SOLUTION BRIEF

Zero Trust For Devices

Establishing Chip-Level Trust in Federal Agencies

Recent mandates and regulations have declared the necessity for federal networks to establish and maintain Zero Trust security postures. These include but are not limited to:

- Executive Order (EO) 14028, which called out the need for federal networks to "adopt security best practices" and "advance toward Zero Trust architectures".
- M-22-18 *Memorandum For the Heads of Executive Departments and Agencies*, which aligned Zero Trust goals with FISMA mandates and set forth timelines for completion.
- NIST SP 1800-35 (Draft) *Implementing a Zero Trust Architecture* (Preliminary Draft) which will lay out a comprehensive strategy to balance access and productivity with Zero Trust concepts like least privilege, default deny, and risk context.

But how will these guides and requirements impact our information and communication technology (ICT) devices? How will they stand up to the multitude of embedded components that exist *within* these devices? This solution brief will answer those questions and give strategists and practitioners guidance in achieving Zero Trust through hardware.

OVERVIEW

WHO SHOULD READ THIS:

Federal security leaders, including authorizing officials (AOs), CISOs and CIOs; security and network architects; teams responsible for data center security, infrastructure security or network security who are planning and executing Zero Trust projects.

WHAT THEY WILL LEARN:

How and why Zero Trust principles apply to devices, and how these principles need to extend down to bare hardware – chips, processors and system components.

FURTHER READING:

TAG Cyber white paper, "Making the Case for Firmware in the Context of Zero Trust Security"

Eclypsium white paper on Executive Order 14028

Teledyne Lecroy & Eclypsium joint white paper, "Applying Zero Trust in the Supply Chain to Prevent DMA Attacks"

UNDERSTANDING ZERO TRUST

The O'Reilly textbook, *Zero Trust Networks: Building Secure Systems in Untrusted Networks* by Evan Gilman and Doug Barth, sets out five basic principles at work in a Zero Trust network design:

1. The network is always assumed to be hostile.
2. External and internal threats exist on the network at all times.
3. Network locality is not sufficient for deciding trust in a network.
4. **Every device, user, and network flow is authenticated and authorized**.
5. **Policies must be dynamic and calculated from as many sources of data as possible.**

This brief focuses on Principles 4 and 5 from the Wiley book, where the Zero Trust "rubber" hits the real-world "road." It's here we start thinking about the natural extension of Zero Trust principles down into the devices and hardware they use, while also calculating the unique, highly dynamic risks not only of our devices, but of the millions of internal firmware and hardware components that make them whole.

ZERO TRUST FOR EVERY DEVICE

Our networks are made up of two entities: people and devices. "People" are the realm of identity and access management (IAM) programs. Devices are ubiquitous, and come in a wide array of forms and uses: endpoints, servers, networked and connected devices, security devices and IoT/OT devices. By now there's little argument that each device needs to be uniquely authenticated, either through user interaction, embedded X.509 certificates, SSH keys or other methods. Inherited trust or legacy permissions aren't allowed in a Zero Trust network.

As the Riley books says in Principle 4, "Every device, user, and network flow is authenticated and authorized."

> **"Every device, user, and network flow is authenticated and authorized"**

But of course every one of these devices is really an amalgamation – a working collection – of other devices: central processing units (CPUs), memory, PCI cards, solid state drives, system management modules (SMM), baseboard management controllers (BMCs) in servers.

Just as the Zero Trust network no longer assumes the identity of a person, it no longer assumes the identity of the whole device. It gets rid of that assumption and replaces it with active, real-time authentication of all the addressable components within those devices. How do we verify device components? Some use signed certificates and keys, but for the majority we can verify them through the firmware and microcode embedded in them by their legitimate manufacturer.

According to research from analyst firm Gartner:

- Every endpoint is delivered, on average, with 15-20 firmware components
- Every server is delivered with around 30 components, and sometimes more than 50
- Every network device is now shipped with dedicated firmware

This firmware represents potential risk, but it also helps us uniquely establish – through version numbers and hash comparisons – the "integrity" of the devices in which it's contained.

ESTABLISHING ZERO TRUST THROUGH FIRMWARE

Firmware is simply software code that's been embedded directly onto hardware components by that hardware's manufacturer, or one of their suppliers. It doesn't reside in common storage locations for files and data, but in specialized chips.

Firmware is increasingly used as an initial attack vector. This table relates common hardware or firmware-based vulnerabilities and the recent exploits that leverage them.

Component	Role	Vuln?	Exploited?
Central Processing Unit (CPU)	Often called microcode, CPU-level firmware is powerful and privileged. Microcode firmware typically resides in special high-speed memory and translates machine instructions, state machine data, or other input into sequences of detailed circuit-level operations.	Yes	Dirty Cow Spectre Meltdown
Unified Extensible Firmware Interface (UEFI)	A specification that defines a software interface between an operating system and platform firmware. UEFI replaces the legacy Basic Input/Output System (BIOS) boot firmware.	Yes	Moon Bounce Cosmic Strand TrickBoot
Trusted Platform Module (TPM)	An international standard for a dedicated microcontroller designed to secure hardware through integrated cryptographic keys. The term can also refer to a chip conforming to the standard.	Yes	Various probing, side- channel, interposer attacks
Management Engines (ME)	Intel's autonomous subsystem that has been incorporated in virtually all of Intel's processor chipsets since 2008.	Unk	Conti is focused here
Baseboard Management Controller (BMC)	Provides the intelligence in an Intelligent Platform Management Interface (IPMI). It is a specialized microcontroller embedded on the motherboard of a server computer. The BMC manages the interface between system- management software and platform hardware through dedicated firmware and RAM.	Yes	iLOBleed USBAnywhere
Network Card (NIC)	A network interface controller (NIC, also known as a network interface card, network adapter, LAN adapter or physical network interface, and by similar terms is a computer hardware component that connects a computer to a computer network.		EtherLED NetSpectre
Direct Memory Access (DMA)	A feature of computers that allows certain hardware subsystems to access main system memory independently of the central processing unit (CPU) and run commands through on-board RAM.	Yes	Various DMA and side channel attacks
Embedded Controller	A microcontroller in computers that handles various system tasks. Usually merged with Super I/O, especially on mobile platforms.	Unk	Multitude • Firmware-based • Network-based • Side-based
System Management Module (SMM)	Chips that enable System Management Mode, which when active provides an alternate firmware-based software system with higher privileges.	Yes	HPE devices AMD chips
Solid State Drive (SSD)	A solid-state storage device that uses integrated circuits to store data persistently, typically using flash memory, and functioning as secondary storage.	Soon	SSD attacks Micron Flex

In addition to these common components, most computers have additional chips for video processing, sound processing, digital signal processing, and other application-specific integrated circuits (ASICs). Each of these components represents a source for compromise or vulnerability. Each must be included in the universe of data we use in the decision-making processes of Zero Trust systems. By assessing the attributes of this low level code – its version, source date, binary signature and provenance – practitioners can build trust in these underlying (and invisible) components.

How do we make those critical Zero Trust decisions? We assess chip-level risk.

ASSESSING CHIP-LEVEL RISK

The 5th and final point from the Zero Trust Networks book cited earlier is both specific and almost impossibly broad: "Policies must be dynamic and calculated from as many sources of data as possible."

> **"Policies must be dynamic and calculated from as many sources of data as possible"**

"Policy" refers to the output from a trust engine. A trust engine, in turn, calculates risk based on system inputs. Given the data on hand, do we trust this component or not? Can we allow this device or this user on the network, or not?

We can generate a Zero Trust test case using an example from the "BMC" row in the table above:

- A subject system on the network is an HPE Gen9 server using iLO4, HP's "integrated lights out BMC" module
- A process (or person) wants to store critical or sensitive data on this server
- But as this post explains, it's difficult to tell whether an implant is at work on this HPE server without doing a firmware-level scan
- Do we trust it? Or do we explicitly distrust it?

The answer depends on the state of the firmware: if an independent firmware-level scan CAN confirm the version of the iLO4 component, we allow it. If we CANNOT confirm the version or the absence of the malicious firmware implant we have to assume this server has been compromised. The trust engine needs to harvest those data points from system firmware and make it available throughout the supply chain.

The key? Traditional vulnerability management, or even the most up-to-date endpoint solutions like EDR and XDR, were never built to assess this level of integrity. They were not designed to harvest data points below the OS, down in the mother board's BMC module.

TRUSTED FIRMWARE IS THE PATH TO ZERO TRUST IN FEDERAL DEVICES

At its core, Zero Trust is about rooting out areas where trust is assumed, based on a perception of risk, and replacing that assumption with active verification of a trust state. For many federal agencies and organizations, firmware and hardware has been a persistent blind spot that has been trusted "by default". This is not only a bad security strategy, it's now contrary to the federal mandates requiring Zero Trust.

Instead, practitioners can use active assessment of the device's firmware to validate Zero Trust posture and integrity for all new devices joining the network. Then, continual assessment of that firmware – as devices are used and updated – provides a full lifecycle approach to achieving Zero Trust for endpoints, servers and network devices.

HOW CAN ECLYPSIUM HELP?

Eclypsium's platform helps cyber security teams extend their Zero Trust program all the way down to the base hardware in their endpoints, servers and critical network equipment. To learn more about this end-to-end approach to achieving a Zero Trust network posture, visit the Eclypsium website or schedule a demo through email.