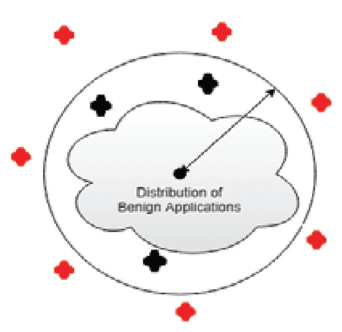
LAMBDA: Lightweight Assessment of Malware for emBeddeD Architectures

Research Objective

To propose a framework for runtime anomaly detection on embedded systems - The framework is capable of performing anomaly detection in a hierarchical manner (i.e. application level, operating system level and processor micro-architecture level) by harnessing the information available at various levels to detect malicious exploits.

Motivation



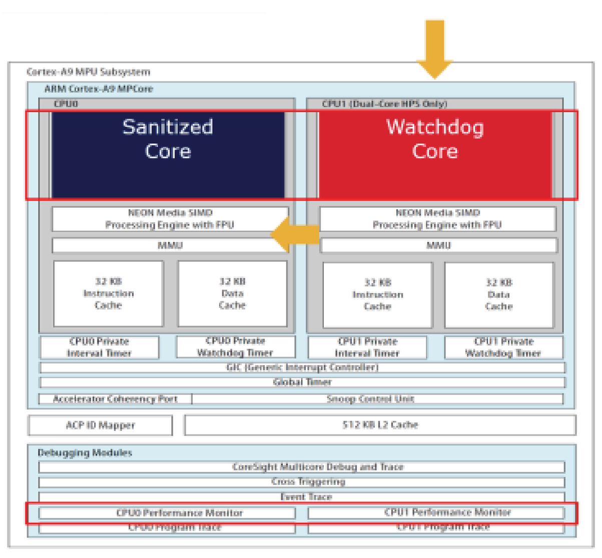
Measure the distance of a program under test from the characteristics of a given set of benign programs.

If the distance is less than a previously defined threshold value, the target program can be treated as a benign program otherwise the program is a malware.

Detection Approach

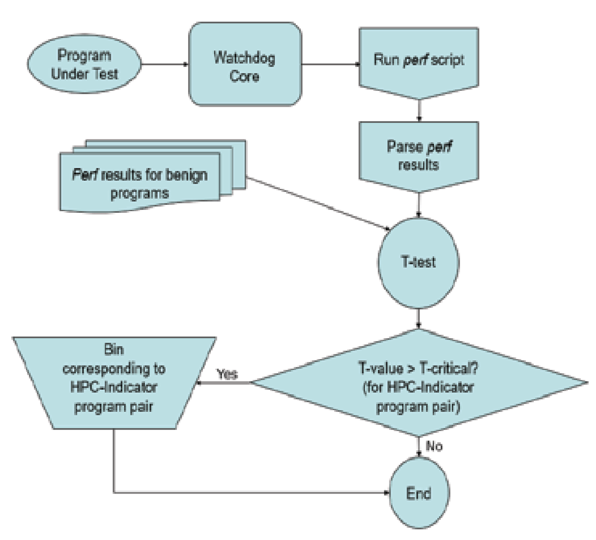
Use (HPC, Indicator) for monitoring.

Control Flow



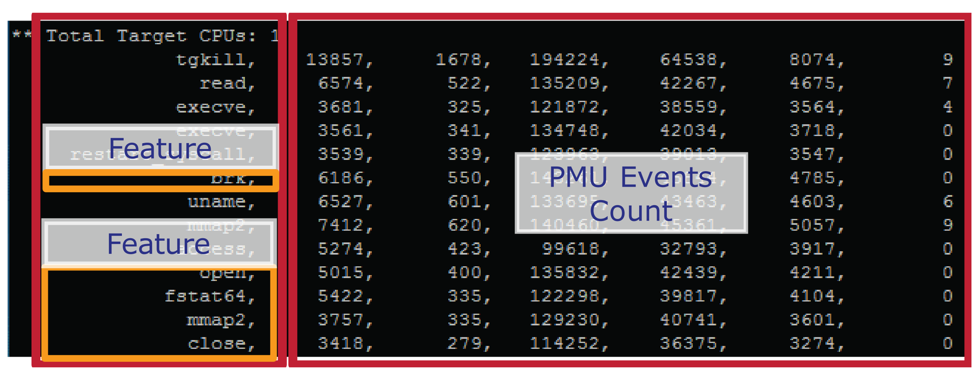
Dual core setup
- Watchdog Core: to monitor all the processes.
- Sanitized Core: to run non-malicious processes.

Creating Bins



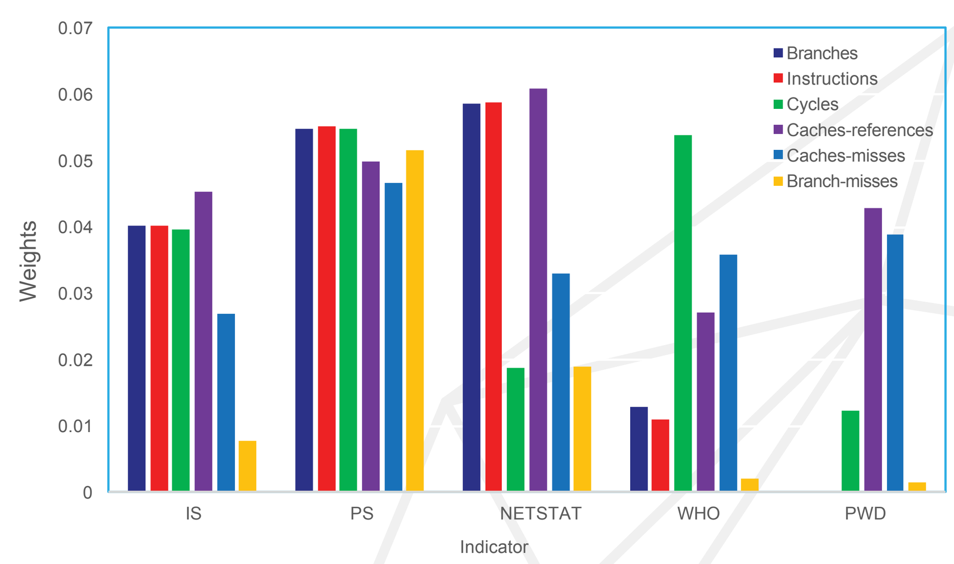
Emphasize on critical HPC-Indicator pair
- Performed to give more weightage on important performance counters and indicator programs.

Data Collection



Hardware Performance Counters
- More efficient to detect Kernel modifying rootkits.
- Easily accessible in most of the Linux based systems.

Normalized Weights

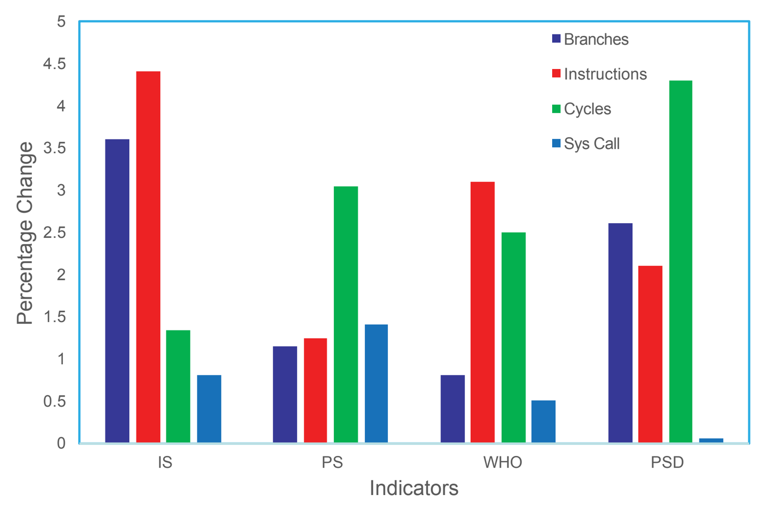


Determine weights during training
- Normalised weights help at runtime to determine distance of malware using statistical T-test.

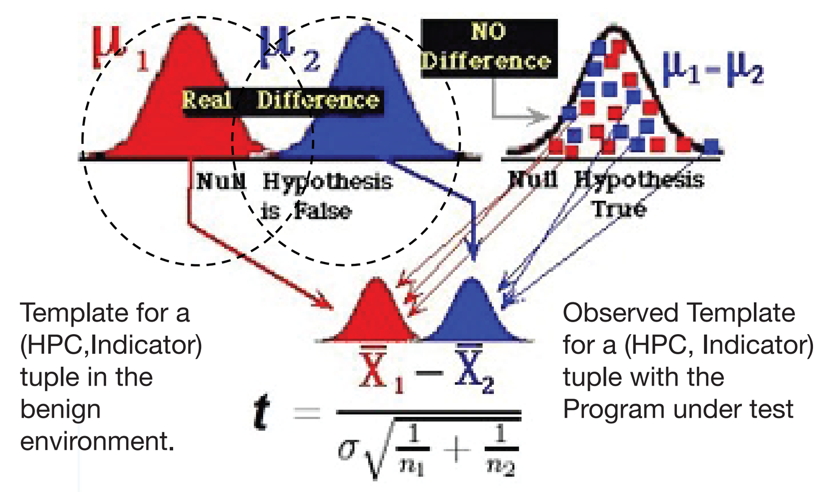
Scoring at Runtime

Calculate the amount of maliciousness of a program under test
- Create bins for program under test at runtime.
- Multiplication of the trained weights with these bins produces score for the program under test.
- Score greater than a pre-defined threshold value signifies the malicious behaviour of the program.

HPC provides more sensitivity & Increased Protection



Run time Statistical T-test



$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sigma\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

The null hypothesis of two equal means is rejected when the test statistic |z| exceeds a threshold of 4.5, which ensures a confidence of 0.99999.

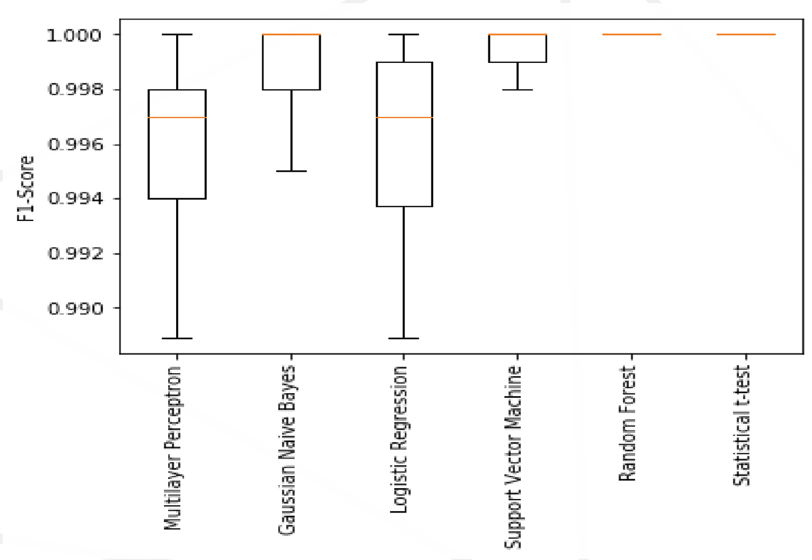
Advantages for HPC observation:

Difficult to manipulate HPC values by the malware.
More sensitive when observed in conjunction with system calls.
Results in better false positives and negatives

Advantages of the approach

- Monitoring only system calls doesn't provide any significant information but monitoring HPCs does. Significant changes can be observed in presence of malware.
- Enables semantic based malware detection.
- Supports multi-core environment.

T-test vs. Machine Learning based Detection



Various machine learning based approaches can also detect malware with significantly higher accuracy. However, cost of training and implementation overhead for embedded platforms is relatively high.

Models	Average Accuracy
Statistical T-test approach	**100%**
Multilayer Perceptron	99.73%
Gaussian Naïve Bayes	99.89%
Logistic Regression	99.69%
Support Vector Machine	99.98%
Random Forest	**100%**

Advantage over Training and Detection Time

	Model Building Time (in milliseconds)	Detection Time (in milliseconds)
Statistical ***T-test*** approach	43.7231	15.3789
Multilayer Perceptron	2036.9895	10.2458
Gaussian Naïve Bayes	**7.1782**	10.4336
Logistic Regression	200.8651	**4.0281**
Support Vector Machine	14.3887	5.1743
Random Forest	85.9585	91.2992

Random Forest algorithm achieves 100% accuracy, but both of its model building time and detection time is higher than statistical T-test due to its complex architecture.

Sai Praveen Kadiyala, Muhamed Fauzi Bin Abbas,
Yash Shrivastava, Sikhar Patranabis,
Manaar Alam, Debdeep Mukhopadhyay,
Siew-Kei Lam, Thambipillai Srikanthan