Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.31, NO.4, Aug. 2021 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2021.31.4.605

API 호출 빈도를 이용한 악성코드 패밀리 탐지 및 분류 방법*

조 우 진,[1†] 김 형 식[2‡]
[1,2]충남대학교 (대학원생, 교수)

Malware Family Detection and Classification Method Using API Call Frequency*

Woo-Jin Joe,[1†] Hyong-Shik Kim[2‡]
[1,2]Chungnam National University (Graduate student, Professor)

요 약

악성코드는 임의의 프로그램을 대상으로 정확하게 식별할 수 있어야 하지만, 분류 기법을 이용하는 기존 연구들은 제한된 샘플에만 적용할 수 있다는 한계가 있다. 본 논문은 임의의 프로그램으로부터 악성코드 패밀리를 탐지하고 분류하기 위해 API 호출 빈도를 이용하는 방법을 제안한다. 제안 방법은 특정 API에 대한 호출 빈도가 임계값을 넘는지 검사하는 규칙을 정의하고, 해당하는 규칙에 의한 비율 정보를 활용하여 특정 패밀리를 식별하는 것이다. 본 논문에서는 결정트리 알고리즘을 응용하여 학습셋으로부터 특정 패밀리를 가장 잘 식별할 수 있는 값으로 임계값을 결정하였다. 4,443개의 샘플을 이용해 학습셋과 시험셋을 나눠 성능을 측정한 결과 패밀리 탐지의 경우 85.1%의 정밀도와 91.3%의 재현율을 보이고, 분류의 경우 97.7%의 정밀도와 98.1%의 재현율을 보여 악성코드 패밀리를 효과적으로 식별할 수 있음을 확인하였다.

ABSTRACT

While malwares must be accurately identifiable from arbitrary programs, existing studies using classification techniques have limitations that they can only be applied to limited samples. In this work, we propose a method to utilize API call frequency to detect and classify malware families from arbitrary programs. Our proposed method defines a rule that checks whether the call frequency of a particular API exceeds the threshold, and identifies a specific family by utilizing the rate information on the corresponding rules. In this paper, decision tree algorithm is applied to define the optimal threshold that can accurately identify a particular family from the training set. The performance measurements using 4,443 samples showed 85.1% precision and 91.3% recall rate for family detection, 97.7% precision and 98.1% reproduction rate for classification, which confirms that our method works to distinguish malware families effectively.

Keywords: Malware, Family, Detection, Classification, API

I. 서 론

한국랜섬웨어침해대응센터에 따르면, 지난해 랜섬웨어(Ransomware) 공격으로 인한 피해액이 약 9조 원을 넘겼다고 한다[1]. 랜섬웨어는 몸값을 뜻하는 Ransom과 악성코드를 뜻하는 Malware의 합

Received(04. 15. 2021), Modified(06. 14. 2021), Accepted(06. 14. 2021)
* 이 연구는 충남대학교 학술연구비에 의해 지원되었음

† 주저자, woojin.jo95@gmail.com
‡ 교신저자, hkim@cnu.ac.kr(Corresponding author)

성어이며, 실행되면 사용자의 데이터를 모두 암호화 시켜 금전을 요구하는 악성 프로그램이다[2]. 기업의 시스템이 랜섬웨어에 감염된다면 막대한 손해가 발생하기 때문에 정상 프로그램과 구분하여 정확하게 식별하기 위한 연구가 필요하다.

악성코드를 식별하기 위해 실행 파일에서 추출하는 정적 정보를 활용할 수 있다. 문자열이나 고유 함수와 같은 정적 정보의 경우 중복되기 어려워 해당 프로그램을 고유하게 식별할 수 있지만, 패킹이나 난독화와 같은 분석 방해(Anti-analysis) 기법으로 쉽게 변형될 수 있다는 한계점이 있다. 따라서, 기존의 악성코드에서 변형된 변종 악성코드까지 식별하는 방법이 필요하다.

정적 정보의 한계점을 극복하기 위해 악성코드를 실행시켜 수집하는 동적 정보를 활용할 수 있다. 동적 정보의 경우 실행파일을 변형하는 분석 방해 기법에 영향을 받지 않기 때문에 변종 악성코드까지 식별하는 것이 가능하다. 그렇지만, 정적 정보와 달리 동적 정보의 경우 추출할 수 있는 정보가 한정적이어서 식별성이 떨어지고, 그 결과 많은 오탐을 발생시킨다는 문제가 있다.

예를 들어, 동적 정보로써 API(Application Programing Interface) 호출 정보가 있는데, 운영체제에서 지원하는 API는 한정적이기 때문에 서로 다른 프로그램이더라도 같은 API를 호출할 수 있다. 특히, 파일을 읽거나 쓰는 것과 같이 자주 사용되는 API는 중복될 확률이 높다.

정적 정보와 동적 정보의 식별성이 떨어지고, 악성코드의 수가 매해 급증하자 최근 악성코드 관련 연구들은 기계학습을 활용한 방법을 연구하기 시작했다. 특히, 이진 분류(binary classification)를 적용해 정상과 악성을 분류하거나[3][4] 다중 분류(multiple classification)를 적용해 패밀리를 분류하는 방식의 연구[5]가 많이 진행되었다. 그렇지만, 기계학습은 학습셋에 최적화되어 패밀리의 종류가 다양한 악성코드를 대상으로 하기에는 적합하지 않다는 문제가 있다.

본 연구에서는 이러한 문제점들을 극복하기 위해 API 호출 빈도를 이용한 악성코드 패밀리 탐지 및 분류 방법을 제안한다. 제안 방법은 특정 API에 대한 호출 빈도가 임계값을 넘는지 검사하는 규칙을 정의하고, 해당하는 규칙의 비율에 기반하여 특정 패밀리를 식별하는 것이다. 이때, 임계값은 결정트리 알고리즘을 응용하여 학습셋으로부터 특정 패밀리를 가장 잘 식별할 수 있는 값으로 결정한다.

II. 관련 연구

악성코드를 식별하기 위해서는 정적 정보와 동적 정보를 분석해야 하지만, 최근 악성코드의 수가 급증하고, 악성코드 분석을 방해하는 분석 방해 기법이 정교해짐에 따라 악성코드를 직접 분석하기 어려워졌다. 기계학습을 이용한 자동화된 악성코드 분석 방법은 이러한 문제점을 극복하기 위한 대안이 될 수 있다. 이번 절에서는 이러한 기존 연구들을 조사하고 한계점을 분석하여 본 연구의 접근 방향을 제시한다.

악성코드와 관련된 많은 기존 연구들은 기계학습의 일종인 분류 기법을 활용하여 악성코드를 식별했다. 예로, 악성코드와 정상코드를 대상으로 이진 분류(Binary classification)를 적용하여 악성코드를 탐지하거나, 다수의 패밀리를 대상으로 다중 분류(Multiple classification)를 적용하여 패밀리를 분류할 수 있다.

Uppal 외 3인은 API 호출 순서를 이용한 악성코드 탐지 방법을 제안했다[3]. API 호출 순서로부터 엔그램(N-gram)에 기반한 시퀀스(Sequence)를 추출하고, 오즈비(Odds Ratio)를 이용한 특징 선택(Feature selection)을 수행하여 식별성 있는 특징값을 정의했다. 그리고, 추출된 특징값을 4가지 분류 알고리즘에 적용하여 악성코드 탐지 성능을 측정한 결과 SVM 알고리즘이 98.5%로 가장 높은 성능을 보였다.

Hansen 외 3인은 API 호출 정보로부터 추출한 호출 순서, 빈도, 인자를 추출하고, 패밀리별 고유 시그니처를 이용한 악성코드 탐지 및 패밀리 분류 방법을 제안했다[5]. 악성코드를 실행시켜 이러한 정보들을 추출하고, 정보 엔트로피(Information entropy)를 이용한 특징 선택을 수행하여 식별성 있는 특징값을 정의했다. 그리고, 랜덤 포레스트(Random forest) 알고리즘을 이용하여 악성코드 탐지에 대한 성능 측정 결과 98.1%의 F1 score와 99.6%의 정확도를 보였으며, 패밀리 분류에 대한 성능 측정 결과 86.4%의 F1 score와 97.8%의 정확도를 보였다.

Garg 외 1인은 API 호출 여부를 이용한 악성코드 탐지 방법을 제안했다[4]. 특정 API의 호출 여부를 추출하고, 랜덤 포레스트를 이용한 특징 선택

을 수행하여 식별성 있는 특징값을 정의했다. 그리고, 6가지 알고리즘을 이용해 악성코드 탐지를 수행한 결과 SVM 알고리즘이 93% 정확도로 가장 높은 성능을 보였다.

기존 연구들은 악성코드를 직접 분석하는 것에는 한계가 있음을 전제로 기계학습을 이용한 악성코드 분석 방법을 연구했다. 그렇지만, 악성코드는 패밀리의 종류가 다양하므로 학습셋에 최적화되는 기계학습은 적합하지 않다는 문제가 있다.

예를 들면, 분류 기법은 학습한 클래스만 분류할 수 있으므로 다중 분류를 적용해 패밀리를 분류하는 경우 학습한 패밀리만 분류할 수 있다. 즉, 학습하지 않은 새로운 패밀리가 포함된 임의의 샘플을 대상으로는 적용할 수 없어 실용성이 없다.

또한, 악성코드와 정상코드를 학습하는 이진 분류를 적용하는 경우 학습셋을 구성하는 데 많은 노력이 든다는 문제점이 있다. 해당 연구들은 악성코드와 정상코드의 행위에는 차이가 있다는 것을 가정하고, 학습셋을 통해 이러한 행위를 학습하는 것을 목표로 한다. 그렇지만, 악성코드 대부분은 기존의 악성코드를 변형한 변종 악성코드이기 때문에 실제로는 악성 행위가 아닌 특정 패밀리의 행위를 학습한다는 문제점이 존재한다.

이러한 문제점을 극복하려면 기존 연구[5]와 같이 학습셋을 구성할 때 서로 다른 패밀리의 악성코드로만 구성해야 하며, 정상적인 행위를 걸러내기 위해 충분한 정상코드가 필요하다. 반면, 본 연구의 경우 기존 연구들과 다르게 규칙에 해당하지 않으면 식별되지 않기 때문에 임의의 샘플을 대상으로 적용할 수 있으며, 규칙을 정의하기 위한 학습셋은 임의의 샘플로 구성하기 때문에 학습셋을 구성하는 데 적은 노력이 든다.

게다가, 본 연구의 경우 기존 연구와 달리 패밀리 정보까지 알 수 있어 활용도가 더 뛰어나다. 예를 들면, 패밀리에 대한 정보를 알게 되면 악성코드 분석가는 대응방안을 결정하거나, 위험도에 따른 우선순위를 부여하는 데 활용할 수 있다.

III. 제안 방법

3.1 API 호출 빈도 기반 규칙 추출 방법

본 논문에서는 API 호출 빈도를 이용한 악성코드 패밀리 탐지 및 분류 방법을 제안한다. 제안 방법은 특정 API에 대한 호출 빈도가 임계값을 넘는지 검사하는 규칙을 정의하고, 해당하는 규칙의 비율에 기반하여 특정 패밀리를 식별하는 것이다.

이때, 특정 API에 대한 임계값은 특정 패밀리를 가장 잘 식별할 수 있는 값으로 결정해야 한다. 단순하게는 특정 패밀리의 샘플들이 호출한 빈도 중 가장 낮은 빈도를 임계값으로 설정하면 특정 패밀리의 샘플들을 모두 식별할 수 있다.

예시로, 'archsms' 패밀리의 샘플들에 대한 API 호출 빈도는 Table 1.과 같다. Table 1.에서 2열부터 5열까지는 샘플 번호이며, 행은 API 명칭이다. 표를 보면 5개의 샘플들의 빈도가 완전히 일치하지는 않지만, 각 API에 대한 호출 빈도에 일관성이 있음을 확인할 수 있다. 이때, 각 샘플이 'NtProtectVirtualMemory' API를 호출한 빈도 중 가장 작은 169를 임계값으로 설정하면 5개의 샘플을 모두 식별하면서, 169 미만으로 호출한 샘플들을 걸러낼 수 있다.

그렇지만, 같은 패밀리에 속하더라도 특정 API에 대한 호출 빈도에 큰 차이가 있으면 가장 낮은 빈도를 임계값으로 결정하는 것이 적절하지 않다. 이러한 호출 빈도의 차이는 단순히 패밀리에 대한 라벨(label)이 잘못되어 다른 패밀리 샘플이 포함되었을 수도 있지만, 특정 조건을 만족할 때까지 실행되는 반복문에서 API가 호출될 때도 발생하기 때문이다.

예시로, 'amonetize' 패밀리에 해당하는 5개의 샘플들의 API 호출 빈도를 표로 나타내면 Table 2.와 같다. 표를 보면 1번 샘플이 'RegOpenKeyExW'와 'RegQueryValueExW' API를 호출한 빈도가 다른

Table 1. API frequency of 'archsms' family

API name	1	2	3	4	5
NtProtectVirtualMemory	169	169	169	169	173
SetErrorMode	118	118	118	118	120
NtFreeVirtualMemory	167	173	160	171	173
GetSystemInfo	114	114	114	114	114

Table 2. API frequency of 'amonetize' family

API name	1	2	3	4	5
InternetSetOptionA	19	19	19	19	19
NtReadFile	39	40	26	26	26
RegOpenKeyExW	309	711	749	755	751
RegQueryValueExW	91	435	453	454	455

샘플에 비해 절반도 되지 않는 것을 확인할 수 있다.

위의 사례와 같이 API 호출 빈도가 크게 차이 날 경우 가장 작은 빈도를 임계값으로 정의하는 것은 많은 오탐을 발생시킬 수 있다. Table 2.에서 'RegOpenKeyExW' API를 호출한 빈도 중 가장 작은 309를 임계값으로 설정한다면 5개의 샘플을 모두 식별할 수 있지만, 309번 미만 호출한 프로그램들까지 밖에 걸러낼 수 없다. 반면, 임계값을 711로 설정한다면 비록 1번 샘플은 식별할 수 없지만, 711번 미만 호출한 프로그램들까지 걸러낼 수 있어 많은 오탐을 줄일 수 있다. 따라서, 미탐이 발생하더라도 오탐을 최소화할 수 있는 임계값을 찾아야 한다.

이를 위해 본 연구에서는 기계학습 기법을 이용해 학습셋으로부터 특정 패밀리를 가장 잘 식별할 수 있는 임계값을 결정한다. 이는 결정트리 알고리즘을 응용한 것으로 특정 패밀리의 샘플들이 호출한 빈도를 후보 임계값으로 하여 성능을 측정하고, 가장 높은 성능을 보이는 빈도를 임계값으로 선택하는 것이다. 이때, 비정상적으로 많거나 적게 호출되는 빈도는 성능이 낮아 효과적으로 배제할 수 있다.

호출 빈도에 대한 성능을 측정하기 위해서는 학습셋의 각 샘플에 패밀리에 대한 라벨을 할당해야 한다. 본 연구에서는 라벨을 할당하기 위해 VirusTotal 서비스와 AVClass 프로그램을 이용한다. VirusTotal은 입력된 샘플에 대하여 70개가 넘는 백신의 진단명을 기록한 보고서를 제공하며[6], AVClass는 VirusTotal의 보고서를 입력으로 받아 불필요한 진단명은 제거하고, 과반수의 진단명을 결과로 내어준다[7].

그렇지만, 하나의 프로그램에 대한 진단명은 백신마다 다를 수 있으므로 AVClass를 통해 다수의 백신이 진단한 라벨을 추출했음에도 진단명이 부정확할 수 있다. 예로, 실제로는 같은 패밀리가 아님에도 같은 라벨이 할당될 수가 있는데, 이 경우 행위가 유사하지 않아 Table 2.의 사례와 같이 임계값을 정의할 때 영향을 주게 된다.

따라서, 행위가 유사한 것을 보장하기 위해 라벨이 같은 샘플들을 대상으로 클러스터링을 수행하여 유사한 샘플들을 그룹으로 묶고, 각 그룹을 대상으로 규칙을 정의한다. 이때, 샘플간에 유사성을 측정하기 위해 API 명칭을 기반으로 한 자카드(Jaccard) 유사도를 이용하며, 계층적 클러스터링[8] 알고리즘을 이용해 유사한 샘플들을 묶는다.

자카드 유사도는 식 (1)과 같이 두 집합 A와 B의 유사도를 측정하는 척도로 A와 B의 합집합 중 A와 B의 교집합의 비율이다. 계층적 클러스터링은 유사도가 임계값(α)을 넘는 샘플들을 같은 그룹으로 묶는 방식으로 동작하는데, 임계값을 결정하는 방법은 실험을 통해 보인다.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

행위가 유사한 샘플들을 묶어 그룹을 형성했다면, 각 그룹으로부터 규칙을 정의한다. 이때, 하나의 규칙은 API 명칭과 빈도 임계값으로 정의되어 〈API, 임계값〉의 형태로 표현할 수 있다. 예로, 〈GetTimeW, 300〉이라는 규칙은 'GetTimeW' API를 300번 이상 호출한 프로그램을 탐지한다.

이러한 규칙은 그룹 내의 각 샘플들이 호출한 API들을 합집합하여, 합집합의 각 API에 대해 정의된다. 그 결과 하나의 그룹으로부터 다수의 규칙이 정의되는데, 그룹 내의 샘플 중 절반이 넘는 샘플들이 해당하지 않는 규칙은 식별성이 없으므로 제거한다.

규칙을 정의할 때, API 빈도 임계값은 그룹 내의 샘플들이 호출한 빈도 중에서 선택한다. 예로, 6개의 샘플로 이루어진 그룹에서 'GetTiemW'라는 API를 3개의 샘플이 모두 7번 호출하고, 나머지 3개의 샘플이 모두 9번 호출했다면, 7과 9가 임계값 후보이다. 즉, 〈GetTimeW, 7〉, 〈GetTimeW, 9〉와 같이 2개의 후보 규칙이 정의된다. 그리고, 학습셋을 이용해 각 후보 규칙에 대한 성능을 측정해 성능이 가장 높은 규칙을 최종 규칙으로 생성한다.

후보 규칙의 성능은 식 (2)와 같이 정확도를 이용해 측정한다. 분모는 학습셋에서 X라는 API를 호출한 샘플의 수이며, 분자는 X를 호출한 샘플 중에서 진양성(True Positives)과 진음성(True Negatives)의 합이다. 진양성은 규칙에 해당하는 샘플 중 실제로 양성인 샘플이고, 진음성은 규칙에 해당하지 않는 샘플 중 실제로 음성인 샘플이다. 여기서, 양성은 식별하고자 하는 패밀리의 샘플이고, 음성은 그 외의 샘플이다. 이때, 양성에 해당하는 샘플의 수가 비교적 적을 수 있으므로, 부족한 비율만큼 가중치를 적용하여 계산한다.

$$Acc(X) = \frac{TruePostives + TrueNegatives}{number\,of\,samples\,calling\,X} \quad (2)$$

각 후보 규칙별로 성능을 측정했다면, 성능이 가장 높은 규칙을 최종 규칙으로 정의한다. 그렇지만, 이는 후보 규칙 중에서 가장 성능이 높은 것이지, 실질적인 성능은 낮을 수 있다. 따라서, 본 논문에서는 최종 규칙의 정확도가 0.5 미만일 경우 해당 규칙은 제거하도록 구현했다. 정확도가 0.5 미만이라는 것은 해당 규칙을 이용해 정확하게 식별한 샘플보다 틀린 샘플의 수가 더 많다는 것이기 때문이다.

위와 같은 과정을 통해 하나의 그룹으로부터 다수의 규칙을 정의할 수 있다. 본 연구에서는 하나의 그룹에서 정의된 규칙을 모두 포함하는 규칙 집합을 생성하여 패밀리를 식별하는 데 활용한다. 이때, 하나의 패밀리에는 여러 그룹이 존재하므로 그룹별로 규칙 집합이 생성되고, 각 규칙 집합은 독립적으로 악성코드 패밀리를 식별하는 데 활용된다.

3.2 API 호출 빈도 기반 규칙 추출 방법

하나의 그룹으로부터 API 호출 빈도에 관한 규칙 집합을 생성했다면, 임의의 프로그램을 대상으로 특정 패밀리를 식별할 수 있다. 규칙 집합 내의 규칙들을 검사 대상 프로그램에 적용하면, 해당하는 규칙의 비율에 기반하여 특정 패밀리를 식별할 수 있다. 이때, 해당하는 규칙의 비율이 임계값(β)을 넘으면 특정 패밀리로 식별하는데, 임계값을 결정하는 방법은 실험을 통해 보인다.

규칙 집합은 패밀리가 같은 샘플들로부터 추출한 것이기 때문에 같은 패밀리에 속하는 프로그램은 규칙 집합과 높은 일치율을 보일 것으로 예상할 수 있다. 그렇지만, 많은 행위를 수행하는 프로그램 또한 해당하는 규칙이 많아 오탐으로 탐지되는 문제가 발생한다. 따라서, 많은 행위를 수행하여 식별되는 오탐을 제거할 수 있는 수단이 필요하다.

오탐을 제거하기 위해서는 최종적으로 식별된 프로그램이 규칙 집합이 생성된 그룹에 포함될 수 있는지 검사해야 한다. 가장 단순한 방법으로는 식별된 프로그램과 그룹 내의 모든 샘플의 유사도를 측정하여 임계값(α)을 넘는지 검사하는 것인데, 이는 많은 비용이 발생한다는 문제가 있다.

본 논문에서는 식별된 프로그램이 그룹에 포함될 수 있는지 검사하기 위해 API 개수를 이용한다. 많은 행위를 수행하는 프로그램의 경우 API 개수가 상대적으로 많을 것이기 때문에 그룹에 포함되기 위해 가질 수 있는 API 개수의 최댓값을 정의하면 효과적으로 제거할 수 있다.

그룹에 포함되기 위한 API 개수의 최댓값은 그룹 내의 샘플들 중에서 API 개수가 가장 적은 샘플과 비교하여 정의할 수 있다. 즉, 그룹 내에서 API 개수가 가장 적은 샘플의 API 집합을 M이라 하고, 규칙 집합에 식별된 샘플의 API 집합을 N이라 할 때, 자카드 유사도가 임계값(α)을 넘기 위해 가질 수 있는 집합 N의 최대 크기를 정의하면 된다.

자카드 유사도는 집합 M과 N의 합집합 중에서 교집합의 비율이기 때문에 집합 N은 집합 M을 포함할 때 최대 개수를 가질 수 있다. 집합 N이 집합 M을 포함한다면, 식 (3)과 같이 분모가 집합 N의 개수에만 영향을 받기 때문이다.

$$J(M,N) = \frac{|M \cap N|}{|M \cup N|} = \frac{|M|}{|N|} \quad (M \subset N) \tag{3}$$

그리고, $J(M,N)$는 임계값(α) 이상이어야 하므로, 식 (4)와 같이 작성할 수 있으며, 식 (4)는 식 (5)의 형태로 변환할 수 있다. 즉, 규칙 집합에 식별된 프로그램의 API 개수는 $\frac{|M|}{\alpha}$ 보다 작아야 한다.

$$J(M,N) = \frac{|M|}{|N|} \geq \alpha \tag{4}$$

$$\frac{|M|}{\alpha} \geq |N| \tag{5}$$

예를 들어, 임계값(α)을 0.55으로 가정하고, 집합 M이 12개의 API로 구성된다면 식 (5)에 따라 집합 N의 크기는 21.81보다 작아야 한다. 따라서, 규칙 집합에 탐지된 프로그램 중 API의 개수가 21개를 넘는 프로그램은 오탐으로 제거한다.

IV. 실 험

4.1 실험 환경 및 실험 개요

본 연구의 객관성을 입증하기 위해 2가지 실험을 진행한다. 실험 데이터는 특정 기관에서 제공받은 4,443개의 악성코드를 이용하며, 악성코드를 통제된 환경에서 실행하기 위해 쿠쿠 샌드박스[9]라는

도구를 이용한다. 악성코드는 동일한 환경에서 실행되지 않으면 실행 흐름이 바뀌어 동적 정보가 달라질 수 있으므로 통제된 환경에서 실행시킬 필요가 있는데, 쿠쿠 샌드박스는 가상머신의 스냅샷(snapshot) 기능을 이용해 모든 샘플을 동일한 환경에서 실행한다. 스냅샷은 특정 시점의 시스템 상태를 저장한 것으로, 시스템 설정이나 실행 중인 프로그램을 복원할 수 있다.

본 연구에서는 4,443개의 악성코드 중에서 샘플의 수가 가장 많은 10개 패밀리를 선정하고, 10개 패밀리에 대한 탐지 및 분류를 수행한다. 각 패밀리에 대한 샘플의 수와 타입은 Table 3.과 같으며, 10개의 패밀리에 해당하지 않는 나머지 샘플들의 경우 규칙 집합을 학습하거나 성능을 측정하는데 사용한다.

악성코드 타입은 악성코드의 행위를 유형별로 분석한 것이며, 예로 Backdoor는 보안 설정을 우회하여 시스템에 침입하는 유형의 악성코드를 의미한다. 미국 보안 기업인 크라우드스트라이크(CrowdStrike)는 악성코드 타입을 다음과 같이 정의했다[10]. Adware는 사용자가 원하지 않는 광고를 보여주며, Worm은 자체 복제를 통해 네트워크를 통해 확산하며, Ransomware는 데이터에 대한 접근을 차단하여 사용자에게 금전을 요구한다.

첫 번째 실험은 클러스터링을 수행하기 위한 유사도 임계값을 결정하기 위한 것이다. 본 연구에서 활용하는 계층적 클러스터링 알고리즘은 사용자로부터 주어진 임계값을 기준으로 그룹을 형성하기 때문에 임계값에 따라 결과가 크게 달라진다. 이러한 임계값은 데이터의 성질이나 분석 목적에 따라 적절한 값이 달라지기 때문에 실험을 통해 클러스터링 결과를 분석하여 결정해야 한다.

Table 3. Number of samples per family

Family	Number of samples	Type
berbew	287	Backdoor
gator	284	Adware
wabot	243	Worm
mira	172	Ransomware
multiplug	123	Adware
loadmoney	126	Adware
gate	105	Ransomware
bladabindi	90	Backdoor
installmonster	81	Adware
browsefox	75	Adware
Total	1,582	4

클러스터링 결과를 분석하는 방법은 다양하지만, 본 연구와 같이 데이터 셋에 라벨이 할당되어 있으면 클러스터링 결과가 원래의 클래스 분포와 얼마나 유사한지를 비교할 수 있다. 그렇지만, 악성코드에 할당된 패밀리 라벨은 부정확하다는 문제가 있으므로 우리는 라벨에 의존하지 않고, 다른 방법을 이용해 클러스터링 결과를 분석한다.

본 연구에서는 하나의 패밀리에 대한 규칙 집합을 생성할 때 패밀리 내의 여러 그룹으로부터 추출하기 때문에 하나의 패밀리에서 다수의 규칙 집합이 생성된다. 이때, 여러 그룹에서 생성된 규칙 집합이 차이가 없다면, 불필요한 식별 작업이 수행되기 때문에 의미 있는 규칙 집합을 추출하기 위해서는 다른 그룹 간에 유사성이 낮아야 한다. 또한, 그룹 내의 샘플들은 일관적인 API 빈도를 보여야 하므로 같은 그룹에 속하는 샘플끼리 유사성은 높아야 한다. 따라서, 첫 번째 실험에서는 이러한 요구사항을 충족하는 적절한 임계값을 찾아야 한다.

두 번째 실험에서는 규칙 집합을 적용할 때 적절한 일치율 임계값을 결정하기 위한 것이다. 일반적으로 패밀리가 같은 악성코드들은 행위가 유사하므로 규칙 집합과 높은 일치율을 보일 것으로 예상할 수 있다. 그렇지만, 적절한 임계값을 설정하지 않으면 성능이 좋지 않은데, 예로 임계값을 너무 높게 설정하면 미탐이 발생할 수 있고, 낮게 설정하면 오탐이 발생할 수 있다.

이러한 임계값은 패밀리별로 최적의 값이 다르다. 예로, 특정 패밀리로부터 식별성이 높은 규칙들이 정의되면 오탐이 줄어들기 때문에 최적의 임계값은 낮아지고, 식별성이 낮은 규칙들이 정의된다면 오탐이 많이 발생하여 임계값이 높아진다.

그렇지만, 패밀리별로 임계값을 정의하는 것은 많은 비용이 들기 때문에 대부분의 패밀리에 적용할 수 있으면서 최적의 값에 근접한 임계값을 정의할 필요가 있다. 두 번째 실험에서는 이를 위해 임의의 패밀리를 대상으로 규칙 집합 일치율에 따른 오탐과 미탐의 발생 비율을 분석하고, 최적의 임계값을 정의한다.

마지막으로, 정의된 임계값을 이용해 악성코드 패밀리 탐지 및 분류 작업을 수행하여 기존 연구와 성능을 비교한다.

4.2 계층적 클러스터링 임계값 정의 방법

첫 번째 실험에서는 계층적 클러스터링을 수행할 때 필요한 유사도 임계값(α)을 결정하는 것을 목표로 하는데, 라벨에 의존하지 않고 클러스터링 결과를 분석하여 결정한다. 적절한 규칙 집합을 추출하기 위해서는 같은 그룹에 속한 샘플들끼리는 유사하고, 다른 그룹에 속한 샘플들끼리는 유사하지 않아야 한다. 이를 위해서 우리는 실루엣 계수(Silhouette coefficient) 척도[11]를 이용해 클러스터링 품질을 판단한다.

실루엣 계수는 하나의 샘플에 대한 응집도(같은 그룹에 속한 샘플들과 가까운 정도)와 분리도(다른 그룹에 속한 샘플들과 떨어진 정도)를 측정한 것이며, 전체 샘플에 대한 실루엣 계수를 평균내면 전체적인 클러스터링 품질을 측정할 수 있다. 실루엣 계수 공식은 식 (6)과 같다. a(i)는 i번째 샘플과 i번째 샘플이 속한 그룹 내의 샘플 간에 거리 평균이며, b(i)는 i번째 샘플과 i번째 샘플에 가장 가까운 그룹내의 샘플 간에 거리 평균이다. 이때, 실루엣 계수는 -1부터 1까지 값을 갖는다.

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{6}$$

우리는 유의미한 결과를 얻기 위해 전체 4,443개 샘플 중에서 유사도가 100%인 샘플 쌍을 제거하고 1,182개 샘플에 대해서 실험을 진행했다. 그리고, 적절한 임계값을 정의하기 위해 유사도를 0%부터 100%까지 1%씩 증가시켜 101번의 클러스터링을 수행했고, 전체 샘플의 실루엣 계수를 평균해서 Fig. 1.과 같은 결과를 얻었다.

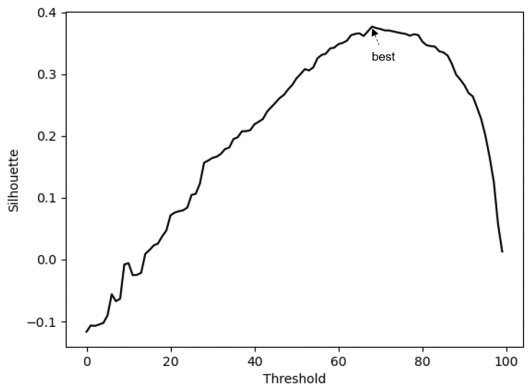

Fig. 1. Silhouette mean per threshold for arbitrary samples

Fig. 1.에서 x축은 유사도 임계값이며, y축은 해당 임계값을 기반으로 클러스터링을 수행하여 측정된 실루엣 계수 평균이다. 실험 결과, 유사도가 69% 일 때 실루엣 계수 평균이 가장 높은 것을 확인할 수 있었다. 일반적으로 군집의 개수를 결정할 때 군집의 개수에 대한 평균 척도(실루엣 계수)를 그래프로 그려서 평가 척도가 최고점에 해당하는 개수로 결정할 수 있다[12]. 본 논문에서도 이러한 접근을 따라 클러스터링을 위한 유사도 임계값(α)은 69%로 설정한다.

4.3 규칙 집합의 일치율에 대한 임계값 정의 방법

두 번째 실험에서는 규칙 집합 일치율에 따른 오탐과 미탐의 발생 비율을 분석하고, 이를 절충하여 적절한 임계값을 정의하는 것을 목표로 한다. 우리는 이를 위해 임의의 패밀리를 대상으로 탐지 성능을 측정하여 평균적으로 성능이 가장 높게 나오는 일치율을 임계값으로 결정한다.

우리는 임의의 패밀리를 선정하기 위해 863개의 패밀리 중에서 Table 3.에서 선정한 패밀리를 제외하고, 10개 이상의 샘플이 존재하는 86개의 패밀리를 선정했다. 그리고, 전체 4,443개의 샘플을 6:4의 비율로 나눠 학습셋(2,665개)과 시험셋(1,778개)으로 설정했으며, 학습셋으로부터 86개의 패밀리에 대한 규칙 집합을 추출했다. 그리고, 추출된 규칙 집합을 시험셋에 적용하여 탐지 성능을 측정했는데, 규칙 집합의 일치율 임계값을 0%부터 100%까지 1%씩 증가시켜 101번의 탐지를 수행하여 Fig. 2.와 같은 결과를 얻었다.

Fig. 2.에서 x축은 일치율 임계값이며, y축은 해당 임계값을 이용해 패밀리 탐지를 수행하여 측정된 성능이다. 본 연구에서는 성능을 측정하기 위한 척도로 F1 score를 이용했는데, F1 score는 정밀도(Precision)와 재현율(Recall)의 조화 평균으로 식 (7)과 같다. 정밀도는 예측한 샘플 중에서 정답인 샘플의 수이며, 재현율은 식별해야 할 샘플 중에서 성공적으로 식별해낸 샘플의 수이다.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{7}$$

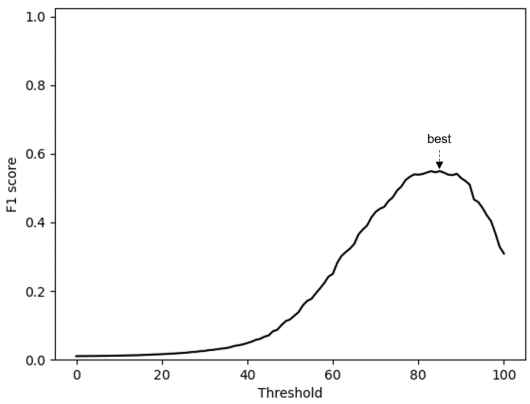

Fig. 2. F1 score per threshold for arbitrary families

실험 결과, 일치율이 85%일 때 F1 score가 가장 높은 것을 확인할 수 있었다. F1 score는 재현율과 정밀도의 조화 평균이기 때문에 오탐과 미탐을 절충한 정도를 측정한 척도가 될 수 있다. 따라서, 규칙 집합을 적용하기 위한 일치율 임계값은 가장 높은 F1 score를 보이는 85%로 설정하는 것이 타당하다.

4.4 패밀리 식별 성능 측정

규칙 집합에 대한 일치율 임계값을 정의했다면, 이를 이용해 패밀리 탐지와 분류 작업이 가능하다. 패밀리 탐지란 임의의 악성코드를 대상으로 특정 패밀리를 탐지하는 작업이고, 패밀리 분류란 주어진 샘플에 패밀리 라벨을 할당하는 작업이다.

패밀리 탐지의 경우 전체 악성코드 샘플 4,443개를 모두 이용하고, 패밀리 분류의 경우 4,443개 샘플 중에서 분류하고자 하는 10개 패밀리에 해당하는 샘플 1,582개만 이용해 성능을 측정한다. 두 실험 모두 데이터 셋을 6:4의 비율로 학습셋과 시험셋으로 나누고 학습셋에서 추출된 규칙 집합을 시험셋에 적용했으며, 총 5번의 반복 실험을 진행하여 평균 성능을 측정했다. 이때, 패밀리 탐지와 분류 모두 식별해야할 샘플은 1,582개 중 40%인 650개로 동일하다.

패밀리 탐지의 경우 10개 패밀리에 대한 규칙 집합을 독립적으로 적용하여 일치율이 85%를 넘으면 해당 패밀리로 탐지한다. 규칙 집합을 독립적으로 적용하기 때문에 하나의 샘플이 여러 패밀리에 탐지될 수 있으며, 최종 성능은 각 패밀리의 탐지 성능의 평균으로 구한다. 반면, 패밀리 분류의 경우 반드시 학습된 패밀리 중 하나로 분류해야 하므로 임계값을 0%로 적용하여 일치율이 가장 높은 규칙 집합의 패밀리로 분류한다.

Table 4.는 패밀리 탐지 및 분류에 대한 성능 측정 결과이다. 패밀리 탐지의 경우 전체 데이터 셋이 863개의 패밀리로 구성되었고, 패밀리 분류의 경우 전체 데이터 셋이 10개의 패밀리로 구성되었다. 패밀리 분류의 경우 분류하고자 하는 패밀리의 샘플만을 대상으로 하므로 임의의 샘플을 대상으로 하는 패밀리 탐지보다 높은 성능을 보일 것을 예상할 수 있다.

실험 결과 패밀리 탐지의 경우 85.1%의 평균 정밀도를 보였지만, 패밀리 분류의 경우 97.7%의 평균 정밀도를 보였다. 패밀리 탐지가 분류보다 정밀도가 낮은 이유는 패밀리 탐지의 경우 임의의 악성코드들도 포함되었기 때문에 오탐이 발생할 확률이 더 높기 때문이다. 특히, 'bladabindi'의 정밀도가 매우 낮은 것을 확인할 수 있었는데, 이는 API 호출 빈도가 해당 패밀리에 대해서는 오탐을 가려낼 수 있는 식별성이 떨어짐을 의미한다.

주목할 점은 패밀리 탐지의 경우 91.3%의 평균 재현율을 보였고, 패밀리 분류의 경우 98.1%의 평균 재현율을 보였다는 것이다. 재현율의 경우 식별해야 할 샘플 중에서 성공적으로 식별한 샘플의 수인데, 패밀리 탐지와 분류 모두 식별해야 할 샘플은 10개의 패밀리에 해당하는 650개로 동일하다. 즉,

Table 4. Performance of detection and classification

Family	Detection		Classification	
	Prec	Rec	Prec	Rec
Berbew	1.0	1.0	1.0	1.0
Gator	1.0	1.0	1.0	1.0
Wabot	0.934	0.984	1.0	0.991
Mira	0.997	1.0	0.973	1.0
Multiplug	1.0	0.8	0.98	0.926
Loadmoney	0.938	0.818	1.0	0.96
Browsefox	0.993	0.894	0.909	1.0
Bladabindi	0.289	0.719	0.939	1.0
Gate	0.725	1.0	1.0	1.0
Installmonster	0.638	0.92	0.968	0.938
Mean	0.851	0.913	0.977	0.981

식별해야 할 샘플의 수는 같지만, 식별한 샘플의 수가 달라 재현율이 달라진 것이다.

식별한 샘플의 수가 다른 이유는 규칙 집합이 학습셋에 최적화되어 정의되기 때문이다. 패밀리 분류의 경우 분류하고자 하는 샘플만을 대상으로 하므로 오탐이 발생할 확률이 낮아 API 호출 빈도 임계값이 낮아진다. 반면, 패밀리 탐지의 경우 임의의 악성코드를 대상으로 하므로 오탐이 발생할 확률이 높아 API 호출 빈도 임계값이 높아진다. 즉, 학습셋에 따라 정의되는 규칙이 달라지기 때문에 재현율이 달라진 것이다.

4.5 기존 연구와 성능 비교

패밀리 탐지 및 분류 성능을 정밀도와 재현율이라는 척도를 이용해 분석했지만, 본 연구의 우수성을 입증하기 위해서는 기존 연구와 동일한 척도를 이용해 성능을 비교할 필요가 있다. 먼저, 패밀리 분류에 관해서 기존 연구와 비교한 결과는 Table 5.와 같다. 패밀리 분류의 경우 분류하고자 하는 패밀리의 샘플로만 데이터 셋을 구성하기 때문에 데이터 셋에 패밀리의 수와 전체 샘플의 수를 기입했다. 기존 연구들은 F1 score를 이용해 성능을 측정했기에 본 연구에서도 평균 정밀도와 재현율에 대해서 F1 score를 이용해 성능을 측정했다.

다음으로 패밀리 탐지에 관해서 비교한 결과는 Table 6.과 같다. 패밀리 탐지에 관해서는 비교하기 적절한 연구가 없어 악성코드 탐지에 관한 연구와 비교했다. 악성코드 탐지는 임의의 프로그램을 대상으로 악성코드를 탐지하는 작업인데, 일반적으로는 정상코드와 악성코드로 데이터 셋을 구성하기 때문에 데이터 셋에 정상코드와 악성코드의 수를 포함했다. Table 6.의 데이터 셋에서 N은 정상코드(Normal)을 의미하고, M은 악성코드(Malware)를 의미한다.

Table 5. Performance comparison for classification

Year	Author	Dataset	F1
2016	Hansen et al.[5]	5 Families (31,295)	86%
2018	Alsulami et al.[13]	38~55 Families (100,000)	77%
2019	Kumar et al.[14]	15 Families (6,370)	93%
Our Research		10 Families (1,582)	98%

Table 6. Performance comparison for detection

Year	Author	Dataset	Acc
2015	Saxe et al.[15]	N(81,900), M(350,016)	95%
2017	Yuxin et al.[16]	N(2,050), M(2,050)	96%
2019	Garg et al.[4]	N+M (35,000)	93%
Our Research		M(4,443)	91.8%

악성코드 탐지에 관한 기존 연구들은 식 (8)과 같이 정확도(Accuracy)를 이용해 성능을 측정했다. 양성(Positives)과 음성(Negatives)이 있을 때, 올바르게 예측한 양성은 TP, 음성은 TN이고, 틀리게 예측한 양성은 FP, 음성은 FN이다.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

기존 연구에서는 악성코드가 양성이고, 정상코드를 음성으로 간주하여 성능을 측정했지만, 본 연구에서는 특정 패밀리를 탐지하는 것이 목표이므로 10개의 패밀리에 해당하면 양성, 나머지는 모두 음성으로 간주하여 성능을 측정했다. 즉, 시험셋 1,778개 샘플 중에서 10개의 패밀리에 해당하는 650개의 샘플은 양성, 나머지 1,128개의 샘플은 음성으로 간주하고 측정했다. 이때, 하나의 샘플이 여러 패밀리에 탐지되면 가장 높은 일치율을 보이는 패밀리로 탐지한다.

본 연구를 기존 연구와 비교한 결과 패밀리 분류의 경우 기존 연구보다 성능이 높았지만, 패밀리 탐지의 경우 기존 연구보다 성능이 떨어지는 것을 확인할 수 있었다. 그렇지만, 본 연구의 경우 기존 연구와 달리 패밀리 정보까지 알 수 있어 활용도가 더 뛰어나다.

4.6 규칙 집합 일치율 임계값에 관한 고찰

본 연구에서는 규칙 집합 일치율에 대한 적절한 임계값을 찾기 위해 임의의 패밀리를 대상으로 임계값 별 성능을 측정하였고 가장 높은 성능을 보이는

85%를 임계값으로 설정했다. 그렇지만, 이는 임의의 패밀리를 대상으로 평균적인 성능을 구한 것이기 때문에 특정 패밀리에 대해서는 최적의 임계값이 아닐 수도 있다.

따라서, 85%라는 임계값이 특정 패밀리의 최적의 임계값에 얼마나 근접했는지 확인할 필요가 있다. 우리는 이를 위해 Table 3.의 10개 패밀리에 대한 임계값 별 성능을 측정하여 Fig. 3.과 같이 시각화했다.

Fig. 3.을 보면, 가장 높은 성능을 보이는 최적의 임계값은 79%로 본 연구에서 정의한 임계값과 다르다. 그렇지만, 임계값을 79%로 했을 때의 성능은 0.899이고, 임계값을 85%로 했을 때의 성능은 0.878로 최적의 성능에 거의 근접했다. 즉, 실험을 통해 정의한 임계값이 최적의 성능에 근접한 성능을 제공한다고 볼 수 있다.

본 논문의 경우 기존 연구와 성능을 비교하는 것이 목적이므로 가장 높은 성능을 보이는 85%를 임계값으로 설정하는 것이 타당하다. 그렇지만, 실제로 악성코드를 탐지할 때는 오탐보다 미탐이 발생하는 것이 더 치명적이기 때문에 재현율을 최대화할 수 있는 임계값을 결정해야 한다.

이를 위해 Fig. 4.와 같이 F1 score에 추가적으로 재현율과 정밀도를 함께 시각화했다. Fig. 4.를 보면 재현율의 경우 임계값이 낮을수록 상승하며, 임계값이 0%일 때 최대가 되는 것을 확인할 수 있다. 그렇지만, 임계값이 0%에 가까울수록 정밀도는 극단적으로 감소하는데, 이 경우 너무 많은 오탐이 발생하여 실질적으로 활용할 수 없다.

악성코드를 탐지하기 위해서는 정밀도와 재현율을

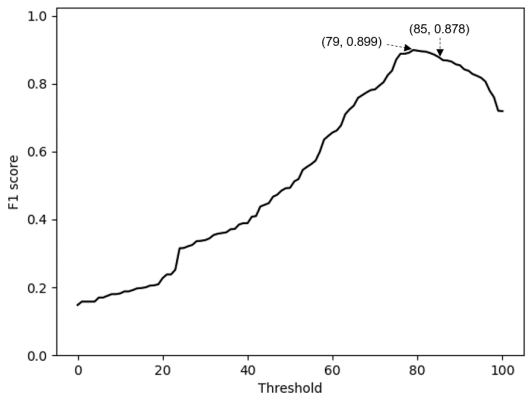

Fig. 3. F1 score per threshold for 10 families

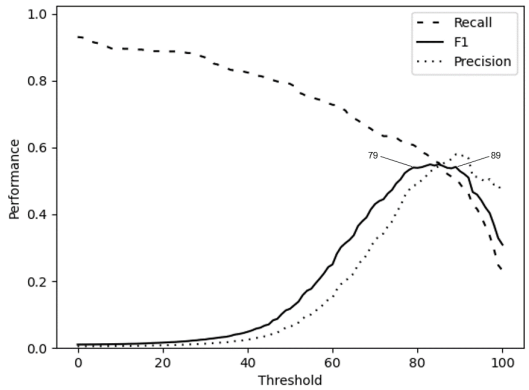

Fig. 4. Performance per threshold for arbitrary families

절충하면서 재현율은 최대화할 수 있는 임계값을 찾아야 한다. F1 score는 정밀도와 재현율의 조화 평균이기 때문에 오탐과 미탐을 절충한 정도를 판단할 수 있는 척도가 될 수 있다. 즉, 악성코드를 탐지하기 위해서는 F1 score를 높게 유지하면서 재현율을 최대화할 수 있는 임계값을 찾아야 한다.

우리는 이를 위해 F1 score가 높게 유지되는 구간을 탐색했다. 그리고, 임계값이 79%보다 작아질 때는 단조 감소하고 89%보다 커질 때는 단조 증가하지만, 임계값이 79~89%인 구간에서는 증가와 감소를 반복하며 1% 미만의 성능 차이를 유지하는 것을 확인할 수 있었다. 게다가, 이 구간에서는 최대의 성능을 보이는 임계값인 85%도 포함되므로 F1 score가 높게 유지된다고 볼 수 있다.

즉, 임계값이 79~89%인 구간에서는 오탐과 미탐이 적절히 절충되었다고 볼 수 있으며, 해당 구간에서는 어떤 임계값을 선택하더라도 성능의 차이가 거의 없다. 그렇지만, 정밀도와 재현율은 큰 차이를 보이는데, 임계값이 79%일 때는 0.48의 정밀도와 0.60의 재현율을 보이지만, 임계값이 89%일 때는 0.57의 정밀도와 0.50의 재현율을 보인다. 즉, 임계값이 89%일 때는 79%일 때보다 재현율이 10% 감소하였고, 이는 10%의 미탐이 발생했음을 의미한다.

악성코드를 탐지할 때는 미탐을 최소화해야 하므로, F1 score를 높게 유지할 수 있는 79~89% 구간에서 재현율을 최대화할 수 있는 79%를 임계값으로 선정하는 것이 타당하다.

V. 결 론

본 논문에서는 API 호출 빈도를 이용한 악성코드 패밀리 탐지 및 분류 방법을 제안했다. 제안 방법은 특정 API에 대한 호출 빈도가 임계값을 넘는지 검사하는 규칙을 정의하고, 해당하는 규칙의 비율에 기반하여 특정 패밀리를 식별하기 위한 것이다. 이때, 임계값을 결정하는 과정에서 결정트리 알고리즘을 응용하여 학습셋으로부터 특정 패밀리를 가장 잘 식별할 수 있는 값을 결정하도록 고안했다.

본 논문에서 제안된 방법은 기존 연구보다 악성코드 패밀리 분류에 대해서는 높은 성능을 보였지만, 탐지에 대해서는 기존 연구에 미치지 못하는 한계가 있었다. 그렇지만, 악성코드 여부만 판단하는 기존 연구와 달리 본 연구는 패밀리 정보를 식별할 수 있다는 점에서 보안 솔루션으로써의 활용도는 더 뛰어나다.

악성코드 패밀리 탐지 과정에서 특정 패밀리에 대해서만 정밀도가 낮은 경우가 있었는데, 이를 보완하기 위해서 API 호출 빈도 외에 추가적인 정보를 결합하기 위한 방안을 연구할 필요가 있다.

References

[1] ranCert, “Ransomware”, https://www.rancert.com/bbs/bbs.php?mode=view&id=679&bbs_id=news&page=8&part=&keyword=, Apr. 2021.

[2] wikipedia, “Ransomeware”, https://en.wikipedia.org/wiki/Ransomware, Apr. 2021.

[3] Uppal, Dolly, et al. "Malware detection and classification based on extraction of API sequences." 2014 International conference on advances in computing, communications and informatics (ICACCI). IEEE, pp. 2337-2342, Sept. 2014.

[4] Garg, Vidhi, and Rajesh Kumar Yadav. "Malware Detection based on API Calls Frequency." 2019 4th International Conference on Information Systems and Computer Networks (ISCON). IEEE, pp. 400-404, Nov. 2019.

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

[6] VirusTotal, “Virustotal” https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works, Apr. 2021.

[7] Sebastián, Marcos, et al. "Avclass: A tool for massive malware labeling." International symposium on research in attacks, intrusions, and defenses. Springer, Cham, pp. 230-253, Sept. 2016.

[8] wikipedia, “Hierarchical clustering”, https://en.wikipedia.org/wiki/Hierarchical_clustering, Apr. 2021.

[9] Cuckoo Sandbox, “Cuckoo Sandbox”, https://cuckoosandbox.org/, Apr. 2021.

[10] CrowdStrike, “Malware Type”, https://www.crowdstrike.com/cybersecurity-101/malware/types-of-malware/, Apr. 2021.

[11] wikipedia, “Silhouette”, https://en.wikipedia.org/wiki/Silhouette_(clustering), Apr. 2021.

[12] Pang-Ning Tan, Introduction to Data Mining, Pearson Education, pp 588, 2019.

[13] Alsulami, Bander, and Spiros Mancoridis. "Behavioral malware classification using convolutional recurrent neural networks." 2018 13th International Conference on Malicious and Unwanted Software (MALWARE). IEEE, pp. 103-111, March. 2018.

[14] Kumar, Nitish, and Toshanlal Meenpal. "Texture-Based Malware Family Classification." 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT).

IEEE, pp. 1-6, July. 2019.
[15] Saxe, Joshua, and Konstantin Berlin. "Deep neural network based malware detection using two dimensional binary program features." 2015 10th International Conference on Malicious and Unwanted Software (MALWARE). IEEE, pp. 11-20, Oct. 2015.
[16] Yuxin, Ding, and Zhu Siyi. "Malware detection based on deep learning algorithm." Neural Computing and Applications 31.2 (2019) pp. 461-472, July. 2017.

〈저자소개〉

조 우 진 (Woo-Jin Joe) 학생회원
2018년: 충남대학교 컴퓨터공학과 학사 졸업.
2018년~현재: 충남대학교 컴퓨터공학과 석박통합 과정
〈관심분야〉 악성코드 분석, 기계학습

김 형 식 (Hyong-Shik Kim) 종신회원
1988년: 서울대학교 컴퓨터공학과 학사 졸업.
1990년: 서울대학교 컴퓨터공학과 석사 졸업.
1997년: 서울대학교 컴퓨터공학과 박사 졸업.
1999년~현재: 충남대학교 인공지능학과 교수
〈관심분야〉 컴퓨터시스템보안