Securing Enterprise Server Firmware: A New Approach

A Lattice Semiconductor White Paper.

October 2018

Implementation of Platform Firmware Resilience (PFR) in hardware based on the new NIST SP 800 193 specification utilizing an FPGA-based root-of-trust device enables a new level of protection for server firmware against cyberattacks. The new Lattice PFR development toolkit simplifies the path to implement an FPGA-based PFR solution.

Learn more:

www.latticesemi.com/PFR

Contact us online:

www.latticesemi.com/contact
www.latticesemi.com/buy

Address:

Lattice Semiconductor
111 5th Ave., Suite 700
Portland, Oregon 97204
United States

Phone: 1 (503) 268-8000

Rev. 1.0 WP0016

TABLE OF CONTENTS

Executive Summary

A typical enterprise server contains multiple processing components, each having its own non-volatile SPI flash memory cache for storing its firmware (the software for the processing component to boot from immediately after powering on). While the use of flash memory is convenient for supporting in-field upgrades and bug fixes, it's also vulnerable to malicious attacks. Through unauthorized access to firmware, hackers can surreptitiously install malicious code in a component's flash memory. This malicious code can hide from standard system-level detection methods and persist through updates and hard disk swaps thereby leaving the system permanently compromised.

To address this, some processing components use on-chip hardware circuits to detect unauthorized firmware modifications. However, other processing components on the board without such countermeasures remain vulnerable, so the entire server is still exposed. To address this problem, the National Institute of Standards and Technology (NIST) released the NIST SP 800 193 specification in 2018, which defines a uniform protection mechanism known as Platform Firmware Resilience (PFR). This specification is based on three guiding principles:

PROTECT

- - -

Protect firmware against attack

DETECT

- - -

Detect compromised firmware stored in SPI flash

RECOVER

- - -

Recover by replacing compromised firmware version with a known good version

PFR functionality relies on an external hardware (silicon) "root-of-trust (RoT)" device. Use of an FPGA-based RoT device to implement a PFR solution results in a more secure, scalable and robust system compared to a MCU-based RoT option. The new PFR development toolkit from Lattice enables server OEMs to add PFR functionality to their existing designs quickly to take advantage of this new and powerful security breakthrough. System architects and system integrators can now more easily design, implement and maintain FPGA-based RoT devices that enable PFR compliance without the need for deep security expertise.

Server Firmware Vulnerable to Cyberattack

Damages caused by cybercrime are expected to reach U.S. $6 trillion by 2021[I]. Cyber attackers continually look for new ways to circumvent security measures in order to:

- See and/or steal proprietary data (credit card numbers, company IP, etc.) stored on the server
- See and/or steal data passing through the server
- Hijack the server to participate in a DDoS attack against another target
- Sabotage the server by making one or more of the server's hardware components inoperable (known as "bricking" the server)

Because operating systems and applications regularly update themselves to add new functionality or fix bugs, they make an attractive target for hackers looking to breach a server. Accordingly, organizations tend to focus their security resources and strategy on protecting operating system and application software. But there's another, less widely known attack vector for hacking a server: firmware.

Firmware includes the first bootable code executed immediately after a server component (i.e. CPUs, network controllers, RAID-on-chip solutions, etc.) is first powered up. A component's processor assumes the firmware is a valid starting point, boots from it and uses it to verify and load higher-level functionality in stages depending on the server's configuration. In some cases, the processing component uses the firmware to perform required functions throughout its entire operating life.

In a 2016 survey conducted by ISACA, over half of respondents that self-described as seeing hardware security as a priority for their organization "reported at least one incident of malware-infected firmware being introduced into a company system," and 17 percent "revealed that the incident had a material impact[II]".

The State of Firmware Security

Server firmware can be hacked at different stages in the supply chain, including:

- **At the OEM:** a malicious operator installs compromised firmware during manufacturing.
- **At system integrator:** unauthorized firmware is installed while configuring the server to meet customer requirements.
- **In transit to a customer**: a hacker downloads malicious code to component SPI memory by opening the server's packaging and downloading unauthorized firmware via a cable.
- **While operating in the field:** a hacker compromises an automated firmware update to replace the legitimate update with bogus firmware that can bypass any existing protection mechanisms.

Currently, a typical server mainboard uses at least two standard firmware instances: Unified Extensible Firmware Interface (UEFI) and the Baseboard Management Controller (BMC). While these interfaces do offer some form of firmware protection, that protection is limited.

Unified Extensible Firmware Interface (UEFI)

UEFI (previously known as BIOS) is a software program responsible for loading a server's firmware to its operating system[III]. Installed at the time of manufacturing, UEFI checks to see what hardware components the server has, wakes the components up and hands them over to the operating system. The specification can detect unauthorized firmware by using a process called secure boot: a security feature that keeps a hardware component from booting if unauthorized firmware is detected[IV]. However, implementation and support for secure boot varies between components and vendors, leaving gaps in the component's security for hackers to exploit. If illegitimate firmware manages to get past a secure boot, UEFI cannot restore the component's firmware to an earlier authorized version and continue to function.

Baseboard Management Controller (BMC)

A BMC is a specialized microcontroller (MCU) on a motherboard responsible for monitoring the physical state of a "computer, network server or other hardware device using sensors and communicating with the system administrator through an independent connection.[V]" Many BMCs do screen their own firmware installations to confirm they are legitimate, but they cannot do the same for other server components. BMCs also cannot prevent malicious code from attacking other firmware on the board. For example, if malicious code were installed in the unused portion of a component's SPI memory, the BMC could not stop the code from entering the server's entire code stream.

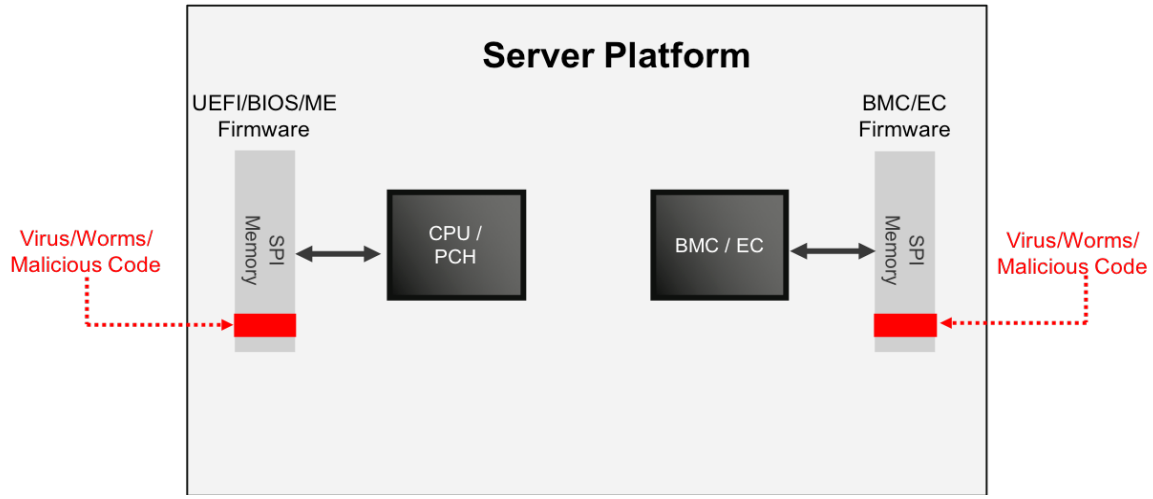

Figure 1: Unified Extensible Firmware Interface (UEFI) and the Baseboard Management Controller (BMC) interfaces offer limited firmware protection.

Platform Firmware Resiliency (NIST SP 800 193 Specification)

To address the security gaps of current firmware standards, in May 2018 NIST published a new standard to provide comprehensive protection to all firmware, including UEFI and BMC. NIST SP 800, referred to as PFR, provides "technical guidelines and recommendations supporting resiliency of platform firmware and data against potentially destructive attacks.[VI]" This specification provides a uniform method of protecting all firmware in a system and can be configured to be non-intrusive for normal system operations, yet still override any component if it determines that unauthorized firmware is attempting to install. PFR also operates independently of whatever security features individual components may support.

The specification outlines three key principles for securing firmware:

- **Protect** – Maintain the component's firmware in a reliable state by preventing unauthorized write access to protected zones of component SPI memory or attempts to erase all or part of the firmware. In some cases, read access to the protected zones is also blocked.
- **Detect** – The ability to validate firmware updates from the OEM before the component processor boots from it. If corrupt or unauthorized firmware is detected, a recovery process is initiated.
- **Recover** – If a compromised or corrupted firmware is detected, the processor boots from a previous trusted version of the firmware (called a "golden image") or enables a full-system recovery using new firmware delivered through a trusted process.

PFR Requires Hardware-based Root-of-Trust

According to the NIST specification, a secure PFR implementation requires a hardware "Root-of-Trust" (RoT) device perform the protect, detect, and recover operations for firmware in the server. A NIST compliant RoT device must perform protect, detect and recover operations on its own firmware before booting and without the aid of any other external component.

A hardware RoT solution must also be:

- **Scalable** – The RoT device must perform protect, detect and recover functions on external SPI images with nanosecond-level response times. This performance requires dedicated processing and I/O ports to keep server performance uncompromised.
- **Non-by-passable** – Unauthorized firmware should not be able to bypass the RoT device to start booting a server with compromised firmware.
- **Self-Protect** – The RoT device must dynamically address a constantly evolving attack surface (the sum of all points in a device or system where an unauthorized user can gain access) to protect itself from external attacks.
- **Self-Detect** – The RoT device must be able to detect unauthorized firmware using a non-by-passable cryptographic hardware block.
- **Self-Recover** – The RoT device must be able to switch over to an earlier golden firmware image automatically when the device discovers unauthorized firmware, ensuring that the server remains in operation.

Protection	Bad firmware detected before boot?	Recover from bad firmware?	Protect all firmware from in-system update attacks during operation?
Methods Embedded in UEFI	**YES**	**NO**	**NO**
Hardware Blocks embedded in BMC	**YES**	**NO**	**NO**
NIST PFR using RoT	**YES**	**YES**	**YES**

Figure 2: Current firmware standards cannot protect component firmware during all phases of operation.

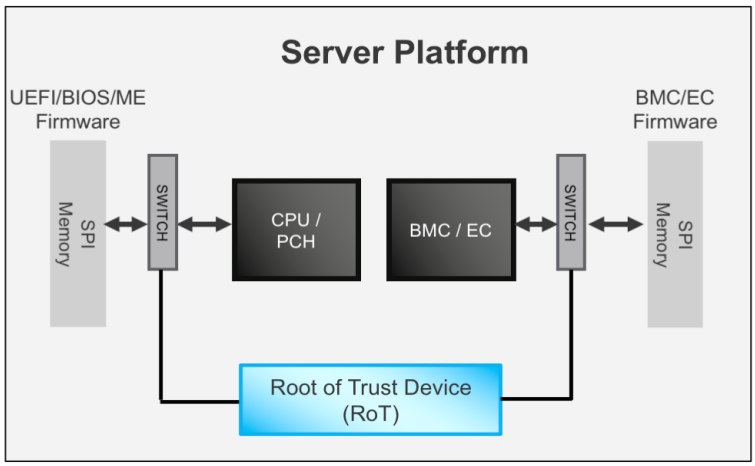

Figure 3: NIST SP 800-193: Platform Firmware Resilience

As Figure 3 illustrates, the RoT device turns on first to cryptographically inspect all component firmware and detect unauthorized modifications. If the RoT detects any corruption, it initiates a trusted firmware recovery process. In extreme cases, when all firmware on the board is compromised, the RoT device can initiate a full-system recovery (through the BMC) by using trusted firmware stored in the RoT device. After the BMC boots from trusted firmware, it retrieves known good firmware from outside the system and replaces the compromised firmware. The RoT device then revalidates all firmware and initiates a board-level power-on procedure in which all the components on the board are turned on and forced to boot from their known good firmware image, and begin normal operation.

To protect the SPI memory from future corruption attempts, the RoT actively monitors all SPI traffic between the SPI memory and its corresponding processor to detect malicious attempts to update firmware and stop its installation.

NIST compliant PFR Implementations

The challenge of implementing root-of-trust in a PLD is doing so in a manner that does not overly burden the OEM. A root-of-trust hardware solution (including a PLD-based solution) must be scalable, meaning it can protect all firmware in the server with response times measured in nanoseconds. It should be able to detect the compromised firmware through cryptographic measurements using a non-modifiable cryptographic block. Inclusion of the full boot sequence control for all server components in conjunction with the PFR implementation makes it impossible to bypass the RoT. Finally, the solution should be able to switch back to an earlier golden firmware image automatically so the server can continue to operate if it discovers a breach of the current firmware.

A hardware-based RoT device, by definition, must be implemented in silicon, and the most commonly used silicon platforms for this purpose are microcontrollers (MCUs) or field programmable gate arrays (FPGAs). An examination of FPGA and MCU operating characteristics and features demonstrates that an FPGA platform best supports PFR at scale.

Root-of-Trust Implementation Using MCU

In the past, MCUs have been used in server hardware products to establish a root-of-trust. In short, a portion of the MCU surface area is reserved for a Trusted Execution Environment (TEE), a section of the MCU physically isolated from the rest of the chip that continuously monitors firmware to confirm it is authorized and functioning normally. Generally, PFR functionality is added to the server by adding a RoT MCU to the existing hardware architecture.

MCUs have limited ability to support verification of multiple firmware instances in a server. This is because they are not able to respond to in-system attacks against all firmware instances in a server without the help of an external device such as a PLD (which monitors the SPI memory traffic in real-time and detects and responds to breaches simultaneously).

The three components used to implement PFR using an MCU illustrated in Figure 4 are:

- **RoT MCU** – The RoT MCU performs detect, recover and protect functions; it is the central component for the RoT implementation.
- **Protect PLD** – Enables PFR implementation at scale for comprehensive protection of the board by simultaneously monitoring SPI transactions between all component processors and their SPI memory devices.
- **Control PLD** – This device integrates all board level power-on and reset sequencing functions with other functions like fan control, SGPIO, I²C buffering, signal integration, and out-of-band communication needed to boot the main board. The RoT MCU commands the control PLD to initiate board power-on. If an extreme recovery is needed, the RoT MCU commands the control PLD to only power on the section of the board used in the trusted recovery process.

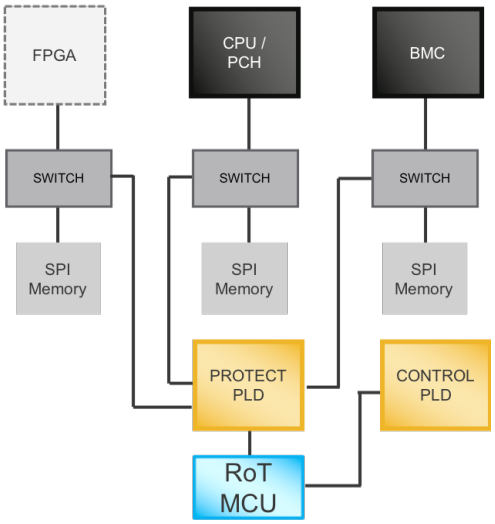

Figure 4: A PFR compliant server using an MCU for root of trust requires additional components (FPGAs) to provide performance if components are to boot simultaneously; a solution that is not scalable for high volume server applications.

The MCU-based approach to PFR has limitations. For example, the control PLD used in the circuit diagrammed in Figure 4 cannot protect its own firmware, meaning this architecture is not fully compliant with the NIST PFR specification. It would be possible to modify the control PLD code and render the RoT MCU ineffective. It is also possible for a permanent denial of service (PDoS) attack to render the system inoperable by erasing these PLDs, making the server incapable of booting. The protect and control PLDs' security gaps make it hard to protect against firmware attacks when the component is in transit or during system integration. In order to be NIST SP 800 193 standard compliant, the RoT MCU must implement PFR functionality for the control PLD and the protect PLD. Implementing recovery and protection functions for those devices using the MCU is difficult. Finally, the MCU-based approach requires additional system-level processes to detect an attack trying to bypass the entire RoT circuit.

Root-of-Trust Implementation Using FPGA

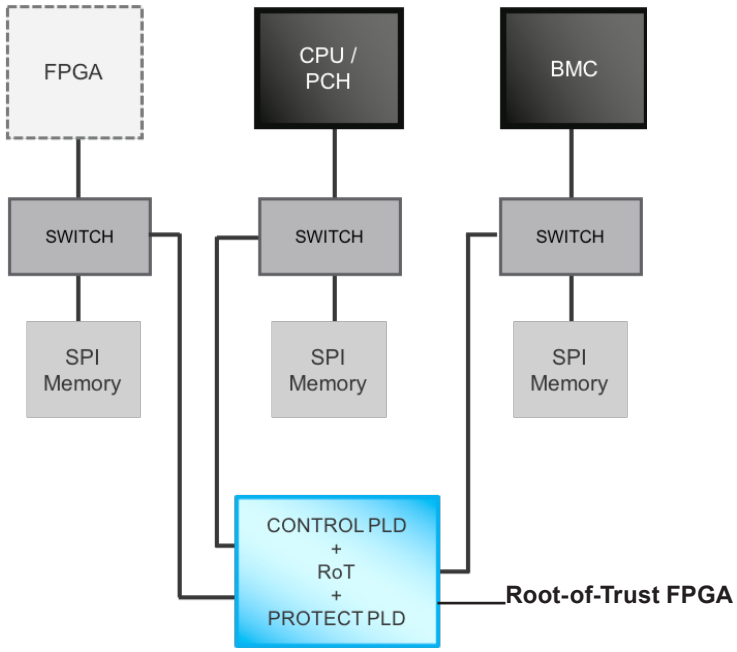

Figure 5: The RoT FPGA integrates the functions of the RoT MCU, Control PLD and the Protect PLD in a single-chip solution that is robust, scalable and impossible to bypass. In a server with a PFR-compliant PLD, the PLD's performance can monitor all component firmware in parallel without the need for additional FPGAs.

Benefits of RoT FPGA based PFR Implementation

As their name implies, PLDs are a type of integrated circuit that can be reprogrammed remotely and nearly instantaneously to adapt to changing conditions. A PLD can physically change its circuitry in a way that, once it detects the presence of unauthorized firmware, makes that firmware unable to install.

Because they are designed to be reprogrammable, PLDs have more I/O connections than MCUs, allowing them to perform multiple functions in parallel instead of in sequence. This makes them much faster at identifying and responding to unauthorized firmware when detected.

Additionally, PLDs use sophisticated simulation software so engineers can confirm their PLD design functionality. These same tools allow engineers to test their designs against various firmware cyberattacks to confirm the PLD can protect itself. Firmware updates for an MCU need more elaborate testing and verification than PLDs because MCUs cannot support functionality verification through simulation. Instead, any updates to MCU firmware have to go through repeated regression (trial and error) testing to confirm the new firmware will not adversely affect other functionality in the MCU; a much lengthier process than running PLD simulation software.

When the characteristics of PLDs and MCUs are compared, it is clear that PLDs provide a higher performing, more robust platform for implementing a RoT in hardware; a necessity for implementing the PFR standard.

Response to Supply Chain Attack: MCU vs. FPGA PFR Solution

If a firmware attack occurs, the two different types of PFR systems take the following measures (in order of implementation):

RoT MCU	RoT FPGA
Detection: The RoT MCU inspects all SPI memory devices by sequentially performing cryptographic measurements to detect the presence of unauthorized firmware. A control PLD (compromised in the supply chain) can bypass detection checks by RoT MCU and make the BMC boot a compromised image. If compromised firmware is detected, the recovery process is initiated by either the protect PLD managing the boot source SPI memory, or through either the control or protect PLD and monitored by the RoT MCU. After the server completes a full boot, the protect PLD actively monitors all SPI transactions simultaneously to block future attacks and inform the RoT MCU when a breach is detected.	**Detection**: The RoT FPGA inspects all SPI memory devices by sequentially performing cryptographic measurements to detect the presence of unauthorized firmware. The FPGA logs a fault within on-chip non-volatile memory for future analysis. The RoT FPGA protects itself from attacks in the supply chain. FPGA-based system integrates this functionality into its hardware. No external communication between RoT and control PLD is needed. That makes the solution more rugged and immune to external attacks. The resulting solution is simpler and fully complies with NIST Standards.

PFR Development Toolkit Eases Path to Implement a FPGA-based RoT

Lattice now offers a PFR development toolkit to simplify FPGA-based RoT implementation. Server component OEMs and system integrators implementing FPGA-based PFR can move forward more quickly to meet time to market demands. The Lattice toolkit includes a software library of functions along with associated IPs and three development boards to implement PFR (including the protect PLD functions). The board-control PLD functions are added into the RoT FPGA design using the Lattice Diamond software tool. The Lattice PFR development toolkit development boards include:

- A RoT FPGA development board.
- An ECP5 FPGA board running a Python script to simulate the server's BMC. Developers can execute commands from the Python script to simulate attacks against components' SPI memory.
- A PFR adapter card that stores BMC code in SPI memory. The PFR function implemented in the RoT FPGA on the development board protects the PFR adapter card firmware from attack (meaning the FPGA-based solution is NIST PFR compliant).

The Lattice toolkit enables users to design, implement and maintain NIST compliant custom PFR implementations without the need for deep security expertise.

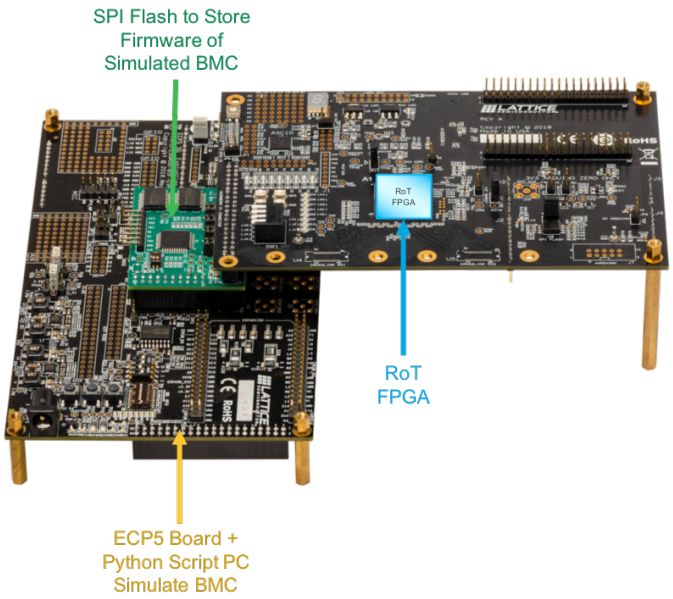

Figure 6: The Lattice FPGA RoT development toolkit features three boards: RoT FPGA development board, an ECP5 board to simulate a server's BMC and a SPI flash board to store the simulated BMC firmware.

Summary

For organizations operating in the digital domain, cybersecurity is a critical issue. Hackers are now targeting enterprise server firmware to gain unauthorized access to data on a server or even render the server permanently inoperative. To combat this, PFR implementation in an FPGA-based RoT device now offers a robust, scalable, and complete based method to protect server component firmware against attacks at any stage in the component supply chain. The new Lattice PFR development toolkit offers a path to accelerate and simplify RoT device development for server security.

---

[i] https://www.csoonline.com/article/3153707/security/top-5-cybersecurity-facts-figures-and-statistics.html
[ii] http://www.isaca.org/Knowledge-Center/Research/Documents/CSX-Firmware_whp_eng_1016.pdf?regnum=461390
[iii] https://whatis.techtarget.com/definition/Unified-Extensible-Firmware-Interface-UEFI
[iv] https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/oem-secure-boot
[v] https://searchnetworking.techtarget.com/definition/baseboard-management-controller
[vi] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-193.pdf

Learn more:

www.latticesemi.com/PFR

Contact us online:

www.latticesemi.com/contact
www.latticesemi.com/buy

Address:

Lattice Semiconductor
111 5th Ave., Suite 700
Portland, Oregon 97204
United States

Phone: 1 (503) 268-8000