Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.29, NO.4, Aug. 2019 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2019.29.4.775

정적 분석 기반 기계학습 기법을 활용한 악성코드 식별 시스템 연구*

김 수 정,† 하 지 희, 오 수 현, 이 태 진‡
호서대학교

A Study on Malware Identification System Using Static Analysis Based Machine Learning Technique*

Su-jeong Kim,† Ji-hee Ha, Soo-hyun Oh, Tae-jin Lee‡
Hoseo University

요 약

신규 및 변종 악성코드의 발생으로 모바일, IoT, windows, mac 등 여러 환경에서 악성코드 침해 공격이 지속적으로 증가하고 있으며, 시그니처 기반 탐지의 대응만으로는 악성코드 탐지에 한계가 존재한다. 또한, 난독화, 패킹, Anti-VM 기법의 적용으로 분석 성능이 저하되고 있는 실정이다. 이에 유사성 해시 기반의 패턴 탐지 기술과 패킹에 따른 파일 분류 후의 정적 분석 적용으로 기계학습 기반 악성코드 식별이 가능한 시스템을 제안한다. 이는 기존에 알려진 악성코드의 식별에 강한 패턴 기반 탐지와 신규 및 변종 악성코드 탐지에 유리한 기계학습 기반 식별 기술을 모두 활용하여 보다 효율적인 탐지가 가능하다. 본 연구 결과물은 정보보호 R&D 데이터 챌린지 2018 대회의 AI기반 악성코드 탐지 트랙에서 제공하는 정상파일과 악성코드를 대상으로 95.79% 이상의 탐지정확도를 도출하여 분석 성능을 확인하였다. 향후 지속적인 연구를 통해 패킹된 파일의 특성에 맞는 feature vector와 탐지기법을 추가 적용하여 탐지 성능을 높이는 시스템 구축이 가능할 것으로 기대한다.

ABSTRACT

Malware infringement attacks are continuously increasing in various environments such as mobile, IOT, windows and mac due to the emergence of new and variant malware, and signature-based countermeasures have limitations in detection of malware. In addition, analytical performance is deteriorating due to obfuscation, packing, and anti-VM technique. In this paper, we propose a system that can detect malware based on machine learning by using similarity hashing-based pattern detection technique and static analysis after file classification according to packing. This enables more efficient detection because it utilizes both pattern-based detection, which is well-known malware detection, and machine learning-based detection technology, which is advantageous for detecting new and variant malware. The results of this study were obtained by detecting accuracy of 95.79% or more for benign sample files and malware sample files provided by the AI-based malware detection track of the Information Security R&D Data Challenge 2018 competition. In the future, it is expected that it will be possible to build a system that improves detection performance by applying a feature vector and a detection method to the characteristics of a packed file.

Keywords: Malware, Machine learning, Feature statistics, Packer, Similarity hashing

Received(03. 15. 2019), Modified(06. 26. 2019), Accepted(07. 11. 2019)

* 본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터 지원사업의 연구결과로 수행되었음 (IITP-2018-2018-08-01417*)

† 주저자, sjkim395@gmail.com
‡ 교신저자, kinjecs0@gmail.com (Corresponding author)

I. 서 론

신규 및 변종 악성코드가 지속적으로 발생하고 있는 가운데 AV-TEST의 2017/2018 보고서에 따르면 모바일, IoT, windows 운영체제, mac 운영체제 등을 대상으로 한 악성코드 침해공격이 계속해서 증가하고 있다[3, 4]. 보고서 결과와 같이 무수히 출현하는 신·변종 악성코드로 인해 패턴 기반의 대응만으로는 악성코드 탐지에 한계가 존재하고 있으며, 난독화, 패킹, Anti-VM 등으로 정적·동적 분석 기술들의 분석 및 대응은 어려워지고 있다[16,17,18].

이에 여러 알고리즘을 이용하여 탐지 성능을 강화하는 것이 중요하며 악성코드를 탐지하기 위한 연구는 지속해서 제안되고 있다. 그중 유사성 해시 기반 탐지기법은 대표적으로 ssdeep, TLSH, DHASH, nilsimsa hash, SDHASH 등이 존재한다[12]. 유사한 파일이 유사한 해시값을 가지게 하는 해시 기법이며 해시된 데이터를 유클리드 거리 측정법과 같은 유사도 값을 도출하여 파일의 유사한 정도를 비교하는 방식이다. 도출된 유사도 값을 비교하여 어느 정도 유사한지 판단할 수 있으며, 적정 threshold를 지정하여 악성코드의 탐지가 가능하다. 유사성 해시를 이용한 악성코드 탐지는 유사한 파일을 찾기 위해 사전에 분석한 데이터를 기반으로 유사도를 분석하기 때문에 신규 악성코드 발생 시 탐지에 어려움을 겪게 된다. 기계학습 기법의 supervised learning 방식을 이용하면 기존에 보유한 데이터의 특성을 나타내는 feature를 이용하여 모델링을 할 수 있다. 이때, 기존 데이터의 label이 주어진 상태로 학습하기 때문에 새로운 데이터의 결과는 학습된 label 중 하나로 나타나게 된다. 새로운 데이터가 접근하면 학습된 모델을 이용하여 기존 데이터의 악성코드와 유사한 신규·변종 악성코드의 탐지가 가능하다.

본 논문에서는 유사성 해시 기반 시그니처 방식과 정적분석 feature에 기반을 둔 기계학습 방식을 연계하여 악성코드를 분류하는 시스템을 소개한다. 이를 위해 PE(Portable Executable) 파일이 실행되는 데 필요한 정보가 존재하는 PE header에서 정적분석 기반 feature를 추출하고 유사 해시 기법의 ssdeep, TLSH, DHASH 알고리즘과 기계학습의 SVM, DNN, Decision Tree, k-NN 알고리즘을 이용하여 실험한다. 해당 연구를 통해 기존에 존재하는 악성코드를 빠르고 정확하게 분석하는 유사성 해시 기법과 변종 악성코드도 분석이 가능한 기계학습으로 악성코드 분석의 정확도 향상에 이바지할 수 있다. 또한, 정적분석 기술의 대응을 어렵게 하는 난독화기술, 패킹기술에 대응하기 위해 악성코드를 특징에 따라 분석한 결과를 바탕으로 향후 악성코드 분석 분야의 연구 방향을 제시한다.

II. 관련 연구

2.1 유사성 해시 기반 방식의 악성코드 분석 연구

Li 외 6인은 퍼지 해시 기반의 악성코드 클러스터링 연구에서 기존의 문제를 극복하는 새로운 알고리즘을 제안하였다[5]. sDHASH와 mvHash-B와 같은 블록 기반 퍼지 해시 알고리즘에서 입력순서가 달라지면 일관성없는 결과가 발생하는 문제가 있다. 즉, 동일한 데이터를 다른 순서로 입력할 경우 상이한 유사도를 갖는다. 또한, 블록 기반 퍼지 해시 알고리즘은 데이터의 길이가 다른 경우, 유사도가 달라지는 문제가 있다. 동일한 데이터 중 하나에 무의미한 패딩을 추가할 경우 상이한 유사도를 갖게 되는 것이다. 해당 연구에서는 앞선 문제들이 기존의 알고리즘에 거리 계산법에서 기인한다고 판단하고, 이를 해결하기 위한 새로운 거리 계산 알고리즘을 이용한 악성코드 클러스터링을 제안했다.

Dong-woo Goh와 Huy-kang Kim은 악성코드의 API 콜 시퀀스를 이용하여, 악성코드의 유사성 해시를 기반으로 클러스터링을 분석하는 기법에 대해 제안했다[6]. cuckoo sandbox를 이용하여 가상분석 환경으로 API 콜 시퀀스를 획득하고, TLSH 알고리즘으로 해싱 후 유사도를 계산하여 distance matrix를 생성하면 클러스터링하여 악성코드를 카테고리별로 식별한다. API명이 상이하지만 동일한 기능을 수행하는 경우 TLSH 해시값이 달라지는 문제점을 해결하기 위해 API 추상화 방식을 사용하였으며, API 콜 시퀀스 데이터로 악성코드의 유형을 분석하는 실험을 진행했다.

Min 외 2인은 안드로이드 환경에서 발생하는 악성코드의 탐지를 위한 시그니처 기반의 악성코드 분석 시스템을 제안했다[7]. 해당 논문에서는 모바일 악성코드의 기하급수적인 성장은 주로 악성코드를 변형시켜 쉽게 변종 악성코드를 생성하기 때문이며, 이를 위해 악성코드를 수집, 관리하기 위한 체계적인 프로세스가 필요하다고 언급한다. 리패키징 및 난독화된 악성코드를 구분하기 위해 opcode 수준에서 악

성코드의 공통점을 분석하여, 악성코드를 자동으로 수집, 관리, 분석, 추출이 가능한 시스템을 실험했다.

2.2 기계학습 기반 방식의 악성코드 분석 연구

Ahmadi 외 2인은 Microsoft Kaggle 변종 악성코드 챌린지 대회에서 제공한 악성코드를 대상으로 악성코드 식별 실험을 진행하여 99.77%의 정확도를 확인했다[8]. 패킹 및 난독화된 악성코드의 분류를 위한 feature 추출에 중점을 두었으며, 심볼(-,+,*,?,@,[,]), entropy, opcode, API의 사용 빈도, 섹션 등을 feature로 활용하였다. 기계학습 알고리즘으로는 SVM, XGBoost를 활용하였으며, 단일 모델의 성능 향상을 위해 bagging을 사용하여 악성코드 식별 실험을 진행했다.

Zhong 외 2인은 바이너리 파일 내부에 존재하는 API 호출 정보, String 문자열, 함수의 basic block 정보를 IDA Pro로 추출하여 feature로 이용하고 판단할 수 없는 파일의 패밀리를 분류하는 시스템을 제안하였다[9]. 바이너리 파일에서 추출한 정보에 N-gram을 반영하여 feature를 가공하였으며, 파일의 특성을 강하게 나타내는 특성함수를 찾기 위해 one class SVM 알고리즘을 적용했다. 그리고 악성코드를 패밀리 별로 분류하기 위해 각 패밀리에 해당하는 특성함수, 함수의 이진 데이터, 문자열, API 목록 정보를 데이터베이스에 저장했다. 해당 데이터베이스를 이용하여 패밀리를 알 수 없는 파일을 분류하는 연구를 진행하였다.

Islam 외 3인은 파일의 string 정보, API 콜 시퀀스와 같은 정적 정보와 동적 정보를 모두 이용한 feature를 추출하고 WEKA 소프트웨어에서 제공하는 SVM, Decision Tree, 랜덤 포레스트, IB1 알고리즘으로 악성코드 식별 성능을 비교하는 실험을 진행했다[10]. 정적 feature를 추출하기 위해 패킹된 파일은 모두 언패킹하여 데이터로 함수의 길이와 문자열 정보를 추출하고, 동적 feature로는 호출된 windows API의 이름과 매개변수로 구성된 API 기능을 추출한 후 기계학습에 활용하였다.

III. 제안모델

3.1 개요

Fig. 1.은 본 논문에서 제안하는 유사성 해시 기반 시그니처 방식과 기계학습 기반의 악성코드 탐지 과정을 보여준다. 우선 학습데이터의 feature를 추출 및 가공 후 유사성 해시 알고리즘으로 해시값을 추출하여 저장한다. supervised learning 기법인 기계학습 알고리즘의 training을 통해 모델을 생성한다. 다음으로 테스트데이터가 입력되면 학습데이터와 동일한 방법으로 feature를 추출 및 가공하고 유사성 해시 알고리즘과 기계학습 알고리즘으로 악성코드를 탐지한다. 이때, 유사성 해시 알고리즘마다 판단 가능한 유사도 값에 대한 threshold를 지정하고 악성코드, 정상파일, 탐지 불가로 판단한다. 탐지가 불가능한 데이터를 대상으로 기계학습 알고리즘을 ensemble한 결과를 도출한다. 학습데이터는 KISA에서 제공하는 데이터와 정보보호 R&D 데이터 챌린지 2018 대회의 AI 기반 악성코드 탐지 트랙에서 제공하는 데이터를 포함하여 58만 개를 사용하였으며, ssdeep의 경우 NIST와 VirusShare에서 공유하는 ssdeep 해시를 추가로 활용하였다[1,2,13,14]. 테스트데이터 A는 정보보호 R&D 데이터 챌린지 2018 대회의 AI 기반 악성코드 탐지 트랙의 본선 1차 데이터를 사용하였으며, 테스트데이터 B는 본선 2차 데이터를 사용하였다. 테스트데이터 A, B는 각각 악성코드 7300개, 정상파일 2700개로 구성된다.

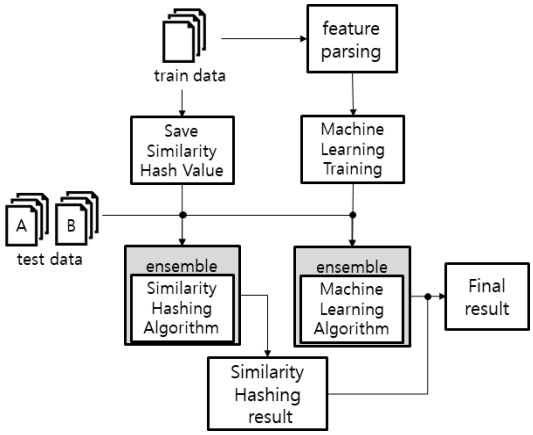

Fig. 1. Process of malware identification system

3.2 유사성 해시 기반 악성코드 분석 알고리즘

실험에 사용되는 유사성 해시 알고리즘은 ssdeep, TLSH, DHASH로 ssdeep은 퍼지 해시 도구를

기반으로 한 유사 악성코드 분류 기법이다[11]. 파일이 입력되면 롤링 해시를 사용하여 해시한 후 생성된 해시값들을 연결한다. 롤링 해시는 파일을 슬라이딩 윈도우 방식으로 해시한다. 슬라이딩 윈도우 방식은 일정한 윈도우의 크기를 유지한 채 바이트씩 이동하며 읽기 때문에 파일의 시작 부분에 문자열이 삽입되더라도 전체 해시 결과가 변경되지 않아 기존 해시 방식의 취약점을 보완한다. 이렇게 생성된 해시는 공통된 내용을 가진 파일에서 어느 정도의 유사성을 나타낼 수 있다. 동일하거나 매우 유사한 파일은 100의 유사도를 가지게 되고 유사하지 않을수록 0과 가깝다. Oliver 외 2인이 제안한 TLSH는 트랜드마이크로 사에서 개발한 새로운 유사성 해시 알고리즘으로 바이트 스트림에 대해 유사성 비교가 가능하도록 퍼지 해시를 사용한다[12]. 동일한 바이트가 연속될 경우 해시값을 생성하지 않기 때문에 TLSH를 적용하는 데이터는 복잡성이 존재하는 바이트 스트림 형태여야 한다. ssdeep이나 sDHASH의 경우 유사도 값은 0부터 100 사이이지만 TLSH의 유사도 값은 여러 상황에 적용할 수 있도록 1000 이상의 값도 가질 수 있도록 설계되었으며, 가장 유사하다고 판단하는 데이터의 유사도는 0이 된다. DHASH는 이미지에 대해 해시값을 도출하여 유사한 해시값을 판별하기 위해 사용되는 해시 방식으로 파일을 이미지 형태로 변경한 후 이미지의 바이너리 데이터를 이용하여 해시를 수행한다. 추출된 해시값과 해밍 거리를 이용한 유사도 계산으로 유사한 이미지를 판별할 수 있으며, 유사할수록 0에 가까운 값이 도출된다.

본 논문에서는 ssdeep의 유사도 threshold를 80으로 지정한다. 이보다 낮은 유사도를 이용할 경우 더 많은 파일의 악성 여부를 판단할 수 있지만, 악성코드 분석 정확도가 하락할 수 있다. TLSH는 30 이하의 값을 가지는 데이터의 경우 높은 정확도를 가지므로 threshold를 40 이하로 제한하였다.

3.3 PE 파일의 feature 추출 및 가공

PE(Portable Executable) 파일 포맷은 windows 운영체제에서 정의한 포맷으로 실행파일, DLL, 오브젝트 등을 위한 파일 형식이다. PE 파일에서 실행에 필요한 정보는 PE 구조의 헤더와 섹션에 존재한다[15,20]. 그중 실질적으로 실행 코드에 대한 정보는 모두 섹션 영역에 존재하고 있으며, 섹션명에 따라 담고 있는 내용이 다르므로 섹션명에 기반한 특징도 파악이 가능하다. PE 파일에 import된 DLL, API 정보는 악성 행위의 특징을 가지고 있으며, 섹션명, entropy는 파일에 적용된 난독화 기술에 대해 규칙성을 가지고 있다. 본 논문에서 사용된 feature는 PE 헤더 57종류의 값을 가공한 94개 feature와 DLL명과 API명을 가공한 320개, entropy를 가공한 256개로 총 670개 feature를 기계학습에서 사용하였다.

import된 DLL과 API는 악성코드와 정상파일 사이에서 차이가 존재하며, 특정 유형의 악성코드 여부와 제작자의 성향, 제작방법을 확인할 수 있으므로 feature로 활용할 수 있다. Fig. 2.는 정적분석으로 추출 가능한 imported DLL과 API의 일부를 나타낸다. Fig.3.은 본 논문에서 활용되는 256개의 DLL, API feature가 악성코드와 정상파일 사이에서 수치상의 차이가 존재함을 보여준다.

분석대상 파일별로 가변적인 DLL, API의 개수와 이름을 가지기 때문에 본 연구에서는 대량의 데이터에서 모든 DLL, API의 정보를 feature 담기 위해 feature 해싱 기법을 적용하였다. Fig. 4.는 분

Imported DLL_List
WINHTTP.dll
PSAPI.DLL
KERNEL32.dll
USER32.dll
GDI32.dll
ADVAPI32.dll
SHELL32.dll
ole32.dll
:
:

Imported API_List
VirtualFreeEx
CreateRemoteThread
WriteProcessMemory
GetModuleHandleA
FindResourceA
CopyFileA
FindClose
FindFirstFileA
:
:

Fig. 2. Example of imported DLL and API

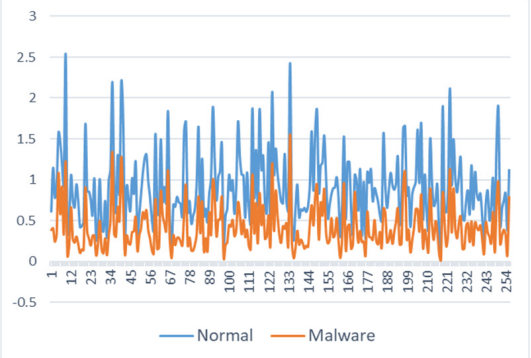

Fig. 3. Distribution between Malware and benign file of DLL and API using 256 features

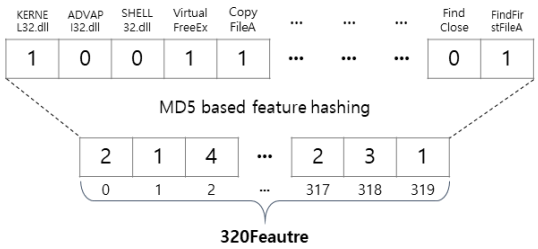

Fig. 4. Example of applying hash technique to imported DLL and API

석대상 파일 전체에서 추출한 DLL, API 목록에서 각 파일에 존재하는 DLL, API 리스트에 MD5 기반 feature 해시 기법을 적용하여 320개의 DLL, API feature로 가공한 과정을 보여준다.

entropy 값은 파일의 섹션마다 존재하여 0~8 사이의 값으로 복잡도 및 무질서한 정도를 나타낸다. entropy를 이용하면 파일이 패킹으로 인해 압축이나 암호화되어있는지 알 수 있다. 패킹기법은 파일을 압축하는 기법으로 파일의 용량을 줄이고, 데이터의 분석을 어렵게 만든다. 패킹기법이 적용되지 않은 PE 파일은 중간의 코드 영역을 제외하고는 대부분 0x00 바이트로 이루어지기 때문에 높지 않은 수치의 entropy를 가지게 된다. 패킹기법이 적용된 파일의 섹션은 원래 코드가 다른 코드로 대체되어 변경되며, 예를 들어 UPX 패커 사용 시 패킹된 바이너리의 기본 헤더는 변경되지 않았으나 섹션은 UPX0, UPX1, UPX2로 변경되고, ASPack 패커 사용 시 기존 섹션들은 그대로 존재하지만 .aspack 섹션과 .adata 섹션이 추가된다.

섹션명의 규칙적인 변경에 따라 변화하는 entropy 값을 이용하여 악성코드와 정상파일 사이의 섹션별 entropy의 경향성을 가진 유의미한 feature를 추출하기 위해 feature 해시 기법을 적용하였다. entropy의 경우 섹션별 entropy 값이 의미가 있는 것이기 때문에 entropy 값과 섹션명을 각각 16개의 feature로 가공하여 16*16으로 총 256개를 feature를 생성하였다.

3.4 기계학습 기반 악성코드 분석

기계학습 알고리즘은 SVM, DNN, Decision Tree, k-NN, Ensemble을 사용하였다[19,21]. SVM은 분류와 회귀분석을 위해 사용되는 지도학습 모델의 일종으로 데이터를 두 개의 클래스로 분류하여 판단할 수 있게 하는 비확률적 이진 선형 분류 모델을 생성한다. 생성된 분류 모델은 두 개의 데이터 집합의 경계를 표현하게 된다. 각 집합과 경계 사이의 공간을 마진이라 하며, 마진이 가장 큰 폭을 가지는 경계를 찾는 것을 목적으로 한 알고리즘이다. SVM은 비선형 데이터를 위한 방식으로도 사용할 수 있으며, 기존의 알고리즘 중 내적 연산을 비선형 커널 함수로 대체한 것을 말한다.

DNN은 입력 계층, 은닉 계층, 출력 계층이 연결된 구조로 비선형 데이터를 모델링 할 수 있는 알고리즘이다. gradient decent를 이용하여 weight 값을 조정하면서 학습이 수행되기 때문에 알고리즘은 최적의 weight 값을 찾는 것을 목표로 동작한다. cost 함수를 이용하여 실제값과 예측값 사이의 차이를 계산하여, cost 함수를 최소화하도록 weight 값을 조정한다.

k-NN은 거리 계산 알고리즘을 이용하여 테스트 데이터에서 가장 가까운 k개의 학습데이터를 찾는 알고리즘이다. 가장 가까운 k개의 데이터 중 가장 많은 label이 속하는 그룹으로 분류할 수 있다. k가 작을수록 데이터의 지역적 특성이 크게 반영되며, k가 커질수록 모델이 정규화된다. 제안하는 시스템에서는 k 값이 각각 1, 10, 100이고 유클리드 거리 측정법을 사용한 모델, k 값이 10이면서 코사인 거리 측정법을 사용한 모델, k 값이 10이고 유클리드 거리 측정법을 사용하면서 거리에 제곱 역수 가중치를 부여한 모델을 생성한다. 총 5가지 모델의 결과는 majority vote 방식을 이용하여 하나의 결과로 k-NN 결과로 이용한다.

Ensemble이란 여러 모델을 이용하여 데이터를 학습하고, 모든 모델의 예측결과를 평균하여 최적의 결과를 예측하는 기법이다. 여러 모델은 Decision Tree를 이용하여 구성하였으며, Bagging Boosting 기법을 이용하여 예측결과를 판단했다. Decision Tree는 트리구조의 형태로 학습되는 모델이며, bagging(bootstrap aggregating)은 ensemble 기법의 하나로 데이터에 대해 여러 개의 bootstrap을 생성하고 각 bootstrap을 모델링 한 후 결합하는 방법이다. bootstrap이란 random sampling을 통해 raw 데이터로부터 생성되는 크기가 동일한 여러 개의 표본자료를 말한다.

본 논문에서는 악성코드와 정상 파일의 분류를 위해 SVM 알고리즘에 670개의 feature를 활용하여 단순한 초평면을 가지지 않는다. 때문에 비선형 데이

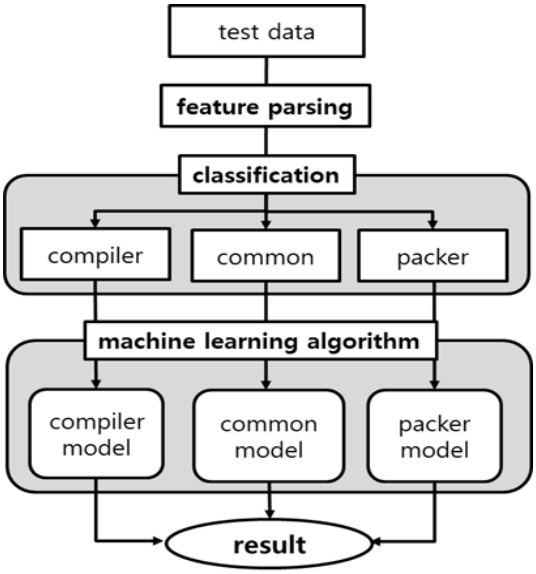

Fig. 5. Machine learning malware detection process by using packer

터에 더 적합한 3차 다항 커널 SVM을 활용하여 데이터를 분석하였다. DNN 알고리즘의 경우 activation 함수는 ReLU(Rectified Linear Unit), cost 함수는 cross entropy, weight 초기화 함수는 xavier를 사용했으며, cost 함수의 최소화를 위해 weight 값을 갱신하는 알고리즘으로 gradient decent를 사용하였다. Ensemble은 30개의 Decision Tree에 대해 bagging 방식을 이용하였으며, 유사 해시 기반 시그니처 탐지기법과 그 외의 ML 알고리즘은 majority vote 방식을 이용하였다.

기계학습 알고리즘을 사용한 학습에서 데이터를 패커 사용 여부에 따라 분류하여 서로 다른 경향을 띠는 데이터가 혼재되어 발생하는 잡음을 완화하였다. 패커의 사용 여부에 따라 패킹된 데이터와 컴파일링된 파일, 그 외의 일반적인 파일로 분류하였으며, Fig. 5.는 분류된 데이터셋을 이용한 기계학습 실험의 흐름도이다.

IV. 실험 및 결과

4.1 유사성 해시 알고리즘의 분석 정확도 및 판단 범위 비교

Table 1.은 유사성 해시 알고리즘을 이용한 악성코드 분석에서 테스트데이터 A를 대상으로 알고리즘별 threshold에 따른 정확도를 나타낸 표이다. 유사성 해시 알고리즘을 사용한 악성코드 분석방법은 기존에 준비한 학습데이터와 다른 유형의 데이터일 경우 판단할 수 없다. 유사도 값은 설정한 threshold를 악성 판단에 사용하게 되며, 전체 테스트데이터 중 threshold 내에 유사도 값이 추출된 데이터의 비율은 coverage로 나타낸다. coverage는 ssdeep이 51.38%로 가장 높으며, 세 가지의 유사성 해시 알고리즘을 ensemble한 결과는 coverage가 47.41%로 TLSH나 DHASH의 단일 coverage보다 상승하였다. threshold는 세 가지 알고리즘 모두 정확도가 97% 이상으로 높게 도출되도록 설정된 것을 확인할 수 있다. 3가지 유사성 해시 알고리즘을 ensemble한 경우에는 정확도가 99.16%로 상승했다.

4.2 패커 분류 비교

패커 사용 여부를 기준으로 학습데이터를 분류할 경우 정확도 향상에 효과가 있는지 테스트데이터 A를 대상으로 DNN 알고리즘을 사용하여 Fig.2.와 같이 실험한다. Table2.는 패커 사용 여부에 따른 학습데이터별 정확도와 학습데이터 분리 전, 후를 비교한 실험 결과표이다. 패커 사용 여부에 따른 학습데이터 분류 시 정확도가 92.96%에서 93.46%로 약간 상승하여, 결과에 큰 영향을 끼칠 것으로 보이지 않는다. 다만 패킹된 파일의 결과가 다른 파일의

Table 1. Experimental results on the accuracy of test data A for each similarity hashing algorithm

algorithm	threshold	coverage	$coverage \times number\ of\ malware$	$coverage \times number\ of\ benign$	accuracy
ssdeep	〉=80	51.38%	4186	952	97.00%
TLSH	〈=40	46.11%	3441	1170	97.12%
DHASH	〈=1	30.03%	2292	711	97.54%
similarity hashing	majority voting	47.41%	3936	805	99.16%

Table 2. DNN Results of test data A with packer usage

packer	data percentage	accuracy	precision	recall	F1-score
compiler	51.48%	94.619%	96.817%	95.758%	96.284%
packer	28.19%	89.713%	93.117%	93.952%	93.533%
common	20.33%	95.721%	95.840%	97.727%	96.774%
result without applying packer	100%	93.460%	95.496%	95.562%	95.529%
overall result based on packer usage	100%	92.960%	95.003%	95.068%	95.036%

Table 3. Experimental results on similarity hashing and machine learning using test data A

algorithm	accuracy	precision	recall	F1-score	false positive rate
k-NN	92.140%	96.264%	97.767%	97.010%	23.074%
decision tree	94.130%	91.558%	97.315%	94.349%	14.481%
SVM	94.050%	93.969%	96.466%	95.201%	12.481%
DNN	93.460%	94.551%	95.562%	95.054%	12.222%
similarity hashing and ML	95.770%	98.371%	98.438%	98.405%	11.444%

Table 4. Experimental results on similarity hashing and machine learning using test data B

algorithm	accuracy	precision	recall	F1-score	false positive rate
k-NN	92.430%	92.273%	97.822%	94.966%	22.148%
decision tree	94.520%	95.206%	97.397%	96.289%	13.259%
SVM	94.370%	95.404%	96.959%	96.175%	12.630%
DNN	94.130%	95.660%	96.329%	95.993%	11.815%
similarity hashing and ML	95.90%	96.354%	98.096%	97.217%	10.037%

결과보다 약 5%가량 낮은 것을 확인할 수 있다. 이는 패킹된 파일에서 정적 정보 추출 시 정적 정보도 일부 패킹되어 의미를 상실하기 때문이다. 패커 사용 여부에 따른 학습데이터 분류는 패킹된 파일에만 추가적인 분석방법을 실시할 경우 전체적인 결과 개선에 효과적일 것으로 기대된다.

4.3 기계학습 및 유사성 해시 알고리즘의 분석 정확도 비교

기계학습 알고리즘의 분석 실험과 기계학습 알고리즘과 유사성 해시 알고리즘을 ensemble한 실험은 2가지 테스트데이터로 실험을 수행하였다. 테스트데이터 A의 결과는 Table3,에서 확인할 수 있고, 테스트데이터 B의 결과는 Table4.에서 확인이 가능하다. 유사성 해시와 기계학습의 알고리즘을 모두 활용한 실험은 유사성 해시로 악성코드를 우선 탐지 후, 판단하지 못하는 범위의 데이터를 기계학습으로 탐지한 결과이다. 각각의 기계학습 알고리즘은 92~94% 탐지 정확도를 가지지만 유사성 해시 알고리즘과 모두 ensemble하면 95.77%와 95.9%로 정확도가 개선되며, 오탐율도 약간 낮아지는 것을 확인할 수 있다.

V. 결 론

본 논문에서는 악성코드 분석에서 이용되는 유사성 해시 알고리즘과 기계학습 알고리즘을 ensemble하여 악성코드 탐지 정확도를 개선하는 기법을 제안하였다. 유사 해시 기반 시그니처 기법을 이용해 변종 악성코드를 탐지하였으며, 기계학습 알고리즘에서 패커 사용 여부에 따라 분류된 데이터로

학습을 수행하여 패킹된 파일의 잡음을 제거하고자 하였다.

유사성 해시 알고리즘으로 ssdeep, DHASH, TLSH를 사용하여 파일의 해시값을 추출하였고, 해시를 비교하여 유사한 파일을 탐지할 수 있었다. 각각의 알고리즘이 97% 이상의 분석 정확도를 유지할 수 있도록 threshold를 조정해서 coverage 감소와 정확도 증가 사이의 적절한 값을 사용하였다. 3개의 유사성 해시 알고리즘을 Majority vote 방식으로 ensemble 하였으며, 기계학습 알고리즘과 연계하여 분석 정확도를 개선하였다. 기계학습 알고리즘으로는 k-NN, Decision Tree, SVM, DNN을 사용하였으며, 학습데이터를 패커 사용 여부로 분류하였다. Decision Tree는 bagging 방식으로 ensemble하였고, ensemble한 Decision Tree를 포함하여 4개의 기계학습 알고리즘에 Majority vote 방식으로 ensemble을 적용했다.

본 연구의 검증을 위해 2만 개의 정상 파일과 악성 파일을 1만 개씩 테스트데이터로 이용하여 실험을 수행하였다. 각각의 1만 개 데이터는 2018 R&D 데이터 챌린지 대회의 본선에서 사용된 1, 2차 데이터이며, 대회에서 제공된 데이터는 국내 백신 4사와 KISA에서 선정한 데이터이다[2]. 따라서 해당 데이터를 이용한 테스트는 본 논문에서 제안하는 시스템이 유효하다는 것을 보여준다. 유사성 해시 기반 시그니처 기법의 경우 적절한 threshold를 적용하여 97% 이상의 높은 정확도를 확보했지만, coverage가 낮은 이유로 기계학습 알고리즘과 ensemble 해서 결과를 개선하였다. 기계학습의 경우 30만여 개의 학습데이터와 670개의 feature로 실험하였으며, 모든 알고리즘에 대해 92% 이상의 높은 성능을 보였다.

본 연구는 기존 악성코드 분석에서 사용한 유사성 해시 알고리즘 및 기계학습 알고리즘의 ensemble을 통해 변종 및 신규 악성코드를 탐지하고자 하였다. 유사성 해시 알고리즘으로 정확도 99% 이상의 데이터를 우선 탐지하여 정확도를 높였으며, 기계학습 알고리즘에서 사용한 패커 사용 여부를 이용한 학습데이터의 분류를 통해 기계학습 시 유사한 데이터끼리 학습모델을 생성할 수 있도록 실험을 진행하였다. 실험 결과 KISA 주관 데이터 챌린지 대회 데이터를 대상으로 단일 알고리즘의 사용보다 향상된 95%의 결과를 확인하였으며, precision, recall, F1-score의 경우에도 96% 이상의 높은 수치를 확인하였다. 오탐율은 10%와 11.4%로 나타나 오탐을 개선할 경우 정확도를 개선할 수 있을 것으로 보인다. 학습데이터의 정상 파일 구성에서 windows 운영체제의 기본 파일 비중이 높았으므로 다양한 정상 파일의 구성을 통해 오탐율 개선이 가능할 것으로 기대된다. 또한, 패커 사용 여부의 비교 실험에서 패킹된 데이터의 결과가 가장 취약함을 확인하였다. 패킹된 파일의 탐지 정확도를 개선한다면 더욱 높은 악성코드 식별이 가능할 것이라 기대한다.

References

[1] KISA and KIISC, http://datachallenge.kr/challenge18/malware/introduction/

[2] KISA, https://www.kisis.or.kr/kisis/subIndex/283.do

[3] AV-TEST, The AV-TEST security report 2017/18, AV-TEST, Jul. 2018.

[4] Cisco, Cisco 2018 annual cybersecurity reposrt, Cisco, Jan. 2018

[5] Y. Li, S.C. Sundaramurthy, A.B. Bardas, X. Ou, D. Caragea, X. Hu and J. Jang, "Experimental study of fuzzy hashing in malware clustering analysis," 8th Workshop on Cyber Security Experimentation and Test ({CSET} 15). Aug. 2015.

[6] Dong-woo Goh and Huy-kang Kim. "A study on malware clustering technique using API call sequence and locality sensitive hashing," Journal of the Korea Institute of Information Security & Cryptology, 27(1) pp.91-101. Feb. 2017

[7] Z. Min, M. Sun and J.CS. Lui, "Droid analytics: a signature based analytic system to collect, extract, analyze and associate android malware," 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp. 163-171, July 2013.

[8] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, pp. 183-194, Mar. 2016.

[9] Y. Zhong, H. Yamaki and H. Takakura, "A malware classification method based on similarity of function structure," 2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet, pp. 256-261, July 2012.

[10] R. Islam, R. Tian, L.M. Batten and S. Versteeg, "Classification of malware based on integrated static and dynamic features," Journal of Network and Computer Applications, vol. 36, no. 2, pp. 646-656, Mar. 2013

[11] J. Kornblum, "Identifying almost identical files using context triggered piecewise hashing," Digital investigation, vol. 3, pp. 91-97, Sep. 2006

[12] J. Oliver, C. Cheng and Y. Chen. "TLSH--a locality sensitive hash," 2013 Fourth Cybercrime and Trustworthy Computing Workshop, pp.7-13, Nov. 2013.

[13] NIST, http://www.nsrl.nist.gov/ssdeep.htm

[14] Virusshare, https://virusshare.com

[15] M.Z. Shafiq, S.M. Tabish, F. Mirza, M. Farooq, "PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime," International Workshop on Recent Advances in Intrusion Detection, LNCS 5758, pp. 121-141, 2009

[16] Dong-hwi Shin, Chae-tae Im, Hyun-cheol Jeong, "The packer detection signature generation based on unpacking algorithm characteistic," Korea Computer Congress 2010, pp. 56-60, June 2010

[17] G. Taha, Counterattacking the packers, McAfee Avert Labs, 2007

[18] M. Bat-Erdene, T. Kim, H. Park and H. Lee, "Packer detection for multi-layer executables using entropy analysis," Entropy, vol. 19, no. 3, Mar. 2017.

[19] Ho-dong Lee, Reverse engineering 1 (file structure section), Hanbit Media, Oct. 2016.

[20] M. Sikorski and A. Honig, Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software, Acorn Publication, Oct. 2013

[21] S. Marsland, Machine Learning: An Algorithmic Perspective, Jpub, Dec. 2016

〈저자소개〉

김 수 정 (Su-jeong Kim) 학생회원
2018년 2월: 호서대학교 정보보호학과 졸업
2018년 3월~현재: 호서대학교 정보보호학과 석사과정
〈관심분야〉 시스템 보안, 악성코드 분석, 기계학습

하 지 희 (Ji-hee Ha) 학생회원
2018년 2월: 호서대학교 정보보호학과 졸업
2018년 3월~현재: 호서대학교 정보보호학과 석사과정
〈관심분야〉 정보보호, 악성코드 분석, 암호학

오 수 현 (Soo-hyun Oh) 종신회원
1998년 2월: 성균관대학교 전기전자 및 컴퓨터공학부(공학석사)
2003년 8월: 성균관대학교 전기전자 및 컴퓨터공학부(공학박사)
2004년 3월~현재: 호서대학교 컴퓨터정보공학부 교수
〈관심분야〉 암호학, 네트워크 보안, IoT 보안

이 태 진 (Tae-jin Lee) 종신회원
2003년 1월~2017년 2월 : 한국인터넷진흥원 팀장
2017년 3월~현재: 호서대학교 컴퓨터정보공학부 교수
〈관심분야〉 시스템 보안, 악성코드 분석, 기계학습