Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.34, NO.1, Feb. 2024 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2024.34.1.31

LLVM IR 대상 악성코드 탐지를 위한 이미지 기반 머신러닝 모델*

박 경 빈,[1†] 윤 요 섭,[1] 또 올 가,[2] 임 강 빈[3‡]
[1,2,3]순천향대학교 (학생, 대학원생, 교수)

Image-Based Machine Learning Model for Malware Detection on LLVM IR*

Kyung-bin Park,[1†] Yo-seob Yoon,[1] Baasantogtokh Duulga,[2] Kang-bin Yim[3‡]
[1,2,3]Soonchunhyang University (Undergraduate, Graduate, Professor)

요 약

최근 정적분석 기반의 시그니처 및 패턴 탐지 기술은 고도화되는 IT 기술에 따라 한계점이 드러나고 있다. 이는 여러 아키텍처에 대한 호환 문제와 시그니처 및 패턴 탐지의 본질적인 문제이다. 악성코드는 자신의 정체를 숨기기 위하여 난독화, 패킹 기법 등을 사용하고 있으며 또한, 코드 재정렬, 레지스터 변경, 분기문 추가 등 기존 정적분석 기반의 시그니처 및 패턴 탐지 기법을 회피하고 있다. 이에 본 논문에서는 이러한 문제를 해결할 수 있는 머신러닝을 통한 LLVM IR 코드 이미지 기반 악성코드 정적분석 자동화 기술을 제안한다. 바이너리가 난독화되거나 패킹된 사실에 불구하고 정적 분석 및 최적화를 위한 중간언어인 LLVM IR로 디컴파일한다. 이후 LLVM IR 코드를 이미지로 변환하여 CNN을 이용한 알고리즘 중 전이 학습 및 Keras에서 지원하는 ResNet50v2으로 학습하여 악성코드를 탐지하는 모델을 제시한다.

ABSTRACT

Recently, static analysis-based signature and pattern detection technologies have limitations due to the advanced IT technologies. Moreover, It is a compatibility problem of multiple architectures and an inherent problem of signature and pattern detection. Malicious codes use obfuscation and packing techniques to hide their identity, and they also avoid existing static analysis-based signature and pattern detection techniques such as code rearrangement, register modification, and branching statement addition. In this paper, We propose an LLVM IR image-based automated static analysis of malicious code technology using machine learning to solve the problems mentioned above. Whether binary is obfuscated or packed, it's decompiled into LLVM IR, which is an intermediate representation dedicated to static analysis and optimization. "Therefore, the LLVM IR code is converted into an image before being fed to the CNN-based transfer learning algorithm ResNet50v2 supported by Keras". As a result, we present a model for image-based detection of malicious code.

Keywords: LLVM IR, Image based, Malware Detection, ResNet50V2

Received(09. 11. 2023), Modified(1st: 11. 22. 2023, 2nd: 01. 12. 2024), Accepted(02. 02. 2024)

* 본 연구는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(2021R1A4A2001810)

* 본 연구는 2022년도 교육부의 재원으로 한국연구재단의 지원을 받아 수행된 지자체-대학 협력기반 지역혁신 사업(2021RIS-004)의 결과임

* 본 연구는 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. 2022-0-01197, 융합보안대학원(순천향대학교))

† 주저자, xeraph@sch.ac.kr

‡ 교신저자, yim@sch.ac.kr(Corresponding author)

I. 서 론

최근 고도화된 IT 기술에 따라 악성코드는 분석을 어렵게 하기 위해 여러 보안 기술을 적용하고 있다. 따라서 정적분석 진행 시, 많은 어려움을 겪는다. 정적분석을 진행할 때, 동적분석과는 다르게 모든 코드에 대해서 무조건 실행될 코드 부분뿐만 아니라 모두 분석해야 한다는 특징을 가지고 있다. 하지만, 요즘 작성되는 악성코드들은 난독화, 패킹, 코드 및 섹션 재정렬 사용, 레지스터 변경, 분기문 추가 등을 이용해 기존 코드를 수정함으로써 기존의 시그니처 탐지 및 패턴 탐지 기법이 제대로 효과를 발휘하지 못한다[1]. 이는 변종 악성코드라고 하며 이에 대응하고 탐지할 수 있는 기술은 아직 개발 중이나 시그니처 및 패턴 탐지 기술과 같은 정적분석의 한계가 있어 악성코드 및 변종 악성코드를 탐지하기에는 어려운 상황이다.

난독화되거나 패킹된 악성코드에 정적분석을 진행하기 위해서 언패킹이나 난독화를 푸는 방법이 있지만, 결과 코드에 기존의 정적분석 방법들을 적용하기가 어렵다. 하지만, 머신러닝을 활용한 탐지 기술의 경우 머신러닝만의 특징이자 장점을 사용하여 이를 해결할 수 있다. 기존의 데이터로만 악성코드를 탐지하는 것이 아닌 데이터를 기반으로 하며, 취합 및 통계적인 특성을 추출하여 결과를 도출한다[2].

또한, 기존의 머신러닝 기반 정적분석은 일반적으로 어셈블리어(assembly language) 기반이었으나 디스어셈블러의 비효율적인 효과에 따라 분석의 한계가 존재한다. 따라서 본 논문에서는 컴파일 상위레벨 중 중간언어인 IR(Intermediate Representation) 코드로 리프팅(lifting) 한다. 어셈블리어로 표현되지 않았던 모든 과정을 표출하고 표출된 코드를 통해 머신러닝만의 특징인 통계적 특징을 추출하여 새로운 탐지 기술을 제시한다.

본 논문에서 제시하는 악성코드 탐지 기술은 LLVM IR 코드 기반으로 적은 데이터를 가지고 높은 효율을 낼 수 있으며 비슷한 특징을 가진 데이터들을 인식할 수 있는 CNN(Convolutional Neural Network) 전이학습 알고리즘 모델을 사용하였다. 이는 악성코드를 탐지할 수 있고 비슷한 특징을 가진 변종 악성코드도 머신러닝의 특성을 통해 탐지할 수 있다. 또한, TensorFlow의 Keras에서 지원하는 ResNet50v2를 사용하여 탐지 안전성을 높였다.

본 논문의 구성으로는 다음과 같다. 2장에서는 본 논문과 관련된 연구를 소개하고 3장에서는 LLVM IR에 관한 배경지식을 설명한 후, 본 논문에서 사용된 디컴파일 방법을 서술한다. 4장에서는 LLVM IR를 통한 활용방법을 소개하며 5장에서는 CNN 모델에 대한 구성을 설명하고, 6장에서는 CNN 학습 결과를 제시하고, 제7장에서 결론으로 마무리한다.

II. 배 경

2.1 IR(Intermediate Representation)

IR은 컴파일 과정 중 존재하는 중간 단계 중 하나이다. 이는 어셈블리어에서 표현되지 않는 과정을 표출한다. 이에 현시점의 일반적인 어셈블리어 정적분석보다 더 나은 정적분석을 할 수 있다. 이러한 IR 코드도 여러 종류가 존재하며 대표적으로 LLVM IR, P-code 등이 존재한다. 본 논문에서는 아키텍처의 구조나 바이너리의 구조의 영향을 받지 않고 독단적으로 하나의 구조를 정의하여 데이터의 일관성을 확보할 수 있는 LLVM IR을 사용한다.

중간 표현 언어인 IR은 대상의 세부정보를 추상화하는 강력한 형식의 RISC(Reduced Instruction Set Computing) 명령 집합이며, 특정 기계에 국한되지 않는 특징을 갖고 있다. 고정된 레지스터 집합 대신에 중간 표현 언어인 IR은 %0, %1과 같은 형식의 임시 집합을 사용한다. 컴파일러는 보통 C, C++, Java 등 프로그래밍 언어로 작성한 코드를 타겟 아키텍처에 독립적인 IR로 변경하는 부분과 IR을 타겟 아키텍처의 기계 코드로 변경하는 부분으로 나뉜다. IR은 소스 언어와 독립적이면서 semantic 분석 단계에서 실제 어셈블리 코드를 직접 생성하여 대상 시스템의 작업을 표현해준다. 이에 컴파일 과정 중 semantics 과정을 거치고 IR Generator 통해 IR 코드가 생성된다.

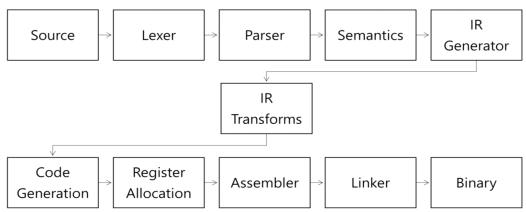

Fig. 1. Process of generating IR during the compile process

2.2 LLVM 프레임워크

LLVM은 여러 플랫폼에서 빠르게 실행되며 어셈블리와 유사한 낮은 수준의 프로그래밍 언어인 IR 코드를 생성해주는 컴파일러이다. 어느 아키텍처 환경에서도 자신이 정의하는 하나의 공통된 구조로 이루어진다. 이에 LLVM은 여러 아키텍처의 차이를 해결할 뿐만 아니라 LLVM이 가지고 있는 독자적이고 일관적인 IR 코드 즉, LLVM IR 코드를 사용할 수 있기에 본 논문에서 LLVM을 사용하였다.

LLVM 관련 프로젝트, 중간 표현, 디버거, C++표준 라이브러리 구현 적용에 사용된다[3].

LLVM은 프론트엔드(frontend)와 백엔드(backend)로 구분 할 수 있다. 프론트엔드는 언어를 정의하고 백엔드는 바이너리 코드를 생성하는 구조이며, 사람이 읽을 수 있는 어셈블리어 형식인 Three address code 형식의 IR을 지원한다. 이는 LLVM IR을 의미하며 프론트엔드에서 IR 코드를 가져와 LLVM IR을 생성하고 이후, 생성된 LLVM IR을 최적화 후 백엔드에서 바이너리 코드를 생성한다. 이를 기반으로 컴파일러 시스템이 IR 코드를 제공하며, LLVM만의 IR로 변환을 통해 Intel 및 AMD 등과 같은 아키텍처에 대한 종속적인 어셈블리 언어를 생성할 수 있다.

LLVM IR은 언어 독립적인 명령어 집합을 지원한다. 각 명령어는 Static Single Assignment 방식으로 생성된다. 즉, 데이터 유형이 레지스터라고 변수가 한 번 할당하면 이후 레지스터로 고정된다.

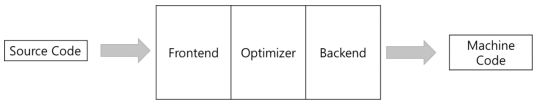

Fig. 2. Structure of LLVM

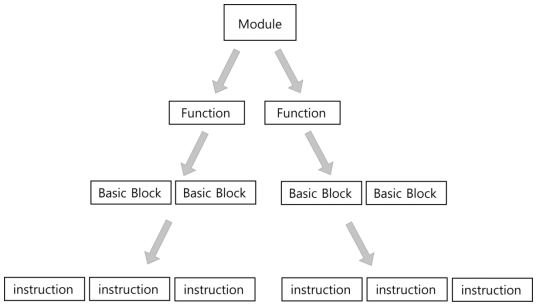

Fig. 3. Structure of LLVM IR

이러한 방법은 변수 간의 종속성 분석을 단순화하여 도움이 된다.

LLVM IR 코드의 구조는 가장 큰 개념인 모듈(module)이 존재한다. 그 이후에는 이 모듈에 단계적으로 구성요소가 포함되어있는 형태로 구성되어 있다. 모듈은 글로벌 변수와 지역 변수를 포함하고 있는 함수 여러 개로 구성되어 있다. 그 함수를 구성하는 요소는 BasicBlock이라고 칭한다. 이 BasicBlock 구조는 single entry와 single exit section을 나타낸다. 이 구조의 표현은 각 명령어로 진행한다. 결과적으로 LLVM IR 코드는 65여 가지의 명령어 및 나머지 플랫폼에 맞는 각각의 Intrinsic Function으로 이루어져 있다.

2.3 LLVM IR 리프팅 방법

위와 같은 LLVM IR을 확보하는 방법으로는 여러 가지의 방식이 존재한다. 소스코드를 컴파일하는 과정 중 IR 코드가 생성되는 과정까지만 컴파일하는 방법, 컴파일된 바이너리 파일을 역으로 컴파일하여 IR 코드 생성되는 부분으로 다시 올라가는 방법(리프팅), IR 코드 자체를 정의하여 바이너리 파일을 IR로 변환해주는 방법이 있다. 본 논문에서는 컴파일된 바이너리 파일을 역으로 컴파일하는 RetDec 디컴파일러를 사용하였다.

2.3.1 RetDec(LLVM IR decompiler)

RetDec은 LLVM 기반의 오픈소스 디컴파일러이다. 이 디컴파일러의 특징으로는 특정 대상 아키텍처, 운영 체제, 실행 파일 형식으로 제한되지 않으며, ELF, PE, Mach-O, COFF, Intel HEX, raw machine code등의 파일 형식을 지원한다는 점이 있다. RetDec은 자세한 정보가 포함된 실행 파일의 정적분석을 제공하며 컴파일러와 패커 탐지 및 고급 언어인 C, python 유사 언어로 출력하는 특징을 갖고 있다. 또한, OS 로더 시뮬레이션, High-level 언어를 구조화할 수 있는 기능이 존재한다. 이러한 기능들로 다양한 아키텍처 및 파일 포맷을 지원하며 LLVM IR을 기반으로 하여 바이너리에서 이미지를 추출하는 디컴파일러이므로 본 논문에서 사용하였다.

RetDec은 LLVM 에뮬레이터가 포함되어 있다. RetDec의 디컴파일 과정은 바이너리 파일 전처리를

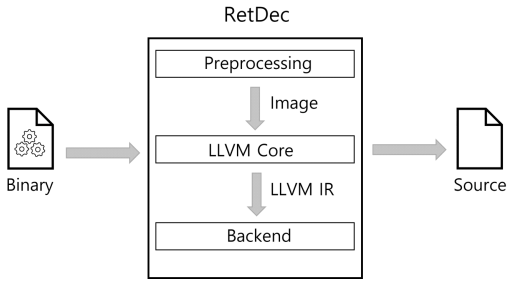

Fig. 4. Process of decompiling a binary by RetDec

통하여 이미지로 변환하고 LLVM Core를 통해 LLVM IR로 변환한다. 이후 백엔드 과정을 거쳐 최종적으로 소스 코드 파일이 생성된다.

2.3.2 RetDec 바이너리 파일 전처리 과정

바이너리 파일 전처리 상세 과정은 File format library에서 LLVM 형식으로 바이너리를 변환하고 Yara를 통해 패킹 및 압축 상태를 파악하고 JSON metadata를 추출한다. 그리고 압축 및 패킹이 되어 있다면 Image loader library를 통해 image를 로드해 언패킹을 진행하여 image 파일로 저장한다.

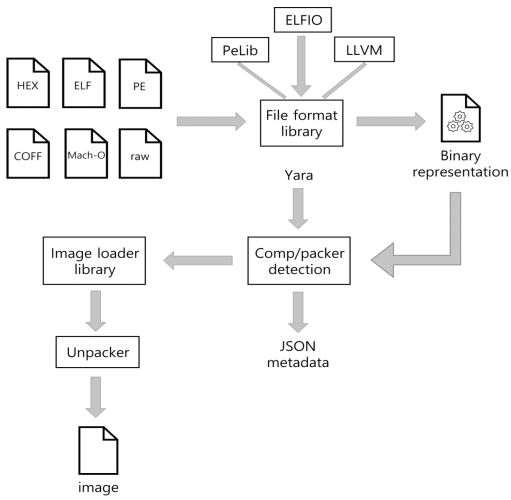

Fig. 5. RetDec preprocessing to make image

III. 관련 연구

기존 연구들을 통해 확인한 결과, 현재의 시그니처 및 패턴 탐지 기법의 한계점이 존재한다. 이에 여러 연구에서 머신러닝을 이용한 악성코드 탐지 기법이 필요함을 알 수 있었다. 이에 관련된 연구를 소개한다.

정적 분석 기반 기계학습 기법을 활용한 악성코드 식별 시스템 연구에서는 지속적으로 증가하는 악성코드 침해 공격으로 인해 발생하는 시그니처 기반 탐지 방법의 악성코드 탐지 한계점에 대응하였다. 유사성 해시 기반의 패턴 탐지 기술과 패킹에 따른 파일 분류 후 정적 분석 적용으로 기계학습 기반 악성코드 식별이 가능한 시스템을 제안하였다. 이는 기존에 알려진 악성코드의 식별에 강한 패턴 기반 탐지, 신규 및 변종 악성코드 탐지에 유리한 기계학습 기반 식별 기술을 모두 활용하여 보다 효율적인 탐지가 가능하다. 실험은 유사성 해시 알고리즘인 ssdeep, TLSH, DHASH을 사용하여 진행하였다. 학습 데이터는 한국인터넷진흥원에서 제공한 데이터와 정보보호 R&D 데이터 챌린지 2018 대회의 AI 기반 악성코드 탐지 트랙에서 제공한 정상 파일, 악성코드를 포함하여 58만 개의 데이터를 사용하였으며 95.79% 이상의 정확도를 도출하여 분석 성능을 확인하였다. 향후 지속적인 연구를 통해 패킹된 파일의 특성에 맞는 feature vector와 탐지 기법 추가 적용을 통해 탐지 성능을 높이는 시스템 구축이 가능할 것이라고 기대 효과를 언급하였다[4].

LLVM IR을 이용한 악성코드 탐지 관련 연구들을 통해 정적 분석과 함께 동적 분석까지도 폭넓게 LLVM IR을 이용한 악성코드 분석 방법이 연구되고 있다.

자체 수정 코드의 유무를 탐지하는 정적 분석 방법이 존재한다. 자체 수정 코드는 실행되는 동안 자신의 코드를 수정하기에 정적 분석을 우회하는 데 사용된다. 따라서 악성코드를 분석 및 탐지하기 위해서는 해당 코드의 유무를 검출할 수 있어야 한다. 이와 같이 정적 분석을 위한 도구를 LLVM IR에 기반으로 만든다면 효과적인 분석이 가능하나, 바이너리 리프팅 되었을 때 거짓 양성이 다수 포함되어 있으므로 바이너리를 정적 분석에 알맞도록 바이너리 리프팅을 진행해야 한다[5].

다양한 디바이스의 확대로 인해 여러 플랫폼을 상대로 하는 멀웨어 프로그램 또한 증가하고 있다. 소프트웨어 보안 회사는 이러한 플랫폼을 상대로 하는 보안 기술이 충분히 확보되지 않았으며, 몇 개의 기술만이 플랫폼에 대해 독립적인 상황이다.

한 연구에서는 다시 타게팅할 수 있는 리버스 컴

파일러(retargetable reverse complier)를 제안한다. 제안한 컴파일러는 아직 개발 초기 단계이다. 본 컴파일러는 플랫폼의 특정한 바이너리 어플리케이션을 high-level language로 변환해서 표현해주며, 동일한 방법으로 추후에 분석할 수 있으므로 정적분석에 도움이 될 것으로 기대할 수 있다. 이 연구의 독특한 솔루션은 아키텍처 표현 언어인 ISAC과 LLVM 시스템에 기초를 두고 있다. 본 논문에서는 이러한 방법으로 만들어낸 기술이 읽을 수 있는 high-level language 코드를 생성하는 것을 확인하였다[6].

동적 분석이 진행되지 않는 대표적인 케이스는 DDL(Data Definition Language) 파일이 존재하지 않는 경우, 인자가 필요한 경우, 가상머신을 탐지하는 경우 등 다양한 조건들로 악성 행위를 분석하기 어렵게 만드는 트리거가 존재한다. 이 연구에서는 트리거가 필요한 악성코드에 대해 바이너리 리프팅 기술을 활용하여 새로운 동적 분석 방법을 제안하였다. 바이너리 리프팅은 소스 코드가 없는 바이너리를 LLVM IR로 변환하여 특정 아키텍처에 맞게 수행할 수 있도록 하는 기술로서 이를 활용하여 입력 값 유무에 따른 악성코드를 판별한다. 전달 인자를 사용하는 코드와 사용하지 않는 코드 간의 LLVM IR을 비교 분석하여 전달 인자에 따른 악성코드 동작 여부를 판별하여 대량의 악성코드 동적 분석 시스템의 분석률을 높이는 방안을 제안하였다. 이러한 방법은 바이너리가 필요로 하는 입력 값의 카운트를 기반으로 임의의 값을 대입하여 동적 분석을 한 단계 더 진행할 수 있으며, 자동 분석이 되지 않는 대량의 악성코드를 분석할 수 있도록 하는 개선 방안으로 이어질 수 있다[7].

Opcode 빈도수 기반 악성코드 이미지를 활용한 CNN 기반 악성코드 탐지 기법 연구에서는 딥러닝 알고리즘을 활용한 악성코드 자동 분석 기법을 제안하였다. 제안한 악성코드 자동 분석 기법은 악성코드의 semantic한 정보를 탐지에 활용하기 위해 바이너리 명령어 Opcode 시퀀스 데이터로부터 2-gram 빈도수를 추출하였다. 이를 기반으로 생성한 Opcode 2-gram 빈도수 데이터를 이미지화하기 위해 군집화 기반 이미지 좌표 설정을 하였다. 이미지 변환 과정에서, 이미지를 구성하는 행과 열의 좌표 인덱스 설정을 위한 두 가지 군집화 방법 및 군집화 거리 계산 방법을 제안하였다. 이미지화한 데이터를 CNN 알고리즘을 이용하여 분석하였다. 학습에는 악성코드 10,000개, 정상 프로그램 10,000개로 구성된 대량의 데이터 셋을 활용하여 CNN 기반 악성코드 탐지 성능을 확인한 결과, 91%의 정확도로 악성 코드를 탐지할 수 있는 것을 확인하였다[8].

IV. LLVM IR 기반 악성코드 탐지 모델

4.1 기계학습 기반의 악성코드 탐지 기법 분석

최근 IT 기술의 발달로 매년 새로 생겨나는 악성코드의 양이 크게 증가하고, 기존의 시그니처 기반 악성코드 탐지 및 데이터 분석의 문제를 극복하기 위해 기계학습을 이용한 악성코드 탐지 기법을 소개하였다. 본 논문에서는 실행 파일의 바이너리 코드를 이미지화하였을 때 나타나는 특징을 기반으로 합성곱 신경망을 사용하였다. 학습 시 40만 개의 마이크로소프트 데이터셋을 이용하였으며 정확도 96.2%, 오탐률 0.1%을 확인하였지만 본 모델의 알고리즘은 행위 기반의 악성코드 탐지 한계가 존재한다. 기계학습은 악성코드 분석에 많은 도움을 줄 수 있지만, 악성코드가 시스템 메모리(RAM)에서 바로 실행되어 별도의 실행 파일을 남기지 않는 파일리스(fileless), 배포 소스로부터 모방 모델 제작 및 위조 입력 데이터를 만들어서 기계학습 모델을 공격하는 적대적 공격의 단점이 존재하여 이와 관련한 대응 방안 연구가 필요하다[9].

4.2 대용량 소스코드 취약점 스캔을 위한 CNN 모델

수백만 개의 이미지를 정확하게 분석할 수 있는 딥러닝 기반 이미지 분류기법들을 이용해 보안 측면에서 이 연구에서는 대규모의 소스코드 취약점을 스캔할 때, 확장성과 정확성 모두 확보한 모델을 제작하는 것을 목표로 한다. 따라서 프로그램의 세부정보를 보존하면서 소스코드를 효율적으로 이미지로 변환할 수 있는 새로운 아이디어를 제안하였다.

해당 연구는 함수 관점으로 취약점을 바라보았으므로 취약한 함수 13,689개와 취약하지 않은 함수 26,970개로 구성된 데이터셋을 이용해 모델을 구성하고 평가하였다. 최종적으로 2500만줄에 대한 연구를 추가 진행한 결과 VulCNN 모델이 대규모의 취약점을 탐지할 수 있음을 확인하였으며 스캔 결과를 통해 NVD 기준 보고되지 않은 73개의 취약점을 찾아낼 수 있었다[10].

4.3 LLVM IR 기반 악성코드 탐지 모델

이 연구는 숨겨진 마르코프 모델과 LLVM 중간 표현을 사용하여 변종 멀웨어를 탐지하는 새로운 방법을 제안한다. 새로운 접근 방식은 변종 코드에 존재하는 다양한 불확실한 변환을 LLVM IR로 통일하여 단순화함으로써 HMM의 정확도를 향상시키는 것을 목적으로 한다. 이 연구는 본 연구에서 제안하는 것과 다르게 구조화되지 않은 어셈블리 언어 코드를 단순화된 LLVM IR로 변환한다는 차이점이 있다. 이 연구는 LLVM IR을 사용해 많은 코드 난독화가 역전되어 단순화된 형태의 명령어가 생성되는 현상을 가지고 악성코드를 탐지할 수 있다고 주장하였다[11].

4.4 CNN ResNet50V2

CNN은 다른 머신러닝 알고리즘과 다르게, 데이터의 특징을 수동으로 지정 및 추출할 필요 없이 데이터로부터 직접 특징을 추출하여 학습되기에 자동화에 알맞은 기술이다. 이러한 점을 통해 LLVM IR을 이미지로 변환하고, 변환된 이미지의 특징을 자동으로 인식하여 학습하고 이를 분류할 수 있기에 본 논문에서 언급된 시그니처 및 패턴 탐지 기법의 한계를 해결할 수 있어 LLVM IR 코드의 활용방법으로 CNN 알고리즘을 사용하였다. 그리고 CNN은 여러 학습 알고리즘이 있으며 그중 이미지 분류에 매우 적합하고 모델을 안정적으로 학습시킬 수 있도록 해주는 알고리즘인 ResNet50V2 즉, 전이학습 알고리즘을 사용하였다. 이에 ResNet50V2을 사용하여 변환된 LLVM IR 이미지를 악성코드 이미지와 일반 바이너리 파일을 구분하여 분류하는 것에 매우 적합하여 본 논문에서 사용하였다.

ResNet은 Shortcut을 사용하여 학습률을 높인 모델을 의미한다. 과거의 머신러닝 모델은 발전할수록 레이어의 증가로 인해, 인공신경망이 깊어질수록 Gredient Vanishing 문제가 심화되는 Degradation Problem 현상이 발생하였다. 이러한 문제의 원인은 Weight의 분포가 균등하지 않고 역전파가 제대로 이루어지지 않기 때문이다.

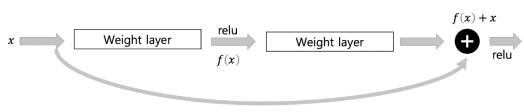

Fig. 6. Process of residual learning

이를 해결하기 위해 Residual Learning을 사용하는 Shortcut Connection을 진행하여 올바른 학습이 되도록 한다. 이러한 방법은 레이어 간 연결이 순서대로 연속적으로 존재하게 만드는 것이 아니라, 중간을 뛰어넘어 전달하는 shortcut이 추가된 것이다.

기존 네트워크를 $H(x)$(x는 layer의 input)라고 할 때, $F(x)=H(x)-x$로 네트워크를 변형시켜서 $F(x)+x$를 $H(x)$에 근사하도록 학습하는 것이다. layer에 관한 입, 출력의 차이에 대해 학습이 되어 문제를 해결할 수 있으며, 이러한 점이 residual learning이며, Shortcut Connection에 적용이 되어있다.

Shortcut Connection 알고리즘은 이전에는 입력 이미지의 크기를 다양하게 하거나 필터 크기를 다양하게 한 후, 각 값을 계산하는 형식을 사용하였다. 이렇게 되면 이전의 것이 다음 값으로 전달되어 영향을 미치게 된다. 이렇게 되면 입력부 분에 가까운 하위 레이어에는 매우 단순한 구조가 이루어져 있지만, 상위 레이어는 매우 구조적이고 복잡한 부분이 학습되는 특징을 가지고 있다. 하지만 Shortcut Connection을 사용하면, 이미지를 직접 계산하는 것이 아닌 수식적으로 비교했을때, 현재 레이어의 출력값과 이전 레이어의 출력값을 더해 입력을 받아 이전의 값에서 현재 값이 얼마나 바뀌었는지 나머지(residual)를 계산해주기 때문에, 그 나머지만 계산하면 되므로 모델 전체에 부하가 줄어들게 된다. 결과적으로 이 알고리즘을 적용하여 기존의 CNN 모델보다 더 빠른 모델이 생성된다[12].

4.5 이미지 변환

본 논문에서는 위와 같이 CNN 전이학습 알고리즘을 선정했으므로, 이미지 변환 과정이 요구된다. RetDec으로 디컴파일하여 추출된 ".ll" 파일을 umpy 리스트 형식으로 파일을 불러온다. 이때 uint8 형식으로 불러오며 세로, 가로 사이즈 각각 동일하게 128 픽셀로 배열을 생성한다. 이후 만들어진 배열을 numpy 모듈의 reshape 함수를 통해 128 픽셀 만큼의 배열을 도트로 표현해 이미지를 생성한다. 이후, 회색 조 변환 기법(grayscale) 전처리를 진행하여 이미지를 어두운색과 밝은색으로 이진적 구분하여 CNN 모델 성능을 향상시켰다.

Algorithm 1: Converting LLVM IR into image

Input: LLVM IR L
Output: Image I
function LLVM_IR_to_IMAGE(L, I)
  parse L
  set width 128, height 128
  read width+height bytes from L as a 1D array of unsigned integers
  reshape the array into a 2D array with height, width
  create an image from the 2D array and **save** as I
  convert I into grayscale
end function

Fig. 7. Algorithm to convert LLVM IR into grayscale image

4.6 CNN 모델 학습을 위한 데이터셋 구성

악성코드 탐지 모델에 사용할 데이터셋의 구성을 설명한다. 데이터셋에는 1920개의 일반 프로그램 샘플, 악성코드 데이터베이스인 MalwareBazaar [13]에서 추출한 1920개의 악성코드 샘플이 존재하며, 총 3840개의 샘플로 구성되어 있다.

4.7 LLVM IR 기반 악성코드 이미지 탐지 절차

본 논문에서 제시하는 CNN 모델의 전반적인 프로세스는 바이너리 파일을 RetDec을 통해 LLVM IR로 변환한 후 이미지로 다시 한번 더 변환한다. 이를 데이터 셋으로 구성한 후 CNN ResNet50V2 즉, 전이학습 알고리즘으로 학습을 진행하여 모델을 제작하고, 제작된 모델을 통해 해당 바이너리 파일이 악성코드인지 탐지하는 것이 CNN 모델의 최종 모습이다.

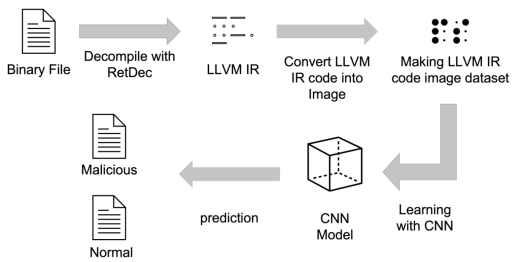

Fig. 8. Overall process of the model

4.8 CNN 모델 구성

전처리 과정 후 데이터의 정상, 악성 여부를 구분한 데이터셋을 구성하고 CNN 알고리즘을 이용한 학습을 진행한다. 본 논문에서는 CNN을 이용한 알고리즘 중 전이학습을 사용하기 위해서 TenserFlow[14]의 Keras에서 지원하는 전이 학습 기능이 내장되어있는 ResNet50V2[15]라는 사전에 훈련된 모델을 사용하여 자동으로 필터 생성 및 모델이 안정적으로 생성하도록 한다.'

본 논문의 모델처럼 전이학습을 진행한 기계학습 모델은 각 데이터의 공통점을 찾는 특징을 갖고 있어, 새로운 변종 악성코드가 존재하여도 공통적인 특징을 기반으로 탐지할 수 있는 이점이 존재한다.

CNN을 전이학습 모델로 설정하고 input shape는 데이터셋 이미지 사이즈인 128로 지정하였고, Optimizer는 adam으로 지정하였다. 그리고 adam으로 지정한 이유는 구현이 간단하고 그 안의 연산이 타 optimizer에 비해 효율적이기에 사용하였다. 이후 batch size는 4, epochs는 10, verbose는 1로 지정하여 학습을 진행하였다.

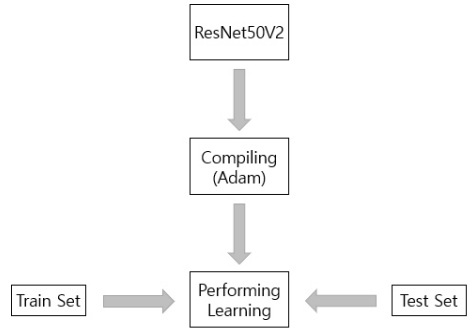

Fig. 9. Diagram of the learning process after model creation

```
model = ResNet50V2(include_top=True, weights=None, input_shape=(128, 128, 1), cl
asses=2)
model.summary()

model.compile(optimizer = 'adam',
              loss = 'categorical_crossentropy',
              metrics = ['accuracy'])

history = model.fit(X_train, y_train,
                    batch_size = 4,
                    epochs = 10,
                    verbose = 1,
                    validation_data=(X_test, y_test))
```

Fig. 10. CNN model source code

V. 악성코드 탐지 모델 성능 평가

본 논문에서 학습된 모델을 약 500개의 테스트 데이터 셋을 별도로 구축하여 테스트를 진행한 결과 정확도는 약 0.9, 정밀도는 0.87, 재현율은 0.93, 마지막으로 ROC AUC는 0.89로 평균 이상의 준수한 수치를 확인할 수 있다.

결과를 직관적으로 확인하기 위해 학습된 모델과 구성된 데이터셋을 통하여 무작위로 10개의 파일을 선정하여 예측을 진행하였다. 이는 데이터셋에 라벨링을 해놓았으나 예측을 진행할 땐 고려하지 않고 최종 결과를 확인 시에만 표출하도록 하였다. 최종적으로 10개의 파일 중 9개의 정상 예측을 하고 1개의 파일은 제대로 예측하지 못하였다. 이는 정확도 최대 90.02%의 모델을 사용하였으며, 정확도에 알맞은 예측값을 보여주고 있다.

또한 오차 행렬 성능 평가 지표를 통해 471개의 데이터의 분류상태를 확인할 수 있다.

총 471개의 데이터 중 191개를 normal로 인식하였고, 이는 실제로 normal로 정확하게 인식하였으나 32개의 데이터는 normal 데이터이지만 malware로 인식되었다. 또한 233개의 데이터는 malware로 정확하게 인식되었지만, 15개의 malware는 normal로 인식하였다. 이러한 결과를 바탕으로 모델의 민감도를 평가하면 malware의 경우 233/(233+15) = 0.939로 나타나며, normal의 경우 191/(191+32) = 0.856으로 나타났다.

허나 이 수치는 과적합으로 예상되어, 데이터셋 구성을 악성코드 900개에서 1920개, 일반 프로그램 900개에서 1920개로 추가하여 총 1800개 데이터셋에서 3840개 데이터 셋으로 추가하여 구성하였다.

이후 테스트셋을 일반 프로그램 3000개, 악성코드 3000개로 진행하여 총 6000개의 테스트셋을 구

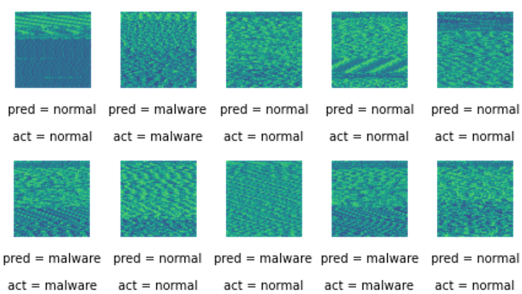

Fig. 11. Malicious code detection results

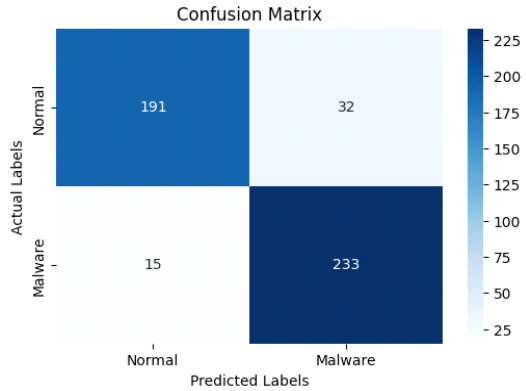

Fig. 12. Confusion matrix results

성하여 정확도를 측정해 정확도 63.4라는 결과를 나타냈다. 그리고 본 논문에서 제시한 모델 외 다른 탐지 모델과 비교하여 성능 비교를 진행하였다.

비교한 타 모델 CyberMachine이라는 오픈소스 라이브러리이며, Decision Tree, Random Forest, K-최근접 이웃 알고리즘 3개를 사용해 타겟 파일의 PE 정보를 분석하여 이 파일에 대해 2개 이상의 알고리즘이 양성이라고 판단 되는 경우 이 파일을 악성 파일이라고 판단하는 모델이다.

모델 CyberMachine 또한 약 2000개의 데이터셋을 오차 행렬 성능 평가 지표를 통해 평가를 진행하였고, normal 데이터의 모델 민감도는 494/(494+506) = 0.49로 나타났으며, malware 데이터의 경우 600/(600+359) = 0.625 수치를 나타냈다. 따라서 본 논문에서 제시한 악성코드 탐지 모델이 CyberMachine 모델과 비슷한 성능을 가져 악성코드 탐지가 정상적으로 진행이 된다고 판단할 수 있다[16].

또한 다른 모델인 DeepMalwareDetector를 본 논문에서 제시한 CNN 모델 데이터셋과 동일하게 진행하였으나, 이 모델에서는 이미지 사이즈를 256 크기로 지정하였기에, 이 사이즈에 맞춰 학습을 진행하였다. 이에 정확도 100%라는 과적합 현상이 발생하였다. 결론적으로 타 모델들과 본 논문에서 제시하는 모델을 비교하였을 때, 본 논문에서 제시하는 모델이 비교적 정상적인 탐지가 가능한 모델이라고 평가할 수 있다[17].

VI. 결 론

본 논문에서는 기존의 시그니처 및 패턴 탐지 기법과 같은 악성코드 정적분석의 한계점을 해결할 수 있는 악성코드 정적분석 자동화 기술을 제안한다. 컴파일 중간 단계인 IR 코드로 리프팅 해주는 RetDec을 기반으로 하여 LLVM IR 코드로 디컴파일한다. 이후, LLVM IR 코드를 이미지로 변환하여 전이 학습 알고리즘 중 하나인 ResNet50v2을 사용하였다. 정상 코드 1920개, 악성코드 1920개 총 3840개의 데이터셋을 기준으로 학습된 모델의 정확도는 최대 64.00%, 평균 63.4%의 악성코드 탐지 성능을 확인하였다. 이를 통해 IR 코드기반 이미지 변환으로 악성코드 탐지가 가능하다는 결과를 얻을 수 있었고, 추후 연구를 통해 향상된 성능의 모델을 개발하는 것이 목표이다.

References

[1] JungBeen Yu, MinSik Shin, and Taekyoung Kwon, “Analysis of Research Trend on Machine Learning Based Malware Mutant Identification,” .REVIEW OF KIISC, 27(3), pp. 12-19, Jun. 2017

[2] Bong Ki Jung and Kim Jong Hyun, “A Survey on Malware Detection using Enhanced Data Mining Techniques,” KICS Summer Conference, 2021, pp. 1026-1027, Jun. 2021

[3] The LLVM Compiler Infrastructure “LLVM“, llvm.org, 8th Aug. 2023

[4] Su-jeong Kim, Ji-hee Ha, Soo-hyun Oh and Tae-jin Lee, “A Study on Malware Identification System Using Static Analysis Based Machine Learning Technique,” Journal of the Korea Institute of Information Security & Cryptology, 29(4), pp. 775-784, Aug. 2019

[5] Jae-IL Yu and Kwang-hoon Choi, “An LLVM-Based Implementation of Static Analysis for Detecting Self-Modifying Code and Its Evaluation,” Journal of the Korea Institute of Information Security & Cryptology, 32(2), pp. 171-179, Apr. 2022

[6] Lukáš Ďurfina, Jakub Křoustek, Petr Zemek, Dušan Kolář, Tomas Hruska, Karel Masařík, and Alexander Meduna, “Design of a retargetable decompiler for a static platform-independent malware analysis,” Information Security and Assurance: International Conference, pp. 72-86, Aug. 2011.

[7] Shin-Woon Hwang and Jonghee Youn, “A study of malware argument detection,” A study of malware argument detection, The Korea Information Processing Society Spring Conference 2021 , 28(1), pp. 181-182, May. 2021

[8] Seok Min Ko, JaeHyeok Yang, WonJun Choi, and TaeGuen Kim, “CNN-Based Malware Detection Using Opcode Frequency-Based Image,” Journal of the Korea Institute of Information Security & Cryptology, pp. 933-943, 32(5), Oct. 2022

[9] Dong-Geun Lee, ″Analysis of Malware Detection Techniques based on Machine Learning,″ Master's Thesis, Department of Convergence Service Security Engineering Graduate School of Soonchunhyang University, Feb. 2018

[10] Yueming Wu, Deqing Zou, Shihan Dou, Wei Yang, Duo Xu and Hai Jin, ″VulCNN: An Image-inspired Scalable Vulnerability Detection System,″ ICSE ′22: Proceedings of the 44th International Conference on Software Engineering, pp. 2365-2376, May. 2022.

[11] Mahajan, Ginika, and Raja. ″Metamorphic Malware Detection Using LLVM IR and Hidden Markov

Model," Proceedings of the International Congress on Information and Communication Technology, pp. 411-421, Jun. 2016.
〔12〕 Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778, Jun. 2016.
〔13〕 Malware Bazaar by ABUSE "Malware Bazaar", https://bazaar.abuse.ch, 10th. Aug. 2023
〔14〕 Tensorflow,"TensorFlowjs" https://www.tensorflow.org/js?hl=ko, 11th. Aug. 2023
〔15〕 Tensorflow,"ResNet50V2" https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet_v2/ResNet50V,11th. Aug. 2023
〔16〕 github,"CyberMachine" https://github.com/emr4h/CyberMachine, 20th. Aug. 2023
〔17〕 github,"DeepMalwareDetector" https://github.com/islem-esi/DeepMalwareDetector, 21st Aug. 2023

〈저자소개〉

박 경 빈 (Kyung-bin Park) 학생회원
2024년 2월: 순천향대학교 정보보호학과 졸업
〈관심분야〉 정보보호, 시스템보안

윤 요 섭 (Yo-Seob Yoon) 학생회원
2024년 2월: 순천향대학교 정보보호학과 졸업
〈관심분야〉 정보보호, 바이너리분석

또 올 가 (Baasantogtokh Duulga) 학생회원
2023년 2월: 순천향대학교 정보보호학과 졸업
2023년 3월~현재: 순천향대학교 모빌리티융합보안학과 석사과정
〈관심분야〉 정보보호, 취약점분석

임 강 빈 (Kang-bin Yim) 종신회원
1992년 2월: 아주대학교 전자공학과 졸업
1994년 2월: 아주대학교 전자공학과 석사
2001년 2월: 아주대학교 전자공학과 박사
2003년 3월~현재: 순천향대학교 정보보호학과 교수
〈관심분야〉 정보보호, 취약점분석, 자동차보안