Obfuscation-Resilient Static Analysis for Binary Malware Detection

Sean Hodgson, Emily West, Tashi Stirewalt, and Assefaw Gebremedhin

ABSTRACT

As cyber threats become more sophisticated and our dependence on software systems increases, it is crucial to implement robust security measures. One significant aspect of software security is analyzing binary code files, often targeted by cyberattacks. Although various techniques for static analysis of binary code files are available, considerable challenges remain. These challenges include but are not limited to, overcoming code obfuscation techniques, handling dynamically generated or self-modifying code, scaling to large codebases, heavy reliance on domain expertise, and high rates of false positives and negatives in practice. In this study, we propose a tournament selection style dynamic programming LCS algorithm for learning opcode sequence signatures of similarity between different binary files known to be similar. Using similarity measures of Cosine, Jaccard, and Damerau-Levenshtein we explore the threshold prediction approach. In addition, we explore a vectorization process using a Bag-of-Words (BoW), TF-IDF, and statistical opcode combined feature embedding to train a Random Forest ML model to classify binary files into a multiple-class environment. Our experimental results demonstrate that the approach shows great potential and lays a solid foundation for efficient prediction tasks for detecting similarity between binary files resiliently to obfuscation.

RELATED WORK

The field of static analysis of binary code files has undergone significant research. Key papers include Bin Zeng’s work on foundational static analysis techniques [1], Nikos Karampatziakis’s model using structural SVMs [2], and Young-Hyun Choi et al.’s framework for taint analysis of binary executable files. Subsequent studies have introduced machine learning to binary code analysis, such as Azadhe Jallian et al.’s static signature-based malware detection method [3] and Rajchada Chanajitt et al.’s multiclass malware classification using opcode sequences [4]. However, each approach has its limitations, such as scalability, complexity, and heavy reliance on domain expertise. Despite these challenges, each paper contributes to the development of more secure and reliable software systems.

OPCODE MNEMONICS

Opcode mnemonic sequences are fundamental components of binary executables, as they represent the specific operations that a CPU will execute. These sequences can be obtained using “disassemblers”, and they are essential for analyzing binary files. Comparing opcode sequences can help identify similar operations between binaries, even if they are obfuscated or self-modifying. Different processors have unique opcode sets, and manufacturers ensure opcode compatibility across generations. A computer instruction begins with the opcode and specifies the memory address of operations that a CPU will execute. Opcode mnemonic sequences are fundamental components of similarity measures of malware. This provides a strong basis for efficient prediction tasks that can effectively detect similarities between binary files, even in the presence of intentional obfuscation.

MALWARE CLASSES

The Dike Malware Dataset, created by a Github user named iosifache, contains both benign and malicious PE and OLE files. The dataset was created by downloading PE files, sorting them into malicious or benign categories, and renaming them using their hash. Malicious OLE files were downloaded from MalwareBazaar, and benign ones were manually downloaded. A Google Cloud Function was used to scan the hashes with the VirusTotal API. The dataset is useful for training AI algorithms, with numeric labels that can be transformed for standard classification. All malware executable binaries in the dataset were downloaded using a Ubuntu Virtual Machine, and their opcode sequences were extracted for analysis.

VECTORIZATION & ML

The opcode sequences were transformed into feature embeddings for machine learning using Bag of Words (BoW), Term-Frequency Inverse-Document-Frequency (TF-IDF), and basic statistical measures. The Random Forest Classifier is a machine-learning algorithm used for both classification and regression tasks. It is an ensemble method that combines multiple decision trees to make more accurate predictions.

REFERENCES


ACKNOWLEDGEMENTS

This work is supported by funding for the VICEROY Northwest Institute for Cybersecurity Education and Research (CySER) provided by The Office of the Undersecretary of Defense for Research and Engineering, in collaboration with the Air Force Research Laboratory and Griffiss Institute.