Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.32, NO.4, Aug. 2022 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2022.32.4.629

API 호출 구간 특성 기반 악성코드 탐지 기술

김 동 엽,[1†] 최 상 용[2‡]
[1,2]영남이공대학교 (학생, 교수)

Malware Detection Technology Based on API Call Time Section Characteristics

Dong-Yeob Kim,[1†] Sang-Yong Choi[2‡]
[1,2]Dept. of Cyber Security, Yeungnam University College (Student, Professor)

요 약

최근 사회적 변화와 IC T기술의 발전에 따라 사이버 위협 또한 증가되고 있으며, 사이버위협에 사용되는 악성코드는 분석을 어렵게 하기 위해 분석환경 회피기술, 은닉화, 파일리스 유포 등 더욱 고도화 지능화 되고 있다. 이러한 악성코드를 효과적으로 분석하기 위해 머신러닝 기술이 활용되고 있지만 분류의 정확도를 높이기 위한 많은 노력이 필요하다. 본 논문에서는 머신러닝의 분류성능을 높이기 위해 API호출 구간 특성 기반 악성코드 탐지 기술을 제안한다. 제안하는 기술은 악성코드와 정상 바이너리의 API 호출 순서를 시간을 기준으로 구간으로 분리하여 각 구간별 API의 호출특성과 바이너리의 엔트로피 등의 특성인자를 추출한 후 SVM(Support Vector Mechine) 알고리즘을 이용하여 제안하는 방법이 악성바이너리를 잘 분석할 수 있음을 검증하였다.

ABSTRACT

Cyber threats are also increasing with recent social changes and the development of ICT technology. Malicious codes used in cyber threats are becoming more advanced and intelligent, such as analysis environment avoidance technology, concealment, and fileless distribution, to make analysis difficult. Machine learning technology is being used to effectively analyze these malicious codes, but a lot of effort is needed to increase the accuracy of classification. In this paper, we propose a malicious code detection technology based on API call interval characteristics to improve the classification performance of machine learning. The proposed technology uses API call characteristics for each section and entropy of binary to separate characteristic factors into sections based on the extraction malicious code and API call order of normal binary. It was verified that malicious code can be well analyzed using the support vector machine (SVM) algorithm for the extracted characteristic factors.

Keywords: Malware, Cuckoo Sandbox, Machine Learning, API, Classification

I. 서 론

최근 코로나19와 같은 사회적인 변화는 ICT기술의 발전과 활용성을 증가시켰으며, 이에 따라 ICT시스템에 대한 사이버위협 또한 증대되고 있다. ENISA(The European Union Agency for Cybersecurity)에서 발표한 'Threat Landscape 2021' 보고서에 따르면, Ransomware, Malware 등과 같은 악성코드를 사용하는 사이버 공격이 주요 위협으로 대두되고 있다. 특히 코로나19 팬데믹의 영향으로 재택근무, 원격근무 등이 증가함에 따라 악성코드의 공격은 전 세계적으로 증가한 것으로 분석되고 있다[1]. 향후 4차 산업혁명이 본격화된다면 사이버위협은 더욱 증가할 것으로 예상되며, 악성코

Received(03. 03. 2022), Modified(04. 25. 2022), Accepted(05. 30. 2022)
† 주저자, qweasd5566@ync.ac.kr
‡ 교신저자, spikechoi@ync.ac.kr(Corresponding autor)

드에 의한 피해가 더욱 심각해질 것으로 예측하고 있다.[2][3].

악성코드와 정상 파일을 분류하기 위한 기술로는 실행기반 동적분석 기술과 코드기반 정적분석 기술이 전통적으로 연구되어 왔다. 하지만 정적분석 기술은 코드난독화 등의 기술을 적용함에 따라 악성행위를 추출하는데 한계가 있으며[4], 동적 분석의 경우 분석환경 회피기술, 은닉화, 파일리스 등의 공격 방법으로 인해 분석에 한계가 있다[5], 최근에는 인공지능 등 4차 산업혁명 기반 기술을 활용한 악성코드 식별을 위한 연구가 지속적으로 이루어지고 있으나[6], 분류의 정확도 향상을 위해서는 특성인자 추출 등에 대한 지속적인 연구가 필요한 실정이다.

이러한 이유로 본 논문에서는 악성코드와 정상 파일을 분류하기 위해 머신러닝에서 사용할 수 있는 효과적인 특성인자를 추출할 수 있는 방법을 제안한다. 본 논문에서 제안하는 방법은 악성코드와 정상파일 분류를 위해 API(Application Programming Interface)를 효과적으로 활용할 수 있는 방법을 제안한다. 제안하는 방법은 바이너리에 포함된 API를 추출하여 API의 호출 빈도, 호출시간 등의 특성에 따라 정상과 악성 바이너리를 학습시키고, 코드 엔트로피의 특성을 병합하여 다양한 기계학습 알고리즘을 적용하는 방법으로 제안하는 방법이 악성과 정상 바이너리 분류에 효과적임을 실험을 통해 검증하였다.

본 논문은 2장에서 악성코드 분류를 위한 기존의 연구내용을 분석하고, 3장에서는 제안하는 특성인자 선정 방법에 대한 구체적인 설명을 포함하며, 4장에서 실험을 통해 제안하는 방법의 효과를 검증한다.

II. 관련 연구

2.1 서명 탐지

서명탐지는 전통적으로 악성코드의 특징들을 모아 저장 후 의심스러운 파일과 비교하여 탐지하는 방법이다. 탐지 속도가 매우 빠르고, 이미 알려진 악성코드에 대해 탐지율이 매우 높은 장점이 있는 것으로 알려져 있다. 주로 사용되는 특징으로는 Hash, String, PE Header 등이 있다. 서명 탐지는 기존 악성코드의 변종 또는 새로운 악성코드 탐지에는 한계점이 존재한다[7].

2.2 정적분석

정적분석은 프로그램을 실행시키지 않고 분석하는 방법이다. 실행파일을 소스코드로 변환해 주는 IDA Pro, Ghidra와 같은 디컴파일러를 사용하여 바이너리를 소스코드와 유사하게 변환하여 분석하는 방법으로 기존의 소스코드와 비슷하게 보이기 때문에 세세한 분석이 가능하다는 장점이 있다. 하지만 정적분석을 어렵게 하는 요소들인 Nop 삽입, 패킹, 난독화 등의 기술이 적용되어 있는 경우가 있어 분석에 한계가 있다[8].

2.2.1 정적분석 기반 기계학습 기법을 활용한 악성코드 식별 시스템

이 연구는 바이너리의 Header, Payload 등에서 PE 헤더 57종류의 값을 가공한 94개 특성과 DLL 명과 API 명을 가공한 320개, Entropy를 가공한 256개 총 670개의 특성을 추출한 후 퍼지해시 기반 유사도 분석 알고리즘인 ssdeep, TLSH, DHASH 등을 적용하여 악성 바이너리를 분석한다. 일반적인 해시 알고리즘의 경우 한 글자만 달라져도 해시 값이 완전히 달라지기 때문에 달라지기 때문에 파일의 유사도를 판단하는 것이 거의 불가능하다. 때문에 퍼지해시를 사용하는 유사성 해시 알고리즘을 사용하는 것으로 확인되며, 추출한 특성인자를 활용하여 SVM, DNN, Decision Tree, k-NN 알고리즘에 적용하여 바이너리 분류 성능을 실험하였다[9].

2.2.2 DLL/API 통계적 분석을 통한 Feature 추출 및 ML 기반 악성코드 탐지 기법

PE-Header에 존재하는 IAT(Import Address Table)를 이용하여 DLL 1,622개 API 65,300개의 목록을 작성하였고, 이 중 유의미한 차이를 나타내는 특성을 선별하기 위해서 정상 파일에서의 출현 비율과 악성 파일에서 출현 비율을 가공하여 Range 정책을 선정하고 DNN 모델을 통해 이렇게 선정된 Range의 값을 적절히 조절해가며 최적값을 찾아가는 실험을 진행하였다[10].

2.3 동적분석

동적분석은 정적 분석과는 다르게 프로그램을 실

행시켜 프로그램의 행위를 확인하는 방법이다. 동적 분석의 경우 프로그램을 실행시켜 API, 네트워크, 프로세스, 자원 접근 등의 행위를 확인하고, 이상행위를 찾아내는 분석 방법이다. 프로그램을 실행시켜 행위를 확인하는 방법을 사용하게 됨으로 세세한 코드 분석은 하기 어렵지만 정적분석을 어렵게 하는 요소인 nop 삽입, 패킹, 난독화 등의 기술들을 우회하여 분석을 진행할 수 있다는 장점과 빠르게 분석 가능하다는 장점 또한 가지고 있다. 일반적으로 동적분석에서 특성인자를 추출하기 위해 샌드박스를 사용하며, 대표적인 샌드박스로는 Cuckoo Sandbox[11]가 있다.

2.3.1 API call 단계별 복합분석을 통한 악성코드 탐지

윈도우 환경에서 악성코드들의 Nativce API Call 추출과 은닉속성 분석을 통해 악성코드의 여부를 판단하고, Normal API 호출 함수의 기능을 분류해 악성코드의 행위를 분석하였다. API 호출을 모니터링하고, 정보를 수집할 때 전역 후커를 설치할 경우 시스템에 많은 부하를 가져오기 때문에 API 호출 함수의 특성과 기능에 따라, API 취약 행위별, Native API의 은닉 속성별로 분류하여 선택적으로 추출함으로써 부하를 줄였다. 정상 코드의 API 추출 결과, 악성코드 API 추출 결과를 근거로 판별함수를 구하여 탐지하고 Support Vector Machine을 통해 분류 성능 확인까지 실험을 진행하였다[12].

2.3.2 API Call Time Interval을 활용한 머신러닝 기반의 악성코드 탐지

Sandbox를 통하여 API Call의 시간 정보를 추출하여 API 호출 사이의 Time Interval을 계산하여 이를 특성으로 사용하였다. 또한 API를 호출한 횟수가 많아질수록 변화량이 작은 시간 간격이 증가할 수 있고, 결국 평균 시간차를 계산할 경우 변화량이 큰 호출 정보의 의미를 축소시키는 것을 전체 API Call 횟수를 기준으로 초기, 중기, 말기 3단계로 구분해 특성을 추출함으로써 방지하고 랜덤포레스트 모델에 적용하는 실험을 진행하였다[13].

2.4 Machine Learning

머신러닝의 대표적인 분류 알고리즘으로는 K-NN, Decision Tree, Support Vector Machine이 있다. K-NN 알고리즘은 거리 기반 분류 모델로서 데이터 거리에 대한 측정을 유클리드 거리, 맨해튼 거리 등을 사용하여 가장 가까운 속성에 따라 분류하는 알고리즘이고, Decision Tree의 경우 일련의 분류 규칙을 통해 데이터를 분류, 회귀하는 모델이다.

본 연구에서는 SVM(Support Vector Machine)을 사용하였다. SVM은 지도학습의 일종으로 분류(Classification)나 회귀 분석에 사용이 가능하며, 특히 분류 쪽의 성능이 뛰어나고, 실제로 많은 연구에서 사용되고 있다. 또한 SVM은 기본적으로 선형 분류 알고리즘이나 본 연구에서는 선형 분류를 할 수 없어 kernel trick을 사용하여 벡터 내적 연산을 대체하는 비선형 커널 함수 RBF(Radial Basis Function)를 사용하여 비선형 데이터를 분류하였다[14].

III. API의 호출 구간 특성 기반 악성코드 탐지 기술

3.1 Feature Selection Method using API Call Time Section

제안하는 모델에서 효과적인 분류를 위해 가장 중요한 정보는 호출하는 API의 정보이기 때문에 동적 분석이 필요하였고, 대표적인 동적 분석 플랫폼인 Cuckoo Sandbox를 사용하여 Feature 정보를 획득하였다. 우선 PE 파일을 Cuckoo Sandbox에 전달하면 Cuckoo Sandbox가 분석 후 분석 결과 파일이 JSON 형태의 파일로 생성된다. 생성된 JSON 파일로부터 프로그램 실행 시간 정보, API 호출 정보, Entropy 정보를 Python을 사용하여 파싱 후 프로그램의 실행시간 정보를 활용하여 5구간으로 나누어 각 구간에 대해 API 호출을 계수하였다. Fig. 1., Fig. 2.은 정상파일, 악성파일 각각 5,000개씩 분석했을 때의 결과이다. 이를 첫 번째 특성인자로 사용하였고 두 번째로는 Entropy라는 특성인자를 사용하였는데 Section 별 Entropy 값 중 가장 높은 값을 사용하였다. 마지막으로 전체 호출 API Call 정보를 활용하여 General API, Native API 호출의 비율을 계산하여 각 구간별 10개씩 총 API Feature 50개, API Feature에 대한 점수, Entropy, API Ratio 총 53개의 특성을

Table 1. If the program runs for 100 seconds

	Sec
Section 1	0~19
Section 2	20~39
Section 3	40~59
Section 4	60~79
Section 5	80 ~ 100

사용해 SVM 학습에 사용하였다.

Table 1.은 프로그램 실행시간이 100초인 경우의 예시이다. 0 ~ 19초 사이에 호출된 API의 경우 1구간에 포함된다.

3.2 특성인자 선택

3.2.1 API

API(Application Programming Interface)는 응용 프로그램에서 사용할 수 있도록, 운영 체제나 프로그래밍 언어가 제공하는 기능을 제어할 수 있게 만든 인터페이스이다. 정상 파일과 악성 파일은 이 API 호출에 차이를 보이고, 실행되는 시간을 각 구간별로 나누었을 때도 호출하는 API가 Fig. 1., Fig.2.에서 볼 수 있듯이 각 구간별로도 정상 파일과 악성 파일이 호출하는 API의 차이를 확인할 수 있었기 때문에 이 특징을 선정하였다.

3.2.2 Entropy

정보의 무질서도를 의미하는 Entropy는 악성 파일의 경우 정상 파일에 비해 높은 확률로 패킹 또는 실행 압축, 난독화가 되어 있어 높은 Entropy의 값을 가지게 된다. 하지만 정상 파일도 보안을 위해 패킹, 실행 압축과 같은 기법을 사용하는 경우가 있어 절대적인 특성으로써 사용할 수는 없지만, 부가적인 특성으로는 충분히 의미가 있다.

Fig. 3의 경우 악성 파일 5,000개와 정상 파일 5,000개를 사용하여 x 축에는 Entropy 값, y 축에는 그 Entropy의 값을 가지는 파일의 개수를 표현하였다. 악성 파일의 경우 Entropy 7.0 이상에 50% 넘게 분포해 있는 것을 확인할 수 있었고 이는 확실히 정상 파일과 악성 파일의 Entropy의 차이가 유의미하다는 것을 나타내므로 특성으로 선정하였다.

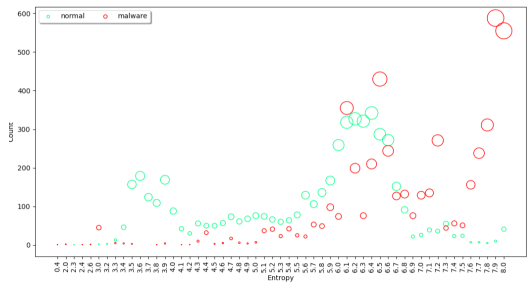

Fig. 3. Entropy Distribution Chart

1		2		3		4		5	
NtWriteFile	2,233,173	NtWriteFile	2,007,130	NtWriteFile	1,832,426	NtMapViewOfSection	2,920,283	NtMapViewOfSection	7,175,800
NtReadFile	1,783,784	NtReadFile	1,433,821	NtReadFile	1,404,646	NtWriteFile	1,636,843	NtReadFile	4,608,431
NtClose	1,626,243	NtDelayExecution	1,381,056	NtDelayExecution	959,661	DeleteFileW	1,587,457	NtWriteFile	4,272,866
NtDelayExecution	1,596,694	NtClose	924,871	NtClose	917,956	NtReadFile	1,488,308	LdrGetProcedureAddress	2,138,836
NtQueryDirectoryFile	1,547,124	GetForegroundWindow	676,449	DeleteFileW	779,199	NtClose	950,087	SetFilePointer	2,020,440
Process32NextW	905,577	NtQueryDirectoryFile	626,421	NtQueryDirectoryFile	639,963	NtDelayExecution	922,190	DeleteFileW	1,792,890
FindFirstFileExW	828,562	NtQuerySystemInformation	544,558	NtQuerySystemInformation	519,662	NtQueryDirectoryFile	605,199	NtClose	1,688,867
GetForegroundWindow	731,053	LdrGetProcedureAddress	539,168	LdrGetProcedureAddress	513,679	LdrGetProcedureAddress	509,026	NtDelayExecution	1,170,695
NtQuerySystemInformation	602,614	HttpSendRequestW	455,088	GetForegroundWindow	500,506	NtQuerySystemInformation	471,984	RegQueryValueExW	1,139,286
LdrGetProcedureAddress	579,645	FindFirstFileExW	403,748	NtCreateFile	391,889	NtCreateFile	421,283	NtQueryDirectoryFile	793,647

Fig. 1. Malware API call top rank for each section

1		2		3		4		5	
ReadProcessMemory	468,177	ReadProcessMemory	374,152	ReadProcessMemory	359,729	ReadProcessMemory	283,655	ReadProcessMemory	837,909
NtClose	120,217	CryptDecodeObjectEx	86,094	CryptDecodeObjectEx	88,161	GetSystemMetrics	113,194	RegQueryValueExW	191,419
NtReadFile	92,330	NtReadFile	81,710	GetSystemMetrics	74,574	NtReadFile	87,547	CryptDecodeObjectEx	177,895
CryptDecodeObjectEx	77,684	GetSystemMetrics	61,713	NtClose	65,920	CryptDecodeObjectEx	80,785	GetSystemMetrics	153,053
GetSystemMetrics	65,758	NtClose	59,637	NtReadFile	65,023	NtClose	74,501	NtClose	136,405
RegQueryValueExW	38,999	DeviceIoControl	43,064	GetKeyState	51,024	GetKeyState	51,624	RegOpenKeyExW	101,654
DeviceIoControl	38,346	GetKeyState	39,563	RegQueryValueExW	45,614	RegQueryValueExW	44,273	LdrGetProcedureAddress	94,742
GetKeyState	36,274	NtCreateFile	27,351	DeviceIoControl	43,187	RegOpenKeyExW	34,390	NtReadFile	88,326
LdrGetProcedureAddress	28,927	RegQueryValueExW	19,983	RegOpenKeyExW	34,458	DeviceIoControl	29,439	RegCloseKey	66,550
NtFreeVirtualMemory	27,802	Process32NextW	18,878	NtCreateFile	28,258	RegCloseKey	27,431	DeviceIoControl	58,981

Fig. 2. Normal API call top rank for each section

3.2.3 API Ratio

Fig. 1, Fig. 2를 통해 악성 파일과 정상 파일이 호출하는 API가 차이가 존재함을 알 수 있었다. Native API란 API 이름 앞에 Nt가 붙어있고, API 중에서 OS에서 지원하는 Subsystem에서 NT의 도움을 받고자 사용되는 함수의 모음이다.

Fig. 1과 Fig. 2를 보면 악성 파일과 정상 파일별 호출하는 API를 보면 악성 파일의 경우 Native API가 정상 파일에 비해 높은 순위에 많이 포함되어 있는 것을 확인할 수 있다. 이를 통해 General API와 Native API의 호출 비율에 차이가 존재한다는 것을 알 수 있었다. 이러한 특성을 사용하여 각 파일마다 General API와 Native API의 비율을 계산 후 그 비율의 차이를 특성으로 사용하였다. 이때 호출 개수는 똑같은 종류의 API 호출이 되어도 하나만 계수하였다.

IV. 실험 및 결과분석

4.1 실험 환경 및 데이터

본 연구에 사용한 환경은 Cuckoo Sandbox와 머신러닝 환경을 분리하여 진행하였고, Cuckoo Sandbox의 환경은 OS Ubuntu 18, CPU 4core, RAM 16GB의 사양이다. ML의 환경의 경우 OS Windows 10, CPU 4core, RAM 24GB의 하드웨어 사양을 사용했고, ML을 위한 개발언어로는 Python 3.9 버전과 ML 라이브러리 scikit-learn의 SVM을 사용하였다.

분류실험에 사용한 악성코드는 KAIST사이버보안연구센터에서 제공받은 데이터를 사용하였고, 정상 파일의 경우에는 Windows 10 32bit에서 Python을 사용하여 수집하였다. Cuckoo Sandbox로 분석한 파일의 수는 악성 파일 6,790개 정상 파일 5,180개이지만 실제 사용한 데이터는 악성 파일 6,000개 정상 파일 3,000개를 사용하였다.

4.2 실험 방법

PE File을 Cuckoo Sandbox를 통해 생성된 JSON 파일 중 악성 파일 6,000개 정상 파일 3,000개를 선택 후 특성을 선택하여, API의 경우 Fig. 1을 기준으로 각 구간별로 순위에 있는 API가 호출되는지를 확인하고 호출이 될 경우 순위에 따라 1위는 1.5에서 10위 0.6까지 총 5구간의 합을 사용하였다. 이렇게 될 경우 최소 0점에서 10.5*5인 최대 52.5의 점수를 가지게 된다. 하지만 Cuckoo Sandbox는 가상환경이다 보니 Anti VM기술이 적용되어 있거나 API 호출 자체를 하지 않아 API 호출 정보가 없는 경우도 존재하였다. 이러한 경우 정상 파일의 경우에는 정상 파일의 점수 평균으로 대치하였고, 악성 파일의 경우에는 악성 파일의 평균으로 대치하였다. Entropy는 자동으로 계산한 Entropy 수치 자체를 그대로 사용하였다.

SVM의 파라미터의 경우 대표적으로 gamma와 c가 있다. gamma 값은 결정 경계를 얼마나 유연하게 설정할 것인지를 결정하는 파라미터이다. 이 값을 너무 높이면 학습 데이터에 의존하게 되기 때문에 오버 피팅이 발생하게 되고, 또 너무 낮추면 언더 피팅이 발생할 수 있게 되기 때문에 적절한 값 설정이 필요하다. c 값의 경우 이상치에 대한 마진 값을 설정하는 파라미터이다. c 값이 높을 경우 마진 오류는 적어지나 도로의 폭도같이 좁아지고 낮을 경우 마진 오류는 크게 나나 도로의 폭은 넓어지게 된다. 이 두 파라미터의 최적값을 찾아보았을 때 gamma 2.25, c 1의 값이 나왔고, 이 값을 사용하였다.

4.3 실험 결과 및 분석

실험에서는 본 연구에서 제시하는 특성 3가지를 이용하여 {Score, Entropy} 조합, {Score, Rate} 조합, 세 가지 전부 사용한 경우에 대해 각각 실험을 진행해 Table 2.와 같은 결과를 도출해 내었다.

9,000개의 데이터 중 80%의 데이터를 모델을 학습시키는 Train에 사용하였고, 남은 20%를 모델을 평가하는 Test에 사용하였다. 또한 각 특징을 조합하여 실험해 각 특징별 민감도, 재현율, 정확도를 뽑아 보았고 정확도의 경우 (Score, API Rate) 조합, (Score, Entropy) 조합, (Score, Entropy, API Rate) 순으로 높은 것을 알 수 있고, (Score, Entropy, API Rate) 조합의 경우 정확도가 95.5%와 Fig. 4에서 확인할 수 있듯이 특징별 기여도도 골고루 분포해 있는 것을 확인할 수 있었다.

이는 본 연구에서 제시하는 특징들이 악성 파일과 정상 파일을 분류하는 데 유의미하다는 것을 보여주는 결과라고 판단된다.

Table 2. The Results of each Combination

C	Train		Test	
Score Entropy	Precision	94.97%	Precision	94.43%
	Recall	97.01%	Recall	96.93%
	Accuracy	94.59%	Accuracy	94.11%
Score Rate	Precision	92.78%	Precision	92.73%
	Recall	96.18%	Recall	96.18%
	Accuracy	92.41%	Accuracy	92.38%
Score Entropy Rate	Precision	96.54%	Precision	95.77%
	Recall	98.43%	Recall	97.59%
	Accuracy	96.61%	Accuracy	95.5%

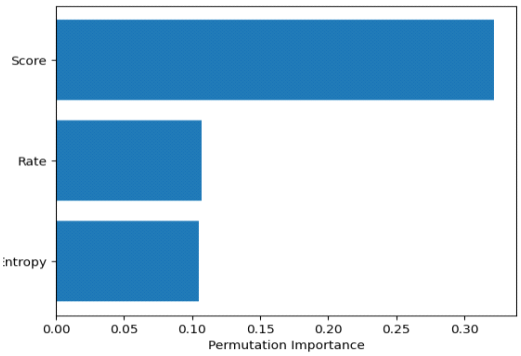

Fig. 4. Permutation Importance

V. 결 론

본 논문에서는 머신러닝을 이용한 악성코드 식별의 정확도를 높이기 위한 방법으로 파일을 실행할 때, API의 호출 시간을 구간별로 나누어 호출되는 API의 특성을 악성과 정상을 분류하는 특성인자로 사용하는 방법을 제안하였다. 제안한 방법은 대표적인 머신러닝 알고리즘인 SVM을 이용하여 학습시키고 분류한 결과 95.5%의 유의미한 정확도를 확인할 수 있었으며, 이를 통해 제안하는 방법이 악성코드 분류에 효과적인 특성인자 추출방법임을 검증하였다. 다만, 제안하는 방법을 실험할 때 이번 연구에는 SVM만을 가지고 실험하였으나, 향후 더 다양한 머신러닝 알고리즘을 이용하여 검증할 필요가 있으며, 제안하는 특성인자와 기존 연구된 특성인자를 혼합하여 정확도를 높이는 연구가 지속적으로 이루어져야 할 것이다.

References

[1] "ENISA Threat Landscape 2021" ENISA, Dec. 2021.

[2] Song Geunhye, Lee Seungmin, "Fourth Industrial Revolution and Security Paradigm Changes." Institute for Information & communication Technology Planning & evaluation, Weekly technology trend., 2018,16-27

[3] Kim Jongrak., "The relationship between artificial intelligence and information security,", Communications of the Korean Institute of Information Scientists and Engineers (Communications of KIISE), 2018, 14-16.

[4] HWANG, Ho; MOON, Daesung; KIM, Ikkun. "Trend and Issue Dynamic Analysis for Malware". In: Proceedings of the Korea Information Processing Society Conference. Korea Information Processing Society, 2015. p. 418-420.

[5] HWANG, Ho; MOON, Daesung; KIM, Ikkun. "Efficient Exploring Multiple Execution Path for Dynamic Malware Analysis", Journal of the Korea Institute of Information Security & Cryptology, 26(2), pp.377-386, 2016,

[6] Hwang, Ho, Daesung Moon, and Ikkun Kim. "Trend and Issue Dynamic Analysis for Malware." Proceedings of the Korea Information Processing Society Conference. Korea Information Processing Society, 2015.

[7] Taejin Lee, "Trend of Intelligent Malicious Code Analysis Technology Using Machine Learning," Korea Institute Of Information Security And Cryptology 28(2), 12-19, Dec. 2018

[8] Nam-Youl Park, Yong-Min Kim, Bong-Nam Noh, "A Behavior based Detection for Malicious Code Using

Obfuscation Technique," Journal of the Korea Institute of Information Security & Cryptology 16(3), 17-28, Jun. 2006
[9] Su-jeong Kim, Ji-hee Ha, Soo-hyun Oh, Tae-jin Lee, "A Study on Malware Identification System Using Static Analysis Based Machine Learning Technique," Journal of the Korea Institute of Information Security & Cryptology, 29(4), 775-784, Aug. 2019
[10] Ji-hee Ha, Su-jeong Kim, Tae-jin Lee, "Feature Extraction using DLL/API Statistical Analysis and Malware Detection based on Machine Learning," Journal of the Korea Institute of Information Security & Cryptology, 43(4), 730-739, Apr. 2018
[11] JAMALPUR, Sainadh, et al. "Dynamic malware analysis using cuckoo sandbox". In: 2018 Second international conference on inventive communication and computational technologies (ICICCT). IEEE, 2018. p. 1056-1060.
[12] Tae-woo Kang, Jae-ik Cho, Man-hyun Chung, Jong-sub Moon, "Malware Detection Via Hybrid Analysis for API Calls," Journal of the Korea Institute of Information Security & Cryptology, 17(6), 89-98, Dec. 2007
[13] Young Min Cho, Hun Yeong Kwon, "Machine Learning Based Malware Detection Using API Call Time Interval," Journal of the Korea Institute of Information Security & Cryptology, 30(1), 51-58, Feb. 2020
[14] Wikipedia, "Support-vector machine" https://en.wikipedia.org/wiki/Support-vector_machine, Jan. 2022

〈저자소개〉

김 동 엽 (Dong-Yeob Kim) 학생회원
2019년 2월~현재: 영남이공대학교 사이버보안계열 재학
〈관심분야〉 네트워크 보안, 시스템 보안, 웹 보안, 악성코드 분석

최 상 용 (Sang-Yong Choi) 정회원
2000년 2월: 한남대학교 수학과 졸업
2003년 2월: 한남대학교 컴퓨터공학과 석사
2014년 2월: 전남대학교 정보보안협동과정 박사
2012년 2월~2015년 12월: 한국과학기술원 사이버보안연구센터 책임연구원
2015년 12월~2017년 9월: 한국폴리텍대학 정보보안과 조교수
2017년 10월~2019년 2월: 한국과학기술원 사이버보안연구센터 연구교수
2019년 3월~현재: 영남이공대학교 사이버보안계열 조교수
〈관심분야〉 네트워크 보안, 웹 보안, 악성코드 분석, 정보보호 교육