Enhancing IoT Security: Novel Mechanisms for Malware Detection using HPCs and Neural Networks

Shashwat Adhikari
University of Plymouth
shashwat.adhikari@plymouth.ac.uk

Hafizul Asad
University of Plymouth
hafizul.asad@plymouth.ac.uk

Kevin Jones
University of Plymouth
kevin.jones@plymouth.ac.uk

***Abstract*—With the exponential rise of internet-connected devices, the number of Internet of Things (IoT) devices has surpassed that of traditional IT devices. This proliferation in IoT adoption can be attributed to the growing demand for manufacturing automation and the desire for enhanced quality of life, leading to the production of smart devices in various industries. However, this rapid adoption has caught the attention of malicious actors, resulting in a significant increase in cyber-attacks targeting IoT devices. In response to this emerging threat landscape, research on IoT security has been active. Nevertheless, the lack of commercial tools specifically designed for IoT device security raises concerns about the ability of security research and adoption to keep pace with the rising number of malicious actors. To address this gap, this study focuses on introducing a novel mechanism for detecting malware in IoT devices. By conducting experiments, we demonstrate that using Hardware Performance Counters (HPCs), complemented by physical features such as power consumption, can improve the current malware detection capabilities. Specifically, we employ Recurrent Neural Networks (RNN) and Multi-Layer Perception Neural Networks (MLP) to achieve a remarkable detection accuracy of 95% within a timeframe of less than 10 seconds from infection.**

***Index Terms*—Internet of Things (IoT), Hardware Performance Counters (HPC), Malware Detection, Neural Networks (NN)**

I. Introduction

As the number of internet connected devices continues to rise exponentially, the arms race between malicious threat actors and the security research community is also seen to escalate proportionally. In recent years, however a new trend has emerged, the number of IoT devices has overtaken the number of traditional IT devices connected to the internet [9]. Every single industry has found a use of this technology, and as such the adoption rates are astronomical. This explosive adoption rate has been noticed by the malicious actors and subsequently the following effects were noticed: a 300% increase in the number of cyber-attacks in IoT devices around 2019 [6], and 25% of all cyber-attacks targeting IoT devices in 2020 [20]. Research in IoT security has also been vigorous, however there is an acute lack of commercial tools designed specifically for the security of IoT devices, fuelling concerns that security research and adoption may not be keeping up with the increase in malicious actors.

In recent years, research on IoT security has grown significantly, but there are clear indicators of gaps in understanding and implementation of effective security measures. Literature regarding malware detection approaches in IoT devices over three years between 2019 and 2022 shows a notable shift in research focus from the traditional signature-based methods to machine learning-based approaches [22], [8]. The research gaps identified include the need for standard datasets and more robust techniques, as well as the emphasis on interpretability of machine learning models and their transferability across IoT domains. Notably, none of the surveyed papers include studies that employ HPCs for malware detection in IoT devices.

Therefore, this study proposes a novel malware detection mechanism for IoT devices utilizing Hardware Performance Counters (HPC). HPCs are specialized registers integrated within the hardware architecture of a processor, designed to monitor various aspects of system performance such as processor usage, memory usage, and cache behaviour. As explained by Intel Corporation, HPCs provides low-level performance information and is primarily designed to be used for debugging, system optimization, and performance profiling [2], it counts specific events such as the number of instructions executed, the number of cache misses, the number of branch mispredictions, and the number of cycles elapsed during program execution and much more. While the concept of hardware counters for performance monitoring and analysis dates back to the 1970s with the development of performance monitoring hardware for IBM mainframes [12], its first recorded application in security is attributed to a 2003 publication [14]. Despite the presence of contradictory arguments in the existing literature, we hypothesize that HPCs can serve as an effective means of malware detection in IoT devices. To guide the investigation towards this goal, a set of specific research objectives has been identified:

- Are HPCs effective in differentiating between benign and malicious datasets in the context of malware detection in IoT devices?
- Are there any potential limitations of HPCs due to their probabilistic nature and if so, to what extent can it impact their effectiveness in malware detection?
- How effective of a malware detection solution can be designed using the HPC data from IoT devices in terms of detection accuracy and speed?

And in answering these questions, we have determined that malware detection using HPCs is not only viable but produces comparable, if not better, classification accuracy when compared to similar existing mechanisms. The proposed mechanism utilizing Neural Networks is the primary contribution of this study.

II. Methodology

We have adopted an experimental approach for this research due to the scarcity of relevant datasets for malware detection in IoT devices. The subsequent sections will provide detailed descriptions of the experimental architecture, collected data, and the subsequent exploration and analysis conducted with the objective of designing a robust solution for malware detection in IoT devices.

A. Experimental Architecture

Given the acute lack of datasets tailored for IoT security, we have adopted an experimental approach towards data collection and analysis. For this purpose, an experimental setup has been designed aimed towards extracting Hardware Performance Counters and other pertinent features from IoT devices. The experimental setup consists of several Raspberry pis connected in a sandboxed network with simulated internet connection provided by INetSim.

B. Data Collection

HPCs rely on physical registers to store the counts of sub-architectural events and are limited by the number of such registers present in the particular CPU model. In this study, the Raspberry Pi model 2bs has been used, which is equipped with ARM Cortex A9 CPUs that incorporate 58 events; however, it contains only 7 monitoring registers. Typically, each register has the capacity to monitor a single event; however, by utilizing time-multiplexing techniques, a total of 14 events have been collected for the initial analyses. These are:

- No. of Cycles
- No. of Functions
- No. of Float
- No. of Load
- No. of Store
- No. of Network Interrupts
- No. of Committed Instructions
- No. of Integer Instructions
- No. of Branch Instructions
- No. of Data Cache Hits
- No. of Data Cache Misses
- No. of Instruction Cache Hits
- No. of Instruction Cache Misses

The Raspberry Pis in our study serve as a platform to simulate a range of IoT devices using a combination of modified device firmware and industry benchmarks. The collection of HPC data is achieved using an in-built linux tool "perf stat" tool, which is configured to poll the registers of the simulated devices at a granularity of 1ms. During the experimentation phase, a benchmark is executed for a duration of 2 minutes with and without the presence of malware running in the same threads. This results in the generation of approximately 120,000 observations per run. To minimize the noise from extraneous factors, the IoT device firmware running on the Raspberry Pi is pinned to a specific thread, and the monitoring tool is set to observe only these threads. To further reduce noise, most background processes are terminated during data collection. The malware executables are obtained from a GitHub repository containing live samples [23] and Malware Bazaar, an Open-Source Intelligence platform [1]. The current dataset under analysis includes 24 malware samples, see table I.

TABLE I
Malware Samples used

Family	Samples	Family	Samples
Botnets	Mirai Kaiten Qbot Linux.Wifatch Hajime Mozi lightaidra	Rootkits	Bashlite BrickerBot Rustock Zues/Zbot Alureon Ztorg
Spyware	Tycoon Erebus Pegasus FruitFly IoTroop	Ransomwares	Ryuk GrandCrab Maze Revil RansomExx NotPetya

III. Data exploration

This section presents the initial data exploration and analysis conducted to support the research objectives. We first examine the non-deterministic characteristics of HPC data in section III-A. Then, using correlations (section: III-B) and clustering (section: III-C), we demonstrate that despite the observed non-deterministic characteristics, the execution of malware on an IoT device results in detectable variations in HPC values. Additionally, using these same techniques, we identify a subset of HPC features that are more influenced by malware execution.

A. Probabilistic Nature of HPCs

There are several approaches and techniques that can be used to estimate the degree of non-determinism in a given context. For example, one approach is to measure the variation or deviation of the data points from their mean or expected values [3]. Another approach is to use statistical models to compare the distribution of the data points in different runs [13]. Following the first approach, a statistical summary typically includes measures of central tendency, such as the mean, median, and mode, as well as measures of dispersion, such as the standard deviation and range. Comparing the statistical summary across multiple runs of the same malware, we observed variations in the interquartile range, mean and standard deviations in most of the HPC events, see figures 2, 3, 4. However, these were not consistent across the collected counters, for instance, No. of Float instructions and No. of integer instructions were constant in their statistical summary,

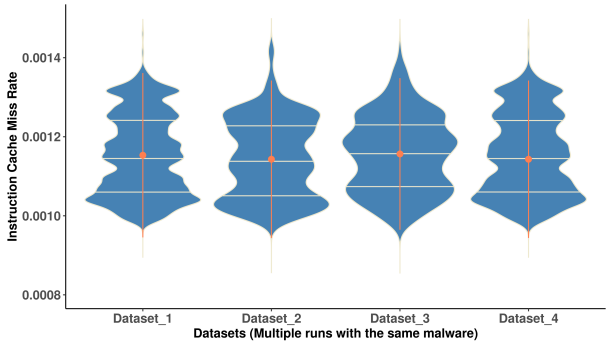

Fig. 1. Comparison of Instruction Cache Miss rates between multiple runs

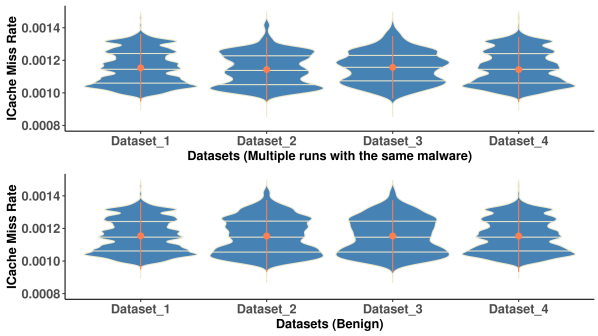

Fig. 2. Comparison of ICache Miss rates between different runs and between malicious and benign datasets

whereas the most differences were observed in the Data cache and Instruction cache miss rates.

Next, we employed the Kernel Density Estimate (KDE) method to estimate the probability density function of the HPC events. It provides a way to estimate the distribution of a continuous variable based on a sample of data. KDE operates by wrapping a kernel function around each sampled data point. The shape of the curve used to approximate the probability density function is determined by the kernel function. Using the Gaussian kernel, we plot the distribution of the observations in the dataset. Here too, we observed the most variations in the Data/Instruction cache miss rates, Fig 3. Both the Statistical summary and kernel density estimates are visualized using violin plots, where:

- The blue area represents the kernel density distribution, which indicates the probability density of the variable at different values. It displays how the data points are distributed throughout the dataset, with longer horizontal peaks indicating areas of greater density.
- The horizontal yellow lines denote the quantiles, which are statistical measures that divide the dataset into equal parts. The three lines represent the upper quartile (Q3), median (Q2), and lower quartile (Q1) of the data, arranged from top to bottom.
- The orange dot represents the mean, which is the arithmetic average of the dataset. The vertical orange line extending from the dot represents the variance, which indicates how much the data points are spread around the mean value.

The violin plots in Fig 1 show that the four datasets used for visualization have different distributions. Note that while the following behaviour is present on the entire dataset, for visualization purposes, only four data samples are used. The quartiles (horizontal yellow lines) show that the median (Q2) of the third dataset is higher than the other three datasets, indicating that its values are generally higher. The third dataset's upper quartile (Q3) and lower quartile (Q1) are closer together, indicating a more evenly distributed dataset. The first and third datasets have a higher mean (orange dots) than the other two. Each dataset has a different variance, shown by vertical orange lines. The second dataset has a lower variance,

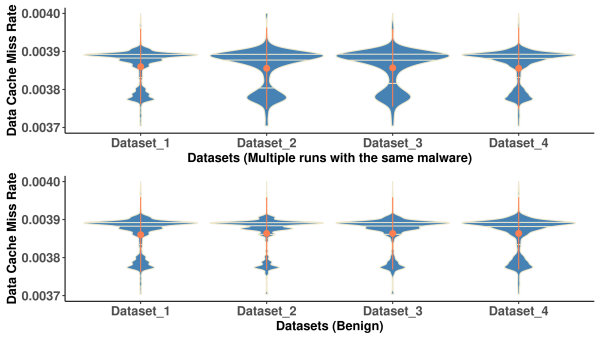

Fig. 3. Comparison of Data Cache Miss rates between different runs and between malicious and benign datasets

indicating that data points are clustered around the mean. The first dataset has the highest variance, meaning its data points are more spread out than the other three. The quartiles and medians also indicate that the datasets are skewed differently.

A similar comparison on the benign data also shows the same non-determinism, Figure 2. Furthermore, when benign datasets are compared to the malicious datasets, there are additional visual differences, especially when comparing datasets 2 & 3 between the malicious and benign datasets, the shape of the distribution function varies significantly. Data Cache miss rate also follows similar trends (Figure 3).

Conversely, we do not observe any non-determinism in the following two events: No. of Integer instructions & No. of Float instructions, Fig. 4.

From the comparison of non-determinism, the following conclusions can be made about the set of HPC data we have collected so far:

- Non-Determinism is present in HPC values, however not all the events display such behaviour
- No. of Integer Instances, and Number of Float Instances are fairly constant between runs, whereas Instruction cache and Data Cache miss rates vary the most.

The impact of non-determinism on HPC's malware detection capability is investigated in further sections. Results from violin plots and *a table* illustrate that the variations caused by malware are more significant than those resulting

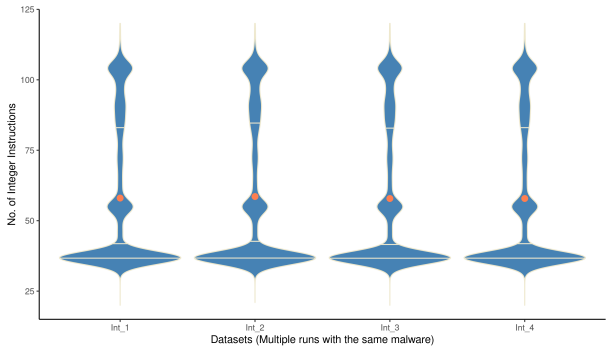

Fig. 4. No. of Integer Instructions

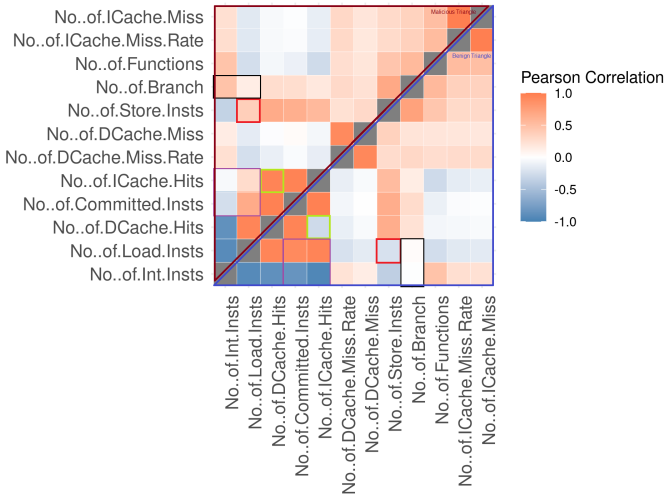

Fig. 5. Pearson Correlation Coefficient Heatmap

from probabilistic factors. Nevertheless, we will demonstrate that malware execution induces observable variations in HPC values through multiple visualization techniques. These techniques will also aid in selecting relevant features for data classification.

B. Pearson Correlation Analysis

Next, we perform a correlation analysis to study the correlations between different collected events. Pearson correlation coefficient is a statistical measure that quantifies the strength of the linear relationship between two variables. It is used to determine how closely related two variables are to each other. Pearson's correlation coefficient is calculated as the covariance of the two variables divided by the product of their standard deviations.

The resulting coefficient is represented by the symbol "r" and can take values between -1 and 1. When the value of "r" is close to -1, it indicates a perfect negative linear relationship, while a value of 0 indicates no linear relationship between the two variables. When "r" is close to 1, it indicates a perfect positive linear relationship between the two variables. Pearson correlation coefficient is frequently used in machine learning applications to determine the degree of correlation between two variables, which is useful for feature selection, model building, and evaluating model performance [18].

The computation of the Pearson correlation coefficient between all features of a dataset generates a correlation matrix, which can be depicted using a heatmap, as illustrated in Figure 5. Typically, the triangles on two sides of a correlation matrix along the diagonal contain the same information. This is because the correlation matrix is a square matrix, where each row and column represent a variable in the dataset, and each cell in the matrix represents the correlation coefficient between two variables. The diagonal of the matrix always contains a correlation coefficient of 1 since it represents the correlation of each variable with itself. The upper and lower triangle of the matrix represent the same set of pairwise correlations, in a different order. Therefore, this allows us to calculate a similar correlation matrix for a malicious dataset and replace the upper triangle with the malicious correlation, allowing for comparison between them to determine how this relationship between the features is affected by the addition of malware running on the device.

In Figure 5, we have highlighted the sections where the introduction of malware has had a significant impact on the correlation; specifically, the pair of red highlighted cells; these correspond to the correlation coefficient between the number of load and store events. Notably, the benign dataset (bottom triangle) exhibits a negative correlation between these variables, whereas the malicious dataset (upper triangle) shows a positive correlation. This is an interesting observation, which implies that the inclusion of these features will be significantly helpful for the Neural Network models to classify benign versus malicious datasets. This correlation heatmap suggests the following features to be interesting: No. of Load, Store, Branch, committed instructions, and the DCache & ICache miss rates. Note that these are also the same events in which we observed higher rates of non-determinism.

C. Clustering Analysis

Next, we use K-Means clustering to further confirm our hypothesis that malware induced variations in HPCs are significant and observable. K-Means has been used for feature selection in various fields, including computer vision and bioinformatics [17] [15]. Typically, when performing clustering with a high-dimensional dataset, it is reduced to two dimensions using various techniques like Principal Component Analysis, however this process is known to lose certain information [11]. Therefore, we will use K-Means clustering without dimensionality reduction: The clustering results of K-Means will be visualized with two features at a time, and the plots can then be used to evaluate the separation of the clusters. Well-separated clusters with clear boundaries indicate that the features used are useful in grouping data into two distinct clusters (Malicious or Benign), while overlapping or poorly separated clusters suggest that the features may not be suitable for classification. There are several published works on the use

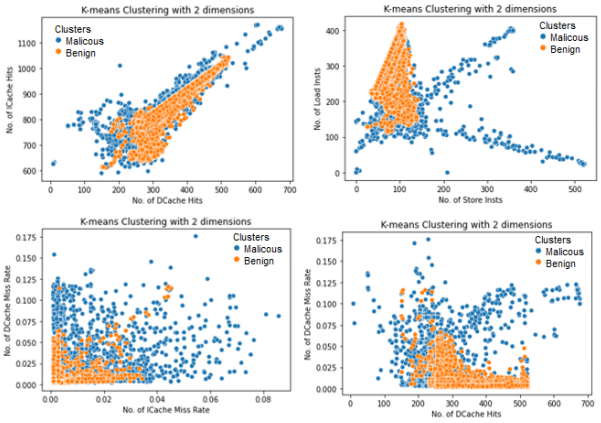

Fig. 6. K-Means Clustering

of K-means clustering for high dimension data similarly, [24] and [5].

Take the four plots in Figure 6 for example, the top two clusters are well-separated, with clear outliers, whereas the bottom two plots have certain overlaps. Evaluating the similar plots between different combinations of features gives us another list of features that are potentially significant in the classification of malicious and benign data: No. of Branch, Load, Store, and Functions.

IV. Results

Following the presented initial exploratory data analysis and feature selection, a range of Neural Network models are trained and evaluated in terms of their prediction accuracy and detection speed. Neural Networks are preferred over traditional machine learning models for time series classification tasks, as they are better able to handle complex non-linear relationships and process large amounts of data [10]. Specifically, Recurrent Neural Networks (RNNs) are often used for time series classification because of their ability to capture temporal dependencies in the data through the use of feedback loops in their architecture. Additionally, some studies have shown that Multi-Layer Perceptron (MLP) and certain Convolutional Neural Network (CNN) models also perform exceptionally well in time series classification when dealing with large and complex feature sets [7]. Therefore, drawing from literature studied and discussed in previous sections, we have implemented the following NN Models: MLP (Multi-Layer Perception); RNN (Recurrent Neural Network); MCDCNN (Multi Channel-Deep Convolutional Neural Network); FCN (Fully Convolutional Network); and CNN (Convolutional Neural Network). In addition to comparing the models with each other, we will also be comparing the different feature sets within each model. These feature sets are described in table II.

Figure 7 provides a comparison of the predictive accuracy of the chosen models utilizing different feature sets. The results indicate that the MLP model utilizing the star feature set outperforms other models and feature sets. These findings

TABLE II
Feature Sets under test

Feature Sets	Description
Baseline	All 14 HPC events
Baseline + Power	All 14 HPC events and Power Consumption data
Star Features	No. of Load, Store, Branch, Functions, ICache/DCache miss rates & Power Consumption
Partial Frequency Domain data	Star Features + (Power consumption and No. of Branch in their frequency domain)

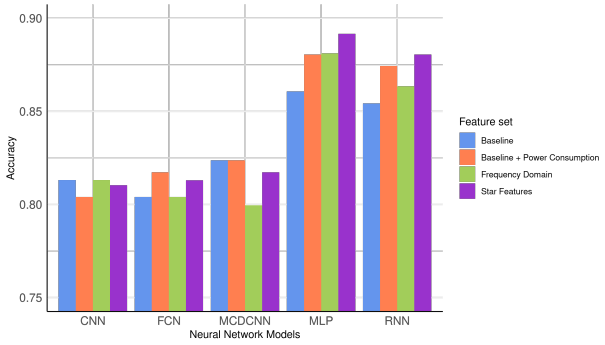

Fig. 7. Accuracy % of different models using different feature sets

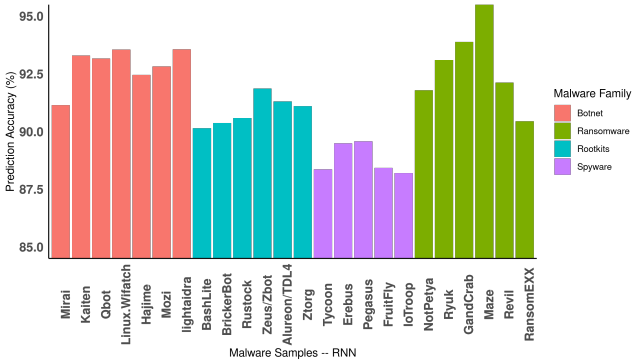

Fig. 8. Accuracy percentage of different malware samples using RNN

support the results of the HPC data exploration, which guided the selection of features included in the star feature set. Notably, MLP and RNN models perform better than the three CNN-based models. This suggests that learning the temporal dependencies in the time series HPC data is crucial in our specific use case.

Having confirmed that MLP and RNN are the top-performing models among those tested based on the results of the model comparison above using different feature sets, we now aim to investigate the detection accuracy and speed of these models exclusively and drop the other models in consequent tests.

Further investigation into the prediction accuracy of individual malware samples, as depicted in Figure 8, reveals some interesting insights. Notably, ransomwares exhibit exceptional classification accuracy, with one sample surpassing 95% prediction accuracy. Botnets closely trail ransomwares, achieving accuracy rates ranging between 91% and 94%. In certain cases,

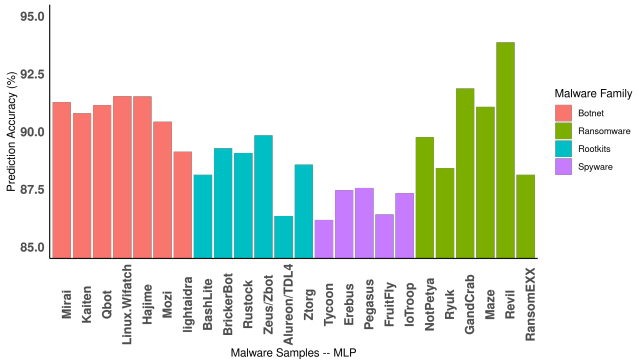

Fig. 9. Accuracy percentage of different malware samples using MLP

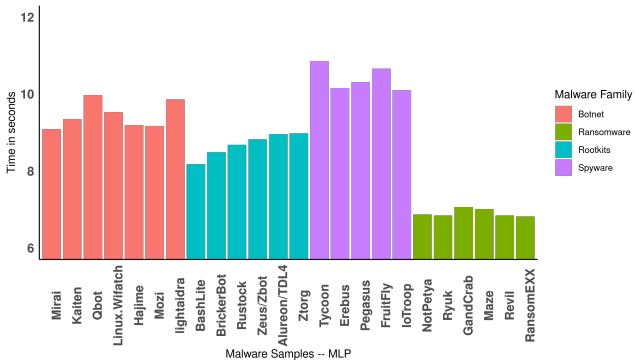

Fig. 10. Detection Speed of different malware samples using MLP

botnets even outperform specific ransomware samples. This higher accuracy when compared to other malware families can be attributed to the inherent characteristics of ransomwares and botnets, which involve extensive code execution and network activities during the initial stages of infection. On the other hand, spywares which are specifically designed to remain concealed until necessary, demonstrate some of the lowest prediction accuracy values among the tested malware families. Their evasive nature and limited observable behaviours make them inherently challenging to detect accurately. In terms of the models employed, MLP exhibits a similar pattern as the RNN: Higher accuracy with ransomwares, followed by botnets then rootkits and finally spywares. However, the overall accuracy levels achieved by MLP are lower compared to those attained by RNN across all malware families.

Shifting our focus to the detection speeds, it is observed that ransomwares exhibit the shortest detection times in our experiments. Using RNN, we were able to detect ransomware execution in approximately 5 seconds, as illustrated in Fig 11. Similarly, MLP demonstrates the ability to detect ransomware execution within a range of 6 to 7 seconds, as seen in Fig 10. Evidently, the distinguishing characteristics that contribute to higher detection accuracy of ransomwares also facilitate faster detection times. However, it is crucial to emphasize the considerable disparity observed in the detection speeds across different malware families. The notable differences in

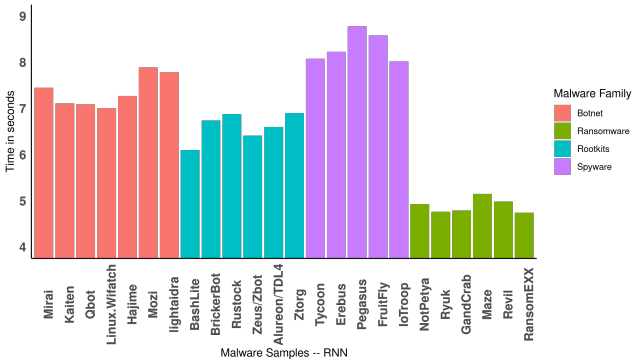

Fig. 11. Detection speed of different malware samples using RNN

detection times highlight the varying complexities associated with each malware type.

V. Discussions and Limitations

The findings of this study provide clear evidence that Hardware Performance Counters (HPCs) can be effectively utilized for malware detection, demonstrating both satisfactory accuracy and speed. Despite the initial concerns about the non-deterministic nature of HPCs, it was observed that this characteristic has minimal impact on the effectiveness of the detection mechanism. Previous works have highlighted the detrimental effects of such non-determinism in security applications. However, the evidence gathered in this research shows that neural network models have the capability to learn and adapt to this non-deterministic behaviour, thereby mitigating its influence. In contrast, older studies employing signature or threshold based detection mechanisms appear to be more susceptible to the challenges posed by non-determinism in HPCs.

In order to provide a complete and honest evaluation of the findings presented, it is also important to acknowledge the limitations of this study.

We initially aimed to collect HPC data directly from IoT devices to facilitate this study, however, we encountered some challenges: Although the processor and kernel utilized in the IoT devices have the capability to support HPC monitoring, accessing them requires shell access which is actively prevented by the security measures in place. As a result, collecting HPC data from the IoT devices directly introduces significant processing overheads that must be considered in the study. Therefore, for this study we utilized raspberry pis with simulated IoT devices, however we are actively pursuing the extraction of HPC data from the IoT devices themselves to enable a more comprehensive and accurate analysis of malware detection mechanism.

The HPC data used in our research is specific to ARM platforms. Although we made sure to include events present in other CPU architectures as well, nevertheless, the specific implementations are known to vary with changes in the platform, potentially affecting the HPC values. As a result, we plan to collect data from other platforms, conduct similar

experiments, and assess the generalizability of our models and findings.

VI. Related Works

A. Malware Detection in IoT devices

In recent years, there has been a significant surge in research activity within the domain of IoT security. However, a common comment in the literature is that IoT security research is still in its formative stages; a similar statement is always present in most publications, indicating that there may remain significant gaps in the understanding and implementation of effective security measures for IoT systems. As a starting point for this literature review two survey papers were reviewed, both aimed to classify existing works and provide a taxonomy of the studies available in the domain of Malware Detection approaches in IoT devices, but they were published three years apart. Starting at 2019 [22] provides an overview of malware detection techniques in IoT and highlights their advantages and limitations. Then in 2022 [8] focuses on analysing the various malware detection systems proposed for the Enterprise IoT environment. Comparing these studies provides some interesting insights into the progress in this discipline from 2019 to 2022, they are summarized below.

The shift in research focus from traditional malware detection methods to machine learning-based approaches is one of the most significant changes observed in the domain over three years. Traditional techniques such as signature-based and anomaly-based detection methods were the primary topic of discussions in the 2019 study. The 2022 study, on the other hand, focuses on a detailed discussion of machine learning-based approaches, such as deep learning and reinforcement learning. This shift is also reflected in the research gaps identified in both publications: The 2019 study identified problems such as a lack of standard datasets for evaluating IoT malware detection techniques and a need for more robust and efficient techniques that can handle the high volume and variety of data generated by IoT devices, whereas the 2022 paper pinpoints a different set research gaps; a lack of attention to explainability and interpretability of machine learning models and a need for more research on the transferability of machine learning models across different IoT domains. Overall, the key contributions of both studies include identifying research gaps and proposing taxonomies or comprehensive surveys of existing literature. It is worthy to note that neither of the survey papers discussed above incorporates any studies that employ HPCs for the purposes of malware detection within IoT devices. The subsequent section highlights the existing research that does utilize HPCs for security, although very little research exists applying this to IoT devices. Similarly, the work presented in [19] claims to be the pioneering hardware-assisted malware detection mechanism. However, our work distinguishes itself through the utilization of distinct underlying features. While we employ hardware performance counters, the study in question relies on data extracted from the embedded trace buffer of a processor. This buffer stores trace information generated during a processor's execution, primarily intended for debugging and performance analysis purposes.

B. Hardware Performance Counters

HPCs have been used for security purposes in multiple studies with varying levels of success over the years, although mostly for general purpose computers running Intel x86 or AMD platforms. [4] provides a comprehensive survey of the challenges and perils of using HPCs for security, however the literature used draws from works in other domains such as profiling, performance optimizations, and OS support as well. It concludes that more than 45% of the reviewed literature do not recommend the use of HPCs. This is attributed to two main reasons frequently cited in the literature: non-determinism and a lack of portability. The survey study also reports that only about 10% of the works using HPCs address and explicitly argue how they account for this non-determinism. Drilling down into the security research featured in this study, the next sections summarizes the previous relatively successful attempts to use HPC for security and how they address the non-determinism of HPCs:

The authors behind [16] note that HPC data can be non-deterministic due to various factors, such as hardware differences, software variations, and operating system behaviour. To address this issue, they propose a calibration method that involves collecting HPC data from a reference program that is executed on the same hardware and software configuration as the target program. The reference program's HPC data is used to create a baseline for the expected HPC data of the target program. The authors then compare the HPC data of the target program to the baseline to detect any discrepancies that could indicate malicious activity. Through experiments, the authors show that the proposed calibration method can effectively reduce the impact of non-determinism on the integrity-checking process. They also note that the calibration process can be time-consuming but argue that the benefits of improved accuracy and reduced false positives outweigh the costs. Overall, the authors provide a practical approach for addressing the issue of non-determinism when using HPCs for program integrity checking, although with a very high processing overhead.

[25] proposes a novel approach to detect Return-Oriented Programming (ROP) attacks using HPCs. It collects HPC data during the execution of a program and applies a detection algorithm to identify attacks. The approach was evaluated using several real-world applications and demonstrated high detection accuracy with a low false-positive rate. The authors, however, do not address the issue of non-determinism of HPCs. Instead, they acknowledge this issue and propose a technique to detect ROP attacks using a small set of performance counters. They argue that non-determinism can be mitigated by selecting the appropriate performance counters and adjusting the threshold values to account for variations. They also note that their technique is designed to detect ROP attacks with high accuracy, even in the presence of non-deterministic behaviour.

[21] proposes a new technique for detecting kernel rootkits, which are malware that can modify the control flow of the kernel to gain control of the system. The proposed technique uses HPCs to monitor the system call and interrupt numbers in the kernel, which are used to detect modifications to the kernel control flow. It uses a combination of threshold-based and machine-learning-based techniques to detect anomalies in the HPC data and identify the presence of a rootkit. The authors evaluated NumChecker using a number of different rootkits and showed that it can successfully detect all of them with a low false-positive rate. They also compared NumChecker with other rootkit detection techniques and showed that it outperforms them in terms of accuracy and detection speed. The authors conclude that NumChecker is a promising approach to detecting kernel rootkits and that it can be integrated with existing security solutions to provide additional protection against these types of attacks. The authors do not explicitly address the issue of non-determinism of HPCs in this paper.

VII. Conclusion

Towards the goal of enhancing malware detection in IoT devices, this study focuses on HPCs and their suitability for the task. To achieve this, an initial exploratory data analysis and feature selection are conducted to identify relevant features for effective malware detection. The selected features include a subset of the HPC events, and great care has been given in ensuring that the chosen feature sets capture the necessary information to effectively classify malicious and benign datasets. The study then compares the predictive accuracy and detection speed of the Neural Network models using different feature sets. The following conclusions are drawn from the study:

- Ransomwares exhibit exceptional accuracy, with samples surpassing 95% prediction accuracy.
- Botnets closely follow, achieving accuracy rates between 91% and 94%, sometimes even outperforming specific ransomware samples.
- Spywares, designed to remain hidden, demonstrate lower prediction accuracy due to their evasive nature and limited observable behaviours.
- Overall accuracy levels achieved by MLP are lower than those attained by RNN across all malware families, this highlights the significance of capturing temporal dependencies in time series data for effective malware detection.
- Ransomwares are detected with the shortest detection times in the experiments. RNN can detect a ransomware execution in approximately 5 seconds, while MLP achieves detection within a range of 6 to 7 seconds.
- The faster detection times are attributed to the same distinguishing characteristics that contribute to higher detection accuracy: extensive code execution and network activities during the initial stages of infection.
- The notable variation in detection speeds across different malware families emphasize the varying complexities associated with each type. Understanding these differences can inform the development of targeted defence mechanisms and response strategies tailored to specific malware types.
- Neural network models display the capability to learn and adapt to the non-deterministic behaviour of HPCs, thereby mitigating its detrimental impact on security applications.

In conclusion, this study was aimed towards addressing the existing gap in malware detection for IoT devices. Our approach introduced a novel mechanism that uses Hardware Performance Counters (HPCs). After exploring the probabilistic nature of HPCs, spending some time in feature selection, and through extensive experiments utilizing Recurrent Neural Networks and Multi-Layer Perception Neural Networks, we achieved a detection accuracy of 93% - 95% within a timeframe of less than 10 seconds from infection, thus demonstrating the efficiency of our proposed solution.

References

[1] Abuse.ch. *Malware Sample Exchange*. 2023. URL: https://bazaar.abuse.ch/.

[2] Reza Azimi, Michael Stumm, and Robert W. Wisniewski. "Online performance analysis by statistical sampling of microprocessor performance counters". In: *Proceedings of the 19th annual international conference on Supercomputing*. ICS '05. New York, NY, USA: Association for Computing Machinery, June 2005, pp. 101–110. ISBN: 9781595931672. DOI: 10.1145/1088149.1088163. URL: https://dl.acm.org/doi/10.1145/1088149.1088163 (visited on 06/20/2023).

[3] Carlos E. Budde et al. "A Statistical Model Checker for Nondeterminism and Rare Events". en. In: ed. by Dirk Beyer and Marieke Huisman. Vol. 10806. Cham: Springer International Publishing, 2018, pp. 340–358. ISBN: 9783319899626 9783319899633. DOI: 10.1007/978-3-319-89963-3_20. URL: http://link.springer.com/10.1007/978-3-319-89963-3_20 (visited on 04/01/2023).

[4] Sanjeev Das et al. "ROPSentry: Runtime defense against ROP attacks using hardware performance counters". en. In: *Computers & Security* 73 (Mar. 2018), pp. 374–388. ISSN: 0167-4048. DOI: 10.1016/j.cose.2017.11.011. URL: https://www.sciencedirect.com/science/article/pii/S0167404817302481 (visited on 03/29/2023).

[5] Chris Ding and Xiaofeng He. "K-Means Clustering Via Principal Component Analysis". In: *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004* 1 (Sept. 2004). DOI: 10.1145/1015330.1015408.

[6] Zak Doffman. *Cyberattacks On IOT Devices Surge 300% In 2019, 'Measured In Billions', Report Claims*. en. URL: https://www.forbes.com/sites/zakdoffman/2019/09/14/dangerous-cyberattacks-on-iot-devices-up-300-in-2019-now-rampant-report-claims/ (visited on 06/20/2023).

[7] Hassan Ismail Fawaz et al. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4 (July 2019). arXiv:1809.04356 [cs, stat], pp. 917–963. ISSN: 1384-5810, 1573-756X. DOI: 10.1007/s10618-019-00619-1. URL: http://arxiv.org/abs/1809.04356 (visited on 03/31/2023).

[8] Akshat Gaurav, Brij B. Gupta, and Prabin Kumar Panigrahi. "A comprehensive survey on machine learning approaches for malware detection in IoT-based enterprise information system". In: *Enterprise Information Systems* 17.3 (2023), p. 2023764. DOI: 10.1080/17517575.2021.2023764. eprint:

https://doi.org/10.1080/17517575.2021.2023764. URL: https://doi.org/10.1080/17517575.2021.2023764.

[9] *Global IoT and non-IoT connections 2010-2025*. en. URL: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/ (visited on 06/22/2023).

[10] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. "Recurrent Neural Networks for Time Series Forecasting: Current status and future directions". en. In: *International Journal of Forecasting* 37.1 (Jan. 2021), pp. 388–427. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2020.06.008. URL: https://www.sciencedirect.com/science/article/pii/S0169207020300996 (visited on 03/31/2023).

[11] Hung Hung and Su-Yun Huang. *On the efficiency-loss free ordering-robustness of product-PCA*. Tech. rep. arXiv:2302.11124 [stat] type: article. arXiv, Feb. 2023. DOI: 10.48550/arXiv.2302.11124. URL: http://arxiv.org/abs/2302.11124 (visited on 06/21/2023).

[12] IBM. *Setting up IBM Z and DS8000 Hardware Performance Monitoring Using Open Source Tools*. en. 2021. URL: https://community.ibm.com/community/user/storage/blogs/eric-reid/2021/07/20/setting-up-ibm-z-and-ds8000-hardware-performance-m (visited on 06/20/2023).

[13] Michael A. Irvine and T. Déirdre Hollingsworth. "Kernel-density estimation and approximate Bayesian computation for flexible epidemiological model fitting in Python". en. In: *Epidemics* 25 (Dec. 2018), pp. 80–88. ISSN: 1755-4365. DOI: 10.1016/j.epidem.2018.05.009. URL: https://www.sciencedirect.com/science/article/pii/S1755436518300185 (visited on 06/20/2023).

[14] Gaurav S. Kc, Angelos D. Keromytis, and Vassilis Prevelakis. "Countering code-injection attacks with instruction-set randomization". In: *Proceedings of the 10th ACM conference on Computer and communications security*. CCS '03. New York, NY, USA: Association for Computing Machinery, Oct. 2003, pp. 272–280. ISBN: 9781581137385. DOI: 10.1145/948109.948146. URL: https://dl.acm.org/doi/10.1145/948109.948146 (visited on 06/20/2023).

[15] Guangrong Li et al. "A novel unsupervised feature selection method for bioinformatics data sets through feature clustering". In: *2008 IEEE International Conference on Granular Computing*. Aug. 2008, pp. 41–47. DOI: 10.1109/GRC.2008.4664788.

[16] Corey Malone, Mohamed Zahran, and Ramesh Karri. "Are hardware performance counters a cost effective way for integrity checking of programs". In: *Proceedings of the sixth ACM workshop on Scalable trusted computing*. STC '11. New York, NY, USA: Association for Computing Machinery, Oct. 2011, pp. 71–76. ISBN: 9781450310017. DOI: 10.1145/2046582.2046596. URL: https://dl.acm.org/doi/10.1145/2046582.2046596 (visited on 03/29/2023).

[17] Maziar Moradi Fard, Thibaut Thonet, and Eric Gaussier. "Deep k-Means: Jointly clustering with k-Means and learning representations". en. In: *Pattern Recognition Letters* 138 (Oct. 2020), pp. 185–192. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2020.07.028. URL: https://www.sciencedirect.com/science/article/pii/S0167865520302749 (visited on 06/21/2023).

[18] Andreas C. Müller and Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. en. Google-Books-ID: vbQlDQAAQBAJ. "O'Reilly Media, Inc.", Sept. 2016. ISBN: 9781449369897.

[19] Zhixin Pan, Jennifer Sheldon, and Prabhat Mishra. "Hardware-Assisted Malware Detection using Explainable Machine Learning". In: *2020 IEEE 38th International Conference on Computer Design (ICCD)*. 2020, pp. 663–666. DOI: 10.1109/ICCD50377.2020.00113.

[20] Gaurav Sharma. *A sharp increase in cyberattacks on IoT devices: Check Point*. en. Apr. 2023. URL: https://securitybrief.co.uk/story/a-sharp-increase-in-cyberattacks-on-iot-devices-check-point (visited on 06/20/2023).

[21] Xueyang Wang and Ramesh Karri. "NumChecker: detecting kernel control-flow modifying rootkits by using hardware performance counters". In: *Proceedings of the 50th Annual Design Automation Conference*. DAC '13. New York, NY, USA: Association for Computing Machinery, May 2013, pp. 1–7. ISBN: 9781450320719. DOI: 10.1145/2463209.2488831. URL: https://dl.acm.org/doi/10.1145/2463209.2488831 (visited on 03/29/2023).

[22] Mohammad Wazid et al. "IoMT Malware Detection Approaches: Analysis and Research Challenges". In: *IEEE Access* 7 (2019), pp. 182459–182476. DOI: 10.1109/ACCESS.2019.2960412.

[23] Shahak Shalev Yuval Nativ. *theZoo*. https://github.com/ytisf/theZoo. 2023.

[24] Hong Zeng and Yiu-ming Cheung. *Feature Selection for Clustering on High Dimensional Data*. Vol. 5351. Dec. 2008, p. 922. ISBN: 9783540891963. DOI: 10.1007/978-3-540-89197-0_85.

[25] HongWei Zhou et al. "HDROP: Detecting ROP Attacks Using Performance Monitoring Counters". en. In: *Information Security Practice and Experience*. Ed. by Xinyi Huang and Jianying Zhou. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 172–186. ISBN: 9783319063201. DOI: 10.1007/978-3-319-06320-1_14.