Time Series-based Malware Detection using Hardware Performance Counters

Abraham Peedikayil Kuruvila
The University of Texas at Dallas
Richardson, Texas
Abraham.Kuruvila@utdallas.edu

Sayar Karmakar
University of Florida
Gainesville, Florida
sayarkarmakar@ufl.edu

Kanad Basu
The University of Texas at Dallas
Richardson, Texas
Kanad.Basu@utdallas.edu

***Abstract*—With the advent of Internet-of-Things (IoT), Malware has been exponentially proliferating across a plethora of platforms including PC, mobile, and other embedded devices. Software-based solutions, such as Anti-Virus Software (AVS), are ineffective against modern Malware and incur an abundance of computational overhead. This has motivated researchers to develop Hardware-assisted Malware Detection techniques utilizing Hardware Performance Counters (HPCs). However, traditional HPC-based Malware detection does not account for the temporal order of the data. Consequently, false positives, *i.e.*, benign application being classified as Malware, become a major predicament. Furthermore, some devices are extremely limited in their hardware profiling capabilities, resulting in a limited feature space. To address these issues, we propose employing HPC data in conjunction with time series-based classifiers. Additionally, we introduce a Sequential Time Series-based Detection (SEQ-TSD) framework for identifying Malware. The proposed methodology uses *only a single HPC*, thereby reducing the profiling overhead. Our experimental results prove that the proposed framework can bolster the performance using *only a single HPC* to detect Malware with up to 95% accuracy, while incurring only a 5.56% false positive rate. Furthermore, we demonstrate that combining multiple HPCs in conjunction with SEQ-TSD boosts the average detection accuracy up to 97.91%.**

***Index Terms*—Hardware Performance Counters, Time Series Classification, Malware Detection.**

I. Introduction

The widespread proliferation of computing devices across the PC and mobile spectrum has engendered a ubiquitous production and dissemination of Malware. Malware can be of various types, *e.g.*, Trojans, Worms, Viruses, and Botnets, with objectives ranging from leaking sensitive information to denial of service. Traditionally, Anti-Virus Software (AVS) has been employed for identifying these malicious applications to ensure system security. Typically, AVS functions by executing the Malware in a virtual machine to monitor program behavior and track Application Programming Interface (API) usage [1]. Subsequently, a benign/Malware classification is produced by matching the encountered behavior with known data. Attackers and AVS have an ongoing cat-and-mouse game, as attackers design malevolent programs to evade AVS detection, but AVS bolsters its system to address novel subversion attempts. This ever lasting scrimmage results in AVS incurring a plethora of computational bandwidth [2]. This has led researchers to explore alternatives to AVS for Malware detection.

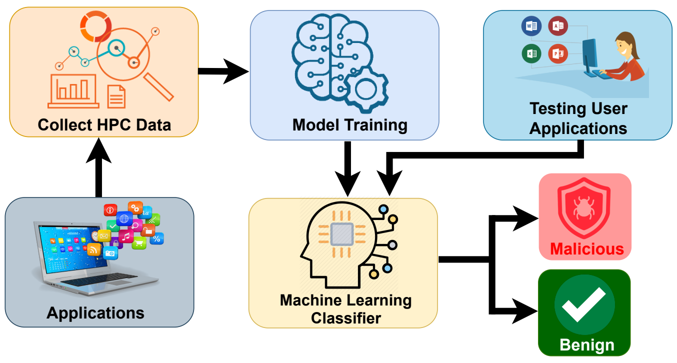


Figure 1. Overview of Hardware-assisted Malware Detection.

One promising substitute for AVS is a Hardware-assisted Malware Detector (HMD). This solution utilizes hardware features as the root-of-trust, which are harder to compromise by an adversary [3]. Recently, a collaboration between Microsoft and Intel has developed a HMD known as Threat Detection Technology (TDT). This helps endpoint security solutions harness CPU telemetry and hardware acceleration for identifying threats and anomalous activities. More specifically, it utilizes a combination of CPU telemetry and Machine Learning (ML) heuristics to detect specific behavior such as cryptojacking and ransomware detection [4]. Consequently, Intel's TDT is one of the first CPU-based malware behavior-monitoring techniques that goes beyond signature and file-based methods.

One of the most popular hardware features used in HMDs is the Hardware Performance Counter (HPC). HPCs monitor low-level microarchitectural events such as *CPU-cycles*, *Cache-misses*, and *Branch-misses*. HPCs can be used in conjunction with ML classifiers to create robust HMDs [3]. Figure 1 shows how HMDs are used to differentiate between benign and malicious applications.

Traditional HPC-based Malware detection techniques are not without pitfalls. Existing HMDs have no regard for the temporal order of HPC data, *i.e.*, the data is scrambled with no recognition of the sequential attributes. Consequently, HPC-based Malware detection suffers from a high number of false positives, *i.e.*, benign programs being labeled Malware [5]. Disregarding the transient sequence of HPCs induces these false positive as Malware and benign applications can share similar behaviors [6]. Furthermore, HPC-based Malware detection is dependent on the existence of hardware features in the processor or controller that operates the system to be protected. Some embedded devices have a minimal amount of hardware profiling capabilities, and the lack of available features culminates in poor model performance. Therefore, it is imperative to reduce the feature size, *i.e.*, the number of HPCs to be monitored.

To address these challenges, we propose respecting the transient order of the HPCs by utilizing Time Series-based Classifiers (TSCs) for Time Series Detection (TSD). We motivate TSD by first showing that the HPCs show significant autocorrelation. We build a formal statistical hypothesis test which corroborates with our intuition that time-series dependence should not be ignored, leading us to develop a time-series based methodology. Our experimental results demonstrate the performance improvement when utilizing TSCs trained on *a single HPC* and multiple HPCs. This bolsters our initial conjecture that the temporal attributes actually encapsulate better classifying information rather than single observations. Finally, we propose sequential time series detection (SEQ-TSD) employing *a single HPC* for devices with limited availability of hardware resources.

To the best of our knowledge, this is the first work that exploits TSCs in conjunction with ensemble properties for Malware detection. Specifically, our contributions are:

- We demonstrate autocorrelation of HPCs by providing a formal statistical test to show the existence of autoregressive coefficients through the Durbin-Watson statistic to motivate the need for time series-based classification,
- We assess the impact of classification capabilities and performance metrics furnished by TSCs when utilizing a *single HPC* and multiple HPCs,
- We design SEQ-TSD, incorporating multiple TSCs for accurate Malware detection utilizing *a single HPC* for devices with limited feature space, and,
- We evaluate combining multiple HPCs in conjunction with our proposed SEQ-TSD for unconstrained devices.

The rest of the paper is organized as follows. Section II presents the background on HPCs and our motivation for time series-based classification. Section III describes prior research on HPC and time series-based Malware detection. Section IV describes our proposed methodology, and Section V presents the experimental results. Finally, Section VI concludes the paper and gives directions for our future works.

II. Background and Motivation

A. Hardware Performance Counters

Hardware Performance Counters (HPCs) are special purpose registers that are found in most modern processors [3], [7]. They track low-level microarchitectural events including *LLC-store-misses, L1-dcache-stores, Ref-cycles,* etc. HPCs were originally developed for performance tuning and software optimization as they enable profiling an application. The quantity of available HPCs is dependent on the employed processor. A *Raspberry Pi 3* supports 20 HPCs, but an Intel i5-4210U has 28 HPCs. However, only a finite number of HPCs can be concurrently monitored, typically a maximum of four depending on the processor architecture [3], [7], [8]. Prior research has shown that an application's HPC trace can be employed in conjunction with ML models for Malware detection [3], [8], [9].

B. Challenges of HPC-based Malware Detection

One major issue with HPC-based HMDs is the large false positive rate [5]. A HMD approach that labels a benign program as Malware is disastrous, since inhibiting users from utilizing required applications would not see wide-scale utilization. A challenge with traditional HPC-based Malware detection is the utilization of scrambled HPC data. Therefore, Malware that incorporates benign commands would culminate in current HMDs incurring a plethora of false positives as a single sample count is not optimal for determining a program's classification [6]. As an example, the *PNSCAN* Malware employs the benign *ping* command. Scrambled data is insufficient for distinguishing between a benign program with the *ping* command and the pernicious *PNSCAN* Malware, as there would be analogous samples between the two.

HMDs are also reliant on hardware features existing in the processor or controller that regulates the system to be secured. While HPCs are integrated in most modern processors, there are still a plethora of devices that do not incorporate them. As an example, Texas Instruments' Solar Micro Inverter, which utilizes a TMS320F28035 32-bit micro-controller, has extremely limited profiling capabilities. Only a single feature can be profiled through the development environment's debug server scripting tool. Consequently, the utilization of a single feature would be insufficient for traditional HMDs to furnish high performance. To address these challenges, we develop SEQ-TSD that provides high accuracy and maintains a low false positive rate while only employing *a single HPC.*

C. Motivation for Time Series-based Malware Detection

In this subsection, we show significant autocorrelation for the traces of HPCs *Branch-instructions, Branch-misses, Cache-misses, and LLC-load-misses* which were collected from an ARM Cortex A53 processor. We collected HPC data from Malware procured from Virusshare [10], and our employed benign applications consists of the MiBench and Phoronix benchmark [11]. We provide a thorough explanation of our experimental framework in Section V. We utilized the *testcorr* and *IMTest* libraries in *R* to furnish a statistical hypothesis test for the existence of Autoregressive coefficients through the Durbin-Watson (DW) statistic [12]. These findings bolster a solid statistical motivation to select classifiers that respect the time-series structure instead of TMD that scrambles it.

We propose a formal hypothesis test to validate existence of auto-correlation across the HPCs. For a stationary time-series y_t, auto-correlation at lag $h \geq 0$ can be defined simply by:

$$\rho(h) = Corr(y_t, y_{t+h}) \tag{1}$$

This definition is valid for any stationary time-series, with the understanding that if the observed process has a trend and a stationary error component, this would be computed after one regresses the trend part out. For simplicity, we restrict ourselves to a very simple model: Mean + Autoregressive (1) (AR(1)) error structure. The Auto Correlation Function (ACF) plots we presented in Figure 2(a) suggests such AR(1) phenomenon. We describe what an AR(1) error process looks like in the following detailed model description. The counts y_t for a fixed HPC is modelled as follows:

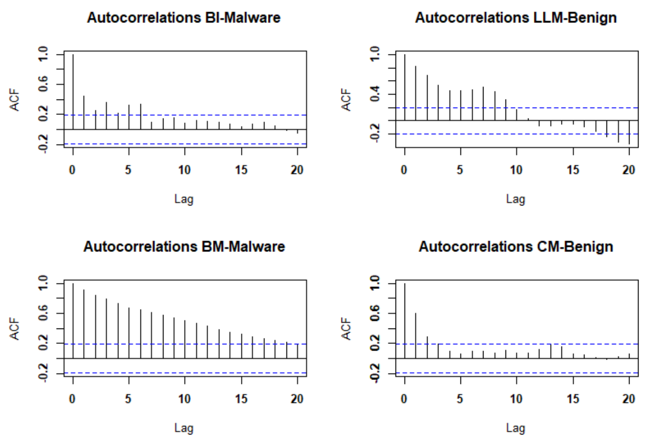
$$y_t = \mu + e_t, e_t = \rho e_{t-1} + u_t \tag{2}$$

where u_t stands for an independent white noise process and ρ stands for possible auto-correlation at lag one. Such an error process e_i is called an AR process with lag one (AR (1)) since e_i is regressed on e_{i-1}, its own lagged value. We will formally test the following hypothesis:

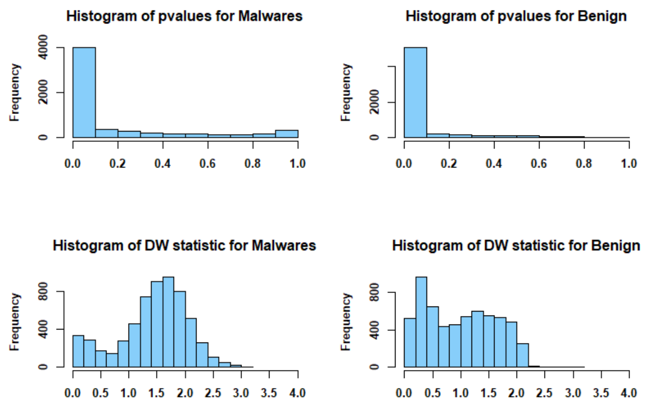
$$H_0 : \rho = 0 \text{ vs. } H_1 : \rho \neq 0 \tag{3}$$

We first compute the residuals $\hat{e}_i$ from our data as $\hat{e}_i = y_i - \bar{y}$. The Durbin-Watson statistic [12], [13] for the residual reads:

$$d = \frac{\sum_{t=2}^{T} (\hat{e}_t - \hat{e}_{t-1})^2}{\sum_{t=1}^{T} \hat{e}_t^2} \tag{4}$$



(a)



(b)

Figure 2. ACF Functions and DW Test Results for Different HPCs. Branch-instructions (BI), Branch-misses (BM), LLC-load-misses (LLM), Cache-misses (CM)

We use Figure 2(b) to plot the test statistic values and *pvalues* for all possible programs and HPCs in our dataset. These show the DW test statistic values are predominantly below two, which denotes significant positive correlation. Moreover, the *pvalues* for both malwares and benign dataset are mostly zero, which essentially rejects the hypothesis proposed in Equation 3 for any pre-specified level for most of these programs. These results lay the foundation of the main motivation for exploring TSCs in this paper.

We note that one could also formally test for other correlations by computing the $\hat{\rho}(h)$ for the series as follows:

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}, \hat{\gamma}(k) = \frac{1}{T}\sum_{t=1}^{T-k} X_t X_{t+h} - \bar{X}^2 \qquad (5)$$

and then formally test $H_0 : \rho(h) = 0$ versus $H_1 : \rho(h) \neq 0$. For brevity we do not present results on higher lags individually. Instead, we present a result for simultaneously testing whether the first m lags are 0 or not denoted as:

$$H_0 : \rho(1) = \cdots = \rho(m) = 0 \text{ vs. } H_1 : \rho(i) \neq 0 \qquad (6)$$

for some $1 \leq i \leq m$. The employed cumulative Ljung-Box (LB) statistic test is defined as:

$$LB = (T+2)T \sum_{k=1}^{k=m} \frac{\hat{\rho}(k)^2}{T-k} \text{ and } t_k = \sqrt{T}\hat{\rho}(k) \qquad (7)$$

where t_k can be used to test an individual lag k. Additionally, the *testcorr* package provides a set of robust $\tilde{Q}$ statistical tests, whose details we exclude for brevity but can be explored in [14], for evaluating the conditions in Equation 6. Figure 3 presents the LB and robust tests with the corresponding cumulative tests for evaluating the simultaneous hypothesis. We only used one instance of one HPC from each class, but the results uniformly showed the existence of a significant correlation for both individual and cumulative tests. We also compiled a pvalue histogram similar to 2(b) for the cumulative tests with $m = 10$ and $m = 15$ which we omit for brevity, as they furnished similar results. Consequently, these results bolster our conjecture that the temporal order is important.

III. Related Work

A. HPC-based Malware Detection

Malware, a portmanteau for malicious software, is any application that intrudes on a system. They exhibit a spectrum of behaviors from spying on sensitive information to inducing a device shutdown. To overcome the challenges that AVS incurs, HPC-based Malware detection has been proposed.

HPC-based Malware detection was first proposed in [15]. This was further improved to incorporate ML classifiers along with HPC data for distinguishing between malicious and benign programs [3]. NumChecker

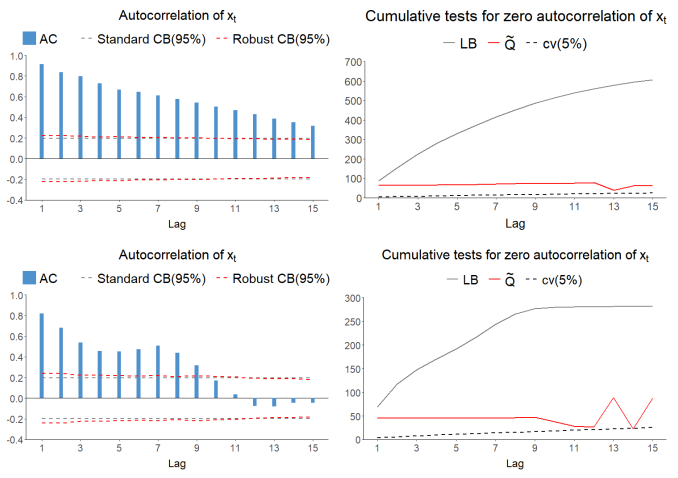


Figure 3. Robust and Cumulative Tests for Branch-misses (BM) Malware ((a) and (b)) and for LLC-load-misses (LLM) Benign ((c) and (d)).

was developed to utilize HPC values monitored using virtual machines for measuring system call events to identify malignant kernel control-flow alterations [8]. ConFirm was proposed for detecting firmware modifications through a HPC-based comparison scheme [16]. Furthermore, a two-stage ML-based classification technique was presented wherein feature selection is utilized to determine the best HPCs to be used in a bi-stage classifier for determining malicious software [17]. Several researchers have proposed techniques to secure pregnable systems from malicious applications using HPCs [9], [18]–[22].

B. Time Series-based Detection

In the realm of time series-based detection, prior works in various fields have exploited the sequential temporal order. In the medical field, time series classification has been utilized for determining a patient's reaction to an interferon-β treatment for multiple sclerosis [23]. The finance industry has also taken advantage of time series data, as various stochastic models trained on temporal variations of a stock price have been utilized to predict short-term movements in the market [24]. With respect to Malware detection, an entropy time series-based approach was developed for detecting Malware from portable executable files, where a shapelet TSC was utilized to extract discriminative features from the entropy signals [25]. A multivariate time-series analysis for Android Malware detection was produced by utilizing an autoregressive moving average model, where the data is compared for detecting malicious codes [26]. Prior research has utilized background network noise in conjunction with a non-stationary autoregressive model for intrusion detection in network traffic [27]. Krish-

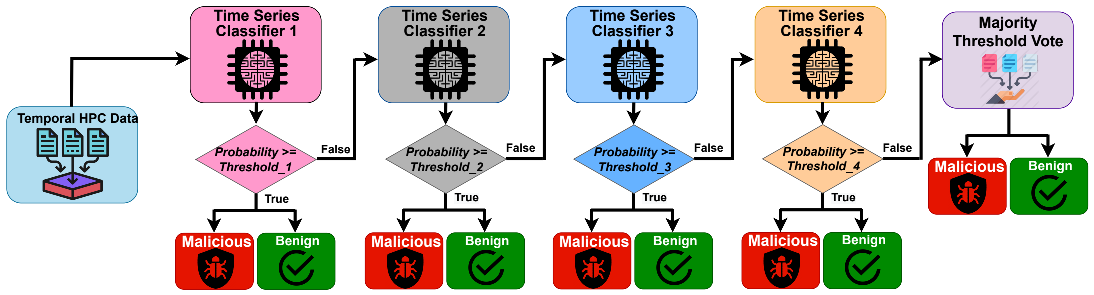


Figure 4. Overview of the Proposed SEQ-TSD.

namurthy et al. proposed an anomaly detection technique for securing cyber-physical systems that utilized an iterating interval for extracting features from multiple HPCs [28]. However, they did not use a statistical time series-based model, as the pre-processed features were used to train a Support Vector Machine (SVM) classifier. Furthermore, the scope of their work is limited to multi-HPC utilization for anomaly detection in programmable logic controllers rather than Malware detection. To the best of our knowledge, this is the first work that employs bare-bone HPC data in conjunction with TSCs for Malware detection utilizing *a single HPC*.

IV. Methodology

This section contains a chronological development of our proposed time series-based Malware detection methodology (TSD) for improving accuracy, reducing false positives, and minimizing feature overhead.

A. Time Series-based Classification

Time series classification considers data consisting of both (i) time points at which they are observed, and (ii) observations at those time points notionally represented as x(t_1), x(t_2),...,x(t_T) for observations at t_1, t_2,...t_T [29].

In this paper, we utilize the Time Series Forest (TSF) for all Time Series Detection (TSD) experiments. Prior research has shown it is computationally efficient and furnishes better performance compared to other TSCs [30]. TSF is an interval-based classifier that utilizes information from various intervals of a series [31]. This model functions by taking input data, splitting into intervals, extracting features including slope, standard deviation, and mean from the intervals, and training the model on these features [32]. The interval features of mean (f_1), standard deviation (f_2), and slope (f_3) for given intervals t_1 and t_2 and values at specific times v_i are defined as follows:

$$f_1(t_1,t_2) = \frac{\sum_{i=t_1}^{t_2} v_i}{t_2 - t_1 + 1} \tag{8}$$

$$f_2(t_1,t_2) = \begin{cases} \sqrt{\frac{\sum_{i=t_1}^{t_2}(v_i - f_1(t_1,t_2))^2}{t_2 - t_1}} & t_2 > t_1 \\ 0 & t_2 = t_1 \end{cases} \tag{9}$$

$$f_3(t_1,t_2) = \begin{cases} \hat{\beta} & t_2 > t_1 \\ 0 & t_2 = t_1 \end{cases} \tag{10}$$

where $\hat{\beta}$ is the least squares regression line [30]. Moreover, the interval-based properties of the TSF classifier enable robustness as the temporal features calculated over the interval features can capture the transient characteristics of a program.

B. Combining HPCs

While TSD exhibits advantageous properties by accounting for the temporal HPC order, we additionally explore employing multiple HPCs towards an improved performance. When analyzing different time series segments of an entire HPC series, these units are less in number compared to traditional Malware detection samples, where the data is scrambled. Scrambled data considers each unit separately rather than as a single series. However, time series segment samples contain more classification information since the trace is longer. Consequently, it is natural to explore how the utilization of more than one HPC can improve upon the accuracy as additional time series segments are incorporated. Since HPC data has a positive correlation when utilized in unison, employing them together provides an improved performance. Multi-feature utilization incorporates more information for the TSC to furnish improved predictions.

C. Sequential Time Series Framework

Malware that incorporates benign commands would culminate in current HMDs incurring a plethora of false positives. Since HMDs furnish a classification on individual samples, Malware with benign behaviors would incur similar HPC values to benign data. Furthermore, embedded devices that lack thorough profiling capabilities are unable to provide many hardware features. To address these issues, we propose Sequential Time

Series Detection (SEQ-TSD) for Malware detection, that utilizes *a single HPC* while maintaining a minimal false positive rate. We utilize time series classifiers in our proposed framework, so the temporal order of the HPC data is respected. Therefore, for Malware like *PNSCAN* which employs the benign *ping* command [6], the time series HPC traces would be different between the two, thereby lowering the amount of incurred false positives.

Figure 4 depicts the proposed SEQ-TSD when four classifiers are utilized. It should be noted that any number of classifiers can be utilized, albeit with a difference in performance. Each classifier in our framework utilizes a different time series length following an ascending order. Algorithm 1 provides an overview of the operational process of SEQ-TSD. The proposed methodology functions with the first time series classifier predicting the incoming testing data. If the confidence probability exceeds a unique predefined threshold, the prediction is taken. Otherwise, the next classifier will predict on that data. Similarly, the prediction is taken or the testing data is moved to the next classifier. When the last classifier is reached, if the model's required threshold is not met, a majority prediction vote is taken based on the confidence probabilities of all the classifiers.

By utilizing predefined thresholds for each classifier, the false positive rate is lowered. As an example, let us consider that HPCs are collected every one millisecond, and the time series lengths of the classifiers in Figure 4 are 5, 10, 15, and 20, respectively. For obvious malicious time series HPC traces, the first classifier would be able to identify them with a high confidence probability. Consequently, we specifically set the initial classifiers in our framework to utilize high thresholds to ensure that only these samples get classified. For samples that don't meet this threshold, they will propagate through the framework to classifiers employing longer time series lengths. The longer HPC traces provide more information, thereby enabling these models' decisions to have better confidence probabilities. In the case that none of the models' predictions are taken, the employment of a majority vote exploits ensemble advantages, where a single wrong model prediction does not subvert the accuracy.

Algorithm 1 Sequential Time Series Detection Application

Input: HPCs of Malware and Benign Applications, C Number of Classifiers to be Utilized in SEQ-TSD
Output: SEQ-TSD Accuracy

```
1: application(){
2: #define int $pass, fail$, array $input\hat{X}$, $pred\hat{Y}$
3: $input\hat{X}$ = Read HPC Trace
4: while $input\hat{X}$ do
5:     $pred\hat{Y}$, $threshold_n$ = $Classifier_n.predict(input\hat{X})$
6:     if $threshold_n >= Classifier_n.threshold$ then
7:         if $pred\hat{Y}_n$ passed then
8:             {pass++}
9:         else
10:            {fail++}
11:        $pred\hat{Y} \leftarrow$ empty Array
12:    else if $n == C$ then
13:        $\{Majority\_Vote(pred\hat{Y})\}$
14:        $pred\hat{Y} \leftarrow$ empty Array
15:    else if $n < C$ then
16:        {n++}
17:        $\{input\hat{X}$ = $Get\_Trace(n)\}$
18: output SEQ-TSD_accuracy = (pass / (pass + fail))
19: end
20: }
```

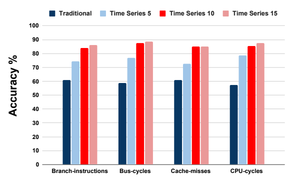
V. Experimental Results

A. Experimental Setup

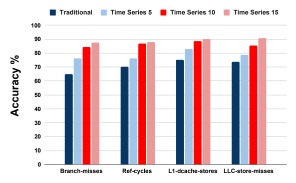
For our experiments in this paper, we collected HPC data from two different processors using the linux command *perf stat*. In this work, we have retrieved four HPC measures per *perf stat* command every *one millisecond* from the target processor. First, we collected HPCs from an Intel i5-4210U processor utilizing x86 architecture for 100 benign and 100 malicious programs. Next, we employed a *Raspberry Pi 3 Model B*, which utilizes an ARM Cortex-A53 processor, to collect HPCs for 300 benign and 300 Malware. The benign programs include various sorting and computational algorithms, in addition to the MiBench and Phoronix benchmark [11]. These benchmarks represent real-life applications that a regular user would use. We obtained the processor-specific Malware from Virusshare [10].

For all Traditional Malware Detection (TMD) experiments, the Random Forest (RF) classifier is employed. Prior research has shown that RF models consistently outperform alternative ML solutions [5], [33]. Therefore, we used this classifier for our experiments. This analysis can be extended to other classifiers as well, albeit with a difference in classification accuracy. For time series lengths greater than one (a sample consisting of consecutive HPC data), the Time Series Forest

(TSF) model was employed. We utilized the *Sktime* and *Scikit learn* libraries in *Python* for building the ML models. Both classifiers incorporate an 80:20 spilt with 80% of the data used for training and 20% for testing. To corroborate the advantages of time series detection (TSD), we encompass a wide range of HPCs. We specifically profile the HPCs *Branch-instructions*, *Bus-cycles*, *Cache-misses*, and *CPU-cycles* for the ARM Malware Dataset (AMD), and *Branch-misses*, *Ref-cycles*, *L1-dcache-stores*, and *LLC-store-misses* for the x86 Malware Dataset (XMD). Other HPCs could be utilized, albeit with a variation in performance.

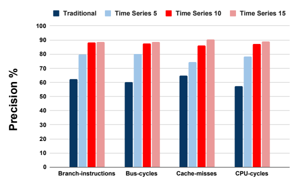


(a)

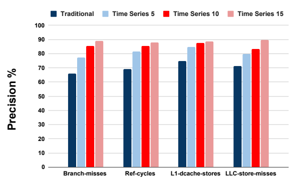


(b)

Figure 5. Accuracy of Varying Time Series Length Classification for (a) ARM Malware Dataset (AMD) and (b) x86 Malware Dataset (XMD).



(a)



(b)

Figure 6. Precision of Varying Time Series Length Classification for (a) ARM Malware Dataset (AMD) and (b) x86 Malware Dataset (XMD).

B. Time Series vs. Traditional Malware Detection

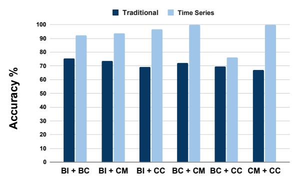
To demonstrate the advantages of time series-based classification for bolstering performance and minimizing the false positive rate, we evaluated both the AMD and the XMD using different time series lengths for *only a single HPC*. Figures 5(a) and 5(b) present our furnished experimental results in terms of classification accuracy. When the time series length is one, which is essentially TMD, the classification accuracy for AMD dataset is 60.89% for *Branch-instructions*, 58.91% for *Bus-cycles*, 60.85% for *Cache-misses*, and 57.33% for *CPU-cycles* with the average classification accuracy being 59.49%. For the XMD, the classification accuracy is 65.07% for *Branch-misses*, 70.32% for *Ref-cycles*, 75.17% for *L1-dcache-stores*, and 73.82% for *LLC-store-misses* with an average of 71.09% accuracy.

For AMD and XMD, utilizing a time series length of five showed significant improvement in each individual HPC's performance. For AMD, we observed an average accuracy increase of 16.21% for a mean absolute accuracy of 75.7%. Similarly, for XMD, the average accuracy improvement was 7.46% culminating in a maximum accuracy of 78.56%. We extended this experiment to include longer time series traces of lengths 10 and 15. We observed that an increasing trace length engenders an increase in classification accuracy. For AMD, we obtained an average improvement of 26.09% and 27.34% for time series lengths of 10 and 15, respectively. For XMD, the corresponding enhancements in accuracy are 15.28% and 17.96%. We obtained a maximum accuracy of 88.61% and 90.93% for HPCs *Bus-cycles* (AMD) and *LLC-store-misses* (XMD), respectively, with sample length 15.

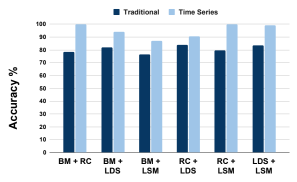
While the aforementioned results highlight the detection capabilities of TSD, any AVS is concerned with the false positive rate as well. A major predicament of TMD is the high false positive rate, as explained in Section II-B. Figure 6 shows the precision of the TMD and the TSD with varying time series lengths. The precision indicates the percentage of samples correctly identified as malicious; therefore, a high precision equates to a lower false positive rate.

Similar to Figure 5, we observed an increment in precision as the time series length increased. Initially, the average precision was 61.12% and 70.28% for AMD and XMD, for *a single HPC*, respectively, which corroborates the issue of the high number of false positives incurred in TMD. When utilizing a time series length of five, the overall precision for AMD was increased to 78.08%, and XMD improved to 80.86%. These results highlight the benefits of respecting the temporal order of the data as a small time series length was able to bolster the precision significantly. Our best precision results were obtained when employing a time series length of 15. The overall precision was 89.08% for AMD and 88.8% for XMD, thereby considerably reducing the false positive rate and improving confidence in the classifier.

Our experimental results show that when compared to TMD, TSD can furnish higher Malware classification accuracy and precision utilizing only *a single HPC*. Moreover, both metrics improve as the time series length increases. This is because the samples in TSD contain more temporal information. On the other hand, TMD incorporates scrambled data resulting in a lower precision, as Malware and benign applications will have analogous HPC values. Therefore, utilizing time series HPC traces enables developing robust models furnishing improved performance with minimal false positive rates as justified by our results in Figures 5 and 6.

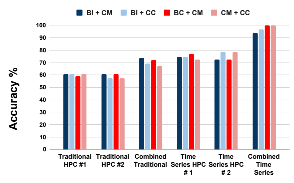


(a)

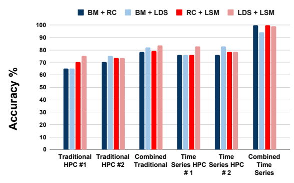


(b)

Figure 7. Accuracy of HPC Combinations for Traditional and Time Series Traces. Branch-instructions (BI), Bus-cycles (BC), Cache-misses (CM), CPU-cycles (CC), Branch-misses (BM, Ref-cycles (RC), L1-dcache-stores (LDS), LLC-store-misses (LSM)

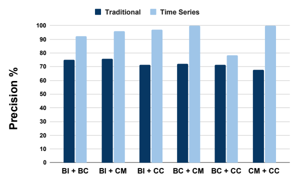


(a)

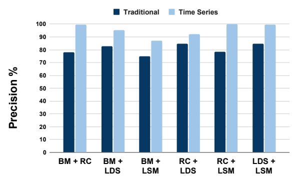


(b)

Figure 8. Performance Comparison between Single and Combinational Traditional and Time Series Traces. For each X HPC + Y HPC pair, X is represented in #1 columns, and Y is presented in #2 columns. Branch-instructions (BI), Bus-cycles (BC), Cache-misses (CM), CPU-cycles (CC), Branch-misses (BM, Ref-cycles (RC), L1-dcache-stores (LDS), LLC-store-misses (LSM)



(a)



(b)

Figure 9. Precision of HPC Combinations for Traditional and Time Series Traces. Branch-instructions (BI), Bus-cycles (BC), Cache-misses (CM), CPU-cycles (CC), Branch-misses (BM, Ref-cycles (RC), L1-dcache-stores (LDS), LLC-store-misses (LSM)

C. Time Series Detection by Combining HPCs

In this experiment, we explore the combination of HPCs, *i.e.*, utilizing *two HPCs* for the time series-based model. Combining HPC traces is beneficial, as multi-feature utilization enables extracting additional information for better classification performance. As explained in Section II-A, most processors allow monitoring up to four HPCs concurrently. We utilized the same HPCs from Figure 5 and evaluated the different combinations in groups of two HPCs for TMD and TSD. For TSD, we employed a sample length of five for the AMD and XMD. We constrain ourselves to utilize different HPCs per combination without repetitions.

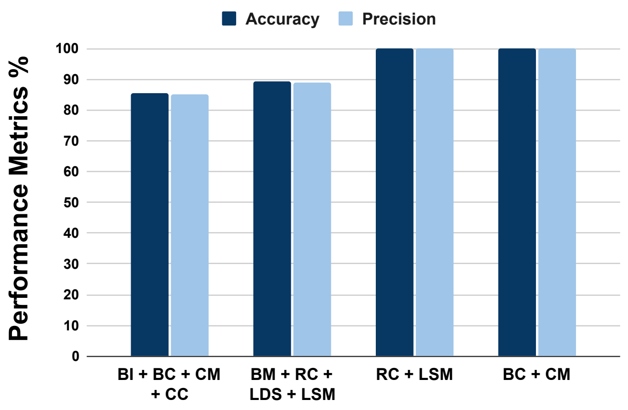


Figure 10. Traditional Malware Detection using 4 HPCs vs. Time Series Detection using 2 HPCs. Branch-instructions (BI), Bus-cycles (BC), Cache-misses (CM), CPU-cycles (CC), Branch-misses (BM, Ref-cycles (RC), L1-dcache-stores (LDS), LLC-store-misses (LSM)

From our results in Figures 7(a) and 7(b), combining HPCs bolstered the TMD accuracy in both datasets. When utilizing only one HPC, the average HPC accuracy for the AMD was 59.49%, but combining HPCs raised the average to 71.21%. Similarly, the XMD saw an improvement from an average HPC accuracy of 71.09% to 80.67%. Subsequently, we repeated our experiments for TSD. From Figures 5(a) and 5(b), the average time series HPC accuracy for length five was 75.71% and 78.56% for AMD and XMD, respectively. However, when analyzing the combinations of HPCs, these values increased to 93.22% and 95.11%.

Figure 8 shows a performance comparison of the individual accuracies of the utilized HPCs and the combinational results for TMD and TSD. Initially, the TMD accuracy of *Branch-misses* was 58.91% and 60.85% for *Cache-misses*. The utilization of these two HPCs furnished 72.16% accuracy. However, using a time series length of five, the individual accuracies were 77.04% and 72.62% for *Branch-misses* and *Cache-misses*, respectively. However, the combination of these features when using a TSC furnished a 99.97% accuracy. This is a 27.81% improvement over TMD, which corroborates the advantageous properties of TSD.

Additionally, we analyzed the precision of the various pairs for TMD and TSD, as shown in Figure 9. The overall precision for combinational TMD pairs was 72.22% and 80.64% for AMD and XMD, respectively. However, the average precision of the time series pairs was 93.97% for AMD and 95.67% for XMD, demonstrating an improvement of 15-20%. Therefore, even for multiple

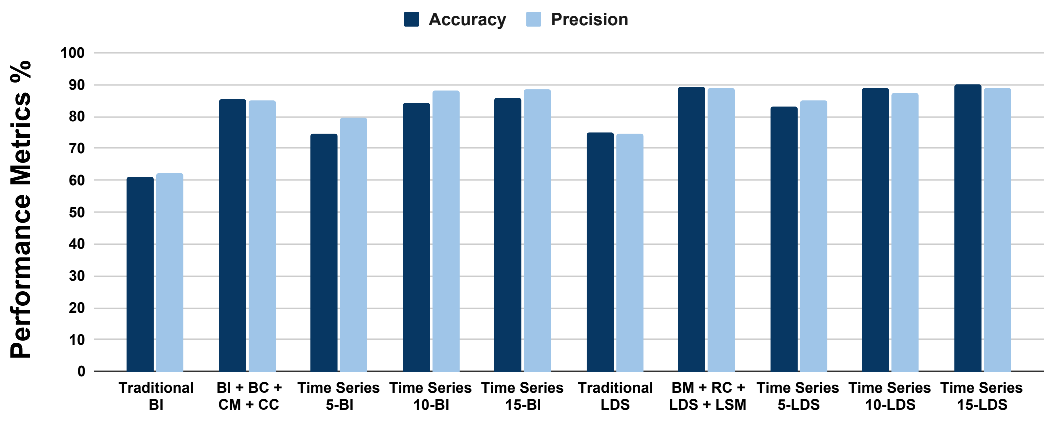


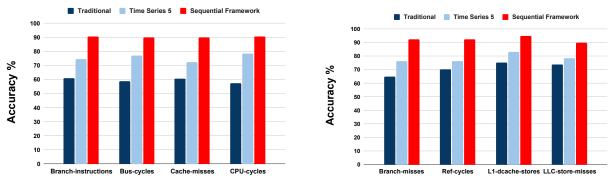
Figure 11. Performance Comparison between TMD using One HPC, TMD using Four HPCs, and TSD using One HPC. Branch-instructions (BI), Bus-cycles (BC), Cache-misses (CM), CPU-cycles (CC), Branch-misses (BM, Ref-cycles (RC), L1-dcache-stores (LDS), LLC-store-misses (LSM)

HPCs, the time series analysis can significantly improve the false positive rate.

To conclude our experiments for combining HPCs, we evaluated TMD utilizing four features versus TSD employing two, as shown in Figure 10. We utilized the same set of four HPCs from Section V-B for AMD and XMD. The accuracy and precision was 85.6% and 84.95% for AMD, and 89.27% and 88.79% for XMD. Since we can collect up to four HPCs concurrently, our results show the best performance that can be achieved with TMD. However, both accuracy and precision of TMD are still significantly less than the performance furnished from TSD, which had near perfect accuracy and precision employing only two HPCs. Consequently, these experimental results substantiate the advantages of multi-feature utilization and demonstrate the benefits of employing TSD.

D. *Malware Detection for Constrained Devices*

To corroborate the advantages of time series-based classification for devices with limited profiling capabilities, we analyzed the performance of TMD and TSD when utilizing different numbers of HPCs. Figure 11 shows the results for TMD using one and four HPCs, and TSD using *a single HPC* at varying time series lengths. We utilized the HPC *Branch-instruction* from AMD and *L1-dcache-stores* from XMD to demonstrate the performance improvement when different techniques are employed. For *Branch-instructions*, the TMD accuracy and precision was 60.89% and 62.32%, respectively. Furthermore, TMD's best performance is achieved when employing the four HPCs from Figure 10 concurrently. This improved the model's accuracy and precision to 85.6% and 84.95%, respectively. However, utilizing a time series length of 15 with *Branch-instructions* was sufficient for TSD to furnish a higher accuracy and precision of 85.97% and 88.46%, respectively. We obtained similar results for *L1-dcache-stores*. Consequently, TSD using TSCs provides better performance with less incorporated features. TMD is heavily reliant on the existence of hardware features and their profiling capabilities in processors and controllers as explained in Section II-B. The limited feature space in some devices can inhibit them from exploiting HMDs for ensuring system security. TSD provides a robust solution for constrained devices, thereby enabling them to incorporate HMDs for comprehensive Malware detection.



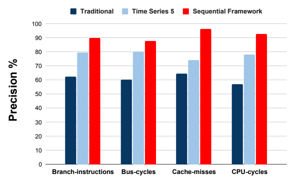
(a) (b)

Figure 12. Accuracy Comparison between TMD, TSD, and SEQ-TSD.

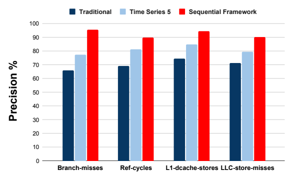
E. *Evaluating Sequential Time Series Framework*

In this section, we analyze the proposed SEQ-TSD, explained in Section IV-C, that utilizes *a single HPC*. We incorporated five TSFs into our framework, utilizing varying time series lengths of 5, 15, 25, 35, and 45. These sample lengths were chosen as they enable quick detection but permit sampling for longer periods to address more complex Malware. We built two frameworks, one each for AMD and XMD. For each SEQ-TSD, we evaluated using the HPCs employed in Section V-B, and include a performance comparison with TMD and TSD.

Our experimental results are presented in Figure 12. Previously, the average HPC accuracy furnished for the TSD of length five was 75.71% and 78.56% for AMD and XMD, respectively. However, our proposed method was able to boost this average accuracy by 14.7% and 13.94% to 90.41% and 92.5% for AMD and XMD, respectively. Achieving this performance with *a single HPC* is critical because this confirms the effectiveness of our presented framework for Malware detection with minimal feature overhead. In devices with limited hardware features, TMD would face a high false positive rate. However, our proposed methodology enables providing accurate detection by exploiting advantageous time series classifier and ensemble properties.



(a)



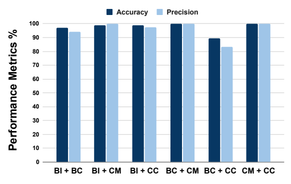
(b)

Figure 13. Precision Comparison between TMD, TSD, and SEQ-TSD.

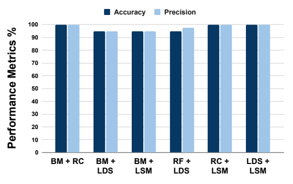
To further substantiate our claims of our proposed technique, we analyzed the precision of TMD, TSD, and SEQ-TSD. In Figure 13, the average precision of TMD for AMD and XMD was 61.12% and 70.28%. These precision values are not ideal for Malware detection, as the number of false positives would be high. For TSD, the average precision values of AMD and XMD were 78.08% and 80.86%. This is an improvement but the number of false positives would still be significant as roughly 22% of benign applications were erroneously identified as Malware. On the other hand, the proposed SEQ-TSD had an average precision of 91.83% and 92.64% for AMD and XMD, respectively. Consequently, the SEQ-TSD had an average false positive rate of 7.76%, which is a significant improvement over TMD.

Our technique was successful in bolstering the precision, thereby lowering the amount of false positives which are a major detriment in TMD. In Figure 13, our best result was obtained when employing HPC *L1-dcache-stores* which had an accuracy of 95% and a precision of 94.44%. Since the fundamental goal of our proposed method is to provide improved detection using *a single HPC* and minimize the false positive rate, the sequential time series capabilities of our framework are verified.

To conclude our experiments, we utilized the combinational pairs from Section V-C for multi-feature utilization in our SEQ-TSD as shown in Figure 14. The average accuracy and precision for AMD was 97.36% and 95.83%. For XMD, we obtained an overall accuracy of 97.5% and a precision of 97.91%. When analyzing the average time series performance metrics from Figures 7 and 9, the sequential model was able to improve the accuracy and precision by 4.14% and 2.7% for AMD and 2.39% and 2.23% for XMD. Moreover, when contrasting the furnished results with average measurement metrics of Figure 12, we observed an improvement of 7.09% accuracy and 6.08% precision for AMD. For XMD, we obtained a 5% and 5.27% increase in accuracy and precision, respectively. We conclude that utilizing multiple HPCs in conjunction with our proposed SEQ-TSD furnishes the best overall accuracy and precision. However, for low resource devices with minimal hardware profiling capabilities, SEQ-TSD still provides favorable Malware detection with a low false positive rate when employing only *a single HPC*.



(a)



(b)

Figure 14. SEQ-TSD utilizing Combinational HPCs. Branch-instructions (BI), Bus-cycles (BC), Cache-misses (CM), CPU-cycles (CC), Branch-misses (BM, Ref-cycles (RC), L1-dcache-stores (LDS), LLC-store-misses (LSM)

VI. Conclusion and Future work

While HPC-based Malware Detection has improved system security by addressing the challenges of AVS, the high false positive rate and profiling limitations of some devices remain a detriment. In this paper, we proposed utilizing TSCs to ensure the temporal sequence of the HPCs is respected. Our experiment results substantiate the advantageous properties of TSD over traditional TMD. Furthermore, we presented SEQ-TSD that uses *a single HPC* for Malware detection. Our experiment results show that our proposed technique is able to furnish a 95% accuracy with a minimal false positive rate of 5.56% when employing just a single HPC. Consequently, our methodology is suitable for embedded devices with minimal hardware registers. Additionally, we evaluated the ability of combining HPCs in conjunction with our SEQ-TSD, where we obtained an average 97.91% detection accuracy. In the future, we plan to extend our TSD to microarchitectural attacks such as Spectre, Meltdown, ZombieLoad, and Rowhammer.

References

[1] V. Lou, "Application behavior based malware detection," Aug. 17 2010, patent 7,779,472.

[2] M. Ozsoy, *et al.*, "Hardware-based malware detection using low-level architectural features," *IEEE Transactions on Computers*, vol. 65, no. 11, pp. 3332–3344, 2016.

[3] J. Demme *et al.*, "On the feasibility of online malware detection with performance counters," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 559–570, 2013.

[4] "Intel collaborates with microsoft against cryptojacking," https://www.intel.com/content/www/us/en/newsroom/news/intel-microsoft-scale-threat-detection-cryptojacking.htmlgs.zw9v32, 2021.

[5] B. Zhou *et al.*, "Hardware performance counters can detect malware: Myth or fact?" in *Asia Conference on Computer and Communications Security*. ACM, 2018, pp. 457–468.

[6] Z. Pan *et al.*, "Hardware-assisted malware detection using explainable machine learning," in *Conference on Computer Design*. IEEE, 2020, pp. 663–666.

[7] K. Basu *et al.*, "A theoretical study of hardware performance counters-based malware detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, no. 1, pp. 512–525, 2019.

[8] X. Wang *et al.*, "Reusing hardware performance counters to detect and identify kernel control-flow modifying rootkits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 3, pp. 485–498, 2015.

[9] M. B. Bahador *et al.*, "Hpcmalhunter: Behavioral malware detection using hardware performance counters and singular value decomposition," in *International Conference on Computer and Knowledge Engineering*. IEEE, 2014, pp. 703–708.

[10] "Virusshare.com," https://virusshare.com/, (Accessed on 10/23/2019).

[11] "Mibench version 1," http://vhosts.eecs.umich.edu/mibench, 2020.

[12] J. Durbin *et al.*, "Testing for serial correlation in least squares regression: I," *Biometrika*, vol. 37, no. 3, pp. 409–428, 1950.

[13] ——, "Testing for serial correlation in least squares regression iii." *Biometrika*, vol. 58, no. 1, pp. 1–19, 1971.

[14] V. Dalla, L. Giraitis, and P. C. Phillips, "Cran - package testcorr," https://cran.r-project.org/web/packages/testcorr/index.html, 2021.

[15] C. Malone *et al.*, "Are hardware performance counters a cost effective way for integrity checking of programs," in *Workshop on Scalable trusted computing*. ACM, 2011, pp. 71–76.

[16] X. Wang, *et al.*, "Malicious firmware detection with hardware performance counters," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 160–173, 2016.

[17] H. Sayadi *et al.*, "2smart: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection," in *Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2019, pp. 728–733.

[18] V. Jyothi *et al.*, "Brain: Behavior based adaptive intrusion detection in networks: Using hardware performance counters to detect ddos attacks," in *International Conference on VLSI Design*. IEEE, 2016, pp. 587–588.

[19] X. Wang *et al.*, "Hardware performance counter-based malware identification and detection with adaptive compressive sensing," *ACM Transactions on Architecture and Code Optimization*, vol. 13, no. 1, pp. 1–23, 2016.

[20] A. Tang *et al.*, "Unsupervised anomaly-based malware detection using hardware features," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2014, pp. 109–129.

[21] N. Patel *et al.*, "Analyzing hardware based malware detectors," in *Design Automation Conference*. IEEE, 2017, pp. 1–6.

[22] H. Sayadi *et al.*, "Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification," in *Design Automation Conference*. IEEE, 2018, pp. 1–6.

[23] I. o. Costa, "Constrained mixture estimation for analysis and robust classification of clinical time series," *Bioinformatics*, vol. 25, no. 12, pp. 6–14, 2009.

[24] S. Arik *et al.*, "Supervised classification-based stock prediction and portfolio optimization," *arXiv preprint arXiv:1406.0824*, vol. 2, no. 12, pp. 1–5, 2014.

[25] O. Patri *et al.*, "Discovering malware with time series shapelets," in *Hawaii International Conference on System Sciences*. IEEE, 2017, pp. 1–10.

[26] K.-H. Kim *et al.*, "Android malware detection using multivariate time-series technique," in *Asia-Pacific Network Operations and Management Symposium*. IEEE, 2015, pp. 198–202.

[27] Bahaa-Eldin *et al.*, "Time series analysis based models for network abnormal traffic detection," in *International Conference on Computer Engineering & Systems*. IEEE, 2011, pp. 64–70.

[28] P. Krishnamurthy *et al.*, "Anomaly detection in real-time multi-threaded processes using hardware performance counters," *IEEE Transactions on Information Forensics and Security*, vol. 15, no. 1, pp. 666–680, 2019.

[29] M. Löning *et al.*, "sktime: A unified interface for machine learning with time series," in *NeurIPS Workshop on Systems for Machine Learning*. NeurIPS, 2019, pp. 1–9.

[30] H. Deng *et al.*, "A time series forest for classification and feature extraction," *Information Sciences*, vol. 239, pp. 142–153, 2013.

[31] A. Bagnall *et al.*, "The great time series classification bake off: An experimental evaluation of recently proposed algorithms," *CoRR*, vol. 1602, pp. 1–16, 2016.

[32] "A brief survey of time series classification algorithms," https://towardsdatascience.com/a-brief-introduction-to-time-series-classification-algorithms, 2020.

[33] A. P. Kuruvila, *et al.*, "Analyzing the efficiency of machine learning classifiers in hardware-based malware detectors," in *Computer Society Annual Symposium on Very Large Scale Integration*. IEEE, 2020, pp. 452–457.