Identifying ATT&CK Tactics in Android Malware Control Flow Graph through Graph Representation Learning and Interpretability (Student Abstract)

Jeffrey Fairbanks[1], **Andres Orbe**[2], **Christine Patterson**[3], **Edoardo Serra** [3], **Marion Scheepers** [3]
[1] Northwest Nazarene University
[2] New Jersey Institute of Technology
[3] Boise State University
jfairbanks@nnu.edu, aeo34@njit.edu, christinepatterson@u.boisestate.edu, {edoardoserra,mscheepe}@boisestate.edu

Abstract

To mitigate a malware threat it is important to understand the malware's behavior. The MITRE ATT&ACK ontology specifies an enumeration of tactics, techniques, and procedures (TTP) that characterize malware. However, absent are automated procedures that would characterize, given the malware executable, which part of the execution flow is connected with a specific TTP. This paper provides an automation methodology to locate TTP in a sub-part of the control flow graph that describes the execution flow of a malware executable. This methodology merges graph representation learning and tools for machine learning explanation.

Introduction

Malware applications create significant monetary damages and represent a menace for people. The challenging task of mitigating the effects of a malware application requires deep understanding of what the application does. Understanding of malware actions is aided by connecting these actions with the specific macro tactics, techniques, and procedures (TTP) enumerated in the MITRE ATT&CK ontology. The Control Flow Graph (CFG) of an application (or malware) describes the actions of the program during execution, and the flow of all the internal and external function calls.

To the best of our knowledge related works focus on the detection of malware or on the classification of the malware family. Often the MITRE ATT&CK® TTP are applied to associate a malware to a specific family of malware (e.g., spyware, Trojan, etc.), and this association of malware family to TTP is done by a human. The aim to this paper is to use the control flow graph to identify which subset of the actions in the graph has high likelihood of being responsible for the specific TTP.

We propose a novel approach to identify ATT&CK® TTP in a CFG by applying Graph Machine Learning techniques on Android Malware. Specifically, our approach associates the TTP with a subgraph of a CFG. We use the Graph Neural Network and SIR-GN node representation learning approach to process the CFG, and create a model that classifies the associated TTP. Furthermore, we use attribution techniques to identify the subgraph in the CFG connected with the specific TTP.

Methodology

Our methodology has three parts: (1) data collection and processing for training a graph TTP classifier, (2) SIR-GN graph representation learning procedure integrated in the graph TTP classification, (3) an Attribution procedure explaining the TTP classification task by propagating the results through the graph representation learning procedure to identify a CFG subgraph responsible for the specific TTP.

Data Collection

We use a sub sample of the Android malware (apk) provided by Virus Total [1]. For each of this malware we use the Virus Total API to collect the human curated list of TTPs (from MITRE Att&ck). Successively we convert each android malware into its corresponding control flow graph (CFG).

To obtain the CFG graph from each apk malware we use AndroGuard [2] which is a tool for static analysis of android executables. The final output is a set of different directed graphs, one for each android apk, and a ground truth which associates to each graph a list of TTPs.

SIR-GN for TTP Classification Task

Graph data are not naturally processed through standard machine learning models. Graph representation learning such as SIR-GN (Joaristi and Serra 2021) produces a vectorial representation for each node.

Given the vectorial representation of SIR-GN, (Layne and Serra 2021) provides a procedure to create a unique graph representation technique. Such techniques identify groups of nodes in a fixed number. Each group contains nodes with similar vectorial representations. Given this set of groups the method creates a pseudo adjacency matrix working on the groups that, once flattened, represents the vectorial representation of the graph. Then, the vectorial representations of two graphs are comparable if the computation of the node representations and the definition of the groups of nodes for the pseudo adjacency matrices are identical for the two graphs. To guarantee this property we use inferential SIR-GN (Layne and Serra 2021) which is a procedure able to

[1]www.virustotal.com
[2]https://github.com/androguard/androguard

perform inferences and that is pretrained on a specific family of directed random graphs. Note that since the groups are created on the basis of structural similarities among the nodes, the graph representation is invariant under permutation of the nodes in the graph.

This methodology assures a fast and comparable creation of graph vectorial representations. Once the vectorial representation of each graph is created, we use a standard machine learning model to classify the presence of specific TTP. The main technique we use is a random forests classification algorithm. This algorithm gives the best classification performance. It is important to note that graph neural networks can achieve the same task. However, it is experimentally demonstrated that they do not perform as well as SIR-GN.

TTP Attribution to Identify Subgraphs in the CFG

To identify the subgraph in the CFG responsible for the TTP classification we use SHAP(Lundberg and Lee 2017) and the interpretability procedure provided for Graph Representation in inferential SIR-GN (Layne and Serra 2021).

More specifically, SHAP is a procedure to interpret standard classification models based on the Shapley value solution concept for coalition games. SHAP, given a specific example represented by a vector of features, is able to give an attribution value for each feature in the input to the classifier. This attribution value explains the relevance of that feature for the classification. In particular, since our classification model is the random forest, we use SHAP defined for tree-based models (Lundberg et al. 2020), which is the most efficient.

Once each feature receives its attribution values, these values have to be propagated from the vectorial graph representation to the graph itself. This task is accomplished using the propagation procedure described in (Layne and Serra 2021). This procedure moves the attribution value of each edge in the pseudo adjacency matrix, created for the graph representation, to the original graph by weighting each original edge according to its contribution to the specific pseudo edge feature value. Then, each edge of the graph is provided an attribution value describing its importance for the TTP Classification. By selecting, based on attribution value, the top important edges, this procedure locates the subgraph of the CFG responsible for the specific TTP classification.

Experiments

Data Collection Information

We collected 3250 malware apks, providing 3250 graphs with an average of 5775 nodes an 12581 edges per graph. In total our dataset has 136993 nodes and 333854 edges. This set of malware has the following TTPs: “Initial Acces”, “Execution”, “Defense Evasion”, “Discovery”, “Confidential Access”, “Lateral Movement”, and “Collection”. In terms of binary classification, the first four TTPs have a ratio 40-70 % (contains the TTP) vs 60-30 % (does not contain the TTP). The remaining three are drastically unbalanced with ratio around 13 % vs 87 %.

Technique	F1 Score	Accuracy
GIN	0.627	0.669
GAT	0.495	0.675
SIR-GN	0.927	0.896

Table 1: Average Binary TTP Classification Results

Classification Results

We compare our procedure combining SIR-GN and Random Forest with Graph Attention Network (GAN) (Veličković et al. 2017) and Graph Isomorphic Network (GIN) (Xu et al. 2018). The average results for all seven binary TTP classifications are reported in Table 1. Our procedure clearly shows superior performance in comparison with graph neural networks.

Identify the Subgraph for the TTP Classification

We perform a qualitative analysis to validate how the api calls of each subgraph responsible for a TTP classification are related to the specific TTP. This analysis shows that the api calls selected by our method are always logically related to the TTP definition. In the case of “Initial Access” TTP that represents the vectors adversaries use to gain an initial foothold into a mobile device. Our automatic procedure identifies as the most important API calls the ones of user interface (i.e., android/app and android/widget need for the ads that would pop up within the browser) and ”com/madhouse-/android/ads/bj/getLeft” (Madhouse is a famous mobile ad).

Acknowledgements

This research was supported by the National Science Foundation awards NSF REU #1820685 and #CCF 1950599, and Idaho Global Entrepreneurial Mission/Higher Education Research Council #IGEM22-001

References

Joaristi, M.; and Serra, E. 2021. SIR-GN: A Fast Structural Iterative Representation Learning Approach For Graph Nodes. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(6): 1–39.

Layne, J.; and Serra, E. 2021. Inferential SIR-GN: Scalable Graph Representation Learning. arXiv:2111.04826.

Lundberg, S. M.; Erion, G.; Chen, H.; DeGrave, A.; Prutkin, J. M.; Nair, B.; Katz, R.; Himmelfarb, J.; Bansal, N.; and Lee, S.-I. 2020. From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence*, 2(1): 56–67.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, 4768–4777.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.