# Hardware-assisted Cybersecurity for IoT Devices

Fahim Rahman, Mohammad Farmani, Mark Tehranipoor, and Yier Jin
Department of Electrical and Computer Engineering, University of Florida
{fahim034,mfarmani}@ufl.edu, {tehranipoor,yier.jin}@ece.ufl.edu

***Abstract*—Internet of Things (IoT) has become an integral part of the modern era as it provides an ease of life with higher flexibility, easier control, and wide connectivity through numerous applications. However, security vulnerabilities in the IoT domain have given rise to potential threats and attacks that can compromise critical infrastructures and national security, cause physical and financial loss. Being common and easier targets due to low power and low processing capabilities, lightweight firewall, and availability for service, IoT devices require lightweight but enhanced security to thwart different cyber attacks. In this paper, we champion for hardware-assisted defense mechanisms against cyber attacks in IoT domain. We highlight that hardware-assisted techniques indeed offer an additional layer of protection with respect to traditional software-only cybersecurity. However, to offer a comprehensive security, many challenges including area and power footprint, as well as security strength, need to be addressed.**

## I. Introduction

Internet of Things (IoT), the network of ubiquitous smart objects, has become an integral part of modern day-to-day life enabling novel applications and services, ranging from home automation to embedded medical devices and personal gears. Fast network connectivity and intelligent sensors allow these IoT devices to collect, process, and relay information in an efficient and seamless way. Advanced lightweight low-power technologies have made it even more possible to employ IoT devices in remote locations requiring no/minimal physical observation and maintenance. Although seemingly harmless, these IoT devices are not free from security and privacy concerns as there exist numerous threats and vulnerabilities in the modern IoT framework.

Security vulnerabilities in the IoT domain have given rise to countless threats and attacks that can potentially compromise critical infrastructures and national security, cause physical and financial loss, and more. McAfee quarterly threat report (Jan-Mar 2017) reveals that there are 176 new cyber-threats every minute [1]. Mirai-botnet based recent DDoS attack on low-cost IoT devices infected over 2.5 million devices within only four months [1]. Such attack volume, exploiting various security vulnerabilities, is expected to grow even larger with an estimated 26 billion connected devices by the end of 2020 [2]. As of today, most of the prevailing IoT devices do not offer adequate security measures to defend against these ever-growing pool of attacks and threats. Further, having easy network connectivity as an inherent feature, these IoT devices have become lucrative targets for remote attacks. Although techniques have been proposed for enhancing security and trust for connected devices, those may not be suitable solutions for lightweight IoT applications where processing capability, memory, and power are scarce. Moreover, most of such cybersecurity solutions are employed in the software domain which has its own set of challenges and vulnerabilities. Hence, it is crucial that one must explore, and, if suitable, employ hardware-assisted security as well with the software-only protections for IoT devices to thwart away prevailing and unforeseen threats. In this paper, we make a comprehensive study of recent hardware-based security solutions and highlight existing challenges and future research directions to offer a holistic security for IoT devices.

The rest of the paper is organized as follows: in Section II, we provide preliminaries to common attacks and vulnerabilities for modern day IoT devices and establish the threat model. In Section III, we analyze various hardware-assisted techniques for ensuring IoT security. In Section IV, we highlight some prevailing challenges and future research directions in hardware security domain for establishing a secure and trusted IoT system. Section V concludes the paper.

## II. Common Cyber-threats for IoT Devices

Before employing a protective mechanism against various attacks on IoT devices, it is crucial that one understands the attacker's capability and adversarial gains and targets. For a remote IoT device, an attacker may gain physical access to the device since regular monitoring and continuous protection for such lightweight and low-cost devices may not always be practically and financially feasible. A physical access to such a device opens up possibilities of probing attack, physical tampering attack, the inclusion of hardware bugs and Trojans, and replacement with fake devices [3]. However, in this work, we focus only on the cyber-attacks where the attack vectors are exercised via software and networks and a physical access to the device is not mandatory. We also assume that the network itself is untrusted and the attacker can gain access to the device without any loss of adversarial capability. Further, the attack itself is not transparent to the user, i.e., concealed from the the deployed security scheme, and the attacker can successfully launch the first phase of the attack without prominent changes in computation timing and processing power consumption.

The adversarial gain of an attacker from a remote cyber-attack is either to 'distort' the output (and subsequent actions due to faulty data/control) of the device, or to 'disrupt' the ongoing processes (e.g., denial of service), or to 'disclose' any secret information residing in the device such as secret keys and passwords [4], [5]. As shown in Fig. 1, an IoT device may be affected by a variety of attacks as follows.

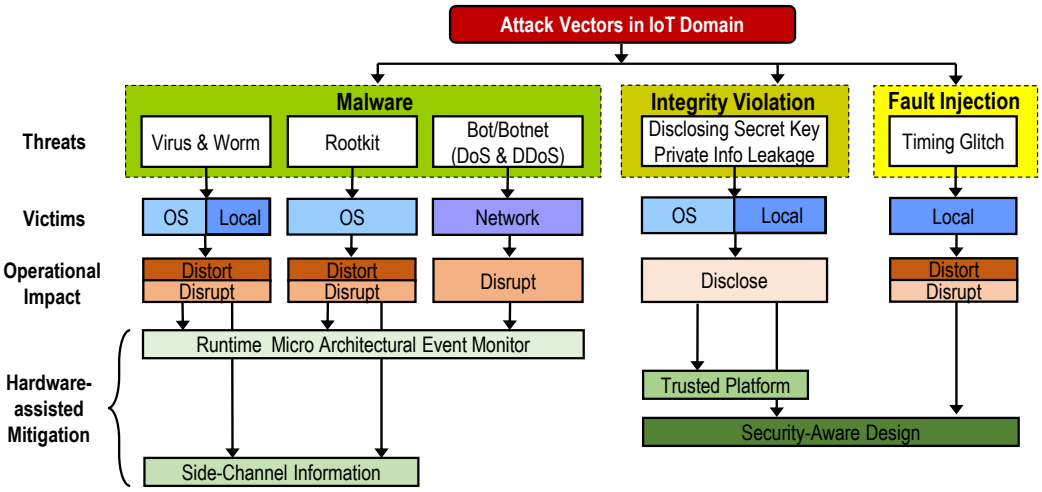


Fig. 1. Common cyber-attacks for IoT devices and hardware-assisted mitigation techniques.

**Malware.** Modern day IoT devices may be infected by a variety of malware at different stages of its lifetime. Common malware, such as virus, worms, and Trojans, usually target local and OS-level exploitation based on the attack and execution complexity. The primary behavior of such virus is to disrupt the ongoing operation and cause an illegitimate interrupt or a routine call for malicious activity. The rootkit, on the other hand, disrupts the OS level legitimate executions since the original firmware is infected and modified. It can provide continued privileged access to a system while actively hiding its presence. Additionally, IoT devices, if not protected properly, can fall into the victim of the denial of service (DoS) or distributed denial of service (DDoS) attacks that are becoming a major problem day by day. Such an affected device can work as a bot (or 'zombie') to infect other valid devices in the network or can consume network bandwidth and processing power giving the attacker additional resources.

**Integrity Violation.** Getting the access to the private key (used for encryption) and/or private information residing in the IoT device is a lucrative adversarial gain for an attacker since it allows to compromise the 'root-of-trust' of the systems. It enables the attacker to gain control of the secure communication and execution processes of the device.

**Software-assisted Fault Injection.** Another class of attacks on IoT devices is the software-assisted fault injection in the hardware during device runtime. Although, this type of attacks are relatively complicated since they require in-depth knowledge of the platform dependent hardware as well as the underlying software; the remedy of such an attack is very difficult to implement. For example, a runtime timing glitch attack can be executed via software exploitation if there lacks a security-aware design in the hardware module [6]. Since this class of attacks search for and utilize minute vulnerabilities in the hardware, employing software-only defensive mechanisms are often futile.

We note that there have been various proposed software-only solutions to mitigate the existing treats for IoT devices [7]–[10]. However, the advanced attacks and cyber-threats for modern IoT devices are not always preventable by such software-only techniques because:

- The software-based defense mechanism, employed in the IoT device, itself may vulnerable to remote attacks. Additionally, it does not necessarily have a large coverage of defense, can often be bypassed and compromised without the user's knowledge. For example, the DoubleAgent attack has compromised many known anti-virus software causing a loss of integrity [11].
- Software-based anti-malware solutions (programs) require regular updates and patches to keep the system protected. Many new threats do emerge by exploring such updates as well since an smart analysis of such updates reveals the points of vulnerabilities and many existing user do not always comply to such update at the moment of release. Additionally, these approaches are often vulnerable to zero-day attacks.

## III. Hardware-assisted Security Techniques

Hardware-assisted security has become a promising alternative to the software-only defense mechanism as the latter alone do not provide the level of security needed for present day IoT devices. Hardware-based methods leverage the hardware modules and can collect micro-architectural information to analyze the prevailing software-level threats and vulnerabilities. As shown in Fig. 1, hardware-assisted techniques provide a wide range of solutions, described as follows, for secure and trusted IoT applications.

### *A. Secure and Trusted Hardware Architectures*

One of the primary requirements for performing a secure information transaction among the IoT devices via an untrusted network is to employ a trusted and secure key management and data processing scheme in hardware. The trusted platform modules (TPMs) have been well adopted in this regard to serving as the hardware root of trust [12]. Such TPMs allow

using cryptographic keys that can be tied to certain platform measurements and are protected from disclosure to any other untrusted hardware components, processes, or software. The discrete-TPM (dTPM) chips and plastic circuit board modules can offer a larger coverage of services allowing sharing the resources among multiple applications on the same physical machine [13]. Additionally, ARM TrustZone [14] and Intel Software Guard Extension (SGX) [15]-enabled architectures add new features to modern-day SoCs to offer trusted and secure environment for execution of security-critical processes in the face of the privileged kernel and software being potentially malicious [16], [17]. On the other hand, secure processors such as AEGIS [18] and Ascend [19] leverage single-chip architectures to ensure private and authentic processing with encrypted and obfuscated instruction executions. However, such proposed designs mostly offer security against physical attacks, such as tampering and probing, and do not necessarily have provisions to offer safeguard against prevailing cyber threats if the program itself is compromised.

### *B. Micro-architectural Event Monitoring for Security*

Although TPMs and secure architectures offer a trusted environment for security-sensitive applications, such a hardware is generally expensive, power-hungry, and is not suitable for lightweight and low-cost IoT applications. Furhter, malware such as virus, Trojans, and bots can infect IoT devices in disguise if the network is not sufficiently protected. Once infected, it is very difficult to detect the malware as it can bypass the anti-malware programs in the device. Hardware-based micro-architectural event monitoring for recognition of malware and anomaly detection can help in such cases. It offers a fine-grain filtering for individual executions, can collect multi-dimensional information, and provides a faster response than the software-only anti-malware counterparts.

The heart of such hardware-based monitors is the performance monitoring units (PMU) available in the most modern processors and SoCs [20], [21]. The basic goal of PMUs is to provide the performance insight of the CPU by capturing a set of micro-architectural events and respective counts using the inbuilt hardware performance counters (HPCs). For example, one or more HPCs in the PMU can sample how many times a pre-defined event (enabled by the associated architecture), such as cache misses, occurs during the program runtime to evaluate the performance of the system under test. PMUs in ARM and Intel x86 architectures can be controlled via software modules such as Linux Perf tool [22]. It can provide real-time feedback to diagnose bugs or identify bottlenecks in the software based on the hardware platform. Although originally designed for performance monitoring, the PMU can be intelligently used for security applications by analyzing whether an runtime event profile is malicious in nature. Another advantage is that, being an integrated part of the hardware, the PMU operates transparently to any software running on the processor and cannot be fooled by external malicious software. That is the hardware monitor itself is process-oblivious. Since any malware or even a modified firmware or rootkit needs to perform certain executions, PMU-driven event monitoring is potentially capable to detect such malicious activities.

Tang et al. [23] proposed a method for anomaly-based detection of malware using lower-level micro-architectural features collected from HPCs in modern processors. Since the micro-architectural characteristics of the benign programs can be noisy due to diffusion of multiple program executions within a given time window, it is quite difficult to characterize and distinguish a malicious program. However, a careful feature selection, i.e. event for the HPCs to collect, and extraction can lead to the benign program characterization with unsupervised ML techniques. This model, built on the sub-semantic micro-architectural features, then can be leveraged for an offline classification between the original and the possibly modified malicious program.

Jyothi et al. [24] proposed a host-based DDoS-detection framework called BRAIN (BehavioR based Adaptive Intrusion detection in Networks). It uses hardware features to model the benign and DDoS behavior. To detect DDoS attacks, it uses machine learning techniques on modeled application behavior and network statistics both. Since the correlation between the network and the application statistics with HPC data is non-trivial, high-fidelity hardware events need to be selected. The authors proposed to implement an integrated DDoS Detection Engine (DDoSDE) that monitors both the hardware and network behavior to detect the DDoS/DoS attacks. The DDoS Prevention Interface (DDoSPI) responds against any detected attack by blacklisting IPs (and removing if needed) based on a dynamic network and HPC-based threshold that thwarts the attacker from learning the policy.

Wang et al. [25] proposed a low-cost host-based validation tool, namely ConFirm, to detect the malicious modifications to the firmware of embedded systems, such as calling, injection, or execution of codes that are not a part of the original code flow execution. The technique exploits hardware event counts via HPCs creating internal check-points within the original firmware. An event-count of a possibly maliciously modified firmware can, therefore, be matched within the checking points (checking window) to validate the integrity of the malware. Malone et al. [26] also considers a similar threat of modified firmware. However, for accurate detection, the authors analyzed malicious static and dynamic program modification by modeling the architectural characteristics of benign programs using linear regression models.

### *C. Security Enhancement via Machine Learning Techniques*

One major obstacle for leveraging micro-architectural event monitoring for security is that the same micro-architectural event can occur in a similar manner (i.e., frequency count and event profile) during a valid operation and, therefore, it may not be an obvious indicator for flagging a particular program (or execution) as a malicious one. However, researchers have implemented different machine-learning techniques to learn and differentiate between such events to identify any kind of anomaly with a higher detection accuracy and lower false

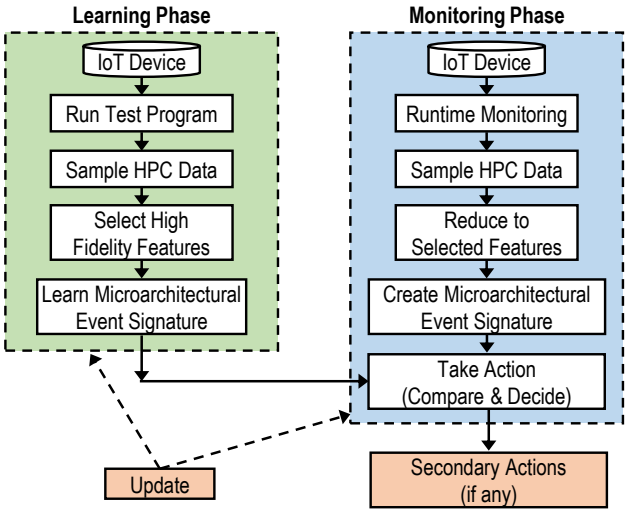


Fig. 2. Hardware Performance Counter-based Malware Detection.

positive/negative [27], [28]. Two fundamental requirements for such techniques are: 1) selecting high-fidelity micro-architectural features for event collection via HPCs, and 2) choosing efficient machine learning techniques for classification/detections. Demme et al. [27] envisioned the architecture necessary to support behavioral-based malware detection in the hardware using HPC data. The key observations for building such a framework is that

- The semantics of a program do not change significantly even though the attacker tries to restructure it.
- While accomplishing a particular task, there exist sub-tasks that cannot be radically modified.

Based on the given assumptions, a machine learning-based hardware-assisted anomaly detection unit should perform the following general tasks, as shown in Fig. 2:

1) **Data Collection and Feature Selection**. It decides what micro-architectural event data needs to be collected and how the detection engine should store and process the collected information.
2) **Data Analysis**. It determines the malicious behavior (if any) by analyzing the data. ML classifiers are used for learning, testing, and validating the correlation between the collected data and untrusted behavior.
3) **Action System**. It takes action when a threat is detected. It could be simple as just flagging/reporting the event to the user for potential threat or crucial as shutting down the complete unit to protect critical data/system.
4) **Secure Updates**. It keeps the detection engine up-to-date by dispatching necessary updates with newly learned classifiers and new vulnerabilities. Such an update must be secure to indicate that it can be done only by a trusted authority.

The detection module receives HPC information periodically from the target module where the untrusted program or malware is running. The system architecture should allow the detection module to run at the highest privilege level and independently of any other program. Also, it should give access to physical memory to store HPC data and have isolated memory and execution for itself so that the detection module itself does not get corrupted. The amount of storage required to store the ML data varies greatly depending on the type of classifier being used for analysis. As one can understand, the accuracy of the ML technique used and the fine-grained resolution of the sampled HPC data for selected events play a vital role to improve the accuracy and performance of the overall detection system.

To further examine the impact of different ML techniques for malware detection, Patel et al. [28] presented a comprehensive analysis using runtime HPC information. The analysis shows that the OS Kernel level software implementation of various ML methods are extremely slow, in the range of milliseconds, with respect to the malware runtime and data sampling at the hardware level. It is apparent that the software-level classification techniques are not suitable enough for high-resolution data capture and anomaly detection with high confidence. Therefore, a hardware-based ML technique implementation is required for lower latency and higher accuracy. The authors synthesized different ML methods in a programmable platform (Virtex 7) for a comparative analysis and it was found that OneR technique was the most efficient benign vs. malware classifier with the highest accuracy/area and the lowest power-delay product, with an overall detection success of around 81%.

### *D. Leveraging Side-channel Information*

Power side-channel-based instruction-level disassemblers have been demonstrated by Eisenbarth et al. [29] and Park et al. [30]. The implication of such a disassembler is manifold - it can be used for code tracing and reconstruction, firmware reverse engineering, hardware-software co-attestation, and, most importantly, for the integrity verification of a software running in an IoT device. Msgna et al. [31] has leveraged the power side-channel information for embedded software integrity verification by constructing instruction-level power templates, using selected ML techniques, from multiple identical (by design) processors and matching the application runtime power consumption profile against them through RSA signature screening scheme. However, such a disassembly scheme is basically suitable for simple microcontrollers. With the increasing complexity of the modern day processors and extensive pipelining, it becomes extremely difficult to offer an efficient and guaranteed anomaly detection via instruction-level disassembly using only power-side channel. Moreover, extremely large data and complex post-processing suggest it to be an off-the-chip anomaly detection and integrity verification scheme. However, such a scenario may not be practical for IoT devices.

Further, Clark et al. [32] have demonstrated that a prevailing malware can be detected using side-channel power disruptions in medical devices. The proposed system monitors the power consumption of the embedded device and employs machine learning technique to detect potential abnormal be-

havior/signature from the device. Gonzalez et al. [33] has shown that one can accurately discriminate between encrypted and unencrypted transmissions in a software defined radio platform using power fingerprinting. Such techniques can be used for IoT device attestation and authentication in an untrusted network.

Additionally, Nazari et al. [34] has implemented a runtime monitoring system using electromagnetic emanations (EM) as a side-channel. It can potentially detect abnormal behavior during program execution such as malware or other code injections using supervised ML classifiers. This scheme does not need to perform any malware characterization; rather it uses the peaks in the measured EM spectrum during the program execution and compares them with the training-phase golden data. This technique is potentially well-suited for security monitoring of IoT and embedded devices since it does not need additional resources on the monitored machine, does not require wired connection as in the case for power side channel information collection, or does not impact the monitored (malicious or not) program.

### *E. Security-aware Design and Control*

Tang et al. [6] has demonstrated the first software-exposed fault-attack that exploits the security-obliviousness of the common on-chip energy management mechanisms. The authors developed a malicious kernel driver to exploit the on-chip dynamic voltage and frequency scaling (DVFS) modules for inducing timing faults during the program execution. Such an attack can eventually extract secret cryptographic keys and alter kernel privilege in the ARM Trustzone. Although carried out via a malicious program on a Nexus 6 device, this attack leverages the implementation vulnerabilities of the existing hardware module and, therefore, can potentially be extended to remotely attack and compromise IoT devices.

The defense against such an attack requires removing multiple implementation vulnerabilities of the on-chip energy management module. The DVFS, traditionally being security-oblivious, can be operated way outside of the vendor-specified ranges. One can address this by enforcing the operating limits within a specified range. Further, the DVFS regulator should be well isolated from each other so that the protected environment cannot be compromised by another module. And, finally, necessary redundancy and check-points implemented both in hardware and software can remove the fault should it occur.

Additionally, Spisak et al. [35] leveraged the PMUs of a processor for implementing a hardware-assisted rootkit. Using this kernel mode rootkit, an attacker can redirect existing control flow to malicious code without any kernel image modification. As it is seen, such a PUM-assisted rootkit design can possibly trap system calls and other interrupts driven entirely by the PMU. Although the proposed attack was carried out on a Qualcomm Snapdragon processor used in Nexus 6 device, a similar or inspired attack can be launched on modern IoT devices with comparable processing architecture. However, this attack has its own weakness because other programs with kernel-level privilege are capable of reading, writing, or tampering the counters in PMUs. To prevent this rootkit attack, one may employ dynamic event code filtering, reset the counter value before overflowing, and keep monitoring the PMU itself for any anomalous behavior.

## IV. Challenges and Future Directions

Although the aforementioned hardware-assisted security techniques provide a wide range of protections for modern IoT devices, several challenges and limitations exists with respect to the detection or prevention efficiency, associated software, and hardware footprint. Some major challenges regarding a successful implementation are:

- Existing trusted and secure hardware, such as a TPM, commonly suffers from a relatively larger footprint and is usually power hungry. A plug-and-play adaptation of such devices and architectures is therefore not suitable for lightweight IoT devices where the processing unit is less powerful and area and energy are scarce.
- Hardware performance counter-based malware detection methods deeply rely on efficient and complex machine learning techniques to differentiate between the valid and malicious operations. However, such ML techniques are usually implemented on software, where the HPC data is collected in the hardware and fed to the software program for online/offline processing and detection. This, unfortunately, creates the bottleneck in the detection scheme as the software-based classification is very slow with respect to the accumulated hardware data. It forces the event/data sampling to be done at a much lower rate (i.e., high event sampling interval) increasing the risk of malicious events or executions to go unnoticed. Eventually, it reduces the success rate and increases operational overhead [28].
- Most of the ML techniques used in the aforementioned schemes require a large data for training, testing, and validation. Such a golden model-based scheme is only suitable for known cyber threats as it allows a proper characterization of the malicious events in the training phase. Hence, detecting zero-day vulnerabilities and unknown threats using such a scheme is quite challenging.
- Side-channel based IoT monitoring and malware detection systems require additional hardware for data collection and processing. Here, self-monitoring may create interference with the collected signatures or the monitoring unit may itself get compromised. On the top of that, obtaining power side channel information requires additional wire-connection to the device being monitored. Although EM monitoring does not need a wire-connection, however, the resolution of the EM-collecting antenna and environmental interference greatly impact the quality of the detection. For both the power and EM-based monitors, placing additional monitoring hardware may not be a feasible solution for low-power remote IoT devices.

Hardware-assisted security for IoT devices may act complementary to the existing software-only defense mechanisms.

A key research thrust in this direction may be to build hardware accelerators for high-speed on-chip ML implementations. Designing lightweight ML techniques is also crucial for event-recognition based malware/anomaly detection. In addition to PMUs, various other on-chip sensors, such as temperature and aging sensors, and novel architectures may also be employed for ensuring security.

## V. Conclusion

With the rise of the IoT devices and IoT-centric applications, it is imperative that we provide a trusted and secure platform for this domain to offer maximum defense against prevailing and future cyber attacks. Existing security solutions are not adequate since such defense techniques do not offer a wide and strong coverage. Hence, it is high time to employ both hardware and software solutions in a hybrid manner. Hardware-assisted protection for the security of IoT devices has proven their importance and versatility over time and across multiclass attack vectors. However, there is no doubt that further research is required to develop and design appropriate security mechanisms with high accuracy and low overhead for lightweight IoT applications.

## Acknowledgement

This work is partially supported by Cisco, Inc.

## References

[1] McAfee Labs Threats Report: April 2017. [Accessed: 05 December 2017]. [Online]. Available: https://www.mcafee.com/April2017ThreatsReport

[2] P. Middleton, P. Kjeldsen, and J. Tully, "Forecast: The internet of things, worldwide, 2013," *Gartner Research*, 2013.

[3] J. Wurm, Y. Jin, Y. Liu, S. Hu, K. Heffner, F. Rahman, and M. Tehranipoor, "Introduction to cyber-physical system security: A cross-layer perspective," *IEEE Transactions on Multi-Scale Computing Systems*, 2016.

[4] W. B. Miller, "Classifying and cataloging cyber-security incidents within cyber-physical systems," 2014.

[5] CAPEC: Mechanisms of Attack. [Accessed: 05 December 2017]. [Online]. Available: https://capec.mitre.org/data/definitions/1000.html

[6] A. Tang, S. Sethumadhavan, and S. Stolfo, "CLKSCREW: Exposing the perils of security-oblivious energy management," in *26th USENIX Security Symposium (USENIX Security 17)*, Vancouver, BC, 2017, pp. 1057–1074.

[7] G. Carl, G. Kesidis, R. R. Brooks, and S. Rai, "Denial-of-service attack-detection techniques," *IEEE Internet computing*, vol. 10, no. 1, pp. 82–89, 2006.

[8] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[9] M. Christodorescu, S. Jha, S. A. Seshia, D. Song, and R. E. Bryant, "Semantics-aware malware detection," in *Security and Privacy, 2005 IEEE Symposium on*. IEEE, 2005, pp. 32–46.

[10] G. Zhao, M. Liu, and S. Badrinath, "Virus detection and removal system and method for network-based systems," Jul. 18 2006, uS Patent 7,080,407.

[11] New Attack Uses Microsoft's Application Verifier to Hijack Antivirus Software. [Accessed: 05 December 2017]. [Online]. Available: https://www.bleepingcomputer.com/news/security/new-attack-uses-microsofts-application-verifier-to-hijack-antivirus-software/

[12] W. Arthur and D. Challener, *A Practical Guide to TPM 2.0: Using the Trusted Platform Module in the New Age of Security*. Apress, 2015.

[13] Trusted Platform Module (TPM) on Windows 10 IoT Core. [Accessed: 05 December 2017]. [Online]. Available: https://docs.microsoft.com/en-us/windows/iot-core/secure-your-device/tpm

[14] ARM TrustZone. [Accessed: 05 December 2017]. [Online]. Available: https://developer.arm.com/technologies/trustzone

[15] Intel Software Guard Extensions (Intel SGX). [Accessed: 05 December 2017]. [Online]. Available: https://software.intel.com/en-us/sgx

[16] ARM, "Security technology building a secure system using trustzone technology (white paper)," 2009.

[17] V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptology ePrint Archive*, vol. 2016, p. 86, 2016.

[18] G. E. Suh, C. W. O'Donnell, and S. Devadas, "Aegis: A single-chip secure processor," *IEEE Design & Test of Computers*, vol. 24, no. 6, 2007.

[19] C. W. Fletcher, M. v. Dijk, and S. Devadas, "A secure processor architecture for encrypted computation on untrusted programs," in *Proceedings of the seventh ACM workshop on Scalable trusted computing*. ACM, 2012, pp. 3–8.

[20] ARM CortexA9 Technical Reference Manual - Chapter 11 Performance Monitoring Unit. [Accessed: 05 December 2017]. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0433c/BEHGGDJC.html

[21] Intel 64 and IA-32 Architectures Software Developers Manual - Volume 3. [Accessed: 05 December 2017]. [Online]. Available: https://software.intel.com/sites/default/files/managed/a4/60/325384-sdm-vol-3abcd.pdf

[22] The Unofficial Linux Perf Events Web-Page. [Accessed: 05 December 2017]. [Online]. Available: http://web.eece.maine.edu/~vweaver/projects/perf_events/

[23] A. Tang, S. Sethumadhavan, and S. J. Stolfo, "Unsupervised anomaly-based malware detection using hardware features," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2014, pp. 109–129.

[24] V. Jyothi, X. Wang, S. K. Addepalli, and R. Karri, "Brain: Behavior based adaptive intrusion detection in networks: Using hardware performance counters to detect ddos attacks," in *VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), 2016 29th International Conference on*. IEEE, 2016, pp. 587–588.

[25] X. Wang, C. Konstantinou, M. Maniatakos, and R. Karri, "Confirm: Detecting firmware modifications in embedded systems using hardware performance counters," in *Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on*. IEEE, 2015, pp. 544–551.

[26] C. Malone, M. Zahran, and R. Karri, "Are hardware performance counters a cost effective way for integrity checking of programs," in *Proceedings of the sixth ACM workshop on Scalable trusted computing*. ACM, 2011, pp. 71–76.

[27] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo, "On the feasibility of online malware detection with performance counters," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 559–570.

[28] N. Patel, A. Sasan, and H. Homayoun, "Analyzing hardware based malware detectors," in *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017, p. 25.

[29] T. Eisenbarth, C. Paar, and B. Weghenkel, "Building a side channel based disassembler," in *Transactions on computational science X*. Springer, 2010, pp. 78–99.

[30] J. Park and A. Tyagi, "Using power clues to hack iot devices: The power side channel provides for instruction-level disassembly." *IEEE Consumer Electronics Magazine*, vol. 6, no. 3, pp. 92–102, 2017.

[31] M. Msgna, K. Markantonakis, D. Naccache, and K. Mayes, "Verifying software integrity in embedded systems: A side channel approach," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2014, pp. 261–280.

[32] S. S. Clark, B. Ransford, A. Rahmati, S. Guineau, J. Sorber, W. Xu, K. Fu, A. Rahmati, M. Salajegheh, D. Holcomb *et al.*, "Wattsupdoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices." in *HealthTech*, 2013.

[33] C. R. A. Gonzalez and J. H. Reed, "Detecting unauthorized software execution in sdr using power fingerprinting," in *MILITARY COMMUNICATIONS CONFERENCE, 2010-MILCOM 2010*. IEEE, 2010, pp. 2211–2216.

[34] A. Nazari, N. Sehatbakhsh, M. Alam, A. Zajic, and M. Prvulovic, "Eddie: Em-based detection of deviations in program execution," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ACM, 2017, pp. 333–346.

[35] M. Spisak, "Hardware-assisted rootkits: Abusing performance counters on the arm and x86 architectures." in *WOOT*, 2016.