AutoEncoder 기반 역난독화 사전학습 및 전이학습을 통한 악성코드 탐지 방법론

장재석 [1], 구본재 [1], 엄성준 [2], 한지형 [1*]
[1] 서울과학기술대학교 컴퓨터공학과
[2] 서울시립대학교 전자전기컴퓨터공학부
jangjs1027@seoultech.ac.kr, rnqhswo332@seoultech.ac.kr, junjun607@uos.ac.kr, jhhan@seoultech.ac.kr

Malware detection methodology through on pre-training and transfer learning for AutoEncoder based deobfuscation

Jae-Seok Jang[1], Bon-Jae Ku[1], Sung-Jun Eom[2], Ji-Hyeong Han[1*]
[1]Dept. of Computer Science and Engineering, Seoul National University of Science and Technology
[2]School of Electrical and Computer Engineering, University of Seoul

요 약

악성코드를 분석하는 기존 기법인 정적분석은 빠르고 효율적으로 악성코드를 탐지할 수 있지만 난독화된 파일에 취약한 반면,, 동적분석은 난독화된 파일에 적합하지만 느리고 비용이 많이 든다는 단점을 가진다. 본 연구에서는 두 분석 기법의 단점을 해결하기 위해 딥러닝 모델을 활용한 난독화에 강한 정적분석 모델을 제안하였다. 본 연구에서 제안한 방법은 원본 코드 및 난독화된 파일을 grayscale 이미지로 변환하여 데이터셋을 구축하고 AutoEncoder 를 사전학습시켜 encoder 가 원본 파일과 난독화된 파일로부터 원본 파일의 특징을 추출할 수 있도록 한 이후, encoder 의 output 을 fully connected layer 의 입력으로 넣고 전이학습시켜 악성코드를 탐지하도록 하였다. 본 연구에서는 제안한 방법론은 난독화된 파일에서 악성코드를 탐지하는 성능을 F1 score 기준 14.17% 포인트 향상시켰고, 난독화된 파일과 원본 파일을 전체를 합친 데이터셋에서도 악성코드 탐지 성능을 F1 score 기준 7.22% 포인트 향상시켰다.

Keywords: 악성코드, 정적분석, 역난독화, AutoEncoder, 사전학습, 전이학습, grayscale

1. Introduction

악성코드는 하나의 컴퓨터, 서버 또는 컴퓨터 네트워크에 피해를 입히도록 설계된 모든 소프트웨어 [1]를 의미한다. 악성코드는 목표를 감염시키기 위해 정상적인 소프트웨어인 척 위장을 하기 때문에, 이를 탐지하고 방지하는 노력이 필요하다.

악성코드를 탐지하는 방식으로는 크게 정적분석과 동적분석이 있다. 정적분석은 관측된 악성코드의 특징을 바탕으로 프로그램이 악성코드인지를 분석하는 기법이고, 동적분석은 프로그램을 가상 운영체제에서 프로그램을 실행한 후 프로그램이 동작하면서 발생되는 정보를 바탕으로 악성코드를 분석하는 기법이다. 정적분석은 프로그램을 실행하지 않고, 얻을 수 있는 정보를 바탕으로 악성코드를 탐지하는 기법이기 때문에, 빠르고 가상 운영체제를 필요로 하지 않아 비용이 적은 기법이지만, 기존에 탐지된 악성코드를 바탕으로 새로운 코드를 탐지하는 것이기 때문에 난독화와 같이 변조된 코드를 제대로 분석하지 못한다는 단점을 가진다. 반면 동적분석은 가상 운영체제 공간에서 프로그램을 실행하고 실행된 정보들을 바탕으로 악성코드를 탐지하기 때문에, 난독화된 파일도 문제없이 분석할 수 있다는 장점을 가진다. 하지만 가상 운영체제에서 프로그램을 실행시키고 정보를 받는 과정 속에서 탐지 속도가 느리고 비용이 많이 필요하다는 단점을 가진다. 이런 점 때문에 기고문 '효율적인 보안시스템 구축을 위한 분석기술의 통합'[2]에서는 정적분석과 동적분석의 장점을 합친 새로운 방식이 필요하다고 하였다.

본 연구에서는 난독화된 파일도 분석이 가능한 동적분석과 빠르고 적은 비용으로 악성코드 분석이 가능한 정적분석의 장점을 모두 갖춘 딥러닝 기반 악성코드 탐지 모델을 제작하기 위해 난독화된 파일로부

*corresponding author: Ji-Hyeong Han

[1] https://www.itworld.co.kr/news/110408
[2] https://www.comworld.co.kr/news/articleView.html?idxno=49618

터 원본파일의 특징을 추출하는 AutoEncoder 를 사용하였다. 실행 파일을 딥러닝 모델에 입력하기 위해 파일을 grayscale 이미지로 변환 후 딥러닝 모델로 탐지하는 기존 연구[1]의 방법론을 활용하였다. 본 연구에서는 AutoEncoder 모델로 난독화된 파일로부터 원본 파일의 특징을 추출하도록 사전학습을 시킨 후 전이학습을 하는 방법론을 최초로 제안하였고, F1 score 을 기준으로 해당 방법론을 사용한 모델이 사용하지 않은 모델에 비해 성능이 7.22%포인트 향상되었음을 확인하였고, 특히 난독화된 파일을 탐지하는 성능은 14.17%포인트 향상되었음을 확인하였다. 연구에서 사용한 실험 코드는 다음 링크[3]에서 확인할 수 있다.

2. Related Work

딥러닝을 기반으로 하는 악성코드 정적분석기법은 크게 pefile 의 opcode 를 추출하여 opcode 정보를 자연어 처리 모델을 바탕으로 학습하여 탐지하는 방법[2]과 파일의 바이너리 정보를 8bit 를 수치로 변경하여 정사각형의 grayscale 이미지로 변환한 후 탐지하는 방법[1]이 있다. Opcode 를 기반으로 탐지하는 방법론은 자연어 처리 모델을 사용하고, grayscale 이미지로 변환하여 탐지하는 방법론은 이미지 분류 모델을 사용한다.

Santos et al. [2]은 기존의 악성코드 탐지 방식은 새로운 악성코드를 탐지하지 못하거나 감염된 이후에 탐지를 할 수 있다는 단점이 있어, 이를 해결하기 위해 opcode sequence 의 빈도수를 기반으로 악성코드를 탐지하는 방법론을 제안하였다. 저자들은 제안한 방법이 새로운 악성코드들에 대해 높은 탐지율을 보였으나, 제안한 방식이 정적분석이기 때문에 난독화에 취약하다고 하였다.

Kalash et al. [1]은 grayscale 이미지를 기반으로 악성코드를 탐지하는 방법론을 제안하였다. 저자들은 파일의 바이너리 정보를 8bit 씩 받아 grayscale 이미지로 변환시키는 방법론을 제안했으며, 악성코드 탐지를 위한 CNN 모델을 학습 후 성능을 측정한 결과, 기존 방식들에 비해 더 높은 악성코드 탐지 성능을 보였다.

Opcode sequence 빈도수를 기반으로 탐지하는 방법론의 경우에는 빈도수를 기반으로 얻은 정보를 바탕으로 탐지를 하므로 난독화를 통해 의미 없는 opcode sequence 를 등장시킬 경우, AutoEncoder 을 통해 역난독화 학습을 하기 어려울 것으로 보았다.

반면 grayscale 이미지를 기반으로 탐지하는 방법론의 경우, 입력 이미지의 크기가 고정되므로 정보의 손실이 발생될 수도 있으나 코드 파일의 처음부터 끝까지의 정보를 한 번에 담은 정보를 입력 받을 수 있다는 장점을 가졌다. 이런 점들을 반영하여 본 연구에서는 grayscale 이미지를 활용하여 악성코드를 탐지하는 방법론을 사용하였다.

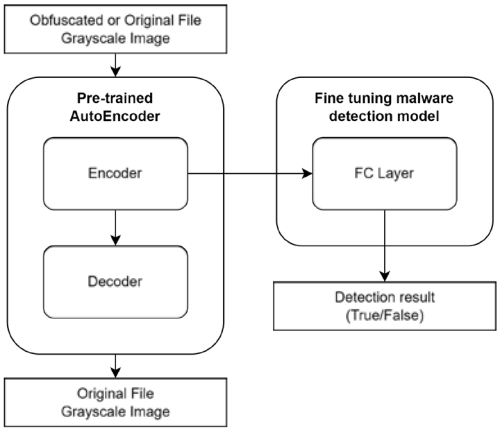
Grayscale 이미지를 이용해 AutoEncoder 모델을 학습시켜 악성코드를 탐지하는 방법론[3, 4]은 기존에도 존재하였다. 그러나 AutoEncoder 를 활용하여 악성코드를 탐지하는 방법론은 원본 파일을 입력 후 단순히 원본 파일을 추출하여 원본 파일의 특성을 조금 더 잘 이해하도록 사전학습시키는 방법론[3]과 악성코드 이미지로만 AutoEncoder 모델을 학습시킨 후 재구성 오류 점수를 얻고 Threshold 를 지정하여 탐지하는 방법론[4]만이 있었을 뿐 난독화된 파일로부터 원본 파일로 역난독화에 대한 사전학습을 이용한 방법론은 존재하지 않았으므로, 본 연구에서 제안하는 방법론과의 차이점이 있다.

3. Proposed Framework

본 논문에서 제안하는 AutoEncoder 기반 난독화된 파일에 대한 사전학습 후 전이학습을 통한 악성코드 탐지 모델의 구조는 그림 1 과 같다.

1. 먼저 AutoEncoder 모델에 난독화된 파일 또는 원본 파일을 Encoder 에 입력을 시키면 원본 파일이 Decoder 의 출력으로 나올 수 있게 사전학습을 시킨다.
2. 사전학습을 마친 AutoEncoder 의 Encoder 을 FC Layer 과 연결한 후 악성코드를 탐지하는 전이학습을 시킨다

(그림 1) 본 연구에서 제안하는 악성코드 탐지 모델 구조



난독화된 파일뿐만 아니라 원본 파일을 인코더의 입력 데이터로 사용하는 이유는 [3]에서 특정 파일로부터 해당 파일이 나오도록 AutoEncoder 모델을 학습시키면 오차역전파로 확인할 수 없는 파일의 특성을 학습하기 때문에 성능이 더 좋아지게 한다고 해서였다. 또한 AutoEncoder 가 난독화된 파일과 원본 파일로부터 원본파일의 특징을 정확히 포착하여 추출할 수 있도록 하기 위해 두 종류의 데이터를 입력으로 사용하였다.

4. Experiments

4.1 Experimental Methods and Environments

본 연구에서는 악성코드 탐지 모델을 사전학습 및 전이학습 후 성능을 측정하기 위해 Train dataset, validation dataset, test dataset 의 비율을 7:1:2 로 설정하였다. 사전학습 모델의 Epoch 는 100, 200 으로 나누어 실험을 진행하였고 validation loss 값이 가장 적은 시점의 모델을 사용하였다. 전이학습 모델은

[3] https://github.com/ZangZaeSeok/MalwareDetection

Epoch 를 30 으로 설정하였고, validation dataset 의 F1 score 가 가장 높은 시점의 모델을 사용하였다. Batch size 는 실험환경에서 최대로 사용할 수 있는 크기인 2 로 설정하였으며, loss function 은 binary cross entropy 를 사용하고 optimizer 는 Adam 을 사용하였다.

추가로 본 연구에서는 AutoEncoder 모델을 통한 사전학습의 효과를 확인하기 위해 전이학습 모델과 동일한 구조의 Default 모델을 학습시켰다. Default 모델의 hyper parameter 는전이학습 모델과 동일하다.

실험은 Intel (R) Xeon(R) CPU @ 2.30GHz (Dual-Core)와 Nvidia Tesla K80 혹은 Nvidia Tesla P100 를 갖춘 컴퓨터 환경에서 이루어졌다.

본 연구에서는 F1 score 를 평가지표로 사용하였다. F1 score 는 전체 악성코드들 중 얼마만큼을 예측하였는지를 의미하는 Recall 과 악성코드로 탐지한 샘플들 중 실제 악성코드 샘플의 비율인 Precision 의 조화 평균값이다. F1 score 는 상반된 관계의 Recall 과 Precision 을 동시에 고려하기 때문에 분류 모델의 실제 성능을 평가하는 지표로 사용된다.

4.2 Experimental Data

본 연구에서는 DikeDataset[4]에서 제공하는 악성코드 및 정상 파일들을 Dataset 으로 사용하였다. 악성코드 및 정상파일에 대응되는 난독화된 파일을 얻기 위해서 패킹 및 변조를 해주는 Themida[5]를 사용하여 파일들을 난독화하였다. 이후 대응되는 두 파일을 같은 샘플군에 속하게 데이터를 분할하였다: 원본 파일이 train dataset 에 속할 경우, 난독화된 파일도 train dataset 에 속하게 분할하였다. 이후 파일들은 바이너리 정보를 8bit 씩 묶어서 1~255 값을 가지는 리스트를 만든 후 정사각형 이미지를 만들기 위해 2d array 에 순서대로 그 값을 채운 후, 남은 영역은 0 으로 채우는 방식으로 grayscale 이미지로 변환하였다. 이미지의 크기는 파일마다 다르나, 악성코드 탐지 모델은 고정적인 크기의 이미지를 입력 받기 때문에 512*512 크기의 이미지로 변환하였다. 각 샘플 및 train, validation, test dataset 의 분포는 표 1 과 같다.

<표 1> Train, validation and test datasets.

	Train	validation	Test	Total
Malware	1,175	169	336	1,680
Benign File	638	91	183	912
Total	1,813	260	519	2,592

4.3 Experimental Results

표 2 는 Default model 과 사전학습 Epoch 이 100 과 200 인 전이학습 모델의 성능 측정 결과를 보여준다. 먼저 원본 파일에 대한 탐지 성능은 사전학습 Epoch 이 100 인 전이학습 모델이 성능이 가장 좋았다. Epoch 이 200 인 모델은 그 성능이 Default model 에 비해서 떨어졌지만, 난독화된 파일에서 악성코드를 탐지하는 성능은 14.17%포인트 더 좋았다. 또한 원본 코드와 난독화된 코드 전체에서 악성코드를 탐지하는 성능에 있어서는 98.53%로 Default model 에 비해 7.22%포인트의 성능향상이 있었다. 특히 전체 데이터에 대해서 사전학습 epoch 이 증가할수록 그 성능이 향상되는 것을 확인할 수 있었다. 이를 통해 난독화에 대한 사전학습이 악성코드 탐지에 도움을 줄 수 있다는 것을 확인할 수 있었다.

<표 2> 각 모델 F1 score 측정 결과

	Original Code	Obfuscated Code	Total
Default Model	98.27%	84.95%	91.31%
AutoEncoder based Model (100)	99.7%	92.95%	96.35%
AutoEncoder based Model (200)	97.94%	99.12%	98.53%

5. Conclusion

본 연구에서는 악성코드 정적분석과 동적분석의 장점을 결합한 방법론을 난독화에 강한 정적분석 기법을 AutoEncoder 을 통한 사전학습을 통해 구현하였다. 실험을 통해 사전학습이 난독화에 강한 모델을 만드는데 큰 도움을 준다는 것을 보였다.

추가로, 이미지 분류 등에 더 좋은 성능을 보이는 모델들을 기반으로 AutoEncoder 모델을 사용해서 연구를 추가적으로 진행해볼 계획이며, Multiple instance learning 을 통해 grayscale 이미지의 어떤 부분이 해당 파일을 악성코드로 분류하게 했는지 bounding box 를 탐지하는 연구를 진행할 계획이다.

Acknowledgement

본 프로젝트는 과학기술정보통신부 정보통신창의인재양성사업의 지원을 통해 수행한 ICT 멘토링 프로젝트 결과물입니다

참고문헌

[1] Kalash, M., Rochan, M., Mohammed, N., Bruce, N. D., Wang, Y., & Iqbal, F., Malware classification with deep convolutional neural networks., In *2018 9th IFIP international conference on new technologies, mobility and security (NTMS)* (pp. 1-5)., 2018

[2] Santos, I., Brezo, F., Ugarte-Pedrero, X., & Bringas, P. G., Opcode sequences as representation of executables for data-mining-based unknown malware detection., *Information Sciences*, *231*, (pp. 64-82)., 2013

[3] Kumar, S., Meena, S., Khosla, S., & Parihar, A. S., AE-DCNN: Autoencoder Enhanced Deep Convolutional Neural Network For Malware Classification., Convolutional Neural Network For Malware Classification., In *2021 International Conference on Intelligent Technologies (CONIT)* (pp. 1-5)., 2021

[4] Jin, X., Xing, X., Elahi, H., Wang, G., & Jiang, H., A malware detection approach using malware images and autoencoders., In *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)* (pp. 1-6)., 2020

[4] https://github.com/iosifache/DikeDataset
[5] https://www.oreans.com/Themida.php