Evolution of Embedded Platform Security Technologies: Past, Present & Future Challenges

Fahad Siddiqui, Sakir Sezer
The Centre for Secure Information Systems (CSIT), Queen's University Belfast
Belfast, United Kingdom
f.siddiqui, s.sezer@qub.ac.uk

***Abstract*—In recent years, the proliferation of intelligent embedded technologies is opening venues to new service and computing models, providing diverse socio-economic benefits. These intelligent technologies are giving rise to a wide range of public and private applications by sharing and analysing generated data. This includes smart home, smart health, smart city, autonomous vehicles, smart grid and smart manufacturing etc. However, where this sharing of data brings benefits and opportunities, it simultaneously presents security risks and challenges. The realisation and prototyping of such technologies require a computing hardware widely available in the form of an embedded platform. The security perimeter and the attack surface of these platforms rely on their supported security and defence mechanisms. This paper aims to build a body-of-knowledge in this area for the security research community. It present the state-of-the-art security frameworks and architectures, discuss architectural shortcomings and root-causes of leading security technologies rather than discussing vulnerabilities and attacks. The paper concludes advocating secure-by-design platform approach and classifying platform security methods to realise robust embedded platform security architecture.**

***Index Terms*—Secure-by-design, Platform Security, Security micro-architecture, Active Defence, Trusted Execution Environment, Arm TrustZone, Intel SGX, Hex-FIVE MultiZone.**

I. Introduction

The reliance on intelligent computing technologies has grown opportunities for adversaries to conduct malicious activities, steal valuable data, launch attacks and compromise public and private digital services [1], [2], [3], [4]. The international government agencies have released cybersecurity regulations [5] to manage such challenges by advocating businesses and technology manufacturers to comply and adhere to these regulations. This need for compliance has triggered a chain reaction among service providers, original equipment manufacturers and embedded platform manufacturers and posed them to design, develop and integrate platform defence mechanisms [4], [6], [7]. These defence mechanisms can facilitate designing *detect*, *respond* and *recover* functions in case of malicious activity, software malfunction or adversarial attack [3], [4], [8], [9]. Since such incidents cause severe damage to the business reputation and safety of their users, the need for strong and robust platform security has become an obligatory design and operational security requirement for next-generation intelligent embedded technologies [3], [2], [10].

In past, the computing technologies have witnessed shifts between centralised and decentralised control (from mainframes to PCs) to an important wave of centralisation by moving control, data and intelligence of the computing system back to the cloud [11]. This trend has followed by another wave of decentralised in the form of edge computing [4]. These technological advancements have guided the evolution of both new computing paradigms, threats, attacks vectors and security architectures to meet the diverse security requirements [12], [13]. In response, novel defence and security mechanisms have been incrementally introduced into security architectures, to strengthen security perimeter and reduce attack surface [2], [6], [9], [14], while maintaining data confidentiality, integrity and availability of digital services.

The objective of this paper is to synthesise the knowledge on past and present of embedded platform security technologies. Primarily focusing on the leading commercial embedded security technologies and discuss the root cause of their architectural security shortcomings. Section II presents the evolution of embedded platform security architectures. Section III covers the most-relevant embedded security frameworks and bodies that define the baseline security requirements. These security requirements are the cornerstone of leading commercial embedded security technologies such as Intel SGX, Arm TrustZone and Hex-FIVE MultiZone Security discussed in Section IV. This followed by a discussion on future directions covering secure-by-design approach, classification of security methods and defence mechanisms presented in Section V.

II. Evolution of Embedded Platform Security

The evolution of embedded security platform architecture has been guided by technological advancements and a cause and effect of developing application requirements and design challenges [3], [12], [13], [15]. This includes the realisation of mix-critical applications, broader network connectivity, a need for code/data protection, isolation and segregation of resources leading to wider availability of embedded multiprocessing [6], [7], [9]. Where these computing platform advancements brought benefits, they have exposed platforms to wide-range security challenges due to increased system attack surface and micro-architecture vulnerabilities [12], [16], [17], [18], [19], [20], [21], [22].

The *Early Systems* were highly constraint in terms of resources, area, memory, performance, power and designed to solve domain-specific problems. They had no connectivity, hence their security requirements were limited to physical security. Though they found vulnerable by unauthorised access

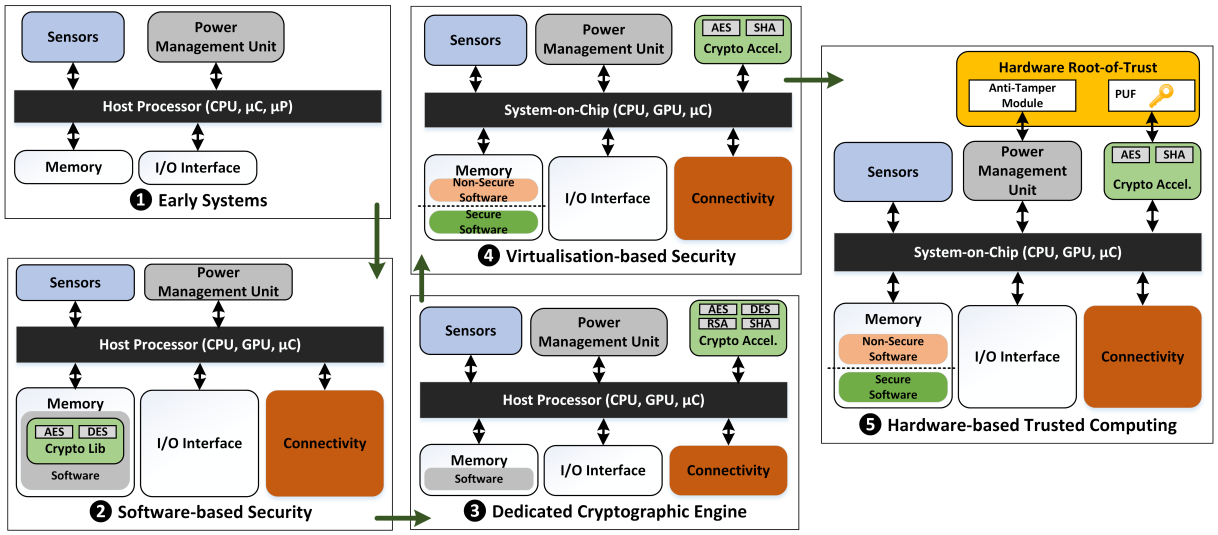

Fig. 1. Evolution of Embedded Platform Security architectures to harness, build and maintain robust security to meet diverse security requirements and technological challenges posed by dynamic threat landscape.

and modification to the system. To address this challenge and unleash their true potential, broader network connectivity had been introduced to the platforms that enabled opportunities for *pervasive computing* as illustrated in Fig. 1. It allowed automation and optimisation of industrial control processes by communicating with other systems using a dedicated/internal private network [11]. Though safe business operations require protection of secret data, thus a *Software-based Security* approach was introduced that executes symmetric cryptography libraries (AES, DES) as illustrated in Fig. 1 to protect data [15]. With technological advancements, the application requirements started to become complex, computationally expensive and require execution of asymmetric cryptography algorithms for secret key exchange that posed serious performance issues. To circumvent this problem, the computationally intensive cryptography services were offloaded from a host processor to a *Dedicated Cryptographic Engine* (Fig. 1).

The realisation of mix-critical automotive and industrial control applications become commonplace and require multiprocessing. Since the cost of silicon was still expensive and embedded architectures were area and memory constrained, the security architects had introduced *Virtualisation-based Security* approach. This allows to isolate and segregate computing resources [7], [14]. [13], [23]. Using virtualisation, a physical processor and platform resources are logically divided into multiple domains as shown in Fig. 1. Examples include Arm TrustZone, Intel SGX, Hex-Five MultiZone Security etc. The mass deployment and adoption of mobile and edge computing fuelled data-driven value business models [4], [11], [12]. This tremendously increased the threat landscape and by increasing the attack surface of these platforms [6], [7]. Active involvement of third-party vendors during manufacturing and supply-chain processes had exposed businesses to *device identity* challenges, leading to wide-range of intellectual property theft, device cloning and counterfeit attacks etc. that severely damages the businesses both financially and morally. To approach this challenge, a *Hardware-based Trusted Computing* approach was introduced [15], [24] as illustrated in Fig. 1. A hardware *root-of-trust* components that serve an immutable trust anchor is tightly integrated into the platform security architecture. Depending on the design and complexity, it can generate secret keys, store secret keys inside a tamper-proof secure storage and sign to verify digital secrets [24].

The majority of existing embedded technologies use one of the five discussed security architecture depending on their domain-specific design, application and security requirements.

III. Embedded Security Frameworks & Bodies

The emergence of novel security threats and attack vectors have initiated the need for a strategy to protect embedded devices and digital services. This requires the creation of fully collaborative and open ecosystem, where all stakeholders can efficiently deliver innovative digital services while providing greater security with convenience for users. Therefore, the manufacturers have created several industry-driven bodies that collaborate, define, develop, refine and standardise technology-specific security frameworks [24], [25], [26], [27].

A. GlobalPlatform

GlobalPlatform [25] is a non-profit, industry-driven association of leading silicon, original equipment manufacturers and digital service providers. GlobalPlatform defines a baseline security specification to securely manage and protect services offered by connected embedded devices throughout their life cycle by standardising three areas of technology development:

1) *Secure Elements (SE)* - a digital trust anchor providing *root-of-trust* services to the platform. It is fundamental to establish, build and maintain device security based on the principle-of-trust [25] (Fig. 2). The example of secure elements are Smartcards, SIM, ePassports, eID.

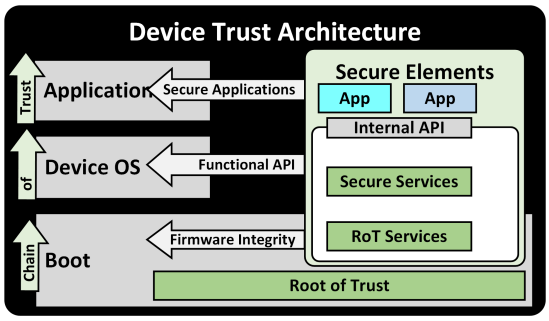

Fig. 2. GlobalPlatform Device Trust and Security Architecture [28].

2) *Trusted Execution Environment (TEE)* - is a secure area of the processor which ensures that sensitive data is stored, processed and protected in an isolated and trusted environment. It allows protecting system data and critical assets [29] (Fig. 2) by isolating and segregating resources between trusted and rich operating system. Qualcomm TEE, Arm Trustonic TEE, Sierra TEE, Huawei TEE, Linaro TEE (OPTEE).
3) *Device Trust Architecture (DTA)* enables seamless interaction and deployment of secure digital services among stakeholders [28]. In a connected ecosystem, it is essential that a device should serve its identity and data in a way that any other device or application can trust [24]. Therefore, DTA shows how *Secure Elements* can be used to build an attestable *Chain-of-Trust* (Fig. 2) by loading the operating system, executing *Trusted Execution Environment* and execute software services across connected layers (cloud, network, edge).

B. Trusted Computing Group

Trusted Computing Group [24] is a member-driven workgroup that enables the benefits of trust in digital computing devices from mobile to embedded systems, as well as networks, storage, infrastructure, and cloud security. They proposed a set of hardware and software root-of-trust technologies that enabled the construction of trusted platforms [24]. Their core technologies include specifications and standards for the *Trusted Platform Module* (TPM), *Trusted Network Communications* (TNC) and network security and self-encrypting drives.

C. Arm Platform Security Architecture

It is an architecture-agnostic digital platform security framework developed by Arm Holdings, as a shared responsibility by contributing to the embedded processor ecosystem and to help to secure digital devices [26]. The framework guides to identify, analyse and prioritise security requirements of an application use case. It helps designers and developers to securely architect, implement and certify system hardware and software specifications of the application use case [26].

D. GlobalPlatform IoTopia

GlobalPlatform has launched a new initiative called IoTopia [27] to provide a common framework for standardising the design, implementation guidelines in-line with the global security requirements, certification, deployment and management of the next-generation connected device. It consists of four pillars: *Secure by Design, Device Intent, Autonomous, Scalable, Secure On-boarding of devices* and *Device Life-cycle Management*. This initiative is expected to lead specification development and standardisation process of next-generation connected technologies in future.

GlobalPlatform being the industry standard, Section IV discuss its vendor-specific implementation.

IV. Vendor Specific Implementation of GlobalPlatform's Device Trust Architecture

A. Intel Security Guard Extension (SGX) Technology

Intel *Software Guard Extensions* (SGX) is a set of instructions that increases the security of application code and data providing protection from disclosure or modification [16]. To realise GlobalPlatform's *Device Trust Architecture*, the developers partition sensitive information into areas of execution in memory called *enclaves*. To support this security model, a new set of security-related CPU instructions has been introduced. The application developer can set aside regions of data that are more private known as enclaves [6]. This private data cannot be read or modified except by the function running within the enclave itself, preventing direct attacks on executing code or data stored in the memory.

The software model of SGX security technology is built around untrusted and trusted (enclave) run-time systems as shown in Fig. 3. During run-time execution, firstly the untrusted part of the application executes and creates an enclave using SGX instructions. This enclave is built into a special encrypted memory region with restricted entry/exit location defined by the developer [6] which helps to prevent enclave's code and data leakage. Secondly, the trusted function is called, where the execution transitioned to the enclave using the entry point into the enclave. At this stage, the enclaves have access to secure data that can be used to process secret data. Once finished, the trusted function returns control back to untrusted function. The secret data remains in the trusted memory, and the processor returns back to normal execution.

To identify the security shortcomings of Intel SGX security technology beyond the discussion of exploited vulnerabilities and attacks, it is crucial to understand the underlying architectural security aspects especially the enclave entry/exit mechanisms. Intel SGX offers system call mechanism to switch between trusted and non-trusted execution environments as illustrated in Fig. 3. The processor orchestrates the system call into synchronous and asynchronous enclave's entry/exit events [6]. It provides software-based protection that enclave's memory can not be accessed from outside the enclave. Though the enclave shares the same address space of an unprivileged host application managed by the untrusted operating system as illustrated in Fig. 3. The trusted code inside the enclave is allowed to freely access the unprotected memory locations outside the enclave vital to carry out bulk input/output data transfers in the shared address space. An asynchronous entry/exit point can be exploited by attackers causing interrupt or

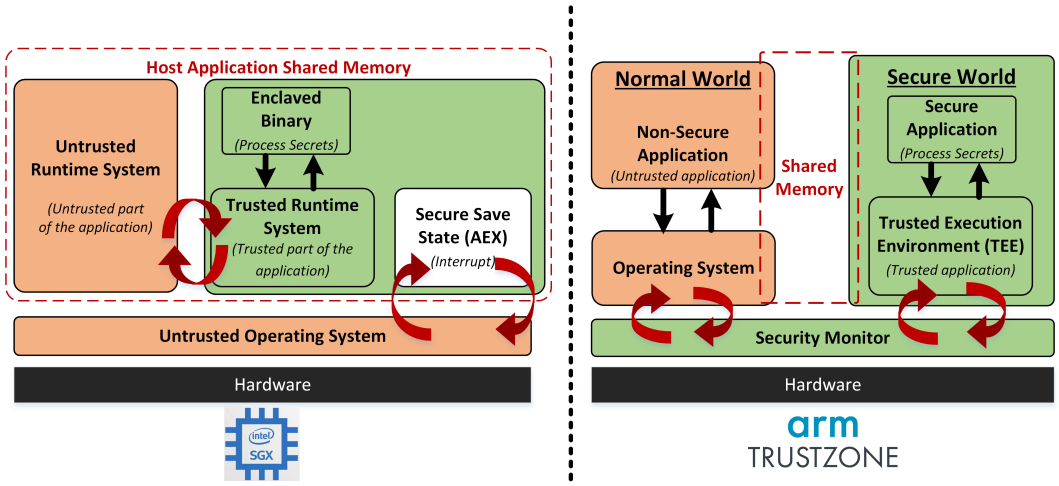

Fig. 3. Software security models of leading commercial platform security technologies by Intel and Arm [6]. The block diagram shows the isolation boundaries and specifies their cross-domain entry/exit points.

exception to the processor during secure execution of enclave as shown in Fig. 3. Both synchronous and asynchronous entry/exit points have been widely exploited to launch a range of attacks including advance side-channel, data-leakage and micro-architectural attacks e.g. Foreshadow [16], Spectre [17], Meltdown [18], RAMBleed [19], and PlunderVolt [20].

B. Arm TrustZone Technology

Arm TrustZone is a *System-on-Chip* security technology and CPU system-wide approach that provides a software-controlled hardware-enforced isolation mechanism [6], [7]. It provides foundations to realise GlobalPlatform's *Device Trust Architecture* as well as to architect robust platform security using Arm *Platform Security Architecture* guidelines. TrustZone security technology uses *virtualisation-based security* approach, where a physical processor and system resources are logically divides into two domains [13] i.e. *secure world* and *non-secure world* as shown in Fig. 3. The implementation of *secure world* requires secure software to run on the processor, to maintain code and data confidentiality and integrity [7]. To virtually isolated, a privileged software layer called *security monitor* acts as a bridge between domains [6], [7] as shown in Fig. 3. This logical isolation is further extended to system memory, peripherals and custom logic by presenting domain identifier (NS-bit) and propagating it through the system bus. To filter the domain bounded control and data communication request, *TrustZone Address Space Controller* (TZASC) and *TrustZone Peripheral Protection Controller* (TZPC) [30] enforce logical isolation between system memories and peripherals respectively as illustrated in Fig. 4.

To secure embedded edge and intelligent applications, TrustZone is available for Cortex-A and Cortex-M processors based on the Armv7-A, Armv8-A and Armv8-M instruction set architectures. The working principle of TrustZone for Cortex-A and Cortex-M processor is similar except few hardware and software differences [30] as shown in Fig. 4. For Cortex-A, the *Memory Protection Controller* (MPC) and *Peripheral Protection Controller* (PPC) use the concept of secure and non-secure address aliases to control the accessibility of each memory page or peripheral. In Cortex-M, both *Security Attribution Unit* (SAU) and *Implementation Defined Attribution Unit* (IDAU) are used together to calculate the memory map partitioning into secure/non-secure address ranges as shown in Fig. 4. Comparing their software stacks, no privileged *security monitor* software code is required to switch between domains for Cortex-M processors. The interrupts and state transitions are memory-mapped and directly managed in hardware. Thus, eliminates the domain switching overhead and efficient to meet time-critical application requirements. However there are multiple secure function entry/exit points compared to single *(Security Monitor)* in case of Cortex-A.

Though the robustness and security strength of this relies on the open component of Arm architecture. The *secure monitor* is the only entry/exit point between domains as illustrated in Fig. 3 that builds the isolation barrier. This important secure software code is developed by the software/application developers, thus prone to human errors/bugs leading to vulnerabilities [7], [13] and microarchitectural side-channel leakage attacks e.g. Spectre [17], BOOMERANG [21], Nailgun [22].

C. Hex-FIVE MultiZone Security

Hex-FIVE MultiZone security technology introduces *security through separation* to approach security vulnerabilities [7], [13], [22] caused due to integration of untrusted third-party software (libraries, APIs, drivers) into TEE. It builds security boundaries by creating a policy-based security environment for the processor. Moreover, on contrary to the traditional multi-source monolithic software development method [14] that presents greater attack surface and increased system vulnerability, the policy-based environment takes benefit of the rich operating system and enforces security through light-weight and responsive bare-metal nano-Kernel [30]. To minimise the risk of a human error, third-party integration and attack surface, a bare-metal kernel manages the hardware security

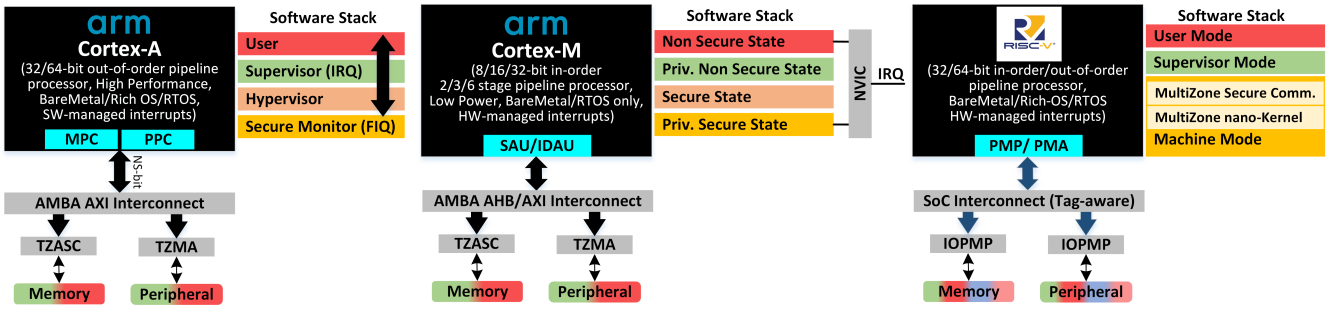

Fig. 4. Architectural comparison of Arm TrustZone Cortex-A, Arm TrustZone Cortex-M and Hex-FIVE MultiZone Security Technology [30]. Arm supports a coarse-grained, two domain software compartmentalisation, compared to fine-grained multi-domain software compartmentalisation supported by Hex-FIVE.

blocks. A policy definition is the only interaction between the developer and TEE [14]. The key software components are:

- **MultiZone nanoKernel** – a lightweight bare-metal kernel providing policy-driven hardware-enforced separation of RAM, ROM, I/O and interrupts rather than complex, multi-threaded, heavy trusted OS.
- **InterZone Messenger** – a communication infrastructure to exchange secure messages across zones on no shared-memory basis rather than vulnerable shared-memory.

From the architectural perspective, MultiZone uses similar *virtualisation-based security* approach as Arm TrustZone. Though it provides fine-grained software compartmentalisation for an unlimited number of security domains compared to coarse-grained (dual-world) compartmentalisation by Arm TrustZone as shown in Fig. 4. This fine-grained domain isolation granularity is achieved by presenting multiple bits (instead of a single NS-bit) as domain identifier and propagating them through the system communication bus. The *Physical Memory Protection* (PMP) and *Physical Memory Attribute* (PMA) controllers are responsible to filter multi-domain bounded control and data communication requests within the CPU, while *Input/Output Physical Memory Protection* (IOPMP) enforces logical separation among peripherals as shown in Fig. 4. These system-level security components enable the realisation of hardware-enforced software-defined multiple security domains [30]. Hex-FIVE introduced a software-only MultiZone Security for Arm Cortex-M devices without TrustZone technology [31],

After a detailed discussion on architectural aspects of leading commercial security technologies and their shortcomings in Section IV. It is evident that the vast majority of software security methods are not able to circumvent the explosion of security vulnerabilities and attacks launched by malicious actors. Because, they are often ad-hoc and passive, designed to mitigate a certain class of known attacks [2], [3], [4]. As they are not targeted to fix the root-causes of the underlying platform security architecture, rather focused circumvent the vulnerability itself or poor use of secure-design practices.

V. Secure-by-Design & Security Methods

To approach these diverse security challenges, there is a need for *secure-by-design* active-defence platform architecture rather than an afterthought [1], [2], [3], [4]. This needs active defences [3], [8], [9], [10] embedded within the platform's hardware and software stack complimenting each other. These defence mechanisms facilitate *detection* of malicious activities, initiate active respond against these activities and *recover* the platform and its services back to its trusted secure state [2], [3]. To design such a robust *secure-by-design* architecture, it is vital to understand the modus operandi of an attacker.

What an attacker wants? Acquire privileges access to adversely control the system behaviour and functionality.

How it can be achieved? By exploiting known vulnerabilities of a software running on the device, and force control flow diversion to gain privilege access.

To *detect* an attacker's control flow diversion, a defence mechanism is needed based on the following security methods:

① ***Trust-based security*** is a passive defence method based on the *principle-of-trust* as defined by GlobalPlatform in *Device Trust Architecture*. It builds a *chain-of-trust* which is a series of nested tasks and vulnerable as its weakest link. During system boot-up, *secure boot* technology checks the integrity of a multi-source monolithic system software to *detect* unauthorised modifications and tampering [15]. Both Arm TrustZone and Intel SGX security technologies use trust-based security. Being a passive defence method, it does not provide run-time *respond* and *recover* functions and found ineffective against wide-range of vulnerabilities and attacks specifically targeting run-time TEE and micro-architecture [6], [7], [12], [13], [16], [19], [20], [21], [22].

② ***Information Flow Tracking*** is an active defence method that provides run-time security against malicious attacks and software malfunction. A run-time execution (dynamic) information is captured and compared against defined security primitives in the form tags (metadata). A flexible security architecture not only checks the security tags against gathered execution information but also capable of dynamically updating these tags. Hence, it allows fine-grained software compartmentalisation and to *detect* run-time malicious activities and attacks, which can be used to enforce *respond* and *recover* functions. Arm CHERI and Dover Draper are the examples.

③ ***Access-Control*** is an active defence method that provides run-time security against malicious attacks and software malfunction. To better understand, the access-control method is

analogous to strategic defence of a medieval military castle, where several soldiers are strategically deployed at various locations both inside and outside of the castle. They actively police and monitor all physical movements. These soldiers hold an access-control list of personnel authorised to enter into the designated location and to access confidential information. These soldiers are analogous to the distributed *access-control* security primitive that allows fine-grained software compartmentalisation, segregation of platform resources [8]. A detection of run-time policy violation can be used to detect malicious activities, software malfunction and attack that can facilitate *respond* and *recover* functions [2], [9]. Hex-FIVE MultiZone Security for RISC-V is an example of the implementation of a policy-based access-control method.

A. Future Challenges & Discussion

To realise both active defence security methods, a new capability enhanced hardware is required composed of different resource-specific monitors (pointers, memory bounds, register permissions and decision blocks). In the case of *access-control* security method, they can be incorporated and integrated at the peripheral and memory boundaries. On the other hand for *information flow tracking* security method. they shall be embedded deep inside the processor data-path to gather run-time information and requires redesigning of the processor data-path. The integration of capability hardware components into the processor data-path would require security extended *Instruction Set Architecture* (ISA). This would increase in the critical-path and in-turn reduce the performance of the processor. From a software perspective, software-defined security primitives would be required to define the security-enhanced capability. To efficiently take advantage of the security-enhanced capability would require a tremendous software development effort, involving re-writing of the majority of currently existing software stack including operating systems, hypervisors, software libraries, device drivers etc.

VI. Conclusion

It has been evident that software security technologies are struggling to handle the explosion of security vulnerabilities and attacks. As they have been designed to mitigate a certain or class of known attacks making them ad-hoc and passive without considering the underlying micro-architectural aspects of the security technology. There is a strong need to investigate and explore embedded security micro-architectures in close conjunction to existing software approaches, and use the learning outcomes to redesign active defence platform security (*detect*, *respond* and *recover*).

References

[1] Cabinet Office, National security and intelligence, HM Treasury, and The Rt Hon Philip Hammond MP, "National Cyber Security Strategy 2016 to 2021," Tech. Rep.

[2] F. Siddiqui, M. Hagan, and S. Sezer, "Establishing Cyber Resilience in Embedded Systems for Securing Next-Generation Critical Infrastructure," in *Proc. 32nd IEEE International Conference on System-on-Chip Conference (SOCC)*, 2019, pp. 218–223.

[3] A. Kott and P. Theron, "Doers, Not Watchers: Intelligent Autonomous Agents Are a Path to Cyber Resilience," *IEEE Security Privacy*, vol. 18, no. 3, pp. 62–66, 2020.

[4] M. Hagan, F. Siddiqui, and S. Sezer, "Enhancing Security and Privacy of Next-Generation Edge Computing Technologies," in *Proc. 17th International Conference on Privacy, Security and Trust (PST)*, 2019, pp. 1–5.

[5] "Guidance on implementing the EU Directive on the security of Network and Information Systems," Tech. Rep.

[6] J. Van Bulck *et al.*, "A Tale of Two Worlds: Assessing the Vulnerability of Enclave Shielding Runtimes," in *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019, p. 1741–1758.

[7] S. Pinto and N. Santos, "Demystifying Arm TrustZone: A Comprehensive Survey," *ACM Comput. Surv.*, vol. 51, no. 6, Jan. 2019.

[8] F. Siddiqui, M. Hagan, and S. Sezer, "Embedded policing and policy enforcement approach for future secure IoT technologies," in *Living in the Internet of Things: Cybersecurity of the IoT*, Mar. 2018, pp. 1–10.

[9] F. Siddiqui, M. Hagan, and S. Sezer, "Pro-Active Policing and Policy Enforcement Architecture for Securing MPSoCs," in *Proc. 31st IEEE International System-on-Chip Conference (SOCC)*, 2018, pp. 140–145.

[10] M. Hagan *et al.*, "Enforcing Policy-Based Security Models for Embedded SoCs within the Internet of Things," in *Proc. IEEE Conference on Dependable and Secure Computing (DSC)*, 2018, pp. 1–8.

[11] P. Garcia Lopez *et al.*, "Edge-Centric Computing: Vision and Challenges," *SIGCOMM Comput. Comm. Rev.*, vol. 45, no. 5, 2015.

[12] N. Neshenko *et al.*, "Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations," *IEEE Comm. Surveys Tutorials*, vol. 21, no. 3, 2019.

[13] D. Cerdeira *et al.*, "SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-Assisted TEE Systems," in *Proc. IEEE Symposium on Security and Privacy (SP)*, May 2020, pp. 636–652.

[14] S. Pinto and C. Garlati, "Multi Zone Security for Arm Cortex-M Devices," in *Embedded World Conference*, Feb. 2020.

[15] S. Ray *et al.*, "System-on-Chip Platform Security Assurance: Architecture and Validation," *Proc. IEEE*, vol. 106, no. 1, pp. 21–37, Jan. 2018.

[16] J. V. Bulck *et al.*, "Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution," in *Proc. 27th USENIX Security Symposium (USENIX)*, Aug. 2018, p. 991–1008.

[17] E. M. Koruyeh *et al.*, "Spectre Returns! Speculation Attacks using the Return Stack Buffer," in *Proc. 12th USENIX Workshop on Offensive Tech. (WOOT)*, Aug. 2018.

[18] M. Lipp *et al.*, "Meltdown," *CoRR*, vol. abs/1801.01207, 2018.

[19] A. Kwong *et al.*, "Rambleed: Reading bits in memory without accessing them," in *41st IEEE Symposium on Security and Privacy (S&P)*, 2020.

[20] K. Murdock *et al.*, "Plundervolt: Software-based Fault Injection Attacks against Intel SGX," in *Proc. 41st IEEE Symposium on Security and Privacy (S&P'20)*, 2020.

[21] A. Machiry *et al.*, "BOOMERANG: Exploiting the Semantic Gap in Trusted Execution Environments," in *Proc. of the Network and Dist. System Security Symp.*, 2017.

[22] Z. Ning and F. Zhang, "Understanding the Security of ARM Debugging Features," in *Proc. IEEE Symposium on Security and Privacy (SP)*, May 2019, pp. 1156–1173.

[23] M. Schwarz *et al.*, "Practical Enclave Malware with Intel SGX."

[24] T. Eisenbarth *et al.*, "Reconfigurable Trusted Computing in Hardware," in *Proc. ACM Workshop on Scalable Trusted Computing*, 2007, p. 15–20.

[25] GlobalPlatform, "Introduction to Secure Elements," Tech. Rep., 2018.

[26] Arm, "WhitePaper: Platform Security Architecture Overview," Tech. Rep., June 2019.

[27] GlobalPlatform, "Introducing IoTOPIA: A practical implementation guide to secure IoT devices across all markets and in line with global requirements," Tech. Rep., October 2019.

[28] GlobalPlatform, "An Introduction to GlobalPlatform's Device Trust Architecture," Tech. Rep., July 2019.

[29] GlobalPlatform, "Introduction to Trusted Execution Environments," Tech. Rep., 2018.

[30] RISC-V Security: Arm TrustZone Technology vs RISC-V MultiZone Security.

[31] (2020) Quick Safe Alternative To TrustZone. [Online]. Available: https://hex-five.com/multizone-security-for-arm-cortex-m/