Detect and Prevent Security Vulnerabilities in your Hardware Root of Trust

Introduction

Computer hardware is omnipresent, with more than one trillion semiconductor devices sold in 2018[1]. Such large growth in the number of semiconductor devices is driven by many factors, including the rapidly expanding sector of the Internet of Things (IoT), which has resulted in the proliferation of simple microcontrollers in all kinds of devices, and the ongoing development of customized processors for new applications. For instance, highly customized Application Specific Integrated Circuits (ASICs) are used to accelerate various applications, including virtual reality, computer vision, robotics, speech recognition, and autonomous vehicles. Many of these applications previously ran purely in software on a general-purpose processor but are now being migrated to custom chipsets or Field Programmable Gate Array (FPGA) based systems.

These highly specialized ASIC and FPGA systems control many critical aspects of our daily lives. We trust our computer systems for a variety of different activities, including secure storage and transmission of financial data, identifiable personal information, and biometric data such as facial recognition characteristics and fingerprints. We rely on these datasets to support crucial infrastructure, autonomous vehicle function, and home security. Traditionally, data security was a software-based issue, but the advent of custom hardware applications has led to an increase in hardware-based attacks. As a result, hardware security is becoming more important every day. In order to secure an entire system, it is no longer sufficient to look at software alone – each layer of the computing stack must be analyzed as a system, with hardware being the basis of that system. Therefore, security may be conceptualized as a trust handoff.

Figure 1: Chain of Trust Connecting System Components from Hardware

Hardware is at the root of the trust chain. Software runs on chipsets in every system meaning that if the hardware itself is not secure the most advanced software-level defenses can still be circumvented.

However, it is important to emphasize that analyzing hardware in isolation also does not guarantee system-level security. Composing different parts of a system together can result in vulnerabilities due to incorrect assumptions made about the larger system when analyzing the security of the individual components. If there are any broken security links in the chain between hardware, boot code, firmware, operating system, and to other layers, then there may be system-wide security vulnerabilities. These system security concerns multiply as hardware is becoming more diverse, complex, and customized to provide the highest performance and flexibility for their end applications.

Many security attacks over the last several years have been system-level exploits rooted in hardware security deficiencies, highlighting the importance of examining security across both the hardware and software stack.

System-level Exploits Driven by Hardware Security Vulnerabilities

Hardware has contributed to many recent system-level exploits in numerous market verticals, including datacenters, aerospace and defense, and IoT. Securing underlying hardware from the beginning is a critical step in order to reduce the cost of an exploit.

Pre-development

In a cloud environment, datacenters provide access to compute-centered processors, custom accelerators, and Field Programmable Gate Arrays (FPGAs), enabling cutting edge applications to be deployed without the overhead of investment in a custom compute infrastructure. Datacenter hardware is shared between many different customers, leading to concerns about isolation between applications.

Traditionally, the operating system provided acceptable guarantees about process isolation, but recent attacks such as Spectre and Meltdown[2] demonstrate that a bug-free OS and bug-free software can still be completely compromised by an attacker running unprivileged software on the same machine. These attacks, and many others including Foreshadow[3] and Spoiler[4] exploit the fact that speculative execution, necessary to push performance boundaries, leaves traces of sensitive data in the hardware which can be extracted using cache timing side-channels.

While Meltdown and Spectre are advanced hardware-based attack mechanisms, datacenter hardware is often plagued with more basic vulnerabilities. One such example is an attack capable of completely replacing firmware on a server baseboard management controller (BMC) with malware[5]. The BMC enables a server administrator to perform tasks previously requiring physical access to the server, such as updating the operating system and power-cycling the system. If compromised, the entire machine is vulnerable, as much of this infrastructure hardware sits below the visibility of the operating system or hypervisor, where most security protections are built. The attack is possible because the hardware configuration of the communication bus connecting the BMC to the host machine does not authenticate transactions originating from the host machine. The result is that the private physical memory space of the BMC can be manipulated directly through this interface allowing software on the host machine to replace existing BMC firmware with malware, which allows a rogue attacker to take over the server and compromise any confidential or proprietary data that might be stored there.

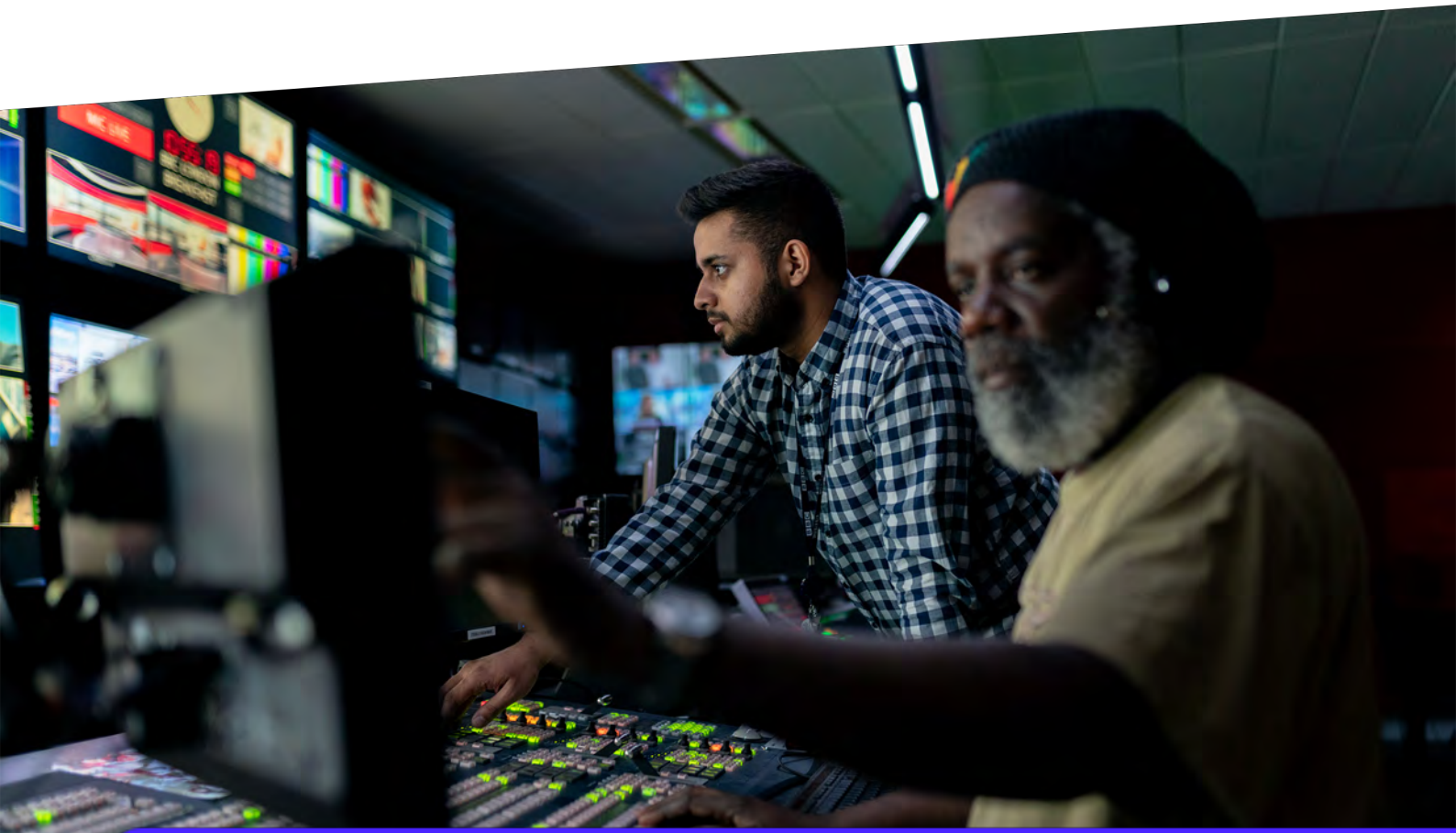
Datacenter customers often choose to protect specific portions of their application in a secure enclave. Secure enclaves are execution environments with strong hardware-enforced security guarantees, such as isolation and protection from all other processes, including the operating system. An example is SGX provided by Intel[6]. However, the presence of hardware security features does not guarantee security if those features are not thoroughly vetted. The Foreshadow attack demonstrated successful recover of secrets processed within SGX enclaves, including cryptographic material whose secrecy is critical to providing remote attestation capabilities for thousands of SGX applications[7]. Detection of hardware vulnerabilities in secure enclaves can prevent against such types of attacks in the future.

Aerospace and Defense

The security and trustworthiness of microelectronics that are used in military and aerospace applications are of utmost importance. Any successful attacks on these systems can have disastrous effects including loss of intellectual property and compromises to national security. Ensuring that these systems are designed and deployed with protections against security exploits is challenging due to the rapid evolution of threats and the growing use of microelectronics in all sectors of aerospace and defense.

Further exacerbating this problem is the diversity of the supply chain from design to tape-out, with multiple different parties with varying levels of trust responsible for each stage. To combat these issues, multiple different Department of Defense (DoD) initiatives striving to develop a portfolio of microelectronics protections and design requirements have been put in place.

For example, The DoD Microelectronics Innovation for National Security and Economic Competitiveness (MINSEC) initiative has allocated $2 billion to advance microelectronics assurance[8]. DARPA's Electronics Resurgence Initiative (ERI), as part of the MINSEC initiative, has allocated $1.5 billion dollars to advance microelectronics trust and security. New technologies to detect and prevent security vulnerabilities in aerospace and defense systems are critically necessary.



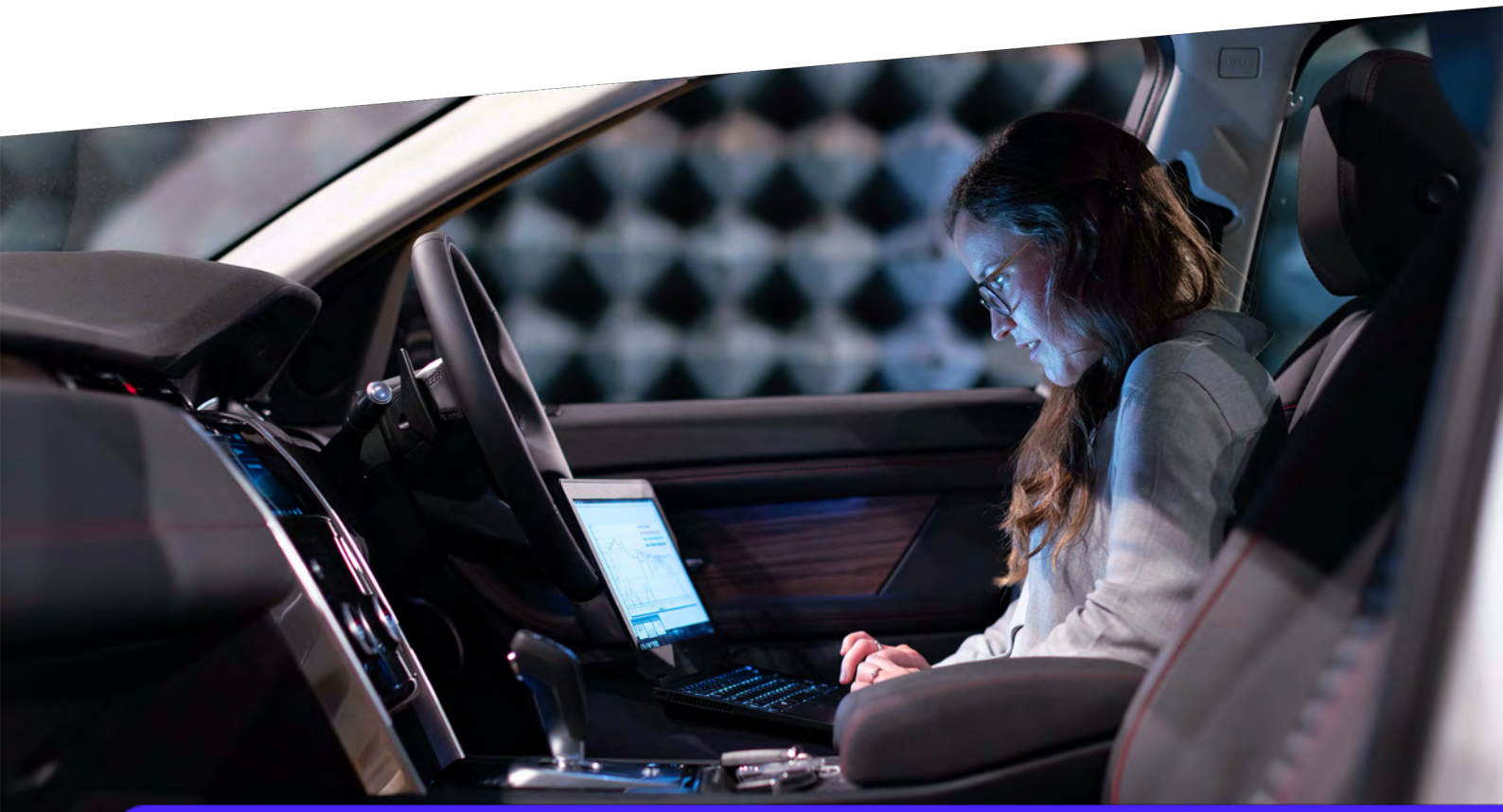
IoT

Attackers have capitalized on the proliferation of insecure IoT products by infecting large numbers of IoT devices, which can be controlled to launch large scale coordinated Distributed Denial-of-Service (DDoS) attacks. The Mirai botnet was able to infect and control approximately 600,000 devices spread across the world and successfully DDoS a number of high-profile targets, including DNS providers and telecoms[9].

Although the basic form of Mirai relies on the use of a small set of default passwords, more advanced variants exploit bugs in remote firmware update mechanisms[10], illustrating the importance of firmware authentication that needs to be built and protected into the hardware itself.

The security of chips dedicated to sending and receiving information through wireless communication protocols, such as Bluetooth Low Energy (BLE), is extremely important as BLE is commonly used in medical devices and network access points. In a recent vulnerability, BleedingBit, malicious advertising packets overflow the firmware stack provided by the hardware vendor, allowing an attacker to gain control of the chip[11].

These advertising packets can be transmitted by anyone with physical proximity to the chip, making the vulnerability extremely severe. Detecting and preventing these vulnerabilities in hardware reduces the cost of fixing these vulnerabilities throughout the supply chain.



Hardware Roots of Trust

With the proliferation of hardware and firmware-based attacks, providing trust and security services at the hardware-level is paramount. One popular way to provide on-chip security is to utilize a Hardware Root of Trust (HRoT). A Hardware Root of Trust is a minimum set of hardware and software dedicated to providing security from the moment the system is powered on.

Typically, the Hardware Root of Trust is embedded in a larger system which contains a processor designated to run a rich operating system and wide variety of applications. Security-critical functionality is offloaded to the Hardware Root of Trust, whose software and hardware is less complex than the main application, providing a smaller and more impenetrable attack surface.

Companies are either building their own Hardware Root of Trust or licensing HRoT intellectual property (IP) from a third party as part of their development. Unlike purely software-based security strategies, a Hardware Root of Trust builds core security mechanisms into the actual hardware. Depending on the end application, there are a variety of security services HRoTs typically perform. These include secure boot, secure debug, secure storage, key generation and management, secure firmware and software update,

Trusted Execution Environments (TEEs), secure communication, runtime monitoring to detect and report violations of specific security policies during system operation, and mechanisms to detect and react to physical tampering and fault attacks.

To provide these services, HRoTs include the following hardware blocks: cryptographic accelerators, one-time programmable memory, secure persistent storage, secure memory, and a microcontroller unit (MCU) for executing trusted software. Figure 2 provides a high-level block diagram for a typical HRoT.

Hardware Roots of Trust Continued...

The MCU is the "brain" of the HRoT and executes the trusted computing base (TCB), which is a small well-verified trusted software program running at the highest privilege level. This software is responsible for coordinating usage of different security features to provide a specific security service.

For example, secure boot is the procedure which brings the system out of reset and transfers control to a trusted firmware image. Secure boot requires accessing secure memory regions, retrieving cryptographic keys from secure persistent storage, and using various cryptographic accelerators to authenticate a firmware image. While all of this could theoretically be done without trusted software executing on a microcontroller, the ability to offload some functionality to software greatly simplifies the development of the HRoT, provides configurability necessary to tailor an HRoT design to many target applications, and increases flexibility. However, the opportunity for customization makes security verification of all possible system configurations infeasible making it important to verify the complete HRoT system, which includes both hardware and software components, for each target application.

The MCU trusted computing base can also provide the ability to construct trusted execution environments called containers or enclaves, which have stronger isolation guarantees than regular process execution. An application requiring a rich set of OS features and software libraries might execute on the Application central processing unit (CPU) most of the time but use the HRoT to construct an enclave for decrypting and processing sensitive data.

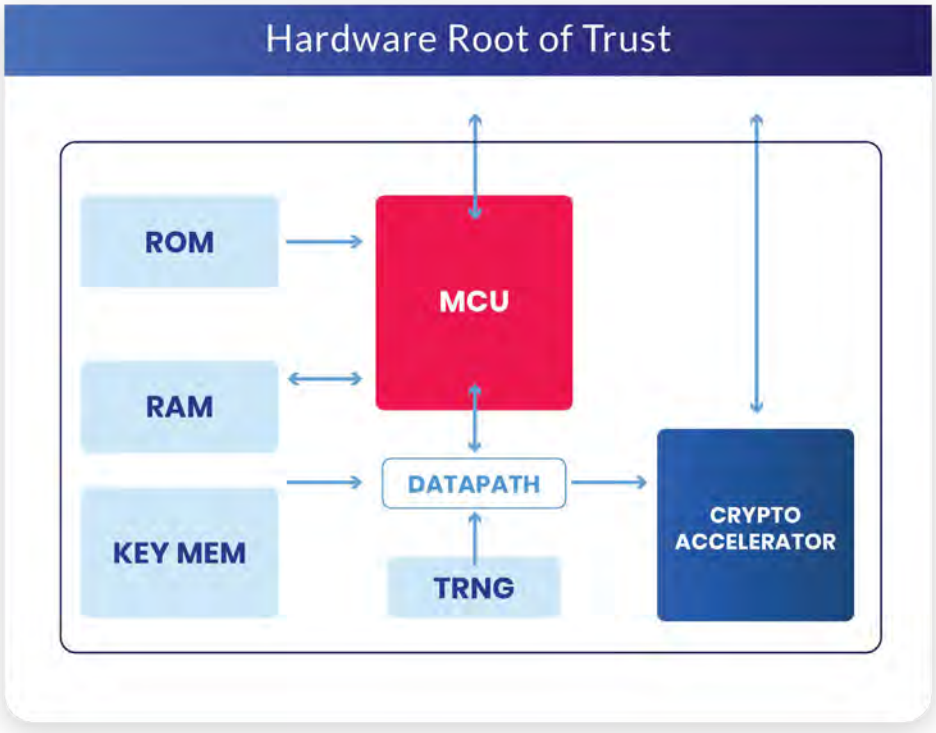


Figure 2: Hardware Root of Trust Block Diagram

A large majority of hardware blocks are completely contained within the Hardware Root of Trust boundary. However, a mechanism must exist for the HRoT to communicate with the rest of the system so software running on the application CPU can create secure enclaves and access the cryptographic accelerators. In Figure 3, the HRoT is connected to the rest of the platform through an on-chip system bus (such as an AXI interconnect).

Any interfaces connecting the HRoT to surrounding logic must be fully verified to ensure that HRoT functionality cannot be corrupted by malicious input from the external system, and that sensitive information such as device-specific encryption keys and other important assets never leak beyond the Hardware Root of Trust boundary, often called the security perimeter. Additionally, if the HRoT IP is highly configurable during the hardware design and integration phase, it is crucial to detect and prevent vulnerabilities on the specific configuration instantiated in the platform.

This requires security analysis to be performed at system level to ensure that the integration of the HRoT hasn't introduced any security vulnerabilities.

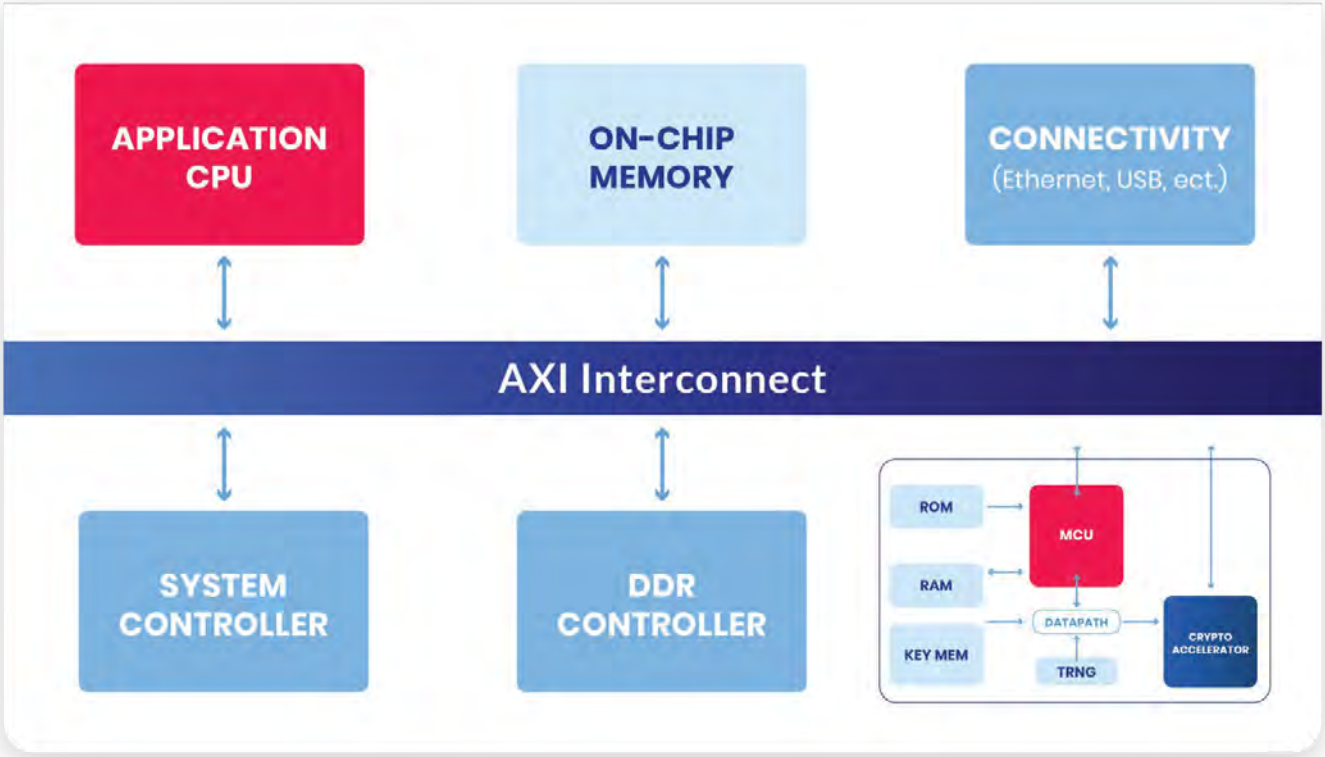


Figure 3: Hardware Root of Trust Integration within a Larger System

The Benefits of a Secure HRoT Implementation

Despite the fact that Hardware Root of Trust designs use the minimal amount of hardware and software necessary to accomplish security objectives, the complexities and intricacies in understanding whether they are introducing vulnerabilities can make one question the trust being put in them. In order for products to ship with differentiated and high value security offerings, ensuring your intellectual property and your customers' information is securely protected is essential for maintaining brand leadership and driving revenue growth. Following a concept successfully executed by Microsoft in the software domain, a Security Development Lifecycle (SDL)[12] applied to hardware provides best in class security that detects and prevents vulnerabilities while providing a robust security offering. An SDL starts with the specification of a security Threat Model and then a method for detecting violations to that Threat Model. A primary component of a hardware SDL is understanding the difference between a security feature and a secure feature.

A HRoT is an excellent security feature to improve overall system security, but it is important to demonstrate that the HRoT is a secure feature. For example, if the system relies on the HRoT to perform a secure boot procedure, but an attacker is able to bypass the secure boot process entirely, extract or modify assets used to authenticate firmware, or force certain stages to be skipped such as secure erasure of key material used in the boot process, the attacker will be able to run their own malicious code with the highest privilege level after the system is powered on and circumvent the core security features. Securing the debug and test infrastructure for the HRoT and larger system is essential. Debug and test capabilities are crucial to successful deployment of the system but are often at odds with security due to the increased visibility and access provided.

Solutions for securing the debug and test interface range from completely disabling the feature before deployment to providing the ability to expose varying degrees of debug and test functionality based on authentication and access control policies. These strategies must be implemented correctly. Errors in debug mode configuration can lead to severe vulnerabilities in the system, especially for applications in the IoT and automotive space where an attacker likely can gain physical access to the system.

Beyond secure boot and debug, other features provided by the HRoT such as memory protections and access control policies for System-on-Chip (SoC) peripherals are highly configurable either in software or in the hardware IP itself. Mistakes in configuration and usage of HRoT features have the potential to introduce vulnerabilities and often cannot be detected and prevented until the entire system is completely analyzed.

Leakage of information such as cryptographic key material, either directly or indirectly through timing-based or power-based side channels, is a weakness an attacker will exploit to gain unauthorized access to the system. Any form of indirect leakage represents a potential concern andmust be identified and addressed as the whole system relies on the HRoT for security.

Existing HRoT Security Verification Techniques

Security verification is difficult because of the fundamental asymmetry between attackers and defenders. An attacker only has to discover and exploit a single vulnerability to achieve their goals whereas the system design and verification teams must anticipate and defend against the complete set of possible threats.

Many security vulnerabilities hide within modes of the design not exercised by typical system usage and go undetected during traditional functional verification. Successful security verification requires a shift in thinking. Instead of focusing solely on "does the design function properly," the verification strategy needs to be centered around the questions: "what information in my design needs to be protected?" and "where does that information flow and how is that information accessed?"

Current methodologies being deployed to address security are simulation-based verification, manual design review, formal verification methods, and penetration testing. Applying one or all of these methods is still insufficient to address hardware security. Moreover, these existing approaches are time consuming, hard to measure, and are often a burden to engineering schedules.

These approaches are at odds with the existing verification and development process where engineering teams are focused on meeting schedules while product security teams are focused on building out unique and differentiated security products. This hurdle needs to be overcome to get a secure product to market without introducing added costs or delaying products schedules.

Manual Review

Manual review of the design architecture and code review to enumerate and prioritize threats to the system and identify potential vulnerabilities in the implementation is an important process for security verification, but requires engineers with both hardware design, verification, and security expertise.

Manual design review is not scalable, nor is it complete, and must be supplemented by other tool-aided verification methods such as simulation-based functional testing or formal analysis in order for it to yield any meaningful results on modern designs.

Penetration Testing

Penetration testing is another popular strategy for system security analysis. Often, engineers on the "red team" adopt the role of a potential attacker and perform penetration testing on the product in order to highlight areas of the system susceptible to real-world attacks and engineers on the "blue team" harden the design against possible red team attacks. Penetration testing is typically done post-silicon to better mimic the conditions under which an attacker will attempt to infiltrate the system, meaning that any vulnerabilities discovered will have to be mitigated with software or firmware to avoid a costly silicon re-spin. As with pre-silicon manual design review, penetration testing is an important component in hardware security verification. However, it requires engineers with specialized knowledge rarely available within most organizations.

Simulation-based Function Verification

Simulation-based functional verification is the work-horse of the semiconductor industry and every single digital chip designed today has undergone a significant amount of simulation-based verification before tape-out. Functional verification components include a set of tests which exercise specific design scenarios and checkers. These verify values in the design adhere to specific rules or match expected values from a golden model of the system. With respect to security, the main weakness of simulation-based verification is that security vulnerabilities are unknowns often unrelated to core design functionality. Due to modern design complexity, it is impossible to exhaustively test the entire design during simulation, therefore some security vulnerabilities are bound to go undiscovered.

Moreover, tracking where information flows in the design is at the core of the three major security objectives of confidentiality, integrity, and availability. Currently, no existing tools used in simulation-based functional verification can track information flows in the design. Labor-intensive negative tests must be developed to check security-specific corner cases for the highest priority threats, but without the ability to track information flow in the design, negative testing is extremely limited.

For example, verifying that a cryptographic key does not leak to an interconnect on the surface appears to be straightforward but in reality, is extremely difficult. It is possible to record the value of the key as it enters the encryption module then check if that exact same value appears on the system interconnect, but the key can go through an infinite number of simple transformations (ex. exclusive-or with plaintext, bit shift, etc.) from which an attacker can easily recover the original key value. These simple transformations are difficult to identify as "dangerous" using current tools, and more complex indirect leakages through timing side-channels are impossible to detect.

Formal Verification

To overcome the limitation of incomplete design coverage, formal verification is seen as an alternative to simulation-based testing for security verification. Formal verification techniques reason about properties on an abstract model of the system for all possible inputs rather than running a large number of tests. As such, if a security vulnerability violates a property verified using formal analysis, it is guaranteed to be discovered.

The limitation with formal verification is the size of the design that can be analyzed, and the set of simplifying assumptions which must be made to make verification of larger designs tractable. Formal verification may be appropriate as part of a larger strategy, but it does not solve the larger security problem crossing hardware and software. As design size (both hardware and software) increases, it becomes difficult to provide the same level of security guarantees, requiring significant manual effort to correctly model the design at an abstract enough level to draw meaningful conclusions without too many assumptions.

To combat the drawbacks related to these traditional security verification techniques, Cycuity provides security verification products that detect and prevent hardware security vulnerabilities within the existing chip verification strategies. In particular, Cycuity's Radix-S™ product is software that is used in conjunction with industry-standard functional simulation tools, enabling security verification while functional verification is being performed.

Radix is low-effort to deploy while simultaneously providing a significant increase in confidence in the security of the design at the system level. Radix integrates seamlessly with existing simulation-based functional verification environments and can increase security coverage re-using existing functional tests, eliminating the need to create security-specific test vectors. This is possible due to our patented technology capable of detecting unexpected and unidentifiable information flows in the system during functional simulation.

System Security with Cycuity Logic's Radix-S Software

Radix-S is a software package that identifies security vulnerabilities in your HRoT design, implementation, and the surrounding system. Whether you develop your own HRoT, or integrate existing HRoT IP into a larger system, Radix scans both system hardware and software during the pre-silicon design and verification simulation stages to identify system-level security vulnerabilities. Identifying security vulnerabilities at the early stages of the design cycle with Radix allows designers to prevent vulnerabilities before deployment.

Capabilities

Radix-S ships with Hardware Root of Trust Threat Models common to applications in all market verticals that span both hardware and software, along with a framework for detecting violations to the Threat Models. Radix takes as input the hardware design files typically written in a Register-Transfer Logic (RTL) language and the software executing on the system to identify any violations of the security rules. Radix does so by leveraging the customer's existing functional verification environments, so deployment does not require additional resources or modifications to existing engineering infrastructure. Examples of threats covered by the security threat models and rules include:

- Unprivileged access to your customer's proprietary or confidential data
- Unauthorized access to keys used to sign and authenticate your boot images
- Side channel leakages of critical customer information
- System-level compromises arising from HRoT misconfigurations

Directly out of the box, Radix can analyze your HRoT design and the surrounding system to protect against these security threats.

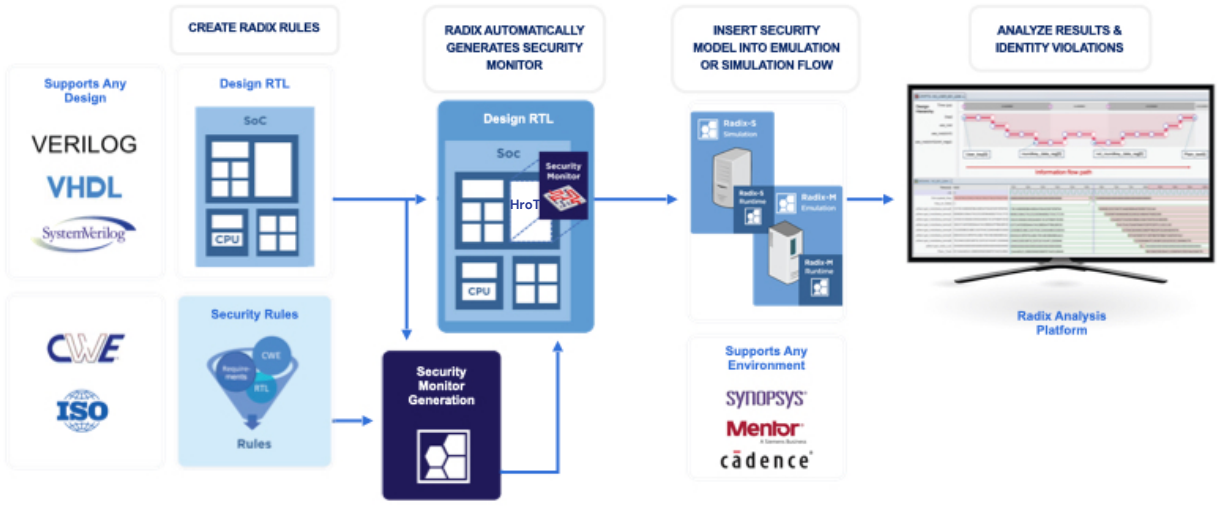


Figure 4: The Workflow of Radix

Creating a Security Monitor

Once the design RTL is available and the security rules are created, Radix analyzes the system design and the security threats that are being protected against to create a Security Monitor (SM). The SM is a monitor that continuously checks the original design and the software running on top of it for the threats that were chosen in the previous step. The SM is synthesizable hardware, allowing it to be added to simulation-based functional verification runs to check for security violations at the same time that functional tests are running. This is shown in Figure 4.

The Security Monitor integrates into standard functional verification environments without disruption to existing workflows. The scope of the security monitor is not required to be identical to the scope of the design RTL simulated in the existing verification environment. Both the threat models and SM can be developed once then re-used during different stages in the design schedule.

For example, if key leakage to the boundary of the HRoT is the focus of the security analysis, the Security Monitor generated for the HRoT can be simulated alongside the RTL for the entire SoC in order to detect key leakage occurring while actual platform software is running. Alternatively, if a suite of regression tests targeting the HRoT are available earlier, the same Security Monitor can be run in the verification environment for the HRoT.

Existing functional verification environments already include both design RTL and software to be run on the platform, meaning Radix can perform system-level security analysis of both the hardware and software without requiring the development of custom test infrastructure.

Radix Features for Efficient Security Analysis

After analyzing the security threat model on the system design, Radix provides detailed security reports regarding the system's susceptibility against the threats defined in the model. Radix identifies how many threats the chip design is susceptible to and reports analytics in order to understand the likelihood of exploit.

Radix contains an interactive platform to aid in visualization of information flow throughout the design, as seen in Figure 5. This platform includes a waveform viewer which annotates the simulation trace with data about information flows under specific threat models. If any security rules fail during simulation, Radix also provides a visualization of a concrete path through the design hierarchy showing information flow causing security rule violation. These analysis features make exploring information flow in the design more efficient and are unique to Radix.

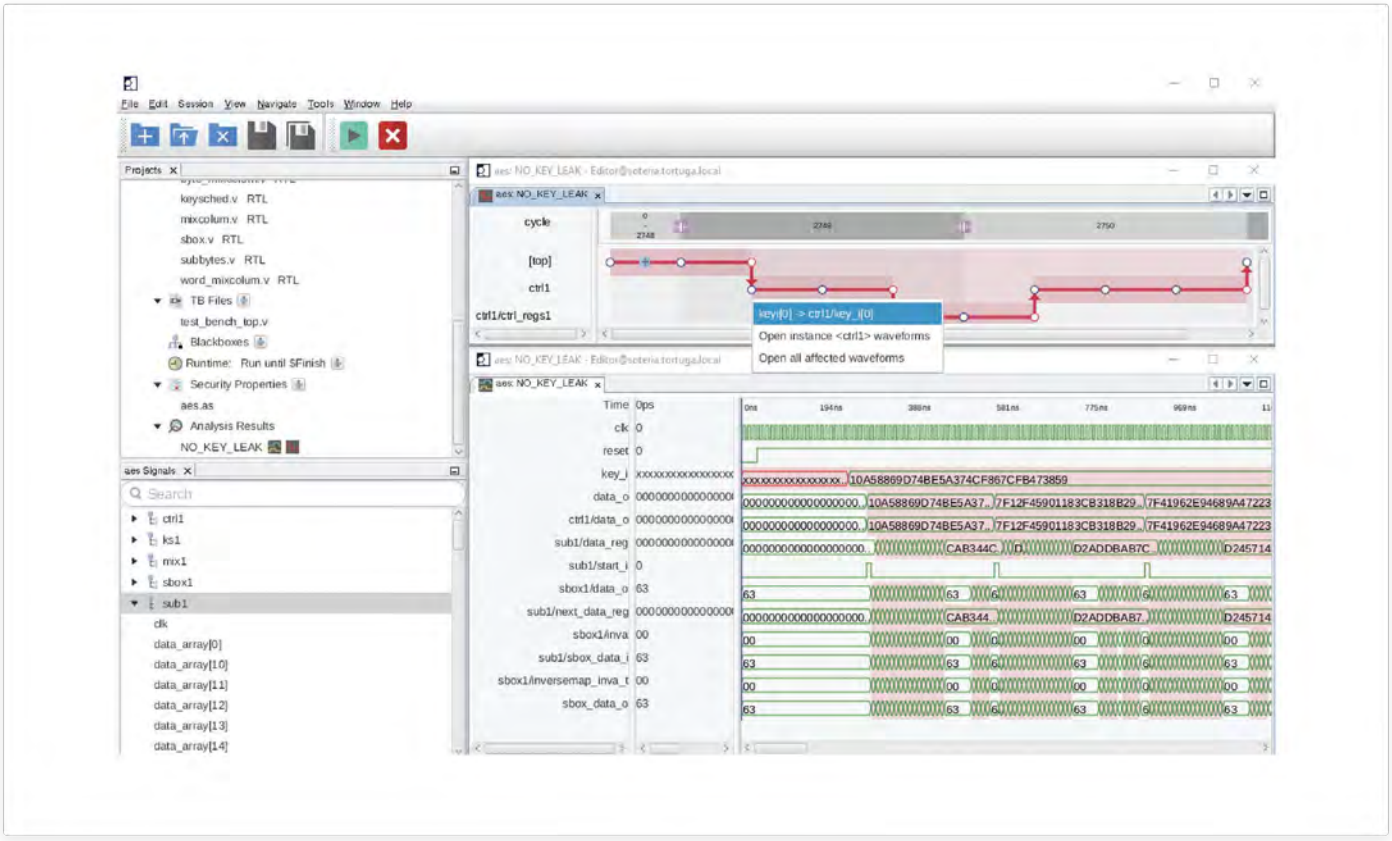


Figure 5: Radix Interactive Analysis Platform

Conclusion

Hardware is becoming more complex, customized, and ubiquitous. This is driving security features into hardware and creating more attack vectors that exploit hardware vulnerabilities. The existence and exploitation of these hardware vulnerabilities can increase time-to-market, reduce chip vendor trust, and lead to costly lawsuits and chip recalls. A popular new method for ensuring the security of a computing system is to employ a Hardware Root of Trust (HRoT), which is the foundation of security for the system. However, without security verification of the HRoT and the entire system around it, security violations may still exist.

Cycuity's Radix software checks your entire system, including the HRoT, the rest of the hardware, and the software for system-level security vulnerabilities. Radix does this by leveraging your existing functional verification environment, without causing any disruption in your workflow and without needing to write new tests specifically for security.

With Radix, security vulnerabilities can be identified and prevented before tape-out or deployment. This provides system designers with a reliable and efficient way to increase the security of their systems, resulting in many benefits including sales enablement, protection iiifrom exploits, and brand trust.

About Cycuity

Cycuity is a cybersecurity company that provides industry-leading solutions to address security vulnerabilities overlooked in today's systems. Cycuity's innovative hardware security verification platform, Radix, enables design teams to identify and prevent system-wide exploits arising around a Hardware Root of Trust that are otherwise undetectable using current methods of security review.

[1] https://www.semiconductors.org/more-than-1-trillion-semiconductors-sold-annually-for-the-first-time-ever-in-2018/
[2] https://meltdownattack.com/
[3] https://foreshadowattack.eu/
[4] https://arxiv.org/abs/1903.00446/
[5] https://www.theregister.co.uk/2019/01/24/bmc_pantsdown_bug/
[6] https://software.intel.com/en-us/sgx/
[7] Van Bulck, Jo, et al. "Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution." USENIX Security Symposium. 2018.
[8] http://www.nationaldefensemagazine.org/articles/2018/6/14/official-pentagon-investing-billions-into-microelectronics
[9] Antonakakis, Manos, et al. "Understanding the Mirai Botnet." USENIX Security Symposium. 2017.
[10] https://krebsonsecurity.com/2016/11/new-mirai-worm-knocks-900k-germans-offline/
[11] https://go.armis.com/bleedingbit/

[12] https://www.microsoft.com/en-us/securityengineering/sdl/

cycuity.com