
# THE 2021 CYBERSECURITY EXECUTIVE ORDER: ZERO TRUST, FIRMWARE IN THE SUPPLY CHAIN, AND THE DEMAND FOR DEVICE INTEGRITY

The May 12 *Executive Order on Improving the Nation's Cybersecurity* is a watershed document that introduces new perspectives and directions for the prevention of cyber attacks. Two sections of the Executive Order stand out as clear mandates for Federal Agency cybersecurity teams, but also as innovative, new approaches for civilian teams who need to uplevel their strategies to counter new adversary tactics:

1. Section 3, the which calls for "Modernizing Federal Government Cybersecurity," focusing especially on the design and implementation of Zero Trust architectures in government networks, and;
2. Section 4, which concentrates on strengthening and securing the software supply chain.

While all ten sections of the Executive Order serve as clear instructions for federal agencies and forward-thinking guidance for CISOs in the commercial sector, these two sections mark significant departures from previous best practices. They also highlight a need for security architects, analysts, and threat teams to begin taking seriously the role of firmware in establishing device integrity.

## SECTION 3: ZERO TRUST ARCHITECTURE IN THE EXECUTIVE ORDER

Among a number of key takeaways, the executive order triggers a chain of logic that calls for attention from CISOs, security architects and practitioners alike:

- For cybersecurity programs to be successful they must rely on Zero Trust strategies, tactics and postures
- A successful Zero Trust program must have an active, expanded understanding of Device Integrity
- Device Integrity in turn requires deep, firmware- and hardware level discovery, evaluation, and remediation capabilities

The Executive Order points the reader to standards and documentation from the National Institute of Standards and Technology (NIST) and other federal sources. One of those, NIST SP 800-207, defines "Zero Trust" and provides critical context for understanding the scope of Executive Order:

> "Zero trust (ZT) is the term for an evolving set of cybersecurity paradigms that move defenses from static, network-based perimeters to focus on users, assets, and resources. ...
>
> "Zero trust assumes there is no implicit trust granted to assets or user accounts based solely on their physical or network location (i.e., local area networks versus the internet) or based on asset ownership (enterprise or personally owned).
>
> "Authentication and authorization (both subject and device) are discrete functions performed before a session to an enterprise resource is established."

The NIST special publication goes on to say, "Zero trust focuses on protecting resources (assets, services, workflows, network accounts, etc.), not network segments, as the network location is no longer seen as the prime component to the security posture of the resource."

This definition urges cybersecurity strategists, implementers, and practitioners to rethink the inherent, implicit, and increasingly tenuous trust the industry has previously placed in endpoints, servers, and devices throughout modern networks.

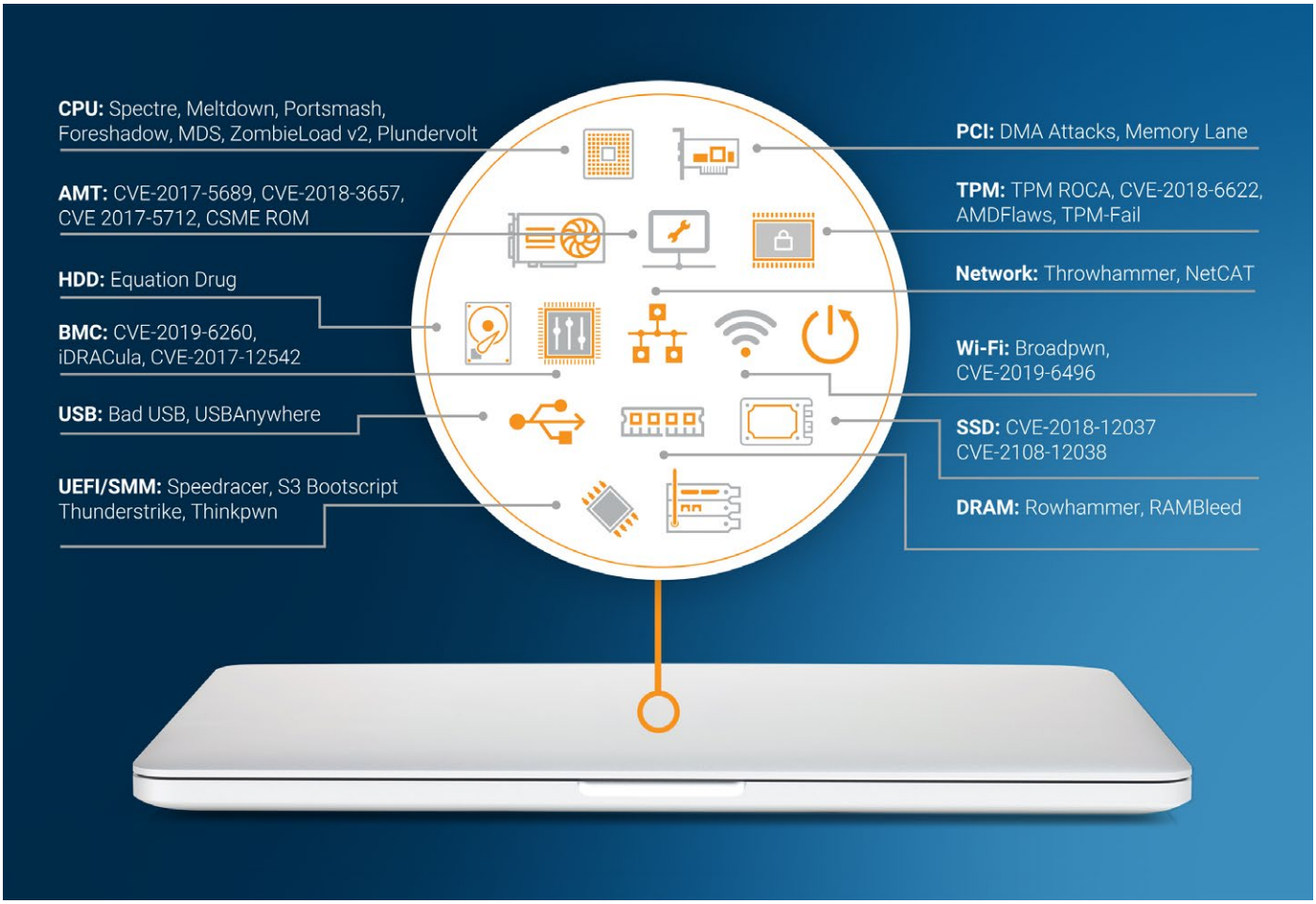
## SECTION 3: ZERO TRUST AND THE DEVICE

The key point from the NIST description above is a significant shift in perspective for modern CISOs: Zero Trust strategies will force cybersecurity teams to move from a "construct" level to an "atomic" level. To use a metaphor, we can no longer focus, as a construct, on better, higher, stronger castle walls. We must instead be constantly, intimately, and immediately aware of the security posture of every person, house, cart, and pet inside the castle, at all times and in real-time. This continual awareness needs to encompass the risks, threats, and defensive capabilities of every entity in the network.

The *Executive Order* is very deliberate in its emphasis on adopting "Zero Trust Architectures" in both design and actual practice. Zero Trust as a security strategy assumes a few specific concepts that need to understood and affirmed:

1. **Default Deny:** every new connection from every device must be denied by default rather than approved by default until successfully authenticated for every session. Previous permissions or authentications can no longer be "inherited" from previous sessions.
2. **Contextual Authentication:** That authentication must be contextual in nature, dependent on the risk, threat, or security posture surrounding that device and its user at the current moment. An authentication granted yesterday may not work today if the level of detected risks or vulnerabilities has changed.
3. **Granular Control:** Not just "default deny" and "contextual authentication" for every entity or member of a system, but every component within the system. This means groups of devices -- as well as people -- cannot be approved without uniquely authenticating and authorizing its individual members or components. We will expand on this idea in the next section.
4. **Dynamic and Real-Time:** Static, inherited lists of assets and their components, whether users, hardware, or software, are no longer acceptable. Because of the speed driven by our digital transformation efforts, cybersecurity systems that detect assets, users and workloads should be highly dynamic and real-time in nature. For example, these systems should assume that every device in the inventory today may be replaced tomorrow. They should be capable of discovering new devices or device components in real-time as soon as they enter the environment.

To be sure, there are additional components in creating and executing Zero Trust strategies. But these four points provide an initial and foundational understanding.




*Systems assessing for device integrity will include hardware- and firmware-level assessment of not just vulnerabilities, but also anomalies and misconfigurations*

## SECTION 3: ZERO TRUST IN THE CONTEXT OF FIRMWARE

The four principles above provide a framework for understanding how Zero Trust strategies must apply to the less visible parts of the network, namely hardware and its attendant firmware. Firmware is pervasive in every computing device. A typical laptop computer has more than a dozen internal components such as UEFI/BIOS system firmware, Trusted Platform Modules (TPM), peripheral devices, storage devices, or network interface cards. Each component runs millions of lines of code, developed by a myriad of vendors in a complex supply chain.

1. **"Default Deny" in Firmware:** InfoSec practitioners tend to think of the "Default deny" concept as an authentication issue, and in the broadest sense, it is. But we need to look past the act of "authenticating" an entity, whether user or device, using traditional authentication factors of Knowledge, Possession, or Inherence. If authentication means "corroboration of a claimed identity," then we must corroborate all the associated parts of that claimed identity, including, if applicable, the bare metal hardware, its various components, and the underlying, embedded firmware that instructs it on how to run and ensures it has not been tampered with. The principle of "default deny" suggests that firmware that is not correctly signed or certified should not be allowed to run, and should, in turn, prevent the device it serves from booting.
2. **"Contextual Authentication" in Firmware:** The concept of "contextual authentication" has a unique application with regard to firmware. As the general description above states, "An authentication granted yesterday may not work today if the level of detected risks or vulnerabilities has changed." In a firmware-centric example, if a device attempts to connect

and it contains a firmware version that has recently been shown to be vulnerable or misconfigured, this authentication request should be denied. A recent example of this sort of firmware vulnerability can be found in CVE-2019-3707 where multiple vulnerabilities were discovered in firmware supporting Dell systems and their ability to use Dell's proprietary remote access capabilities. The practice of contextual authentication for devices suggests that a Dell device containing one of the affected firmware versions should be denied access to the network, even if it was allowed access the previous day.

3. **"Granular Control" as it Relates to Firmware:** It's no longer enough to simply acknowledge, "every device has firmware." In fact, every component of every device — from the ubiquitous Unified Extensible Firmware Interface found in nearly all computers to on-board memory and from networking components to video drivers — has its own firmware. Firmware instances are now nested within numerous integrated containers and it's not uncommon for endpoints and servers to arrive into service with firmware files that number in the dozens.

4. **"Dynamic and Real-Time" in the Context of Firmware:** Creation, deployment, maintenance, and replacement of hardware has evolved to be a continuous rather than periodic exercise, and is now done in real-time. Virtual servers are spun up and down by the thousands and in a fraction of a second. This includes their firmware. Every device in every household, and every adjacent device in a business network, is dynamic, possibly ephemeral, and likely to be constantly changing. This includes their millions of lines of associated firmware. Static or infrequent inventories cannot secure and assure the integrity of these devices.

Firmware and the devices they support are not only foundational to all compute and networking systems, but a key piece of creating and executing Zero Trust strategies.

## SECTION 4: FIRMWARE IN THE SUPPLY CHAIN

A significant portion of Section 4 of the Executive Order is spent in defining the requirement that "critical software" must be accompanied by a Software Bill of Materials (SBoM) to assure it maintains its intended integrity:

> "... maintaining accurate and up-to-date data, provenance (i.e., origin) of software code or components, and controls on internal and third-party software components, tools, and services present in software development processes, and performing audits and enforcement of these controls on a recurring basis;"

This has proven to be especially difficult with embedded firmware found in endpoints, servers, IoT devices, and network devices. It's a difficult task even in the seemingly well-protected systems supporting cloud services, as in the case of vulnerabilities in Supermicro server firmware that were disclosed in 2019.

The complexity of modern technology supply chains introduces numerous opportunities for risk: device OEMs depend on a network of component suppliers, who often source underlying components from other suppliers. In most cases, each of these devices has its own attendant and required piece of firmware. A compromise at any of these points in the supply chain can put the integrity of the device at risk. Vulnerabilities in any components might allow malicious actors to tamper with the device later in the supply chain either during the manufacturing process or at a Value Added Reseller (VAR), or during the firmware update process.

Provenance — the record of ownership of a work that serves as a guide to authenticity or quality -- is in many cases a stickier problem in firmware than it is with other kinds of software or with hardware. System components are often chosen based on price, and can be changed based on upstream deals and revised contracts. Their libraries and firmware details are often not visible to the enterprise security teams who, in the end, are responsible for securing the systems.

Despite these challenges the Executive Order asks cybersecurity teams to do two things today:

1. Impress upon their vendors the need for complete SBOMs
2. Enable their teams to leverage a publicly disclosed SBOM to verify their equipment throughout the acquisition and deployment processes

It is our stance that both of these actions need to include a complete assessment of firmware.

## SECTION 4: WHY ALL FIRMWARE IS "CRITICAL SOFTWARE"

The Executive Order focuses intently on the notion of "critical software."

> "The security and integrity of 'critical software'"— software that performs functions critical to trust (such as affording or requiring elevated system privileges or direct access to networking and computing resources) — is a particular concern."

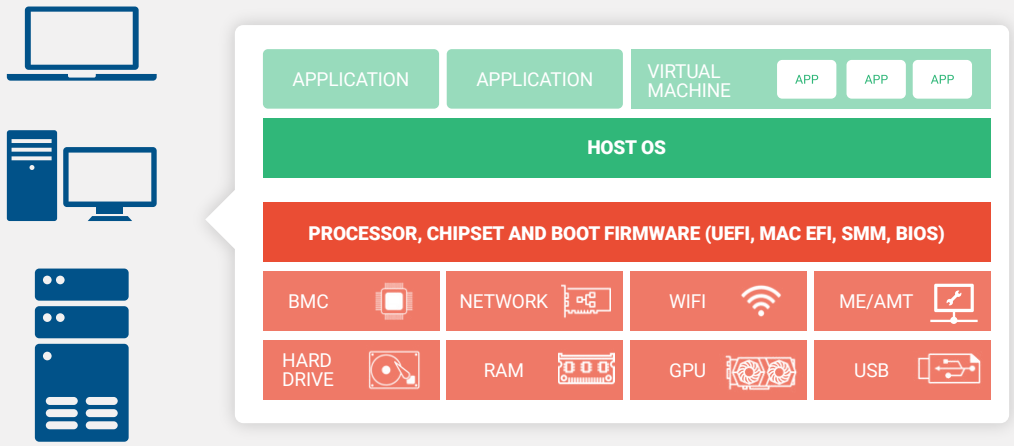
A review of the criteria in the Executive Order makes it clear that firmware not only meets the definition provided but is often overlooked and under audited. The requirements for critical software listed in the EO maps directly to the role firmware plays in modern infrastructure :

- *"The level of privilege required to function"*: Firmware occupies security layers below the operating system — the "sub-zero" security rings — and so often assumes ultimate privileges.
- *"Integration and dependencies with other software"*: Firmware is, by its very nature, the first functional building block upon which operating systems, applications, and services are both integrated and dependent.
- *"Direct access to networking and computing resources"*: Definitionally, firmware is by its very nature "on or in" nearly all networking and compute resources and so has the most direct kind of access.
- *"Performance of a function critical to trust"*: Some of the most critical components of our trust infrastructure, from trusted platform modules to boot processes of all kinds are entirely dependent on firmware.
- *"Potential for harm if compromised"*: There may be no more efficient (nor lasting) way to harm a system than through a firmware compromise. Malicious code attacks, LoJax and TrickBoot, and attacks on Baseboard Management Controllers (BMC) as we saw in the attacks against Supermicro servers, are possibly the most harmful attack that cyber defenders can experience. In addition, firmware attacks are highly persistent, and often last beyond wiping and restoring systems down to their bare metal configurations.

The amount of code, microcode and embedded firmware "below" the operating system is not only increasing, but more frequently targeted by attackers.

Ultimately, compromises of firmware can disable confidentiality of systems, obliterate Integrity, and interrupt availability of systems and services. In addition to the loss of data and destruction of the computing device, modified firmware can enable the transparent modification of data and subvert other security controls leading to a complete lack of trusted computation. With these points in mind it's difficult to argue against the notion that all firmware is, indeed, critical software.

Gartner puts an exclamation on this argument in their 2020 report "Roadmap for Improving Endpoint Security" when they say, "Firmware may well be the next endpoint battleground for advanced adversaries as script controls tighten."




*The amount of code, microcode and embedded firmware "below" the operating system is not only increasing, but more frequently targeted by attackers.*

## HOW *DEVICE INTEGRITY MANAGEMENT* ADDRESSES BOTH "ZERO TRUST" AND "CRITICAL SOFTWARE"

"Integrity" is a useful word that has both specific, narrow interpretations as well as interpretations that are evocative and thought-provoking. Integrity can mean:

- the quality or state of being complete or undivided: COMPLETENESS
- an unimpaired condition: SOUNDNESS
- firm adherence to a code of especially moral or artistic values: INCORRUPTIBILITY

"Device Integrity" is the practice of assuring the *completeness*, *soundness*, and *incorruptibility* of the device and its underlying firmware.

- **Completeness:** Is the firmware instructing the device complete and up-to-date? Has another version been introduced to patch or update the currently observed firmware version?
- **Soundness:** Are there any known vulnerabilities or active exploits against this version of the firmware? How are we prioritizing or mitigating these vulnerabilities?
- **Incorruptibility:** Device Integrity asks if the firmware has been configured and deployed in a way that makes it as secure as it can be for its current usage and situation.

The practice of Device Integrity answers these questions about every device on the network, whether endpoint, mobile, IoT, or network device. Because every device's behavior is controlled by the firmware that acts as its instruction-giving DNA, we apply these questions first and foremost to the underlying firmware, but where applicable we apply it to the bare hardware as well.

In a Zero Trust world, we will review each of these attributes — completeness, soundness, and incorruptibility — with every device that joins the network. A failed check on any of these will begin reducing whatever degree of integrity we've deemed sufficient to allow that device to join the network and do its job.

We would also start our review of this firmware with the assumption that it is, indeed, a piece of what we'd consider being "critical software". It is highly privileged, it is integrated with other parts of the software, it has direct access to the underlying hardware, it is critical to the notion of "trust," and the potential for harm, if compromised, can be quite high.

Device Integrity, and the questions it asks and the answers it provides, gives us the tools we need to secure and gain assurance of the integrity of the most hidden parts of any device, and by extension the device itself.

## DEVICE INTEGRITY REQUIRES REAL-TIME THREAT INSIGHT

Assuring a device's *completeness*, *soundness*, and *incorruptibility*, by means of deep assessment and evaluation of its firmware images or embedded microcode, is a significant step toward assuring the overall integrity of the device. But due to the hyperspeed, dynamic nature of new attacks, the completeness, soundness, and incorruptibility of any device and its firmware must be combined with active analysis of ongoing threats and exploits.

At the RSA Conference in 2021, Guardicore researchers Ofri Ziv and JJ Lehman demonstrated an inspired and unfortunately simple firmware-level implant. In the session "WarezTheRemote? Under the Couch, and LIstening to You," the two showed how unknown vulnerabilities in a piece of firmware that exists in virtually every modern house could be manipulated with privacy-shattering results.

Comcast's ubiquitous XR11 voice remote, which uses RF communication with the set-top box and over-the-air firmware upgrades, was compromised by a malicious firmware implant. The result? In minutes and with minimal expense a remote control that exists in almost 60 million US homes was hacked to continuously and remotely record audio without user interaction.

This simple firmware-level hack demonstrates how likely it is that a fast-moving flood of new device-level vulnerabilities will upend the threat landscape over the next 12 months. Enterprise security teams, however, may fairly ask "Why do I care if I don't have any Comcast remotes on hand?"

The answer may be in how this session reveals easy, inexpensive exploits against weaknesses in other enterprise-class networked devices, like conference phones, video equipment, and VPN systems to name a few. This session also demonstrated the degree to which our most common devices are unseen, untested, and unprotected.

## THE THREE PILLARS OF DEVICE INTEGRITY

It's tempting to ask why the problems of device integrity cannot be solved by existing vulnerability management, patch management or threat detection products. The simple answer is that they cannot because they were not built for it. These products were largely designed and deployed before the threats of firmware- and device-level compromise became real and widespread. Simply put, they're not made for the deep, firmware-level nature of this newest generation of threats and exploits.

An enterprise-class solution to the problem of device integrity will address these problems through an IDENTIFY, VERIFY and FORTIFY formula that works at speed and scale.

**Identify:** The solution would be able to discover devices, servers, and endpoints and throughout the enterprise and public networks. It would provide a complete view of the entire digital environment or focus on a specific group of devices, with insight into firmware and components that define an organization's security posture at all times. It would judiciously employ a mix of agent-based and agentless technology to discover newly introduced devices in real-time. This identification would include advanced threat detection to alert on threats such as hardware implants, backdoors, and other malicious code.

**Verify:** The solution would verify devices against a trusted baseline. It would compare this data against the industry's largest global firmware reputation database, checking firmware identities against millions of firmware hashes and dozens of enterprise hardware vendors to identify changes to baselines. It would find outdated firmware and expose tampering. It would provide insight into weaknesses and threats within devices, and also detect the risks associated with hardware profile changes, tampering, and compromise. It would use agent-based and agentless scanning to detect firmware vulnerabilities and missing protections in devices of all types.

**Fortify:** When misconfigurations, vulnerabilities, or threats are discovered, the solution would accelerate patching and update efforts, enabling staff to address weaknesses and save time. When threats are encountered, the platform would prevent damage by employing robust APIs that drive automated orchestration actions, such as applying updates or the quarantine of affected devices. In addition, device analysis and forensics capabilities would enable digital forensics to gather evidence to investigate the context of an attack, identify and limit the exposure of a breach, and add to a complete incident response playbook.

## WHAT YOU CAN DO TODAY

The *Executive Order* on Improving the Nation's Cybersecurity outlines a long but accessible road to a new level of security and resiliency. It also outlines a "1, 2, 3" path to act on these new requirements today that merges with a new understanding of the role of devices and firmware in securing our critical systems and data.

Among these steps:

1. Adopt Zero Trust strategies, tactics, and postures
2. For your Zero Trust program to successful, establish an active, expanded understanding of Device Integrity that focuses on firmware-level vulnerabilities, threats, and risks
3. To achieve Device Integrity across a large enterprise, deploy deep, firmware- and hardware-level discovery, evaluation, and remediation capabilities, in the formula of IDENTIFY, VERIFY and FORTIFY

The Eclypsium Platform is an enterprise-class solution that ensures an organization's critical devices are continually protected, including laptops, servers, switches, routers, and other systems. The platform supports a wide range of operating systems including Linux, Windows, macOS, Cisco IOS, and more.

Because every device has dozens of components that all rely on firmware and have their own unique vulnerabilities and threat models, Eclypsium delivers the same visibility and security across all components including system UEFI and BIOS, processors and chipsets, PCI devices, server BMCs, networking components, peripheral devices, Trusted Platform Modules, Intel's Management Engine and more. It integrates with tools such as Microsoft SCCM, Intune, or Tanium and manages access with popular SSO providers.

Users of major SIEM solutions like Splunk and QRadar can import and visualize device-level and event data and create dynamic analytics. The Eclypsium Platform also provides a rich set of REST APIs for integration into other existing security solutions.

CISOs, security and risk experts, and security architects who would like more information on firmware risk, device integrity, or the Eclypsium platform are encouraged to contact Eclypsium at info@eclypsium.com.