Opcode 및 Call Graph 기반 악성코드 탐지 연구

김진하[1], 임을규[1]
[1]한양대학교 컴퓨터소프트웨어학과
bradypus404@hanyang.ac.kr, imeg@hanyang.ac.kr

A Study on Malware Detection based on Opcode and Call Graph

Jinha Kim[1], Eul Gyu Im[1]
[1]Dept. of Computer Science, Hanyang University

요 약

본 논문에서는 악성코드 탐지를 위한 정적 분석 기반의 새로운 기법을 제안한다. 본 연구는 악성코드 파일에서 Opcode와 Call Graph를 각각 추출하고, 이를 바탕으로 두 가지 머신러닝 모델을 학습한 후 앙상블 기법을 사용하여 최종적으로 악성코드를 분류하는 방법론을 제시한다. Opcode 기반 분석에서는 어셈블리어를 추출하여 정규표현식을 이용해 명령어를 추출했고 N-gram 기법을 활용해 명령어의 흐름을 분석한다. Call Graph 분석에서는 함수 호출 관계를 추출하고 Diffpool 기법을 통해 그래프를 벡터화한다. 제안된 시스템은 개별 모델의 장점을 결합하였고 적은 데이터에 비해 높은 정확도와 탐지율을 달성한다. 특히, 신종 악성코드나 변종 악성코드를 탐지하는 데 있어서 기존 서명 기반 기법에 비해 더 높은 성능을 보일 가능성이 높다. 실험 결과, 91%의 정확도와 88%의 재현율을 보여주어 높은 결과를 나타냈다. 본 연구는 악성코드 탐지 분야에서 정적 분석 기법의 가능성을 제안한다.

1. 서론

악성코드는 컴퓨터 시스템과 네트워크에 부정적인 영향을 미치는 프로그램으로 전 세계적으로 큰 위협이 되고 있다. 악성코드 탐지 및 분류는 정보보안에서 중요한 과제이며, 다양한 기법들이 개발됐다. 기존의 서명 기반 탐지 기법은 알려진 악성코드를 탐지하는 데 효과적이지만 새로운 변종 악성코드를 탐지하는 데에는 한계가 있다. 그리고 악성코드를 동적 분석하기 위하여 실행시키면 위험성이 있고 가상환경에서 실행시키기 때문에 특정 기법에 따라 추출된 결과가 다를 수 있다. 이에 따라, 악성코드의 행위 패턴을 기반으로 동적 분석 기법이나 악성코드 파일 자체의 구조적 특징을 이용한 정적 분석 기법이 연구되고 있다. 본 연구에서는 정적 분석 기법을 사용하여 악성코드의 Opcode와 Call Graph를 추출하고, 이를 머신러닝 모델에 적용하여 악성코드를 분류하는 시스템을 제안한다. 제안된 시스템은 두 가지 분석 기법을 통합하여 높은 탐지 성능을 달성하고자 한다.

2. 관련 연구

가. Opcode 기반 접근 방식

Opcode 기반 탐지 기법은 악성코드의 명령어 흐름을 분석하여 악성 행위를 탐지하는 방법이다. 이 기법은 악성코드 파일의 어셈블리 코드에서 명령어 Opcode를 추출하고, 이를 텍스트 마이닝 기법으로 특징 벡터를 생성하여 머신러닝 모델을 학습한다.

Lee et al. (2023)은 30,000개의 IoT 악성코드를 탐지하기 위해 Opcode 범주 시퀀스를 N-gram 및 엔트로피 값 기반의 특징 벡터를 제안하고, 이를 다양한 머신러닝 모델에 적용하여 98% 이상의 높은 정확도를 달성하였다[1].

Renjie Lu (2023)은 악성코드의 어셈블리어를 Word Embedding하고 모델링하여 LSTM 기반으로 Opcode 시퀀스를 분석하고 탐지와 분류에서 각각 평균 AUC 0.99와 0.987을 달성하였다[2].

Hoang et al.(2023)은 Opcode n-gram을 사용한 통계 기반 및 머신러닝 기반 악성코드 탐지 모델을 제안하고, 각각 92.75%와 96.29%의 탐지 정확도를 달성하였다[3].

나. Graph 기반 접근 방식

그래프 기반 탐지 기법은 프로그램의 함수 호출 관계를 Call Graph로 표현하여, 프로그램의 전체적인 흐름과 행위를 분석한다. 이러한 그래프는 정적 분석 도구를 통해 추출되며, 각 노드는 함수 호출, 간선은 함수 간의 호출 관계를 나타낸다.

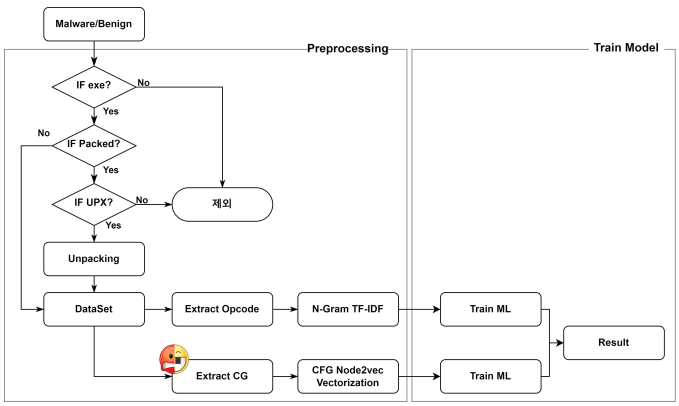
Mehadi et al. (2017)은 함수 호출 그래프를 벡터로 변환하여 머신러닝 모델에 입력으로 사용하는 기법을 제안하였다. 이 방법은 기존 그래프 편집 거리(GED) 기반 방법에 비해 계산 복잡도가 낮고 성능이 우수하여, 1000개의 악성코드 샘플을 대상으로 97.9%의 정확도를 기록하였다[4].

Wu et al. (2022)은 함수 호출 그래프와 그래프 임베딩 네트워크를 사용하여 악성코드 탐지 및 분류를 위한 GEMAL 시스템을 제안하였다. 실험 결과, WUFCG 데이터셋에서 99.16%의 탐지 정확도, BIG-2015 데이터셋에서 99.81%의 분류 정확도를 달성하였다.[5].

Mester et al. (2022)은 정적 호출 그래프와 그래프 합성 신경망(GCN)을 결합하여 악성코드 분류를 수행하는 시스템을 제안하였다. 실험 결과, 223개의 악성코드 패밀리로 구성된 데이터셋에서 0.576의 매크로 F1-score를 달성했으며, 기존의 호출 그래프 기반 방법보다 성능이 우수하였다[6].

3. 방법론

가. 구성도



(그림 1) 탐지 모델 구성도

본 연구에서는 exe 확장자와 Win32 환경에서 작동하는 파일을 대상으로 전처리 과정을 수행하고. 이를 기반으로 학습 모델을 구축하여 악성코드 탐지 시스템을 설계한다. (그림 1)과 같은 순서도에 따라 전처리 과정을 거쳐, Opcode와 Call Graph를 각각 추출하고, 이를 머신러닝 모델에 학습시킨 후 최종 결과를 도출한다. 본 장에서는 Opcode 추출 및 임베딩, Call Graph 생성 및 벡터화, 앙상블 학습의 세부 과정을 설명한다.

나. Opcode 추출 및 임베딩

전처리 단계에서는 악성코드 파일로부터 Opcode를 추출한다. 이 과정에서 PE(Portable Executable) 포맷의 헤더 정보, 바이너리 정보, 어셈블리 정보, 섹션 정보를 분석한다. 추출된 파일에서 정규표현식을 사용하여 Opcode만 선별하고 이를 머신러닝 모델에 사용한다. Opcode만 추출하는 이유는 Operand 값은 각 레지스터나 메모리 주소를 나타내어 프로그램마다 달라질 수 있으며, 이는 학습 시 편향을 일으킬 수 있기 때문이다. 따라서 Operand를 배제하고, 명령어 흐름만을 학습에 사용한다.

실행 파일에서의 특정 행위를 탐지하기 위해서는 여러 명령어의 흐름을 보는 것이 가장 중요하다. 정상 파일과 악성코드에서 특정 흐름이 반복적으로 보이면 흔한 정상 행위일 것이고 특정 악성코드들에서만 보이는 흐름이 있다면 해당 행위가 악성 행위와 관련될 가능성이 높다.

본 논문에서는 흐름을 추적하기 위하여 N-gram으로 여러 명령어 그룹을 만들고 TF-IDF 값을 측정해 반복적으로 많이 나온 것은 중요도를 낮추고 많이 나오지 않은 것에 대해 중요도를 높인다.

다. Call Graph 생성 및 벡터화

Opcode보다 전반적인 악성코드의 흐름을 분석하기 위해 Call Graph를 생성한다. Call Graph는 angr 도구를 사용하여 정적 분석을 통해 추출하며, 이 그래프는 악성코드의 함수 간 호출 관계를 나타낸다.

추출된 Call Graph는 Diffpool 기법을 사용해 고정된 차원의 벡터로 임베딩한다. Diffpool 기법은 전체 그래프를 하나의 고정된 차원의 벡터로 만드는 기법이로, 그래프의 인접 노드들을 묶어 하나의 노드로 축소하고 묶은 노드를 다시 인접한 노드와 결합하여 축소하는 작업을 반복하여 고정된 차원으로 출력한다[7].

ML 학습을 위해서는 고정된 차원을 입력값으로 넣어야 하므로 Diffpool을 사용하여 하나의 그래프가 몇 개의 노드로 구성되어 있던 고정된 차원으로 출력한다.

라. 앙상블 학습

Opcode와 Call Graph 기반의 모델을 각각 학습시킨 후, 두 모델의 출력을 결합하여 최종 결과를 도출한다. 앙상블 기법을 통해 두 모델의 장점을 결합하여 더 높은 탐지율과 정확도를 얻을 수 있다. 본 연구에서는 가중치 방식을 이용하여 두 모델의 예측 결과를 결합하여 최종적으로 악성코드를 분류한다.

4. 실험

가. 실험 환경

본 연구에서는 악성 155개와 정상 137개를 사용하여 학습을 진행한다. Opcode는 Windows 가상환경을 이용해 추출하고 Host Windows에서 학습한다. Call Graph는 Ubuntu 가상환경의 공식 angr[8] Docker를 이용하여 Call Graph를 DOT 파일로 추출하고 공식 Diffpool Docker를 이용해 벡터화를 진행하여 데이터 추출을 마친 후, Opcode와 마찬가지로 Host Windows에서 학습한다. 두 모델이 학습된 후 두 결과를 기반으로 가중치를 적용하여 정확도를 도출한다.

나. 학습 결과

Opcode와 Call Graph를 이용해 Random Forest, Support Vector Machine, Gradient Boosting, K-Nearest Neighbors 모델을 이용해서 정확도, 정밀도, 재현율(Recall), F1 Score를 측정한 결과 <표 1>, <표 2>와 같다.

<표 1> Opcode Model 성능 평가

모델	정확도	정밀도	재현율	F1-score
RF	0.9157	0.9172	0.9157	0.9156
SVM	0.8871	0.8888	0.8871	0.8870
GB	0.9200	0.9216	0.9200	0.9199
KNN	0.8643	0.8668	0.8643	0.8640

<표 2> Call Graph 성능 평가

모델	정확도	정밀도	재현율	F1-score
RF	0.8986	0.9172	0.9157	0.9156
SVM	0.8871	0.8888	0.8871	0.8870
KNN	0.9078	0.8668	0.8643	0.8640

Opcode에서는 Gradient Boosting 모델이 92%의 정확도로 제일 높았고 정밀도, 재현율, F1-Score에서도 모두 높은 것을 확인할 수 있다. 그 아래로는 RF, SVM, KNN을 순서대로 정확도를 확인할 수 있다. Call Graph 모델은 KNN 모델이 91%의 정확도로 가장 높은 결과를 보인다. Diffpool 임베딩 기법이 가까이 있는 노드를 그룹화하는 방법을 반복적으로 사용하므로 가까운 것을 기준으로 분류하는 KNN이 가장 높은 정확도를 보인다.

<표 3> 모델 앙상블 결과

모델	정확도	정밀도	재현율
GB+KNN	0.9151	0.8843	0.8940

Opcode와 Call Graph 모델에서 제일 정확도가 높은 모델을 선정하여 Bagging 기법으로 앙상블하여 정확도를 추출한 결과 <표 3>과 같이 정확도 91%, 정밀도 88%를 기록했다.

5. 결론

본 연구에서는 악성코드 탐지를 위해 Opcode와 Call Graph 기반의 정적 분석 기법을 이용하였으며, 앙상블 학습을 통해 적은 데이터에도 불구하고 높은 탐지율을 달성하였다. Opcode 기반 모델은 악성코드의 명령어 흐름을 분석하여 세부적인 특징을 파악할 수 있었으며, Call Graㄴph 기반 모델은 프로그램의 전반적인 구조적 특징을 학습함으로써 악성코드 탐지에 기여하였다. 본 연구 결과는 향후 악성코드 정적 탐지 시스템 개발에 유용하게 활용될 수 있을 것으로 기대된다. 향후에는 악성코드의 데이터양을 늘리고 Opcode와 Call Graph를 통한 각각의 모델에 대한 파라미터 값을 세부적으로 조정하여 정확도를 높일 예정이다.

사사문구

This work was supported by National Research Foundation (NRF) grant (No. NRF-2021R1F1A1061362) funded by the Korea government (MSIT), and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2021-0-00590).

참고문헌

[1] Lee, S., Kim, J., Park, H., "Robust IoT Malware Detection and Classification Using Opcode Category Features on Machine Learning,"

IEEE Transactions on Information Forensics and Security, vol. 18, no. 2, pp. 234-247, 2023.
[2] Lu, Renjie, "Malware Detection with LSTM using Opcode Language," Proceedings of the International Conference on Machine Learning and Cybernetics, Guilin, China, 2023, pp. 345-350.
[3] Hoang, X.D., Nguyen, B.C., Ninh, T.T.T., "Detecting Malware Based on Statistics and Machine Learning Using Opcode N-Grams," 2023 RIVF International Conference on Computing and Communication Technologies, Hanoi, Vietnam, 2023, pp. 118-123.
[4] Hassen, M., Chan, P.K., "Scalable Function Call Graph-based Malware Classification," Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, Scottsdale, AZ, USA, 2017, pp. 239-248.
[5] Wu, X., Wang, Y., Fang, Y., Jia, P., "Embedding vector generation based on function call graph for effective malware detection and classification" Neural Computing and Applications, vol. 34, no. 4, pp. 8643-8656, 2022.
[6] Mester, A., Bodó, Z., "Malware Classification Based on Graph Convolutional Neural Networks and Static Call Graph Features," Proceedings of the 35th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Kitakyushu, Japan, 2022, pp. 528-539.
[7] Ying R., You J., Morris C., Ren X., Hamilton W. L., Leskovec J., "Hierarchical Graph Representation Learning with Differentiable Pooling," Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vancouver, Canada, 2019
[8] Shoshitaishvili, Y., Wang, R., Salls, C., Stephens, N., Polino, M., Dutcher, A., Grosen, J., Feng, S., Hauser, C., Kruegel, C., Vigna, G., SoK: (State of) The Art of War: Offensive Techniques in Binary Analysis, IEEE Symposium on Security and Privacy, San Jose, 2016, pp. 138-157.