**Cyber Resilient Platforms**

adjective: **resilient**

… able to withstand or recover quickly from difficult conditions.

*Paul England, Ronald Aigner, Andrey Marochko, Dennis Mattoon, Rob Spiger, and Stefan Thom*

*Microsoft Corporation*

# Summary

This document is a high-level overview of The Cyber Resilient Platforms Program (CyReP) - a Microsoft-led industry initiative to improve the security and resiliency of computers, with particular emphasis on cloud-managed IoT devices. The CyReP Program includes hardware and protocol specifications, as well as open-source software that enables the security features.

One of the primary goals of CyReP is to enable a rich ecosystem of hardware and software components that can be used to build systems and devices that meet the requirements of NIST SP-800-193 (DRAFT) *“Platform Firmware Resiliency Guidelines.”*

# Introduction to CyReP

[NIST SP-800-153](https://csrc.nist.gov/publications/detail/sp/800-193/draft) (DRAFT) [1] identifies the following three principles for building resilient systems:

**Protection:** Mechanisms for ensuring that Platform Firmware code and critical data remain in a state of integrity and are protected from corruption.

**Detection**: Mechanisms for detecting when Platform Firmware code and critical data have been corrupted.

**Recovery**: Mechanisms for restoring Platform Firmware code and critical data to a state of integrity in the event that any such firmware code or critical data are detected to have been corrupted, or when forced to recover through an authorized mechanism.

Well-designed Internet-connected devices *protect* themselves against cyber-threats, and device vendors employ a wide range of hardware and software-based protection technologies to keep systems secure. Unfortunately, bugs and misconfigurations still lead to damaging exploits. A Cyber Resilient Platform contains additional mechanisms that allow exploits and vulnerabilities to be *detected*, and for devices to be *recovered* if they are compromised or hung.

Mechanisms for detection and recovery are already available for some classes of computer platform: for example, Baseboard Management Controllers (BMCs) and Service Processors (SPs) in conjunction with BIOS/UEFI firmware performs this function in centrally managed data centers and servers. Unfortunately, existing technology is ill-suited to IoT because of cost, power-demands, and the lack of an out-of-band control network.

The CyReP Program seeks to enable comparable manageability and security for the next generation of IoT devices. CyReP hardware building-blocks can serve as a foundation for building enhanced firmware and data protection, exploit/vulnerability detection, and reliable centrally-managed recovery into even the tiniest of devices.

CyReP hardware building-blocks can benefit any sort of system software. A simple MCU running a library OS may use CyReP hardware as the primary security technology. Devices that use a full-fledged operating system may use CyReP hardware to recover systems when all other cyber-defenses have failed.

Microsoft is working with industry to standardize CyReP hardware specifications: draft specifications include [Cyber-Resilient Platform Requirements](https://www.microsoft.com/en-us/research/publication/cyber-resilient-platform-requirements/) [2] and [Hardware Requirements for a Device Identifier Composition Engine](https://trustedcomputinggroup.org/work-groups/dice-architectures/) [3].

CyReP hardware is coupled with CyReP system-software to build end-to-end security solutions. Microsoft is open-sourcing portable libraries that can be incorporated into any system software, and is also open-sourcing ports to popular system software and devices.

A cornerstone of IoT device security is ongoing management, including firmware updates and security configuration changes. CyReP devices support secure and reliable centralized management through CyReP protocols. Microsoft is working to standardize protocols in the Trusted Computing Group (TCG). Microsoft is also providing open-source library code that implements the standards, which can be incorporated into both clients and servers.

Finally, Azure IoT supports highly scalable and reliable [management of CyReP devices](https://azure.microsoft.com/en-us/blog/azure-iot-supports-new-security-hardware-to-strengthen-iot-security/), and the next generation of Windows IoT can use CyReP features.

# CyReP Hardware Building-Blocks

CyReP hardware building-blocks allow device vendors to establish a small and well-protected *Root of Trust for Resiliency (RTRes – “are-tee-rez”)* for the device. The RTRes enjoys robust protection against malware – both when in persistent storage, and when running. A CyReP platform also provides mechanisms for the RTRes to be regularly scheduled, or invoked on demand by authorized controlling entities. The exact capabilities of the RTRes are determined by the device vendor, but secure recovery and update are expected to be its core functions.

The specific hardware resiliency building blocks requirements are specified in [2] and are sketched here.

**Storage Protection Latches**

A “Protection Latch” is an access guard that is (a) open at platform reset, (b) can be closed at any time by running software, and (c) can only be re-opened by a platform reset that passes control back to boot code. (Other names for protection latches are power-on write protection, and sticky bits.)

Storage write-protection latches are an extremely powerful building block for building resilient systems. Early boot code is typically much simpler than OS/application code, so if early boot code write-protects itself (and other critical programs and state) using a storage protection latch, then the attack surface of the stored device firmware becomes tiny. CyReP devices use write-protection latches to protect the critical management and resiliency functions and may use latches to protect the entirety of firmware. A platform reset (triggered by users, or by a CyReP watchdog timer) ensures that the RTRes can be invoked reliably when needed.

Storage read-protection latches allow device-vendors to provision devices with cryptographic keys that are only accessible to early boot code. Early boot code can derive additional keys that can be used by device firmware to authenticate the device and state. This is described in more detail in [4]. Read-protection latches are a low-cost alternative to dedicated crypto-processors and key-stores.

Many SoCs and some storage controller standards provide write-protection latches that meet CyReP requirements, and Microsoft is working to ensure ubiquity.

**A Secure Execution Environment for the Root of Trust for Resiliency**

CyReP devices mustensure that platform reset returns the device to a state in which early boot code can execute reliably (almost all platforms implement a reset function that meets these requirements.) A platform that *only* provides a protected environment for early boot code can only perform resiliency functions at boot time.

CyReP devices *may* provide a protected post-boot/run-time execution environment for the RTRes or an RTRes fragment to run. Devices that provide a run-time execution environment must also provide a reliable (non-maskable) interrupt mechanism that ensures that it runs periodically.

Essentially all larger microprocessors and some microcontrollers provide run-time isolation technologies that meet CyReP requirements. However, if the RTRes shares the execution environment with other complex functions (e.g. a Trusted Execution Environment (TEE) or an OS or hypervisor), then bugs in the other components may impair the reliability of the RTRes. For this reason, CyReP hardware vendors should provide dedicated resiliency functions, and CyReP firmware architects should only use these functions for the resiliency mission.

**Watchdog Timers**

Devices that are working properly – not hung, and not infected with malware – will reliably execute the management functions that their authorized management service instructs them to perform. However, if a device is hung or infected with malware, the device may simply ignore the request. CyReP devices include machinery to ensure that devices can be reliably managed, even when the device is hung or infected with malware.

Devices that allow easy user-servicing – for example, manual reset or power-cycling – can build highly resilient and reliably recoverable systems using just protection latches and a boot-time secure execution environment. Basically, a platform reset evicts malware, and the RTRes has a “safe place to stand” to perform repair or servicing.

Devices that require reliable *remote* management need a mechanism that an authorized cloud service can use to invoke the RTRes when needed. This requirement is challenging because few devices can afford a separate highly-protected out-of-band network connection (such as those employed by Service Processors), which implies the need for mechanisms that work reliably even when the firmware that manages the network stack is hung or compromised.

The approach to reliable remote management supported by CyReP devices is based on the idea of a cryptographically protected “execution lease” that is periodically granted by the management service. A device that uses an execution lease will continue performing its normal function as long as the authorized cloud controller periodically issues a *deferral ticket* that authorizes another quantum of execution (e.g. an hour, a day, a week). If a deferral ticket is *not* provided in time, then the platform will reset itself and pass control to the RTRes for recovery.

The mechanism that manages the execution lease is called an *Authenticated Watchdog Timer* (AWDT). AWDTs can be implemented in hardware or may be part of the RTRes executing in a protected execution environment. CyReP-compliant AWDTs are described in [5].

Devices that employ an AWDT “fail safe” because any sort of hang or compromise will result in control being transferred to the RTRes. Device vendors should decide whether this is appropriate behavior and what actions the RTRes should take when invoked. Device vendors should also balance false-positive (unwanted) device resets that occur because of network interruption with the maximum latency for devices to return to a safe state. Note that in most cases (non-compromised devices) non CyReP-based mechanisms can be relied upon to perform timely servicing.

The very simplest form of cyber-resilient Watchdog Timer prescribed by the CyReP hardware specification is a *Non-Deferable Watchdog Timer* (ND-WDT). Once set, a Non-Deferable Watchdog Timer will unconditionally reset the platform when the timeout expires. This behavior contrast with a conventional Watchdog Timer, whose reset can be deferred indefinitely (possibly by malware).

Because it is very simple, a Non-Deferable Watchdog Timer can be a suitable choice for the very simplest of MCUs, but to provide reliable RTRes-based management with an ND-WDT, occasional device resets are necessary. CyReP specifications and white papers suggest mitigations for service disruptions associated with occasional platform resets.

Finally, CyReP platforms must provide conventional watchdog timers to provide minimal latency recovery in the case of simple hangs.

A platform that meets the CyReP hardware requirements is termed a *Cyber Resilient Platform.* Depending on system design, the resiliency features may be implemented entirely in a SoC (System on Chip) or may be distributed across subsystems (e.g. storage controllers and custom logic).

## Using CyReP Hardware Building Blocks

A Cyber Resilient Platform is designed to provide a resilient *foundation* for an arbitrary Trusted Computing Base, rather than a full system-software solution. The actual Trusted Computing Base may be a very simple application package – for example in a sensor-style IoT device – or may be a full-fledged hypervisor running multiple operating systems and applications. The Trusted Computing Base will also typically employ additional runtime hardware-based protection technologies such as processor privilege levels to protect itself if they are available. For complex systems, CyReP capabilities are designed to supplement rather than replace existing protection technologies, and provide remediation if all other protections fail: *i.e.* if the TCB itself is compromised.

The CyReP hardware features (together or separately) can be utilized by standalone devices, but are most powerful when used in conjunction with a vendor or owner-operated cloud management service. Use of a centralized service allows devices to be managed at scale – for example, by providing a single point for the health of devices to be assessed and remediated when needed.

The resiliency features are designed to be both simple to implement in hardware, and simple for software to use. The simplicity increases the chance that systems built using these technologies will be resilient in the face of a determined cyber-attack.[[1]](#footnote-1)

Complete CyReP hardware requirements are documented in [2] [3] [5].

# CyReP Foundational Management Protocols

CyReP platforms provide a security and resiliency *foundation*, but are (largely) uninvolved in the run-time security of the device. This means that CyReP management protocols are a subset of a device’s likely management protocols. The scope of the CyReP Foundational Management protocols, and how they relate to higher-level protocols is illustrated in Figure 1.



Figure 1: Classification of management functions and protocols. Foundational Management Functions are the set of functions that are required to ensure that the TCB is up-to-date and operating properly. If the TCB is operating properly, then higher-level management functions can perform reliably.

Microsoft is standardizing Device Attestation and Device Identity protocols in the Trusted Computing Group. Microsoft’s proposal is [6]. This protocol is currently supported by Azure IoT’s Device Provisioning Service (the protocol is a certificate profile for TLS with client certificates.)

Microsoft also welcomes feedback on the Attention Trigger protocol implicit in the Authenticated Watchdog Timer specification [5].

Recovery is a combination of an Attention Trigger, and push/pull of a firmware update.

# The CyReP Root of Trust for Resiliency (RTRes)

A commonly-cited definition of the Trusted Computing Base is from the orange book: [7]

The heart of a trusted computer system is the Trusted Computing Base (TCB) which contains all of the elements of the system responsible for supporting the security policy and supporting the isolation of objects (code and data) on which the protection is based.

Unfortunately, for most IoT devices, this definition implies that the TCB is the entirety of the application and operating system. This is because the applications themselves are Internet accessible and often enjoy low-level access to device sensors and actuators. This means that a bug in the application is as bad as a bug in the TCB. Further, IoT devices are usually fixed function, and conventional TCB responsibilities, like separating users or enforcing integrity levels, are not used.[[2]](#footnote-2)

A practical consequence is that from a conventional system security perspective, the TCB is not a helpful concept, *except* that it can provide a foundation for managing (e.g. updating or halting) the application programs that it hosts.

The CyReP initiative builds on this observation, and extracts and elevates the parts of the TCB that are responsible for foundational management and recovery (including recovery of the TCB itself) into a set of capabilities we call the *Root of Trust for Resiliency*, or RTRes. Although a strict interpretation of the definition of the TCB would include the RTRes, it is convenient to define the RTRes as being foundational to, rather than a subset of, the TCB. See Figure 2.



Figure 2: Relation between the Root of Trust for Resiliency and the Trusted Computing Base

Vendors can choose the precise functions of the RTRes, but the following security functions will be common to a wide class of devices:

* RTRes and Firmware Stored Program Protection
  + The RTRes uses CyReP hardware capabilities to write-protect itself and other critical firmware and state (ideally including the remainder of the TCB and applications)
* Secure boot and other firmware integrity assessment functions
* Creating keys and certificates for identifying the device
  + Platform keys can be derived from keys fetched from read-protected storage, or other secure elements
* Cryptographically attesting the current firmware version or firmware identity
* Assessing whether the device is authorized and healthy, and contacting the management service for servicing instructions
* Secure and reliable update of firmware

A detailed description of the operation of an archetypal RTRes is presented in [7].

Microsoft will be open-sourcing portable libraries and boot code ported to popular devices that implement these functions.

## RTRes Architecture

Several RTRes functions need significant system software/library support: for example, a network stack is required to download a firmware patch.

The hosting environment for the Root of Trust for Resiliency for a PC/server-style device is the BIOS/UEFI-layer, as illustrated in Figure 3 (a). Unfortunately, an independent network stack for recovery adds hardware and ongoing maintenance costs, so this architecture is a poor choice for IoT devices.

An alternative architecture is to re-use the functions in the main device firmware, and employ CyReP hardware protections to enhance the security and resiliency of the combined TCB/RTRes functions. At first sight, this architecture seems a poor choice: the entirety of device firmware is likely to be vastly more complicated than the subset of the functionality required by the Root of Trust for Resiliency, so it is not immediately apparent that a firmware-integrated RTRes results in an architecture that is more robust than simple device-firmware-mediated update. Fortunately, there are two vulnerability mitigation strategies that can vastly improve the assurance level of the combined complex system.

The first mitigation is to protect the entirety of the TCB plus RTRes early in boot *during normal operation*. This means that if the complex system is compromised at run-time, malware cannot persist itself, and a platform reset will always return the system to a clean state.

The second mitigation is to define a boot-time-selected *“safe mode”* or *“RTRes mode”* for the firmware, which only runs the functions that are essential for recovery or other RTRes management functions. The reason why this improves resiliency is that OS bugs are only exploitable if attackers can reach them. The main threat to IoT devices is from network-based attacks, so a recovery mode that only runs applications and services necessary for the recovery mission, and only connects to cryptographically authenticated endpoints, presents a far smaller surface for attackers. Practically, we expect that a well-designed RTRes-mode can be just as safe as a dedicated recovery module (perhaps safer because the main firmware network stack will be better tested).

This is illustrated in Figure 3 (b) and is discussed more in [7].



Figure 3: Two approaches to implementing an RTRes. Devices that separate “firmware” (e.g. the BIOS) from “software” (e.g. the OS and applications) can implement RTRes functions in the firmware. An alternative approach is a combined system that implements a boot-time “safe mode” or “RTRes Mode” that only (i) loads/runs the minimum set of functions required for the resiliency mission, and (ii) strictly limits network connections to authorized entities like the management controller. Note that the RTRes components (and the OS components on which it depends) **must** be write-protected during normal operation for highest assurance.

# Conclusions

Many SoCs, storage controllers and platforms include dedicated security and resiliency features, but widespread adoption has been hindered by lack of ubiquity, lack of a coherent architecture employing the features, lack of open-source software enabling the features, and lack of protocol standards that align with the hardware security features.

The CyReP industry initiative is attempting to remedy this with a comprehensive set of standards and solutions that can be the foundation for the next generation of IoT devices.

# Bibliography

|  |  |
| --- | --- |
| [1] | A. Regenscheid, *Platform Firmware Resiliency Guidelines - SP 800-193 (DRAFT),* 2017. |
| [2] | P. England, R. Aigner, A. Marochko, D. Mattoon, R. Spiger and S. Thom, *Cyber-Resilient Platform Requirements,* 2017. |
| [3] | *Trusted Platform Architecture - Hardware Requirements for a Device Identifier Composition Engine (DRAFT).* |
| [4] | P. England, A. Marochko, D. Mattoon , R. Spiger, S. Thom and D. Wooten, *RIoT - A Foundation for Trust in the Internet of Things.* |
| [5] | P. England, R. Aigner, A. Marochko, D. Mattoon, R. Spiger and S. Thom, *Authenticated Watchdog Timers.* |
| [6] | P. England, R. Aigner, A. Marochko, D. Mattoon, R. Spiger, S. Thom, K. Kane and G. Zaverucha, *Device Identity with DICE and RIoT: Keys and Certificates,* 2017. |
| [7] | *Trusted Computer System Evaluation Criteria,* 1985. |
| [8] | P. England, R. Aigner, A. Marochko, D. Mattoon, R. Spiger and S. Thom, *Cyber Resilient Systems Software Architecture.* |

1. Vendors may add additional security and resiliency technologies to meet specialized requirements (for example, dedicated security or management processors.) [↑](#footnote-ref-1)
2. Vendors are encouraged to structure the Trusted Computing Base to enforce safety restrictions on the applications that it hosts, but this is outside the scope of the CyReP hardware functions. [↑](#footnote-ref-2)