Malware identification on Portable Executable files using Opcodes Sequence

Alexndre R de Mello[1], Vitor G Lemos[2], Flávio G O Barbosa[3], Emílio Simoni[4]

[1]CERTI Foundation; [2]PSafe Cyberlabs; [3]SENAI Institute of Innovation in Embedded Systems; [4]AHT Security

1. Introduction

This paper assesses a method that employs opcode sequence analysis, Graph Theory, and Machine Learning to identify malware on Portable Execution files without the need for execution. We introduce the OSG (Opcode Sequence Graph), a concept for malware detection on Portable Executable (PE) files based on Opcode Sequence, Graph Theory, and Artificial Intelligence with two new methods: **OSGT (Opcode Sequence Graph Theory) detector** and **OSGNN (Opcode Sequence Graph Neural Network)**.

2. OSG

The OSG was designed considering the detection time, feature size, and capacity to detect malware. To implement the methods using the OSG methodology, the following steps are necessary:

- Extract the Opcode Sequence
- Create the Opcode Graphs
- Create features from the Opcode Graphs
- Train a machine learning classifier

2.1 OSGT

The OSGT follows the OSG methodology by creating a graph for each section of the file that contains instructions, create features from each graph, and train tree-based models to classify the files as malware or benign. The OSGT workflow is depicted by Figure 1.

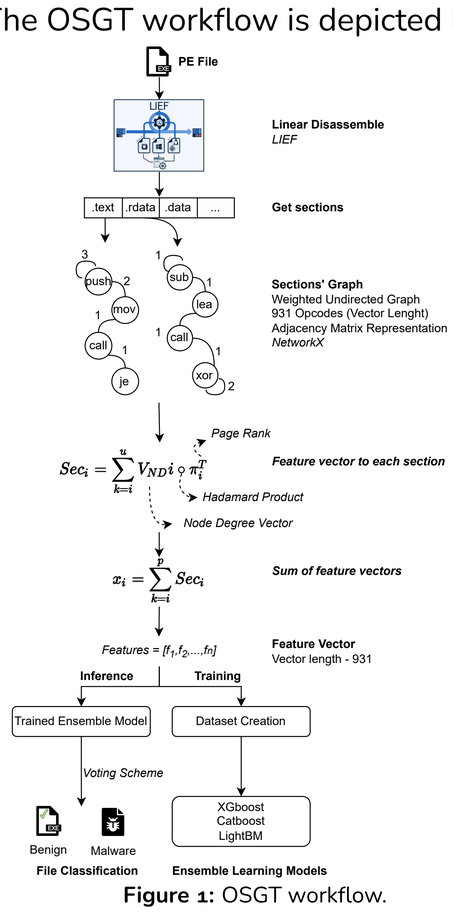

Figure 1: OSGT workflow.

The detailed process to create a graph from a file's section is depicted by Figure 2, and it is used by both OSGT and OSGNN.

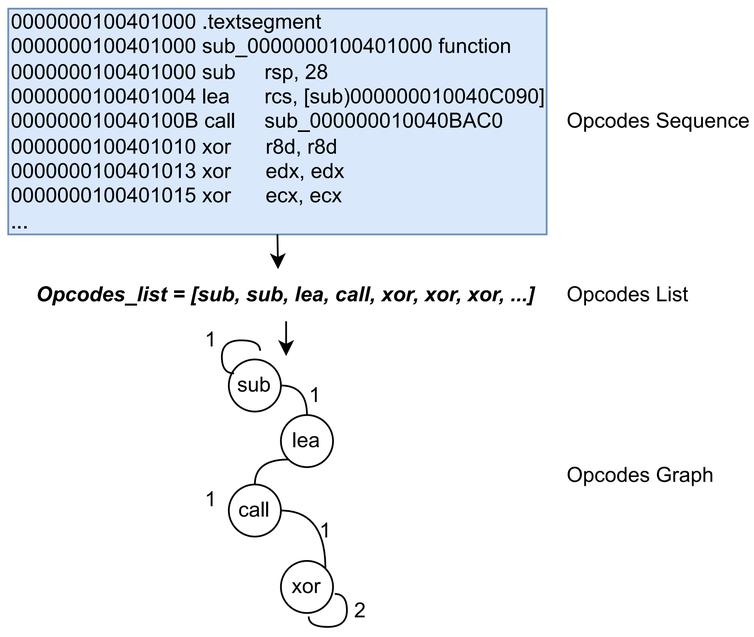

Figure 2: Regional map with some colours

2.2 OSGNN

The OSGNN creates a weighted undirected graph to each file by extracting the opcodes instructions following the CFG order and uses the General Graph Neural Networks (GeneralGNN) to classify the files. Figure 3 depicts the overall workflow process for the OSGNN.

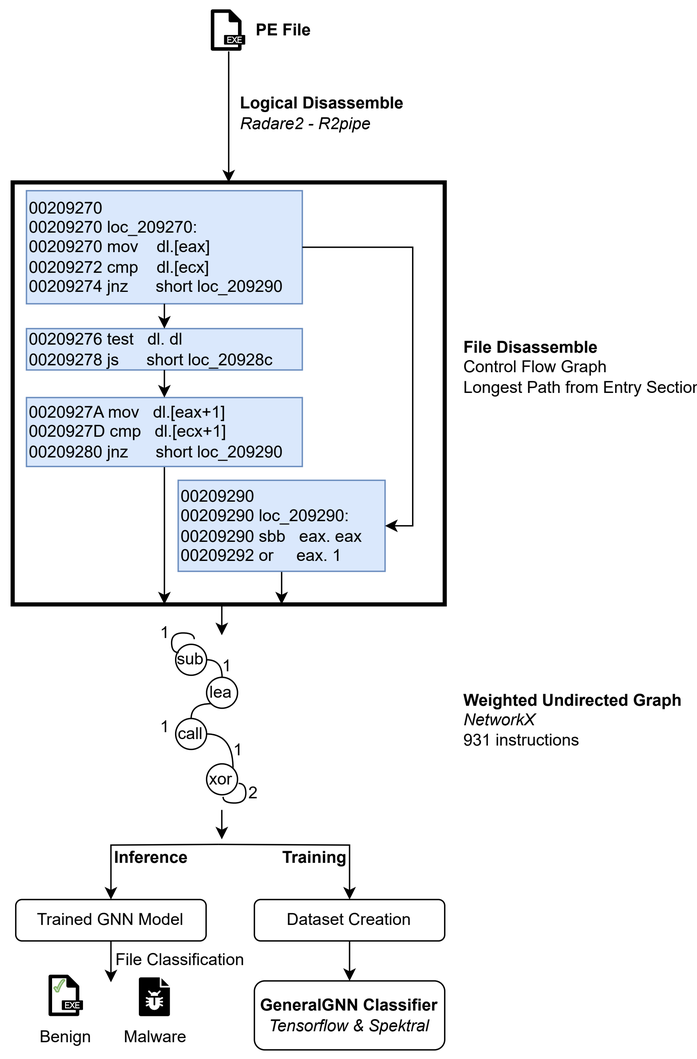

Figure 3: OSGNN workflow.

2.3 OSG academic dataset

Table 1 describes the OSG academic dataset (58,4Gb). The PE files are in $.exe$ or $.dll$ format, the malware source is the *https://bazaar.abuse.ch/*, and the dataset is available at *https://github.com/areeberg/OSG*.

Data	# Samples	Source
Malware Training	10000	Malware bazaar
Malware Testing	4000	
Trusted Training	10000	Randomly selected and evaluated
Trusted Testing	4000	
Total Files	28000	

Table 1: The OSG academic dataset description.

3. Results

We compare the OSGNN and the OSGT (with two voting systems variations) with two baseline classification methods, C-LSTM and bidirectional LSTM (bi-LSTM) with attention module.

	Accuracy(%)	TPR	FPR	MDR
OSGT XGboost	97.32	0.9846	0.0214	0.9846
OSGT LGBM	96.48	0.9803	0.0511	0.9808
OSGT Catboost	96.91	0.9854	0.0471	0.9854
OSGT Consensus Voting	**98.29**	0.9740	**0.0082**	0.9746
OSGT Major Voting	98.24	**0.9910**	0.0259	**0.9910**
OSGNN	97.92	0.9732	0.0350	0.9725
C-LSTM	89.69	0.9423	0.0765	0.8978
bi-LSTM w/ att	92.69	0.8954	0.1330	0.9451

Table 2: Detection performances on OSG academic dataset.

4. Conclusions

This paper presents the OSGT and the OSGNN, two methods for static-based malware detection. Both methods combine graph theory with artificial intelligence using the opcode sequences inputs for detecting malware portable executable. Both proposed methods handle the challenge of processing long opcode sequences without losing information in the learning process. Another strong aspect of the proposed method is that it is scalable for extended opcode sequences, as the set of opcodes is predefined, so there is a maximum number of nodes to each graph. The results show that **OSGNN** provides **99%** of malware detection, and the **OSGT** has the best accuracy overall (**98.29%**) with a fast feature extraction process.