동적-정적 분석 데이터와 딥러닝을 이용한 난독화된 악성코드 탐지 기법

김해수[1], 김미희[2]
[1]한경국립대학교 컴퓨터응용수학부
[2]한경국립대학교 컴퓨터응용수학부, 컴퓨터시스템연구소
e-mail:{ww232330, mhkim}@hknu.ac.kr

Obfuscated malware detection Approach using Dynamic and Static Analysis Data and Deep Learning

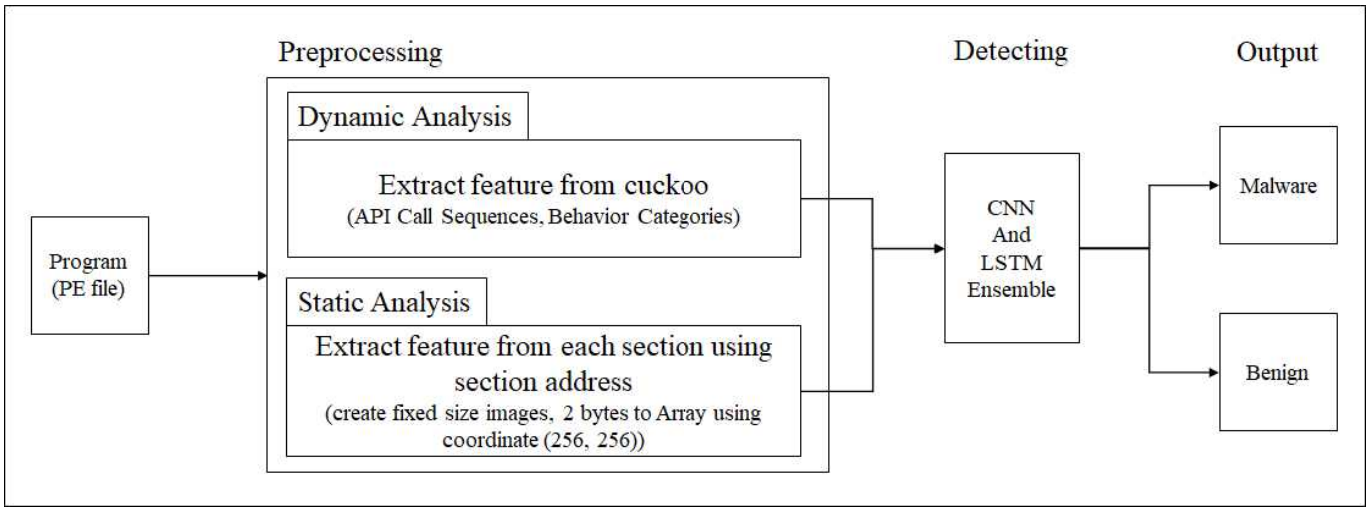
Hae-Soo Kim[1], Mi-Hui Kim[2]
[1]School of Computer Engineering & Applied Mathematics, Hankyong National University
[2]School of Computer Engineering & Applied Mathematics, Computer System Institute Hankyong National University

요 약

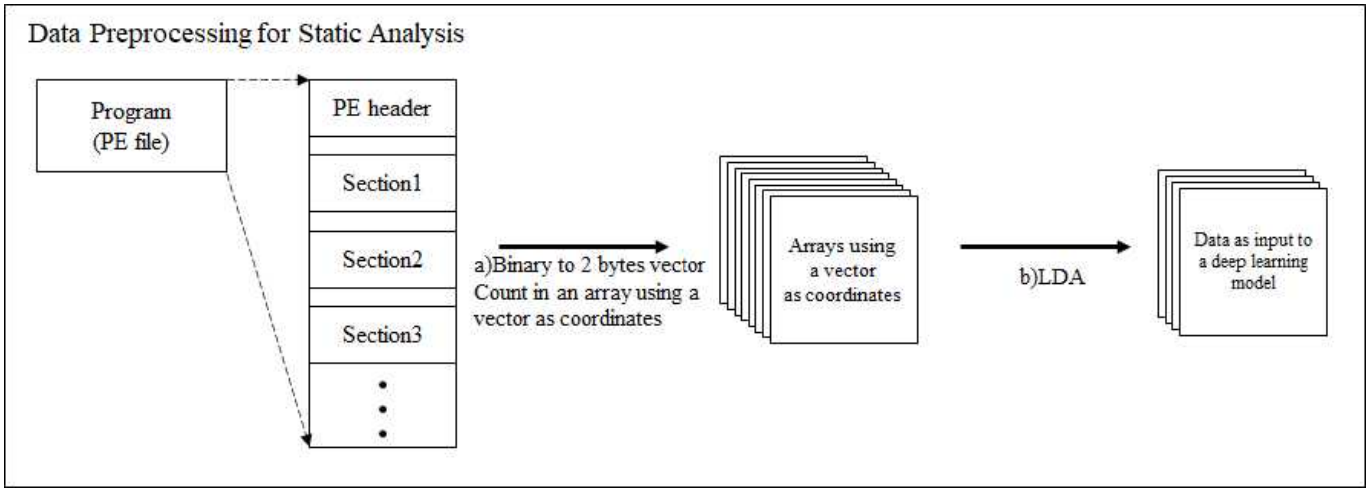
악성코드를 유포할 때 프로그램 코드만으로 악성코드의 유무를 확인할 수 없도록 조치하여 분석을 지연시키는 방식을 사용하는 방향으로 발전하고 있다. 악성코드를 실행하지 않고 코드와 구조만으로 분석하는 정적 분석으로는 악성코드를 판별할 수 없어 코드를 직접 실행해 분석하는 동적 분석을 이용해야 한다. 본 논문에서는 난독화된 비정상적인 코드를 직접 실행한 동적 분석데이터와 일반적이지 않은 섹션들의 정보를 추출한 정적 분석데이터를 이용해 동적-정적 분석 데이터와 딥러닝 모델을 통해 난독화 및 패킹된 악성코드를 탐지하는 기법을 제안한다.

1. 서론

악성코드의 지속 매커니즘 (Persistence Mechanism)이 발전하고 있다. 프로그램 코드만으로는 악성코드의 유무를 확인할 수 없도록 처리하는 방식을 주로 사용하고 악의적인 행위를 하는 부분을 압축하고, 실행 파일이 동작할 때 압축을 풀어 메모리에 적재하는 방식인 패킹(Packing)과 코드의 변수, 클래스, 문자열 등을 해석하기 어렵게 변경하거나 실행되지 않는 임의의 코드를 삽입하는 등 가독성을 낮추는 방식인 난독화(Obfuscator)를 주로 사용한다[1]. 악성코드를 실행하지 않고 코드와 구조만으로 분석하는 정적 분석으로는 난독화 및 패킹된 악성코드를 판별할 수 없다. 따라서 코드를 직접 실행 분석하는 동적 분석을 통해 악의적인 행위를 판



(그림 1) 제안 기법의 전체 구조도



(그림 2) 정적 분석을 위한 데이터 전처리 과정

단해야 한다. 본 논문에서는 코드를 직접 실행한 동적 분석데이터뿐만 아니라 난독화, 패킹된 비정상적인 섹션들의 정보를 추출한 정적 분석데이터를 이용해 동적-정적 분석 데이터와 딥러닝 모델을 통한 악성코드를 탐지하는 기법을 제안한다.

2. 관련 연구

[2]의 연구에서 각 API Call이 발생하는 시간 차이를 전체 API Call 횟수를 기준으로 3단계로 구분하여 특징값을 추출했다. 특징값으로는 시간 차이, 평균 소요 시간, Time Interval의 급증과 급감 그래프의 기울기 등이 있고 탐지 기법으로 랜덤 포레스트(Random Forest)를 이용했다.

[3]의 연구에서 특정 API의 호출 빈도를 측정하고 계층적 클러스터링을 이용해 API마다 임곗값을 넘는 샘플들을 같은 그룹으로 묶는 방식으로 특정 악성코드 패밀리가 주로 호출하는 API의 비율을 식별한다. 특정 패밀리에 속하지 않은 샘플들은 정상 파일로 식별하는 방법을 통해 탐지를 시행했다.

3. 제안 기법

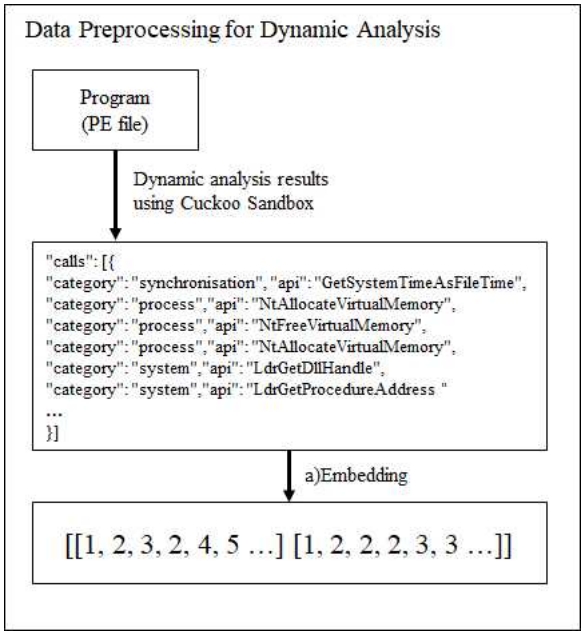
본 장에서는 제안 기법을 소개한다. 그림 1은 제안 기법의 전체 구조도이다. PE 포맷 프로그램을 전처리(Preprocessing)단계에서 동적 분석(Dynamic Analysis)과 정적 분석(Static Analysis)으로부터 데이터 추출, 정제 후 탐지(Detecting)단계에서 CNN과 LSTM 앙상블 모델을 통해 악성, 정상 프로그램으로 분류한다.

3.1 정적 분석 데이터 전처리

정적 분석은 프로그램을 실행하지 않고 대상의 기능을 분석하는 것이다. 그림 2는 본 논문에서 정적 분석을 위한 데이터 전처리 과정을 보여주는 그림이다. 입력 데이터는 (그림2, a) 각 섹션의 바이너리 정보를 2 Bytes 단위로 추출해 좌표로 이용하여 256x256 크기의 고정된 배열에 1씩 더하는 방식으로 생성한다. 전체 바이너리 정보를 하나의 배열에 입력하면 크기가 큰 프로그램의 경우 바이너리 정보가 하나의 배열에 밀집해 인공지능을 통한 분석이 어려워 (그림2, a) 방법으로 전처리한다. 데이터는 (256 x 256 x 섹션 수)의 형태로 생성하고 제안하는 모델의 CNN(Convolutional Neural Network)[4] Layer Block의 입력이 된다. CNN의 특성상 고정된 크기의 입력층을 가진다. PE 파일의 섹션 수는 프로그램마다 다르기에 입력의 크기를 고정할 필요가 있고 분류 알고리즘에 주로 사용되는 (그림2, b) LDA(Linear Discriminant Analysis)[5]를 이용하여 고정된 크기의 데이터로 변환한다.

3.2 동적 분석 데이터 전처리

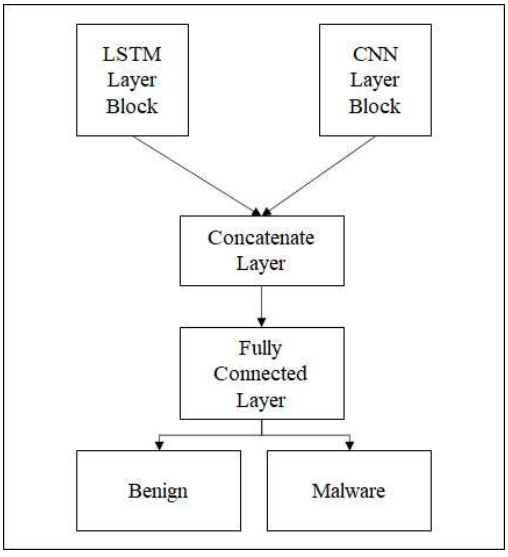
동적 분석은 대상 프로그램을 직접 실행하여 대상의 기능을 분석하는 것이다. 악성코드는 직접 실행할 때 시스템에 악영향을 주기에 독립된 환경에서 실행하여 분석해야 한다. 그림 3은 본 논문에서 동적 분석을 위한 데이터 전처리 과정을 보여주는 그림이다. Cuckoo 샌드박스[6]를 통해 동적 분석한 결과(api, category)를 (그림3, a) 임베딩하여 숫자 형태의 벡터로 변환한다. [[API Sequences][Categories of each API sequence]] 형태의 자연어로 이루어진 데이터는 임베딩 과정을 통해 [[1, 2, 3, 2, 4, 5 …] [1, 2, 2, 2, 3, 3 …]] 와 같이 변환 후 LSTM(Long Short-Term Memory)[7] Layer Block의 입력이 된다.



(그림 3) 동적 분석을 위한 데이터 전처리 과정

3.3 앙상블

3.1절과 3.2절 과정으로 얻은 결과를 통해 정적 분석데이터는 CNN Layer Block, 동적 분석데이터는 LSTM Layer Block으로 탐지하는 다중모델을 구성하였다. 그림4는 두 모델을 앙상블 하는 방법에 대한 그림이다. 두 모델의 출력 가중치를 Concatenate Layer를 통해 하나로 통합하고 Fully Connected Layer를 통해 악성코드 여부에 관한 결과를 출력한다.



(그림 4) 악성코드 탐지를 위한 모델 구조

4. 결론

본 논문에서는 PE 포맷 프로그램의 각 섹션에서 추출한 바이너리 정보를 배열에 입력하는 방식과 API Call Sequence와 API의 행위 카테고리를 임베딩하는 방식의 데이터 전처리하는 과정 및 두 데이터를 위한 앙상블 모델을 제안하였다. 제안 기법을 통해 난독화된 악성코드를 효과적으로 탐지할 수 있을 것이다.

향후 연구에서는 기존 연구들과의 실험을 통한 비교 분석하여 제안 기법의 실효성을 보이고자 한다.

5. Acknowledge

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.2018R1A2B6009620), 교신저자 김미희.

참고문헌

[1] "난독화와 코드 가상화 - 악성코드에 사용되는 바이너리 보호기법", igloo, Feb. 13, 2019 [Internet], https://www.igloo.co.kr/security-information/난독화와-코드-가상화 - 악성코드에-사용되는-바이너/

[2] 조영민, 권헌영, “API Call Time Interval을 활용한 머신러닝 기반의 악성코드 탐지”, 정보보호학회 논문지, vol.30, no.1, pp.51-58, 2020

[3] 조우진, 김형식, “API 호출 빈도를 이용한 악성코드 패밀리 탐지 및 분류 방법”, 정보보호학회논문지, vol.31, no.4, pp.605-616, 2021

[4] K. O'Shea and R. Nash, “An introduction to convolutional neural networks,” arXiv preprint arXiv:1511.08458, 2015.

[5] Gaber, T, Tharwat, A, Ibrahim, A and Hassanien, AE, “Linear discriminant analysis : a detailed tutorial”, AI communications, vol.30, no.2, pp.169-190. 2017

[6] Cuckoo Sandbox tool, [online] https://www.cuckoosandbox.org

[7] S. Hochreiter and J. Schmidhuber, “LONG SHORT-TERM MEMORY,” Neural Computation, Vol.9, No.8, pp.1735-1780, 1997.