Behavioral detection of malicious activity in an EDR context

Ianău Andrei-Ioan

* University of Medicine, Pharmacology, Sciences and Technology "George Emil Palade"
Instructor: Sándor Miklós Szilágyi

Abstract- The proliferation of malware has made it necessary to improve automatic detection and classification methods. Machine learning, particularly the use of neural networks and deep learning techniques, has proven to be effective in identifying patterns in large datasets. These techniques, which have advanced significantly in recent years, have demonstrated superior classification accuracy in areas such as computer vision and natural language processing. The increase in the number and variety of malware samples necessitates the use of machine learning techniques in order to accurately detect and classify them. Moreover, the increasing capabilities of malicious actors to hide and obfuscate the malware comes as a hindrance in detecting them. Even so, the behavior of the malware remains the same so the dynamic analysis proves useful in identifying and stopping 0 day exploits and advanced persistent threats. The behavior of a program is intrinsically the same across multiple layers of intent if the scope remains the same.
Index Terms- behavior, machine language processing, malware, natural language processing.

INTRODUCTION

Similar to the NLP process, Teufl et al. call the process Machine Language Processing (MLP). In [1] the authors use the e-Participation analysis architecture, extract the various NLP techniques and adopt them for the malware analysis process. In [2] they transfer the performance improvements achieved in the area of neural networks to model the execution sequences of disassembled malicious binaries. Pre-trained neural Language Models (PTLM), such as CodeBERT, are recently used in software engineering as models pre-trained on large source code corpora as described in [3].

The authors in [4] present hybrid-Flacon, a hybrid pattern Android malware detection and categorization framework. In the [5] the researcher explores an ensemble a mechanism, which presents how the combination of byte-code and native-code analysis of Android applications can be efficiently used to cope with the advanced sophistication of Android malware. In the transformation of byte-code, the paper [6] was also used for the understanding of MLP. From [7] we get to see how they present a simple gradient-descent based algorithm for finding adversarial samples, which performs well in comparison to existing algorithms.

Huang et al. design Gossip in [8], a novel approach to automatically detect malicious domains based on the analysis of discussions in technical mailing lists (particularly on security-related topics) by using natural language processing and machine learning techniques. In order to reduce the manpower of feature engineering prior to the condition of not to extract pre-selected features In the paper [8] they develop a coloR-inspired convolutional neuRal networks (CNN)-based AndroiD malware Detection (R2-D2) system.

In the paper [9] Sewak et al. investigated and compared one of the Deep Learning Architecture called Deep Neural Network (DNN) with the classical Random Forest (RF) machine learning algorithm for the malware classification. Similar to natural language processing the authors from the paper [10] propose a novel and efficient approach to perform static malware analysis, which can automatically learn the opcode sequence patterns of malware. To this end the researchers from the paper [11] propose a joint learning approach to generating instruction embeddings that capture not only the semantics of instructions within an architecture, but also their semantic relationships across architectures. The authors in [12] propose deep learning approximate

matching (DLAM), which achieves much higher accuracy in detecting anomalies in fuzzy hashes than conventional approaches.

From the paper [13], the authors propose a cloud-based malware detection system called SaaS by leveraging and marrying multiple approaches from diverse domains such as natural language processing (n-gram), image processing (GLCM), cryptography (fuzzy hash), machine learning (random forest) and complex networks. In the paper [14] the authors propose a binary code feature extraction model to improve the accuracy and scalability of ML-based ISA identification methods. In the paper [15] the writers propose a new method which automatically detects new malware subspecies by static analysis of execution files and machine learning.

The paper [16] presents a method for using convolutional neural networks (CNNs) to classify and analyze malware. The authors propose a new architecture for CNNs, called the "MalConv" model, which is specifically designed to handle the unique characteristics of malware samples. They also present a new dataset of malware samples, called the "Malimg" dataset, that is used to train and evaluate the performance of the MalConv model. The results show that the MalConv model outperforms existing methods for malware classification and analysis, achieving high accuracy and providing valuable insights into the structure and behavior of malware samples.

All of the mentioned works use a static analysis approach in detecting the malicious samples. But there are also papers that take into consideration the dynamic analysis methodology to asses if a sample is malicious. Authors from the paper [17] propose an effective and automatic malware detection method using the text semantics of network traffic. Another approach assessed is a novel dynamic malware analysis method, which may generalize better than static analysis to newer variants as explained in [18]. To leverage the application of deep learning architectures towards cyber security, the authors in [19] consider intrusion detection, traffic analysis and Android malware detection. An anomaly detection method is implemented using Natural Language Processing (NLP) based on Bags of System Calls (BoSC) for learning the behavior of applications on Windows virtual machines running on Xen hypervisor [20]. AVMiner is presented in [21], an expansible malware tagging system that can mine the most vital tokens from AV labels. MalBehavD-V1 is presented in [22], a new behavioral dataset of Windows Application Programming Interface (API) calls extracted from benign and malware executable files using the dynamic analysis approach. MDTA is the best suitable and manageable approach for analyzing behavioral reports using a machine learning algorithm for providing security measures to identify malware without the intervention of the investigator as presented in [23]. The authors in [24] propose a deep inspection approach for multi-level profiling of crypto-ransomware, which captures the distinct features at Dynamic link library, function call, and assembly levels. In the paper [25] is presented an alternative method for malware detection, which makes use of assembly opcode sequences obtained during runtime, taking into consideration the preprocessing from [26]. The authors in [27] propose a Dynamic Ransomware Detector based on the improved TextCNN(DRDT). In the [28] the authors selected 50 among the most notorious malware variants to be as exhaustive as possible.

DISCUSSION

The article [17] presents a method for detecting Android malware using text-based features extracted from network flows. The authors propose a new approach for representing network flows as sequences of tokens, which can then be input to a classifier for malware detection. They also introduce a new dataset, called the Android Network Flows Dataset (ANFD), for evaluating the performance of their proposed method. One of the key contributions of this paper is the development of the ANFD dataset, which includes both benign and malicious Android programs. The authors also propose a novel method for generating the tokenized representation of network flows, which allows the classifier to effectively distinguish between benign and malicious programs based on their network behavior. The authors thoroughly evaluate the performance of their proposed method using the ANFD dataset, and show that it is able to achieve a high level of accuracy in detecting Android malware. In addition, they compare the performance of their method to several other state-of-

the-art approaches, demonstrating its superiority. The paper presents a promising approach for detecting Android malware using text-based features extracted from network flows. Overall, the introduction of the ANFD dataset and the tokenized representation of network flows are particularly noteworthy contributions.

In the paper [18], the authors introduce a new dataset and a method for detecting malware using machine learning and native API system calls. The proposed approach represents the sequence of native API system calls made by a program as a fixed-length feature vector, which is then input to a classifier. The introduced dataset includes both benign and malicious programs, and the authors use it to evaluate the performance of their proposed method. In the paper it is described how the features were grouped into various n-grams and weighted with Term Frequency-Inverse Document Frequency. It is mentioned that Linear Support Vector Machines (SVM) optimized by Stochastic Gradient Descent and the traditional Coordinate Descent on the Wolfe Dual form of the SVM are effective in this approach, achieving a highest of 96% accuracy with 95% recall score.

The paper [29] presents a method for detecting and preventing malicious beacon nodes in wireless sensor networks (WSNs). These malicious nodes can disrupt the location discovery process and compromise the security of the network. The authors propose a new technique for detecting malicious beacon nodes by analyzing the patterns of the beacons' radio frequency (RF) signals. They use a machine learning algorithm to classify the beacon nodes as either malicious or benign based on their RF signal patterns. The authors also evaluate the performance of the proposed technique using simulations and show that it is able to detect malicious beacon nodes with high accuracy while having low false positive rate.

In the paper is presented the way the API calls were processed. As you can see in the figure represented below, the system trace contains a lot of details that are not necessary to the model. This means that a preprocessing is needed. A python script was used to take the System Call Trace and add them as tokens without the calling parameters, resulting in only the names of the functions. This allows for generalization as the first two rows are descriptive of a DLL upload, the parameters are the same everywhere so it does not add any new information to the already called function. Of course, the memory that the parameters are stored have no significant true value as the memory is pseudo-randomly assigned so they are also removed. And in the end, the size factor at the end will also be removed because it just says the stack size at which the program runs.

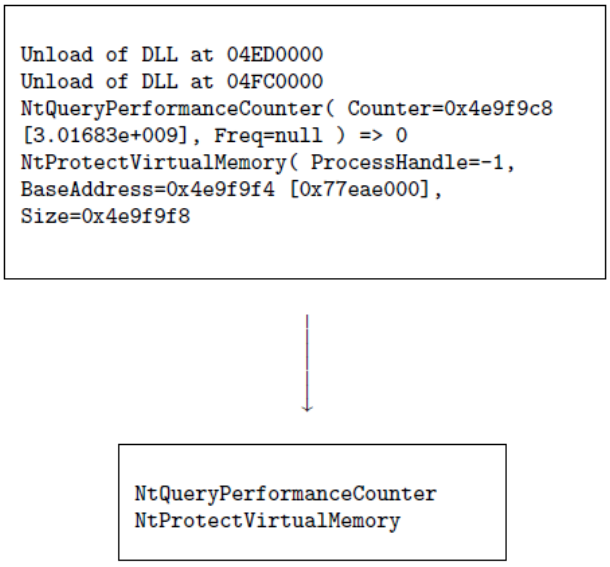

Table 1 Transforming the system trace into preprocessed input

The evaluation on 80% training and 20% testing shows that the proposed method is able to achieve a high level of accuracy in detecting malware and outperforms several state-of-the-art approaches. In addition, the use of a new dataset and fixed-length feature vectors derived from native API system calls are significant contributions of the paper.

	Precision	Recall	F1-Score
Benign	1.00	0.82	0.92
Malware	0.94	1.00	0.97
Average/Total	0.96	0.95	0.95

Table 2 Results of SGD Classifier

	Precision	Recall	F1-Score
Benign	1.00	0.79	0.88
Malware	0.91	1.00	0.95
Average/Total	0.94	0.93	0.93

Table 3 Results of SGD Classifier

	Avg. Precision	Avg. Recall
TFIDF & 10-Grams	0.93	0.92
TFIDF & Unigrams	0.90	0.89
Term Frequency & 10-grams	0.78	0.78
None	0.75	0.72

Table 4 Average Scores Comparing Options

Overall, the proposed method shows promise in the detection of malware. One of the key differences between this paper and other papers is the focus on native API system calls as a feature for malware detection. Many other papers on malware detection focus on different types of features, such as code, API calls, network behavior, or runtime opcodes. The use of native API system calls as a feature allows the NtMalDetect approach to potentially capture different characteristics of malware that may not be captured by other features. Another difference is the use of a machine learning classifier rather than a deep learning classifier. While many papers on malware detection use deep learning techniques, the NtMalDetect approach uses a machine learning classifier, which may have different strengths and limitations in terms of performance and complexity.

[19] is a paper that explores the use of deep learning techniques for enhancing cyber security. The authors present several case studies in which deep learning architectures have been successfully applied to various cyber security tasks, such as intrusion detection, malware classification, and network traffic analysis. One of the key insights of the paper is the demonstration that deep learning can be a powerful tool for addressing complex cyber security challenges. The authors provide examples of how deep learning models can be trained on large datasets and can learn to automatically extract relevant features from the data, leading to improved performance on various tasks. One of the strengths of the paper is the inclusion of several real-world case studies, which illustrate the effectiveness of deep learning in various cyber security applications. The authors also provide an overview of the different deep learning architectures that have been used in these case studies and discuss their pros and cons. Overall, the article is a valuable resource for those interested in using deep learning for enhancing cyber security. It provides a comprehensive overview of the state-of-the-art in this area and illustrates the potential of deep learning techniques for addressing complex cyber security challenges.

In the paper [20], the authors present a novel approach for detecting anomalous system call sequences (ASCSs) in computer systems using a combination of natural language processing (NLP) techniques and virtual memory introspection (VMI). The authors propose a method for representing ASCSs as natural language texts

and apply NLP techniques to identify anomalies. They also introduce a new dataset, called the Virtual Memory Introspection Dataset (VMID), for evaluating the performance of their proposed method. The VMID dataset, which includes both normal and anomalous system call sequences, is a key contribution of the paper. The authors also present a method for generating a text representation of ASCSs, which allows the NLP-based approach to effectively identify anomalies. The proposed method is thoroughly evaluated on the VMID dataset, yielding a high level of accuracy (99%) in detecting anomalous system call sequences. In addition, the authors compare the performance of their method to several state-of-the-art approaches, demonstrating its superiority.

The framework of the system is comprised of four elements:

"1) Virtualization - Virtualization: In this module, smart memory introspection is performed on Virtual Machine (VM) using VMI API to introspect and perform memory forensics. This module consists of different sub-modules such as Introspector and Security Agent.

a) Introspector: This module extracts low-level data from the memory of virtual machines running on a hypervisor, and transfers this data to agent listener(s) for anomaly analysis. The Introspector interfaces with hypervisors to ensure that the state of the virtual machines (running, stopped, or shut-down) can be manipulated, and VMs can be added and deleted as needed.

b) Security Agent: This sub-module initiates scans on VMs using the LibVMI library to perform introspection. Its primary mechanism is to extract data from a VM and send the data to the agent listener for further analysis. The Security Agent has various features that allow the agent to scan processes, invariant data structures, and to monitor files changes.

2) Advanced cyber analytics: This module comprises of different machine learning and deep learning algorithms to train the model and perform a test on that model for further prediction and analysis of data. The baseline data is considered as benign data, and the test vector injected data is known as malicious data. The data extracted by using the introspection module is stored on a database server and then analyzed using different cutting-edge machine learning techniques.

3) Malware repository: This repository consists of a massive set of malware that compromises kernel-level data structures. This repository includes different malware for Windows and Linux. This malware repository also consists of custom malware sets to compromise the specific context of kernel data structures.

4) Test control center: With the help of the Test Control Center module, the operator can control and manage the whole framework and its modules with a user interface. The operator can handle the VM operations, such as creation, deletion, stop, start, pause, and view. Also, the operator can control the VMs by installing or running malware and benign applications. The operator visualizes the processed results from the Advanced Cyber Analytics module for further analysis."

The feature extraction mechanism is getting the system call trace of the functions called by the malware sample. Each system call is then represented on a vector and then a cosine similarity function is applied to check if the behavior is anomalous.

Reviewing the article [21], we find that it presents an algorithm for mining anti-virus labels from a large dataset of malware samples. The authors propose a novel approach named AVMiner, which is designed to be expansible and able to preserve the semantic meaning of the labels. They also introduce a new dataset, called the Malware Labels Dataset (MLD), for evaluating the performance of their proposed method. Unfortunately, the method is not detailed in the article. However, one of the key contributions of the paper is the development of the MLD dataset, which includes a large number of malware samples with manually-annotated labels. The authors also propose a novel method for mining the labels from the dataset, which is able to identify a large number of relevant labels and preserve their semantic meaning.

The paper [22] introduces MalDetConv, a method for detecting malware using natural language processing (NLP) and deep learning techniques based on the behavior of the malware. The authors propose a new approach for analyzing the behavior of malware and classifying it as malicious or benign based on extracted features. They also present the Malware Detection Corpus (MalDetCorpus), a new dataset for evaluating the performance of their proposed method. The MalDetCorpus dataset, which includes a large number of malware samples with annotated behavior descriptions, is a key contribution of the paper. The

authors also propose a novel method for extracting features from the behavior descriptions using NLP and deep learning techniques, allowing the MalDetConv framework to effectively classify malware based on its behavior.

In the work [24] we find a description of a new way of analyzing a type of malware called crypto-ransomware using machine learning. The authors introduce a method for representing the malware as a combination of features extracted from different parts of it, such as the code and network behavior. They also introduce a new dataset, called the Crypto-Ransomware Profiling Dataset (CRPD), to test their method. The CRPD dataset is a key part of the paper because it includes a large number of examples of crypto-ransomware with important information about them. The authors also propose a new way of representing crypto-ransomware using multiple features, which allows for a more thorough analysis of the malware. The authors test their method using the CRPD dataset and show that it is very good at analyzing crypto-ransomware. This new method shows promise for helping to understand and protect against crypto-ransomware.

In the paper [25] is presented a method for detecting malware that targets the Windows operating system using deep learning and information about the runtime opcodes of the malware. Runtime opcodes are the instructions that a computer executes while a program is running. The authors propose a new approach for representing the runtime opcodes of Windows programs as a sequence of tokens, which can then be input to a deep learning classifier for malware detection. They also introduce a new dataset, called the Windows Malware Detection Dataset (WMDD), for evaluating the performance of their proposed method. The authors collect a dataset of Windows programs, including both benign and malicious programs, and annotate the runtime opcodes of each program. Then they preprocess the runtime opcodes by representing them as a sequence of tokens. The method proposed in the paper involves converting the opcodes to a sequence of hexadecimal values and then applying a tokenization process to represent each value as a unique token. They train a deep learning classifier on the preprocessed runtime opcodes to detect Windows malware. After this, they test the classifier on a separate dataset of Windows programs to evaluate its performance in detecting malware. If necessary, in the paper it is recommended to fine-tune the classifier by adjusting its architecture, training procedure, or other parameters to improve its performance on the dataset at hand.

The paper [27] presents a method for detecting ransomware attacks in real-time using a convolutional neural network (TextCNN). The authors propose a new approach for representing the API calls made by a program as a sequence of tokens, which can then be input to the TextCNN for classification. The authors also introduce a new dataset, called the Ransomware Dynamic Detection Dataset (RD3), for evaluating the performance of their proposed method. The authors also propose a novel method for generating the tokenized representation of API calls, which allows the TextCNN to effectively classify programs based on their API usage. The authors thoroughly evaluate the performance of their proposed method using the RD3 dataset, and show that it is able to achieve a high level of accuracy in detecting ransomware. In addition, they compare the performance of their method to several other state-of-the-art approaches, demonstrating its superiority. Overall, the paper [27] presents a promising approach for detecting ransomware attacks in real-time and contributes to the ongoing research in this area. The introduction of the RD3 dataset and the tokenized representation of API calls are particularly noteworthy contributions.

From the paper [28] we find that “UMUDGA” presents a new algorithm, called the Unsupervised Method for Profiling Algorithmically Generated Domain Names, for use in detecting botnets. The authors propose a method for generating the dataset by crawling the internet to collect a large number of algorithmically generated domain names (AGDNs) and labeling them as either benign or malicious based on various features. One of the key contributions of this paper is the development of the UMUDGA dataset, which includes a large number of AGDNs with manually-annotated labels. The authors also propose a method for generating the dataset, which allows for the collection of a diverse and representative set of AGDNs. The authors evaluate the usefulness of the UMUDGA dataset for detecting botnets by training and testing a machine learning classifier on the dataset. They show that the classifier is able to achieve a high level of accuracy in detecting botnets based on AGDNs.

CONCLUSIONS

The approach to detect malicious activity with NLP techniques presents itself as worthy of trial in the market as it seems that the results are promising.

Bibliography

[1] P. Teufl, U. Payer and G. Lackner, "From NLP (Natural Language Processing) to MLP (Machine Language Processing)," 2010.

[2] B. Kolosnjaji, A. Zarras, G. Webster and C. Eckert, "Deep Learning for Classification of Malware System Call Sequences," p. 137–149, 2016.

[3] D. Goel, R. Grover and F. H. Fard, "On The Cross-Modal Transfer from Natural Language to Code through Adapter Modules," 2022.

[4] P. Xu, C. Eckert and A. Zarras, "hybrid-Falcon: Hybrid Pattern Malware Detection and Categorization with Network Traffic and Program Code," 2021.

[5] P. Xu, "Android-COCO: Android Malware Detection with Graph Neural Network for Byte- and Native-Code," 2021.

[6] P. Teufl, U. Payer and G. Lackner, "From NLP (Natural Language Processing) to MLP (Machine Language Processing)," 2010.

[7] U. Jang, X. Wu and S. Jha, "Objective Metrics and Gradient Descent Algorithms for Adversarial Examples in Machine Learning," December 2017.

[8] C. Huang, S. Hao, L. Invernizzi, J. Liu, Y. Fang, C. Kruegel and G. Vigna, "Gossip," April 2017.

[9] M. Sewak, S. K. Sahay and H. Rathore, "Comparison of Deep Learning and the Classical Machine Learning Algorithm for the Malware Detection," 2018.

[10] R. Lu, "Malware Detection with LSTM using Opcode Language," 2019.

[11] K. Redmond, L. Luo and Q. Zeng, "A Cross-Architecture Instruction Embedding Model for Natural Language Processing-Inspired Binary Code Analysis," 2018.

[12] F. Uhlig, L. Struppek, D. Hintersdorf and K. Kersting, "Transformer-Boosted Anomaly Detection with Fuzzy Hashes," 2022.

[13] Y. Zhang, W. Ren, T. Zhu and Y. Ren, "SaaS: A situational awareness and analysis system for massive android malware detection," *Future Generation Computer Systems,* vol. 95, p. 548–559, June 2019.

[14] D. Sahabandu, S. Mertoguno and R. Poovendran, "A Natural Language Processing Approach for Instruction Set Architecture Identification," 2022.

[15] Y. Nagano and R. Uda, "Static analysis with paragraph vector for malware detection," January 2017.

[16] B. Kolosnjaji, G. Eraisha, G. Webster, A. Zarras and C. Eckert, "Empowering convolutional networks for malware classification and analysis," May 2017.

[17] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao and M. Conti, "Detecting Android Malware Leveraging Text Semantics of Network Flows," *IEEE Transactions on Information Forensics and Security,* vol. 13, p. 1096–1109, May 2018.

[18] C. W. Kim, "NtMalDetect: A Machine Learning Approach to Malware Detection Using Native API System Calls," 2018.

[19] R. Vinayakumar, K. P. Soman, P. Poornachandran and S. Akarsh, "Application of Deep Learning Architectures for Cyber Security," p. 125–160, 2019.

[20] S. K. Peddoju, H. Upadhyay, J. Soni and N. Prabakar, "Natural Language Processing based Anomalous System Call Sequences Detection with Virtual Memory Introspection," *International Journal of Advanced Computer Science and Applications,* vol. 11, 2020.

[21] L. Chen, Z. He, H. Wu, Y. Gong and B. Mao, "AVMiner: Expansible and Semantic-Preserving Anti-Virus Labels Mining Method," 2022.

[22] P. Maniriho, A. N. Mahmood and M. J. M. Chowdhury, "MalDetConv: Automated Behaviour-based Malware Detection Framework Based on Natural Language Processing and Deep Learning Techniques," 2022.

[23] S. S. Vanjire and M. Lakshmi, "MDTA: A New Approach of Supervised Machine Learning for Android Malware Detection and Threat Attribution Using Behavioral Reports," p. 147–159, July 2021.

[24] S. Poudyal and D. Dasgupta, "Analysis of Crypto-Ransomware Using ML-Based Multi-Level Profiling," *IEEE Access,* vol. 9, p. 122532–122547, 2021.

[25] E. S. Parildi, D. Hatzinakos and Y. Lawryshyn, "Deep learning-aided runtime opcode-based Windows malware detection," *Neural Computing and Applications,* vol. 33, p. 11963–11983, March 2021.

[26] P. Teufl, U. Payer and G. Lackner, "From NLP (Natural Language Processing) to MLP (Machine Language Processing)," 2010.

[27] B. Qin, Y. Wang and C. Ma, "API Call Based Ransomware Dynamic Detection Approach Using TextCNN," June 2020.

[28] M. Zago, M. G. Pérez and G. M. Pérez, "UMUDGA: A dataset for profiling algorithmically generated domain names in botnet detection," *Data in Brief,* vol. 30, p. 105400, June 2020.

[29] D. Liu, P. Ning and W. Du, "Detecting Malicious Beacon Nodes for Secure Location Discovery in Wireless Sensor Networks," 2009.

[30] K. Bojan , Z. Apostolis , W. George and E. Claudia , "Deep Learning for Classification of Malware," 2016.