CNS final project proposal

Graph Matching Networks for Malware detection

R07922055 李承軒 B04902103 蔡昀達 B06902006 王俊翔

- ## Problem description

Problem definition

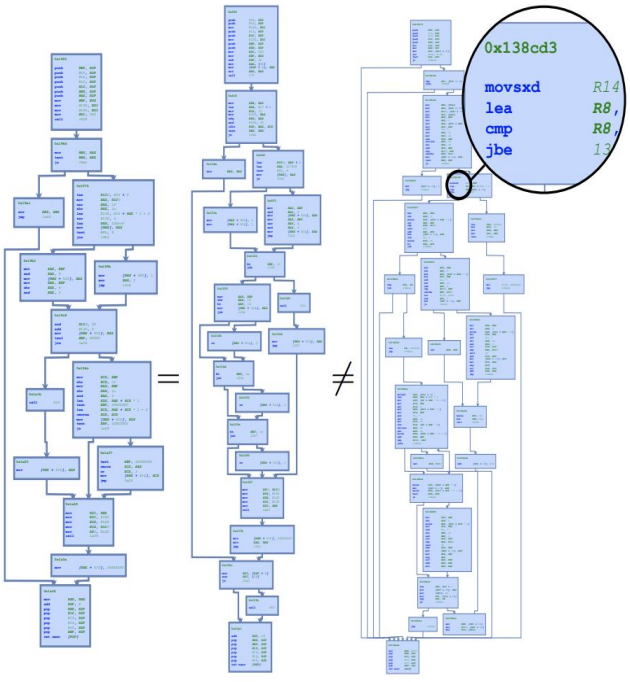
Nowadays, malware has become a big threat to computer systems. Most of the malware intrude users' systems as they download files over Internet. When people get access to those infected files, it is likely to bring up such a big harm to their operating system, network devices and so on. There are several common malware types existing through the years including Trojans, Adware and Botnet.

Therefore, malware detection has become a significant issue recently. Malware detection can be classified into two categories: anomaly-based detection and signature based detection. The former identifies what is considered normal in benign programs and distinguishes whether the testing program meets the benign one; while the latter emphasizes what is considered malicious to inspect the similarity between the feature of the testing program and the malicious program.

Problem to be solved

Our main purpose is to implement a graph matching networks for malware detection. Graph-based features such as control flow graph or function call graph are strong features in malware detection. With advanced deep graph neural network, we can make end-to-end training and automatically extract features from graph structure data and direct similarity compare.

- ## Related work



Static malware analysis mostly based on information in binary file such as API calls analysis, header information, opcode sequences and other general characteristics. In our survey, most related works extract graph based features such as betweenness centrality, closeness centrality, degree, shortest path length, density, # of edges and nodes, which would loss many informations. In contrast, we would like to make end-to-end model with automatically features extracting from graph structure data and preserve the entire graph information including informations such as API calls and opcode sequences and thus will outperform the each of them respectively. In addition, we would like to examine how the graph matching networks for learning the similarity of graph structured objects can be used in detecting malware.

Nguyen, M. H., Le Nguyen, D., Nguyen, X. M., & Quan, T. T. (2018). Auto-detection of sophisticated malware using lazy-binding control flow graph and deep learning. Computers & Security, 76, 128-155.

Alasmary, H., Khormali, A., Anwar, A., Park, J., Choi, J., Nyang, D., & Mohaisen, A. (2019). Analyzing, Comparing, and Detecting Emerging Malware: A Graph-based Approach. arXiv preprint arXiv:1902.03955.

Li, Y., Gu, C., Dullien, T., Vinyals, O., & Kohli, P. (2019). Graph Matching Networks for Learning the Similarity of Graph Structured Objects. arXiv preprint arXiv:1904.12787.

- Plan

Baseline

Methods	**classification algorithm**	**Features**
LSH	KNN	whole file
Malconv	CNN	byte sequence
PE-Miner	decision tree	PE file Structural features
EMBER	gradient boosting tree	General file information,Header information,API calls,Section information,Byte histogram,String information

Our Approach

For each program, we will transfer its binary file to control flow graph to derive its graph structure. A graph neural networks will be trained based on these graph structures to classify whether a program is malicious or benign. With input as graph structures instead of features extracted from graph structural data, our model can easily preserve the graph information and use it to help us classify malware.

Evaluation

We will collect malware from Virushare and more samples from VirusTotal where these samples are scanned by multiple virus scanner. A threshold will be decided to determine whether the sample should be labeled as malicious or benign. For these collected samples, we will split 80 percent of the data for training our model and 20 percent of the data for testing our model performance. We will examine the accuracy and the fp rate for each baseline.

- Timeline

05/07 ~ 05/13 Survey
05/14 ~ 05/20 Collect dataset and build baseline model
05/21 ~ 05/27 Implement our method and Test
05/28 ~ 06/03 Build demo website
06/04 ~ 06/11 Buffer

- Deliverables

A web-based malware detection platform prototype