puzzle needs some dedicated workstation working away for ten years to solve it. If the computing doesn't start until five years after the puzzle was made, then the

(2008 Satoshi Nakamoto)

https://people.csail.mit.edu/rivest/pubs/RSW96.pdf

3. 지능형 악성코드 분석 using NLP
(NLP, National Language Processing)

- **IDEA1** : 프로그램언어와 자연어 간 분류 및 의미분석 유사성 착안 -

TF-IDF, Word2Vec(Code2Vec), K-Fold, K-Means

(Feature Extraction, Relation Analysis, Clustering K-Fold Optimization)

PE Extract (to XML, PEStudioX)

201711_callenge_cisc_or_kr - [Z:₩Study₩2017_Contest₩201711_callenge_cisc_or_kr] - ...₩code₩pe_extract_pestudio.py - PyCharm Community Edition 2017.2.3

File Edit View Navigate Code Refactor Run Tools VCS Window Help

201711_callenge_cisc_or_kr ▸ code ▸ pe_extract_pestudio.py

pe_extract_pestudio

Project

201711_callenge_cisc_or_kr Z:₩Study₩2017_Contest₩201711_callenge_cisc_or_

- 1
- 2
- 3
- 4
- code
  - pe_extract_pestudio.py
- papers_anti_sandbox
- papers_pe_machine_learning
- papers_windows_pe
- tools
- 0a8e89f0cb9334f151f4aacb05f425e4.vir
- challenge_cisc_privacy_201711.pdf
- privacy.hwp
- 개요 – 정보보호 R&D 데이터 챌린지.url
- 새 Microsoft PowerPoint 프레젠테이션.pptx

External Libraries

pe_extract_pestudio.py

```
import os
import shlex, subprocess

Path_PESTUDIO = "..₩₩tools₩₩pestudio₩₩pestudio₩₩pestudiox.exe"
Path_INPUT = "..₩₩4₩₩0_KISA_CISC2017_Intelligent_Mal"
Path_OUTPUT = "..₩₩4₩₩1_PE_PESTUDIO"
#Path_INPUT = "..₩₩4₩₩0.KISA-CISC2017-Intelligent-Mal"
#Path_OUTPUT = "..₩₩4₩₩1.PE_PESTUDIO"

return_code = subprocess.call("dir", shell=True)

filelist = os.listdir(Path_INPUT)
file_count = 0

for x in filelist:
    file_count = file_count + 1
    if x.endswith(".vir"):
        try:
            print(" found ("+str(file_count)+"): " + x)
            #pestudiox.exe -file:inputfilename -xml:outputfilename
            command = Path_PESTUDIO+" -file:"+Path_INPUT+"₩₩"+x+" -xml:"+Path_OUTPUT+"₩₩"+x.replace(".vir", "")+".xml"
            print (command)
            return_code = subprocess.call(command, shell=True)
        except:
            pass
```

Run pe_extract_pestudio

```
found (84): 49bf09214ee263abe30944bfb33a2f36.vir
..₩tools₩pestudio₩pestudio₩pestudiox.exe -file:..₩4₩0_KISA_CISC2017_Intelligent_Mal₩49bf09214ee263abe30944bfb33a2f36.vir -xml:..₩4₩1_PE_PESTUDIO₩49bf09214ee263abe30944bfb33a2f36.xml
found (85): 4d345c9bbc766956d2e0593d7f145dfc.vir
..₩tools₩pestudio₩pestudio₩pestudiox.exe -file:..₩4₩0_KISA_CISC2017_Intelligent_Mal₩4d345c9bbc766956d2e0593d7f145dfc.vir -xml:..₩4₩1_PE_PESTUDIO₩4d345c9bbc766956d2e0593d7f145dfc.xml
found (86): 4db8ea29de568921e0add14b3cbb5ed3.vir
..₩tools₩pestudio₩pestudio₩pestudiox.exe -file:..₩4₩0_KISA_CISC2017_Intelligent_Mal₩4db8ea29de568921e0add14b3cbb5ed3.vir -xml:..₩4₩1_PE_PESTUDIO₩4db8ea29de568921e0add14b3cbb5ed3.xml
found (87): 4ebd556785a59ba61dee55b79e23b189.vir
..₩tools₩pestudio₩pestudio₩pestudiox.exe -file:..₩4₩0_KISA_CISC2017_Intelligent_Mal₩4ebd556785a59ba61dee55b79e23b189.vir -xml:..₩4₩1_PE_PESTUDIO₩4ebd556785a59ba61dee55b79e23b189.xml
found (88): 4ee0dfbb646fbd44762d97a4952e67bd.vir
..₩tools₩pestudio₩pestudio₩pestudiox.exe -file:..₩4₩0_KISA_CISC2017_Intelligent_Mal₩4ee0dfbb646fbd44762d97a4952e67bd.vir -xml:..₩4₩1_PE_PESTUDIO₩4ee0dfbb646fbd44762d97a4952e67bd.xml
found (89): 51265d57bd7a87e0eae385993484c1e9.vir
..₩tools₩pestudio₩pestudio₩pestudiox.exe -file:..₩4₩0_KISA_CISC2017_Intelligent_Mal₩51265d57bd7a87e0eae385993484c1e9.vir -xml:..₩4₩1_PE_PESTUDIO₩51265d57bd7a87e0eae385993484c1e9.xml
found (90): 512d89b87165940ded9439fab243ca3a.vir
..₩tools₩pestudio₩pestudio₩pestudiox.exe -file:..₩4₩0_KISA_CISC2017_Intelligent_Mal₩512d89b87165940ded9439fab243ca3a.vir -xml:..₩4₩1_PE_PESTUDIO₩512d89b87165940ded9439fab243ca3a.xml
```

6:37 n/a UTF-8

PE Extract (to XML, PEStudioX)

file:///Z:/Study/2017_Contest/201711_callenge_cisc_or_kr/tools/pestudio/pestudio/out.xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
<!--
  pestudio 8.69 - Malware Initial Assessment - www.winitor.com
-->
<image name="pestudiox.exe">
  <overview>
    <file-description>Malware Initial Assessment - www.winitor.com</file-description>
    <file-version>8, 69, 0, 0</file-version>
    <created>00:00:0000 - 00:00:00</created>
    <cpu>32</cpu>
    <size>67584</size>
    <type>executable</type>
    <subsystem>Console</subsystem>
    <signature>Microsoft Visual C++ 8</signature>
    <entropy>-</entropy>
    <md5>CAB0248D6E34C591678ZC4A6BF47EBAF</md5>
    <sha1>CB905965C4783FB198769B33472ED9B26785FCDA</sha1>
    <imphash>90FFC7FB2366116E16ECE3878F9EB483</imphash>
  </overview>
  <indicators count="11">
    <indicator severity="2">The file installs an Exception Handler</indicator>
    <indicator severity="2">The file is scored (0/68) by virustotal</indicator>
    <indicator severity="5">The file signature is 'Microsoft Visual C++ 8'</indicator>
    <indicator severity="9">The file references (96) whitelist strings</indicator>
    <indicator severity="3">The file opts for Data Execution Prevention (DEP)</indicator>
    <indicator severity="5">
      The file opts for Address Space Layout Randomization (ASLR)
    </indicator>
    <indicator severity="5">The file opts for cookies on the stack (GS)</indicator>
    <indicator severity="7">The file does not contain a digital Certificate</indicator>
    <indicator severity="9">The file ignores Code Integrity</indicator>
    <indicator severity="3">The file references (7) rtti string(s)</indicator>
    <indicator severity="5">
      The file is statically linked to the C Runtime Library
    </indicator>
  </indicators>
  <virustotal>-</virustotal>
  <dos-stub>
    <size>168</size>
    <md5>CAB0248D6E34C591678ZC4A6BF47EBAF</md5>
    <entropy>5.029</entropy>
  </dos-stub>
  <dos-header>disabled</dos-header>
```



```
  <sections count="5">
    <section>
      <name>.text</name>
      <virtualSize>43887</virtualSize>
      <VirtualAddress>0x00001000</VirtualAddress>
      <SizeOfRawData>44032</SizeOfRawData>
      <PointerToRawData>0x00000400</PointerToRawData>
      <PointerToRelocations>0x00000000</PointerToRelocations>
      <PointerToLinenumbers>0x00000000</PointerToLinenumbers>
      <NumberOfRelocations>0x0000</NumberOfRelocations>
      <NumberOfLinenumbers>0x0000</NumberOfLinenumbers>
      <md5>9FC7A0901C74B09F448346C3EB41AA56</md5>
      <entropy>6.592</entropy>
      <initialized-data>0</initialized-data>
      <uninitialized-data>0</uninitialized-data>
      <discardable>0</discardable>
      <cannotBeCached>0</cannotBeCached>
      <cannotBePaged>0</cannotBePaged>
      <shareable>0</shareable>
      <executable>1</executable>
      <readable>1</readable>
      <writable>0</writable>
    </section>
    <section>
      <name>.rdata</name>
      <virtualSize>11334</virtualSize>
      <VirtualAddress>0x0000C000</VirtualAddress>
      <SizeOfRawData>11776</SizeOfRawData>
      <PointerToRawData>0x0000B000</PointerToRawData>
      <PointerToRelocations>0x00000000</PointerToRelocations>
      <PointerToLinenumbers>0x00000000</PointerToLinenumbers>
      <NumberOfRelocations>0x0000</NumberOfRelocations>
      <NumberOfLinenumbers>0x0000</NumberOfLinenumbers>
      <md5>BF7414114AC87012B75643342B8E8ED0</md5>
      <entropy>5.089</entropy>
      <initialized-data>1</initialized-data>
      <uninitialized-data>0</uninitialized-data>
      <discardable>0</discardable>
      <cannotBeCached>0</cannotBeCached>
      <cannotBePaged>0</cannotBePaged>
      <shareable>0</shareable>
      <executable>0</executable>
      <readable>1</readable>
      <writable>0</writable>
    </section>
```

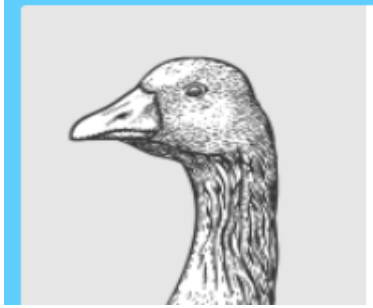
데이터셋 (KISA-CISC2017-Intelligent-Mal) **295개 기초분석 결과** (파일용량 5M 이상제외)

대분류	소분류	검출결과	비고
패킹	Themida	21	대표적인 패커만 확인함 (대략 전체평균은 40% 이상임, `14 BlackHat 등)
	ASPack	5	
	UPX	47	
	MPress	10	
	합계	83(28%)	
디버거	Debugger	42	
ANTI VM	vbox	4	VBoxSv
레지스트리	RegOpenKey	90	
	RegCloseKey	94	
기타	파일생성시간	8	(00:00:0000)
	file-description	51	Empty

* 실험환경, CPU i7, GPU980*2, Windows10, Ubuntu16


Yeah!

kaggle Search kaggle Competitions Datasets Kernels Discussion Jobs ...



jason, min

Joined 3 years ago · last seen in the past day

Competitions Contributor

3 competitions entered

Sort by Best Results

Completed Active **Tutorial**

AdaBoosting, Decision Tree, NN Hybrid

Bike Sharing Demand
3 years ago · Top 1% · Competition Ineligible for Medals
36 entries as a solo competitor

23rd of 3251

3 competitions entered

Sort by Best Results

Completed Active Tutorial

Caterpillar Tube Pricing
2 years ago · Top 38%
1 entries as a solo competitor

500th of 1323

Homesite Quote Conversion
2 years ago · Top 39%
4 entries as a solo competitor

677th of 1764


Yeah!
Yeah!

지능형 악성코드 (고찰)

□ **(지능형 악성코드)** 코드(암호화, 난독화 등), 행위수준에서 이미 **엔트로피**가 매우 높은 상태임
* 코드, 행위의 엔트로피 값을 일반/지능형 악성코드의 구분기준으로 고려할 수 있음

□ **(분석의 어려음)** 지능형 악성코드는 일반 데이터분석이 용이한 수준으로 특징*(feature)을 뽑는데 어려움이 있음
* 정적분석(PE추출, 언패킹, 디스어셈블 등), 동적분석(디버깅 등), 동적행위분석(VM분석 등)

□ **(방법론)** 지능형 악성코드 판단을 강화하기 위해서는, 더 높은 차원의 상관관계분석이 가능한 자동화 분석기법의 적용이 필수적이라 판단됨

⇒ **번역, 요약, 분류, 작성(기사 작성 등) 자연어 처리기술 용도로 개발된 기술을 프로그래밍언어 분석처리에 적용하고자 함**

* 본 연구에서는 PE추출데이터를 기반으로 적용하였으며
향후 언패킹, 디스어셈블, 동적분석, 동적행위분석 등의 추가정보를 이용,
정상/일반악성코드/지능형악성코드 분류 연구를 제안

⇒ **적용가능기술 : TF-IDF, Word2Vec(Code2Vec), K-Fold, K-Means, Vector Distance 연산**

[참고]

Anti-Debugging
Techniques to compromise debuggers and/or the debugging process
"debugger"

Anti-Disassembly
Techniques to compromise disassemblers and/or the disassembling process - packing

Obfuscation
Techniques to make the signatures creation more difficult and the disassembled code harder to be analyzed by a professional

Anti-VM
Techniques to detect and/or compromise virtual machines
"VM, vmware, vbox, virtual"

Malicious Technique
A characteristic we look for in the scope of this research.
Not necessarily all software using such technique is malicious

"wireshark", Hook Imports suspicious APIs

RegCloseKey RegOpenKeyW RegOpenKeyExW CreateToolhelp32Snapshot Process32NextW LoadLibraryW GetModuleFileNameW CreateFileW **IsDebuggerPresent** Process32FirstW connect (Ordinal #4) socket (Ordinal #23) send (Ordinal #19)


ARE YOU
PREPARED?

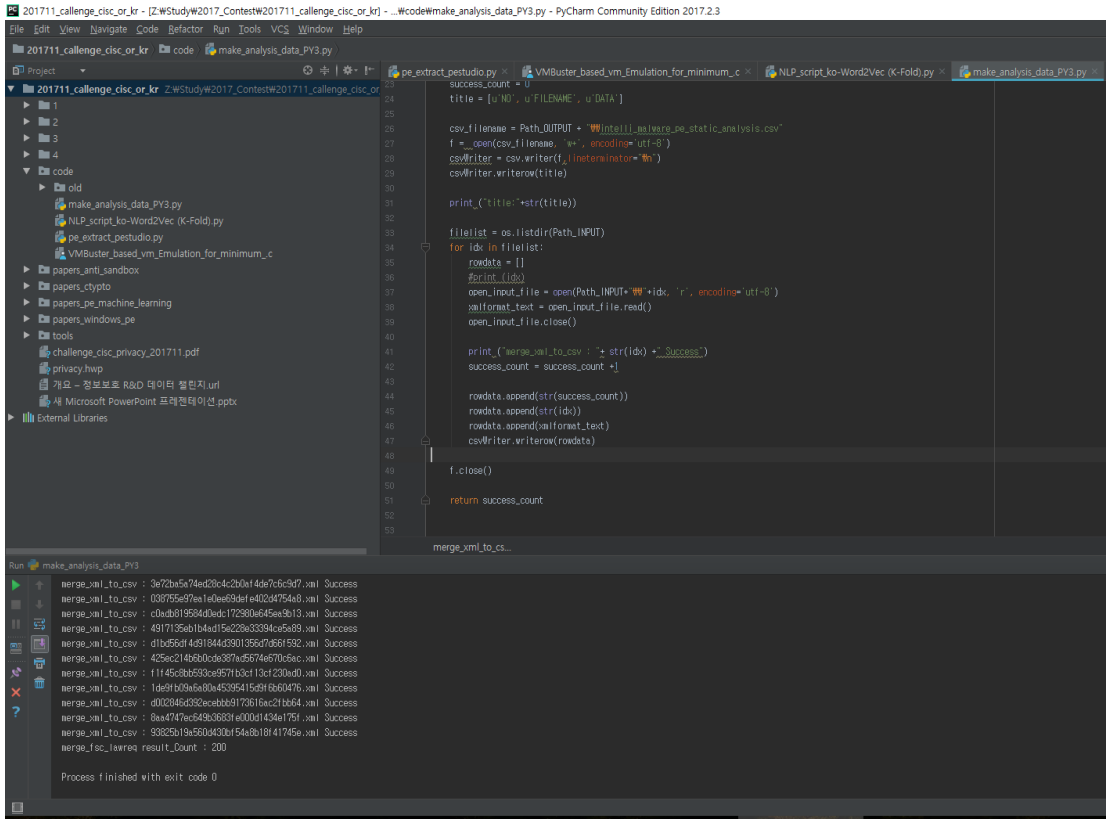

TIME TO PLAN

Code2Vector - 제안

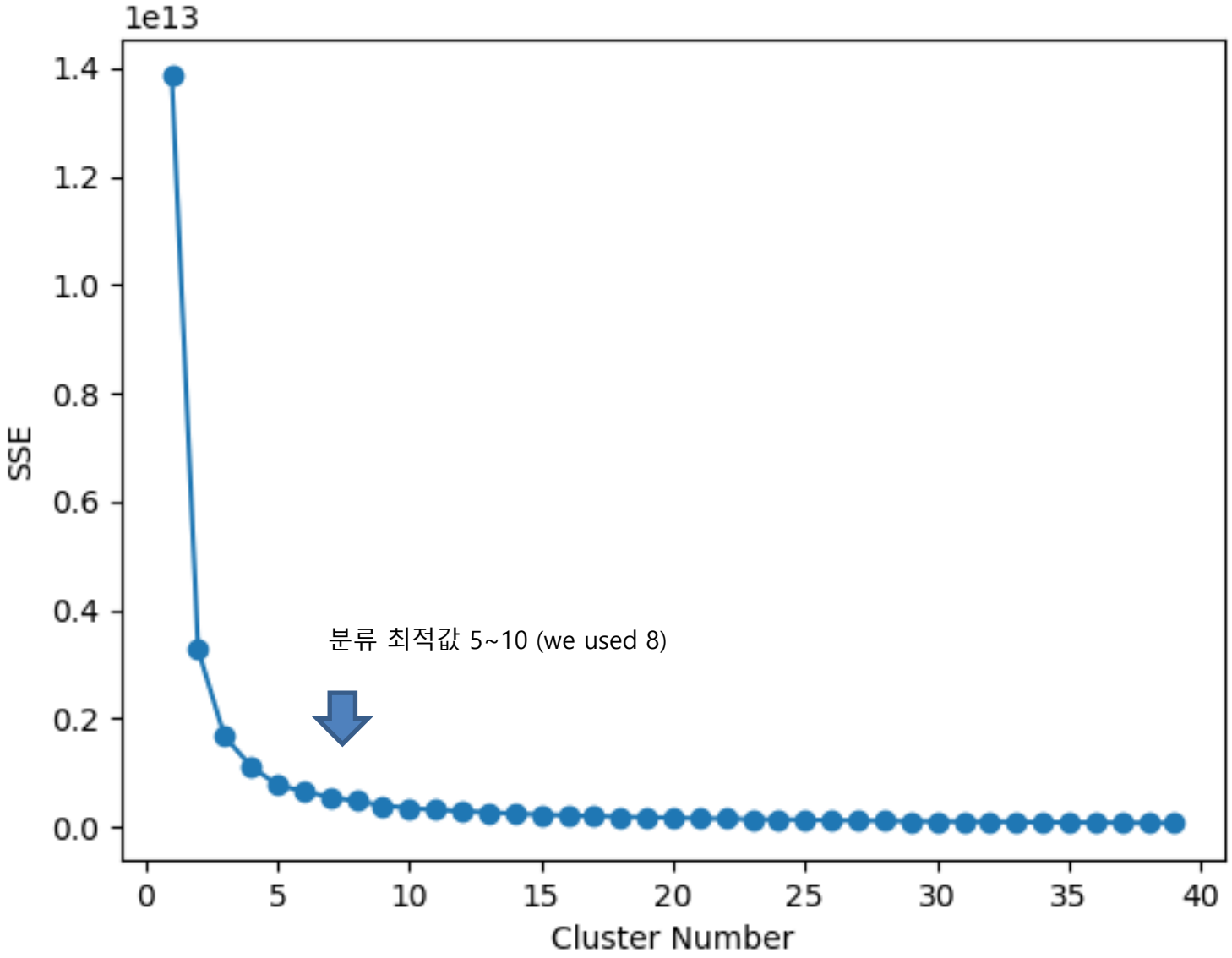
제안 방법론

295개의 악성코드에서 정적, 동적 정보를 획득한 후 각각 중요 정보추출
200개 TF-IDF 백터를 기반으로 악성파일 유사도를 Word2Vec 방법으로 분석하여, 악성코드를 분류한다.

(일반 텍스트문서를 통한 예제 검증 3차 후, 실제 악성코드에 대한 정보를 대상으로 실 구현 및 검증, 본 자료에서는 정적PE정보에 대해서 분류 수행)



	A	B	C
1	NO	FILENAME	Cluster
2	1	c2589f6f42f8e08b26adccb51bd08e87.xml	2
3	2	5d8ba3980cd9c87f4e49da01319bb6cc.xml	2
4	3	ec1cc4c369f0bbd3ae63dfcaa705a2db.xml	2
5	4	71eb05afa48d6990c4b73cc13542b874.xml	2
6	5	a30cac477e5c3281f537bdaa95b0cd64.xml	2
7	6	5eaed1f4be1b3b55281de29f3d9a180c.xml	2
8	7	f0a221fa93d60b90858641a9e8aa96b3.xml	2
9	8	90beb8c82b6322b7a7346fd01c149c0a.xml	2
10	9	a70ac9d6715304311aa4613dd22b6ffa.xml	2
11	10	14d66e18d8611995cb01b805726b5e63.xml	2
12	11	e502c522fd6999cfcd9bad80defae0a1.xml	2
13	12	73ce47c821c4515e981cbfddc6f37202.xml	2
14	13	b47d7947dc039e9c39974ab1b8d3b3b5.xml	2
15	14	6b81d6aedc4ad7ff9b4c0f7cd9f1f7ae.xml	7
16	15	372ed0713e541f9e3271a7807cc3f005.xml	2
17	16	786e0234076abf8a1908d6505fd776c4.xml	2
18	17	5dc56fea6ca6052ec7e0d673e90dfbfd.xml	4
19	18	4ee0dfbb646fbd44762d97a4952e67bd.xml	2
20	19	48afe8490ddfcc9e37dd4f357cc1ace1.xml	2
21	20	ba5a5c9ddb53eae2af28723340acd7a1.xml	7
22	21	1573ffadefb5b904e5f793b4904515e3.xml	1
23	22	9f332f002a7a7fd057d4a1cbbf780a01.xml	7
24	23	b5bbc0d792a1fb2f4c1fd1b6003b6db3.xml	7
25	24	a2cc8fb2f64b60ab0a8b8a93744dcea8.xml	7
26	25	622c8908adb320d1828a697c48bab4f5.xml	7


1e13
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
SSE
분류 최적값 5~10 (we used 8)
0
5
10
15
20
25
30
35
40
Cluster Number

분류	방법	비고
pe_extract_pestudio.py	vir파일(300개) 에서 PE 추출 (295개, XML 포멧)	
make_analysis_daya_PY3.py	PE 추출 XML(295개)에서 CVS로 저장(200개, 저용량 우선)	
NLP_script_ko-Word2Vec (K-Fold).py	추출된 정보를 기반으로 Word2Vec 알고리즘으로 악성코드 분류 수행	K=8

[지능형 악성코드 분류]

Type A : a8c62fc19cbfaa3f1075f4ca5f7ac7b6 등 41개
Type B : 1573ffadefb5b904e5f793b4904515e3 등 19개
Type C : c2589f6f42f8e08b26adccb51bd08e87 등 17개
Type D : 2c6ade9e8752637add9b85bb0df89b7b 등 41개
Type E : 5dc56fea6ca6052ec7e0d673e90dfbfd 등 21개
Type F : e23fd00721f5a745571e4283ccd05ef8 등 47개
Type G : fbf7dfaa30bb99a33caa464d64d107eb 1개
Type H : 6b81d6aedc4ad7ff9b4c0f7cd9f1f7ae 등 13개

* 본 분류기준으로 신규PE파일에 대해 벡터거리연산으로 분류 가능함

(IDEA1) 지능형 악성코드 머신러닝 분석 결과

□ **번역, 요약, 분류, 작성(기사 작성 등) 자연어 처리기술을 프로그래밍언어 처리에 적용하였고, 지능형악성코드 분류를 수행함**

* 본 연구에서는 PE추출데이터를 기반으로 적용함

4. 지능형 악성코드 분석 선제대응 방안(피해최소화) 설계

- **IDEA2** : 계층형 격리를 통한 형상관리, 가상머신 에뮬레이션 설정 -

지능형 악성코드 선제대응을 위해 “질병관리센터 격리체계”를 수용한 방법론 제안

격리와 예비격리

공중 보건 담당 공무원은 SARS(중증 급성 호흡기 증후군), TB(결핵), 또는 이와 유사한 다른 전염병이 발생하는 경우, 이러한 전염병의 전파를 방지하기 위해 "격리(isolation)와 예비격리(quarantine)"라는 방법을 사용합니다.

- **격리**는 이미 병에 걸린 사람에 대한 조치이고,
- **예비격리**는 전염병에 걸린 사람과 접촉했기 때문 곧 그 병에 걸릴 가능성이 있는 사람에 대한 조치입니다.

격리란 무엇입니까?

격리란 전염병에 걸린 사람을 건강한 사람들로부터 떼어놓는 것을 말합니다.

전염병에 걸린 사람들은 가족과 다른 사람들을 보호하기 위해 스스로 격리를 택하는 경우가 많습니다. 수두에 걸린 자녀를 학교에 보내지 않고 집에 있게 하는 것이 격리의 한 가지 예입니다. 병원에서는 SARS나 TB 같은 병에 걸린 환자를 다른 환자들로부터 격리시킵니다.

격리 조치는 반드시 따라야 합니까?

그렇습니다. 보건 담당 공무원이 다른 사람들로부터 떨어져 있어야 한다고 명령하는 경우 이러한 명령을 따라야 합니다.

격리된 후에는 어떻게 됩니까?

병을 앓고 있는 동안 치료를 받습니다. 치료는 다음과 같은 장소에서 받을 수 있습니다.

- 집
- 병원, 또는
- 다른 의료시설

격리된 환자는 격리 구역을 떠나면 안됩니다. 간호하는 사람만이 격리 구역에 들어갈 수 있습니다.

예비격리란 무엇입니까?

예비격리란 전염병에 걸린 사람과 접촉한 사람을 병에 걸리지 않은 사람들로부터 떼어놓는 것을 말합니다.

예비격리 조치는 아직 병에 걸리지는 않았으나 곧 병에 걸려서 전염병을 다른 사람에게 전파할 가능성이 있는 사람을 관찰하기 위해 사용합니다.

예비격리 조치는 반드시 따라야 합니까?

그렇습니다. 보건 담당 공무원은 전염병과 접촉했다고 생각되는 사람에게 다른 사람으로부터 떨어져 있으라고 명령할 수 있습니다.

격리된 후에는 어떻게 됩니까?

지시에 따라 다음과 같이 행동해야 합니다.

- 집에 머물러 있고,
- 다른 사람으로부터 떨어져 있고,
- 스스로 증상이 나타나는지 확인하고,
- 증상이 나타나면 의사에게 연락하십시오.

격리와 예비격리는 사람들이 병에 걸리는 것을 막는 효과가 있습니까?

그렇습니다. 미국에서는 2003년에 격리와 예비격리를 통해서 사망률이 높은 전염병인 SARS의 전파를 막을 수 있었습니다. 이러한 조치는 치료 방법, 백신 또는 특효약이 없는 경우에도 사람들이 건강을 유지하도록 도와줍니다.

더 자세한 정보를 원하시면...

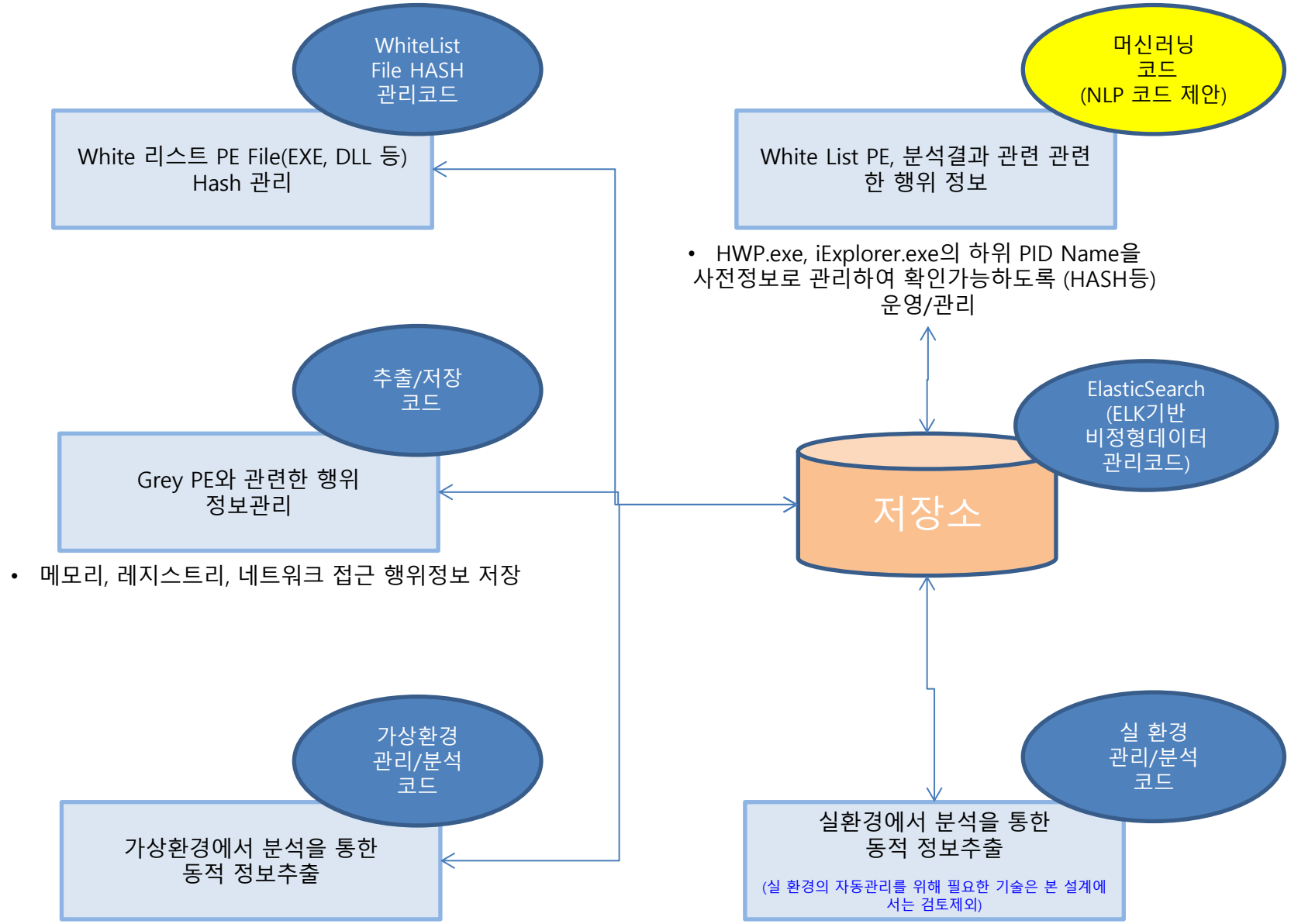
아래의 번호로 CDC (Centers for Disease Control)에 전화하여 영어로 도움을 받으십시오.

800-CDC-INFO (800-232-4636)
888-232-6348 (TTY)

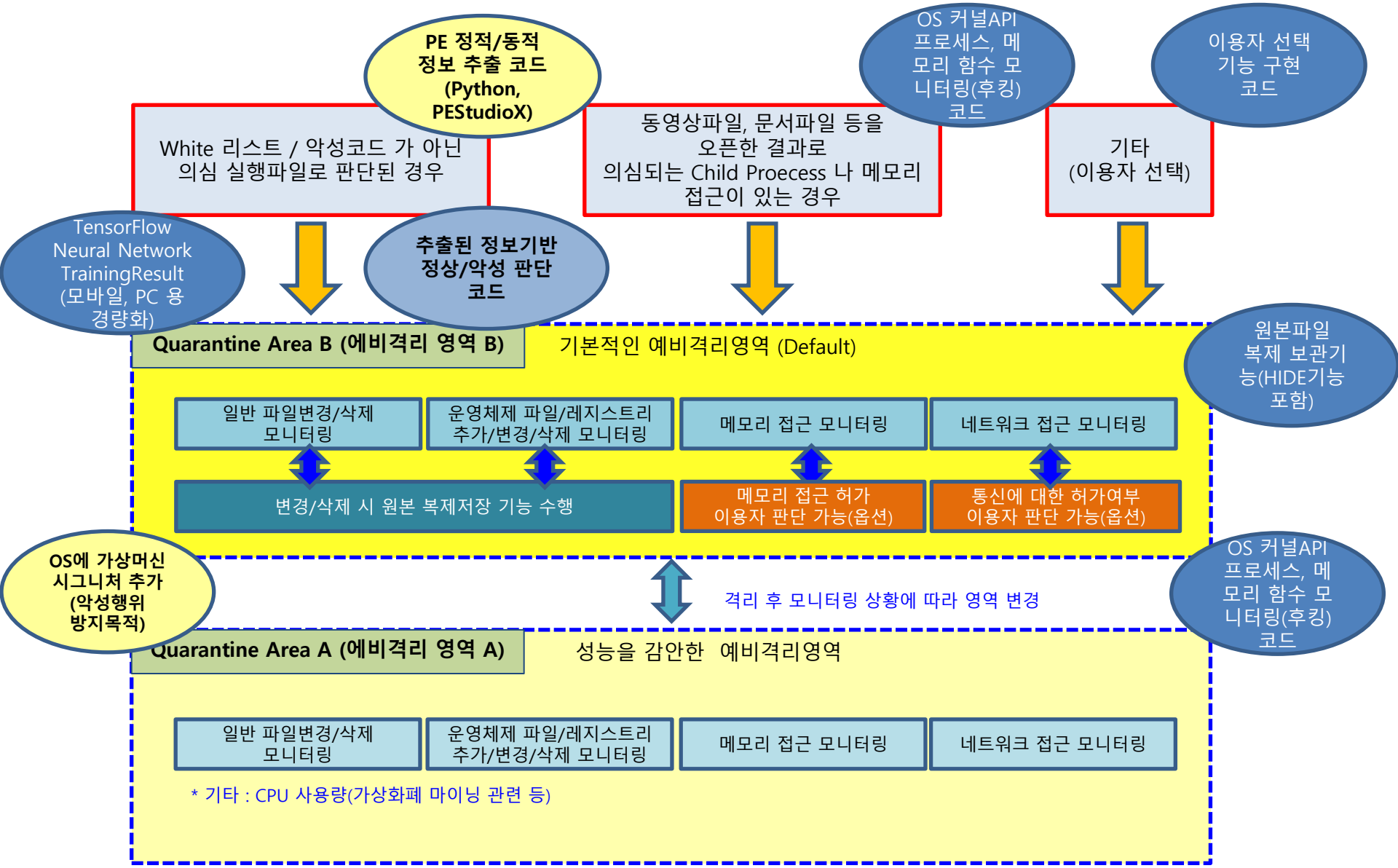
또는 다음의 웹사이트를 방문하십시오.

www.bt.cdc.gov/ncidod
www.dhs.ca.gov/epo

알고리즘 설계 (서버영역)


WhiteList
File HASH
관리코드
White 리스트 PE File(EXE, DLL 등)
Hash 관리
머신러닝
코드
(NLP 코드 제안)
White List PE, 분석결과 관련 관련
한 행위 정보
• HWP.exe, iExplorer.exe의 하위 PID Name을
사전정보로 관리하여 확인가능하도록 (HASH등)
운영/관리
추출/저장
코드
Grey PE와 관련한 행위
정보관리
• 메모리, 레지스트리, 네트워크 접근 행위정보 저장
ElasticSearch
(ELK기반
비정형데이터
관리코드)
저장소
가상환경
관리/분석
코드
가상환경에서 분석을 통한
동적 정보추출
실 환경
관리/분석
코드
실환경에서 분석을 통한
동적 정보추출
(실 환경의 자동관리를 위해 필요한 기술은 본 설계에
서는 검토제외)

알고리즘 설계 (클라이언트 영역)



4-A. 계층형 격리를 통한 형상관리 방법 (Pseudo Code)

예시 : 파일, 레지스트리, 프로세스, 메모리와 같은 커널 함수에 후킹을 걸어둔 후 QA(Quarantice Area)에서 관리중인 프로세스를 모니터링 하는 기능과 정보의 변경전 해당 정보(파일, 레지스트리 등)는 별도의 영역에 보관한다.

```
If pid == QA 내부 PID :
    If WriteFile, WriteRegistry 명령인 경우:
        해당 정보는 미리 백업 후 명령 수행
```

4-B. 격리(가상머신) 에뮬레이션 설정 방법 (Pseudo Code)

예시 : 아래와 같은 VM 탐지 기술이 이용되는 경우 실제 PC의 환경과 달리 가상머신으로 결과값을 리턴한다.

```
void vmx_checking_emulation();
void process_name_checking_emulation();
void class_name_checking_emulation();
void cpuid_checking_emulation();
void cpu_cores_checking_emulation();
void registry_checking_emulation();
void devices_checking_emulation();
void drivers_checking_emulation();
```

(IDEA2) 계층형 격리를 통한 형상관리 & 격리(가상머신) 에뮬레이션 설정

□ VM탐지기술이 적용된 악성코드의 비활성화 효과와 파일변경/삭제 기능이 있는 악성코드에 대한 선제적 형상관리 대응을 통한 피해방지 효과기대

5. 결론 및 향후 연구방안

[결론] "지능형 악성코드" 선제대응방안

□ **번역, 요약, 분류, 작성(기사 작성 등) 등 최신 자연어 처리기술을**
① 프로그래밍 언어 처리에 적용하였고, ②지능형악성코드 분류를 수행함

* (**TODO** A) 중요키워드, 함수 등 의미를 고려한 Code(Word) 추출 연구 필요
  - 언패킹, 디스어셈블, 동적분석, 동적행위분석 등의 추가정보를 이용
  - 정상/일반악성코드/지능형악성코드 분류 연구 제안

* (**TODO** B) Part2에서 제공된 일반악성코드와 비교가능성 검토

□ **VM탐지기술이 적용된 ③악성코드의 비활성화 효과와 파일변경/삭제 기능이 있는 악성코드에 대한 선제적 ④형상관리 대응을 통한 피해방지/대응 방법 제시**

* (**TODO** C) 가상머신 우회, 파일삭제, 암호화 악성코드를 대상으로 **실제 구현** 후 테스트, 결과를 확인하여 효과(%) 확인

"만약 당신이 미래를 꿈꾸지 않거나 지금 기술개선을 위해 노력하지 않는다면 그건 곧 낙오되고 있는 것이나 마찬가지 입니다."

그윈 쇼트웰(Gwynne Shtwell, SpaceX CEO, COO)

감사합니다

(facebook.com/sangshik, mikado22001@yahoo.co.kr)