HPC-based Malware Detectors Actually Work: Transition to Practice After a Decade of Research

Charalambos Konstantinou, Xueyang Wang, Prashanth Krishnamurthy, Farshad Khorrami, Michail Maniatakos, and Ramesh Karri

***Abstract*—For the first time in 2011, researchers proposed using Hardware Performance Counters (HPCs) that are built into all processors as a pragmatic yet zero-cost solution for security. Online monitoring of HPCs can defend against malware using anomaly detection. Over the last decade, HPC-based malware detection transitioned from academic research through government transition to industry adoption. We outline this evolution by presenting use cases on critical power grid infrastructure protection as part of DARPA RADICS program, as well as describing how HPCs are utilized within Intel's HPC-based Threat Detection Technology (TDT), which is further used by Microsoft Defender for Endpoint.**

***Index Terms*—Hardware performance counters, malware detection, cybersecurity, embedded systems, transition to practice.**

I. Introduction

Modern and legacy processors include dedicated registers, called Hardware Performance Counters (HPCs), to track low-level microarchitectural events to monitor and measure events of processes executing on the system. Such events include, for example, number of branch-misses, CPU cycles, and instructions retired. HPCs are available on all modern processors including Intel, ARM, AMD, and Nvidia. Intel processors incorporate HPCs into the Performance Monitoring Unit (PMU). The types of events and the number of HPCs vary among different processor architectures. Recent CPUs from Intel can record over 1,000 events including tens of events related to L3 cache misses, DTLB load misses, and resource stalls. HPCs were introduced and primarily used to debug and tune application performance using detailed performance-related data that they yield with low overhead and high accuracy compared to software profilers.

Over a decade ago in 2011, researchers proposed using HPCs to evaluate static and dynamic integrity of program codes to detect malicious program modifications at load time and at run time, respectively [1]. Extending this paradigm, HPCs can be used for offensive and defensive purposes: HPCs can detect malicious firmware and software [2], [3], ransomware [4], and cryptojacking [5]. Recently, a systematization of knowledge survey paper summarized HPC-based detectors, attacks, and scenarios pointing out ideal use cases of HPCs as well as highlighting drawbacks and scenarios where HPCs do not provide the expected benefits [6]. After approximately 10 years of academic research, however, Intel productized HPC-based threat detection [7] and Microsoft is employing it in its contemporary security products [8].

In this paper, we trace important milestones in how HPCs have been used in research and present case studies of industry and government adoption. Fig. 1 presents the chronology of transition of research, on malware detection using HPCs, to practice. Since the introduction of HPCs for malware detection [1], HPCs have been used to detect variants of malware such as kernel-level rootkits [2], [3], firmware modifications [9], and malware in multi-threaded Cyber-Physical System (CPS) processes [10]. Studies also demonstrated the theoretical guarantees of HPC detectors [11]. The case studies of DARPA RADICS [12], Intel Threat Detection Technology (TDT) [13], and Microsoft Defender [8] show the transition of academic research to practice (TTP).

II. HPCs for Embedded System Security

Existing work in hardware-assisted technologies has shown that such techniques can be utilized for security purposes in terms of *malware prevention and detection* purposes. For instance, hardware-based Trusted Execution Environments (TEEs) including Intel Software Guard Extensions (SGX), ARM TrustZone, and AMD Secure Encrypted Virtualization (SEV), built into computing and server platforms from different hardware 'industry' vendors, offer a secure execution space that can *prevent* malware and provide a higher level of security for trusted applications running on the computing device. There are also 'academic' efforts towards providing hardware-assisted root-of-trust allowing attestation and secure task loading at run-time, e.g., TyTan [15], C-FLAT [16], Keystone [17], etc. As for hardware-assisted malware *detection*, existing work has been focused on collecting hardware traces from different avenues, such as the Embedded Trace Buffer (ETB) and HPCs [18]. Among hardware-based prevention and detection solutions, a few of them apply to embedded systems due to the overlooked security risks and critically the high costs for hardware modifications. In this work, we focus on HPCs to track microarchitectural events which are utilized towards malware detection.

Since HPCs are available in all modern processors, including Intel x86 and x86-64, ARM, MIPS, PowerPC, and Nvidia GPUs, they can yield near zero-overhead approach to count hardware events of applications running on the platform. Therefore, HPCs offer an attractive and flexible capability to verify application integrity with negligible performance

C. Konstantinou is with the Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Saudi Arabia.
E-mail: charalambos.konstantinou@kaust.edu.sa

X. Wang is with Alibaba Group, Shanghai, 201210, China. E-mail: wangxueyang.wxy@alibaba-inc.com

M. Maniatakos is with New York University Abu Dhabi, Abu Dhabi, 129188, UAE. E-mail: michail.maniatakos@nyu.edu

P. Krishnamurthy, F. Khorrami, and R. Karri are with the Department of Electrical and Computer Engineering, New York University Tandon School of Engineering, Brooklyn, NY, 11201, USA. E-mail: {prashanth.krishnamurthy, khorrami, rkarri}@nyu.edu

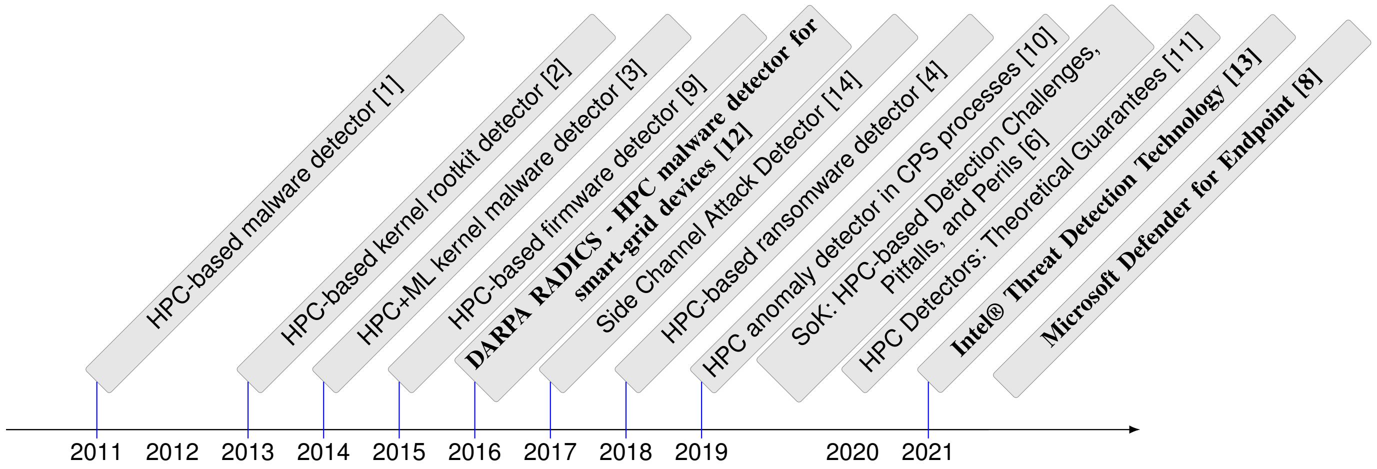


Fig. 1: Chronology of transition of research on HPC-based malware detection to practice.

overhead. While it is not possible to read all available HPCs at the same time, one can time-multiplex the measurements to read more HPC measurement streams, albeit with the overhead of multiplexing. Code execution can be characterized, for instance, by total occurrences of hardware events and by temporal patterns and relations among events. NYU researchers pioneered the use of HPCs to detect malicious modifications [1] rootkits [19], [20] and firmware modifications [9], [21].

1) ConFirm: In 2015, researchers extended HPC-based malware detection to detect firmware modifications in embedded systems using HPCs [9]. The motivation for ConFirm is that embedded devices are integrated in several domains including power grid, home and automation networks, and smart/connected cars. These devices are constrained in terms of performance and resources and hence cannot employ the same heavyweight security measures used in general purpose computers. ConFirm is a low-cost technique HPC-based technique to detect malicious modifications in the firmware of embedded control systems. The ConFirm study evaluated the detection capability and performance overhead on various real-world firmware running on ARM and PowerPC embedded processors. ConFirm was the first work to introduce HPCs to secure the firmware both as a design-for-security concept and as an add-on feature. It was also the first evaluation of an HPC-based security scheme on real-world firmware images used in embedded devices of critical infrastructure.

Before ConFirm, numerous mechanisms were proposed to detect malicious firmware. These mechanisms required either extra hardware components (e.g., Trusted Platform Module) or complex verification methodologies, impacting resource-constrained embedded devices (e.g. computation resources, communication bandwidth, power consumption, and memory use). ConFirm overcame these limitations by observing that a program is a sequence of various types of instructions which during execution can be monitored by low-level hardware events. The behavior of the firmware running on resource-constrained embedded systems, can be uniquely characterized by using HPCs at run time. Moreover, one can monitor the relationship between the counts of the different events. Finally, ConFirm does not need extra hardware components/ports for deployment to legacy devices supporting HPCs.

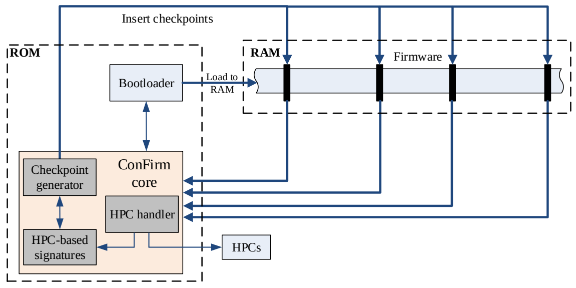


Fig. 2: ConFirm has three components: a module that inserts checkpoints into the monitored firmware, an HPC handler that drives HPCs, and a database that stores HPC-based signatures [9].

ConFirm is a host-based tool that leverages built-in hardware features (HPCs) to detect malicious modifications in embedded firmware. The high-level structure of ConFirm HPC-based monitoring is shown in Fig. 2. The work extended a legacy bootloader with ConFirm with three components: (a) an insertion module that places checkpoints in the monitored firmware, (b) an HPC handler that drives and collects HPC statistics, and (c) a database that stores valid HPC-based signatures. These components are stored in write-protected non-volatile memory to prevent attacks from compromising ConFirm while still allowing authorized updates. Comparing with software solutions, HPC-based solutions rooted in hardware have lower performance overhead and is tamper-resistant.

A. Follow-up Additional Academic Research

Researchers explored a new direction for hardware-based security in embedded systems by reusing hardware features (HPCs) for detecting malicious firmware and software modifications. This research was extended by researchers along many directions and systematized in [6]. We outline four emerging directions in HPC-based security research.

1) *HPC-based monitoring for security of CPS:* HPCs can be used for real-time monitoring of software running on embedded CPS processors [10]. They can apply to detect anomalies in power grid CPS [12], [19], [22].

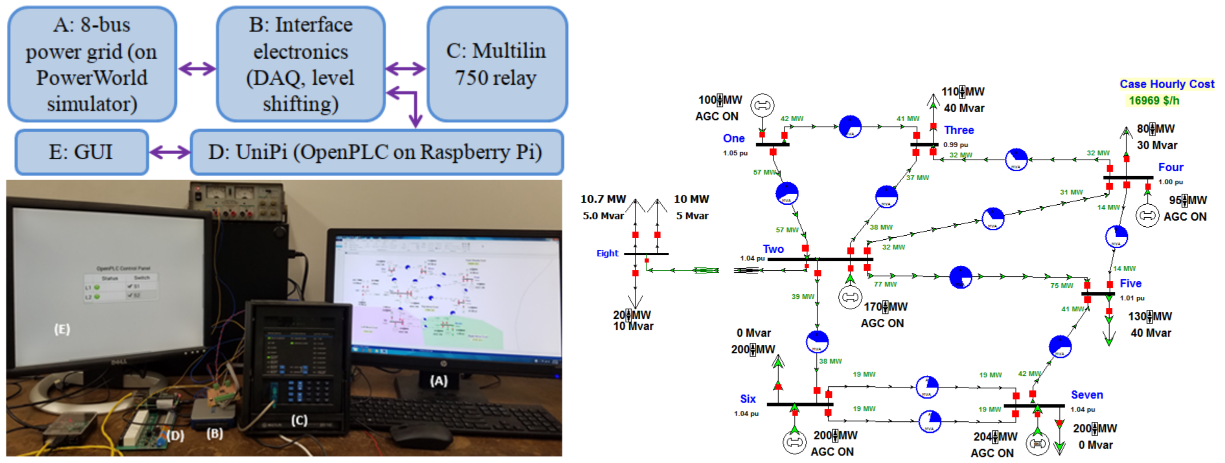

A: 8-bus power grid (on PowerWorld simulator)
B: Interface electronics (DAQ, level shifting)
C: Multilin 750 relay
E: GUI
D: UniPi (OpenPLC on Raspberry Pi)
(E)
(A)
(D)
(B)
(C)
Case Hourly Cost
16969 $/h
One
Two
Three
Four
Five
Six
Seven
Eight
AGC ON

Fig. 3: HIL power grid testbed: (A) a power grid simulator for an 8-bus power grid (shown on the right side), (B) a data acquisition board with level shifting, (C) GE Multilin 750 relay, (D) UniPi SCADA PLC emulator; Raspberry Pi that controls the OpenPLC, (E) GUI terminal to monitor relay status using operator interface running on another Raspberry Pi. UniPI communicates with the GUI via Modbus.

2) *Using HPCs to detect different types of attacks:* Repurposing HPCs for detecting malicious firmware has inspired researchers to explore the feasibility of using HPCs for detecting attacks such as return-oriented programming, ransomware, and side-channels [4], [14]. HPCs can measure mis-predicted return events to detect ROP attacks at run time. Profiling HPCs can detect micro-architectural side-channel-attacks [14]. HPCs can detect ransomware [4].
3) *Feasibility of other built-in hardware components for security:* Modern computer systems have on-chip sensors including thermal, voltage, and frequency sensors and associated reporting interfaces. Readings from these sensors correlate to the behavior of running programs and can be adapted for security monitoring. Combining thermal profiles of processors with HPCs can detect malicious changes due to software and hardware attacks [23].
4) *HPCs can be abused as a security backdoor:* While HPCs can monitor malicious behavior of a program, they also open an avenue for attackers to collect security-sensitive information. Concerns have been raised that HPCs might be abused as a security backdoor. Research efforts have been published along this line. Spisak [24] introduced a hardware-assisted rootkit on ARM and Intel x86-64 architectures. This rootkit allows an attacker to redirect control flow to malicious code by using HPCs count specific events. Alam *et* al. [14] present a micro-architectural side-channel attack by analyzing HPC counts when executing encryption algorithms.

III. Transition to Practice Case Studies

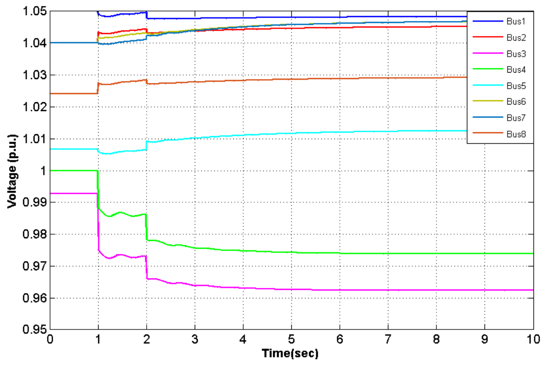
A. DARPA Rapid Attack Detection, Isolation and Characterization Systems (RADICS)

The DARPA RADICS program developed systems to restore power following a cyber-attack on the power grid [12]. DARPA assembled several teams with over 100 participants to solve this critical problem. SRI led the Threat Intelligence for Grid Recovery (TIGR) with NYU on the team [25]. RADICS kicked-off in August 2016 and ended in February 2021. NYU's HPC-based malware detector was part of a defense in depth. Red-team and blue-team rehearsals and deployment on a small-scale substation testbed established effectiveness of the malware detector. DARPA used a substation on Plum Island, off Long Island, NY, as a close-to-live testbed. RADICS teams demonstrated technologies with red-teams injecting malicious code and blue-teams detecting malware and restoring substation devices to create the crank path to restore power[1]. The exercises tested the anomaly detectors in as close to real operational setting as possible since it is infeasible to intentionally bring down all/part of a real power grid or inject malware into operating controller devices for testing.

1) HIL Power Grid Testbed: We implemented some of the RADICS red-team attacks on an OpenPLC controller in a Hardware-In-the-Loop (HIL) testbed in Fig. 3. This HIL testbed emulates an 8-bus power grid in a PowerWorld simulator that interfaces with a physical GE Multilin 750 relay. The relay is controlled through UniPi, a Raspberry Pi single board computer with an additional board providing Input/Output (I/O) interfaces for the PLC including analog and digital I/O. Multilin 750 has analog I/O interfaces for monitoring and RS232, RS485/RS422, and Ethernet ports. A graphical user interface (GUI) is implemented on a second Raspberry Pi using the pvbrowser process visualization browser software. Via this GUI, the operator opens/closes the relay. The operator terminal and the UniPi are connected via Ethernet and communicate using Modbus protocol. The command signal of the OpenPLC passes through a data acquisition board to change status of the relay in the PowerWorld simulator. The relay protects and controls feeders on solidly grounded, high impedance grounded, or resonant grounded systems.

2) A Stealthy Attack on Power Grid: The interplay of cyber and physical components in the power grid CPS allows an attack to exploit cyber vulnerabilities to impact the physical power grid processes [22]. PLCs are common in all CPS

[1]https://www.businessinsider.com/darpa-runs-mock-cyber-attacks-on-small-government-owned-island-2019-5



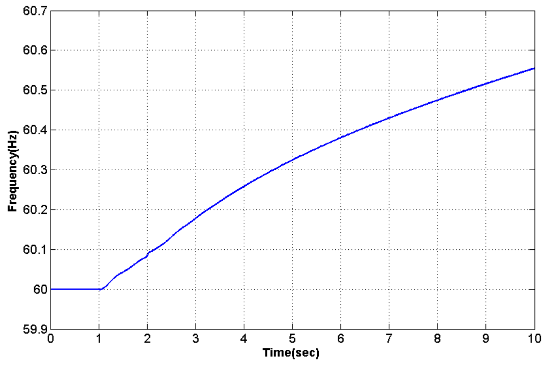


Fig. 4: (a) Voltage on buses when relays on L_{23}, L_{45} open at $t = 1, 2$ s. (b) Frequency when relays on L_{23}, L_{45} open at $t = 1, 2s$.

including the power grid. While networking these components benefit maintenance and upgrade, networking admits remote attacks on physical processes [26]. An attacker can exploit vulnerabilities in the computation, communication, and network nodes. These include vulnerabilities in the supply chain where the malicious code is "designed" into the controllers, remaining latent until triggered. An attacker may (1) modify the controller logic to impact stability, performance, or safety of the physical process, (2) spoof information from sensors, spoof outputs of actuators, or communication between operator control stations and the controllers, and (3) ex-filtrate sensitive data, add backdoors, or give access to unauthorized users.

In one of the attacks on power grid, a stealthy rootkit enables malware in embedded controllers. The attack seeks to undermine stability, efficiency, and safety of the grid. The malware injects a dynamically loaded library into the controller software to overwrite commands from an operator to a relay in the grid. Using open-source kernel- and user-space rootkit libraries, the attacker can create a daemon to replace the OpenPLC controller process with a process that has a malicious library hooked to it. This malicious library hooks into the process to override crucial I/O routines in the controller code. Further, it sends modified operator commands to the physical I/O and incorrect status messages to the operator control station. We validated the attack in a HIL testbed shown in Fig 3. The adversary can modify a process that runs the control logic on a PLC. This control logic is specified via a Structured Text program or a graphical ladder logic and is loaded onto the PLC as an executable. In one embodiment, the adversary gains unauthorized access to the PLC using a vulnerability in the network protocol used to program the PLC and implants the malware. While the attack spoofs actuator commands, it can spoof sensor readings, modify variables and logic controller altering control behavior and exfiltrate sensitive data [27].

3) Effects of Attack on the Power Grid: An informed attacker can calculate the betweenness index and reactance of the lines of the grid to determine that, for example, the transmission line connecting nodes 2 and 3 is critical (Fig. 3). Excluding this line from the network and re-computing the betweenness can yield the next critical line, for example, between nodes 4 and 5. These two lines are good attack targets.

During the attack of opening-closing of relays on critical lines, power flow through the line connecting buses 1 and

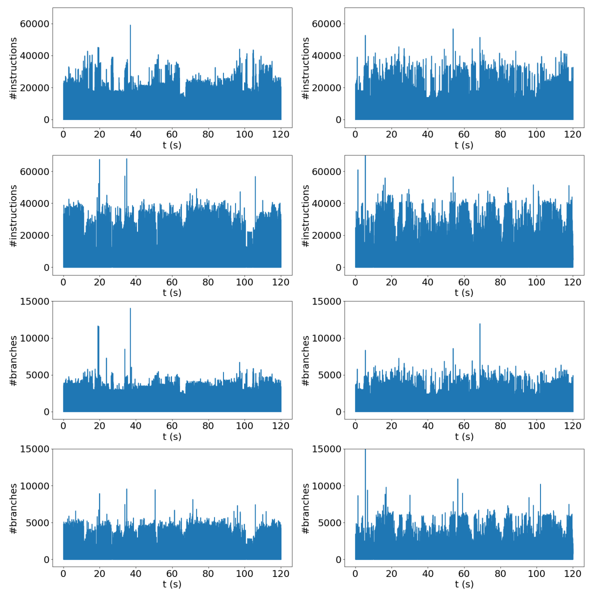


Fig. 5: HPC measurements for instructions and branches for the 2 non-quiescent threads in the OpenPLC under normal (left column) and malware-infested (right column) conditions.

3 reaches 84% of the tolerable power flow and power flow through the line connecting buses 2 and 5 reaches 87% of the tolerable power flow. Congestion in the lines increases maintenance cost and nodal price of the buses corresponding to congested lines. Fig. 4(a) shows impacts of attacks on transient response of the bus voltages. The voltage level at different buses in normal condition should be between 1.05 per unit-0.95 per unit. Although a steep change is seen in the voltage level of the buses when the transmission lines disconnect from the network, all bus voltages remain in range. Fig. 4 (b) shows the frequency response. Transient stability simulations do not show regulation of frequency back to nominal (60 Hz). Besides attacking relays on transmission lines, the attacker can target load relays to degrade performance and stability.

4) HPC-based Detection of Stealthy Malware: Preventing and detecting attacks requires a multi-layered approach [28] from a myriad of vantage points: network, on-device, and process-aware monitors [29]. An ensemble of anomaly detectors can detect the malware. While some detect anomalies relative to a baseline, others do not require measurements from a known-good device. HPCs detect changes in run-time characteristics of code execution. Listings of mapped memory regions of a process detect unexpected dynamically loaded libraries or changes in libraries. Reading the system call table detects changes in memory addresses in the system call table or in the system call handler functions. Techniques to detect anomalies in process listings and file system entries uncover processes and file system entries hidden by a rootkit. We will highlight the HPC-based detection next.

The quad-core ARM Cortex on the Raspberry Pi allows simultaneous reading of 6 HPCs. We use numbers of instructions, branches, stores, cycles, L1 instruction cache misses, and L2 data cache misses in this study. We read these HPCs at a sampling rate of 1 kHz. The HPCs are collected per-thread for each of the 3 threads in the OpenPLC controller process using a measurer process. The HPC measurer uses the PAPI library, connects to the target process and reads the time series of HPCs corresponding to each of the threads. We implemented the HPC measurer in C++, compile it on a separate computer into a statically linked executable and deploy it to OpenPLC.

A baseline data set of HPC readings over a time interval of $120s$ was collected from a good and trusted device. Half of this data set was used to train a one-class Support Vector Machine (SVM) classifier [10]. A sliding time window $0.25s$ long was considered (with a time shift of $0.01s$ between successive time windows). One of the 3 threads in the OpenPLC controller was quiescent. The mean and standard deviations, over the time window, of the measured HPCs for the 2 non-quiescent threads were used to construct the feature vector for the one-class SVM classification of normal/outlier. A $120s$ long data set was collected after deploying the malware.

The time series of HPC measurements from the good device and the malware-infested device are in Fig. 5. While these time series look similar from a macroscopic view except for intermittent non-deterministic spikes, the one-class SVM trained on baseline data accurately distinguishes between data from the good and malware-infected device. The one-class SVM is trained only on data from the baseline device. The SVM was tested on the second half of the baseline data set and the data set collected under malware. A sliding window of 40 normal/outlier detection with majority voting is used to output normal/outlier labels. Classification accuracy for baseline data is 90.06% (i.e., 10% false positives where normal data is marked anomalous) and classification accuracy for data from the malware-infested device was 93.94% (≈6.0% of false negatives where anomalous data is marked normal).

B. Intel Threat Detection Technology (TDT)

After a decade of academic research and proof-of-concept demonstrations, Intel as a major processor vendor has taken an important step to unlock the capabilities of HPC-based malware detection. Intel developed TDT as part of its Hardware Shield suite. Intel TDT leverages HPCs and hardware acceleration of machine learning on an integrated GPU to collect, profile, and detect malicious activities [13].

HPC-based TDT can detect ransomware [7]. Ransomware has emerged as the most prominent threat in cybersecurity, causing millions of dollars in losses due to ransom payments annually. A typical ransomware, such as WannaCry, encrypts the files on an infected computer using a private key encryption algorithm. WannaCry uses AES-128 in Cipher Block Chaining mode with a randomly generated key. These keys are encrypted with an RSA public key that is specific to an attack [30]. Software-based ransomware detectors focus on classes of encryption algorithms, require modifications to the operating systems and have high detection latency. As presented in Fig. 6, software-based detection can be bypassed by adaptive ransomware using including delayed arbitrary starts and memory mapped I/O based file encryption [7].

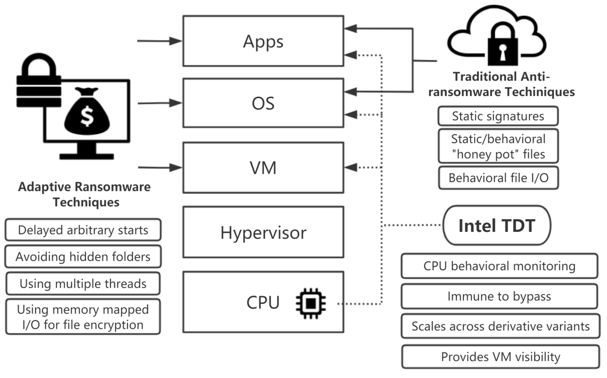

Apps
OS
VM
Hypervisor
CPU
Adaptive Ransomware Techniques
Delayed arbitrary starts
Avoiding hidden folders
Using multiple threads
Using memory mapped I/O for file encryption
Traditional Anti-ransomware Techiniques
Static signatures
Static/behavioral "honey pot" files
Behavioral file I/O
Intel TDT
CPU behavioral monitoring
Immune to bypass
Scales across derivative variants
Provides VM visibility

Fig. 6: Software-based ransomware detection is slow and can be bypassed by adaptive ransomware. HPC-based detectors are low-latency, cannot be tampered and scale to variants of ransomware.

HPC-based TDT is agnostic to encryption algorithms and does not have a performance constraint, as it is implemented in hardware. Ransomware operations involve accessing → opening → encrypting → closing files one after another. Encryption algorithms can be detected by the special patterns in cache-references, cache-misses, branches and branch-misses that can be monitored using HPCs. Fig. 7 shows two HPC-related events for the WannaCry ransomware [30].

C. Microsoft Defender for Endpoints

Microsoft Defender for Endpoints uses Intel TDT to detect unauthorized crypto mining [8]. Such cryptojacking entails maliciously co-opting computing resources of community to mine crypto currencies [8]. This is a formidable threat. Malicious cryptominers can be deployed as native applications that are transferred to other systems with the victim's credentials, or they can come in the form of browser-based malware when the victim visits the malicious website. Software-based

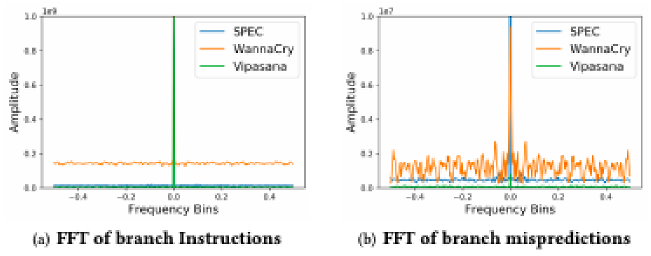


Fig. 7: Branch instruction and misprediction HPCs for SPEC benchmarks and Wannacry Ransomware [30].