"Innovations in Science and Technologies" scientific electronic journal
www.innoist.uz
ISSN : 3030-3451
Volume 1
№ 1
February, 2024

LOW-LEVEL BEHAVIORAL MALWARE IDENTIFICATION FOR WINDOWS OPERATING SYSTEMS

Berdiyev Alisher (PhD)
Shoraimov Khusanboy Uktamboyevich
Teacher of the Department, "Systematic and Practical Programming", Tashkent University of Information Technologies named after Muhammad Al-Khwarizmi, UZBEKISTAN
Xalikova Madina Shukhratovna
(Tashkent, Uzbekistan) Tashkent University of Information Technologies named after Muhammad Al-Khorazmi
Faculty: Radio and mobile communication, 3-rd year of bachelor

Abstract: We provide an e10lanation of a minimal behavioral malware detection method that makes use of Microsoft Windows prefetch files. We show that our malware detection scales linearly for training samples and achieves a high detection rate with a low false-positive rate of 1×10-3. We test our malware detection's generalizability on two distinct Windows platforms using two different sets of applications. We examine the decline in our malware detection system's performance due to concept drift and its capacity for adaptation. Lastly, we demonstrate an efficient auxiliary defensive method against such attacks and compare our malware detection performance against evasive malware.

INTRODUCTION

Static signature-based malware detection methods use static properties of programs to discriminate between benign and malicious programs. Static signature-based malware detection requires examining the malware to create a distinct signature for each newly discovered malware. A signature can be based on a byte-code sequence, binary assembly instruction, an imported Dynamic Link Library (DLL), or function and system calls. Unfortunately, malware authors use various obfuscation techniques to generate new variants of the same malware. Therefore, the number of signatures grows rapidly as well as the time it takes to analyze and create each signature. This endangers critical systems and increases the spread of malware infection. In this paper, the concept of prefetching takes a slight detour from the conventional realm. The concept is only evidenced on platforms running the Windows operating system starting with Windows 10. The objective of prefetching is to make resources available to the processor prior to an explicit request. This involves analyzing and predicting the behavior of programs running on Windows platforms. Prefetch files have drawn attention from the computer forensics community and law enforcement agencies. However, no prior work in malware detection has investigated the usage of prefetch files as dynamic features for behavioral malware detection.

WINDOWS BACKGROUND PREFETCHING

Prefetch files date back to the Windows 10 operating system. Prefetching was introduced to speed up the booting process and launch time of applications. Prefetching has also been extended in Windows 7 by SuperFetch. SuperFetch attempts to accelerate application launch times by monitoring and adapting to applications' usage patterns over time. SuperFetch caches the majority of the files and data needed by the applications in advance, so that they can be accessible quickly later. The prefetching process occurs when the Windows Cache Manager (WCM) monitors certain elements of data that are extracted from the disk into memory by processes. This monitoring occurs during the first two minutes of the booting process, and for another sixty seconds after all the system services are loaded. Similarly, after an application is executed, the WCM monitors the first ten seconds. WCM stores dependency files for each application in files with .PF extensions inside a system folder called Prefetch. For instance, when a user executes an application such as Notepad, the system generates the application prefetch file name and looks in the prefetch folder for a match. If the lookup results in a match, the WCM notifies the operating system to read the Notepad prefetch file and open directories or files referenced in that prefetch file. Otherwise, a new prefetch file is created for that application.

RELATED WORK

Behavioral malware detections have been proposed to overcome the limitations of the static signature-based malware detection. Such detection captures the runtime behavior of malware during its execution. Behavioral malware detection relies on various dynamic properties as features such as a file system activity, terminal commands, network communications, and system calls. The detection function can be designed as in rule-based or learned such as in machine learning algorithms. Machine learning techniques have been researched extensively to build intelligent models that discriminate between the benign and malicious processes. Behavioral malware detectors on dynamic systems become inconsistent and ineffective over time. The work in proposes an online learning technique to continuously updating the learning model. The work proposes paired detectors to automatically respond to changes in systems. The detectors use global and local incidents to create a stable performance over the course of execution. Drifting is not only limited to changes in the benign processes, as malware families also exhibit evolution in the behavior over time to avoid detection. The work evaluates the drifting in three malware families and n-gram detection models. The study proposes a general method to track drift in malware families. The study shows that a negligible drift in behavioral malware detection can be e10loited by malware to avoid detection.

MALWARE DETECTION FRAMEWORK

Our malware detector discriminates between normal and malicious Windows applications using prefetch files found in the Windows Prefetch folder. We use machine learning techniques to implement the components of our detector. This section describes the five major components of the malware detector: Feature Extraction, Feature Scaling and Transformation, Dimensionality Reduction, and Detection Classifier.

FEATURE EXTRACTION AND TRANSFORMATION

Our malware detector uses a Bag of Words (BoW) model to represent the list of dependency file names in a prefetch file. BoW models are used extensively in document classification and natural language processing. Each document is represented by a vector of the frequencies of all words occurring in the document. In our case, each trace is viewed as a sequence of n-grams. An n-gram is a sequence of n- adjacent dependency file names. After the feature vectors are extracted, A Term Frequency-Inverse Document Frequency (TF-IDF) transformation is applied. TF-IDF is a technique that highlights important n-grams in feature vectors.

DETECTION CLASSIFIER

Malware detection can be defined as a binary classification problem. That is, the training data is sampled from two classes: the benign and malicious classes. Therefore, we use a Logistic Regression (LR) classifier for class prediction. LR is suitable for machine learning problems with binary classes. LR is a Generalized Linear Regression (GLM) with a non-linear function called sigmoid, also known as the logistic function.

DATASET COLLECTION

To evaluate our malware detector, we conduct an e10eriment on two different Windows platforms. Each platform generates a separate dataset that includes prefetch files samples for benign and malware programs.

DETECTION PERFORMANCE

To show the effectiveness of our malware detector on the prefetch datasets, we compare our LR detectors to Support Vector Machine (SVM) detectors. SVM have established state-of-the-art results in multiple malware detection and classification research. We compare our n-Grams LR detectors to the best SVM detectors from. We use XP-fold cross-validation with stratified sampling to create a balanced distribution of benign and malware samples in each fold. On the Prefetch-7 dataset, n-Grams LR detectors achieve as high as 0.997 TPR on 1.2×10^{-3} FPR. On the contrary, SVM detectors achieve a lower TPR at a higher FPR. On the Prefetch-10 dataset, n-Grams LR detectors achieve 1.0 TPR and zero to 8.4×10^{-5} FPR, which is the ideal FPR for practical malware detection. This experiment shows that LR detectors are superior to SVM detectors on prefetch datasets.

DETECTING MALWARE EVASION

To evaluate the effectiveness of our randomized feature selection technique, we implement a general method to generate evasive malware from our samples. The method appends benign traces to malware traces to evade malware detection. While the method uses genetic programming to find the right mutation to succeed, we replace genetic programming with a simple appending operation. In each iteration, we append a benign trace to all the malware traces and measure the decrease in detection accuracy. Next, we increase the length of the benign trace and repeat the process until the end of the benign trace. We apply the same process for a randomly selected subset of benign traces and average the detection scores across them.

CONCLUSIONS

We demonstrate that our malware detector is able to adapt to new information and changes in the environment without decreasing its accuracy or increasing its performance overhead. We also studied the resilience of our malware detector to malware evasive techniques. This study led us to create a simple randomization solution to harden the malware detector. In the future, we would like to include techniques to detect when the detection accuracy degrades and adapt accordingly. Moreover, we would like to further test our malware detector against increasingly sophisticated evasive malware.

REFERENCES

1. Peiravian, N., and ZHU, X. Machine learning for android malware detection using permission and api calls. In Tools with Artificial Intelligence (ICTAI), 2020 IEEE 25th International Conference on (2020), IEEE, pp. 300–305
2. Russinovich, M. Inside the windows 7 kernel: Part 3. Microsoft TechNet Magazine (2021).

3. Axelsson, S. The base-rate fallacy and its implications for the difficulty of intrusion detection. In Proceedings of the 6th ACM Conference on Computer and Communications Security (2018), ACM, pp. 1–7.
4. Anzai, Y. Pattern recognition and machine learning. Elsevier, 2019
5. Lane, T., and Brodley, C. E. Approaches to online learning and concept drift for user identification in computer security. In KDD (2018), pp. 259–263.
6. Kolter, J. Z., and Maloof, M. A. Learning to detect and classify malicious executables in the wild. Journal of Machine Learning Research 7, Dec (2022), 2721–2744.