Analyzing Hardware Based Malware Detectors Using Machine Learning Techniques

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

by

Onkar Randive
Bachelor of Engineering
University of Pune, 2016

Director: Houman Homayoun, Professor
Electrical and Computer Engineering Department

Summer Semester 2018
George Mason University
Fairfax, VA

ACKNOWLEDGEMENTS

I would like to acknowledge the help of my Thesis Advisor Dr. Homayoun for giving me this opportunity to work on such an interesting topic. I would also like to thank him for his time and interest in this project and for his guidance in making this work a success. Dr. Rafatirad was extremely helpful in providing a new direction to the project and by providing timely assistance and guidance. Further, I would like to thank Hossein Sayadi, for helping me set clear goals and tasks which helped assure that the project deadlines are met. I couldn't ask for a better mentor than Hossein and I would like to thank him for his constant guidance and support. I would also like to acknowledge guidance and assistance of Dr. Sai Manoj P. D, Nisarg Patel and fellow colleagues at Dr.Homayoun's research lab at George Mason University.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Operating Systems ..... OS
Hardware Performance Counters ..... HPC
Machine Learning ..... ML
Field Programmable Gate Arrays ..... FPGA
Decline of Service ..... DoS
Internet of Things ..... IoT
Anti-Virus ..... AV
Linux Containers LXC
Waikato Environment for Knowledge Analysis ..... WEKA
Multinomial Logistic Regression ..... MLR
Multilayer Perceptron ..... MLP
Principal Component Analysis ..... PCA
Comma Separate Values ..... CSV
Support Vector Machine ..... SVM
Anti-virus ..... AV

ABSTRACT

ANALYZING HARDWARE BASED MALWARE DETECTORS USING MACHINE LEARNING TECHNIQUES

Onkar Randive, M.S.

George Mason University, 2018

Thesis Director: Dr. Houman Homayoun

Growth of malware has been a serious problem in the technology community and would continue to grow with new advances in technology. Traditional software-based malware detection systems have proved to be inadequate. Behavioral malware detection systems have proved to be an improvement but are limited due to the fact that they are resource intensive and still prone to exploitation. Hardware based malware detection has proved to be an effective answer to reduce exploitability of computer systems due to less visibility and access for exploitation.

This work shows the results of using different machine learning classifiers for Hardware based malware detection. Further, it analyzes the hardware implementation of these machine classifiers on an FPGA by discussing the latency and area requirements of the machine learning classifiers. It is proposed that classifiers with less or limited number of features for reduced system overhead especially in resource constrained environments like real time systems or embedded systems. Hence, the number of features fed to the

classifier are reduced by using feature selection technique called Principal Component Analysis. Simple ML Classifiers like JRIP and OneR prove to be more efficient than complex ML classifiers like neural networks. The results for multiclass classification using Multinomial logistic regression, Multilayer Perceptron and Support Vector machines show that PCA assisted multiclass classifies prove to be 7% more efficient than regular Multiclass classifiers. Further the limitations of the above work are discussed by explaining solution leading to future works in this topic.

INTRODUCTION

Development in technology leads to a proliferation of new computing devices along with a rapid growth of malicious software. There are significant efforts in creating more efficient and dangerous attacks by devising new and innovative ways to infiltrate or destroy a computing system. We call this malicious software as malware. Malware is a piece of malicious software which can be used to take control of a system or leak data outside of a secure system etc. McAfee threats report [12] explains that there are 75 million malware samples in 2017 and hundreds of new malware samples are added by the minute. As human civilization gets more fused with technology and computing devices, a greater need to defend against these attacks arises and more robust and efficient Anti-malware systems are required.

Traditional software-based malware detection systems (Anti-virus) have had significant limitations to fight the attacks of modern day high technology malware attacks. In the beginning when malware attacks were new there were signature-based AV systems which kept a huge library of malware signatures in its database. These AV systems tend to need huge amounts of memory to store these signatures along with access to the internet as the signature database has to be updated regularly. Due to the above limitations, behavior based AV systems came to prominence because of their ability to maybe detect zero-day attacks which the signature based AV systems failed to detect. Behavior based AV systems

use the behavior of code to detect a malware program. However, malware creators then started to write bad harmful code to be disguised as good code or harmful code inside a benign program. These continues till this day and even though behavior-based AV systems are getting secure by the day, the similar increase in the quantity and quality of malware. Due to problems like resource intensive and possibility of a loophole in the AV system makes behavioral based systems insecure or incapable of detecting malware attacks in the current scenario.

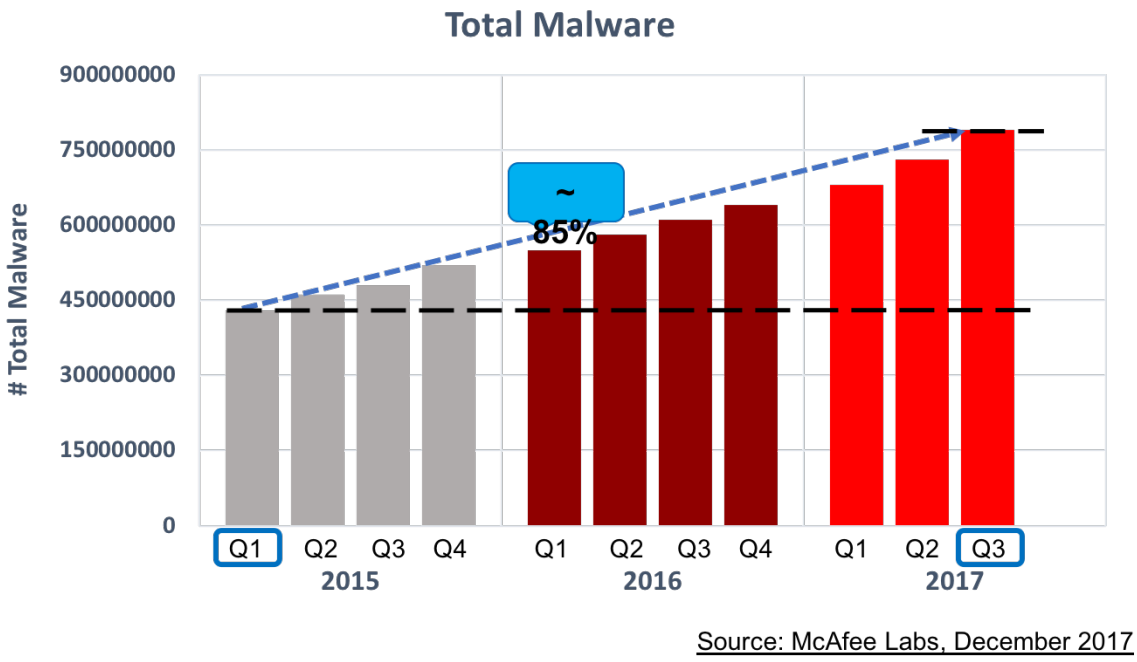
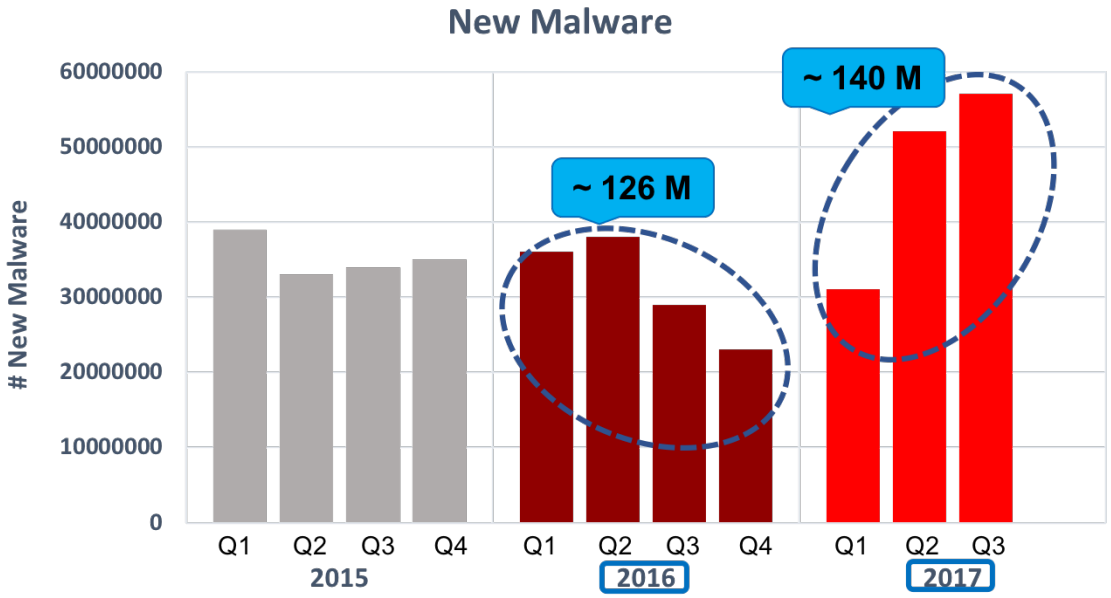


Figure 1: Total Malware samples recorded around the world

The above limitations and drawbacks of software- based malware detection systems have led security researchers to look for new and innovative solutions to win the malware war. One such effective solution is using hardware-based features, the ones which the attacker cannot have access to due to physical limitations rather software or technology limitations. Hardware based malware detectors use some or the other sort of hardware features like microarchitectural events, power usage, memory tracking etc. to determine malicious programs from benign programs.

Using micro-architectural events from the chip has increasingly been studied and researched for being a good solution to the malware race. Hardware Performance Counters (HPC) is a common term in the technology community for using microarchitectural events at user level.



Source: McAfee Labs, December 2017

Figure 2 : Growth of Malware

Hardware Performance Counters

Hardware Performance Counters (HPC) were first introduced for easier profiling and debugging of the system. They are special registers in microprocessors and can be read from the Performance Monitoring Unit available on the chip. HPC were very easily incorporated into modern chips and were soon found to be used for various purposes like performance prediction, power prediction and energy efficiency prediction.

Hardware Performance Counters are used to study the behavior of applications and hence can be used to determine if a given program is a malware or a benign application. Malware show a different behavior than benign programs in terms of the count or values of HPC. Thus, we study these changes or variations or patterns in HPC values using machine learning techniques to predict the nature of the application. Using HPC events has the added advantage of being faster and having less system overhead. But, there is also a limit as to how fast you can read the values from the registers. There is also the limitation of having a specific or few number of registers available on the chip and users might need multiple cycles to extract HPC data as the large number of HPC are multiplexed onto the few available registers.

Types of Malware

Around the world, malware can be divided into various classes depicting the behavior of the type of malware. Of all the total samples of malware available in the world on the internet, around 70% happen to be of the class Trojan. Also, it should be noted that classifying malware into different malware classes may always not be exclusive or

comprehensive. The following pie-diagram displays the distribution of different malware classes present on the internet. We have used five different classes of malware for analysis and for detection and classification. Following are the five malware classes analyzed.

Behavior of different classes of malware:

1) Backdoor

Backdoor are malware that allow access to a computer system by bypassing the normal authentication processes. It essentially provides a backdoor to the host system which then can be used to restrict access to authentic users and further secure remote access of the host system by the attackers or non-authentic users.

e.g. a special username and password hard-coded into the login program.

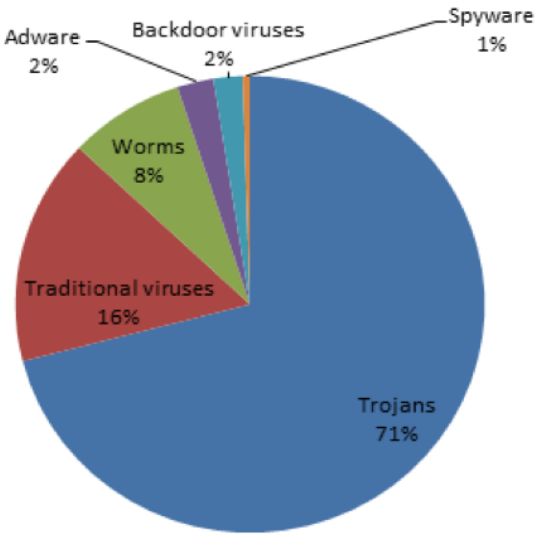


Figure 3 : Malware Distribution (on the internet) into different classes. Source [9]

2) Rootkits

Rootkits are the set of tools deployed immediately after system penetration. Rootkits are used for the following purposes:

a) Maintain access secured via backdoor – local and remote

b) Attack other systems – DoS, Sniffing, Scanning

c) Destroy Evidence – Clear audit trails, prevent audit collection.

d) Avoid being removed by routine maintenance of the system

3) Trojan

Trojans are authentic or genuine looking malware which are used to transport or propagate a virus, worm or install a backdoor onto a computer system. Trojans can also be used to erase or overwrite data present on a computer. Trojans are generally used for phishing bank account details, credit card numbers or key loggers.

4) Virus

Virus and worms are the type of malware which have the ability to duplicate by themselves. Virus can be embedded in a file or a program. Viruses are very specific to an operating system and take advantage of its details and weaknesses. They can infect executable files, disks, email attachments.

5) Worm

Worm are the most dangerous types of malware. Worms are dangerous because of the properties of self-replication, self-contained and they spread autonomously and exponentially. Worms require no human intervention to propagate and spread. Worms consume memory and overload servers and propagate to other servers as they are independent of the Operating System.

RELATED WORK

Significant work has been done in laying the ground work for Hardware based malware detection by Demme et al [3]. Demme et al [3] first showed that a given application can be classified as benign or malware using micro-architectural events. It also showed that the behavior of application can be studied using Hardware Performance counters using offline machine learning techniques. It deeply discussed the feasibility of this technique by going through the limitations and challenges to make the work online and real time. However, all this work was largely limited to Supervised machine learning techniques. Tang et al [15] worked on trying to make the system more robust by removing this limitation and introduced unsupervised machine learning techniques for Hardware based malware classification.

Further work was done by Bahadur et al [2] where they used feature selection by single value decomposition to select the best features to study behavior of the given application. Garcia [5] has discussed feasibility of anomaly detection using unsupervised machine learning techniques. Patel N. et al [4] gives a thorough analysis of using machine learning methods for classifying an application as malware or benign. It also discusses the feasibility of machine learning classifiers when implemented in hardware with respect area, power and latency. Ozsoy et al [13] have discussed the use of sub-semantic features for online detection of malware where they introduced a two-line defense – software and hardware. Further works have also used two level techniques or systems to detect and secure systems from malware attacks. Khasawney et al [11] first introduced the technique

of ensemble learning on machine learning techniques for Hardware based malware detection. H Sayadi et al [16,18] further improves on the techniques of ensemble learning techniques for Hardware based malware detection by analyzing general and ensemble techniques. It also provides a comprehensive analysis of ensemble learning for effective run-time Hardware based malware detection. H. Sayadi et al [16, 18] also uses feature selection techniques for feature reduction to select the top features to be fed to the ensemble classifiers. However, the work in this thesis is largely inspired from the works of N. Patel et al [4] and H. Sayadi et al [16,18] and improves data collection and upon the concept of feature selection by adding it to multiclass classification, which hasn't been done by in any of the prior works.

IMPLEMENTATION

Experimental Setup

All the experiments are conducted on an Intel Haswell Core i5 – 4590 CPU installed with Ubuntu 14.04 with Linux 4.4 kernel. More than 86 microarchitectural events are supported by the Haswell Core i5-4590 among which 52 are hardware events and the rest are Software events. Software events are used for System profiling and tracing. The Haswell Core i5-4590 has 8 performance monitoring registers which actually record the values of hardware events. The 52 hardware events are multiplexed onto these 8 registers. WEKA tool [7] is used to train and test the Machine Learning classifiers and the Xilinx Vivado High Level Synthesis tool is used for hardware implantation of the machine learning classifiers.

Overview

The Figure 4: Experimental Setup-Overview shows a very general overview of the project. First the malware samples are downloaded from the internet from popular malware honey-pots and sites. Then tools available on Virustotal.com [1] are used to classify the malware into separate classes for having a labelled database. We use Perf tool to read and record the values of Hardware Performance Counters. Following is an overview of the complete experimental setup. LXC [8] Containers are used to provide an isolated environment for malware execution by providing operating system level virtualization. The

values from the Perf tool are stored into text files and later combined into a CSV file to be used as input to Machine Learning Classifiers implemented in WEKA [7].

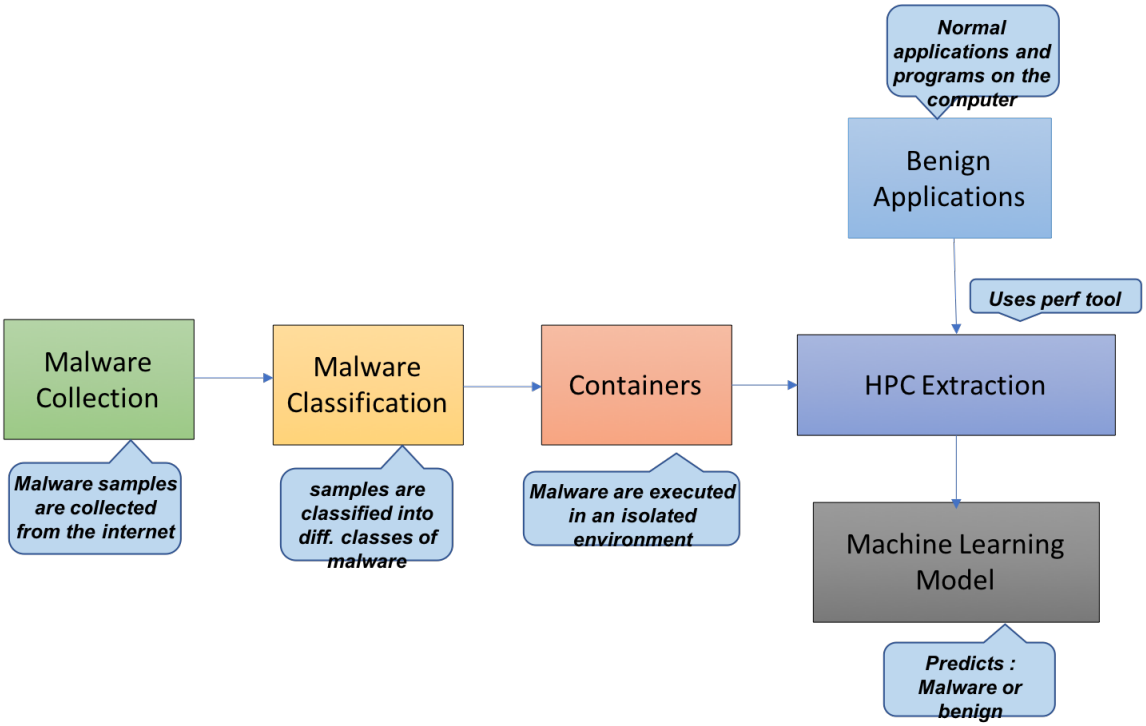


Figure 4: Experimental Setup-Overview

The Figure 5: Detailed Overview of the data collection process gives a better understanding of the underlying workings of the system. Perf tools present in the Linux kernel are used to read the values of the HPC from the Performance Monitoring Unit. The Performance Monitoring Unit is used to take the count the values from the registers present on the chip. HPC are read at the sampling period of 10ms. Containers are the isolated systems where the malware is executed so that the malware does not infect the host system

and the noise from the execution of regular program does not create a bias in the measured values of HPC.

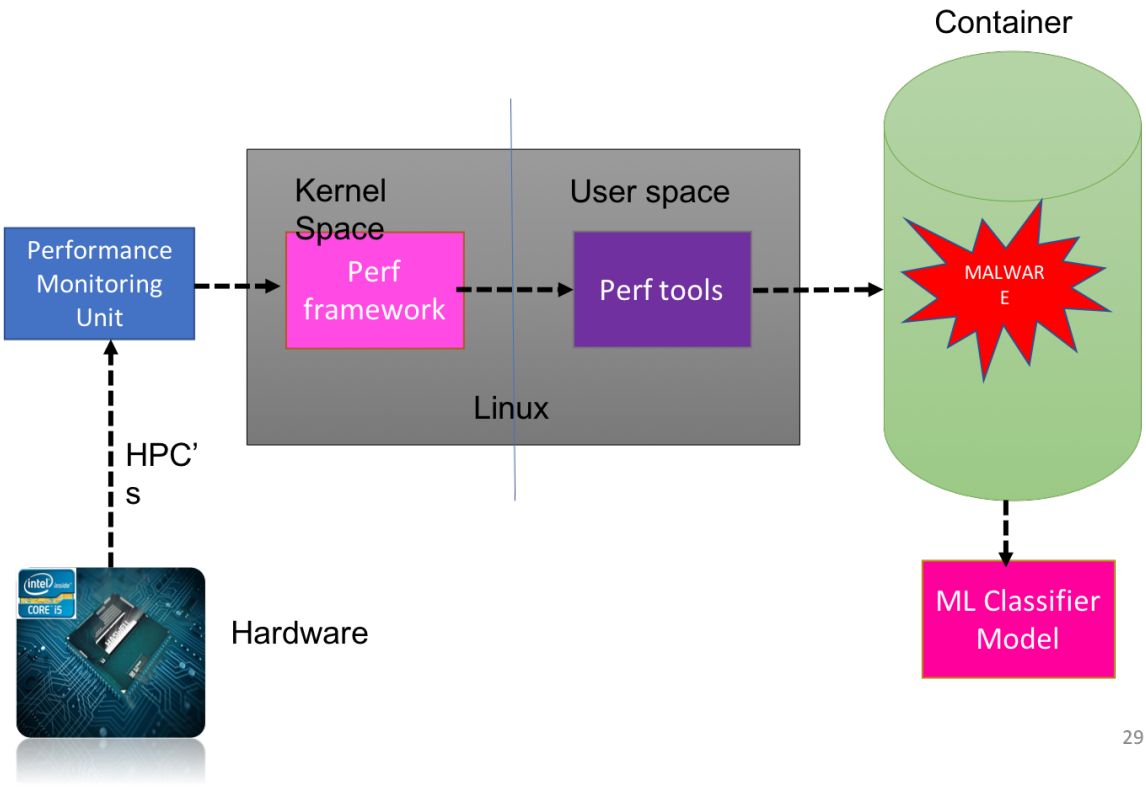


Figure 5: Detailed Overview of the Data Collection Process

Malware Database

More than 3000 samples of malware classified into 5 separate classes constitute the total number of malware samples in the experiments done in this work. All the malware samples are downloaded from the online malware directory – virusshare.com [24]. Further,

these samples are uploaded to virustotal.com [1] to determine their class and to make a labelled database which can also be used for malware classification. The malware database used for experiments roughly reflects the distribution of malware present on the internet around the world. It can be seen by looking at the Figure 3 : Malware Distribution (on the internet) into different classes. Source [9] and Figure 6: Distribution of Malware(used) into classes that the malware database used in the experiments are similar in distribution to the malware samples found around the world.

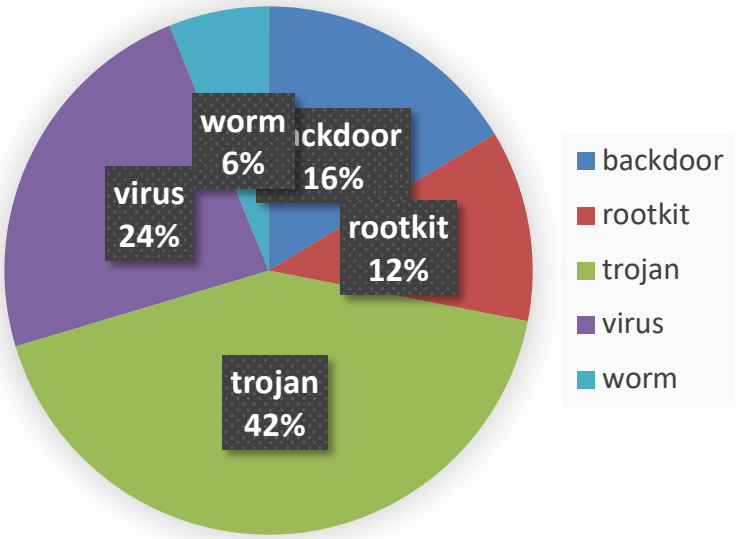


Figure 6: Distribution of Malware(used) into classes

Application	Class	Samples
Malware	Backdoor	452
	Rootkit	324
	Trojan	1,169
	Virus	650
	Worm	149
Benign	Inbuilt or installed Programs	326
		Total = 3,070

Table 1 : Number of samples of different application

Training and Testing Malware Classifiers

The HPC values obtained from the Perf tool are stored in individual text files for each malware sample at real time. The data from all of these text files is later copied combined into a CSV file. Separate CSV files are created for testing and training datasets. Each row represents data extracted at 10ms sampling period. There are around 50,000 such rows in total. Each sample contains17 attributes columns :16 Performance Counters + 17th column (class) represents the application as benign/malware.

Corresponding CSV files are converted to ARFF files for easier implementation of Machine Learning models in WEKA. For certain classifiers the class column is changed to numerical (0/1) for some classifiers in WEKA. The total database is divided into training dataset and testing dataset in the ratio 70% to 30% i.e 70% of the data is used to train the classifiers and the rest of the 30% of the samples are used to test the trained classifiers. There are different ways a classifier can be tested – self-testing, test-set or cross validation. A separate test dataset has been provided as test-set for this work. The Figure 7: Training

and Testing ML Classifiers shows how the HPC data from perf tools is used to train and test ML Classifiers.

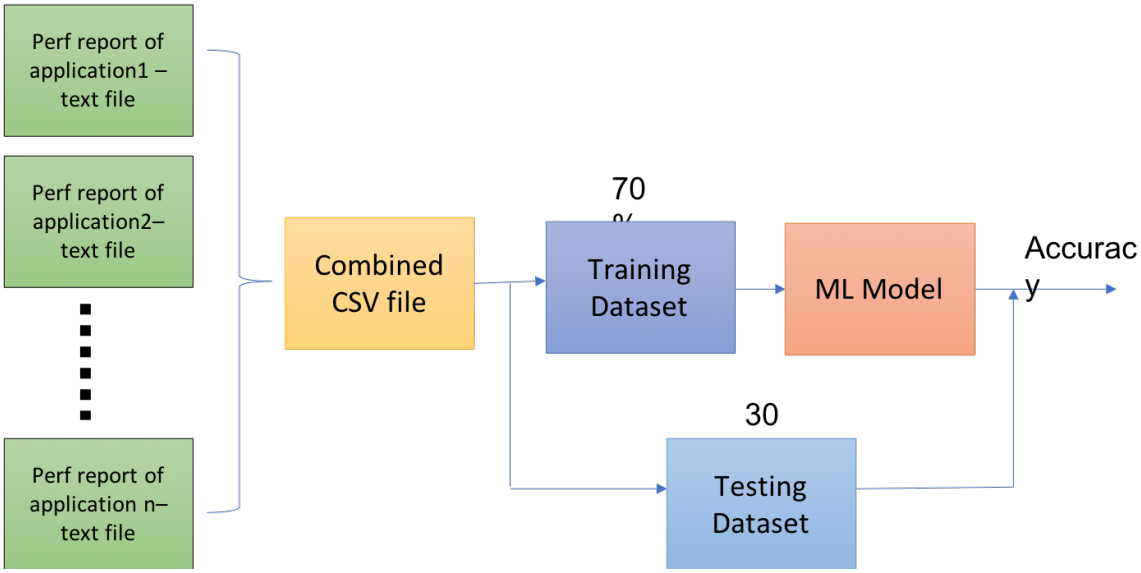


Figure 7: Training and Testing ML Classifiers

FEATURE REDUCTION

In Statistics and Machine Learning, there are usually a lot many factors on which a decision is made. We call these factors as features in Machine Learning which are basically variables based on which the decision is made regarding the output of the classifier. Sometimes, the higher the number of features the more complex the Classifier gets which makes the system using the classifier more complex. Also, most of the times these features are correlated and sometimes some of them are redundant. This is the most important application of Feature Reduction. Feature Reduction is the process of reducing the number of features by selecting the features which have a larger impact on the decision of the ML Classifier. Feature Reduction can be done using two ways – Feature Selection or Feature Extraction. We are going to use one the feature selection technique called as Principal Component Analysis also known as PCA.

Principal Component Analysis

In-order to select the best features which highly correlate to the behavior of the application and decides whether this application is benign application or malware we use feature selection and feature reduction techniques like Principal Components Analysis. Principal Component Analysis converts non-linear multidimensional data into relational linear data. WEKA is used to perform Principal Component Analysis on the Hardware Performance Counters dataset.

WEKA provides the list of principal components by providing a ranking of eigen vectors which specify the relationship between the features. After getting the linear

relationship and the eigen vectors for each of the 16 attributes, we can rank the attributes to get the ranking with respect to eigen vectors. After the attributes are ranked, a PCA plot is plotted with the top 2 PCA eigen vectors. This is actually not pure PCA but a combination of PCA and Clustering technique.

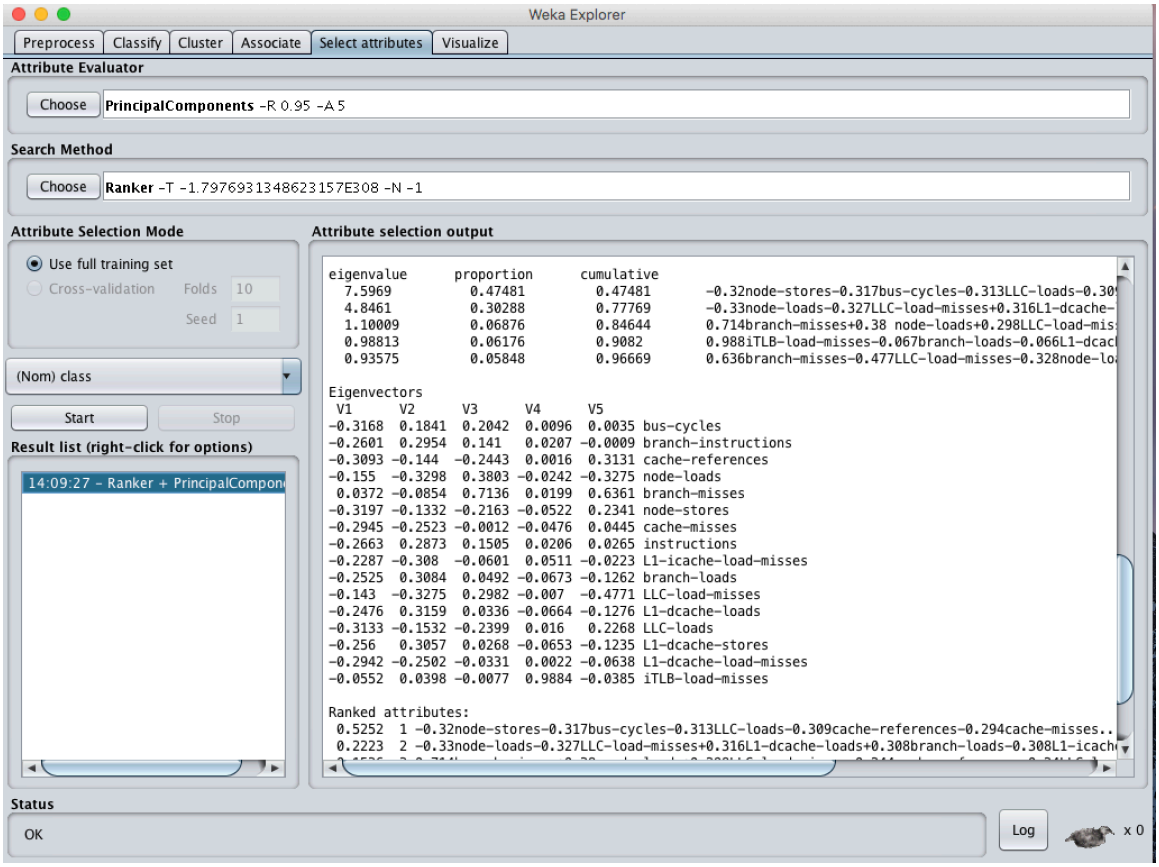


Figure 8: Eigen vectors from WEKA

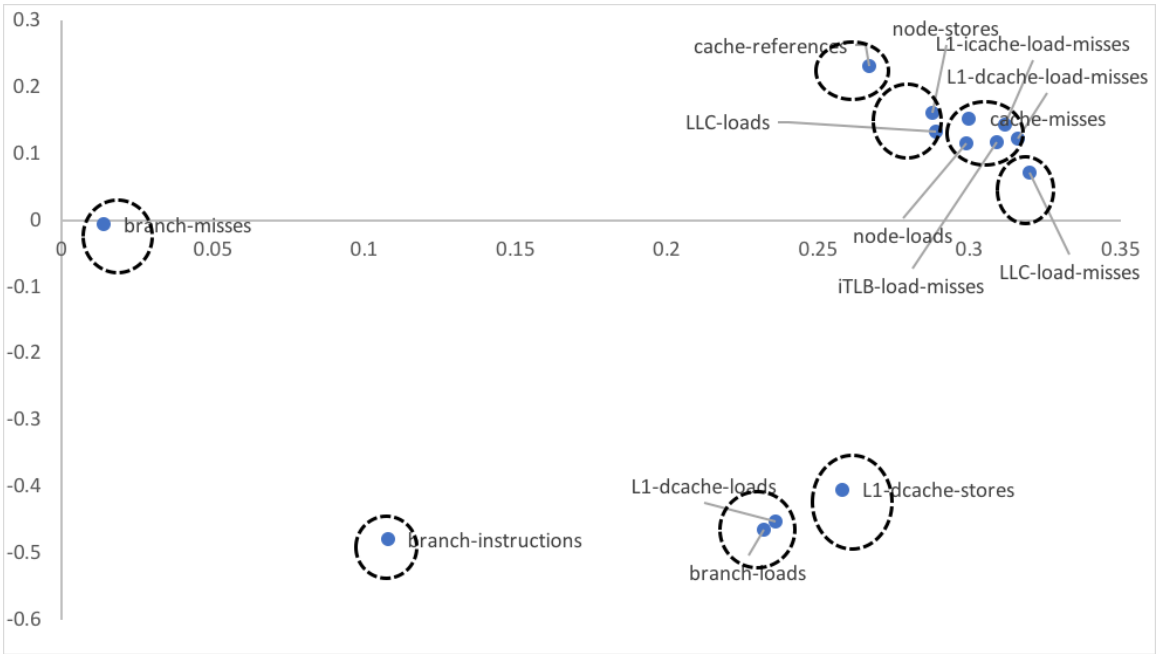


Figure 9: PCA Plot for Rootkit

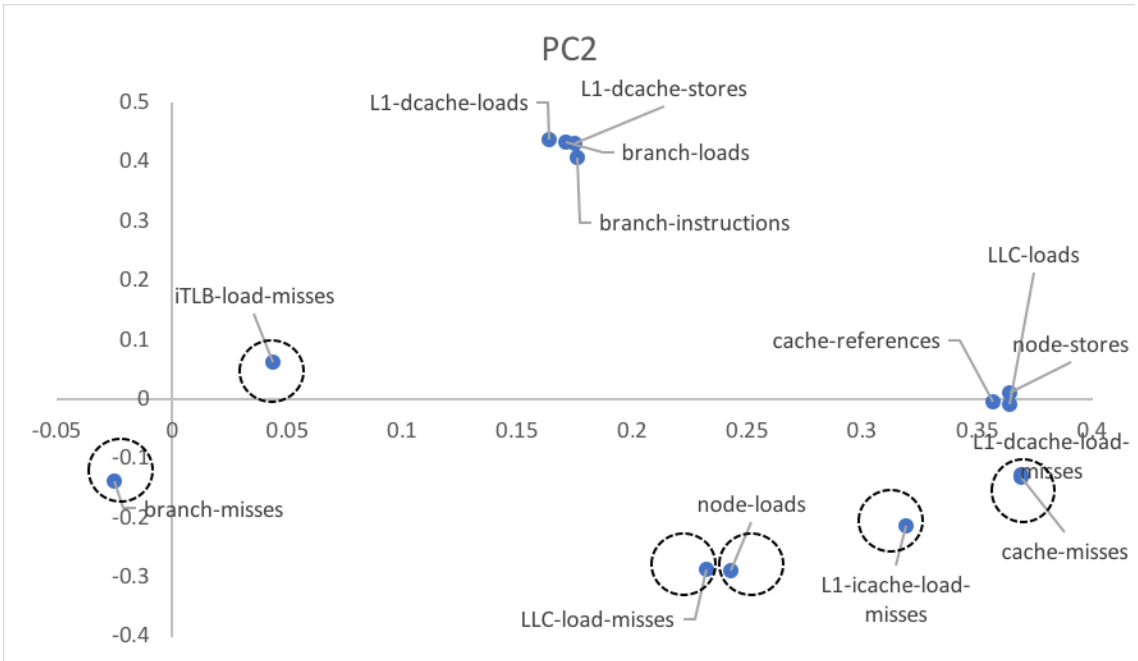


Figure 10: PCA Plot for Trojan

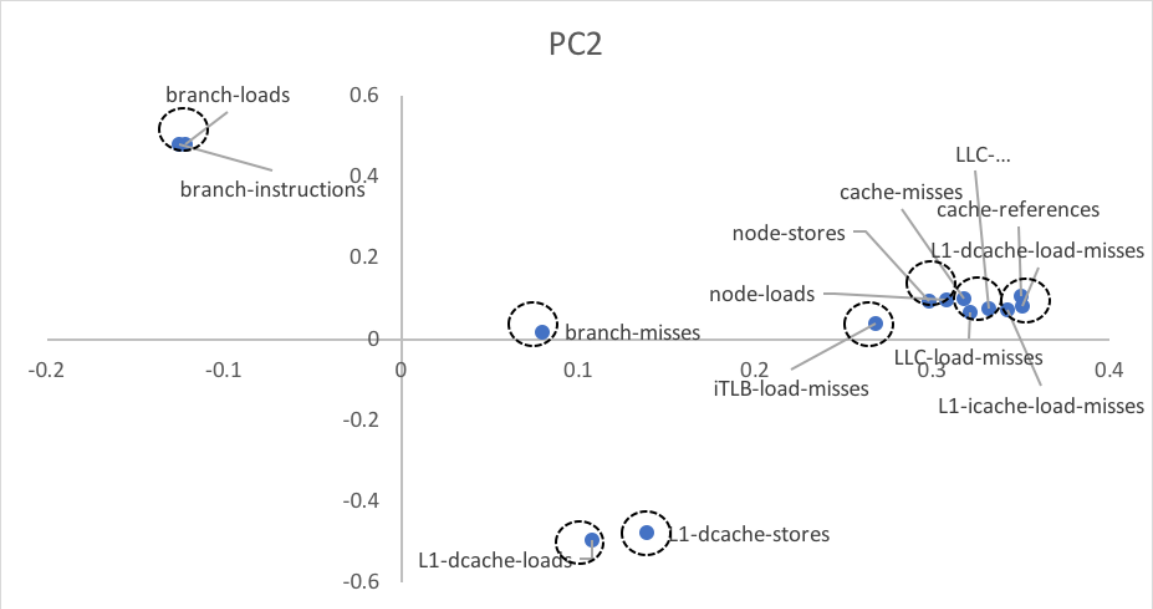


Figure 11: PCA Plot for Virus

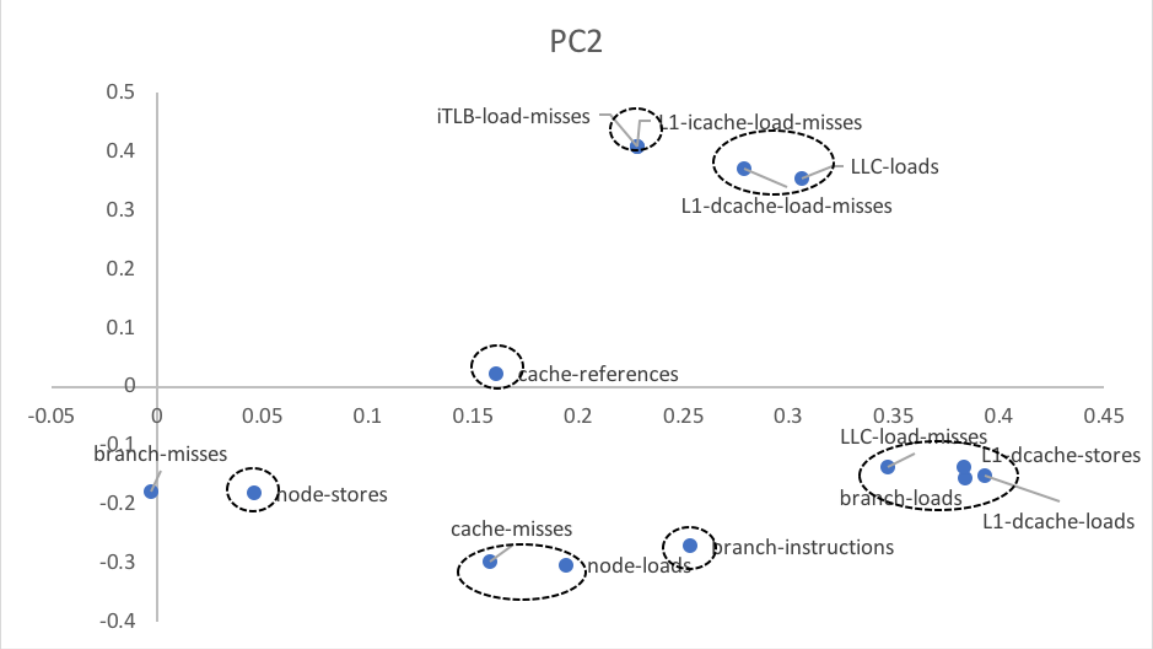


Figure 12: PCA Plot for worm

RESULTS

Accuracy is the percentage of the samples that the ML classifier correctly classifies. **Error! Reference source not found.** depicts the reduced features that PCA has analyzed. It shows 4 features that are common to all the classes. Hence, these are called as common features. Each class of malware has a custom set of 8 features which consist the 8 principal components for that class of malware.

- Reduced features per class of malware

Backdoor	Trojan	Virus	Rootkit	Worm
branch-instr.	branch-instr.	branch-instr.	branch-instr.	branch-instr.
cache-references	cache-references	cache-references	cache-references	cache-references
branch-misses	branch-misses	branch-misses	branch-misses	branch-misses
node-stores	node-stores	node-stores	node-stores	node-stores
branch-loads	cache-misses	LLC-load-misses	cache-misses	cache-misses
L1-icache-load-misses	L1-icache-load-misses	L1-dcache-loads	branch-loads	L1-dcache-loads
LLC-load-misses	LLC-load-misses	L1-dcache-stores	LLC-load-misses	L1-dcache-load-misses
iTLB-load-misses	iTLB-load-misses	iTLB-load-misses	L1-dcache-stores	iTLB-load-misses

Table 2: Reduced features from PCA

For Binary classification, Figure 13: Accuracy Comparison compares the accuracy for different classifiers for 8 features and 4 features respectively and depict that most of the

classifiers have reduced accuracy for reduced number of features. However, some classifiers like J48, OneR show infinitesimal or negligent reduction in accuracy.

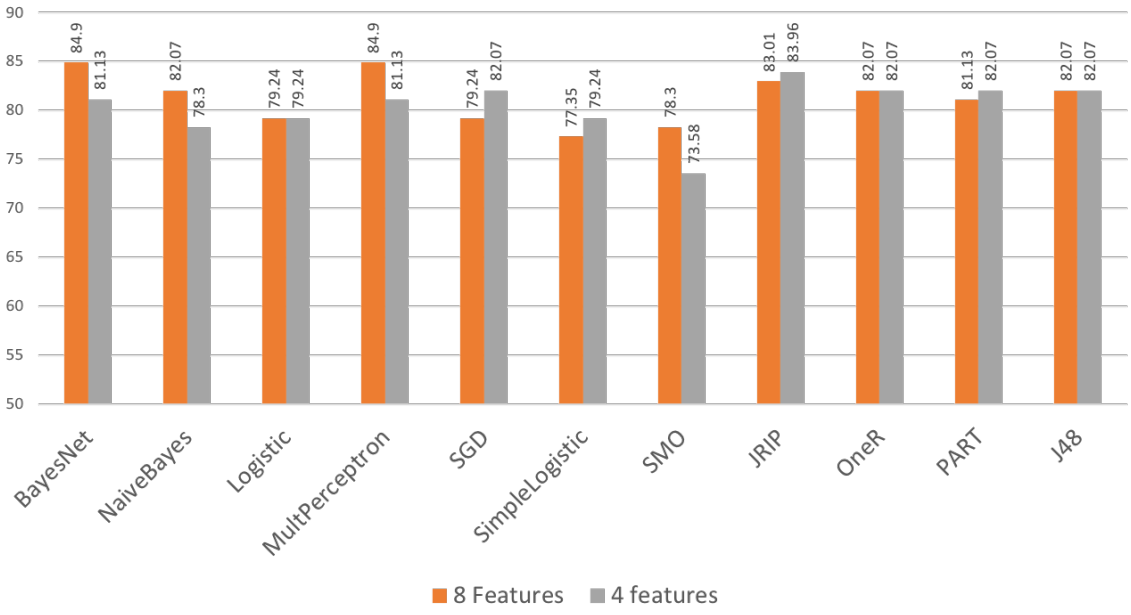


Figure 13: Accuracy Comparison for Binary Comparison

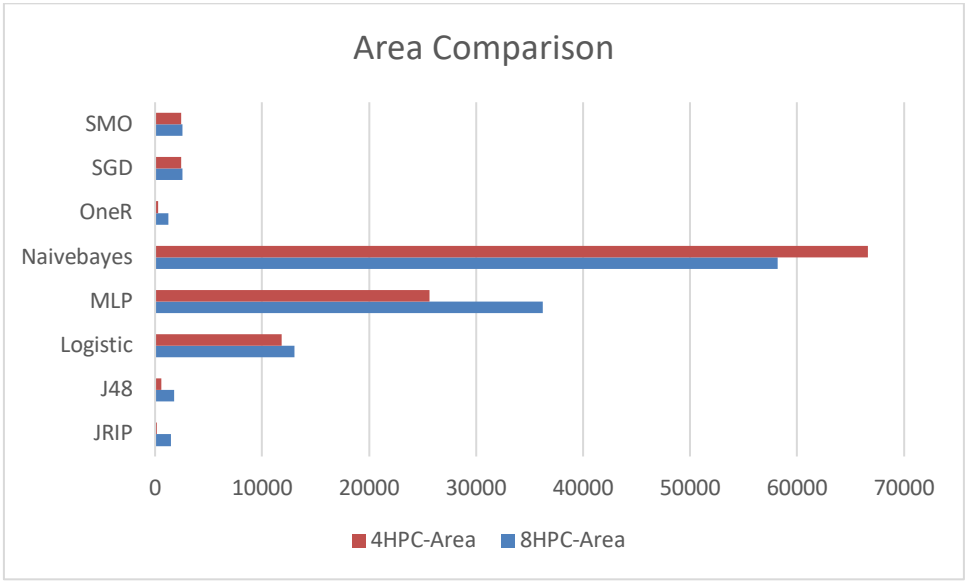


Figure 14: Area Comparison

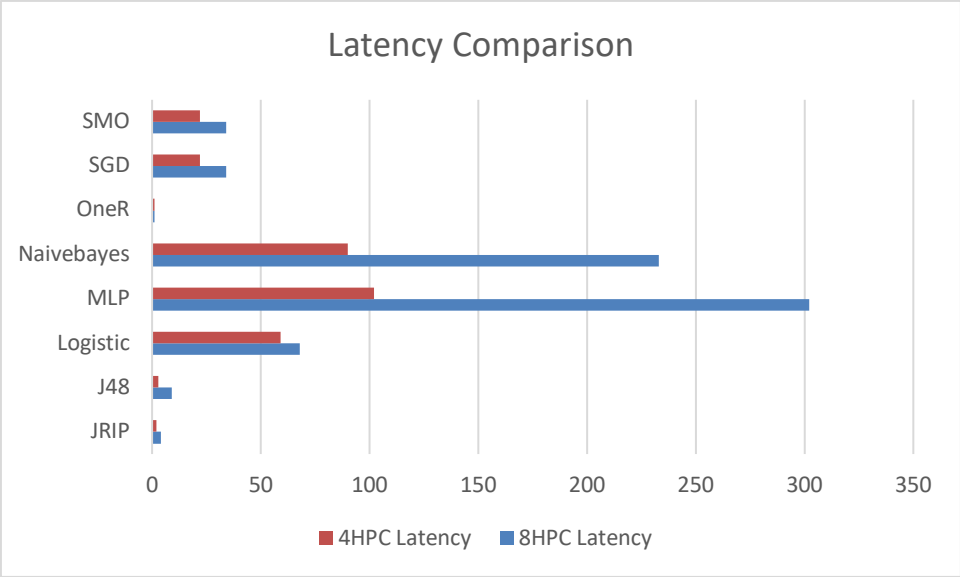


Figure 15: Latency Comparison

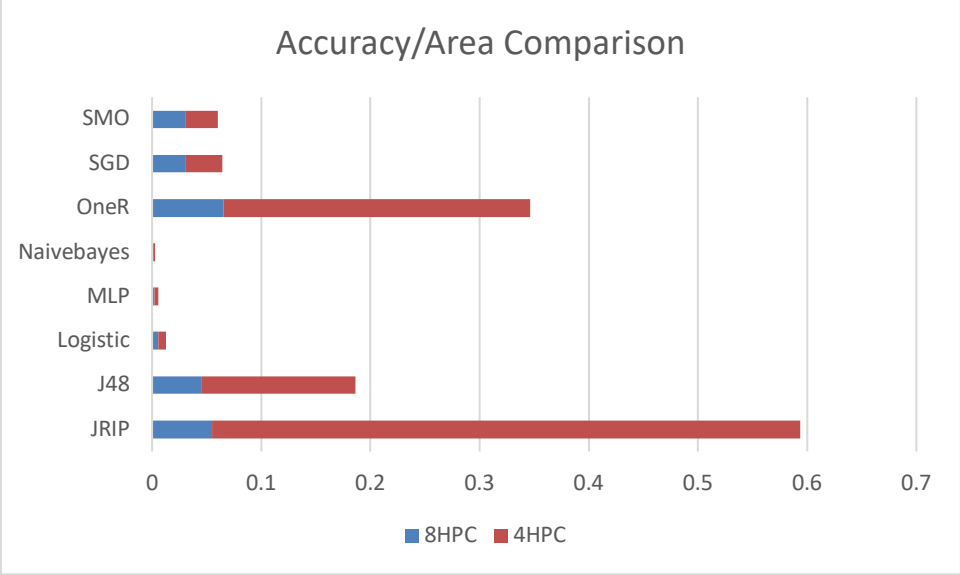


Figure 16: Accuracy/Area Comparison

However, reducing the number of features has significant effect on the latency, area and power required by the ML Classifier. ML Classifiers with the least footprint and which consumes the least power proves much more efficient than the one with higher accuracy for applications such as embedded systems and other real time systems like Internet of Things. Figure 14: Area Comparison and Figure 15: Latency Comparison display the comparison of area and latency comparison obtained from hardware implementation of ML Classifiers. ML classifiers like JRIP and OneR have far better Accuracy/Area ratio after implementing PCA as shown in Figure 16: Accuracy/Area Comparison. Further, it can be observed from the results comparing accuracy for different Multiclass classifiers in Figure 17: Average Accuracy for Multiclass Classification that neural networks like MLP have better accuracies for classifying classes of malware. Figure 18: Per-Class accuracy for different ML Classifiers shows the accuracy for different classes for the various Multiclass classifiers like MLP, MLR, and SVM. Finally, an increase in accuracy of around 7% for Multiclass classifiers can be observed when the accuracy of the ML classifier with PCA 8 custom features are compared to the average accuracy of the non-custom features.

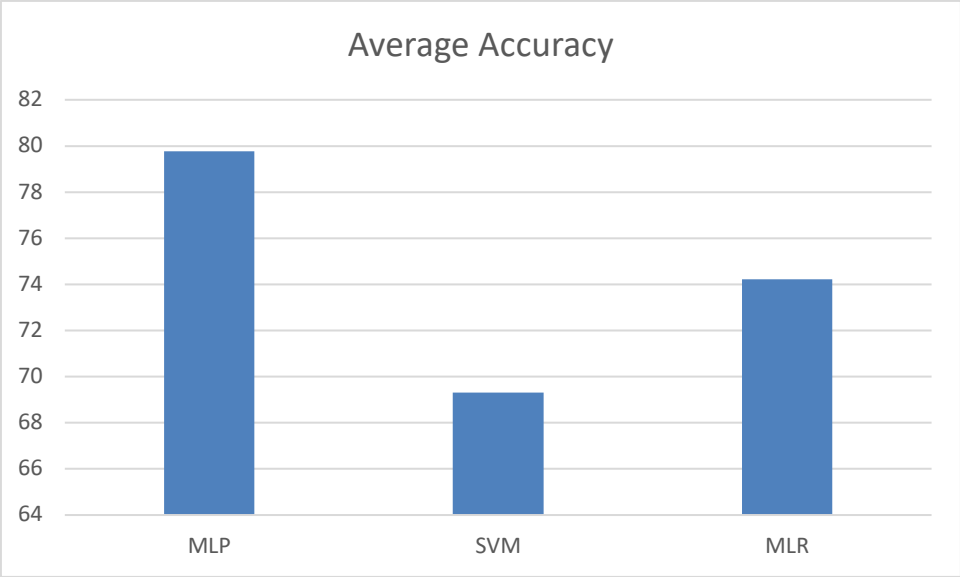


Figure 17: Average Accuracy for Multiclass Classification

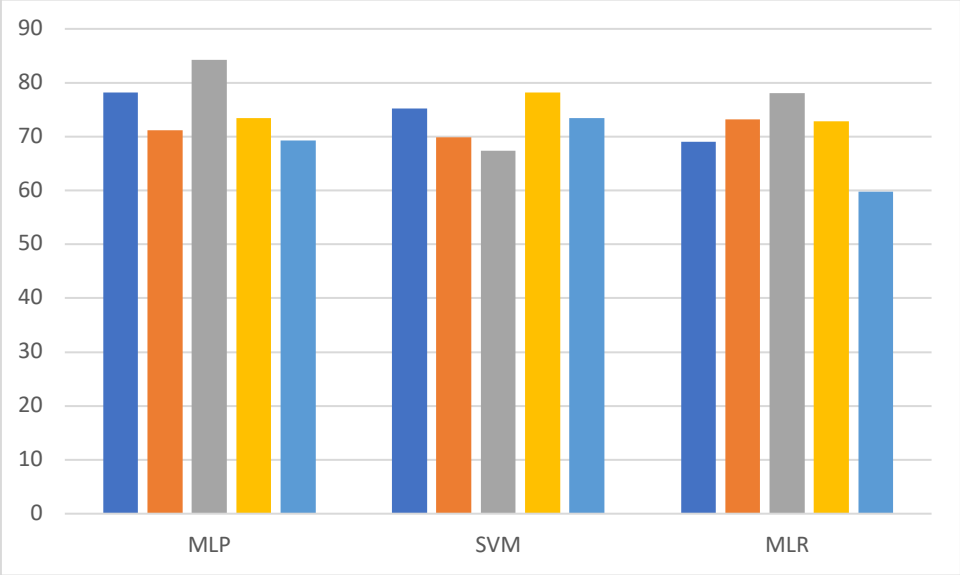


Figure 18: Per-Class accuracy for different ML Classifiers

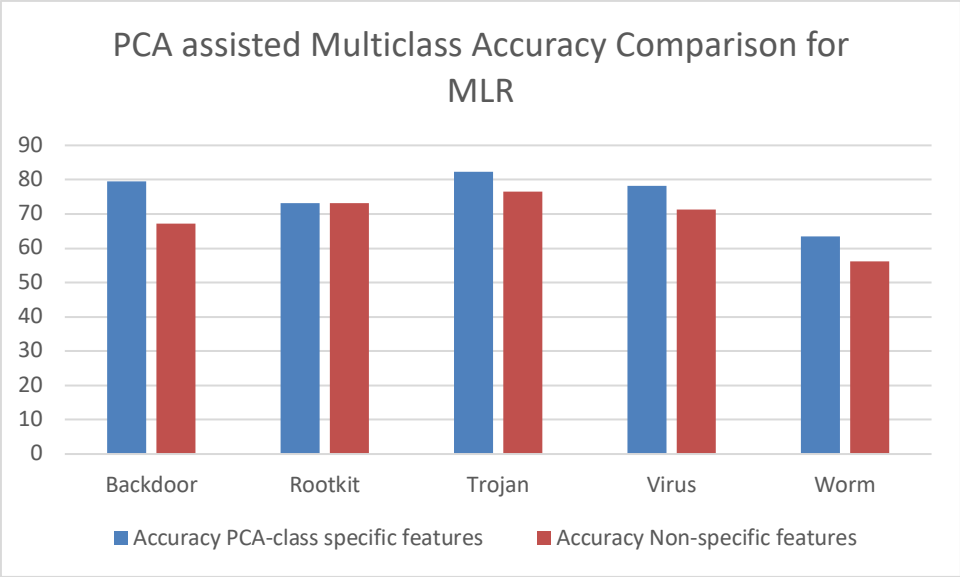


Figure 19: PCA-assisted MLR vs normal MLR

FUTURE WORK

There are a lot of directions that future works in this project can be applied. Some limitations of current work can be eliminated by using different techniques for data collection or different classification techniques. Furthermore, more advanced features can be added to make the current system more efficient and deployable in real time systems or embedded systems.

1. **Better Data Collection:** Currently, the Hardware Performance Collection is limited and can be extended to include other Operating systems other than the Linux OS like Windows. Further, software tools providing more accurate and less granular HPC data can be used.

2. **Alternatives to Machine Learning Techniques for Classification:** There are various limitations of Machine Learning Techniques with regard to malware detection and run-time malware analysis like huge training period, need of large datasets etc. These limitations can be removed by using better and more efficient statistical techniques and methods of classification.

3. **Malware Detection in Resource Constrained Environments:** Different strategies can be deployed to make the system more efficient and feasible to use with embedded devices or IoT devices or similar resource constrained

environments. Work may be done in reducing latency in the process of data collection and by deploying faster classification algorithms.

CONCLUSION

I would like to conclude that this works on Hardware based malware detection analyses different techniques used in hardware-based malware detection like different classes and samples of malware, different detection techniques. It further takes the malware detection system and improves it by adding additional feature like feature reduction and multiclass malware classification. Simple malware classifiers like JRIP and OneR prove to be more efficient in terms of Accuracy/Area as compared to complex ML classifiers like neural networks. We discussed the need of feature reduction for real time systems like embedded systems and reduce the number of feature fed to ML classifiers using feature selection techniques like Principal Component Analysis. The results for multiclass classification using Multinomial logistic regression, Multilayer Perceptron and Support Vector machines point an increase in accuracy of ML classifiers of around 7% if assisted by feature reduction techniques like PCA. Further, limitations like limited database, efficiency of ML techniques in resource constrained environments paves way future works in this topic to make the current system more secure and robust.

.

APPENDIX

1) Intel® 64 and IA-32 Architectures Software Developer's Manual
https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-instruction-set-reference-manual-325383.pdf

REFERENCES

[1] Virus total intelligence service. http://www.virustotal.com/intelligence Accessed: Sept 2017.

[2] Bahador, M.B., Abadi, M., Tajoddin, A., “HPCMalHunter: Behavioral malware detection using hardware performance counters and singular value decomposition”, In IEEE International Conference on Computer and Knowledge Engineering (ICCKE'14), pp. 703-708, October 2014

[3] Demme, J., Maycock, M., Schmitz, J., Tang, A., Waksman, A., Sethumadhavan, S., Stolfo, S., "On the feasibility of online malware detection with performance counters", In Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA ’13), pp. 559–570, 2013

[4] Patel, N., Sasan, A., Homayoun, H., “Analyzing hardware-based malware detectors”, In Proceedings of the 54th Annual Design Automation Conference(DAC’17), Article No.25, June 2017.

[5] Garcia-Serrano, A., “Anomaly detection for malware identification using hardware performance counters”, 2015.

[6] Guthaus, M. R., Ringenberg, J. S., Ernst, D., Austin, T. M., Mudge, T., Brown, R.B., "MiBench : A free, commercially representative embedded benchmark suite", in Proceedings of the Workload Characterization, 2001 IEEE International Workshop, December 2001

[7] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H., "The WEKA data mining software: an update", ACM SIGKDD Explorations Newsletter, June 2009

[8] Helsley, M., “Lxc: Linux container tools". IBM Developer Works Technical Library, 2009.

[9] https://www.bullguard.com/bullguard-security-center/pc-security/computer-threats/malware-definition,-history-and-classification.aspx

[10] H. Sayadi, “A Comparative Study of Machine Leaning Approaches for Malware Detection using Microarchitectural Features” in arxiv preprint, 2018.

[11] Khasawneh, K. N., Ozsoy, M., Donovick, C., Abu-Ghazaleh, N., Ponomarev, D., "Ensemble Learning for Low-Level Hardware-Supported Malware Detection", in Proceedings of the 18th International Symposium on Research in Attacks, Intrusions, and Defenses, 2015.

[12] McAfee Labs. "Infographic: Mcafee labs threats report", March 2017.

[13] Ozsoy, M., Donovick, C., Abu-Ghazaleh, N., Ponomarev, D., “Malware-aware processors: A framework for efficient online malware detection”, In HPCA’15, 2015.

[14] Ozsoy, M., Khasawneh, K. N., Donovick, C., Gorelik, I., Abu-Ghazaleh, N., Ponomarev, D., “Hardware-based malware detection using low-level architectural features”, In IEEE TC, Vol. 65, No. 11, November 2016.

[15] Tang, A., Sethumadhavan, S., and Stolfo, S., “Unsupervised anomaly-based malware detection using hardware features,” in Proceedings of the 17th Int. Symposium on Research in Attacks, Intrusions and Defenses, pp. 109–129, 2014.

[16] H. Sayadi, et al., “Ensemble learning for effective run-time hardware-based malware detection: a comprehensive analysis and classification,” In Proceedings of the 55th Annual Design Automation Conference. June 2018.

[17] H. Sayadi, D. Pathak, I. Savidis, and H. Homayoun, “Power conversion efficiency-aware mapping of multithreaded applications on heterogeneous architectures: A comprehensive parameter tuning,” In 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), (pp. 70-75), January 2018.

[18] H. Sayadi, et al., “Comprehensive assessment of run-time hardware-supported malware detection using general and ensemble learning,” In ACM International Conference on Computing Frontiers (CF), 2018.

[19] H. Sayadi et al., “Customized machine learning-based hardware-assisted malware detection in embedded devices,” In 17th IEEE International Conference on Trust, Security And Privacy In Computing And Communications (IEEE TrustCom-18), 2018.

[21] H. Sayadi, and H. Homayoun, “Scheduling multithreaded applications onto heterogeneous composite cores architecture,” In Green and Sustainable Computing Conference (IGSC), (pp. 1-8). October 2017.

[22] H. Sayadi, "Evaluation of Machine Learning Techniques for Energy-Efficiency Prediction in Heterogeneous Composite Cores Architectures," in arXiv preprint, 2018.

[23] H. Sayadi, N. Patel, A. Sasan and H. Homayoun, “Machine learning-based approaches for energy-efficiency prediction and scheduling in composite cores architectures,” In IEEE International Conference on Computer Design (ICCD), pp. 129-136, Boston, MA, 2017, doi: 10.1109/ICCD.2017.28.

[24] Virusshare Malware Database - https://virusshare.com/ Accessed - September 2017

BIOGRAPHY

Onkar Randive attended the University of Pune, India where he received his Bachelor of Engineering in Electronics and Telecommunication in 2016. He went on to receive his Master of Science in Computer Engineering in 2018 from George Mason University. He has also worked as a Visiting Research Assistant at The University of Southern California – Information Sciences Institute.