Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.32, NO.2, Apr. 2022 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2022.32.2.181

N-gram Opcode를 활용한 머신러닝 기반의 분석 방지 보호 기법 탐지 방안 연구*

김 희 연,[1†] 이 동 훈[2‡]
[1,2]고려대학교 (대학원생, 교수)

A Study on Machine Learning Based Anti-Analysis Technique Detection Using N-gram Opcode*

Hee Yeon Kim,[1†] Dong Hoon Lee[2‡]
[1,2]Korea University (Graduate student, Professor)

요 약

신종 악성코드의 등장은 기존 시그니처 기반의 악성코드 탐지 기법들을 무력화시키며 여러 분석 방지 보호 기법들을 활용하여 분석가들의 분석을 어렵게 하고 있다. 시그니처 기반의 기존 연구는 악성코드 제작자가 쉽게 우회할 수 있는 한계점을 지닌다. 따라서 본 연구에서는 악성코드 자체의 특성이 아닌, 악성코드에 적용될 수 있는 패커의 특성을 활용하여, 단시간 내에 악성코드에 적용된 패커의 분석 방지 보호 기법을 탐지하고 분류해낼 수 있는 머신러닝 모델을 구축하고자 한다. 본 연구에서는 패커의 분석 방지 보호 기법을 적용한 악성코드 바이너리를 대상으로 n-gram opcode를 추출하여 TF-IDF를 활용함으로써 피처(feature)를 추출하고 이를 통해 각 분석 방지 보호 기법을 탐지하고 분류해내는 머신러닝 모델 구축 방법을 제안한다. 본 연구에서는 실제 악성코드를 대상으로 악성코드 패킹에 많이 사용되는 상용 패커인 Themida와 VMProtect로 각각 분석 방지 보호 기법을 적용시켜 데이터셋을 구축한 뒤, 6개의 머신러닝 모델로 실험을 진행하였고, Themida에 대해서는 81.25%의 정확도를, VMProtect에 대해서는 95.65%의 정확도를 보여주는 최적의 모델을 구축하였다.

ABSTRACT

The emergence of new malware is incapacitating existing signature-based malware detection techniques., and applying various anti-analysis techniques makes it difficult to analyze. Recent studies related to signature-based malware detection have limitations in that malware creators can easily bypass them. Therefore, in this study, we try to build a machine learning model that can detect and classify the anti-analysis techniques of packers applied to malware, not using the characteristics of the malware itself. In this study, the n-gram opcodes are extracted from the malicious binary to which various anti-analysis techniques of the commercial packers are applied, and the features are extracted by using TF-IDF, and through this, each anti-analysis technique is detected and classified. In this study, real-world malware samples packed using Themida and VMProtect with multiple anti-analysis techniques were trained and tested with 6 machine learning models, and it constructed the optimal model showing 81.25% accuracy for Themida and 95.65% accuracy for VMProtect.

Keywords: Anti-analysis, N-gram, Malware Detection, Classification, Machine Learning

Received(11. 23. 2021), Modified(01. 28. 2022), Accepted(02. 23. 2022)

* 이 논문은 2022년도 과학기술정보통신부의 재원으로 한국연구재단의 지원을 받아 수행된 연구입니다. (No.NRF-2021R1A2C2014428).

† 주저자, sonysame@naver.com
‡ 교신저자, donghlee@korea.ac.kr(Corresponding author)

I. 서 론

최근 악성코드는 사이버 공간에서 핵심적인 공격 수단으로 활용되고 있으며, 신종 악성코드의 등장으로 피해가 증가하고 있다. AV-Test Institute에서 발표한 자료에 의하면 하루 평균 450,000개의 신형 악성코드가 탐지되고 있으며, COVID-19 이후 악성코드 감염 사이트가 더욱 증가하고 있다[1]. 악성코드를 첨부한 코로나 바이러스 관련 위장 메일, 재택 근무환경을 악용하여 기업 내 네트워크에 불법으로 침입하여 악성코드를 유포하는 등 새로운 방식으로 악성코드가 유포되고 있으며, 개인 및 기업은 쉽게 악성코드에 감염되고 있다. 악성코드가 네트워크 내부로 유입된 순간, 해당 네트워크와 연결된 모든 컴퓨터는 악성코드에 감염될 위험에 처한다. 따라서 이를 초기 단계에 단시간에 식별하고 분석하는 것이 중요하다.

이때, 분석가가 실시간으로 모든 실행 파일들을 직접 분석하여 악성코드를 분류하고 식별해내는 것은 거의 불가능하며 많은 시간이 소요된다. 또한 분석 방지 보호 기법이 적용된 악성코드들은 분석가들의 분석을 어렵게 하며 분석을 위해서는 해당 보호 기법들을 식별하고 이를 해제시키는 작업이 선행되어야 한다. 현재까지 일반적으로 악성코드를 식별할 때, 시그니처 기반 탐지를 사용한다. 하지만 시그니처 기반 탐지법은 공격자가 쉽게 우회할 수 있다. 시그니처 기반 탐지법을 비롯하여 현재 사용하고 있는 악성코드 탐지와 관련된 기술들은 악성코드 자체의 특성을 활용한다. 예를 들어, 악성코드에서 주로 사용되고 있는 API의 종류 혹은 자주 사용되는 문자열을 기반으로 악성코드를 탐지해내는데, 이는 공격자가 악성코드를 난독화시킬 경우 분석가가 API 정보 혹은 문자열 등의 정보를 추출해내기 어렵다는 한계점을 지닌다[2,19]. 또한 악성코드 제작자가 이를 우회하기 위해 기존과 다른 API를 사용하는 등의 새로운 방식으로 악성 행위를 수행하는 신종 악성코드를 제작할 수 있으므로 시그니처 이외의 방법으로도 악성코드를 식별하고 악성코드 분석에 도움이 되는 정보를 추출해 낼 필요성이 있다.

본 연구에서는 악성코드 자체의 특성이 아닌 악성코드에 주로 사용되는 패커의 분석 방지 보호기법을 식별하고 분류할 수 있는 머신러닝 모델을 제안하고자 한다. 패커는 프로그램 내부의 주요 정보와 핵심 알고리즘을 보호하기 위하여 만들어진 도구이나, 현재는 악성코드에서 주로 사용되어 악성코드를 난독화하는 경우가 많다. 실제로, 패커가 사용된 실행 파일 중 정상 파일은 적고, 악성코드가 대다수를 차지한다[3]. 이 점을 활용하여 악성코드를 식별할 때, 패커의 특성을 활용한다면 악성코드 탐지율을 높일 수 있을 것이다. 하지만 정상파일 또한 패커를 사용할 수 있다는 점을 고려해야 한다. 패커의 고유 문자열 혹은 고유 PE 헤더 정보 등 패커 자체의 특성만을 활용하여 머신러닝 모델을 개발한다면, 패킹이 이루어진 정상파일 또한 악성코드로 분류하는 등 오탐이 많아질 것이다.

기존 많은 연구들은 악성코드에 여러 분석 방지 보호 기법이 적용됨을 밝히고 있다[4,5,6,7]. Branco 등[5]의 연구에서는 4,030,945개의 악성코드 샘플을 수집하였고, 이 중 88.96%의 악성코드들이 한 개 이상의 분석 방지 보호 기법을 지니고 있음을 보여주었다. Lee 등[8]은 상용 패커의 분석 방지 보호 기법을 우회하는 방법을 제안하였다. 해당 연구는 시그니처 기반의 특징을 활용하여 우회하였는데, 이는 시그니처 기반으로 분석 방지 보호 기법을 탐지할 수 있음을 의미한다. 하지만 해당 연구[8]는 이러한 시그니처 기반의 탐지는 공격자가 쉽게 우회할 수 있다는 한계점을 보여준다. 따라서 공격자가 우회하기 어려운 방식으로 분석 방지 보호 기법을 탐지하는 연구가 필요하다.

또한 Themida로 패킹된 악성코드 샘플 15개에 대해 분석 방지 보호 기법을 적용하기 전과 후 Virus Total[9]의 악성코드 탐지 툴의 개수를 비교하였다. Virus Total에서는 60여개의 안티 바이러스툴로 악성코드를 탐지하는데, 20개의 패킹된 정상 바이너리 샘플로 실험을 한 결과, 60개 중 20개의 Anti-Virus툴은 85%이상의 정상 바이너리를 악성코드로 분류함을 확인할 수 있었다. 따라서 패킹된 사실만으로도 정상 바이너리를 악성코드로 분류하는 20개의 Anti-Virus툴을 제외한 나머지 툴에 대해 분석 방지 보호 기법을 적용하기 전과 후의 악성코드 탐지율을 비교하였다. 그림 1과 같이 악성코드 탐지율이 평균적으로 보호 기법 적용 후에는 보호 기법 적용 전의 28.18%로 감소함을 확인할 수 있었다. 따라서, 본 연구에서는 패커 자체의 특성이 아닌 악성코드가 활용할 수 있는 패커의 기능에 초점을 맞추어 악성코드가 활용하는 다양한 분석 방지 보호 기법을 탐지하고 분류할 수 있는 머신러닝 모델을 제안한다.

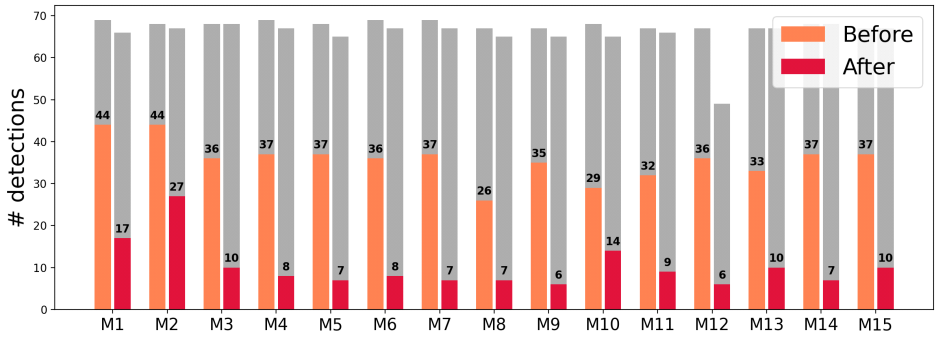

Fig. 1. The number of detections on Virus Total for malwares before and after applying anti-analysis techniques

II. 관련 연구

악성코드 분류와 관련된 연구는 2000년대 초반부터 시작되어 점차 진화되어 왔다. 2000년대 초반에는 정적인 특성에 기반하여 실행 파일의 opcode를 디스어셈블러를 사용하여 추출 후, 그 자체를 시그니처로 활용하여 악성코드를 분류하는 연구가 주를 이루었다[18]. 하지만 opcode 자체는 데이터가 너무 방대하여 악성코드 분류에 사용하기에는 효율적이지 않았다. Pechaz 등[10]은 실행 파일을 디스어셈블링해서 추출한 opcode를 그 자체로 사용하는 것이 아니라, 이들의 엔트로피를 계산해서 이를 머신러닝 모델 피처로 활용하는 방법을 제시하였다.

이어서 API를 활용하는 연구들이 발표되는데, Hansen 등[11]과 Kwon 등[12]이 수행한 연구에서는 API 시퀀스를 사용해서 악성코드를 분류하였다. 하지만 API의 사용은 악성코드 제작자에 의해 쉽게 난독화 될 수 있고, 신종 악성코드에서는 적용되지 않을 수 있다는 한계점이 존재한다. Raff 등[13]의 연구에서는 문자열 정보와 PE 헤더의 정보를 활용해서 악성코드를 탐지해내는데, 이 또한 API 시퀀스와 마찬가지로 난독화가 적용되면 해당 정보를 추출하기 어렵다는 한계점을 지닌다.

Aghakhani 등[14]의 연구에서는 PE 헤더, PE 섹션, DLL 임포트, API 임포트, Rich 헤더, 바이트 n-gram, opcode n-gram, 문자열, 파일 generic 총 9개의 피처를 추출해서 56,539차원의 벡터를 생성하여 머신러닝 분류 모델을 개발하였는데, 해당 연구에서는 악성코드 탐지와 관련된 기존 타 연구에서 활용되었던 대부분의 피처값을 모두 수집해서 모델 학습에 사용하였다. 하지만 해당 연구에서도 밝히기를, 데이터가 너무 방대하였으며 이 중 악성코드와 직접적인 연관이 없는 데이터도 많아 효율성을 떨어뜨렸고, 악성코드 탐지와 직접적인 관련이 있는 피처를 추출하는 과정이 매우 중요하다고 주장하였다. 본 연구에서는 패킹 여부의 특징으로 악성 행위를 판단하는 현재 실세계에서 사용되고 있는 악성코드 분류기의 한계점을 극복하기 위해 악성코드들이 활용할 수 있는 패커의 분석 방지 보호 기법을 탐지하고 분류해낼 수 있는 모델을 제안한다.

Zhang 등[15]의 연구에서는 랜섬웨어의 n-gram opcode를 사용해서 랜섬웨어의 종류를 분류하는 머신러닝 모델을 구현하였다. 하지만 해당 연구는 랜섬웨어에 한정된다는 한계점이 있다. 랜섬웨어는 데이터를 암호화하고 복호화해주는 대신 랜섬을 요구하는 공통된 특징이 있다. 그러므로 암호화 루틴을 활용해서 n-gram opcode를 추출할 수 있으므로, 트로이목마와 같은 다른 악성코드에는 적용되기 어렵다는 한계점을 지닌다. 본 연구에서는 n-gram opcode를 추출하여 다수의 머신러닝 모델을 학습시켜 악성코드에 활용될 수 있는 패커의 분석 방지 보호 기법을 탐지하고 분류하는 모델을 제시한다.

Park 등[16]은 악성코드들이 기존 악성코드를 변형시켜 제작되어 악용되고 있음에 착안하여 악성코드의 종류와 버전을 식별해내는 연구를 수행하였다. 이때 기존의 연구는 패킹된 악성코드를 다루지 않았지만, 해당 연구는 패킹된 악성코드를 포함하여 수집한 악성코드를 종류별로 분류한 후, 같은 종류의 악성코들에 대해서 클러스터링을 수행하여 버전별로 구별해 내는 작업을 수행하였다. 이때, 클러스터링 수행 시,악성코드들의 공통된 피처가 아닌, 악성코드의 각 종류별 피처를 추출하여 이를 머신러닝 모델의 피

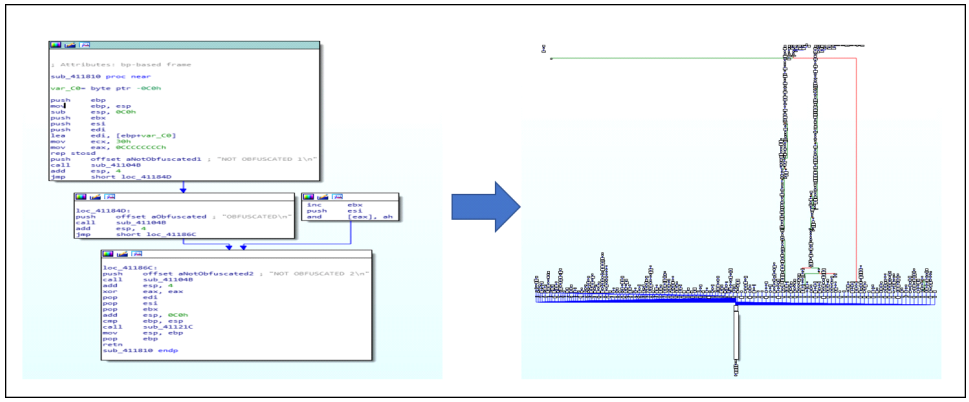

Fig. 2. Control flow graph after applying virtualization obfuscation

처로 활용하였다. 이때 패킹이 된 악성코드들은 패킹이 되지 않은 악성코드와 행위 측면에서는 실행 시 같은 행위를 수행하므로, Cuckoo Sandbox[24]를 활용한 동적분석을 통해 행위정보를 추출하고 벡터화하여 피처로 활용하였다. 하지만 악성코드는 실행 시 정상 프로그램과 달리 실행시간이 매우 길거나 백그라운드에서 멈추지 않고 계속해서 실행되는 경우가 존재한다. 따라서 Cuckoo Sandbox를 활용하여 동적으로 데이터셋을 구축하면 많은 시간이 소요된다. 실제로 Themida로 패킹된 악성코드 50개를 MalwareBazaar[23]에서 yara 룰을 사용하여 수집한 후, Cuckoo Sandbox를 활용하여 [16] 에서 수집한 피처 정보를 추출하는데 걸린 시간을 측정하였다. 이때 타임아웃 10분을 주고 측정하였는데, 50개 샘플 전부 타임아웃을 초과하여 [16]에서 추출한 피처 정보를 얻을 수 없었다. 반면 본 연구에서는 타임아웃 30초를 주어 피처를 추출하였음에도 높은 정확도를 보여주는 머신러닝 모델을 구축할 수 있었다.

Cesare 등 [17]의 연구에서는 MalWise 툴을 개발하여 악성코드를 식별하였다. Cesare 등[17]은 79%의 악성코드가 패킹되었다고 언급하고 있으며, 50%의 악성코드들이 기존의 악성코드에 리패킹이 수행된 악성코드임을 밝히고 있다. 따라서 패킹된 악성코드를 연구대상으로 선정하여, 패킹된 악성코드에 대해서 언패킹 작업을 진행한 후, 제어 흐름 그래프를 사용하여 악성코드 변종, 즉 같은 종류의 악성코드를 식별하고 분류해내는 연구를 수행하였다. 해당 연구는 제어 흐름 그래프를 작성한 후 이를 문자열 시그니처로 변환하여 유사도를 비교함으로써 악성코드를분류하였다. 하지만 많은 난독화 도구에서 사용하는 기법인 코드 가상화 기법이 적용될 경우, 제어 흐름그래프를 이용하여 유사도를 파악하는 것에는 한계점이 있다. 소스코드 가상화 기법은 기존의 소스코드를 가상화시켜 컴파일하여 분석을 방해하는 기법으로 그림 2는 가상화 난독화 기법을 제공하는 소스코드 난독화 도구 중 가장 대표적인 도구인 Code Virtualizer[25]로 난독화를 적용시켰을 때의 제어 흐름 그래프의 변화를 나타낸 것이다. 제어흐름 그래프가 그림 2와 같이 변형된 경우, [17]에서 제시한 방법으로 악성코드를 분류하는 데에 한계점이 존재한다. 하지만 소스코드 가상화 난독화 기술이 적용되었다고 하더라도 본 연구에서 머신러닝 모델을 통해 식별하고자 하는 Anti-Patching, Anti-Dumping, Anti-VM과 같은 패커의 분석 방지 보호 기법은 컴파일 이후에 적용되는 보호 기법이므로, 가상화 난독화 보호 기법에 영향을 받지 않고 피처를 추출할 수 있다.

III. 제안 방법

3.1 연구 대상

3.1.1 상용 패커

본 연구에서는 악성코드를 제공하는 MalwareBazaar로부터 2021년 9월부터 2022년 1월까지 총 4개월 동안 발견된 악성코드를 수집하였다. 그 중 패킹된 악성코드는 989개로, 그림 3은 989개의 악성코드들이 사용한 패커 중 가장 많이 사용된 상위 5개 상용 패커를 나타낸 것이다. 주황색으로 표시한 패커는 .NET 프로그램에 대한 패커이고, 파란색으로 표시

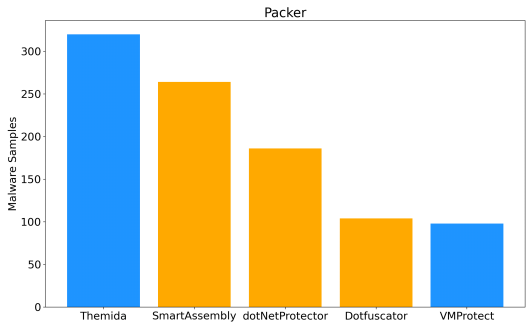

Fig. 3. Commercial packers used by most malwares ('21.9.~'22.1.)

한 Themida와 VMProtect 패커는 EXE 실행 파일에 대해 패킹 및 분석 방지 보호 기법을 제공한다. 따라서 본 연구에서는 EXE 실행 파일에 대해 패킹을 수행하는 상용 패커 중 국내외에서 악성코드 패킹에 가장 많이 사용되는 Themida와 VMProtect를 실험 대상으로 선정하였다.

3.1.2 분석 방지 보호 기법

악성코드 제작자는 상용 패커의 분석 방지 보호 기법을 악성코드에 적용시켜 악성코드의 분석을 방해할 수 있다. 상용 패커들은 여러 분석 방지 보호 기법을 제공하는데, 그 중 표 1과 같이 공통적으로 제공되는 분석 방지 보호 기법인 Anti-Patching, Anti-Dumping, Anti-VM 보호 기법을 연구 대상으로 선정하였다.

Anti-Debugging 보호 기법의 경우, 본 연구에서 데이터셋을 구축하는데 사용하는 DBI의 일종인 PinTool[21]의 동작을 방해하므로, Anti-Debugging 보호 기법은 본 실험에서 제외하였다.

Table 1. Anti-analysis techniques provided by commercial packers

	Anti Debug	Anti Patch	Anti Dump	Anti VM
Themida	O	O	O	O
Obsidium	O	O	X	O
PEtite	X	X	X	X
UPX	X	X	X	X
MPRESS	X	X	X	X
PELock	O	O	O	O
VMProtect	O	O	X	O
Enigma	O	O	X	O

분석가는 동적 분석을 진행할 때, 일반적으로 프로그램의 코드와 데이터를 수정하고 추가함으로써 분석을 진행하는데, 악성코드 제작자는 이러한 코드 패칭을 통한 동적 분석을 방지하기 위하여 Anti-Patching 보호 기법을 악성코드에 적용할 수 있다[18]. Anti-Patching 보호 기법이 적용되어 있을 경우, 코드를 1바이트라도 수정을 한 후 실행을 시키면, 바이너리가 정상적으로 실행되지 않는다.

또한 악성코드의 정적 분석에서 메모리 덤프 파일은 매우 중요하게 사용된다. 분석가는 메모리 덤프 파일을 통하여 악성코드가 사용한 함수의 매개변수 및 언패킹이 완료된 주요 데이터를 얻을 수 있으므로 악성코드 제작자는 메모리 덤프 파일 생성을 통한 정적 분석을 방지하기 위하여 악성코드에 Anti-Dumping 보호 기법을 적용할 수 있다. Anti-Dumping 보호 기법이 적용되어 있을 경우, 분석가는 메모리 덤프 툴을 활용한 메모리 덤프 생성에 실패하게 된다.

Anti-VM 기법은 가상머신에서의 프로그램 실행을 막아주는 보호 기법이다. 악성코드는 실행 시 분석환경에 악영향을 미칠 수 있으므로 분석가는 주로 가상환경에서 악성코드 분석을 하게 되는데, 이러한 분석을 막기 위하여 악성코드 제작자는 악성코드에 Anti-VM 보호 기법을 적용한다. Anti-VM 보호 기법이 적용되어 있을 경우, VMware 혹은 Virtual Machine 등의 가상머신 정보를 문자열과 같은 가상머신의 시그니처 정보와 IN 명령어를 활용하는 방법 등으로 가상머신 환경을 감지하여 실행 중인 바이너리를 강제 종료시킨다[6,8].

3.2 N-gram opcode 기반 분석 방지 보호 기법 피처 추출 방법

3.2.1 개요

본 연구에서는 n-gram opcode 기반으로 분석 방지 보호 기법을 탐지하고 분류하는 방법론을 제안하고 이를 구현한다. 그림 4는 본 절에서 서술할 분석 방지 보호 기법 피처 추출 방법에 대한 개요도이다. 첫 번째 단계는 데이터 전처리 단계로 실험 대상 실행 파일에 대해 패커의 분석 방지 보호 기법(anti-patching, anti-dumping, anti-vm)을 적용시켜 데이터셋을 구축한다. 이후 PinTool을 통해 opcode를 동적으로 추출하여 n-gram opcode

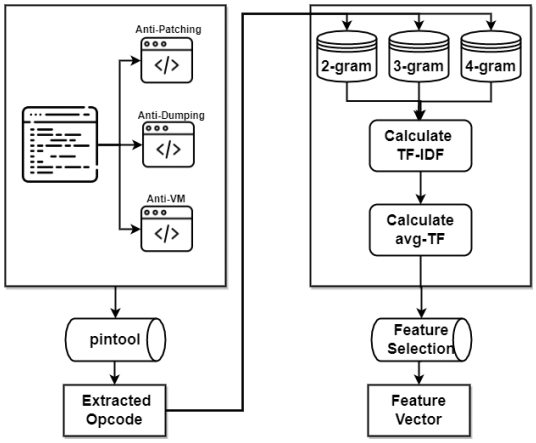

Fig. 4. Overview of Feature Selection

를 생성한다. 두 번째 단계는 유의미한 n-gram opcode 분석 및 추출 단계로, 각각의 n-gram opcode에 대해 3.2.3절에서 제시하는 방식으로 TF-IDF를 계산하여 유의미한 n-gram opcode를 추출한다. 세 번째 단계는 피처 벡터를 추출하는 단계로, avg-TF를 3.2.4절에서 제시하는 방법으로 계산한 뒤, 머신러닝 모델에서 활용할 피처 벡터를 생성한다.

3.2.2 데이터 전처리 방법

상용 패커를 사용해서 각각의 바이너리를 난독화 시키는데, Anti-Patching 보호 기법을 적용한 바이너리(A), Anti-Dumping 보호 기법을 적용한 바이너리(B), Anti-VM 보호 기법을 적용한 바이너리(C), 그리고 어떠한 분석 방지 보호 기법도 적용시키지 않고 패킹만 진행한 바이너리(D) 총 4가지 방식으로 데이터셋을 구축한다. 각각의 4가지 방식으로 구축한 데이터셋에 대해서 opcode를 추출하는데, 패킹이 되어있는 상태이기 때문에 정적으로 추출하는 opcode는 의미가 없으므로 동적으로 실행시켜서 PinTool을 활용하여 opcode를 추출한다. A,B,C,D 데이터셋에서 추출한 opcode를 종합하여 2-gram, 3-gram, 4-gram 각각의 전체 n-gram opcode 종류와 개수를 구한다.

3.2.3 유의미한 n-gram opcode 분석 및 추출 방법

$$TF(i,m) = n(i,B_m),\ m \in (A,B,C,D) \quad (1)$$

$$IDF(i,m) = \frac{\sum_{m\in(A,B,C,D)} |B_m|}{\sum_{m\in(A,B,C,D)} |j : i \in j \mid j \in B_m| + 0.1} \quad (2)$$

3.2.2절에서 구한 전체 n-gram opcode는 하나의 바이너리 안에서도 다양한 종류와 많은 개수로 들어있다. 따라서 3.2.2절에서 전처리를 마친 n-gram opcode 중에서 유의미한 n-gram opcode를 분석하는 것이 중요하다. 본 연구에서는 TF-IDF를 사용하였다. TF(Term Frequency)는 수식 1과 같이 계산하고, TF(i,m)은 해당 n-gram opcode(i)가 특정 분석 방지 보호 기법(m)이 적용된 전체 바이너리 (B_m) 안에서 몇 번 등장하는지를 나타낸다.

IDF(Inverse Document Frequency)는 DF(Document Frequency)의 역수값으로 계산하고 수식 2와 같이 계산한다. DF는 해당 n-gram opcode(i)가 특정 분석 방지 보호 기법(m)이 적용된 전체 바이너리에서 몇 개의 바이너리에 등장하는지를 나타낸다. j는 특정 분석 방지 보호 기법(m)이 적용된 전체 바이너리 집합 중에 n-gram opcode i를 가지고 있는 바이너리를 의미한다. 여러 바이너리에서 동일하게 나타나는 n-gram opcode는 특정 분석 방지 보호기법에서만 나타나는 특징으로 볼 수 없기 때문에 유의미하지 않은 n-gram opcode가 된다. 따라서, 보편적으로 등장하는 n-gram opcode를 제외시켜주기 위해 DF는 역으로 계산해서 IDF를 구성한다. 이때 분모가 0이 될 수도 있으므로 분모 값에 0.1을 더해준다. TF에 IDF를 곱한 TF-IDF값을 3.2.2절에서 구한 각각의 n-gram opcode마다 계산을 해준다.

이후 다음과 같은 조건을 사용해서 유의미한 n-gram opcode를 추출한다. 먼저 n-gram opcode i에 대해서 DF(i,D)의 값이 전체 실험 대상 바이너리 개수의 과반수보다 크면서 $m \in (A,B,C)$에 대해서 DF(i, m)의 값이 크면 이는 패킹 자체 혹은 일반적인 바이너리를 실행했을 때의 n-gram opcode 특성일 가능성이 높으므로 이를 제외시켜준다. 또한 $m \in (A,B,C)$일 때, DF(i, m)의 값이 0인데, DF(i, D)의 값이 0이 아니면 해당 특정 분석 방지 보호 기법만의 특성이므로 매우 유의미한 피처가 될 수 있으나, DF(i, D)의 값이

작으면 해당 분석 방지 보호 기법만의 특성이라고 보기 어려우므로 이는 오히려 특정 분석 방지 보호 기법의 피처를 추출하는데 방해가 될 수 있으므로 이를 제거한다. 마찬가지로, DF(i,D)의 값이 0인데, $m \in (A,B,C)$일 때, DF(i, m)의 값도 작은 경우를 제외시켜 준다. 이렇게 필터링 된 n-gram opcode를 각각의 분석 방지 보호 기법별(A,B,C)로 TF-IDF 값을 기준으로 내림차순 정렬한다. 내림차순 정렬한 결과를 A,B,C별로 10분위로 나누어 각각의 분위마다 A,B,C 결과를 랜덤하게 섞은 후 중복을 제거하여 유의미한 n-gram opcode를 추출한다.

3.2.4 avg-TF를 활용한 피처 벡터 추출 방법

3.2.3절에서 추출한 유의미한 n-gram opcode에 대해서 각각 avg-TF 값을 계산하여 머신러닝 모델에 활용할 피처 벡터를 추출한다. 이때 실험에 사용되는 각각의 바이너리의 특성에 따라 혹은 적용된 분석 방지 보호 기법에 따라 TF 값의 기준이 다를 수 있으므로 각각의 분석 방지 보호 기법이 적용된 전체 바이너리의 n-gram 시퀀스 중 해당 n-gram opcode가 몇 번 등장하는지를 나타내는 avg-TF 값을 수식 3과 같이 계산한다. 여기서 k는 전체 n-gram opcode의 종류를 의미한다.

$$avg-TF(i,m) = \frac{n(i,B_m)}{\sum_k n(k,B_m)}, m \in (A,B,C,D) \quad (3)$$

3.2.3절에서 내림차순으로 정렬하여 추출한 유의미한 n-gram opcode 중 t개의 n-gram opcode를 피처로 선택하는데, 이후 각각의 t개의 n-gram opcode 피처에 대해서 avg-TF를 모두 계산한 후, 하나의 벡터를 구성하여 피처 벡터를 만들어준다. 이때 피처 벡터의 차원은 3.2.3절의 결과에서 선택된 피처의 개수인 t와 같게 된다.

IV. 실 험

4.1 실험 대상 및 머신러닝 기반 학습 모델 구축

실험 대상으로는 MalwareBazaar로부터 2022년 1월에 발견된 악성코드 중 EXE 실행파일 200개를 수집하였다. Themida는 Anti-Patching, Anti-Dumping, Anti-VM 보호 기법을 모두 제공하므로 200개의 악성코드 바이너리에 대해 3가지 분석 방지 보호 기법 및 분석 방지 보호 기법 없이 패킹만 적용시키는 방식까지 총 4가지 방식으로 패킹을 진행한 후, PinTool을 활용하여 opcode를 추출하였다. 그 결과 160개의 바이너리가 오류 없이 Anti-Patching, Anti-Dumping, Anti-VM 보호 기법 모두에 대해 패킹이 이루어졌고, opcode를 추출할 수 있었다. 따라서 160개의 악성코드 데이터 중 140개는 training 데이터로 20개는 test 데이터로 구성된 데이터셋을 구축하였다.

VMProtect는 Anti-Dumping 기능을 제공하지 않으므로, Anti-Patching과 Anti-VM 보호 기법을 200개의 바이너리에 각각 적용시켰고, 143개 바이너리에 대해 오류 없이 보호 기법을 적용시킨 후, opcode를 추출할 수 있었다. 따라서 VMProtect에 대해서는 120개의 training 데이터와 23개의 test 데이터로 구성된 데이터셋을 구축하였다.

3.2.4절에서 추출한 피처 벡터를 학습 피처로 사용하고, 대상 바이너리에 어떤 분석 방지 보호 기법(A,B,C,D)이 적용되었는지를 라벨로 사용해서 지도학습으로 모델을 학습시켰다.(VMProtect의 Anti-Dumping 보호 기법을 제공하지 않으므로 A,B,D 적용) n-gram은 2-gram, 3-gram, 4-gram으로 실험을 진행하였다. 또한 각각의 머신

Table 2. Parameters used in 6 machine learning models

Model	Parameters
Random Forest	max_features='sqrt' n_estimators=300 max_depth=6 min_samples_leaf=16 min_samples_split=8 random_state=0
Naive-Bayes	var_smoothing=1e-09
KNN	k=8
Decision Tree	random_state=0 max_depth=50 max_features='sqrt'
Gradient Boosting	random_state=0
SVM	kernel='linear' C=1.0 random_state=0

러닝 모델에 대해서 2-gram, 3-gram, 4-gram인지에 따라 학습효과를 극대화시키는 파라미터가 달라지고 t의 값에 따라서도 달라지므로 표 2에 제시한 파라미터로 고정시킨 뒤 6개의 머신러닝 모델을 학습시켰다.

4.2 실험 결과 및 분석

본 연구에서는 4.1절에서 제시한 6개의 머신러닝 모델을 학습시키고 평가하여 가장 최적의 머신러닝 모델을 구현하기 위해 정확도, 정밀도 및 F1-score를 평가 기준으로 사용하였다. 3.2.3절에서 제시한 유의미한 n-gram opcode 추출을 수행한 결과, Themida의 경우 1384개의 2-gram opcode, 10877개의 3-gram opcode, 55028개의 4-gram opcode를 추출할 수 있었고, VMProtect의 경우 5609개의 2-gram opcode, 45731개의 3-gram opcode, 138387개의 4gram opcode를 추출할 수 있었다. 그림 5와 그림 6은 각각 Themida와 VMProtect의 분석 방지 보호 기법을 적용시켰을 때, 6개 머신러닝 모델에 대해서 2-gram, 3-gram, 4-gram에 따라 선택된 피처의 개수인 t 값를 증가시키면서 측정한 정확도의 변화를 보여준다. Themida의 경우 2-gram일 때, 전반적으로 가장 낮은 정확도를 보였다. 이는 Themida에서는 2-gram으로는 의미 있는 opcode를 추출할 수 없음을 의미한다. Themida에서는 Gradient Boosting을 사용한 4-gram 머신러닝 모델에서 (t=55000)일 때 정확도 81.25%, 정밀도 0.876, F1-score 0.792로 가장 좋은 성능을 보였다. 표3 (좌)는 그 결과를 나타낸다.

VMProtect는 Gradient Boosting을 사용했을 때, 2-gram과 3-gram 머신러닝 모델에서 높은 정확도를 보여주었다. 이는 Themida에서와 달리

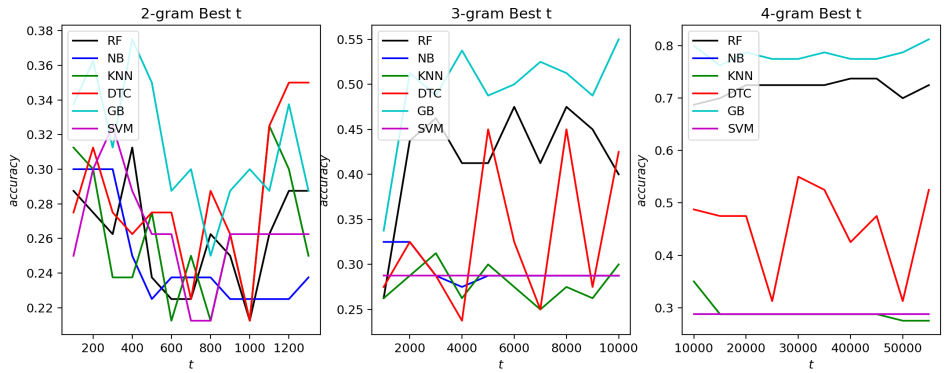

Fig. 5. *Accuracy* with increasing t value in 2-gram, 3-gram, and 4-gram (Themida)

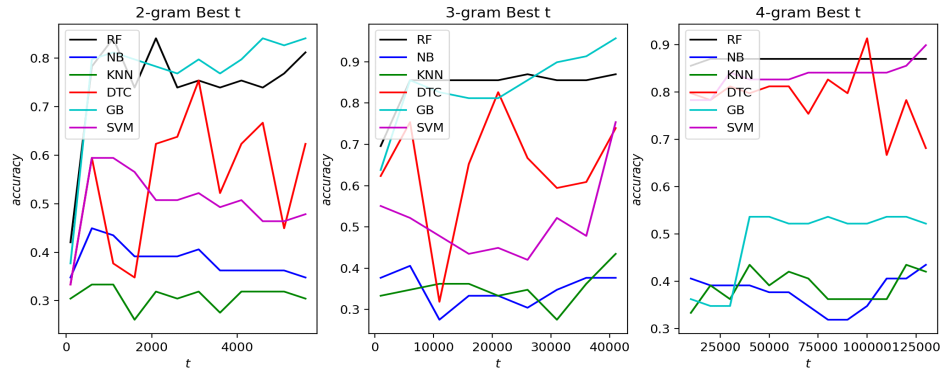

Fig. 6. *Accuracy* with increasing t value in 2-gram, 3-gram, and 4-gram (VMProtect)

2-gram으로도 유의미한 opcode를 추출할 수 있음을 나타낸다. 그 중 가장 높은 정확도를 보인 것은 3-gram(t=41000)일 때이고, 이 경우 정확도 95.65%, 정밀도 0.958, F1-score 0.956으로 가장 좋은 성능을 보였다. 표 3(우)는 그 결과를 나타낸다.

전반적으로 Themida와 VMProtect 모두에서 Gradient Boosting 모델과 Random Forest 모델이 높은 정확도를 보여주었다. 두 모델은 여러 모델의 결과를 모아 최종 결과를 내어 성능을 향상시키는 앙상블 학습 기법을 사용하고 있으므로, 본 연구에는 앙상블 학습 기법을 활용하는 모델이 적합함을 알 수 있다. 한편 Naive Bayes 모델은 Themida와 VMProtect 모두에서 가장 좋지 않은 성능을 보였다. 이는 Naive-Bayes 모델은 학습에 사용된 모든 피처가 서로 독립적이라고 가정하지만, 본 연구에서 피처로 사용한 n-gram opcode는 서로 상당히 의존적이므로 Naive-Bayes 모델을 본 연구에 적합하지 않음을 알 수 있다.

본 연구에서 구축한 머신러닝 모델의 효율성을 평가하기 위해 머신러닝 모델 학습과 테스트에 필요한 시간을 분석하였다. 데이터셋 구축에 소요되는 시간은 5단계로 이루어진다. 1단계는 분석 방지 보호 기법이 적용된 악성코드 샘플에 대해 opcode를 추출하는 단계이다. 이때 악성코드를 동적으로 실행시키면서 opcode를 추출하였는데, 악성코드는 정상 파일과 달리 실행이 오래 걸리는 경우가 많았으며, 실행이 멈추지 않고 무한루프를 도는 경우도 존재하여 opcode를 추출하는 데에 제한이 있었다. 따라서 데이터셋을 구축할 때, 악성코드 실행이 끝날 때까지 opcode를 추출하는 것이 아닌 타임아웃 30초를 주어 타임아웃 시간 동안만 opcode를 추출하여 데이터셋을 구축하였다. 2단계는 추출한 opcode를 n-gram opcode로 변환하는데 걸리는 시간이다. 표 4는 Themida로 패킹된 악성코드 160개 전체 및 VMProtect로 패킹된 악성코드 143개 전체에 대해 2단계에서 소요된 시간을 나타낸 것이다. 3단계는 2단계에서 변환한 n-gram opcode에서 TF와 DF값을 계산하고, 이를 사용해서 TF-IDF값을 계산하는데 걸리는 시간이다. 4-gram의 경우 추출된

Table 3. *Accuracy, Precision, F1-score* with the best t for Themida (4gram, t=55000), and for VMProtect(3-gram, t=41000)

	Themida(4-gram, t=55000)			VMProtect(3gram, t=41000)		
	Accuracy	Precision	F1-score	Accuracy	Precision	F1-score
Random Forest	72.50%	0.820	0.679	86.95%	0.876	0.866
Naive-Bayes	28.75%	0.314	0.168	37.68%	0.279	0.294
KNN	27.50%	0.284	0.169	43.47%	0.476	0.415
Decision Tree	52.50%	0.522	0.522	73.91%	0.740	0.736
Gradient Boosting	81.25%	0.876	0.792	95.65%	0.958	0.956
SVM	28.75%	0.382	0.212	75.36%	0.771	0.747

Table 4. Time required for constructing dataset-1

	Anti-Patching(A)		Anti-Dumping(B)		Anti-VM(C)		None(D)	
	Themida	VMP	Themida	VMP	Themida	VMP	Themida	VMP
transform n-gram opcode	15.55s	39.28s	12.81s	-	12.87s	24.20s	12.89s	39.24s
collect TF, DF & calculate TF-IDF	33.09s	74.00s	28.43s	-	29.15s	45.83s	28.26s	70.31s

Table 5. Time required for constructing dataset-2

	2-gram		3-gram		4-gram	
	Themida	VMP	Themida	VMP	Themida	VMP
select meaningful feature n-gram opcode	0.02s (1384 n-gram)	0.17s (5609 n-gram)	1.15s (10877 n-gram)	11.25s (45731 n-gram)	35.09s (55028 n-gram)	100.11s (138387 n-gram)
construct n-gram feature vector	training: 21.67s	training: 36.24s	training: 27.49s	training: 46.24s	training: 35.15s	trainins: 59.06s
	test: 3.02s	test: 6.61s	test: 3.90s	test: 8.09s	test: 4.91s	test: 10.39s

Table 6. Time required for training and test

	2-gram		3-gram		4-gram	
	Themida	VMP	Themida	VMP	Themida	VMP
Random Forest	0.190s	0.206s	0.351s	0.513s	1.047s	1.107s
Naive-Bayes	0.006s	0.021s	0.047s	0.149s	0.265s	0.488s
KNN	0.014s	0.028s	0.067s	0.136s	0.239s	0.431s
Decision Tree	0.007s	0.018s	0.034s	0.119s	0.183s	0.373s
Gradient Boosting	2.152s	4.212s	18.689s	41.830s	172.664s	129.276s
SVM	0.166s	0.360s	1.425s	1.800s	8.670s	7.019s

n-gram opcode의 종류가 많아 가장 오래 걸렸지만 최대 1분 내외로 소요되었다. 4단계는 TF-IDF 값을 고려하여 2-gram, 3-gram, 4-gram에 대한 유의미한 n-gram opcode 피처를 선택하는데 걸리는 시간이다. 5단계는 학습 데이터와 테스트 데이터에 대해 n-gram 피처 벡터를 생성하는데 걸리는 시간이다. 표 4는 2단계와 3단계에 소요된 시간을 나타내고, 표 5는 4단계와 5단계에 소요된 시간을 나타낸다. 표 6은 각 머신러닝 모델에 대한 학습 및 테스트에 소요된 평균 시간을 2-gram, 3-gram, 4-gram별로 보여준다. Gradient Boosting을 사용한 머신러닝 모델은 다른 모델에 비해 시간이 많이 소요되었지만 최대 3분 내로 학습 및 테스트를 완료하였으며, 나머지 모델은 10초 내로 매우 짧은 시간 내에 학습 및 테스트를 완료하여 본 모델이 효율적으로 활용될 수 있음을 확인할 수 있었다.

V. 결 론

본 연구에서는 n-gram opcode를 활용하여 상용 패커의 분석 방지 보호 기법을 탐지하고 분류해내는 머신러닝 모델을 제안하였다. 악성코드 제작자는 악성코드의 분석을 막기 위해 패커의 여러 분석 방지 보호 기법을 악성코드에 적용하여 배포한다. 시그니처 기반으로 패커의 분석 방지 기법을 탐지하는 방식은 공격자가 쉽게 우회할 수 있는 한계점을 지니며, 시그니처 기반으로 상용 패커의 분석 방지 보호 기법을 우회할 수 있는 연구도 수행된 바 있다. 따라서 본 연구에서는 공격자가 우회하기 어려운 방식인 n-gram opcode의 분포 패턴에 기반하여 Anti-Patching, Anti-Dumping, Anti-VM 기법을 탐지하고 분류해낼 수 있는 최적의 머신러닝 모델을 구축하였다.

본 연구에서는 실제 악성코드 샘플에 대해 6개의 머신러닝 알고리즘을 사용하여 6개의 모델 각각에 대해서 2-gram, 3-gram, 4-gram에서의 n-gram opcode 피처를 추출하여 학습을 진행한 결과, Themida에 대해서는 Gradient Boosting을 활용한 머신러닝 모델에서 81.25%의 정확도와 0.876 정밀도를, VMProtect에 대해서는 95.65%의 정확도와 0.958 정밀도를 보여주는 머신러닝 모델을 구축하였다. 본 연구에서는 악성코드에 가장 많이 활용되는 패커와 해당 패커들이 공통적으로 제공하는 분석 방지 보호 기법을 연구 대상으로 선정하였

다. 악성코드에 적용될 수 있는 패커의 보호 기법을 단시간 내에 탐지하고 분류하여 식별해낼 수 있다는 점에서 악성코드 탐지 및 분석에 효과적으로 활용될 수 있을 것이다. 하지만 본 연구에서 발견한 각 분석 방지 보호 기법의 n-gram opcode 피처를 악성코드 제작자가 식별할 경우, 이를 활용하여 바이너리 동작에 영향을 주지 않고 해당 n-gram opcode를 바이너리에 주입함으로써 적대적 공격을 수행할 수 있다. 이에 대한 해결방안을 고안하는 것이 향후 과제이다.

References

[1] Malware Statistics & Trends Report | AV-TEST, "Total malware", https://www.av-test.org/en/statistics/malware/, 4 Oct. 2021.

[2] Woo-Jin Joe and Hyong-Shik Kim, "Malware Family Detection and Classification Method Using API Call Frequency," Journal of The Korea Institute of Information Security & Cryptology, 31 (4), pp. 605-616, Aug. 2021.

[3] Brand M, Valli C, Woodward A, "Malware forensics: Discovery of the intent of deception," Journal of Digital Forensics, Security and Law, vol. 5, no. 4, pp. 1-13, 2010.

[4] Barbosa, Gabriel Negreira, and Rodrigo Rubira Branco, "Prevalent characteristics in modern malware," Black Hat USA Conference, pp. 1-72, Aug. 2014.

[5] Branco, Rodrigo Rubira, Gabriel Negreira Barbosa, and Pedro Drimel Neto, "Scientific but not academical overview of malware anti-debugging, anti-disassembly and anti-vm technologies," Black Hat USA Conference, pp. 1-27, Aug. 2012.

[6] Chen, Ping, et al, "Advanced or not? A comparative study of the use of anti-debugging and anti-VM techniques in generic and targeted malware," IFIP International Conference on ICT Systems Security and Privacy Protection, vol. 471, pp. 323-336, May. 2016.

[7] Oyama, Yoshihiro, "Trends of anti-analysis operations of malwares observed in API call logs," Journal of Computer Virology and Hacking Techniques, vol. 14, no. 1, pp. 69-85, Feb. 2018.

[8] Lee, Young Bi, Jae Hyuk Suk, and Dong Hoon Lee, "Bypassing Anti-Analysis of Commercial Protector Methods Using DBI Tools," IEEE Access, vol. 9, pp. 7655-7673, Jan. 2021.

[9] Virus Total, "Virus Total Online", https://www.virustotal.com/gui/home/upload, 4 Oct. 2021.

[10] Pechaz, Bassir, Majid Vafaie Jahan, and Mehrdad Jalali, "Malware detection using hidden Markov model based on Markov blanket feature selection method," International Congress on Technology, Communication and Knowledge (ICTCK), pp. 558-563, Nov. 2015.

[11] Hansen, Steven Strandlund, et al, "An approach for detection and family classification of malware based on behavioral analysis," International conference on computing, networking and communications (ICNC), IEEE, pp. 1-5, Feb. 2016.

[12] Kwon, Iltaek, and Eul Gyu Im, "Extracting the representative API call patterns of malware families using recurrent neural network," Proceedings of the International Conference on Research in Adaptive and Convergent Systems, pp. 202-207, Sep. 2017.

[13] Raff, Edward, et al, "An investigation of byte n-gram features for malware classification," Journal of Computer Virology and Hacking Techniques, vol. 14. no. 1, pp. 1-20, Feb. 2018.

[14] Aghakhani, Hojjat, et al, "When Malware is Packin'Heat; Limits of Machine Learning Classifiers Based on Static

Analysis Features," Network and Distributed Systems Security (NDSS) Symposium, pp. 1-20, Jan. 2020.
〔15〕 Zhang, Hanqi, et al, "Classification of ransomware families with machine learning based on N-gram of opcodes," Future Generation Computer Systems, vol. 90, pp. 211-221, Jan. 2019.
〔16〕 Park, Leo Hyun, et al, "Birds of a Feature: Intrafamily Clustering for Version Identification of Packed Malware," IEEE Systems Journal vol. 14. no. 3, pp. 4545-4556, Jan. 2020.
〔17〕 Cesare, Silvio, Yang Xiang, and Wanlei Zhou, "Malwise—an effective and efficient classification system for packed and polymorphic malware," IEEE Transactions on Computers, vol. 62, no. 6, pp. 1193-1206, Jun. 2013.
〔18〕 Roundy, Kevin A., and Barton P. Miller, "Binary-code obfuscations in prevalent packer tools," ACM Computing Surveys (CSUR), vol. 46, no. 1, pp. 1-32, Oct. 2013.
〔19〕 Ali, Muhammad, et al, "MALGRA: Machine Learning and N-Gram Malware Feature Extraction and Detection System," *Electronics, vol. 9,* no. 11, pp. 1-21, Oct. 2020.
〔20〕 Moskovitch, Robert, et al, "Unknown malcode detection using opcode representation," European conference on intelligence and security informatics, vol. 5376, pp.204-215, Dec. 2008.
〔21〕 Pin 3.7 User Guide, "Pin Documentation", https://software.intel.com/sites/landingpage/pintool/docs/97619/Pin/html/, 4 Oct. 2021.
〔22〕 Themida, "Themida Download and Usage", https://www.oreans.com/Themida.php, 4 Oct. 2021.
〔23〕 MalwareBazaar, "Malware Download Site", https://bazaar.abuse.ch/, 4 Oct. 2021.
〔24〕 Cuckoo Sandbox, "Cuckoo Sandbox Download", https://cuckoosandbox.org/, 4 Oct. 2021.
〔25〕 Code Virtualizer, "Code Virtualizer Download", https://www.oreans.com/CodeVirtualizer.php, 4 Oct. 2021.

〈저자소개〉

김 희 연 (Hee Yeon Kim) 정회원
2020년 2월: 고려대학교 사이버국방학과 졸업
2021년 3월~현재: 고려대학교 일반대학원 정보보안학과(소프트웨어보안) 석박사통합과정
〈관심분야〉 정보보호, 취약점 분석, 악성코드 분석, 난독화

이 동 훈 (Dong Hoon Lee) 종신회원
1983년 8월: 고려대학교 경제학과 졸업
1987년 12월: Oklahoma University 전산학과 석사 졸업
1992년 5월: Oklahoma University 전산학과 박사 졸업
1993년 3월~1997년 2월: 고려대학교 전산학과 조교수
1997년 3월~2001년 2월: 고려대학교 전산학과 부교수
2001년 3월~현재: 고려대학교 정보보호대학원 교수
〈관심분야〉 암호 프로토콜, 암호이론, USN이론, 키 교환, 익명성 연구, PET 기술