Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.27, NO.5, Oct. 2017 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2017.27.5.1077

Section, DLL feature 기반 악성코드 분석 기술 연구*

황 준 호,† 황 선 빈, 김 호 경, 하 지 희, 이 태 진‡
호서대학교

Malware Analysis Based on Section, DLL*

Jun-ho Hwang,† Seon-bin Hwang, Ho-gyeong Kim, Ji-hee Ha, Tae-jin Lee‡
Hoseo University

요 약

기존 악성코드를 기반으로 만들어지는 변종 악성코드들은 약간의 패턴 변화로도 기존 보안체계를 쉽게 회피할 수 있고 제작 과정이 간단하여 널리 사용되고 있다. 이러한 악성코드는 일평균 160만개 이상 출현하고 있고, 사이버 공간 뿐 아니라 피해규모가 큰 IoT/ICS로 점차 확대되고 있다. 본 논문에서는 기존에 자주 이용되는 Pattern기반 분석, Sandbox기반 분석, CFG/Strings 기반 분석 등이 아니라, 큰 의미를 부여하지 않았던 PE Section 및 DLL의 특징에 기반한 분석방법을 제안한다. 제안모델을 실제 구축 및 실험결과, 유의미한 탐지율과 오탐율을 기록했으며, 기존의 다양한 분석기술을 복합 운영 시 효과적인 악성코드 대응이 가능할 것으로 기대된다.

ABSTRACT

Malware mutants based on existing malware is widely used because it can easily avoid the existing security system even with a slight pattern change. These malware appear on average more than 1.6 million times a day, and they are gradually expanding to IoT / ICS as well as cyber space, which has a large scale of damage. In this paper, we propose an analytical method based on features of PE Section and DLL that do not give much significance, rather than pattern-based analysis, Sandbox-based analysis, and CFG, Strings-based analysis. It is expected that the proposed model will be able to cope with effective malicious code in case of combined operation of various existing analysis technologies.

Keywords: Malware, DLL, Static analysis, Classification, Section

I. 서 론

전 세계적으로 일평균 악성코드는 약 160만개 이상 출현하고 있다. 이러한 악성코드들은 특정 국가나 공용소프트웨어, 사회기반시설, 금융 서비스, 사물인터넷(IoT) 등을 대상으로 사이버 공격을 수행하기 위한 것으로 최근에는 대규모 악성코드 감염기법의 지능화와 다양한 형태의 랜섬웨어 유포 등으로 인한 피해가 심각하다. 대규모로 제작되어 유포되는 다양한 형태의 변종 악성코드들은 기존의 악성코드에서 약간의 패턴 변화로 Anti-malware 체계의 분석 및 탐지를 회피한다. 이러한 변종 악성코드들은 제작 과정이 간단하고 목적 또한 분명하기 때문에 하루에도 수많은 변종 악성코드들이 제작된다. 전문가가 이러한 변종 악성코드들을 일일이 분석하는 것은 현실적으로 어렵기 때문에 자동화된 메커니즘을 통하여 분류하는 것이 필요하다는 것은 분명하다. PE(portable executable) 파일은 일반적으로 실행 가능한 파일로 EXE, SCR, DLL, SYS 등이

Received(06. 19. 2017), Modified(1st: 08. 17. 2017, 2nd: 09. 11. 2017), Accepted(09. 21. 2017)
* 이 논문은 2017년도 호서대학교의 재원으로 학술연구비 지원을 받아 수행된 연구임(2017-0053)
† 주저자, hwangso93@gmail.com
‡ 교신저자, kinjecs0@gmail.com (Corresponding author)

있는데 악성코드의 경우 대부분 PE 파일 이므로 프로세스나 서비스의 형태로 PE 파일 포맷을 따르게 된다. 따라서 본 논문에서는 악성코드들이 PE 파일 포맷을 따르는 것에 착안하여 PE 파일 포맷에 대하여 분석하고 feature를 선별 및 feature들의 통계학적 근거를 기반으로 한 악성코드 탐지를 수행하고자 한다. 특히 기존의 PE 파일 포맷 분석 방법의 회피기술에 크게 영향을 받지 않는 Section 영역의 특성을 분석하여 feature를 선정하고 통계학적으로 접근함으로서 변종 malware를 분석하는 메커니즘을 제시한다. 이러한 메커니즘은 기존의 PE 분석 방법들이나 단순 통계학적 분석과는 다르게 상호 독립적인 feature들에 대해 bayesian 확률 접근법으로 가능하다. 다음으로 2장에서는 기존 악성코드 분석 기법의 특징 및 장단점을 기술하고, 3장에서는 본 논문에서 제안한 Section, DLL에 기반한 feature를 제안하고, 4장에서는 성능 분석 결과를 제시하고, 5장에서는 본 분석 알고리즘의 의미와 결론을 제시한다.

II. 관련 연구

악성코드 분석에 대한 다양한 연구가 진행되고 있다[1,2]. 악성코드와 정상파일에서 사용하는 API를 추출하고 해당 API가 악성코드에서 발생하는 확률에 따라 가중치를 산정하여 악성코드를 분류하는 연구[3], 파일의 복잡도를 측정하여 실행 압축을 판단하고 악성코드 탐지체계에 적용하여 탐지율을 높이는 방법[4], 바이너리 파일 전체를 길이 N의 Sub-string으로 나누는 N-gram 기법을 활용하여 악성코드 분류에 적용하는 방안[5], 패킹 또는 난독화의 흔적에 대하여 바이너리 파일의 구조에 대한 정적 검사를 수행하여 위험 등급을 산정하여 바이너리의 악성 여부를 판단하는 연구 등이 있다.[6] 이러한 연구 동향 중 API가 악성코드에서 발생하는 확률에 따라 가중치를 산정하여 악성코드를 구분하는 threshold를 지정하는 방법의 경우에는 정적, 동적 분석 기법을 병행하기 때문에 기본적으로 연산량을 많이 필요로 하고 최근의 악성코드들은 API 기반의 탐지 기법을 회피하기 위해 사용하지 않는 API를 코드에 추가한다. 또, 악성코드의 악성 행위를 하는 기능이 API에 기반 하지 않아 분석을 통하여 지정한 악성 행위가 의심되는 API가 호출되지 않음으로 인해 분류 기준에 중대한 문제가 생기는 등 악성코드의 내부 API 설계, 코딩에 따라 정확한 threshold 설정이 어려운 단점이 있다. 변종 악성코드를 탐지하기 위해 opcode sequence에 대한 N-gram(n=3)을 feature로 하고, feature hashing과 클러스터 링 등의 기법을 혼합하여 prototype을 기반으로 악성코드를 탐지하는 방안이 있다. 하지만 위와 같은 악성코드 분석이 가능하려면 난독화 된 악성코드의 의미 없는 코드를 제거하고 Fig. 1.과 같이 normalization 수행하여 해독하는 방안을 사용하여 앞서 난독화된 악성코드를 어셈블하는 것이다[7, 8]. Fig. 1.은 Listing 1의 기존 악성코드를 난독화 한 Listing 2, 그것을 다시 어셈블하는 난독화된 악성코드의 normalization 과정을 나타내는 것으로 앞서 기술한 것과 같이 기존의 분석 모델들은 이러한 부가적인 과정이 필요하다. 또한 제안한 feature의 결과를 AISec의 multple

Table 1. Common Static Analysis Feature

Byte Sequence	File Size
dll name	function name in Dll and relocation table
String information	Control flow graph
Static binary	Disassembled binary files
Dynamic command tracing	System command trace
Assembly instructions (x86)	ASCII format hexadecimal code
opcode sequence	ASCII format hex code

```
    lea  eax, [ebp+Data]
    push esi
    push eax
    call ds:GetWindowsDirectoryA
    lea  eax, [ebp+Data]
    push eax
    call _strlen
    cmp  [ebp+eax+var_129], 5Ch
    pop  ecx
    jz   short loc_40
    lea  eax, [ebp+Data]
    push offset asc_408D80
    push eax
    call _strcat
    pop  ecx
    pop  ecx
loc_40:
    lea  eax, [ebp+Data]
    push offset aServices_exe
    push eax
    call _strcat
```

Listing 1: Example of a malware code fragment.

```
    jmp  lab
lan: add  [esp], 1
    jmp  lay
lop: cld
    jmp  lah
lac: scasb
    jmp  lam
laz: mov  al, 99
    jmp  lop
lav: loop lop
    jmp  law
las: dec  edi
    jmp  lav
lab: mov  edi, offset der
    jmp  laz
lam: push edi
    jmp  lan
lay: pop  edi
    jmp  las
lah: xor  byte ptr [edi],1
    jmp  lac
law: jmp  short der
    ...
der: db 8c 84 d9 ff fe fe
    db ...
    db 01 98
```

Listing 2: Malware instance obfuscated from Listing 1.

```
    lea  eax, [ebp+data1]
    push esi
    push eax
    call ds:GetWindowsDirectoryA
    lea  eax, [ebp+data1]
    push eax
    call _strlen
    cmp  [ebp+eax+data2], 5Ch
    pop  ecx
    jz   short label1
    lea  eax, [ebp+data1]
    push offset data3
    push eax
    call _strcat
    pop  ecx
    pop  ecx
label1:
    lea  eax, [ebp+data1]
    push offset data4
    push eax
    call _strcat
```

Listing 3: Normalized malware instance derived from Listing 2.

Fig. 1. Normalization of Obfuscated Malicious Code

data source에 추가하여 분류 알고리즘(SVM, Decision Tree, kNN 등)을 이용하는 머신러닝 프로세스에 적용[9]하면 기존의 악성코드 탐지 정확도 보다 기대되는 탐지율이 나올 수 있다.

본 논문에서는 이러한 문제점들을 기반으로 기존에 큰 의미를 부여하지 않았던 다양한 section에 대해 분석하고 샘플 label별 유사도를 측정하여 악성코드를 분석하는 메커니즘을 제시한다. 특히, 본 논문에서는 악성코드를 실행하지 않고 사용하는 동적 라이브러리 정보 (DLL: dynamic linking library), PE (portable executable) 등을 활용해 악성코드의 구조를 분석할 수 있는 기본 특징이외에도 악성코드 분석회피에 사용되는 다양한 난독화 기법의 영향을 크게 받지 않는 feature들을 주로 사용하였다. 이는 기존의 PE 분석 방법들이나 단순 통계학적 분석과는 다르게 상호 독립적인 feature들에 대해 bayesian 확률 접근법으로 가능하다. 다음 장에서 자세한 제안모델을 제시한다.

III. 제안 모델

본 논문에서 제안하는 악성코드 분석시스템의 구성은 Fig. 2.와 같다. 먼저, 기존에 분석 완료된 악성코드, 정상파일에 대해서 Table 1과 같이 이전에 악성코드를 식별하기 위해서 사용한 feature들이 아닌 새로운 feature를 이용한 분석기술을 제안한다. 제안하는 알고리즘은 section과 DLL에 기반한

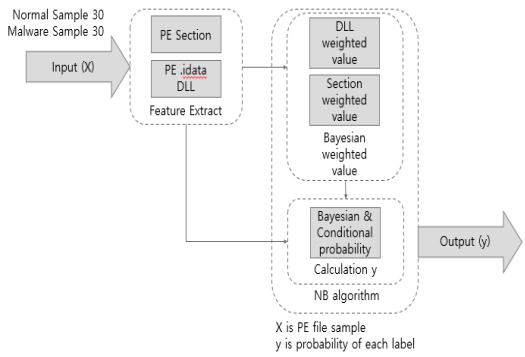

Fig. 2. System Configuration Diagram

feature로 이를 이용하여서 baysian 가중치를 계산하고, 분석대상 파일에 대해서 악성여부를 자동으로 판별한다. PE 파일포맷에서 section과 .idata section에 있는 import된 DLL에 대한 데이터를 추출 한 후, baysian 가중치를 산정하게 되고 이것을 이용하여 분석대상 파일에 대해 bayes 정리와 conditional probability를 이용하여 결과 값을 계산한 후 확률에 근거하여 분석대상 파일에 대한 악성여부를 결정하게 된다.

다음으로 3.1에서는 PE 포맷에서의 section의 의미와 특징을 설명하고, bayesian 모델에 기반한 분석결과를 제시한다. 3.2에서는 section과 연관된 DLL 관점에서 의미와 특징을 설명하고 분석결과를 제시하고자 한다.

Table 2. Purpose of Using General Section in PE file

Division	Main Content
.text	Save the code to be executed when the file is opened
.data	Saved Initialized Global Variables, Saving Static Variables Saving Initialized Global and Static Variables
.rdata	Storing const variables, string constants
.bss	Stores uninitialized global variables, static variables, strings, and other constants
.edata	Save information related to EAT
.idata	Saving information related to IAT
.rsrc	Save resources
.didat	Area for delay loading import data
.reloc	Store relocation information for the executable
.tls	Section for thread local storage declared with thread directives
.crt	Section added to support the C ++ runtime
.sdata	A readable short data section that can be addressed relative to the global pointer
.pdata	Section containing exception information

3.1 Section 기반 악성코드 분석 방안

3.1.1 Section

Section은 PE 파일의 각종 정보가 저장되어 있는 영역으로 Section 별로 용도가 정해져 있다. 그러나, 악성코드는 이 포맷을 그대로 따르지 않고, 소스코드 분석을 어렵게 하기 위한 packing, encryption, polymorphic 등 난독화 기법 등을 적용하면서 Section 이름을 추가·삭제·변경하는 경우가 빈번히 발생한다. Table 2은 일반적인 section명의 사용목적을 나타낸다. 일반적인 PE 파일 포맷에서 section명은 큰 의미를 두지 않는 경우가 많다. 그러나, 악성코드의 제작자가 사용하는 난독화 방법들의 특징을 감안하면 통계적 분석을 통해 유의미한 결과를 산출 할 수 있다. Table 3는 샘플링된 정상 파일과 악성코드에서의 section 발생빈도를 나타낸다. 정상 파일의 경우에는 정해진 PE 포맷에서 빈번히 사용하는 section(.text, .data, .rsrc)을 일반적으로 준용하지만, 악성코드의 경우에는 PE 포맷에서 사용하는 section 뿐만 아니라 기존의 포맷을 따르지 않는 임의의 만들어진 section(CODE, BSS, DATA)을 사용하는 것을 알 수 있다. Table 3은 정상파일과 악성코드 전반이 아니라, sampling된 데이터에서 산출한 데이터지만 정상파일과 악성코드에서 사용빈도가 현저히 차이가 나는 것을 확인할 수 있다.

Table 3. Frequency of Major Sections

Section	Normal file	Malware
.text	100%	80%
.data	100%	20%
.rdata	-	70%
.rsrc	100%	53.3%
.reloc	-	23.3%
.tdata	-	10%
.tls, CODE, DATA, BSS etc	-	20%

3.1.2 Naive Bayes 기반 Section 분석 모델

악성코드와 정상파일에서의 이러한 특징을 이용하여 각 section 별로 가중치를 산정 후 분석대상의 section 구성 비율을 중심으로 악성여부를 판단할 수 있다. Table 4는 정상 파일과 악성코드[28] 간 각각의 section을 전체 section에 대한 비율로 나타낸 가중치 표이다.

Table 4. Bayesian-based Probability Major Section

Section	Normal file	Malware
.text	0.333	0.198
.data	0.333	0.165
.rdata	0.000	0.173
.bss	0.000	0.000
.idata	0.000	0.049
.didat	0.000	0.000
.edata	0.000	0.000
.rsrc	0.333	0.132
.reloc	0.000	0.057
.tls	0.000	0.049
.crt	0.000	0.000
.sdata	0.000	0.000
.tdata	0.000	0.024
CODE	0.000	0.049
DATA	0.000	0.049
BSS	0.000	0.049

Bayesian 확률은 두 확률 변수의 사전 확률과 사후 확률 사이의 관계를 나타내는 정리로 Table 4.의 각 feature들은 단순히 PE 파일 정보 영역을 구분하기 위한 section들의 이름이고 따라서 상호 의존적이지 않으므로 그 관계가 서로 독립이라는 점을 이용하면 다음과 같은 naive bayes 모델을 만들 수 있다.

- C_k : 클래스 k 혹은 Label
- x : 분석할 파일의 N개 특성 벡터

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)} \quad (1)$$

수식 (1)은 naive bayes 분류모델로 기계 학습 분야에서 확률 분류기로 널리 사용되고 있는데 Table 4의 가중치 테이블은 label 별로 section이라는 feature가 서로 독립적이라는 점에서 상호 독립적 확률 변수간의 사전 확률과 사후 확률 사이의 관계를 나타내는 bayesian 정리를 통하여 확률론적으로 유의미하게 사용될 수 있다. 이것을 naive bayes의 확률 모델로 사용하고 분석대상 파일을 입력으로 사용하게 되면 label 별 유사도 결과가 나오게 되고 유사도가 더 큰 값으로 최종 label로 결정하게 된다. 하지만 naive bayes는 특정 feature의 값이 존재하지 않는 경우, 전체 계산값이 0이 되는

문제가 있기 때문에, 분자에 하이퍼 파라미터 값을 더해 주는 laplace smoothing을 적용하게 된다. 하이퍼 파라미터 선택에는 알고리즘마다 여러 가지 수치가 적용될 수 있지만 구현 환경에서의 naive bayes 하이퍼 파라미터는 보통 1을 사용하고 본 논문에서도 이를 이용한다.

- V : 중복을 제거한 벡터의 개수

$$p(C_k|x) = \frac{p(C_k)p(x|C_k)+1}{p(x)+V} \quad (2)$$

이 모델을 악성코드 분석에 응용하면 아래와 같이 적용할 수 있다. 예를들어, 정상파일인 cscript의 경우, {.text, .data, .rsrc}의 Section을 사용하는데 이것을 naive bayes 적용하는 과정은 다음과 같다.

- x : 분석 대상의 section 분포,
  { .text, .data, .rsrc }

- C_k : { Normal, Malware } k = 1, 2

x = {.text, .data, .rsrc} 의 section 구성을 가진 입력 값 cscript가 정상파일일 확률은 아래와 같다.

$$P(Normal|x) = \frac{P(x|Normal) * P(Normal)}{P(x)} \quad (3)$$

x = {.text, .data, .rsrc} 의 section 구성을 가진 입력 값 X의 악성코드일 확률은 아래와 같다.

$$P(Malware|x) = \frac{P(x|Malware) * P(Malware)}{P(x)} \quad (4)$$

Table 5. Malicious Code Analysis by Section Bayesian-based

$NB_{x,c,v}$: Section name x, Label c, Section name vector v에 대한 (2)의 결과 값
calculate $NB_{x,c,v}$
if $NB_{x,c=normal,v} > NB_{x,c=malware,v}$
return Normal
else if $NB_{x,c=normal,v} < NB_{x,c=malware,v}$
return Malware
else
return Half

지금까지 분석된 내용을 바탕으로 악성여부를 판단하는 최종 절차는 Table 5과 같이 label과 feature, 벡터를 계산하고 naive bayes 알고리즘을 이용하여 분석 대상 파일의 정상과 악성 유무를 판단하는 프로세스로 흘러가게 된다.

3.2 DLL 기반 악성코드 분석 방안

3.2.1 DLL

PE파일 포맷의 .idata section에는 import 되어지는 DLL(Dynamic Link Library)의 정보가 있다. PE 파일들은 DLL에서 지원하는 함수들을 사용하기 위해서 DLL을 import하게 된다. Table 6은 일반적으로 PE 파일에서 자주 import되어 사용되는 DLL 파일목록을 나타낸다.

Section feature와 마찬가지로 DLL의 경우에도 PE 파일에 어떤 DLL이 import 되는지는 큰 의미가 없는 경우가 많다. 그러나, 일반적으로 악성코드는 악성행위를 하기 위해서 레지스트리를 접근하는 DLL, 시스템 파일 변경을 위한 DLL, 네트워크 행위에 필요한 DLL 등을 사용하는 경향이 있다. 따라서, 이들을 통계적으로 분석하면 유의미한 결과를 얻을 수 있다. Fig. 3.는 정상 파일과 악성코드[28] 간의 DLL별 import 되는 빈도 차이를 보여준다. 정상 파일의 경우에 ws2_32.dll과 shlwapi.dll을 제외한 DLL들을 악성코드보다 자주 import하는 경향을 알 수 있다. 이렇게 통계학적으로 유의미한 수치를 나타내는 DLL들은 좀 더 큰 가중치 차이로 계산될 것 이고 이는 4장에서 label을 판별하는데 핵심적인 feature로 작용될 것이다.

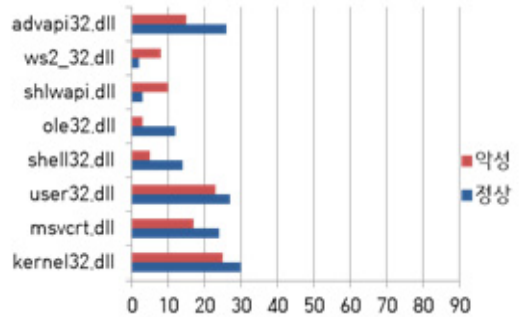

Fig. 3. Example Frequency of DLL Calls for Normal and Malicious Code

Table 6. Frequently Used TOP 15 DLL files

No	DLLs	Frequency	Description
1.	kernel32.dll	3,467,435	Windows NT BASE API Client DLL
2.	user32.dll	912,115	USER API Client DLL
3.	advapi32.dll	441,914	Advanced Win32 application programming interfaces
4.	gdi32.dll	201,687	GDI Client DLL, involved in graphical displays.
5.	wininet.dll	84,100	Internet Extensions for Win32
6.	comctl32.dll	72,010	User Experience Controls Library
7.	shell32.dll	55,169	Windows Shell Common DLL
8.	wsock32.dll	49.133	Windows Socket 32-Bit DLL
9.	oleaut32.dll	49,060	OLE 2 - 32 automation
10.	msvbvm50.dll	44,851	Visual Basic Virtual Machine
11.	ole32.dll	43,531	32-bit OLE 2.0 component
12.	shlwapi.dll	38,718	Library for UNC and URL Paths, Registry Entries
13.	ws2_32.dll	22,486	Windows Socket 2.0 32-Bit DLL
14.	ntdll.dll	18,570	Win32 NTDLL core component
15.	urlmon.dll	16,117	OLE32 Extensions for Win32

3.2.2 Naive Bayes 기반 DLL 분석 모델

위에 기술한 naive bayes를 기반으로 정상파일과 악성코드의 각 feature들에 대해 제안 모델에 적용하여 section DLL feature의 결과에 따라 탐지/미탐으로 결과 값을 도출할 수 있다. 세부 접근방식은 3.1과 유사하므로, 자세한 로직은 생략하고 최종적인 분석 알고리즘은 Table 7과 같다.

Table 7. Malicious Code Analysis by DLL Bayesian-based

$NB_{x,c,v}$: DLL x, Label c, DLL vector v에 대한 (2)의 결과 값
calculate $NB_{x,c,v}$
if $NB_{x,c=normal,v} > NB_{x,c=malware,v}$
  return Normal
else if $NB_{x,c=normal,v} <$
  $NB_{x,c=malware,v}$
  return Malware
else
  return Half

IV. 실험 결과

논문에서 제안한 모델을 구현한 환경은 다음과 같다. 2.93GHz CPU, 1GB ram과 windows 7 환경에서 구축하였다. 또, 정상파일은 Windows xp 샘플을 사용하였으며 악성코드는 API-base의 샘플을 사용[28]하였다. 본 논문에서 제안한 알고리즘으로 정상파일과 악성코드를 판별한 결과를 설명한다. 먼저, Table 8은 section에 기반하여 악성코드를 탐지한 결과 예시를 나타낸다. 악성코드의 경우 정상파일에서 사용하는 section format을 사용하는 경우에는 악성으로 판단하는 가중치가 낮아지기 때문에 `탐지률이 낮아지는 경향이 나타난다. 이를 기준으로 최종적인 악성코드 탐지율과 오탐율은 Table 9와 같다. 정상파일은 100% 정확히 탐지하였으나, 악성

Table 8. Section-based Malicious Code Probability Detection Results

Sample	Normal similarity Probability	Malware similarity Probability	Result
cscript	0.0125	0.0017	Normal
migpwd	0.0125	0.0017	Normal
dfrgfat	0.0125	0.0017	Normal
mnmsrvc	0.0125	0.0017	Normal
dplaysvr	0.0125	0.0017	Normal
04bf52...	0.0125	0.0017	Normal
6d5871...	0.0427	0.0022	Normal
8a3e42...	9.724E-16	2.026E-10	Malware
9df903...	0.0004	0.0022	Malware
22ee7a...	9.724E-16	2.026E-10	Malware

Table 9. Section-based Malware Detection Rate, False Positives

Division	Detect as normal	Detect as Malware	Total(%)
Normal	100%	0%	100%
Malware	20%	80%	100%

코드의 경우 20%의 오탐지율을 기록하였다. 이는 범용적인 특징인 section을 이용한 결과인 것으로 판단된다.

Table 10은 DLL 기반으로 악성코드를 탐지한 분석결과를 나타낸다. 여기서 산출된 수치는 정상일 확률과 악성일 확률을 상대적으로 비교하는 것이므로, 각 확률의 수치 자체에 대한 해석보다는 상대적인 크기로 의미를 부여할 수 있다.

Table 10. Example of DLL-based Malware Detection Result

Name	Normal probability	Malware probability	Result
cscript	2.57287E-11	2.27645E-12	Normal
migpwd	1.34581E-11	4.26835E-13	Normal
eventvwr	8.92056E-14	2.21612E-16	Normal
mnmsrvc	3.08744E-11	6.82935E-12	Normal
dplaysvr	6.4544E-7	2.12102E-7	Normal
.			
.			
eventvwr	0.0006	0.0008	malware
route	6.22239E-11	1.10123E-10	Malware

이를 기준으로 최종적인 악성코드 탐지율과 오탐율은 Table 11와 같다.

Table 11. DLL-based Malware Detection Rate, False Positives

Division	Detect as Normal	Detect as Malware	Total(%)
Normal	84%	16%	100%
Malware	0%	100%	100%

DLL 기반의 분석결과, 정상파일의 경우 84%의 탐지율을 기록했으며, 악성코드의 경우 탐지율 100%를 기록하였다. 이렇듯 정상 파일의 section과 DLL 탐지율에서 section탐지율이 더 높은 수치로 나왔다. 악성 파일에서는 section과 DLL의 탐지율은 DLL탐지율이 더 높은 수치로 보인다. 즉, 결과는 section 기반 악성코드 탐지율과 오탐율과는 어느 정도 반대의 성향을 띄었다. 또한 탐지율 84%로 탐지의 한계가 있었다. 따라서, 정상파일의 오탐율을 낮추고 악성코드 탐지율을 높이기 위해 2개 탐지방안을 통합하여, 2개의 분석결과 중 어느 한쪽이라도 악성으로 결과가 나오면 악성으로 판단하는 알고리즘으로 통합하였다. Table 12, Table 13은 통합했을때의 악성코드 탐지율, 오탐지율을 나타낸다.

Table 12. Section name, DLL Feature Integration Example

Name	Normal probability	Malware probability	Result
cscript	2.57287E-11	2.27645E-12	Normal
migpwd	1.34581E-11	4.26835E-13	Normal
eventvwr	8.92056E-14	2.21612E-16	Normal
mnmsrvc	3.08744E-11	6.82935E-12	Normal
dplaysvr	6.4544E-7	2.12102E-7	Normal
.			
.			
eventvwr	0.0191	0.0029	Normal
route	0.0185	0.0021	Normal

Table 13. Section, DLL's Integrated Detection Rate, False Positives

Division	Detect as Normal	Detect as Malware	Total(%)
Normal	90%	10%	100%
Malware	100%	0%	100%

Table 13의 분석결과, 악성코드는 분석대상 전체에 대해 100% 탐지율을 나타냈고, 정상파일인 경우 10%의 오탐율을 나타나면서 section과 DLL을 따로 분석한 값으로 분류하는 것보다 section과 DLL을 통합하는 알고리즘으로 채택하여 기존의 분석률을 높이는 효과를 보였다.

기존의 API 호출 기반 악성행위 탐지에 관한 연구[29] 등에 따르면 복잡한 악성코드 자동분석 시스템을 구성함에 비해 샘플 4,000개에 대하여 정상 정탐율 83.4%, 악성 정탐율 76.7%의 결과를 도출했는데, 이와 같이, 본 논문에서 제안한 알고리즘은 API에 기반하여 trojan 범주의 악성코드에서 오류가 발생하거나 난독화된 악성코드의 의미 없는 코드를 제거하고 해독하는 과정들이 없고, 패턴을 사용하지 않으면서도 유의미한 분석결과를 산출하였다. 실환경에서 이러한 방법만으로 운영할 수는 없지만, pattern 기반 기술, 기존 PE-header, CFG, sandbox 기반 동적분석 기술과 연계하여 동작할 경우, 유용하게 동작할 것으로 예상된다.

V. 결 론

본 논문에서는 하루에도 엄청난 양이 양산되는 변종 악성코드에 대응하기 위하여 악성코드의 정적 분

석 정보, PE 파일 포맷의 section과 .idata section에 있는 DLL import 정보가 악성코드 제작자의 특징과 통계학적으로 연관성 있음에 착안하여 feature로 선정하고 그것을 기반으로 악성코드를 탐지하는 메커니즘을 제안 및 샘플 데이터에 대한 결과값을 제시하였다. 제안 모델에서는 section과 DLL이 통계학적으로 일정 수준 차이가 있다는 것을 보였고 확률 분류기인 naive bayes에 적용하기 위하여 각 feature별 bayesian 가중치를 산정하였다. 본 메커니즘의 경우에 머신 러닝 지도학습 환경에서 주로 사용되는 navie bayes 분류를 채택함으로서 변종 malware에 대한 파라미터를 추정은 동적 분석 기법에 비하여 빠르고 구조가 간단하다. 하지만 논문에서 제시한 training set이 상대적으로 적었다. 차후 더 많은 training set을 분석한 후, 결과를 나타낸다면 더 정확한 값으로 나올 것이라 판단된다. 향후 본 논문에서 제안하는 feature 외에 다른 feature들과의 교차 검증을 통하여 성능 향상이 가능하고 이러한 메커니즘을 기존의 악성코드 탐지 기법들에 hybrid로 적용 및 운용하여 악성코드 분석 및 변종그룹분류에 많은 기여를 할 수 있을 것으로 기대한다.

References

[1] Syarif Yusirwan S, Yudi Prayudi and Imam Riadi, "Implementation of Malware Analysis using Static and Dynamic Analysis Method," International Journal of Computer Applications. Volume 117 - No.6, pp11-15, May, 2015.

[2] A. Moser, C. Kruegel, and E. Kirda, "Limits of Static Analysis for Malware Detection," IEEE, DOI10.1109/ACSAC. pp421-430, 2007.21

[3] Hiran V. Nath and Babu M. Mehtre, "Static Malware Analysis Using Machine Learning Methods," G. Mart'ınez P'erez et al. (Eds.): SNDS 2014, CCIS 420, pp. 440 - 450, 2014.

[4] Sung-tae Yu and Soo-hyun Oh, "Malware Analysis Mechanismusing Word Cloud based on API Statistics," Vol. 16, No. 10 pp. 7211-7218, 2015.

[5] Han-young Noh, "Complexity-based Packed Executable Classification with High Accuracy," School of Engineering, pp1-35, 2009.

[6] Hee-jun kwon, Sun-woo Kim and Eul-gyu IM, "An Malware Classification System using Multi N-gram," Security Engineering Journal, Volume 9, Issue 6 pp.531-542, 12.2012.

[7] Youngjoon Ki, Eunjin Kim and Huy Kang Kim, "A NoveApproach to Detect Malware based on API Call SequenceAnalysis," International Journal of Distributed Sensor Network, pp.9, 2015.

[8] Scott Treadwell and Mian Zhou, "A Heuristic Approach forDetection of Obfuscated Malware," 2009 IEEE. ISI ,Richardson, TX, USA, pp.291-299, June 8-11, 2009.

[9] R. de Janeiro, "Mohaisen A, Alrawi O (2013) Unveiling zeus: automated classification of malware samples. In:22nd international conference on worldwide webcompanion," ACM New York, NY, USA. pp.829-832, May 2013.

[10] Lee, Taejin, and Jin Kwak. "Effective and Reliable Malware Group Classification for a Massive Malware Environment," International Journal of Distributed Sensor Networks 2016 (2016).

[11] A. Sami, B. Yadegari, H. Rahimi, N. Peiravian, S. Hashemi, and A. Hamze, "Malware detection based on mining API calls," in Proceedings of the 25th Annual ACM Symposium on Applied Computing (SAC '10), pp. 1020 - 1025,ACM,March 2010.

[12] M. K. Shankarapani, S. Ramamoorthy, R. S. Movva, and S. Mukkamala, "Malware detection using assembly and API call sequences," Journal in Computer Virology, vol. 7, no. 2, pp. 107 - 119, 2011.

[13] M. Christodorescu, J. Kinder, S. Jha, S. Katzenbeisser, and H. Veith, "Malware

normalization," EPFL-REPORT 167534, University ofWisconsin, Madison,Wis, USA, 2005.
〔14〕 N. Idika and A. P. Mathur, "A survey of malware detection techniques," 〔Predoctoral Fellowship, and Purdue Doctoral Fellowship〕, Purdue University, February 2007.
〔15〕 Y. Ye, D. Wang, T. Li, and D. Ye, "IMDS: intelligent malware detection system," in Proceedings of the 13th
〔16〕 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1043 - 1047,ACM,August 2007.
〔17〕 Wu. L, et. al , "Behavior-based malware analysis and detection," 2011 first international workshop on complexity and data mining (IWCDM). IEEE, New York, pp 39 - 42, 2011.
〔18〕 Pratiksha N and Deepti V, "Malware detection using API function frequency with ensemble based classifier." Security in computing and communications. Springer, Berlin 2013.
〔19〕 Kyung-moon Woo and Chong-kwon Kim, "Internet Worm Propagation Modeling using a Statistical Method," The Journal of the Korean Institute of Communication Sciences'12 - 03 Vol.37B No.03. pp.212-218, March 2012.
〔20〕 AV-TEST, Total Malware Statistics, https://www.av-test.org/
〔21〕 Mandiant, Cyber Threat Intelligence Report
〔22〕 Kaspersky, https://usa.kaspersky.com/
〔23〕 CTA(Cyber Threat Alliance), http://www.cyberthreatalliance.org/
〔24〕 CSC(Cyber Security Coalition), http://www.cybersecuritycoalition.be/
〔25〕 ACM workshop on Security and artificial intelligence (AISec)
〔26〕 Malware Normalization
〔27〕 https://infoscience.epfl.ch/record/167534/files/malwarenorm
〔28〕 http://ocslab.hksecurity.net/apimds-dataset
〔29〕 Dong-wook Hwang, Hong-koo Kang, Tae-jin Lee, "A Study on Malicious Activity Automatic Detection based on API calls," Proceedings of Symposium of the Korean Institute of communications and Information Sciences, 2016.

〈저자소개〉

황 준 호 (Jun-ho Hwang) 학생회원
2012년 3월~현재: 호서대학교 정보보호학과
〈관심분야〉 머신러닝, 정보보호, 통계학

황 선 빈 (Seon-Bin Hwang) 학생회원
2012년 3월~현재: 호서대학교 정보보호학과
〈관심분야〉 정보보호, 네트워크;

김 호 경 (Ho-gyeong Kim) 학생회원
2012년 3월~현재: 호서대학교 정보보호학과
〈관심분야〉 시스템 보안, 악성코드 분석

하 지 희 (Ji-hee Ha) 학생회원
2014년 3월~현재: 호서대학교 정보보호학과
〈관심분야〉 정보보호, 암호학

이 태 진 (Taejin Lee) 종신회원
2003년 1월~2017년 2월: 한국인터넷진흥원 팀장
2017년 3월~현재: 호서대학교 정보보호학과
〈관심분야〉 시스템 보안, 악성코드 분석, 침해사고 대응