사용자 정의 함수를 이용한 BERT 와 LSTM 기반 랜섬웨어 패밀리 분류 방법 연구

김진하 [1], 최두섭 [1], 임을규 [1]
[1] 한양대학교 컴퓨터 · 소프트웨어학과

bradypus404@hanyang.ac.kr, dslab0915@hanyang.ac.kr, imeg@hanyang.ac.kr

A Study on BERT and LSTM-based Ransomware family classification methods using User-defined functions

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

요 약

최근 악성코드 제작 기술의 고도화에 따라 악성코드의 변종이 전세계적으로 급격히 증가하고 있다. 이러한 대량의 악성코드를 신속하고 정확하게 탐지하기 위한 새로운 악성코드 탐지 기술에 관한 연구가 절실히 필요하다. 본 연구는 기존의 정적 분석과 동적 분석 방법의 한계를 극복하기 위한 방법을 제안한다. 신속한 데이터 수집을 위하여 정적 분석을 이용하여 사용자 정의 함수의 어셈블리어 데이터를 수집하고 BERT 로 임베딩하고 LSTM 으로 악성코드를 분류하는 모델을 제안한다. 분류 데이터는 행위가 정확한 랜섬웨어를 사용하였고 총 세 종류의 랜섬웨어를 분류하였고 다중 분류의 결과로 85.5%의 분류 정확도를 달성하였다.

1. 서론

인터넷 환경의 지속적인 발전은 사이버 보안에 대한 심각한 도전을 던지고 있다. 특히 악성코드의 증가로 인해 컴퓨터 시스템과 사용자의 개인정보가 위협받고 있다. 최근 3 년간 새로운 악성코드는 2021 년도에는 15,000 만 개, 2022 년도는 9,698 만 개, 2023 년도는 10,478 만 개로 매년 약 10,000 만 개 가량의 악성코드가 생성된다[1]. 하지만 기존의 악성코드 분류 기술은 주로 시그니처 기반 정적 분석이나 동적 분석을 기반으로 하고 있다. 시그니처 기반 정적 분석 기법은 단순히 악성코드의 메타데이터를 이용하여 악성코드의 hash, PE Information, Resource 데이터만 사용하여 탐지하기 때문에 동작 흐름을 알지 못한다[2]. 동적 분석 기법은 동작 흐름은 알 수 있어 정확한 탐지는 가능하지만 알려지지 않은 악성코드를 실행해야 한다는 문제점이 있다. 이러한 문제점들로 인하여 새로운 형태의 악성코드나 변형된 악성코드에 대응하기 어렵고 즉각적이 대응이 어렵다는 한계가 있다[3][4].

본 연구에서는 이러한 한계를 극복하기 위해 악성코드의 어셈블리어 데이터를 활용하여 실행하지 않고도 동작 패턴 데이터를 수집하고 분류하는 방법을 제안한다. 이를 위해 어셈블리어에서 시스템 함수는 이름만 추출하고, 사용자 정의 함수의 어셈블리어만을 데이터로 사용하여 데이터의 양을 대폭 줄여 빠른 시간에 데이터를 수집한다. 그 후, Bidirectional Encoder Representations from Transformers(BERT)를 이용하여 데이터를 임베딩하고, Long Short-Term Memory(LSTM)를 활용하여 유사한 동작 패턴을 이용해 악성코드를 분류한다. 이를 통해 새로운 형태의 악성코드에 빠르고 정확히 대응할 수 있는 효과적인 악성코드 분류 모델을 제안한다.

2. Background

2.1. BERT

BERT는 구글에서 개발한 사전 훈련된 자연어 처리 모델이고 양방향 Transformer를 이용하여 단어의 의미를 벡터화 하는 모델이다[5]. 문장 내의 각 단어를 인코딩할 때 해당 단어의 좌우 문맥을 모두 고려함으로써 단어의 의미를 더욱 정확하게 파악할 수 있다. 이러한 특성 덕분에 BERT는 기계 번역, 질의 응답 시스템, 문서 분류 등 다양한 자연어 처리 작업에서 뛰어난 성능을 보인다. 그러므로 BERT는 문맥을 고려한 효과적인 단어 임베딩 기법이다. 본 연구에서는 여러

단어가 합쳐져 있는 시스템 함수의 의미를 벡터화 하기 위해 사용한다.

2.2. RNN

Recurrent Neural Network(RNN)은 순차적인 데이터 모델링 작업을 위해 설계된 인공 신경망의 한 유형이다[6]. 본 연구에서 사용하는 LSTM 은 RNN 의 특수한 변형 중 하나로, 장기 의존성 문제를 해결하기 위해 설계되었다. LSTM 은 입력 게이트, 삭제 게이트, 출력 게이트와 같은 게이트 메커니즘을 도입하여 네트워크 내에서 정보의 흐름을 조절한다. 이를 통해 LSTM 은 중요한 정보를 오랜 시간 동안 유지하고 훈련 중 그라디언트 소실 문제를 완화한다. 이러한 특성으로 LSTM 은 장거리 의존성을 모델링하고 훈련 중 그라디언트 소실과 관련된 문제를 완화하는 데 효과적이며, 이로 인해 언어 생성, 감정 분석, 기계 번역 등의 다양한 자연어 처리 응용 프로그램에서 널리 사용되고 있다. 본 논문에서도 순환 아키텍처를 사용하여 방대한 길이의 어셈블리어를 훈련하고 훈련 중 발생하는 그라디언트 소실 문제를 완화한다.

3. 관련 연구

Ömer Aslan 외 1 인은 Malimg[7], Microsoft BIG 2015[8] 및 Malevis[9]의 데이터셋을 사용했고 VGG16, ResNet50 으로 하이브리드 접근 방식을 사용하여 악성코드를 분류하는 연구를 진행하였다. 실험결과는 97.7%의 정확도와 98.7%의 고정밀도로 악성코드를 효과적으로 분류할 수 있음을 확인했다. 그러나 일부 악성코드 샘플은 코드 난독화 기술로 인해 제대로 분류되지 못한 한계점이 드러났다.[10]

Barath Narayanan Narayanan 외 1 인은 악성코드를 패턴에 기반하여 분류하는 다양한 방법을 실험하며 CNN 과 LSTM 을 특징 추출하는 기법을 소개하며 컴파일 파일과 어셈블리 파일의 특징을 결합하는 앙상블 방법도 제안한다. 결과는 CNN 과 RNN 에서 추출된 특징을 결합한 SVM 이 가장 우수한 결과를 제공한다. 데이터 불균형 또한 해결할 수 있다고 한다.[11]

Sana Aurangzeb 외 3 인은 정적 및 동적 특성을 기반으로 하는 랜섬웨어 탐지, 분류 도구인 big-data based ransomware classification using ensemble machine learning(BigRC-EML)을 제안한다. 동적 데이터와 하이브리드로 두 가지 유형의 데이터셋을 사용하였다. 앙상블 머신 러닝인 BicRC-EML 을 이용하여 실험한 결과 모든 유형의 데이터에서 98%의 정확도를 달성하여 랜섬웨어 탐지에 효과적임을 입증하였다.[12]

4. 본론

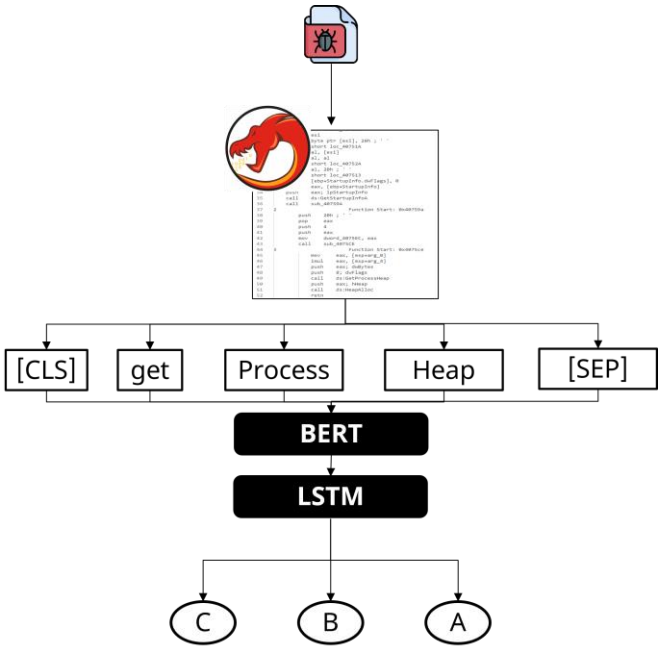
4.1. 데이터 수집

데이터는 VirusShare[13]에서 1,000 개의 랜섬웨어 샘플을 수집하고 그 중 Detect It Easy(DIE)[14]를 이용해 악성코드 패킹 유무를 탐지하여 패킹이 되지 않은 순수한 악성코드를 수집했고 그 중 3 종류의 랜섬웨어 샘플 총 150 개를 수집했다.

4.2. User-Defined Function

User-Defined Function(UDF)은 사용자가 직접 정의하여 만든 함수를 의미한다. UDF 가 필요한 이유는 시스템 함수까지 모든 어셈블리어를 추출하면 개당 평균 10MB 정도의 방대한 양의 어셈블리어가 추출된다. 많은 양의 데이터이지만 해당 데이터는 악성코드의 행위를 담은 것이 아닌 시스템 함수 내부의 어셈블리어가 90%이상을 차지한다. 위와 같이 된다면 인공지능을 학습할 때 불필요한 오버헤드가 생기고 정작 중요한 사용자가 직접 제작한 함수들의 의미가 모호해진다. 그러므로 사용자 정의 함수만 추출하고 시스템 함수는 이름만 가져와서 이름의 의미를 분석하는 방법을 선정하였다.

4.3. 악성코드 분류 구성도



(그림 1) BERT, LSTM 기반 악성코드 분류 구성도

본 연구의 구성도는 그림 1 과 같다. 첫번째로 악성코드의 어셈블리어 정보를 정적으로 추출하기 위해 Ghidra[15] Python Library 를 이용한다. Ghidra 를 이용해 UDF 를 추출하고 시스템 함수는 이름명만 저장하는 자동화된 시스템을 사용한다. 두번째는 첫번째 방식으로 추출된 어셈블리 리스트에서 opcode 와 시스템 함수의 이름을 각각 추출하는 작업을 진행한다. 세번째로는 자연어로 이루어진 opcode 와 시스템 함수를 BERT 를 이용해 임베딩하여 자연어를 벡터화한다. 마지막으로는 순차적인 어셈블리어를 탐지하기 위하여 LSTM 을 이용해서 악성코드를 분류하는 모델을 제작한다.

4.4. 특징 추출

악성코드 샘플에서 데이터를 추출하기 위해 Ghidra 를 사용하여 정적 분석으로 어셈블리어를 추출한다. 분석환경에 악성코드를 삽입 후 Python Script 의 동작은 다음과 같다.

① 정적 분석으로 진행
② “start” or “main” 함수부터 추출을 진행
③ .text 섹션을 추출하고 섹션 주소에 해당하는 함수만 추출하고 나머지 함수는 이름만 저장
④ 2)번 함수의 return 이 나올 때까지 반복

위 조건들을 만족하면서 어셈블리 리스트를 저장한다. 저장된 리스트에서 opcode 와 시스템 함수의 이름을 분리하여 임베딩을 진행하기 위한 준비를 한다.

4.5. 데이터 임베딩

RNN 을 이용하여 자연어로 구성된 악성코드를 분류하기 위해서는 자연어를 벡터 값으로 임베딩이 필요하다. 본 논문에서는 시스템 함수의 이름은, “ GetModuleHandleA”, “ GetProcAddress ”, “lpModuleName”와 같이 여러 단어를 합쳐 놓은 이름이므로 BERT 를 사용하여 단어의 의미를 고려하여 벡터화한다. 중복이 없는 opcode 리스트와 시스템 함수 이름 리스트를 준비한다. BERT 모델은 사전 학습되어 있는 bert-base-cased 를 사용하고 기존에 준비해둔 리스트를 이용해 학습을 진행한다. 학습된 모델을 이용해 학습에 진행된 opcode, 시스템 함수 이름이면 바로 임베딩하여 벡터화 하고 존재하지 않는 데이터라면 예측을 진행하여 임베딩을 진행한다.

4.6. 악성코드 분류 모델

악성코드 분류 모델은 임베딩된 어셈블리 코드와 시스템 함수 이름의 벡터를 입력으로 사용하여 악성코드를 분류하는 역할을 한다. 순차적 어셈블리 데이터는 대부분 매우 길기 때문에 긴 리스트를 잘 이해하는 LSTM 을 사용한다. 입력은 opcode 와 시스템 함수 이름의 벡터로 받는다. 모델은 Sequntial 로 구축했고 임베딩층, LSTM 층, Dense 층으로 이루어진다. LSTM 층은 시퀀스 데이터의 특징을 추출한 후 Dense 를 이용해 랜섬웨어 다중 분류를 수행한다. API 와 opcode 를 분리해서 모델에 각각 학습시킨 후 가중치 평균 기법을 이용해 랜섬웨어 총 3 개로 분류하여 결과를 도출한다.

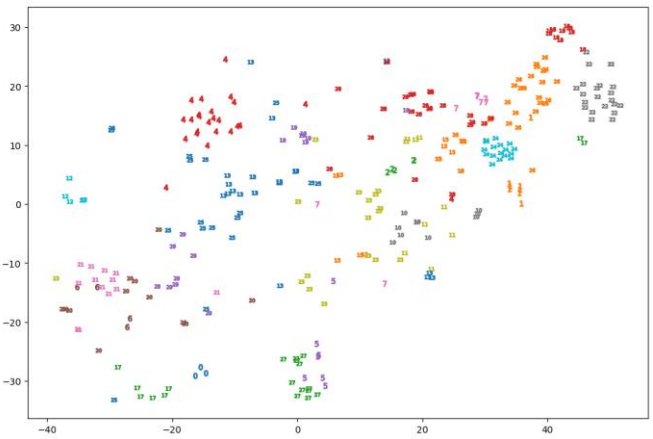
5. 실험 결과

5.1. 실험 환경

데이터 추출 환경은 VMware Workstation 17 pro 가상머신으로 Windows 10 Education 운영체제를 사용한다. Ghidra 는 Version 11.0.3 을 사용하고 Python 은 3.11.03 버전을 사용한다.

5.2. 함수 임베딩

그림 2 는 BERT 를 활용하여 시스템 함수를 분류한 후 클러스터로 선정한 결과를 시각화한 것이고 서로 다른 행위를 의미한다.



(그림 2) BERT Cluster Graph

표 1 로 정리된 내용을 통해 각 클러스터가 어떤 행위를 하는지를 쉽게 파악할 수 있다. 예를 들어, 클러스터 0 은 파일과 관련된 API 인 GetDlgItem 이 있다. 이와 같이 각 클러스터는 비슷한 기능을 하는 API 들을 포함하고 있다. 이러한 결과는 시스템 함수들이 의도한 대로 임베딩되어 클러스터링되었음을 의미한다. 이를 통해 시스템 함수들의 의미를 벡터화하고 유사한 행위를 하는 함수들을 효과적으로 그룹화할 수 있음을 확인할 수 있다.

<표 1> 대표 행위별 API

no	행위 내용	대표 API
0	파일과 관련된 조작	GetDlgItem
1	색상 조정, 키보드	HTUI_ColorAdjustment
2	스레드 관리	CreateThread
3	시스템 정보 및 성능	QueryPerformanceCounter
4	파일 시스템 및 MSI	SetFileAttributesW
5	리소스 관련	FindResourceA
6	스크롤바 관련	FlatSB_EnableScrollBar
7	유틸리티 기능	TlsGetValue
8	드라이버 관리	GetSystemInfo
9	윈도우 클래스 관리	RegisterClassExA
10	프로세스 관리	VirtualProtect
11	SNMP 관련 유틸리티	SnmpUtilUTF8ToUnicode
12	리소스 및 시스템 설정	SystemParametersInfoW
13	윈도우 관련 작업	DefWindowProcA
14	파일 I/O 및 메시지	CreateFileA
15	메뉴 관리 및 파일 조작	AppendMenuW
16	문자열 처리 관련	StrStrA
17	메모리 관리, 예외 처리	_set_new_handler
18	현재 환경 정보 관리	GetCurrentThreadId
19	문자열, 레지스트리	CharNextA

5.3. 분류 결과

본 연구에서 제안한 악성코드 분류 모델의 성능을 평가한 결과, api 기반 LSTM 모델은 81%의 정확도를 도출했고 opcode 기반 LSTM 모델은 78%의 정확도를 도출했다. 이 두 모델 중 api 의 결과가 더 높게 나왔으므로 0.7 의 가중치를 주어 분류를 진행했다. 그 결과로 분류 정확도는 85.5%로 측정되었다. 이 결과는 제안된 모델이 효과적으로 악성코드를 분류했다는 것을 보여준다.

6. 결론 및 향후 연구

본 연구에서는 랜섬웨어 분류를 위한 새로운 방법인 UDF 기반의 랜섬웨어 분류 기법을 제안하였다. 제안한 방법은 사용자 정의 함수를 중심으로 한 어셈블리어 정보를 추출하고, BERT 를 이용한 시스템 함수 이름 임베딩과 LSTM 을 활용한 랜섬웨어 분류 모델을 구축했다. 실험 결과, 제안된 모델은 데이터의 양이 적은 것에 비해 높은 정확도를 보여주었다. 이를 통해 데이터 양의 한계를 극복하면서도 정적분석을 통한 방법이 효과적으로 적용될 수 있음을 확인할 수 있었다.

향후 연구에서는 클러스터링을 세부적으로 진행하여 시스템 함수의 특징을 더 명확하게 할 예정이다. 또한 LSTM 을 이용한 모델을 또 다른 앙상블 기법을 사용하여 분류하는 모델을 생성하는 연구를 진행할 예정이다. 마지막으로 랜섬웨어에만 국한되지 않고 Trojan, Worm, Backdoor, Suspicious 와 같은 악성코드 전반에 대해서 분류 실험을 진행할 예정이다.

ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2022R1A4A1032361)

참고문헌

[1] “Total Amount of malware and pua: New Malware”, AV-TEST, 2024, https://portal.av-atlas.org/malware

[2] “Malware feature information for utilizing artificial intelligence technology”, KISA, 2021.06.07, https://www.boho.or.kr/kr/bbs/view.do?bbsId=B0000127&nttId=36076&menuNo=205021

[3] M. Sikorski and A. Honig, Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software, San Francisco, CA, USA: No starch press, 2012.

[4] S. K. Pandey and B. M. Mehtre, "Performance of malware detection tools: A comparison", Proc. IEEE Int. Conf. Adv. Commun. Control Comput. Technol., pp. 1811-1817, May 2014.

[5] DEVLIN, Jacob, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[6] HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. Neural computation, 1997, 9.8: 1735-1780

[7] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath. “Malware images: visualization and automatic classification”. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, 2011, Pages 1–7

[8] MaleVis: A Dataset for Vision Based Malware Recognition, hacettepe, 2019, https://web.cs.hacettepe.edu.tr/selman/malevis/index.html

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

[10] Ö. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning Algorithms," in IEEE Access, vol. 9, pp. 87936-87951, 2021.

[11] Narayanan, B.N.; Davuluru, V.S.P. Ensemble Malware Classification System Using Deep Neural Networks, Electronics 2020, 9(5):721, April 2020

[12] Aurangzeb, S., Anwar, H., Naeem, M.A. et al. BigRC-EML: big-data based ransomware classification using ensemble machine learning. Cluster Comput 25, 3405–3422, 2022.

[13] online malware repository that produces active malware samples to security researchers, virusshare, https://virusshare.com/

[14] Detect-It-Easy, horsicq, 2024.04.22, https://github.com/horsicq/Detect-It-Easy

[15] GHIDRA, National Security Agency, https://ghidra-sre.org/