Detecting Malware and Ransomware using Hardware Performance Counters

Manaar Alam[1], Sarani Bhattacharya[1]*, Debdeep Mukhopadhyay[1], and Anupam Chattopadhyay[2]

[1]*Indian Institute of Technology Kharagpur*, [2]*Nanyang Technological University Singapore*

Introduction

Malwares are stealthy and hard-to-detect by performance analysis of programs. Various evasive mechanisms adopted by these malwares make them still succeed to execute, affecting precious data. Recent trends try to detect their presence by performing behavioural analysis using dedicated software to quarantine such malicious codes. Often observing high-level information, like system calls, have been evaded by smart malwares. In addition to it, identification and blocking of ransomwares at the earliest along with recovering the contents of the already encrypted files is an open challenge. It is proved that despite advancing in encryption systems, the prominent ransomwares leave a trait in the access of I/O and file-systems.

Advantages of HPCs

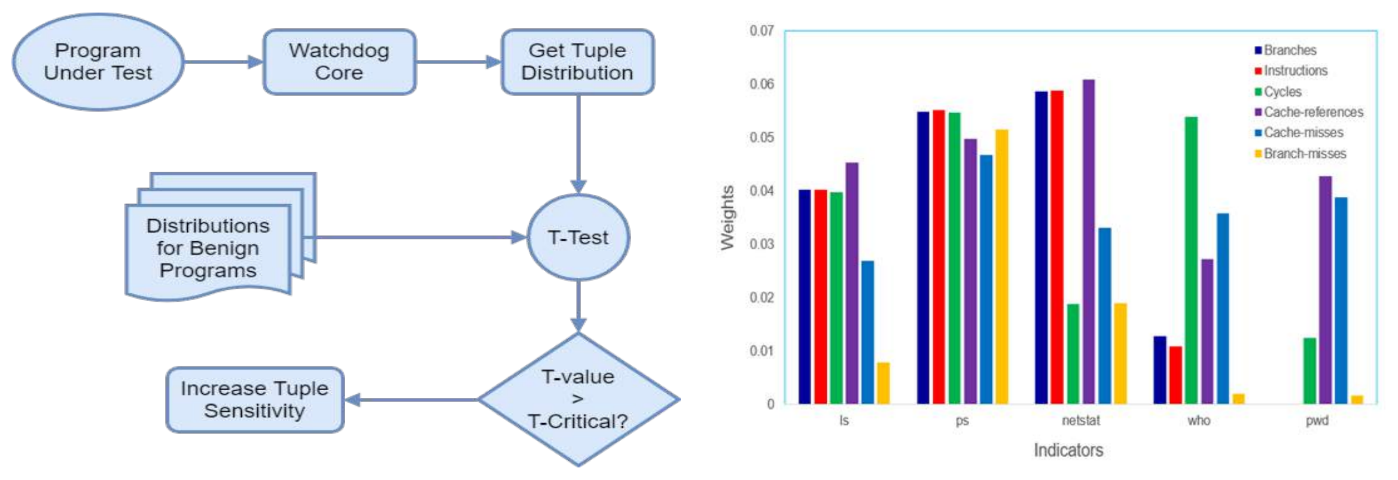
Hardware Performance Counters (HPCs) have certain advantages in detecting the presence of malwares as compared to other system call based detection methods.

1. HPCs provide more sensitive information of a system behavior than system calls.
2. HPCs are difficult to manipulate by the malware writer.
3. Easily accessible in most of the Linux based system.

Monitoring Features

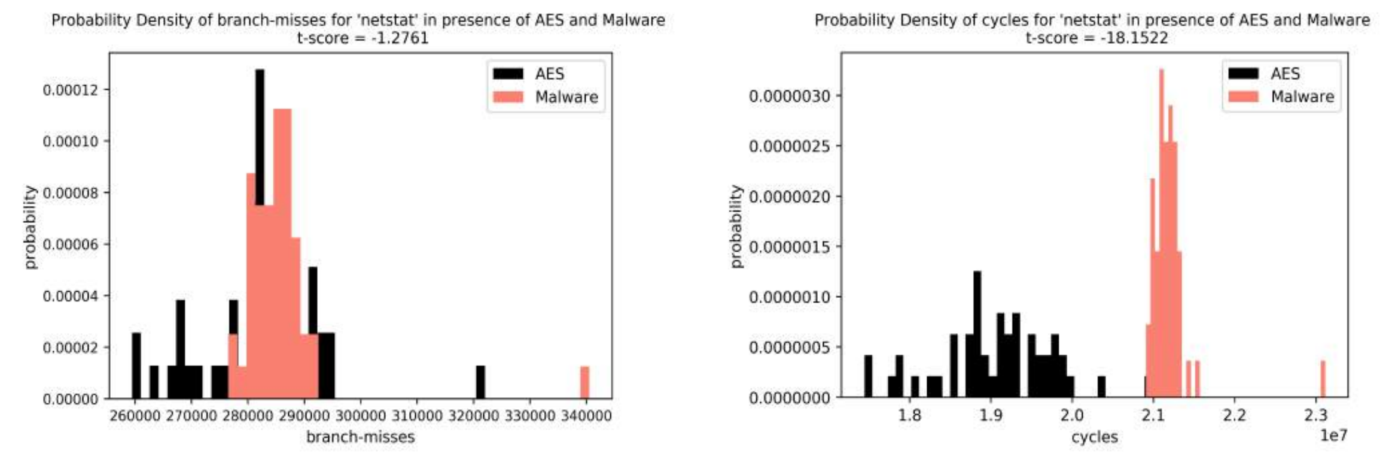
The feature monitored in the detection mechanism is a tuple of (*Indicator*, *Observer*).

- **Indicator**: Benign Operating System library executables, like ls, netstat, ps, who, pwd.
- **Observer**: Low-level hardware events, like cycles, instructions, cache-references, cache-misses, branches, branch-misses.



(a) (b)

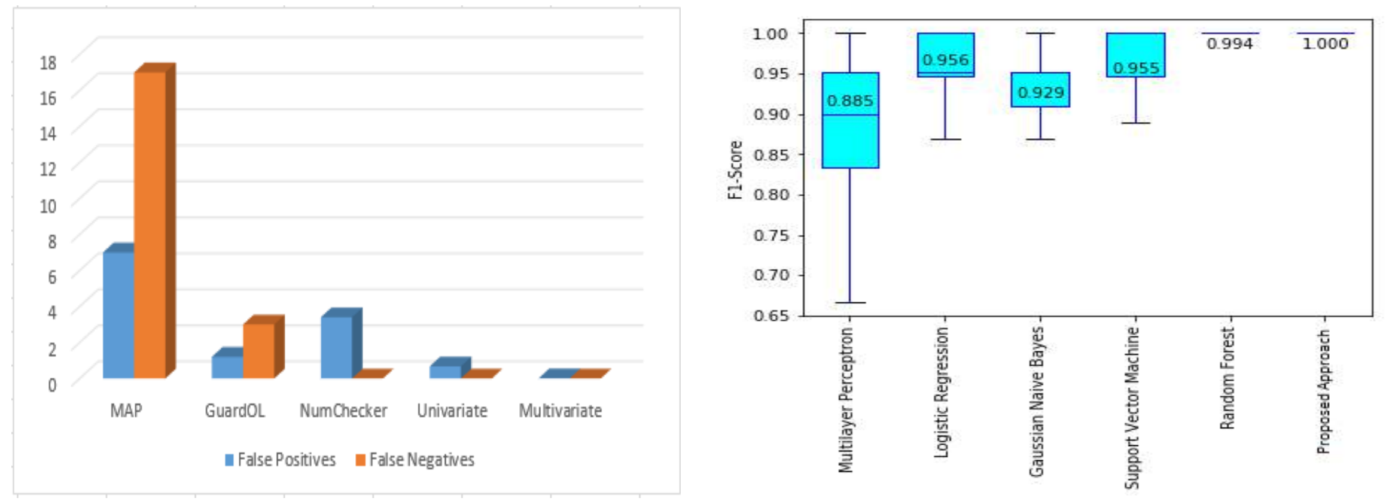
Figure 1: **a)** Assigning Sensitivity to each Tuple, and **b)** Monitored Tuples with their Sensitivity



(a) (b)

Figure 2: **Importance of Tuples**: Distribution of **a)** branch-misses and **b)** cycles for *netstat* in presence of benign *AES* and a Malware

Results



(a) (b)

Figure 3: Performance Comparison with **a)** State-of-the-Art and **b)** Different Machine Learning based approaches

Table 1: Average Training and Detection time for different models

	Training Time (mS)	**Detection Time (μS)**
Multilayer Perceptron	1434.1695	549.9807
Logistic Regression	839.0549	255.8256
Gaussian Naive Bayes	**14.1558**	425.2911
Support Vector Machine	1784.0227	**255.4266**
Random Forest	226.3765	2114.9878
Proposed Approach	378.8816	1777.9335

Table 2: Resource requirement on x86 and ARM processors (per second)

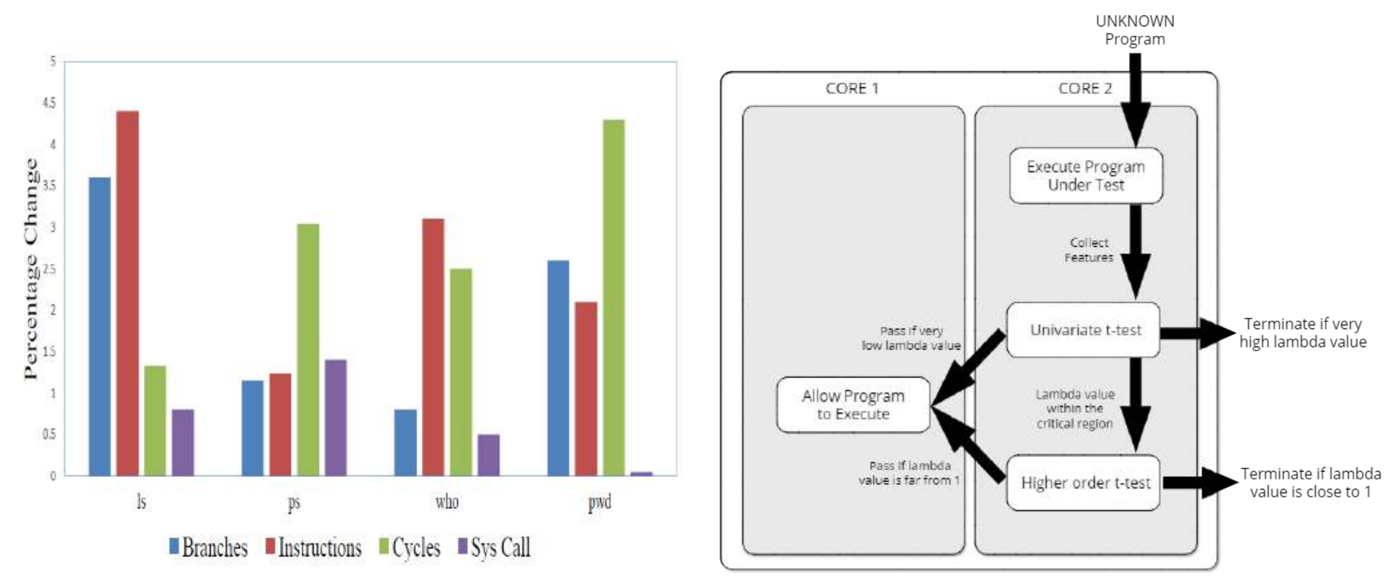
	%CPU Usage	%MEM Usage
x86	9.405%	0.1836%
ARM Cortex-A9	16.788%	0.7981%

Objectives

To highlight the role of low-level hardware events deduced from **Hardware Performance Counters** (HPCs) in detecting the existence of malware execution with the help of two case studies:

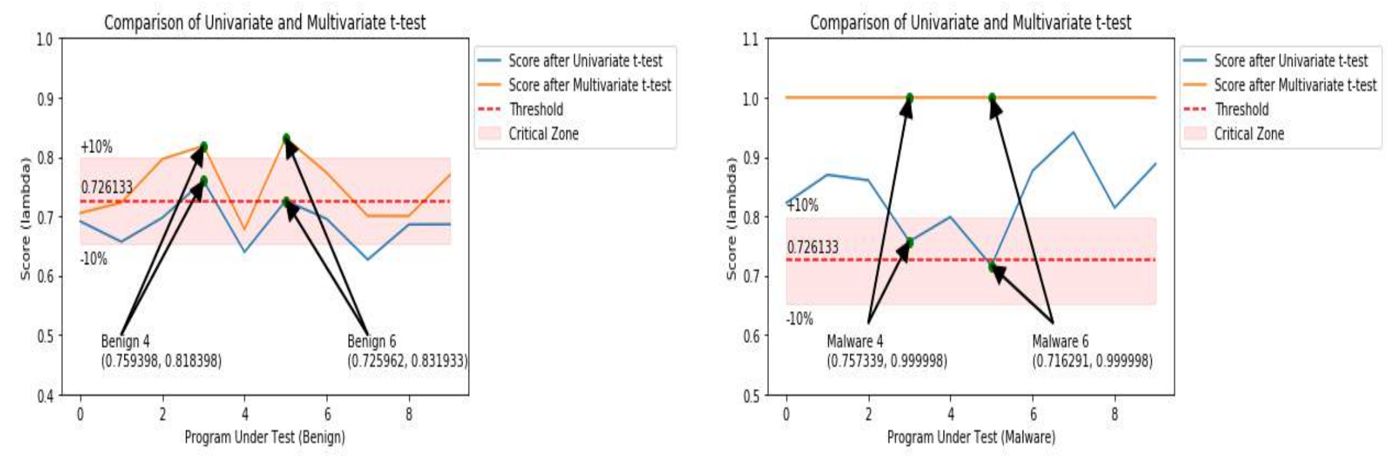
1. Developing a statistical lightweight tool, in the context of embedded platform, to evaluate the potential of a program under test of being a malware.
2. Developing a very fast detection methodology for popular ransomware on standard desktops.

Malware Detection


Percentage Change
ls
ps
who
pwd
Branches
Instructions
Cycles
Sys Call
UNKNOWN Program
CORE 1
CORE 2
Execute Program Under Test
Collect Features
Univariate t-test
Terminate if very high lambda value
Pass if very low lambda value
Allow Program to Execute
Lambda value within the critical region
Higher order t-test
Pass if lambda value is far from 1
Terminate if lambda value is close to 1

(a) (b)

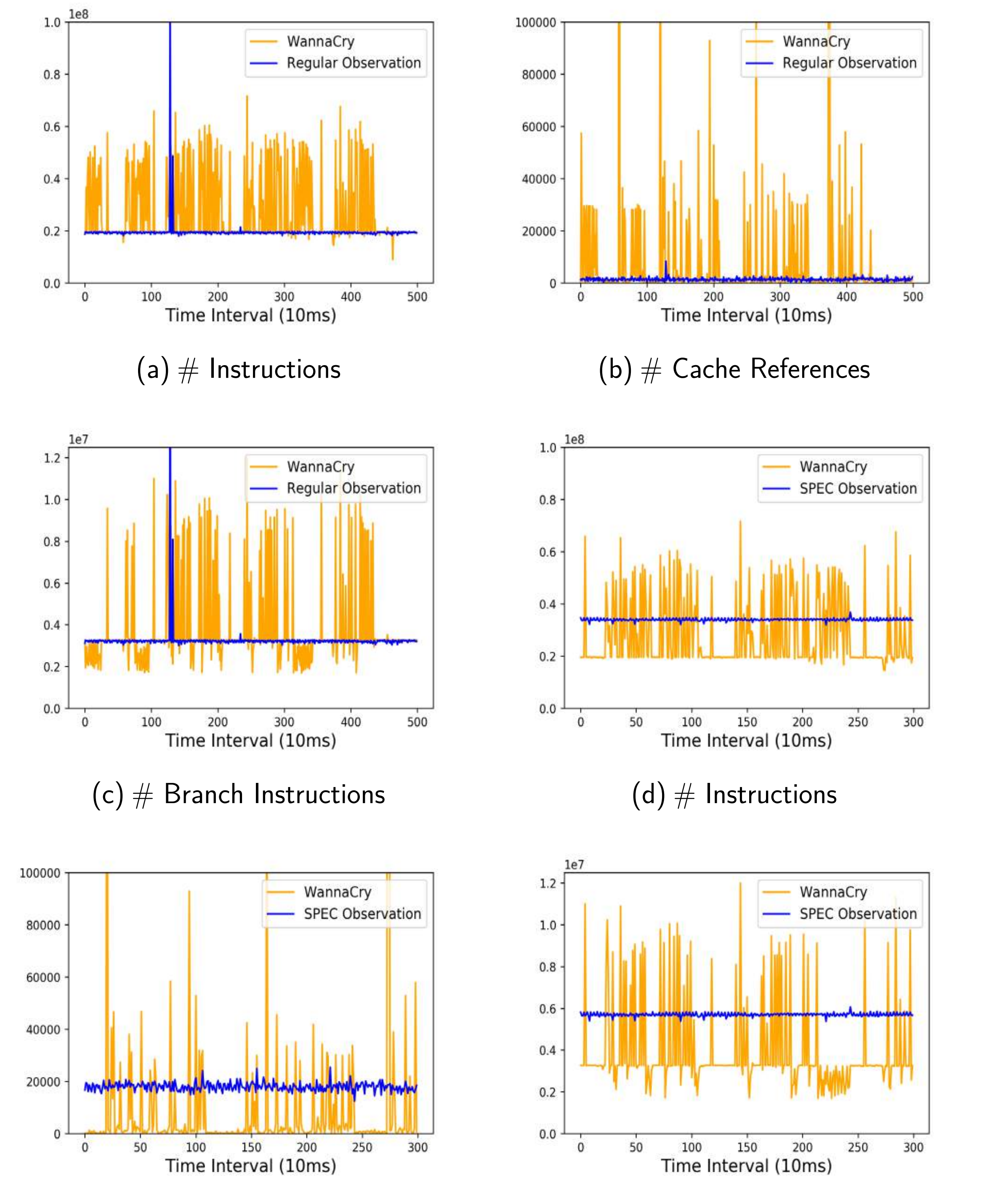
Figure 4: **a)** Advantage over *System Call* based approach, **b)** Control Flow of Malware Detector



(a) (b)

Figure 5: Removal of **a)** False Positives, and **b)** False Negatives using Multivariate *t-test*

Behaviour of Ransomware and SPEC



(a) # Instructions (b) # Cache References

(c) # Branch Instructions (d) # Instructions

(e) # Cache References (f) # Branch Instructions

Figure 6: Variation of HPCs in presence of Wannacry Ransomware (**a), b), c)**) and SPEC (**d), e), f)**)

Contribution

- A lightweight malware detector using hypothesis testing for Embedded Platform.
  1. A $[0, 1]$-metric λ to decide amount of *malware-ness*.
  2. Illustrating the importance of both system calls and HPCs with their relative sensitivity.
  3. A multivariate *t*-test to further improve the accuracy where the confidence on a univariate analysis is low.
- A Neural Network (NN) based anomaly detector to detect Ransomware in standard desktops.
  1. Learning an NN on normal time-series based behaviour of the system under observation with performance event statistics obtained from HPCs.
  2. Transforming time series to the frequency domain and understand the repeatability of data with the help of a second NN to remove false positives.

Ransomware Detection

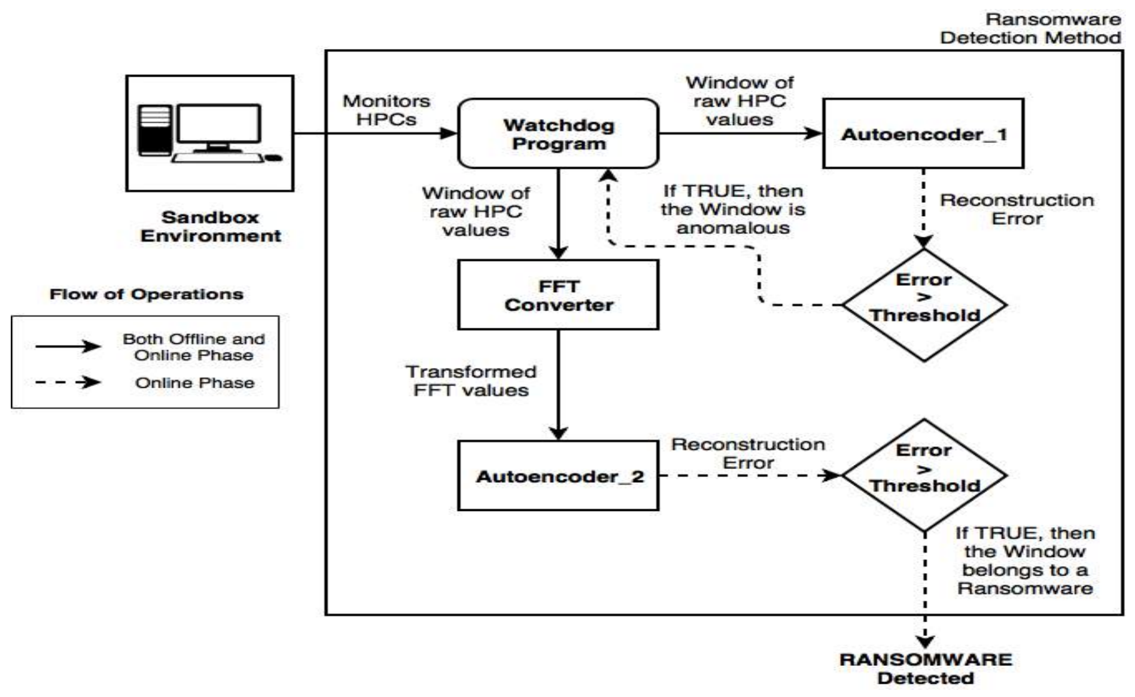
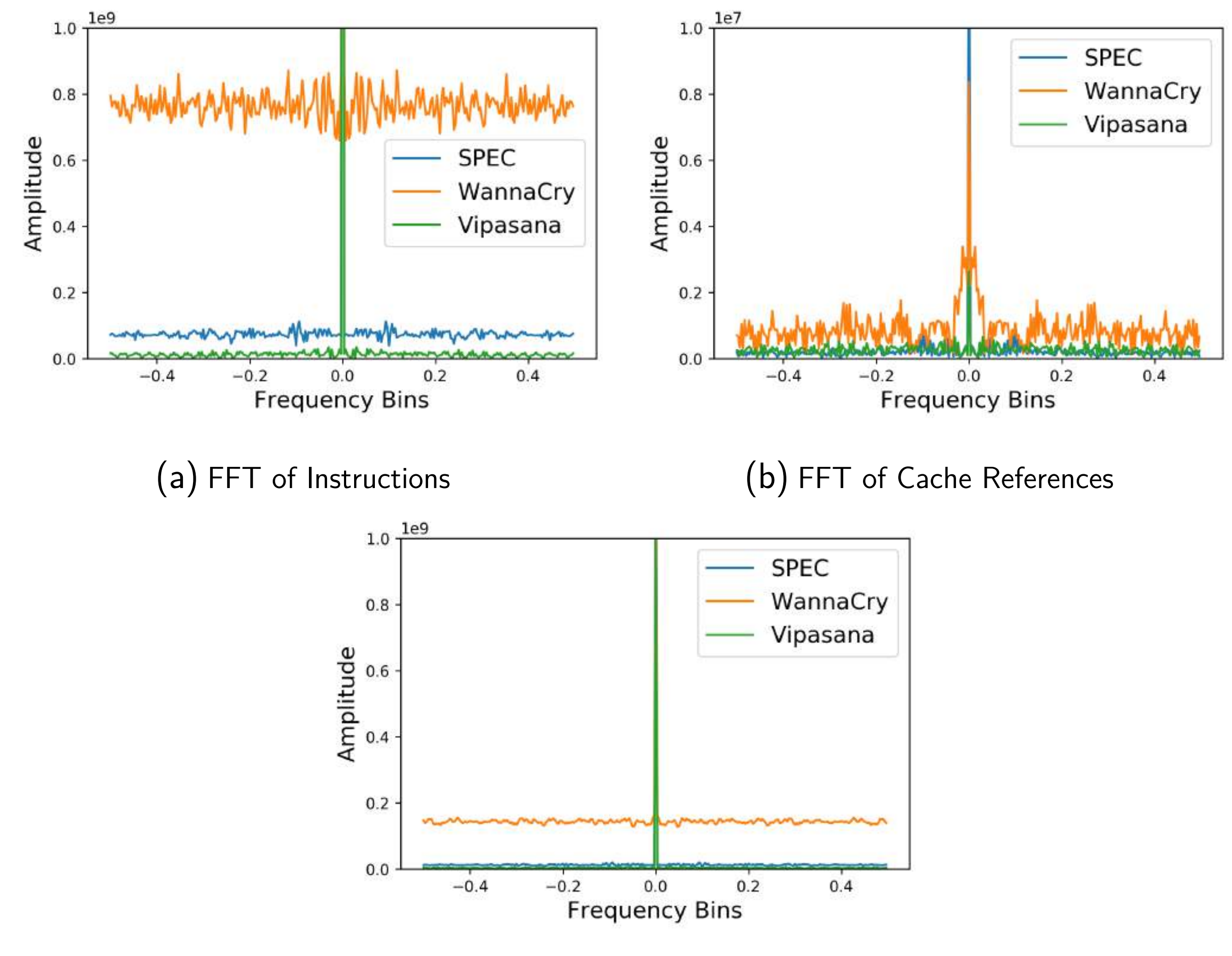


Figure 7: Control Flow of Ransomware Detector

Hardware events likely to change because of the symmetric and asymmetric key encryptions of ransomwares are monitored. Generally the symmetric encryption affects the cache based events while the asymmetric encryptions affect the instruction and branching events.

Behaviour in Frequency Domain

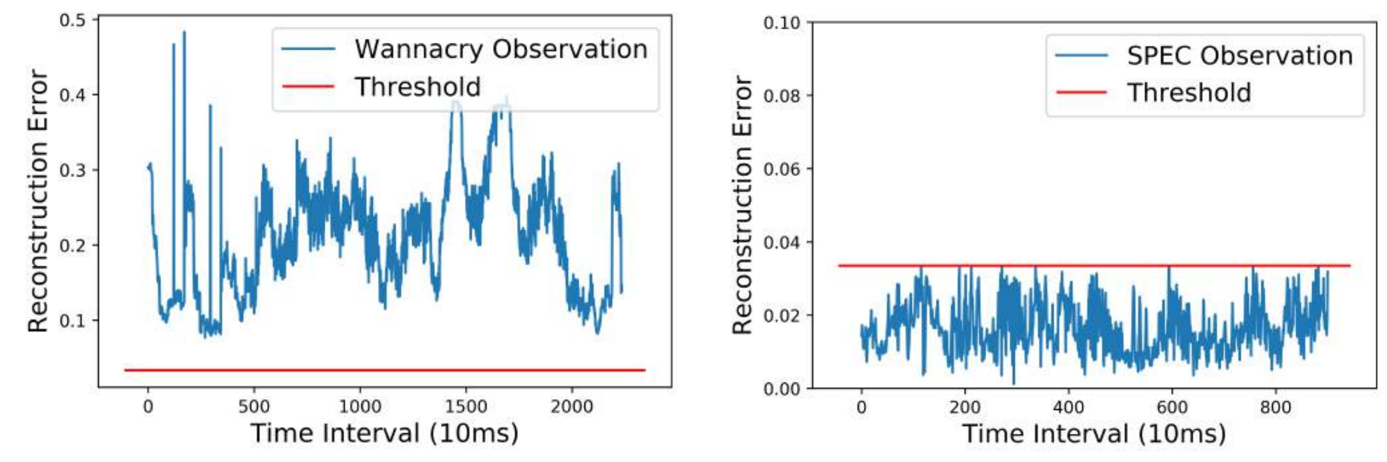


(a) FFT of Instructions (b) FFT of Cache References

(c) FFT of Branch Instructions

Figure 8: Variation of Amplitude in frequency domain of HPCs in presence of SPEC and Wannacry

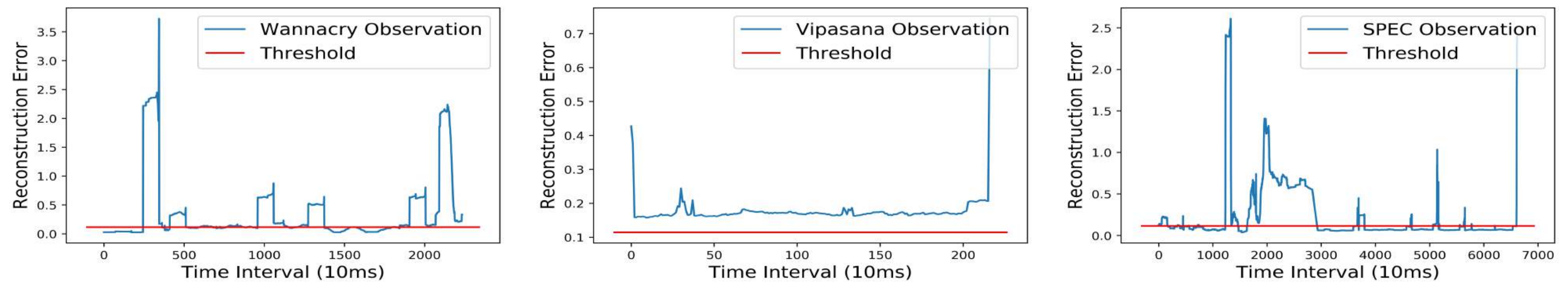
Second Autoencoder Behavior



(a) Wannacry (b) SPEC

Figure 9: Reconstruction Errors in the Second Autoencoder

First Autoencoder Behavior



(a) Wannacry (b) Vipasana (c) SPEC

Figure 10: Reconstruction Errors in the First Autoencoder