Whitelisting Approach Using Hardware Performance Counters in IoT Microprocessors

Ghadeer Almusaddar* Teruo Tanimoto* Takatsugu Ono* Smruti R. Sarangi** Koji Inoue*

* Kyushu University ** Indian Institute of Technology Delhi (IITD)

Dynamic Monitoring of Program's Execution

Approaches to secure computing systems can be done statically or dynamically. Static approaches are limited while dynamic approaches have both energy and computation overheads.

Dynamic based approaches can be implemented through whitelisting or blacklisting approaches

Factors	Whitelisting	Blacklisting
Targets	Valid programs	Invalid programs
Consideration	Behavior of valid programs (Known)	Behavior of invalid programs (Known or Unknown)
Zero-day Vulnerabilities	Risk is less than blacklisting	Issue if the behavior is unknown
Environment	Limited valid programs (Embedded/IoT)	Unlimited valid programs

Whitelisting Approach Using Hardware Performance Counters

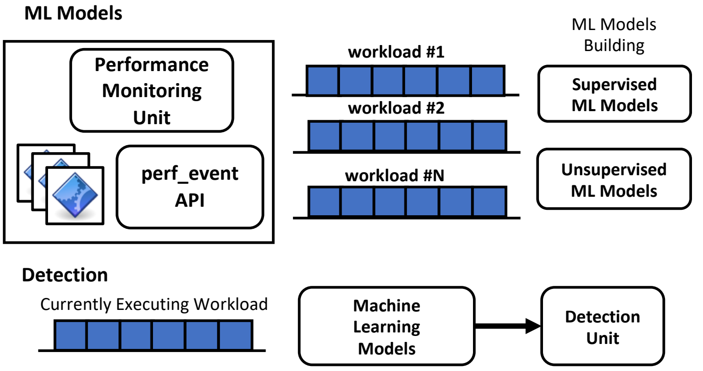
Blacklisting approaches (Malware detection) using hardware performance counters data and through machine learning approaches were explored.

What about whitelisting approach?

Through this work we are aiming into finding answers to the following questions:

- What are the events which should be used for the whitelisting? The number of counters in the performance monitoring unit (PMU) is limited.
- What about changing the input of whitelisted programs? Behaviors of invalid programs would change as we change the input.
- Collecting data at lower granularities introduces overheads. What is the performance of the whitelisting approach at different granularities?
- How can we adapt the whitelisting approach considering targeted environments? We base the collected data on user space only, how to deal with the case of multiple programs executing at the same time?

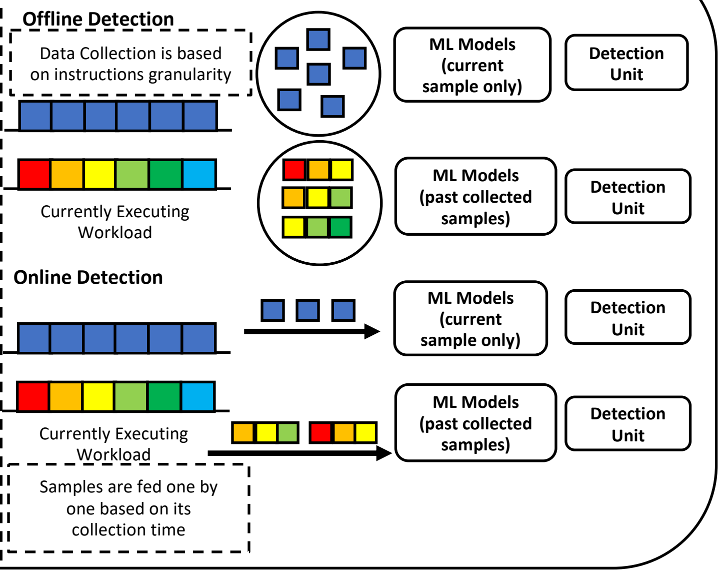
Whitelisting Approach Using Hardware Performance Counters



Events Selection is based on:
Repetitive Behavior
Not affected by Microarchitectural Characteristics

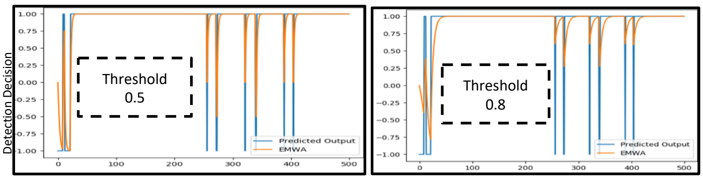
Retired Instructions, Immediate Branches, Comparison Branches, Load Instructions, Store Instructions, L1 Data Cache Accesses

Detection in Whitelisting Approach

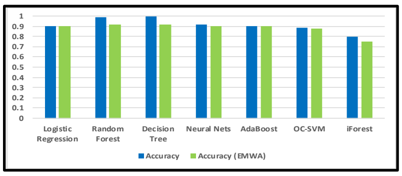

Offline Detection
Data Collection is based on instructions granularity
ML Models (current sample only)
Detection Unit
Currently Executing Workload
ML Models (past collected samples)
Detection Unit
Online Detection
ML Models (current sample only)
Detection Unit
Currently Executing Workload
ML Models (past collected samples)
Detection Unit
Samples are fed one by one based on its collection time

Exponentially Weighted Moving Average (EWMA)

When classifying for samples of whitelisted applications, there was a notable number of false positives (whitelisted samples detected as non whitelisted). In order to come over this challenge, we have used EWMA. Using EWMA, detection decision of the current sample is made considering past detection decisions.



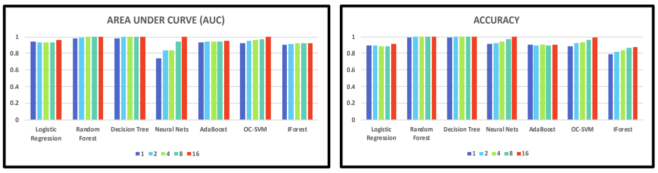
Detection decisions when using and without using EWMA



Accuracy when and without using EMWA for both supervised and unsupervised ML Models

Performance Evaluation

We have evaluated the performance of both offline and online detectors using both supervised and unsupervised ML Models. For these experiments, we have targeted EEMBC benchmarks.



AUC and Accuracy of both supervised an unsupervised ML models

Future Work

For Future work, we will look into the performance of ML models considering different inputs to whitelisted applications and how to adapt building ML models considering such case.

Also, we will look into approaches to adapt the whitelisted approach to targeted environments considering cases such as execution of multiple programs at the same time.