Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.32, NO.5, Oct. 2022 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2022.32.5.945

유사성 해시 기반 악성코드 유형 분류 기법*

김 윤 정,[1†] 김 문 선,[2] 이 만 희[3‡]
[1]이글루코퍼레이션 (사원), [2]소프트버스 (책임연구원), [3]한남대학교 (교수)

Method of Similarity Hash-Based Malware Family Classification*

Yun-jeong Kim,[1†] Moon-sun Kim,[2] Man-hee Lee[3‡]
[1]Igloo Corporation (Employee), [2]Softverse (Principal researcher),
[3]Hannam University (Professor)

요 약

매년 수십억 건의 악성코드가 탐지되고 있지만, 이 중 신종 악성코드는 0.01%에 불과하다. 이러한 상황에 효과적인 악성코드 유형 분류 도구가 필요하지만, 선행 연구들은 복잡하고 방대한 양의 데이터 전처리 과정이 필요하여 많은 양의 악성코드를 신속하게 분석하기에는 한계가 있다. 이 문제를 해결하기 위해 본 논문은 유사성 해시를 기반으로 복잡한 데이터 전처리 과정 없이 악성코드의 유형을 분류하는 기법을 제안한다. 이 기법은 악성코드의 유사성 해시 정보를 바탕으로 XGBoost 모델을 학습하며, 평가를 위해 악성코드 분류 분야에 널리 활용되는 BIG-15 데이터셋을 사용했다. 평가 결과, 98.9%의 정확도로 악성코드를 분류했고, 3,432개의 일반 파일을 100% 정확도로 구분했다. 이 결과는 복잡한 전처리 과정 및 딥러닝 모델을 사용하는 대부분의 최신 연구들보다 우수하다. 따라서 제안한 접근법을 사용하면 보다 효율적인 악성코드 분류가 가능할 것으로 예상된다.

ABSTRACT

Billions of malicious codes are detected every year, of which only 0.01% are new types of malware. In this situation, an effective malware type classification tool is needed, but previous studies have limitations in quickly analyzing a large amount of malicious code because it requires a complex and massive amount of data pre-processing. To solve this problem, this paper proposes a method to classify the types of malicious code based on the similarity hash without complex data preprocessing. This approach trains the XGBoost model based on the similarity hash information of the malware. To evaluate this approach, we used the BIG-15 dataset, which is widely used in the field of malware classification. As a result, the malicious code was classified with an accuracy of 98.9% also, identified 3,432 benign files with 100% accuracy. This result is superior to most recent studies using complex preprocessing and deep learning models. Therefore, it is expected that more efficient malware classification is possible using the proposed approach.

Keywords: Malware, Malware classification, Machine learning, Similarity hash, TLSH

I. 서 론

세계적인 보안 업체 SonicWall에 따르면 2021년 한 해 동안 탐지된 25억 건의 악성코드 중 신종 악성코드는 단 0.01%인 30만 건에 불과한 것으로 나타났다[1]. 이 수치는 현재 발견되고 있는 대부분의 악성코드가 새로운 취약점 및 공격 기법을 사용하는 것이 아닌 이미 알려진 공격 유형을 그대로 활용

Received(08. 16. 2022), Modified(09. 14. 2022), Accepted(09. 14. 2022)
* 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(2021R1A4A2001810).
† 주저자, kyunjeong125@gmail.com
‡ 교신저자, manheelee@hnu.kr(Corresponding author)

하고 있다는 것을 의미한다. 따라서 악성코드의 유형을 자세한 분석 전에 식별할 수 있으면, 해당 유형 악성코드의 알려진 정보를 활용하여 보다 빠르고 정확한 분석이 가능하다. 다만, 수십억 건의 악성코드를 분석가들이 직접 분류하는 것은 매우 어려운 일이다. 따라서 이 문제를 해결하기 위한 연구로 다양한 악성코드 유형 분류 기법들이 제안되었다. 이 연구들은 동적 및 정적 분석을 통해 악성코드의 특징 정보를 수집하고, 이를 바탕으로 악성코드의 행위 및 목적을 나타내는 유형을 분류한다.

여기서 동적 분석 기반 기법들은 악성코드를 실제 실행시킨 뒤, 그 행위를 관찰한 정보를 바탕으로 악성코드의 유형을 분류한다. 이 기법은 특정 상황에서 보다 정확한 악성코드 분류 지표를 제공하지만, 악성코드의 실행 조건에 부합하는 환경을 조성해야 하는 단점이 있다. 예를 들어, 운영체제의 특정 취약점을 악용하는 악성코드는 해당 취약점을 동적 분석 환경이 동일하게 가지고 있어야만 악성 행위를 발현한다. 이러한 환경을 모두 맞추는 것은 매우 어려운 일이다. 따라서 동적 분석 기법은 신종 악성코드를 정밀 분석하기에 매우 적합하지만, 방대한 양의 변종·파생 악성코드를 빠르게 분류하는 경우에는 적합하지 않을 수 있다. 이러한 이유로 정적 분석 기반 악성코드 유형 분류 연구가 활발히 진행되고 있다.

정적 분석 기반 기법들은 일반적으로 악성코드 유형 분류를 위해 악성코드의 바이너리에서 특징 정보들을 추출한 뒤, 유형 분류 알고리즘을 적용한다. 이때 사용되는 특징 정보는 싱글링(shingling) 또는 이미지 변환 방식 등을 사용한다. 다만, 이러한 접근법은 데이터 전처리에 많은 시간이 소요되는 단점이 있다. 예를 들어, 널리 사용되는 N-gram은 싱글링 단위인 N의 크기에 따라 연산량이 기하급수로 증가하며, 악성코드의 특성에 따라 최적의 N 값이 달라질 수 있다. 이미지 변환 또한 많은 연산량이 필요한 작업이다.

이러한 문제를 극복하기 위해 유사성 해시(similarity hash) 기반 유형 분류 기법들이 제안되고 있다. 유사성 해시는 입력받은 데이터가 비슷할수록 유사한 해시값을 생성하는 특징이 있다. 따라서 악성코드의 정적 특성을 하나의 해시값으로 표현할 수 있다. 다만 선행 연구들은 이 해시값에 싱글링 등 기존의 특징 정보 생성 기술을 단순 적용하여 유사성 해시의 이점을 온전히 활용하지 못하고 있다. 이러한 문제를 해결하기 위해 본 논문에서는 유사성 해시값의 각 digest를 하나의 특징 정보로 사용하여 악성코드 유형을 분류하는 새로운 접근법을 제안한다. 이 접근법은 복잡한 데이터 전처리 과정이 필요한 기존의 방식들과 달리 유사성 해시의 빠른 속도와 정확한 유사도 표현 능력을 온전히 활용하는 장점이 있다.

이 접근법을 평가하기 위해 BIG-15(Microsoft malware classification challenge) 데이터셋을 활용한 분류 실험을 진행했다[2]. 실험 결과, 98.9%의 정확도로 악성코드의 유형 정보를 분류할 수 있었다. 이는 복잡한 데이터 전처리 및 여러 딥러닝 모델을 동시에 사용하는 연구들이 95~99%의 정확도임을 비교하면 그에 준하거나 더 뛰어난 결과이다. 무엇보다도, 딥러닝 기반 접근법은 일반적으로 모델 학습 시간에 수십 분~시간이 소요되지만 본 논문의 접근법은 단 7.7초의 학습 시간만 필요했다. 또한 리눅스 시스템 및 KISA[3] 데이터셋에서 수집한 3,432개의 일반 파일을 함께 학습하여 모델을 구성한 결과, 정상 파일에 대한 구분 정확도는 100%였다. 따라서 본 논문이 제안하는 기법을 활용하면 고도의 리버스 엔지니어링 기술이 없는 사용자 및 악성코드 분석가의 분석 시간을 크게 단축시켜 보안업계의 악성코드 대응 능력 향상에 기여할 수 있을 것으로 기대한다.

II. 배경 지식

일반적인 해시 알고리즘과 달리 유사성 해시 알고리즘은 데이터가 비슷할수록 유사한 결과를 출력하도록 설계되어 있다. 이러한 유사성 해시의 기반이 되는 체계는 대표적으로 CTPH와 LSH가 있다. CTPH(Context Triggered Piecewise Hashing)는 입력값을 여러 조각으로 나누어 계산한 값을 결합하여 해시값을 구성하는 piecewise hashing 기법을 기반으로 입력이 유사할수록 유사한 값을 출력하게 하는 기법이다. 기존 piecewise hashing은 입력값을 정해진 크기로 나누어 계산하므로 데이터에 작은 변화가 생겨도 분할되는 내용이 크게 달라진다. 반면 CTPH는 입력값의 문맥을 기준으로 블록화하여 계산한다. 따라서 데이터가 일부 변경되어도 내용을 구분하여 블록의 경계를 결정하므로 해시값 또한 일부만 변경된다[4].

LSH는 입력 데이터가 유사할수록 유사한 해시값을 생성하는 퍼지(fuzzy) 해시이다. LSH는 우선 입력값에 대한 싱글링을 통해 입력된 데이터를 여러

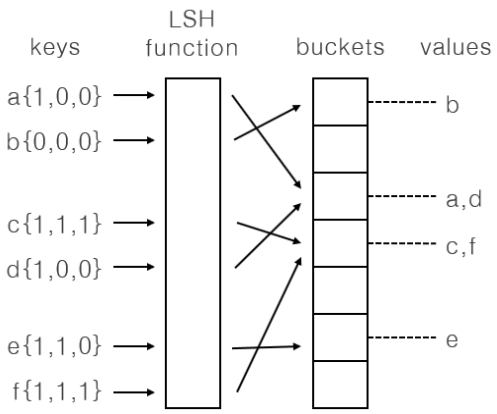


Fig. 1. Example of LSH algorithm

값으로 분리한 뒤, 각 값의 jaccard 거리를 계산한다. 이후 jaccard 거리가 가까운 값을 서로 같은 버킷에 할당한다. 여기서 버킷은 1바이트 해시값을 결정하기 위한 공간이다. 이처럼 LSH는 입력값의 유사도를 계산하는 과정을 통해 각 값들의 지역성(locality)을 보존하여 비슷한 해시값을 생성한다. Fig. 1.은 LSH 여러 입력값이 유사도에 따라 같은 버킷에 할당되는 것을 보여준다.

LSH는 주로 최근접 이웃 탐색 및 데이터 클러스터링에 활용되어 왔으나, 최근에는 악성코드 식별 및 유형 분류에도 적극적으로 활용되고 있다[5,6]. 이 연구들은 LSH 알고리즘뿐만 아니라 LSH의 구조를 기반으로 설계된 파생 알고리즘을 사용한다. 이러한 알고리즘들은 싱글링 크기 및 유사도 비교 메트릭에 대한 변형 등 독자적인 아이디어가 접목되어 보다 정확한 유사도 비교가 가능하도록 한다. 이러한 유사성 해시들은 대표적으로 Nilsimsa와 TLSH 알고리즘이 있다[7,8].

2.1 Ssdeep

Ssdeep은 Kornblum가 제안한 퍼지 해시로 CTPH 방식을 사용하는 유사성 해시의 일종이다. Ssdeep 해시값은 블록 크기와 두 개의 시그니처 값으로 구성된다. 블록 크기는 rolling hash 알고리즘을 이용하여 입력값에 대한 문맥 정보를 바탕으로 블록 크기를 결정한다. Rolling hash 알고리즘은 입력으로부터 이동하는 윈도우에서 바로 해시 함수를 적용하여 빠른 계산이 가능하다. 앞서 구한 블록 크기를 기반으로 spamsum 알고리즘의 traditional hash로 생성한 해시값으로 시그니처를 구성한다. 첫 번째 시그니처는 블록 크기를 이용하여 생성하고 두 번째 시그니처는 블록 크기에 2배한 값을 이용하여 생성한다.

Ssdeep의 출력 길이는 다른 해시와 달리 고정되어 있지 않다. 첫 번째 시그니처 값은 최대 64바이트, 두 번째 시그니처 값은 최대 32바이트 길이이다. Ssdeep 알고리즘은 두 해시값끼리의 유사도를 측정할 수 있으며 0에서 100까지의 정수로 출력되며 100에 가까울수록 유사하다[4].

2.2 Nilsimsa

Nilsimsa는 LSH를 기반으로 스팸 메일 탐지를 위해 고안된 해시 알고리즘이다. 높은 정확도로 유사도 비교가 가능하여 악성코드 분류 분야에서도 활용되고 있다. Nilsimsa 알고리즘은 5바이트 크기의 슬라이딩 윈도우를 사용하여 입력값에 대한 트라이그램(trigram)을 생성한다. 이후 각 트라이그램의 요소에 0~255 범위의 정수를 반환하는 해시 알고리즘을 적용하여 버킷을 채운다. 이때, 버킷에 매핑된 트라이그램 요소의 수가 전체 버킷 크기의 중위수보다 작으면 0, 크면 1을 값으로 하는 해시값을 생성한다.

Nilsimsa 알고리즘의 해시값은 32바이트로 고정되며, 0부터 128 사이의 정수로 표현된다. 입력값에 대한 유사도는 정수로 결정되며 128에 가까울수록 유사하다. 이러한 특징을 통해 생성된 64바이트 해시값의 각 인덱스 크기를 비교하면 두 해시값의 유사도를 계산할 수 있다[7].

2.3 TLSH

TLSH(Trend micro Locality Sensitive Hashing)는 2013년 Trend micro 사가 LSH를 기반으로 만든 해시 알고리즘이다. TLSH는 우선 입력값에 5바이트 크기의 슬라이딩 윈도우 알고리즘을 적용하여 문자열 시퀀스를 생성한다. 이후 생성된 시퀀스의 크기와 동일한 버킷 배열을 생성한다. 각 버킷은 5-gram 문자열 시퀀스에서 추출할 수 있는 triplet의 경우의 수 중에서 상위 6개에 pearson 해시를 적용한 결과가 매핑된다. 상위 6개를 사용하는 이유는 다음 7~10 triplet은 다음 시퀀스에서 중복되어 나타나기 때문이다. 버킷에 값이 할당되면

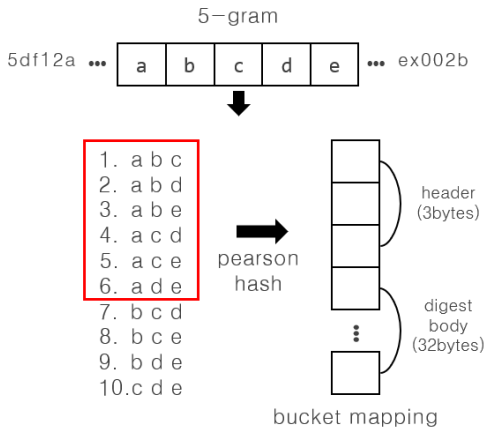


Fig. 2. Example of generating a TLSH hash digest

각 버킷은 사분위수를 사용하여 각각 q1, q2, q3 그룹으로 구분된다. 여기서 q1은 버킷 배열 인덱스의 상위 75%이며, q3는 하위 25%이다.

버킷이 생성되면 우선 해시 전체 35바이트의 값 중 헤더 정보로 상위 3바이트를 할당한다. 헤더에는 입력값의 체크섬, 입력값의 길이, 각 사분위수의 크기 정보가 순서대로 담긴다. 이후 나머지 32바이트는 생성된 버킷을 사분위수 그룹에 서로 다른 값이 생성된다. 이때 생성되는 값은 q1, q2, q3에 각각 00, 01, 10이며 그 외 경우는 11이다. 여기서 해시값 생성에 사분위수를 사용하는 이유는 단순한 문서가 아닌 바이너리 및 이미지 데이터에 대한 비교 정확도를 높이기 위함이다. Fig. 2.는 TLSH 해시값이 생성되는 과정을 간단히 표현한 것이다.

TLSH 알고리즘은 최소 50바이트 길이의 입력이 요구되며 충분한 랜덤성이 있어야 한다. 또한 출력되는 해시값 길이는 고정된 70글자로 3바이트 길이의 헤더와 나머지는 바디로 이루어져 있다. TLSH 유사도는 각 해시값의 헤더 간 거리와 바디 간 해밍 거리를 계산하여 도출한다. 해시값 간 거리가 가까울수록 유사도 점수가 높다[8].

III. 관련 연구

3.1 동적 분석 기반 유형 분류 기법

Jazi Hossein Hadian 등은 동적 분석으로 생성한 악성코드의 CFG(Control Flow Graph)를 바탕으로 악성코드 유형을 분류하는 기법을 제안하였다. 또한 그래프 비교에 필요한 연산량을 줄이기 위해 최적화 알고리즘인 Simulated annealing을 도입했다. 이 기법은 평균 94%의 정확도로 악성코드 유형을 분류할 수 있었다[9].

Pektaş Abdurrahman 등은 Cuckoo 샌드박스로 수집한 악성코드의 API(Application Programming Interface) 호출 순서 정보에 N-gram 알고리즘을 적용하여 특징 정보를 생성했다. 이후 해당 정보를 바탕으로 SVM(Support Vector Machine) 모델을 생성했다. 이 모델은 3,899개의 악성코드 샘플 유형을 97.6%의 정확도로 분류했다[10].

조우진 등은 악성코드의 API 호출 빈도에 따라 악성코드 유형을 분류하는 기법을 제안했다. 이 기법은 악성코드 유형 별 API 호출 임계치를 설정한 뒤, 분석 대상 악성코드의 API 호출 정보가 특정 임계치 값을 초과하면 유형을 특정한다. 이 기법을 바탕으로 1,582개 악성코드 데이터셋에서 유형 분류 실험을 수행한 결과, 10개 유형을 98.1% 분류 정확도 분류했다[11].

고동우 등은 악성코드가 호출하는 API의 호출 순서를 추출한 뒤, 이 정보를 유사성 해시의 일종인 TLSH를 이용하여 악성코드의 유형을 분류하였다. 이 기법은 기존 API 호출 순서 분석에서 발생하는 유사 함수의 모호성으로 인한 미탐지(false negative)를 줄이기 위해 API 추상화 기법을 도입하였다. 또한 효과적인 유사도 비교를 위해 TLSH 알고리즘으로 유사도 정보를 생성하였다. 이후 이 정보에 대해 K-medoids 알고리즘을 적용한 결과, 1,015개의 정상 및 악성 파일의 유형 정보를 72.11%의 정확도로 분류하였다[5].

이러한 동적 분석 기반 기법들은 악성 행위를 판단하는 관점에서 매우 높은 정확도를 보여줄 수 있다. 다만 분류 대상 악성코드의 악성 행위 발현을 위한 개별적인 조건을 만족시켜야 하며, 지능화된 악성코드의 다양한 동적 분석 회피 기술을 우회해야 하는 어려움을 극복할 필요가 있다. 즉 모든 분류 대상 악성코드를 직접 실행해야만 악성코드의 유형을 분류할 수 있는 한계점이 있다.

3.2 정적 분석 기반 유형 분류 기법

Mehadi Hassen 등은 악성코드 변종을 효과적으로 분류하기 위해 제어문(control statement)

싱글링 기반 악성코드 유형 분류 기법을 제안하였다. 싱글링 알고리즘으로는 N-gram을 사용했으며, 악성코드의 함수와 DLL(Dynamic Link Library) 기본 블록(basic block) 내의 opcode 시퀀스를 대상으로 특징 정보를 생성했다. 이후 해당 정보를 Random forest 모델에 적용하여 99.11%의 정확도로 악성코드 유형을 분류하였다[12].

Barath Narayanan Narayanan 등은 악성코드의 어셈블리 코드를 이용하여 유형을 분류하는 기법을 제안하였다. 딥러닝 알고리즘인 LSTM(Long Short Term Memory)과 CNN(Convolutional Neural Networks)을 통해 특징 정보를 추출하고 SVM과 로지스틱 회귀 모델을 통해 유형을 분류한다. 학습한 모델을 BIG-15 데이터셋에 적용한 결과, CNN, LSTM, SVM 모델을 함께 사용할 때 99.8%의 정확도로 악성코드 유형을 분류하였다[13].

Lin Li 등은 더블 바이트 인코딩 기반 데이터 전처리 기법을 바탕으로 악성코드 유형 분류를 수행했다. 이 기법은 ASM 파일과 바이트에서 16개의 opcode 특징 정보를 추출한다. 이후 더블 바이트 인코딩 기법을 사용하여 두 특징 정보를 이미지 데이터로 변환한 뒤, CNN 모델을 학습한다. 모델의 성능 평가 결과, BIG-15 악성코드 데이터셋에서 악성코드 유형을 98.66%의 정확도로 분류했다[14].

Sudhakar 등은 악성코드 유형 분류에 최적화된 새로운 딥러닝 아키텍처인 MCFT-CNN을 제안했다. 이 모델은 대규모 이미지 데이터베이스인 ImageNet을 학습한 모델에 악성코드 유형 분류 데이터셋을 학습시키는 전이학습을 통해 구축된다. 모델의 정확도는 악성코드 이미지 데이터셋 MalImg에서 99.18%, BIG-15에서 98.63%의 정확도를 보였다[15].

Mamoona Khan 등은 악성코드 유형 분류를 위해 바이트 코드에 대한 N-gram, 이미지, 엔트로피, 메타데이터, 문자열 길이 정보를 활용하여 CNN 모델을 학습시켰다. 메타데이터 정보에는 악성코드의 크기와 첫 번째 바이트의 주소가 포함된다. 모델의 성능 실험 결과, BIG-15 데이터셋의 악성코드 유형을 97.8% 정확도로 분류했다[16].

Jake Drew 등은 악성코드의 변종을 생물학적 돌연변이와 유사하다는 가정을 바탕으로 악성코드의 유전자 정보를 통한 유형 분류 기법을 제안했다. 이 기법은 우선 유사성 해시의 일종인 minhash를 사용하여 악성코드의 유전자 정보를 생성한다. 이후 유전자 정보를 분류하는 strand 분류기를 통해 악성코드 유형을 분류한다. BIG-15 데이터셋을 통해 이 기법의 정확도를 확인한 결과, 98% 정확도로 악성코드 유형을 분류했다[6].

박창욱 등은 악성코드의 명령어 및 변수 값이 저장된 .data 영역을 상호 비교하여 악성코드 유형을 분류할 수 있는 기법을 제안했다. 이때 데이터 비교에 소요되는 연산량을 줄이기 위해 유사성 해시의 일종인 Ssdeep을 사용했다. 이 방법의 유효성을 판단하기 위해 8개 유형의 악성코드 200종으로 유사도 비교 실험을 진행한 결과, .data 영역만으로도 악성코드 유형 분류가 가능한 것을 확인했다[17].

이처럼 정적 분석 기반 유형 분류 기법은 악성코드를 실행하지 않고 유형을 분류할 수 있는 장점이 있어 많은 연구에 활용되고 있지만 데이터 전처리에 많은 컴퓨팅 자원이 소모된다. 또한 정적 분석과 동적 분석의 장점을 결합한 하이브리드 기법도 제안되고 있지만 이는 보다 복잡한 특징 정보 추출이 필요한 한계가 있다[18]. 따라서 본 논문은 이러한 문제점들을 개선하기 위한 새로운 접근법으로 유사성 해시값을 특징 정보로 활용하는 악성코드 분류 모델을 제안한다.

IV. 유사성 해시 기반 악성코드 분류 기법

4.1 제안하는 접근법

본 논문은 복잡한 데이터 전처리 없이 악성코드 유형을 분류하기 위한 새로운 접근법으로 유사성 해시 기반 악성코드 유형 분류 기법을 고안하였으며, 이를 검증하는 실험을 수행했다. 검증 실험은 유사성 해시로 변종 악성코드 간의 공통적인 부분을 식별할 수 있는지 확인하는 방식으로 진행했다. 실험에 사용한 악성코드는 하드웨어 취약점 공격을 구현한 Spectre 악성코드의 두 가지 변종이다. 두 변종의 핵심 코드의 유사성은 최신 바이너리 비교 도구인 BinDiff[19]로 확인한 결과 약 86%였다.

실험을 위해 우선 두 변종 악성코드의 핵심 함수의 opcode를 추출한 뒤, 유사성 해시값을 생성했다. 이후 생성된 해시값의 각 인덱스를 비교하여 서로 일치하는 비율을 측정했다. 실험에 사용한 유사성 해시는 현재 악성코드 식별 및 유사도 비교 분야에서 가장 널리 사용되고 있는 Ssdeep, Nilsimsa,

Table 1. The similarity of two variants of spectre-PHT

	digest length	match rate
Ssdeep	54 bytes	0.19%
Nilsimsa	64 bytes	53.1%
TLSH	70 bytes	44.3%

TLSH이다. 실험 결과, 두 변종의 핵심 함수에 적용한 Nilsimsa와 TLSH 해시값이 40~50%로 유사하게 나타나는 것을 확인했다. 즉, Nilsimsa와 TLSH를 통해 생성한 해시값을 악성코드 유형 분류를 위한 특징 정보로 사용할 수 있음을 확인하였다. 반면 Ssdeep 알고리즘은 거의 모든 해시값이 다르게 생성되었다. 이는 해당 알고리즘이 생성된 해시값을 통해 직접적으로 유사성을 나타내지 않기 때문인 것으로 파악된다. Table 1.은 이 실험 결과를 정리한 것이다.

4.2 프로토타입 개발

제안하는 기법의 평가를 위해 프로토타입 도구를 개발했다. 이 도구는 먼저 악성코드 바이너리의 유사성 해시값을 생성한 뒤, 이 해시값을 머신러닝 모델이 학습할 수 있는 특징 벡터로 변환하는 전처리 과정을 거친다. 여기서 전처리 과정은 Fig. 3.과 같이 해시값을 각 1바이트씩 분할한다. 이때, 16진수로 생성되는 해시값을 머신러닝 모델이 인식할 수 있도록 유니코드를 통해 정수값으로 변환한다. 이 과정을 거쳐 생성된 특징 벡터는 머신러닝 모델을 학습하는 데 사용한다.

모델 학습에는 2015년 Microsoft가 kaggle 플랫폼을 통해 개최한 악성코드 유형 분류 챌린지에서

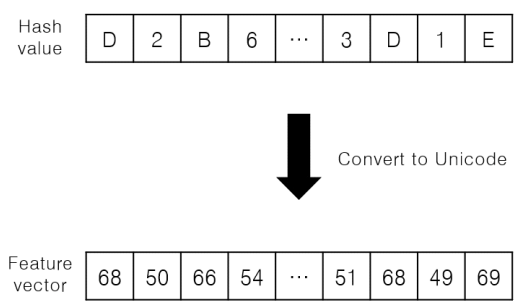


Fig. 3. Example of TLSH feature vector

공개된 BIG-15 데이터셋을 활용한다. 이 데이터셋은 약 500GB 크기로 9개 유형의 10,868개 악성코드 샘플에 대한 .asm 파일과 .bytes 파일을 제공한다. .asm은 IDA pro 디스어셈블러로 생성한 디스어셈블 파일이며, .bytes는 악성코드의 순수한 바이트 코드이다. 이 데이터셋은 방대한 양의 데이터와 정확한 라벨링을 제공하기 때문에 매우 다양한 악성코드 유형 분류 연구들이 인용하고 있다. 또한 현재까지도 관련 연구들이 활발히 진행되고 있으므로 제안하는 기법의 실험 결과에 충분한 근거를 제공할 것으로 판단된다.

4.3 최적 알고리즘 선택

개발한 프로토타입을 활용하여 최적의 유사성 해시 알고리즘과 분류기 모델을 선택하는 실험을 진행했다. 실험 대상 유사성 해시 생성 알고리즘은 Ssdeep, Nilsimsa, TLSH이며, 머신러닝 알고리즘은 DT(Decision Tree), SVM(Gaussian kernel), RF(Random Forest), XGBoost이다. 여기서 Ssdeep 알고리즘은 4.1의 실험에서 본 논문의 접근법이 유효하지 않음을 확인했지만, 악성코드 분석 분야에서 널리 활용되는 알고리즘이기 때문에 실험에서 제외하지 않았다. 머신러닝 모델은 악성코드 분석 분야에서 일반적으로 활용되는 알고리즘으로 선정했다. 실험 결과, TLSH와 XGBoost를 사용한 프로토타입 모델이 가장 높은 분류 정확도를 보임을 확인했다. Table 2.는 각 실험에 대한 분류 정확도를 정리한 것이다.

Ssdeep은 가장 높은 정확도를 보인 XGBoost 모델에서 57% 정확도를 보이는 것에 그쳤다. 이는 Ssdeep이 생성하는 해시값의 길이가 악성코드의 크기에 따라 변하는 특성이 있어 해시값 자체를 활용한 유사도 비교가 어렵기 때문으로 분석된다. 즉, Ssdeep과 같이 CTPH 기반 유사성 해시 알고리즘

Table 2. Accuracy of comparison for similarity hash algorithms

	Ssdeep	Nilsimsa	TLSH
DT	0.45	0.93	0.93
SVM	0.50	0.94	0.97
RF	0.56	0.93	0.98
XGBoost	0.57	0.98	0.98

은 본 연구에서 제안하는 기법에 적용할 수 없음을 확인했다. 이와 달리, Nilsimsa와 TLSH 알고리즘은 모두 XGBoost 모델에서 98%의 정확도로 악성코드 유형 분류 정확도가 높은 것으로 확인되었다. 다만, Nilsimsa 알고리즘은 특징 벡터를 추출하는 전처리 시간이 약 361,000초가 소요되었다. 반면 TLSH는 약 3,644초의 시간만 소요되었다. 따라서 동일한 정확도 대비 전처리 시간이 약 100배 더 빠른 TLSH 알고리즘과 XGBoost 모델을 사용하는 것이 가장 효과적임을 확인하였다. 여기서 동일한 정확도이지만 XGBoost 모델을 선택한 이유는 random forest 모델 대비 F1 점수가 0.1만큼 더 높았기 때문이다.

V. 실 험

5.1 악성코드 유형 분류 결과

제안하는 기법에서 최적의 알고리즘으로 확인된 TLSH와 XGBoost 모델을 사용하여 악성코드 유형 분류 정확도를 측정하는 실험을 진행했다. BIG-15 데이터셋을 통해 실험을 진행했으며, 해당 데이터셋이 제공하는 .asm과 .bytes 데이터 중 .bytes만 사용했다. 학습과 테스트 데이터는 8:2로 분할하였으며, 구축한 XGBoost 모델의 최적 하이퍼파라미터 항목 및 값은 *n_estimators*=250, *eval_metric*=mlogloss, *objective*=softmax, *learning_rate*=0.05이다.

실험 결과, 악성코드 유형 분류 정확도는 98.94%였으며 recall, precision, F1 점수는 각각 0.988, 0.967, 0.976으로 나타났다. 이때 각 지표의 average는 각 클래스에 가중치를 적용하지 않는 macro average이다. 더불어 confusion matrix 분석 결과, 대부분의 악성코드 유형을 98%~100% 정확도로 가깝게 분류한 것을 확인했다(Fig. 4.). 다만, 데이터셋의 5번 유형에 대한 정확도는 약 90% 수준으로 상대적으로 낮게 나타났다. 5번 유형은 Obfuscator.ACY로 난독화가 적용된 악성코드 유형이다. 따라서 정적 분석 기반 접근 방법으로 해당 유형을 완벽히 분류하는 것은 어려울 수 있다. 그럼에도 90%의 정확도를 보인 것은 본 연구의 접근법이 해당 유형의 악성코드를 구분하기에 충분히 유효한 것으로 판단된다.

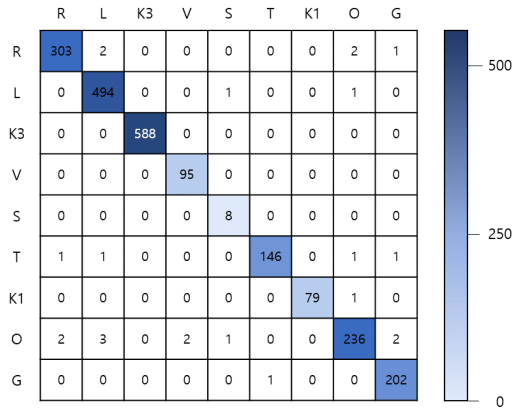


Fig. 4. Confusion matrix of TLSH-based malware family classification method

추가적으로 본 접근법의 실질성을 확인하기 위해 정상 파일과 악성 파일을 구분하는 실험을 진행했다. 정상 파일은 Linux 시스템에서 수집한 432개의 실행 파일과 더불어 KISA에서 제공하는 3000개의 정상 바이너리 파일을 활용했다. 실험 모델은 기존 9개의 악성 클래스 데이터셋에 정상 클래스를 추가하여 학습했다. 모델의 성능 확인 결과, 정상 파일 클래스에 대한 분류 정확도는 100%였으며, 오탐과 미탐이 발생하지 않았다.

5.2 기존 연구와 정확도 비교

제안하는 기법의 유효성을 판단하기 위해 관련 선행 연구들과 정확도를 비교했다. 비교 대상은 동일 데이터셋을 활용한 연구들 중 정적 분석 기법을 사용한 연구들을 중심으로 선정했다. 또한 해당 챌린지 데이터셋에 대한 우수한 결과를 위해 일반적으로 적용하기 어려운 기법을 사용하는 연구는 제외했다. Table 3.은 제안하는 기법과 선행 연구들의 특징을 정리한 것이다. Table 3.의 model은 해당 논문이 사용한 분류 기법이며, dataset type은 본 데이터셋이 제공하는 .asm 파일과 .bytes 파일 중 model 구성에 활용한 데이터셋을 보여준다. method는 악성코드의 데이터를 획득한 접근법을 의미하며 acr.은 유형 분류 정확도이다.

선행 연구와 비교 결과, 본 논문에서 제안하는 기법은 .bytes 데이터셋만을 통해 최신 연구를 비롯한 선행 연구들보다 우수한 결과를 보임을 확인했다. 특히 .asm 데이터셋은 악성코드에 대한 디스어셈블이 선행되어야 분류 모델에 활용할 수 있으므로 .bytes

Table 3. Compared to previous studies

	model	dataset type	method	acr.
[13]	CNN+LSTM+SVM	asm+bytes	static	99.8
our	XGBoost	bytes	static	98.94
[14]	CNN	asm+bytes	static	98.66
[15]	MCFT-CNN	asm+bytes	static	98.63
[6]	MinHash+CNN	bytes	static	98
[16]	N-gram	asm+bytes	static	97.8
[18]	Clustering	asm+bytes	hybrid	95.59

만을 활용하는 접근법이 보다 다양한 환경에서 활용될 수 있을 것으로 예상한다. 또한 세 개의 딥러닝 모델을 결합하여 앙상블 모델을 구성한 연구[7]와 비교하면 정확도가 낮지만, 제안하는 모델의 단순성을 고려할 때 일부 리소스가 제한되는 시스템에서 선택할 수 있는 좋은 대안이 될 것으로 판단된다.

VI. 결 론

대부분의 악성코드는 새로운 취약점 및 공격 기법을 사용하는 것보다는 기존에 알려진 유형을 그대로 활용하는 경우가 많다. 따라서 악성코드의 유형을 사전에 식별할 수 있다면, 해당 유형 악성코드에 대한 알려진 정보를 바탕으로 보다 효율적인 분석이 가능하다. 다만, 쏟아지는 악성코드를 분석가들이 직접 분류하는 것은 매우 어려운 일이므로 악성코드의 유형을 자동으로 분류할 수 있는 기술이 필요하다.

따라서 본 논문에서는 방대한 양의 악성코드의 유형을 보다 빠르고 간단하게 할 수 있는 기법을 제안하였다. 제안하는 기법의 평가 결과, 10.868개의 악성코드 유형을 98.9% 정확도로 분류할 수 있었다. 또한 3,432개의 정상 파일을 함께 학습하여 모델을 구성한 결과, 정상 파일에 대한 구분 정확도는 100%였다. 이러한 결과로 미루어 볼 때 본 모델을 활용하면 고도의 악성코드 분석 지식 없이 대량의 악성코드 유형을 빠르게 분류할 수 있어 관련 업계의 악성코드 대응 능력 향상에 기여할 수 있을 것으로 판단된다.

다만, 제안하는 기법은 복잡한 모델로 구성된 분류 도구에 비해 정확도가 약간 낮으며, 정적 분석의 특성상 패킹 및 난독화된 악성코드에 대해서는 정확도가 낮다. 따라서 향후 연구로는 바이너리 정보뿐만 아니라 제어 흐름 그래프 등 다양한 정적 분석 정보에 대한 유사성을 비교하는 방식으로 패킹 및 난독화된 악성코드의 유형까지 구분할 수 있도록 개선할 예정이다. 또한 제안한 기법의 정확도를 높이기 위해 다양한 머신러닝 모델을 함께 사용하는 앙상블 기법을 적용할 예정이다.

References

[1] Sonicwall, "SONICWALL: 'THE YEAR OF RANSOMWARE' CONTINUES WITH UNPRECEDENTED LATE-SUMMER SURGE," "https://www.sonicwall.com/news/sonicwall-the-year-of-ransomware-continues-with-unprecedented-late-summer-surge/," Oct. 2021.

[2] Kaggle, Microsoft Malware Classification Challenge (BIG 2015), "https://www.kaggle.com/c/malware-classification," 2015.

[3] Kisa, Information Security R&D Data Challenge 2019, "http://datachallenge.kr/challenge19/rd-datachallenge/malware/introduction/," Nov. 2019.

[4] Jesse Kornblum, "Identifying almost identical files using context triggered piecewise hashing," Digital investigation 3, pp. 91-97, Jul. 2006.

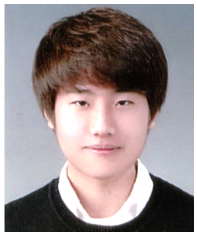
[5] Dongwoo Goh and Huykang Kim, "A Study on Malware Clustering Technique Using API Call Sequence and Locality Sensitive Hashing," Korea Institute of Information Security and Cryptology (KIISC), 27(1), pp. 91-101, Feb. 2017.

[6] Jake Drew, Tyler Moore and Michael Hahsler, "Polymorphic malware detection using sequence classification methods," 2016 IEEE Security and Privacy Workshops (SPW), pp. 81-87,

May. 2016.
[7] Diffeo diffeo, Py-nilsimsa, "https://github.com/diffeo/py-nilsimsa," Apr. 2016.
[8] Oliver, Jonathan, Chun Cheng and Yanggui Chen, "TLSH--a locality sensitive hash," 2013 Fourth Cybercrime and Trustworthy Computing Workshop. IEEE, pp. 7-13, Nov. 2013.
[9] Jazi Hossein Hadian and Ali Akbar Ghorbani, "Dynamic graph-based malware classifier," 2016 14th Annual Conference on Privacy, Security and Trust (PST), pp. 112-120, Dec. 2016.
[10] Pektaş Abdurrahman and Tankut Acarman, "Malware classification based on API calls and behaviour analysis," IET Information Security, vol. 12, no. 2, pp. 107-117, Sep. 2018.
[11] Woojin Joe and Hyongshik Kim, "Malware Family Detection and Classification Method Using API Call Frequency," Korea Institute of Information Security and Cryptology (KIISC), 31(4), pp. 605-616, Aug. 2021.
[12] Mehadi Hassen, Marco M. Carvalho and Philip K. Chan, "Malware classification using static analysis based features," 2017 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1-7, Feb. 2017.
[13] Barath Narayanan Narayanan and Venkata Salini Priyamvada Davuluru, "Ensemble malware classification system using deep neural networks," Electronics, vol. 9, no. 5, 721, Apr. 2020.
[14] Lin Li, Ying Ding, Bo Li, Mengqing Qiao and Biao Ye, "Malware classification based on double byte feature encoding," Alexandria Engineering Journal, vol. 61, no. 1, pp. 91-99, Jan. 2022.
[15] Sudhakar and Sushil Kumar, "MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in Internet of Things," Future Generation Computer Systems, vol. 125, pp 334-351, Dec. 2021.
[16] Mamoona Khan, Duaa Baig, Usman Shahid Khan and Ahmad Karim, "Malware Classification Framework using Convolutional Neural Network," 2020 International Conference on Cyber Warfare and Security (ICCWS), pp. 1-7, Oct. 2022.
[17] Changwook Park, Hyunji Chung, Kwangseok Seo and Sangjin Lee, "Research on the Classification Model of Similarity Malware using Fuzzy Hash," Journal of The Korea Institute of Information Security & Cryptology (KIISC), 22(6), pp. 1325-1336, Dec. 2012.
[18] Yunan Zhang, Chenghao Rong, Qingjia Huang, Yang Wu, Zeming Yang and Jianguo Jiang, "Based on Multi-features and Clustering Ensemble Method for Automatic Malware Categorization," 2017 IEEE Trustcom/BigDataSE/ICESS, 2017, pp. 73-82, Sep. 2017.
[19] Zynamics.com, "BinDiff," "https://www.zynamics.com," 2021.

〈저 자 소 개〉

김 윤 정 (Yun-jeong Kim) 정회원
2021년 2월: 한남대학교 수학과 및 컴퓨터통신무인기술학과 학사
2022년 2월: 한남대학교 컴퓨터공학과 석사
2022년 6월~현재: 이글루코퍼레이션 인프라사업본부 대전사이버분석팀
〈관심분야〉 악성코드 분석, 시스템 보안, 네트워크 보안



김 문 선 (Moon-sun Kim) 정회원
2020년 2월: 한남대학교 컴퓨터통신무인기술학과 학사
2022년 2월: 한남대학교 컴퓨터공학과 석사
2022년~현재: (주)소프트버스 책임연구원
〈관심분야〉 시스템 보안, 네트워크 보안, 바이너리 분석, 공급망 보안

이 만 희 (Man-hee Lee) 종신회원
1995년 2월: 경북대학교 컴퓨터공학과 학사
1997년 2월: 경북대학교 컴퓨터공학과 석사
2008년 8월: Texas A&M 대학교 컴퓨터공학과 박사
1997년~2003년: 한국과학기술정보연구원 연구원
2008년~2009년: Cisco Systems, San Jose
2010년~2012년: 국가보안기술연구소 선임연구원
2012년~현재: 한남대학교 교수
〈관심분야〉 네트워크/시스템/스마트폰/공급망 보안, 고성능 시스템, 컴퓨터교육