SYSTEM IDENTITY WITH INTENT TO MITIGATE CRITICAL PART REPLACEMENT

Chirag Shroff
SYSTEM IDENTITY WITH INTENT TO MITIGATE CRITICAL PART REPLACEMENT

AUTHORS

Chirag Shroff

ABSTRACT

Presented herein is hardware fingerprinting technology that provides the ability to detect tampered hardware using a trust anchor. During manufacturing of networking equipment (e.g., routers, switches, etc.), a manufacturer fingerprints the critical hardware elements of a platform/system, such as Central Processing Units (CPUs) and/or Application-Specific Integrated Circuits (ASICs). The fingerprint is stored in the tamper resistant trust anchor. This fingerprint is then examined at every power cycle (and possibly on-demand) to determine whether a critical component has been replaced.

DETAILED DESCRIPTION

Networking equipment (e.g., routers, switches, etc.) may be susceptible to security threats when shipping from the network equipment vendor/manufacturer and is en route to a customer/user. For example, there is some apprehension that entities may replace Central Processing Units (CPUs) and/or Application-Specific Integrated Circuits (ASICs) with CPUs/ASICs containing Trojans and/or malware. There is concern that vendors of CPUs/ASICs may be influenced to put Trojans into chips, or to insert Trojans in the form of an added die in a package assembly.

Attacks on supply chains may involve replacing CPUs, ASICs, Systems-on-Chips (SoCs), etc. with chips containing Trojan/malware while the products (e.g., networking equipment) are en route to customers. Security research has shown that chip vendors can be influenced to put Trojans into chips, or to insert Trojans in form of an added die in package assembly. Chip guard techniques are described herein to mitigate these threats using unique device identifiers built and stored inside the Trust Anchor Module (TAM) of a device to identify
and track components through the life cycle of the product. Therefore, chip guard techniques may be used to ensure secure asset transfer in the product manufacturing and distribution cycle.

The following description is made with reference to FIGs. 1 and 2. FIG. 1 illustrates a conceptual flow of a manufacturing process to create an Imprint Database (ImprintDB) for chip guard initialization, according to an example embodiment. FIG. 2 illustrates a conceptual flow of a runtime validation process using an Observed Database (ObservedDB) for a chip guard mechanism used for device validation, according to an example embodiment.

To mitigate the threat of Trojans/malware added to products en route to the user, unique identifiers buried inside chips may be used to identify and track components through the lifecycle of the products. This solution is attractive because it is primarily firmware-based. Harvesting unique identifiers through a Joint Test Action Group (JTAG) interface may be untenable because of some vendors’ reluctance to share unique identifiers through such a physical interface. There are also concerns regarding overloading that interface with other functional uses. Many CPUs and ASICs expose unique identifiers through memory mapped register interfaces. Spoofing Peripheral Component Interconnect express (PCIe) (the default bus in current compute resources) memory mapped interfaces using a Trojan module is difficult (e.g., switching a PCIe endpoint with intelligence to swap for a particular memory mapped access).

Thus, chip guard techniques described herein may be used to verify a device manifest to counter chip replacement attacks. Embedded products may implement insertion of a device identifier (e.g., Secure Unique Device Identifier (SUDI)) after assembly is complete. Diagnostic software in conjunction with an automated testing infrastructure may accomplish this task. Once the SUDI is installed on a product, diagnostic software may take additional steps. The diagnostic software may query all previously identified critical components (e.g., CPUs, SoCs, ASICs, etc.) for unique identifiers, collect associated information, and provide that information to the TAM using the manufacturing user session. This process may be referred to as “imprinting.” The imprint is taken during manufacturing and stored in an Imprint Database (ImprintDB) in the TAM as a Known Good Value (KGV). ImprintDB is a master copy that stores the unique identifiers of ASICs, CPUs, SoCs and other devices with their device types specific to a board. In most cases, the unique identifier is device serial number or appropriate value of that device.
The ImprintDB is a KGV database specific to a board. It is programmed on the TAM device as part of the manufacturing process.

As the system powers up, various software entities perform the task of system initialization. During system initialization, the driver responsible for configuring the chip may be responsible for collecting unique identifiers and providing them to the TAM. This process may be referred to as “observing.” Measurements taken at each boot time may be stored in an Observed Database (ObservedDB) in the TAM. Before the system transitions to the operating state (handling user data), a query may be made to the TAM to determine whether the ObservedDB matches the ImprintDB. If the value does not match, the Operating System (OS) boot may be halted as the unmatched value indicates that the chip is compromised. The ObservedDB is collected by firmware every time the board is booted and extended to the TAM device. If some measurements are unavailable through firmware, those may be collected immediately after the available measurements from OS drivers.

The Basic Input/Output System (BIOS) boot process integrates the TAM library to populate the ObservedDB. The BIOS detects various hardware components as part of initialization and uses the TAM library Application Programming Interfaces (APIs) to record the device type and unique identifiers if the detected devices are part of ImprintDB collection. Once all the device types and unique identifiers are written to the ObservedDB, the platform OS invokes the TAM library API to validate the ObservedDB against the ImprintDB. If there is a mismatch, the platform may take appropriate action, such as alarming (e.g., printing out an error.warning message with a pre-defined error code), error logging, holding the boot process, encouraging the user to call customer support, etc. This capability may be built into the TAM with supported TAM service library APIs and/or Command Line Interface (CLI) commands.

While these systems are deployed, an operator may challenge the system with a nonce at any time and send a demand to the system identity databases (ImprintDB and ObservedDB). The TAM may return all databases to the operator with a signed response to the nonce including an ImprintDB hash and ObservedDB hash. The TAM may use an Attestation Identity Key (AIK) for such signing and present an AIK certificate in response to the command as well.
The chip guard feature may be implemented on multiple Field Replaceable Units (FRUs) of a given platform where there is a TAM device present, regardless of whether a CPU is present. Any FRU containing critical components (ASICs, Network Processor Units (NPUs), CPUs, etc.) may have its own trust anchor where unique values can be imprinted in manufacturing. TAM service may support attestation on the ObservedDB such that chip measurements may be attested by a remote attestation server and/or CLI commands.

System configuration may impact system identity, depending on the particular configuration. When the system has a fixed set of critical components, multiple compute and trust anchors within a fixed system (e.g., a system with an x86 compute and Baseboard Management Controller (BMC) based infrastructure) may create complications. Such products may support multiple TAMs and, in turn, each TAM may include a distinct ImprintDB based on the connectivity of its host CPU.

Meanwhile, modular systems (e.g., having many embedded boxes) may use “dumb” linecards and an intelligent Route Processor / Service Processor (RP/SP). A modular system may have linecards with ASICs or, in some instances, linecards with compute elements and ASICs. Each linecard may have its own TAM. In cases involving linecards with the TAM and the CPU complex, the linecard itself may contain its own system identity. In cases involving linecards containing ASICs but not a compute element, the TAM of the linecard(s) may have the system identity information imprinted thereon. Additionally/alternatively, comparison logic may run on the host system (along with an anti-counterfeit check). For a TAM, comparison logic may exist on the TAM device itself, except that observation may be performed by host software on the TAM on such a linecard. At a system level, a composite system identity response may be generated.

FIG. 3 is a block diagram of a network node (e.g., switch, router, etc.) that is configured to perform operations described herein. The network node includes a network interface unit in the form of a plurality of network ports that enable communications over a network, an Application Specific Integrated Circuit (ASIC) that performs network processing functions, one or more processors (e.g., microprocessors or microcontrollers), and a memory. The memory
includes instructions for security verification logic as well as a TAM. As shown, the TAM includes an ImprintDB and an ObservedDB.
FIG. 3

Security Verification Logic

TAM

ImprintDB

ObservedDB

ASIC

PROCESSOR

MEMORY

PORT

PORT

PORT

PORT