한국컴퓨터정보학회 동계학술대회 논문집 제31권 제1호 (2023. 1)

Opcode와 IAT를 활용한 PE 파일 악성코드 탐지

이정훈[O], 강아름[*]
[O]배재대학교 대학생,
[*]배재대학교 교수

PE file malware detection using opcode and IAT

JeongHun Lee[O], Ah Reum Kang[*]
[O]Student, Dept. of Information Security, Pai Chai University,
[*]Professor, Dept. of Information Security, Pai Chai University

● 요 약 ●

코로나 팬데믹 사태로 인해 업무환경이 재택근무를 하는 환경으로 바뀌고 악성코드의 변종 또한 빠르게 발전하고 있다. 악성코드를 분석하고 백신 프로그램을 만들면 새로운 변종 악성코드가 생기고 변종에 대한 백신프로그램이 만들어 질 때까지 변종된 악성코드는 사용자에게 위협이 된다. 본 연구에서는 머신러닝 알고리즘을 사용하여 악성파일 여부를 예측하는 방법을 제시하였다. 일반적인 악성코드의 구조를 갖는 Portable Executable 구조 파일을 파이썬의 LIEF 라이브러리를 사용하여 Certificate, Imports, Opcode 등 3가지 feature에 대해 정적분석을 하였다. 학습 데이터로는 정상파일 320개와 악성파일 530개를 사용하였다. Certificate는 hasSignature(디지털 서명정보), isValidcertificate(디지털 서명의 유효성), isNotExpired(인증서의 유효성)의 feature set을 사용하고, Imports는 Import Address Table의 function 빈도수를 비교하여 feature set을 구축하였다. Opcode는 tri-gram으로 추출하여 빈도수를 비교하여 feature set을 구축하였다. 테스트 데이터로는 정상파일 360개 악성파일 610개를 사용하였으며 Feature set을 사용하여 random forest, decision tree, bagging, adaboost 등 4가지 머신러닝 알고리즘을 대상으로 성능을 비교하였고, bagging 알고리즘에서 약 0.98의 정확도를 보였다.

키워드: 머신러닝(machine learning), portable executable, 배깅 알고리즘(bagging algorithm)

I. Introduction

코로나 팬데믹 사태로 인해 비대면 플랫폼이 증가하고 재택업무를 하는 등 많은 환경 변화가 일어났다. 많은 환경이 변화하면서 악성코드도 변종이 생기고 위협이 증가하였다. 악성코드는 사용자의 PC를 감염시키고 감염된 PC로부터 개인정보를 노출시키거나 파일을 암호화하여 PC 사용을 중지시키는 등 악성코드의 확산으로 인한 피해는 증가하고 있다.

국내외 안티바이러스 업체들이 제공하는 백신 프로그램만으로는 변종 바이러스의 방어에 한계가 있다. 악성코드의 변종을 안티바이러스 업체에서 분석하고 새로운 백신을 만들 때까지 사용자들은 변종 악성코드에 노출되기 쉽고, 그에 대응하는 백신이 나오면 또 다른 악성코드 변종이 생성된다. 새로운 악성코드를 정밀하게 분석하는데 많은 인력과 시간과 비용이 필요하기 때문에 많은 악성코드를 빠르게 분석하고 탐지할 수 있는 모델이 필요하다.

최근 머신러닝이나 딥러닝과 같은 인공지능 기술이 많은 관심을 받고 있으며 여러 연구에서 악성코드를 분류하고 탐지하는 데 딥러닝 기술의 적용을 시도하고 있다. 본 연구에서도 Windows 운영체제에서 사용되는 Portable Executable(PE) 포맷의 실행파일 구조를 분석하고 머신러닝 기술을 접목하여 실행형 파일의 악성여부를 예측하는 모델을 연구하였다.

본 연구에서는 일반적인 악성코드의 구조를 갖는 PE 파일을 대상으로 연구를 진행하였다. PE 구조의 파일에서 파이썬의 Library to Instrument Executable Formats(LIEF) 라이브러리를 사용하여 Certificate, Operation Code(opcode), Imports 등 3가지 feature를 추출하였고, 정상파일 및 악성파일을 대상으로 Opcode 또는 Import Address Table(IAT)의 함수들의 빈도수를 비교하여 Certificate 3개, Imports 40개, Opcode 36개의 feature set을 구축하였다. 파이썬

코드를 활용해 feature set을 대상으로 결과를 comma-separated values(CSV) 파일로 추출하고, 추출된 CSV 파일을 대상으로 random forest, decision tree, adaboose, bagging 머신러닝 알고리즘을 사용해 악성파일 여부를 예측하는 모델을 제시하였다.

II. Preliminaries

1. Related works

PE 파일 실행형 악성코드(Download, Dropper, Miner, Virus, Backdoor)의 공격을 예방하고자 머신러닝과 딥러닝을 활용한 악성코드 탐지 연구가 지속적으로 진행되고 있다. 선행연구에서는 어셈블리 코드로부터 opcode 시퀀스를 추출하고 one-gram이나 tri-gram 시퀀스를 추출하는 N-gram 추출 모듈을 활용한 opcode 기반 악성코드 탐지와 악성코드의 이진 파일을 이미지로 시각화하여 이미지 기반 딥러닝을 하는 binary image를 이용한 연구가 진행되었다.

Opcode와 API를 사용한 malware 탐지에 관한 연구를 정리하였다. SeokMin Ko 외[1]는 바이너리 바이트 기반으로 생성한 이미지가 아닌 바이너리의 명령어 빈도수를 기반으로 이미지를 생성하고 이를 Convolutional Neural Network(CNN)으로 분석한 모델을 설계하고 제시였고 91%의 정확도를 보였다.

Tae-Hyun Ahn 외[2]는 opcode와 Windows API call을 추출하고 특징 벡터를 구성해 Naive Bayes와 K-nearest neighbor 분류기 알고리즘을 사용하는 모델을 제시하였다. 기존의 opcode 또는 Windows API 호출 중 하나만 사용하는 방법보다 제안한 방법이 멀웨어 탐지에서 높은 95.21%의 정확도를 보였다.

Gye-Hyeok Lee외 [4]는 '.text Section' Opcode와 실제 사용하는 Native API의 빈도수를 추출하고 K-means clustering 알고리즘, 코사인 유사도, 피어슨 상관계수를 이용하여 선정한 특징정보들 사이의 연관성을 분석하였다. 또한 웜과 Cerber형 랜섬웨어를 분류, 탐지하는 실험을 통해, 선정한 특징정보가 특정 랜섬웨어(Cerber)를 탐지하는데 특화된 정보임을 검증하였다. 제안된 모델은 93.3%의 탐지율을 보였다.

Hee Yeon Kim 외[5]는 신종 악성코드의 등장은 기존 시그니처 악성코드 탐지 기법들을 무력화시키며 여러 분석 방지 보호기법들을 활용하여 분석이 어렵다고 한다. Hee Yeon Kim의 연구에서는 악성코드 자체의 특성이 아닌, 악성코드에 적용될 수 있는 패커의 특성을 활용하여, 단시간 내에 악성코드에 적용된 패커의 분석 방지 보호기법을 탐지하고 분류해낼 머신러닝 모델을 구축했다. N-gram opcode를 추출하여 Term Frequency-Inverse Document Frequency(TF-IDF)를 활용한 모델을 제시하였고 악성코드 패킹에 많이 사용되는 Themada와 VMProtect 두 가지 데이터셋을 구축하고 6개의 머신러닝 모델로 실험하였다. Themida와 VMProtect 각각 81.25%, 95.65%의 정확도를 보였다.

III. The Proposed Scheme

1. PE structure analysis

PE 파일은 크게 DOS header, NT header, section header, section 으로 구성이 되어있다. DOS 헤더는 PE 파일의 시작 부분이며 e_magic은 DOS의 signature로 MZ(4D 5A)값을 가진다. NT header는 signature, file header, optional header로 이루어져 있으며 signature는 'PE..' (50 45 00 00) 값을 가지며 PE 파일임을 나타낸다. Section header는 .text, .idata, .data, rsrc 등의 섹션이 존재한다. Section 유형으로는 code, data, import API, export API, resource, 재배치 정보, TLS, debugging 등이 존재 한다.

Opcode란 섹션유형 중 code에 위치하며 프로그램을 실행하기 위한 코드를 담고 있는 섹션이다. Opcode는 함수 연산, 자료 전달 연산, 제어 연산, 입출력 연산으로 나눌 수 있다.

2. PE feature extraction

본 연구에서 정상파일은 네이버 소프트웨어 자료실에 공개되어 있는 실행파일과 Windows 10 pro 운영체제를 사용하는 PC의 기본 실행파일을 샘플로 사용하였고, 악성파일은 국내 백신사로부터 제공받았다. Feature 추출을 위해 정상파일 320개, 악성파일 530개를 샘플로 사용하였다.

LIEF 라이브러리를 사용하여 윈도우 PE 포맷 실행파일의 데이터를 불러온 후 Opcode, Imports, Certificate 등 3가지 feature set를 추출하였다. Opcode feature는 tri-gram과 four-gram, 두 가지 n-gram으로 구성하여 비교하였으며 tri-gram에서 더 좋은 성능을 나타냈다.

Imports feature set은 IAT에서 명시된 function 중 정상파일과 악성파일에서 차이를 많이 보인 function을 모아놓은 세트를 말한다. LIEF 라이브러리로는 함수 호출 빈도수가 아닌 존재 유무만 알 수 있기 때문에 파이썬 코드를 사용해 정상 320개, 악성 530개의 파일 중 몇 개의 파일에 function 존재하는지 수치를 CSV 파일로 추출하였다. 각각 추출된 파일을 엑셀 작업을 통해 공통된 function을 따로 저장하고 clean, malware 샘플 중 몇 개의 샘플에 존재하는지 수치를 비교하였다. 공통으로 확인된 function 중 악성파일과 정상파일의 수치에 차이를 많이 보인 상위 22개 function, 공통되지 않고 악성파일에서만 발견된 상위 3개 function, 정상파일에서만 확인된 상위 2개 function, Uniform Resource Locator(URL) 관련 13개 function 등, 총 40개의 feature를 정리했으며 Table 1은 Imports feature set을 정리한 표이다. 대상 파일에 해당 function이 존재하면 1, 존재하지 않으면 0로 추출하였다.

Opcode feature set은 opcode를 tri-gram으로 추출한 후, Imports feature set과 같은 방법으로 악성파일과 정상파일에서 공통으로 확인된 opcode 상위 18개, 악성파일에서만 확인된 상위 13개, 정상파일에서만 발견된 상위 5개, 총 36개의 tri-gram Opcode feature set을 정의하고, Table 2에 정리하였다. 대상 파일에서 해당 tri-gram이 발견되면 패턴 등장 횟수를 추출하였다.

Certificate feature set은 디지털 서명 정보를 사용하며

hasSignature, isValidcertificate, isNotExpired 3가지의 값으로 정의하였다. hasSignature는 디지털 서명 여부를 의미하며 서명이 되어 있으면 1, 되어있지 않으면 0을 태깅하였다. isValidcertificate는 디지털 서명의 유효성을 의미하며 서명이 유효하면 1, 유효하지 않다면 0을 태깅하였다. isNotExpired는 인증서 유효기간의 만료 여부를 의미하며 유효기간이 유효하면 1, 만료되었다면 0을 태깅하였다. Table 3은 추출한 결과 중 악성파일 1개와 정상파일 1개의 서명 정보 예시를 표로 나타내었다.

Table 1. Imports features

Imports features
'RtlLookupFunctionEntry', 'RtlCaptureContext', 'RtlVirtualUnwind', 'GetModuleHandleW', '__C_specific_handler', 'memset', 'memcpy', 'FormatMessageW', 'QueryPerformanceCounter', '_initterm', 'GetSystemTimeAsFileTime', 'memmove', 'SetLastError', 'GetCurrentProcessId', 'GetModuleHandleExW', 'GetCurrentProcess', 'WaitForSingleObjectEx', 'GetCurrentThreadId', 'SetUnhandledExceptionFilter', 'IsDebuggerPresent', 'OutputDebugStringW', '_CxxThrowException', 'TerminateProcess', 'GetModuleHandleA', 'GetLastError', 'UnhandledExceptionFilter', 'LocalFree', 'OpenSemaphoreW', 'memcmp', 'ReleaseMutex', 'GetProcAddress', 'CloseHandle', 'ExitProcess', 'WriteFile', 'LoadLibraryA', 'GetModuleFileNameA', 'GetTickCount', 'ReadFile', 'FreeLibrary', 'RegCloseKey'

Table 2. Opcode features

Opcode features
'MOV, MOV, MOV', 'PUSH, PUSH, PUSH', 'ADD, XOR, ADD', 'ADD, ADD, XOR', 'ADD, OR, ADD', 'OR, ADD, OR', 'XOR, ADD, ADD', 'POPA, ADD, ADD', 'PUSH, PUSH, CALL', 'MOV, PUSH, PUSH', 'PUSH, MOV, PUSH', 'PUSH, PUSH, MOV', 'PUSH, CALL, MOV', 'MOV, PUSH, MOV', 'INT, INT, INT', 'LEA, PUSH, PUSH', 'PUSH, CALL, ADD', 'PUSH, LEA, PUSH', 'XOR, XOR, XOR', 'SBB, ADD, ADD', 'ADD, XOR, XOR', 'JNP, ADD, ADD', 'ADD, ADD, CMP', 'OR, JNP, ADD', 'ADD, OR, JNP', 'AAA, ADD, ADD', 'XOR, AAA, ADD', 'ADD, XOR, JB', 'JO, XOR, JB', 'JAE, ADD, OR', 'XOR, XOR, ADD', 'DEC, CALL, NOP', 'CALL, NOP, DEC', 'MOV, DEC, CALL', 'INT, INT, DEC', 'NOP, DEC, MOV'

Table 3. Certificate features

Certificate features			
filename	hasSignature	isValidcertificate	isNotExpired
cfeaa7...	0	0	1
consent.exe	1	1	1

3. Result

본 연구는 Certificate 3개, Imports 40개, Opcode 36개 총 79개의 feature를 사용해 결과를 CSV 파일로 추출하고 추출된 CSV 파일을 사용하여 random forest, decision tree, bagging, adaboost 등 4개의 머신러닝 모델을 구현하였다. 전체 데이터셋은 정상파일 680개, 악성파일 1140개이고, 이 중 학습에 정상파일 320개, 악성파일 530개를 사용하였다. 학습에 사용되지 않은 나머지 정상파일 360개, 악성파일 610개를 테스트용으로 사용하였고, 모델별 성능을 비교하였다.

Table 4는 모델별 정확도를 정리한 표이다. Bagging 모델이 가장 높은 정확도를 보였다.

Table 4. model accuracy

model	accuracy
Random Forest	0.97628866
Decision Tree	0.967010309
Bagging	0.98556701
Adaboost	0.977319588

가장 높은 정확도를 보인 Bagging model을 저장 후 학습에 쓰이지 않은 정상파일 360개와 학습에 쓰이지 않은 악성파일 610개를 대상으로 악성여부 예측 테스트를 진행하였다. Table 5는 악성파일 610개와 정상파일 360개를 대상으로 테스트한 결과를 confusion matrix로 정리하였다. Table 6은 분류 성능평가지표를 정리한 것이며, 정확도는 약 0.985의 성능을 보였다.

Table 5. Model Confusion Matrix

Bagging Confusion Matrix		actual	
		malware	clean
predicted	malware	604	8
	clean	6	352

Table 6. Model Classification Evaluation Metrics

Bagging	
accuracy	0.985567
recall	0.990164
precision	0.986928
False Positive Rate(FPR)	0.022222

IV. Conclusions

본 연구에서는 파일을 직접 실행하지 않는 정적분석 연구로 진행되었다. PE 파일의 바이너리 데이터에서 Imports, Opcode, Certificate 등 3가지 feature set을 추출하고, 머신러닝 기반의 PE 악성파일 예측 모델을 제안하였다.

정상파일 320개와 악성파일 530개로부터 opcode와 IAT의 함수들을 모두 추출하고, opcode와 function의 빈도수를 분석하여 feature를 추출하였다. Random forest, bagging, decision tree, adaboost 등의

4가지 머신러닝 모델을 구축하였다. 테스트에 사용된 샘플은 학습에 사용되지 않은 샘플로 진행하였으며, 정상파일 360개와 악성파일 610개 총 970개의 파일로 테스트를 진행하였다.

1차 테스트에서는 4가지 머신러닝 모델이 각각 97.6%, 98.5%, 96.7%, 97.7%의 정확도를 보였다. Feature set 수정 후, 2차 테스트에서는 각각 94.9%, 95.7%, 93.1%, 95.0%의 성능을 보였다. Bagging 알고리즘이 가장 높은 정확도를 보였다.

ACKNOWLEDGEMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2020R1I1A1A01061146).

REFERENCES

[1] SeokMin Ko, JaeHyeok Yang, WonJun Choi, TaeGuen Kim, “CNN-Based Malware Detection Using Opcode Frequency-Based Image”

[2] Tae-Hyun Ahn, Sang-Jin Oh, Young-Man Kwon “Malware Detection Method using Opcode and windows API Calls”

[3] Kwang-Yun PARK, Soo-Jin LEE “Bidirectional LSTM based light-weighted malware detection mode using Windows PE format binary dat”

[4] Gye-Hyeok Lee, Min-Chae Hwang, Dong-Yeop Hyun, Young-In Ku, Dong-Young Yoo, “A Study on the Cerber-Type Ransomware Detection Model Using Opcode and API Frequency and Correlation Coefficient“

[5] Hee Yeon Kim, Dong Hoon Lee “A Study on Machine Learning Based Anti-Analysis Technique Detection Using N-gram Opcode”

[6] DaeYoub Kim “Generating Call Graph for PE file”

[7] Jin-Young Cho, Eun-Gi Ko, Hye-Bin Yoo, Mi-Ri Cho, Chang-Jin Seo “A Study on Malware Detecion System Using Static Analysis and Stacking