Integrated Feature Extraction Approach Towards Detection of Polymorphic Malware In Executable Files

Emmanuel Masabo *masabem@gmail.com*
College of Computing and Information Sciences
Makerere University, Kampala, Uganda

Kyanda Swaib Kaawaase *kswaibk@cis.mak.ac.ug*
College of Computing and Information Sciences
Makerere University, Kampala, Uganda

Julianne Sansa-Otim *sansa@cit.ac.ug*
College of Computing and Information Sciences,
Makerere University, Kampala, Uganda

Damien Hanyurwimfura *hadamfr@gmail.com*
College of Science and Technology
University of Rwanda, Kigali, Rwanda

Abstract

Some malware are sophisticated with polymorphic techniques such as self-mutation and emulation based analysis evasion. Most anti-malware techniques are overwhelmed by the polymorphic malware threats that self-mutate with different variants at every attack. This research aims to contribute to the detection of malicious codes, especially polymorphic malware by utilizing advanced static and advanced dynamic analyses for extraction of more informative key features of a malware through code analysis, memory analysis and behavioral analysis. Correlation based feature selection algorithm will be used to transform features; i.e. filtering and selecting optimal and relevant features. A machine learning technique called K-Nearest Neighbor (K-NN) will be used for classification and detection of polymorphic malware. Evaluation of results will be based on the following measurement metrics—True Positive Rate (TPR), False Positive Rate (FPR) and the overall detection accuracy of experiments.

Keywords: Malware Detection, Static Analysis, Dynamic Analysis, Polymorphic Malware, Machine Learning

1. INTRODUCTION

Nowadays, the world relies on information technology (IT) as it facilitates human daily activities. Multiple devices such as personal computers, laptops, tablets, etc., have gained popularity when used for accessing IT. Such devices are widely used in offices, homes, etc., for multiple services. However, there is a great concern regarding security in the use of IT. A lot of malware such as rootkits, spyware, trojan horses, bots and other types are released by attackers on a daily basis. According to the Symantec report [1], there were 317 million pieces of malware injected in year 2014, which means that almost one million new threats were created every day. Many developers have tried to overcome this situation through creation of anti-malware programs—such as Symantec antivirus [1], Lavasoft [2] and many others. However, these anti-malware have quite limited efficiency in identifying and eliminating threats [3]. This gap has attracted much research interest in the area, especially on malware analysis to achieve new reliable and more promising algorithms.

Malware are becoming more sophisticated with polymorphic behaviors [3], [4], [5] in order to hide themselves from analysis and detection. Polymorphism is the capability of the malware to change identity at any instance of infection. It is not a new malware, but it is a variant of existing malware which is packed and contains some code obfuscation. These variants of existing malware will confuse anti-virus and might then be detected as benign due to the lack of appropriate signatures to contain them. A big problem that arises is how to efficiently deal with such polymorphic malware.

Motivation
Previous researchers have met a number of challenges in addressing this issue. Most proposed solutions have been relying on extracting behavioral features from malware and use different machine learning methods to implement detection approaches. It's in that context that this research aims at designing a novel approach in terms of feature engineering and detection mechanisms. This approach will integrate advanced components of two powerful analysis techniques for a comprehensive malware dissection and feature extraction process. These techniques are known as advanced static and advanced dynamic analyses. Structural and behavioral features will be extracted. A Machine learning technique called K-NN will be used in the process of designing or implementing a detection approach. The overall objective will be to achieve high detection accuracy that significantly reduces false alarms, increases the rate of correctly detected malware and outperforms previous approaches. The study will mainly focus on malicious portable executable (PE) files. The PE file format is a data structure that contains necessary information for the Operating System loader to manage executable code [6].

2. RELATED WORK
Malware analysis helps to examine the capabilities of a malicious program in order to better investigate the nature of security breach incident and prevention of any further infections [6]. There are two commonly used malware analysis techniques, i.e. static analysis [6] and dynamic analysis [6].

Static analysis [7], [5] is a process whereby information about malicious program is extracted without being executed. Non execution of the malicious code makes static analysis safer compared to dynamic analysis in which malicious code must be executed on the machine used for analysis [6]. Basic static analysis can show basic information about the malicious program such as its version, file size, file format, any suspicious imports, etc. Basic static analysis is straightforward and quick, but not very effective as important details can be missed [6]. Advanced static analysis deals with code/structure analysis in which the knowledge of assembly language, compiler code and Operating system concepts are required [6]. Malware functionality is analyzed through inspecting the internal code of the malware [4].

Dynamic analysis [7],[8] is the process of analyzing a malicious program through execution and monitor its run time functionality of such an execution. Basic dynamic analysis consists of observing the behavior of a malware and does not require deep programming skills while the advanced dynamic analysis makes a profound examination of the internal state of a running malicious program while extracting detailed information [6]. The code is analyzed at run time and any code hidden through packing is revealed [9]. The identity of a malware is programmatically identified. Function calls, parameter analysis and information flow are all visualized [4].

Most research on malware variants or polymorphic malware is based on behavioral analysis in which malware functionalities are investigated at run time [8]. Ahmadi et. al [7] developed a method to detect malicious files based on behavioral sequential patterns in which the behavior of malicious executables were analyzed. API calls were extracted and a log was created. The repetitive patterns in the API call log were considered to make the initial dataset for classification. The Fisher score algorithm was used for feature selection in their research while support vector machines was combined with decision tree algorithms and used for malware detection. The training dataset contained 806 malware and 306 benign files. A malware detection accuracy of 95% was achieved. Cesare et. al [9] detected new malware samples and variants of existing ones through generating signatures for any newly identified malware. It handles unpacking. The sample consisted of 15409 malware out of which, their results showed 88.26% were classified as variants of existing ones and 34.24% were classified as known malware. The combined static and dynamic analysis in [5] was done on a malicious file called TT.exe which breaks into a system and performs malicious activities. The advantages of combining both methods have been found to be beyond preliminary analysis as a malware can deeply be dissected to reveal more of its functionalities.

Behavioral analysis based on machine learning [10] focused on malware classification and clustering. 1270 malware samples of different format, namely; pdf, executables, html, zipped, jpeg, etc. were investigated. Logic Model Tree and K-Means algorithms were used for the task of classification and clustering respectively. The results show that 18% of analyzed malware were embedded with networking capabilities to connect to the outer world, while 82% aimed to corrupt the system locally or network resources. Malware were also grouped successfully according to their file format types. Comar et. al in [10] combined supervised and unsupervised learning method to capture packets from a live network connection and use the knowledge of existing attacks to classify new network flow as either new attack, existing or variants of existing. K-Nearest Neighbor (K-NN) and Support Vector Machine (SVM) algorithms have been used in the classification process. 216,899 flows have been captured, out of which 4,394 (2%) were found malicious and categorized in 38 known malware classes.

Liang et. al, in [11] proposed a novel method to detect variants based on behavioral dependency. Features were extracted using Temu dynamic analysis software and were customized noise removal. In their research, Jaccard algorithm was used for similarity calculation. Their experiments were done using two different malware and six variants of a Trojan malware called Ghost. Results showed that the two different malware had a weighted similarity of 27%, whereas the six variants had a strong weighted similarity ranging from 86.16% to 96.2%.

Naidu et. al, in [12] proposed a technique that automatically generates super- signatures to contain polymorphic malware. They used hexadecimal characteristics as features as well as Needleman-Wunsch and Smith-Waterman algorithms for string matching. Experiments were done on multiple variants of "JS.Cassandra" malware and detection rate was 96.59%. Table 1 below, discusses more about different techniques as well as their strengths and limitations.

Technique	Characteristics	Strengths/Contribution	Limitations
Malware detection by behavioral sequential patterns][13]	-Uses API calls based features. - Random forest and SVM are used for classification	-Effective in detecting malware variants.	-Static features not considered. -High rate of False Positive detections
Malicious data classification using structural information and behavioral specifications in executables[8]	-Uses common static and API call features -J48 algorithm is used for classification	-Can detect similarities among malware samples	-can't handle anti analysis features
A Behavior-Based Malware Variant Classification Technique[p70]	-API calls based features are used. -Weighted similarity among malware behaviors is calculated using Jaccard similarity algorithm	-Effective at detecting similarity among malware variants	-Static features not considered. -High rate of False Positive detections
Combining supervised and unsupervised learning for zero-day malware detection[14]	-Network flow based features are extracted using IDS/IPS -Uses one class SVM algorithm for classification.	-Effective at detecting polymorphic. -Can detect new malware from a suspicious flow	-Limited to network based features - High rate of False Positive detections

Needleman-Wunsch and Smith-Waterman Algorithms for Identifying Viral Polymorphic Malware Variants[12]	-static Hexadecimal based features are extracted. - Needleman-Wunsch and Smith-Waterman Algorithms are used for creating effective signatures.	-Effective at generating appropriate signatures to contain polymorphic malware.	-only static features are considered. -Can have false positive detections
Proposed solution: Integrated Feature Extraction Approach towards Detection of Polymorphic Malware in Executable Files	Comprehensive dissection of malware using Advanced static analysis and advanced dynamic analysis as discussed in Table 2 and 3. Correlation based feature selection (CFS) algorithm. CFS helps in creating good feature subsets that are highly correlated with the predicted class. This method is chosen because it is fast, produces high ranking and correlated features compared to alternative methods used in other techniques. As we'll have a big feature set, the accurate automated selection is also well done with CFS. K-NN classifier to detect polymorphic malware with high accuracy. Comparing to other methods used in previous methods, K-NN is selected due to its good performance and robustness in dealing with large datasets with many features[15].	This method will address the limitations of previous techniques by developing an approach with the following components: -Detection of polymorphic malware with high accuracy. -Significantly minimized false detection alarms. -Consideration of hidden malware functionalities, especially analysis/detection avoidance capabilities. -Increased detection performance in case of large dataset and many features. -Fast detection process	

TABLE 1: Detection Techniques Comparison.

3. METHODOLOGY

3.1 Proposed Detection Approach

The proposed detection approach is illustrated by the flowchart in figure 1. A malware sample is analyzed using advanced dynamic analysis and advanced static analysis. Dynamic analysis leads to the extraction of behavioral features. For static analysis a sample is first investigated to identify packing traces. If it's packed, it will therefore be unpacked. Once a packet is not packed, structural features are extracted. All features are combined to make a big feature dataset. These features will then be filtered to select a reduced dataset which comprises most optimal features that are relevant for classification task. Lastly, the classification process will be done based on previously preprocessed features in order to detect polymorphic malware.

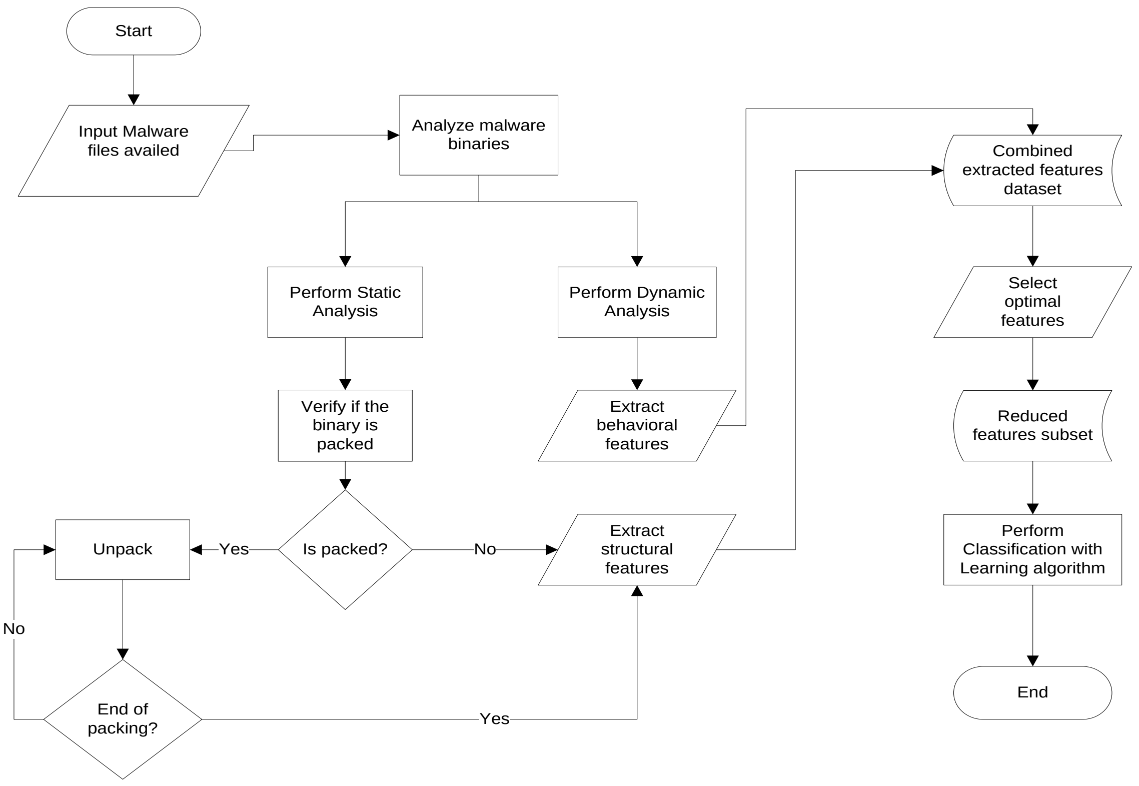
Start
Input Malware files availed
Analyze malware binaries
Perform Static Analysis
Perform Dynamic Analysis
Verify if the binary is packed
Extract behavioral features
Is packed?
Yes
No
Unpack
Extract structural features
No
End of packing?
Yes
Combined extracted features dataset
Select optimal features
Reduced features subset
Perform Classification with Learning algorithm
End

FIGURE 1: The Detection Approach Flowchart.

3.2 Data Collection

We will collect relevant malware samples for researchers collected from online repositories [16] such as—Open Malware, Malware repository, Malware Samples[16].

3.3 Extraction of Features from Malware Data Samples

Having acquired relevant malware samples, the research seeks to have as more descriptive information as possible about a given malware through feature extraction. Features are identification characteristics of malware used to build the detection knowledge. To extract these features this research will employ a combination of advanced static and advanced dynamic analysis techniques. Tools to be used are shown in table 2 and the main features to be extracted are shown in table 3.

Tools	Description
PEiD and UPX	For identifying packer and compiler information and regeneration of original unpacked file.
IDA pro	For disassembling the malware binary for further analysis
Process Monitor	For viewing real-time file system, process activity and registry, Network activity, API calls, Mutex, Self-modifying code traces
Dependency Walker	For exploring the Dynamic Link Libraries (DLL) and imported functions.
Regshot	To capture and compare registry snapshots to discover any modifications
ApateDNS	For controlling DNS requests and response in case of malware network activity
Wireshark:	For capture and analyze network traffic
INetSim	for simulating network services such as DNS, HTTP, HTTPS, FTP, IRC, DNS, SMTP

TABLE 2: Tools and Their Characteristics.

Feature
Strings, DLLs functionality, User defined functions, Self-modifying code traces, Attacker identification, Exception parameters, Passwords, Anti-debug/analysis constructs, Kernel mode activity, User mode activity, File activity, Rootkit functionalities, Windows Digital signature, Hooking, Persistence , privilege escalation attack, DLLs injection, Runtime DLLs, Process replacement functionalities, Entropy, Checksum, API calls, code noise patterns, File activity, Registry, Service, Mutex, Processes, Network

TABLE 3: Features to be Extracted.

3.4 Feature Selection

Not all features extracted will be relevant for this research. Therefore, after extracting features, some features with low impact will be removed because they might have a negative impact on the overall accuracy. Techniques such as Fisher score algorithm [7], correlation based algorithm [14] and tree based algorithm [17] are good at feature selection process. Fisher score algorithm selects high ranking features and tree-based feature transformation approach selects and removes noise from data. Correlation based feature selection (CFS) algorithm helps in creating good ranking feature subsets that are highly correlated with the predicted class, especially in case of large feature dataset. CFS is therefore chosen to be used for this research.

Most relevant features will be retained and will be candidate instances of the training dataset. Relevant features will then form an optimal feature subset to be used in classification. The advantages of feature selection include:

reduced overfitting (avoiding the worst case scenario in prediction), significant reduction of training time and improved accuracy. CFS algorithm will be used for selecting optimal features. The merit of a feature subset S with k features is computed according to equation 1.

$$Merit_{S_k} = \frac{k\overline{r_{cf}}}{\sqrt{k+k(k-1)\overline{r_{ff}}}} \quad (1)$$

where $\overline{r_{cf}}$ is the average value of feature classification correlations, and $\overline{r_{ff}}$ is the average value of feature-feature correlations.

CFS will finally be computed according to equation 2

$$CFS = \max_{S_k}\left[\frac{r_{cf_1}+r_{cf_2}+\cdots+r_{cf_k}}{\sqrt{k+2\left(r_{f_1f_2}+\cdots+r_{f_if_j}+\cdots+r_{f_kf_1}\right)}}\right] \quad (2)$$

, where r_{cf_i} and $r_{f_if_j}$ variables are correlations.

3.5 Designing The Detection Approach

This task will mainly consist of building classification models that will optimally generalize the predictions in detecting polymorphic malware. The choice of a classifier depends on the type of features, dataset size and also the problem to be solved [15]. Classifiers like Decision Trees (DT) [15], Support Vector Machines (SVM) and K-nearest neighbor (K-NN) perform well in different situations [15]. SVM and K-NN are suitable for this research as they can support similarity function testing for prediction. SVM is suitable to work with few data points because it is slow [15]. K-NN is good for many data points and it is fast [15]. Therefore, to perform classification, the research proposes to use K-nearest neighbor (K-NN) algorithm [18] because it has the ability to compute similarities among instances. K-NN will be implemented and customized to meet the challenges of classification. Distances are calculated between the targeted instance and all other instances. The shortest distance shows the strongest similarity. When there is a strong similarity, it is means that there are variants in instances [18]. These variants are signs of polymorphism.
Nearest neighbors will be computed as follows:

1. Let $x_i^{(j)}$represents all training examples, where i is the number of features and j is the number of instances.
2. Let k be the number of nearest neighbors determined beforehand in building (K-NN) model,
3. Any distance between the targeted instance ($x_i^{(b)}$) and all training examples $x_i^{(j)}$. Euclidean distance will be computed as:

   $dist\left(x_i^{(b)}, x_i^{(j)}\right) = \sqrt{\sum_{a=1}^{i}(x_a^{(b)} - x_a^{(j)})^2},$ where $a \leq i$ and $b \leq j$ (3)
4. Identify all categories of training instances for the sorted values under k

3.6 Evaluation and Validation

To evaluate the results, main performance metrics namely True positive (TP), False positive (FP), True negative (TN), and False negative (FN) will be calculated. True Positive rates (TPR) will give the percentage of correctly identified as polymorphic samples. False Positive Rates (FPR) will give the percentage of wrongly identified as polymorphic samples. The overall accuracy of the model will be calculated based on total number in the sample and those that were correctly detected as shown in equation 6. Performance metrics are calculated as follows:

$$TPR = \frac{TP}{TP+FN} \quad (4)$$

$$FPR = \frac{FP}{FP+TN} \quad (5)$$

Overall accuracy is the proportion of the total number of predictions that are correct and will be computed as follows:

$$Accuracy = \frac{(TP+TN)}{TP+FP+TN+FN} \quad (6)$$

4. EXPECTED OUTCOME

The expected outcome is a polymorphic malware detection approach that increases overall detection performance in terms of accuracy and speed. Accuracy is measured by the high rate of malware correctly identified as polymorphic as well as significantly minimized rate of false detection alarms. Detection speed will be high due to the optimized feature engineering process.

5. CONCLUSION AND FUTURE WORK

The research is intending to address the issue of polymorphic malware detection. This will be done by collecting malware samples, analyzing them and extract features using advanced static and advanced dynamic analyses techniques. Feature selection will be done using Correlation Feature selection algorithm. Classification will be done using machine learning technique called K-NN. Evaluation of detection performance will be done based on measuring overall accuracy, true positive rate as well as false negative rates. Future work will focus, firstly on the implementation of the proposed approach and provide simulation results; secondly on the customization of different machine learning algorithms for more optimized higher detection rates.

6. ACKNOWLEDGMENT

We would like to thank the Mobility to Enhance Training of Engineering Graduates in Africa (METEGA) and the Regional Universities Forum for Capacity Building in Agriculture (RUFORUM) for financially supporting our research.

7. REFERENCES

[1] Symantec, “015 Internet Security Threat Report,” Internet Security Threat Report, 2015. [Online]. Available: https://www.itu.int/en/ITU-D/Cybersecurity/Documents/Symantec_annual_internet_threat_report_ITU2015.pdf.

[2] Lavasoft, “Detecting Polymorphic Malware.” [Online]. Available: http://www.lavasoft.com/mylavasoft/securitycenter/whitepapers/detecting-polymorphic-malware. [Accessed: 01-Sep-2016].

[3] A. Sharma and S. K. Sahay, “Evolution and Detection of Polymorphic and Metamorphic Malwares: A Survey,” International Journal of Computer Applications, vol. 90, no. 2, pp. 7–11, 2014.

[4] S. K. Pandey and B. M. Mehtre, “A lifecycle based approach for malware analysis,” Proceedings - 2014 4th International Conference on Communication Systems and Network Technologies, CSNT 2014, pp. 767–771, 2014.

[5] Y. Prayudi and S. Yusirwan, “the Recognize of Malware Characteristics Through Static and Dynamic Analysis Approach As an Effort To Prevent Cybercrime Activities,” Journal of Theoretical and Applied Information Technology (JATIT), vol. 77, no. xx, pp. 438–445, 2015.

[6] M. Sikorski and A. Honig, Practical Malware analysis: The hands-on guide to dissecting malicious software. San Francisco: No Starch Press, Inc., 2012.

[7] M. Ahmadi, A. Sami, H. Rahimi, and B. Yadegari, “Malware detection by behavioural sequential patterns,” Computer Fraud & Security, vol. 2013, no. 8, pp. 11–19, 2013.

[8] S. Kumar, C. Rama Krishna, N. Aggarwal, R. Sehgal, and S. Chamotra, “Malicious data classification using structural information and behavioral specifications in executables,” 2014 Recent Advances in Engineering and Computational Sciences, RAECS 2014, pp. 1–6, 2014.

[9] S. Cesare, Y. Xiang, and W. Zhou, “Malwise-an effective and efficient classification system for packed and polymorphic malware,” IEEE Transactions on Computers, vol. 62, no. 6, pp. 1193–1206, 2013.

[10] D. Arshi and M. Singh, “Behavior Analysis of Malware Using Machine Learning,” in Contemporary Computing (IC3), 2015 Eighth International Conference on, 2015, pp. 481–486.

[11] G. Liang, J. Pang, and C. Dai, “A Behavior-Based Malware Variant Classification Technique,” International Journal of Information and Education Technology, vol. 6, no. 4, pp. 291–295, 2016.

[12] V. Naidu and A. Narayanan, “Needleman-Wunsch and Smith-Waterman Algorithms for Identifying Viral Polymorphic Malware Variants,” 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), no. August, pp. 326–333, 2016.

[13] M. Ahmadi, A. Sami, H. Rahimi, and B. Yadegari, “Malware detection by behavioural sequential patterns,” Computer Fraud and Security, vol. 2013, no. 8, pp. 11–19, 2013.

[14] P. M. Comar, L. Liu, S. Saha, P. N. Tan, and A. Nucci, “Combining supervised and unsupervised learning for zero-day malware detection,” Proceedings - IEEE INFOCOM, pp. 2022–2030, 2013.

[15] J. Park, S. Choi, and D. Y. Kim, “Malware Analysis and Classification: A Survey,” Lecture Notes in Electrical Engineering, vol. 215, no. April, pp. 449–457, 2013.

[16] L. Zeltser, “Malware sample sources for researchers.” [Online]. Available: https://zeltser.com/malware-sample-sources. [Accessed: 28-Feb-2016].

[17] V. Kumar and S. Minz, “Feature Selection: A literature Review,” Smart Computing Review, vol. 4, no. 3, pp. 211–229, 2014.

[18] A. Azab, R. Layton, M. Alazab, and J. Oliver, “Mining malware to detect variants,” Proceedings - 5th Cybercrime and Trustworthy Computing Conference, CTC 2014, pp. 44–53, 2015.