Zero Trust in Embedded Systems

Introduction

The increasing prevalence of the Zero Trust (ZT) security model in enterprise networks has prompted significant security advancements in cloud computing, big data, and edge computing environments. Over the past decade, Zero Trust has evolved from a conceptual framework to an operational paradigm. However, despite its demonstrated benefits in securing enterprise infrastructures, a standardized approach to applying Zero Trust in embedded systems and IoT frameworks remains elusive.

This paper aims to evaluate the feasibility, benefits, and challenges of implementing a Zero Trust Architecture (ZTA) specifically tailored for embedded systems, with a focus on defense, aerospace, and industrial applications. In the enterprise realm, mature implementations such as Google's BeyondCorp, ZScaler, and CrowdStrike utilize and exemplify the effectiveness of Zero Trust principles and tenets. In contrast, embedded systems face unique constraints—limited computational resources, stringent power requirements, and isolated operational contexts—that render many enterprise-level security solutions impractical. While emerging strategies, including microsegmentation, attribute-based encryption, and AI-driven controls, offer promise, they often rely on resources (e.g., separate servers, resource-intensive certificates, edge clouds) that are not readily available in highly constrained environments.

This study is thus driven by several key research questions: How can traditional Zero Trust models be adapted to overcome the resource constraints of embedded systems? What trade-offs exist between enhanced security and performance in these environments? And what practical challenges emerge when implementing ZTA in critical sectors such as defense and aerospace? By addressing these questions, this work seeks to bridge the gap between mature enterprise implementations and the emerging needs of embedded and IoT systems.

Accordingly, the purpose of this study is threefold: first, to provide a comprehensive evaluation of current Zero Trust approaches as applied to embedded systems; second, to demonstrate the possibility of implementing a Zero Trust Architecture for embedded systems; and third, to identify the implementation challenges and potential solutions involved with practical application. A systematic methodology, incorporating comparative analysis and practical case evaluations, underpins this investigation. The study will assess various architectural approaches against key performance metrics—including computational overhead, energy consumption, and latency—to provide a balanced view of the inherent trade-offs. Detailed case studies will further illustrate real-world implementation challenges and potential solutions, ensuring that the analysis is grounded.

The significance of this research lies in its dual contribution to both academic discourse and industrial practice. By systematically evaluating current approaches and identifying viable strategies for adapting Zero Trust to embedded systems, the study fills a critical gap in the literature. Additionally, the findings offer actionable insights for practitioners operating in high-stakes sectors, where balancing robust security with operational efficiency is paramount.

It should also be noted that this research is conducted under the sponsorship of RTX, an American multinational aerospace and defense conglomerate renowned for its work in aircraft engines, avionics, aerostructures, cybersecurity solutions, guided missiles, air defense systems, satellites, and drones. Given RTX's substantial involvement in defense and aerospace, its core business areas and priorities—such as cost-cutting initiatives and a strong R&D focus—will be emphasized throughout this paper.

The remainder of this paper is organized as follows. "Background" provides an expanded context and review of the evolution of Zero Trust in both traditional IT and embedded contexts, as well as characteristics and security measures currently attributed to modern embedded systems. "Architectural Components of Zero Trust" delineates the fundamental architectural components of Zero Trust, as well as supplementary components that are significant in function. "Architectural Components of Zero Trust in Embedded Systems" explores how these components can be adapted to the constraints of embedded systems. "Comparison of Current Approaches to Embedded Security" presents a trade study comparing current approaches to embedded security and a proposed Zero Trust solution, while "Embedded Implementation Challenges" details case studies that exemplify the practical challenges in implementing ZTA in embedded environments. Finally, "Conclusion and Recommendations" concludes with recommendations for future research and development.

This paper aims to contribute to the ongoing discourse on Zero Trust in embedded systems, ultimately providing a clearer definition of the approach and paving the way for standardized, resource-efficient security solutions in the embedded/IoT domain.

Background

The Zero Trust security model represents a paradigm shift from a “trust but verify” mindset to a “never trust, always verify” mindset. Unlike traditional enterprise security models that implicitly trust internal components or subsystems, Zero Trust operates under the principle of removing implicit trust entirely, regardless of a subject’s network location or resource ownership. This means that every resource within the system is continuously monitored, and subjects trying to access resources are thoroughly analyzed before access is granted. This approach is particularly beneficial for systems with many services, endpoints, and devices, as traditional perimeter-based security models struggle to cover wide attack surfaces. By eliminating large implicit trust zones, the Zero Trust model delivers a secure and scalable trust decision framework.

As outlined in NIST Special Publication 800-207, the Zero Trust Architecture (ZTA) is based on several core principles:

- treating all data sources and computing services as ‘resources’;
- securing communication regardless of network location;
- granting resource access on a per-session basis with least privilege;
- determining access through dynamic policy using observable attributes;
- monitoring the integrity and security posture of all assets;
- enforcing dynamic authentication and authorization;
- and collecting comprehensive information about system state to improve security posture.

However, implementing Zero Trust in embedded systems presents unique challenges compared to enterprise networks. Embedded systems face several constraints that may affect the implementation of Zero Trust:

- Embedded systems have limited resources by nature. Memory, power and performance have to be limited to increase the mobility and cost-efficiency of the system. Because of this, embedded systems emphasize the importance of optimization and efficient resource allocation. This can bring up problems when implementing Zero Trust because it inherently requires a significant amount of resources due to the principle of continuous monitoring and verification.
- Legacy hardware is another concern. Due to the longevity of embedded systems they often don’t get updated in the field. They often rely on legacy components with decades-old firmware. Implementing Zero Trust will be difficult without the ability to update knowledge of modern threats or take advantage of better software. On the other hand, having the ability to update in the field introduces a whole new attack surface that is difficult to defend against.
- Many embedded systems, such as autonomous vehicles or drones, operate under strict real-time constraints where microsecond delays can cause catastrophic failures. Zero Trust mechanisms introduce latency through algorithmically calculating trust, as well as

the inherent latency caused by the request-response nature of the Zero Trust access sequence.
- Finally, embedded systems are often deployed in physically accessible environments, exposing them to physical attacks like tampering, side-channel attacks, and other threats not typically faced by enterprise systems.

Accordingly, there are security measures for embedded systems that are not used in traditional IT environments. Embedded systems typically have a stable set of applications, devices, and communication pathways, which can be leveraged to establish static configurations that define trusted interactions. For example, secure communication channels and trusted applications can be enforced at boot time, minimizing implicit trust zones.

Other security architectures in embedded systems include several protective measures including:

- Root of Trust (RoT), which are hardware component(s) that establish the foundation for secure operations though secure boot processes, trusted measurement, and cryptographic services. While RoT provides a baseline, implementations must be robust with hardware-based protections to resist targeted attacks.
- Trusted Platform Modules (TPMs), which are secure crypto-processors designed to perform cryptographic operations, store cryptographic keys, and verify platform integrity. Modern TPMs incorporate tamper-resistant features to protect against physical and logical attacks.
- Trusted Execution Environments (TEEs), which are isolated execution areas in processors where sensitive code can be run securely. While TEEs can potentially be vulnerable to side-channel attacks through power analysis, modern implementations incorporate countermeasures to minimize these risks.
- Systems hardening, which is the process of configuring systems to minimize vulnerabilities by disabling unnecessary services, removing unused components, and implementing strict security configurations.
- Redundancy, which is the implementation of multiple sensors and controllers to ensure reliability and maintain functionality if components fail or are compromised.

While Zero Trust presents a more robust security framework than the current approach, successful implementation requires balancing continuous verification and monitoring with operational efficiency and performance constraints. By adapting Zero Trust principles to the unique characteristics of embedded environments, organizations can strengthen product security while ensuring the continued functionality of critical systems.

- Characteristics of embedded systems (Romiah)
    - Resource constraints (CPU, memory, power)
    - Legacy hardware and software integration
    - Real-time performance requirements
    - Physical security requirements
    - Production to end-of-life security guarantee
- Current embedded security measures:
    - RoT
        - Secure Boot
        - Secure Storage
        - Measurement and Reporting
    - TPMs
    - TEE
    - Systems Hardening
    - Redundancy
    - Secure Communication
    - Access Control and Authentication
    - Monitoring and Intrusion Detection
- Sources:
    - https://learn.microsoft.com/en-us/security/zero-trust/zero-trust-overview
    - https://www.cisa.gov/zero-trust-maturity-model
    - https://www.microsoft.com/en-us/security/blog/2019/10/23/perimeter-based-network-defense-transform-zero-trust-model/
    - NIST SP 800-193: *Platform Firmware Resiliency Guidelines*
    - NIST SP 800-63: *Digital Identity Guidelines*
    - NIST SP 800-207: *Zero Trust Architecture*
    - NIST SP 800-160: *Engineering Trustworthy Secure Systems*
    - IEEE 802.1AR: *Secure Device Identity*
    - IEEE 802.1X: *Port-Based Network Access Control*
    - Google Cloud, "BeyondCorp: A New Approach to Enterprise Security," 2021.
    - https://csrc.nist.gov/glossary/term/roots_of_trust
    - https://www.rambus.com/blogs/hardware-root-of-trust/
    - https://cpl.thalesgroup.com/faq/hardware-security-modules/what-root-trust
    - https://trustedcomputinggroup.org/about/what-is-a-root-of-trust-rot/
    - https://www.dell.com/en-us/blog/hardware-root-trust/
    - https://support.apple.com/guide/security/gatekeeper-and-runtime-protection-sec5599b66df/web
    - https://en.wikipedia.org/wiki/Chain_of_trust
    - https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/protecting-data-at-rest.html
    - https://www.cloudflare.com/learning/security/glossary/data-at-rest/
    - https://www.windriver.com/solutions/learning/embedded-systems-security
    - https://scriptingxss.gitbook.io/embedded-appsec-best-practices/5identity_management

- https://docs.trustauthority.intel.com/main/articles/concept-trusted-boot.html
- https://library.mosse-institute.com/articles/2023/08/boot-security.html?highlight=boot
- https://learn.microsoft.com/en-us/windows/security/hardware-security/tpm/trusted-platform-module-overview
- https://www.intel.com/content/www/us/en/business/enterprise-computers/resources/trusted-platform-module.html
- https://learn.microsoft.com/en-us/azure/confidential-computing/trusted-execution-environment
- https://www.starlab.io/blog/the-criticality-of-trusted-execution-environments-in-linux-based-embedded-devices
- https://www.intel.com/content/www/us/en/business/enterprise-computers/resources/system-hardening.html
- https://www.beyondtrust.com/resources/glossary/systems-hardening
- https://www.splunk.com/en_us/blog/learn/redundancy-vs-resiliency.html
- https://cloud.google.com/docs/security/encryption-in-transit
- https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange
- https://www.cloudflare.com/learning/access-management/what-is-access-control/
- https://www.ibm.com/docs/en/wca/3.0.0?topic=security-authentication-versus-access-control
- https://doi.org/10.3390/app13137507
- https://thesai.org/Downloads/Volume14No11/Paper_54-A_Zero_Trust_Model_for_Intrusion_Detection.pdf
- https://xage.com/blog/mitre-attack-ics-zero-trust-architecture/
- https://www.preprints.org/manuscript/202403.0349/v1
- https://zeronetworks.com/files/brochures/MITRE-Attack-Prevention-Zero-Networks-Infosheet.pdf
- https://www.windriver.com/sites/default/files/2022-05/Whitepaper_A%20Survey%20of%20Information_Security%20Implementations%20for%20Embedded%20Systems_FINAL.pdf

Architectural Components of Zero Trust

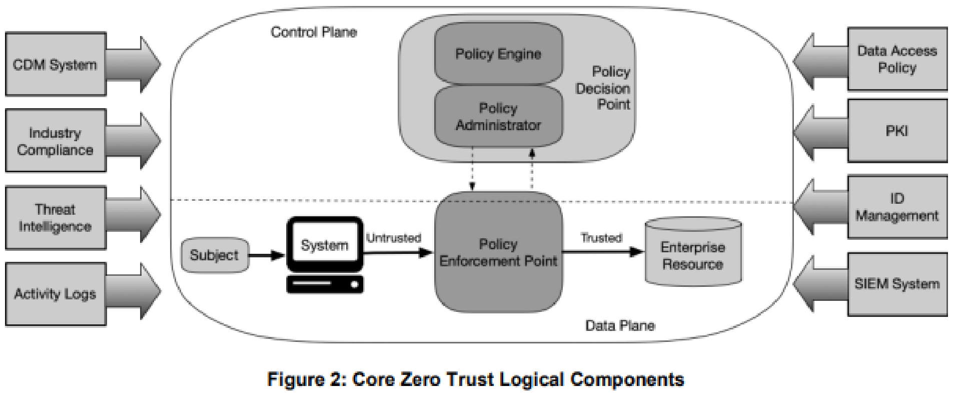


Figure 1: Zero Trust Architecture Core Components. Source: NIST SP 800-207

Core Components:

A Zero Trust Architecture (ZTA) consists of several fundamental components that work together to enforce strict access control and security measures within an organization. One of the most critical components is the Policy Engine (PE), a software system responsible for evaluating and enforcing policies that determine access to resources. The Policy Engine continuously analyzes inputs, applies predefined policies, and makes real-time decisions regarding access requests based on organizational rules and security requirements.

A specialized form of a Policy Engine is the Dynamic Authorization Policy Engine, also known as Attribute-Based Access Control (ABAC). Unlike traditional access control mechanisms, which rely on static rules, Dynamic Authorization evaluates policies in real time based on various attributes associated with users, resources, and environmental conditions. This approach allows for fine-grained access control, ensuring that access decisions are made dynamically based on contextual information, such as user roles, device health, location, and the sensitivity of the requested resource. By continuously assessing these attributes, the Dynamic Authorization Policy Engine ensures that security policies remain adaptive and responsive to changing circumstances.

Once the Policy Engine has made an access decision, the Policy Administrator (PA) is responsible for executing those decisions. The Policy Administrator communicates with the Policy Enforcement Point (PEP) by sending commands to either establish or terminate connections between users and resources. Additionally, the PA generates authentication tokens, authorization credentials, or session-specific access rights that users need to interact with enterprise resources securely.

The Policy Enforcement Point (PEP) acts as the frontline defense mechanism within a Zero Trust Architecture. It serves as a security gateway that enforces access decisions by either allowing or blocking communication between users and resources. The PEP continuously monitors network activity, ensuring that security policies are upheld and access is granted only to authenticated and authorized users. It also works closely with the Policy Administrator to receive policy updates and respond to changing security conditions.

Within a ZTA, the PEP can be implemented in different ways. It can be split into two components: a client-side agent that runs on user devices (such as a laptop or mobile device) and a resource-side gateway that protects critical assets by controlling access at the infrastructure level. Alternatively, a PEP can function as a centralized portal that acts as a gatekeeper for all communication paths, ensuring that every access request is thoroughly verified before being granted.

By integrating these core components, namely the Policy Engine, Policy Administrator, and Policy Enforcement Point, Zero Trust Architecture enforces a security model that eliminates implicit trust and continuously validates every access request. This approach significantly reduces the attack surface and strengthens an organization's overall cybersecurity posture.

Policy Sources Used by Policy Engine:

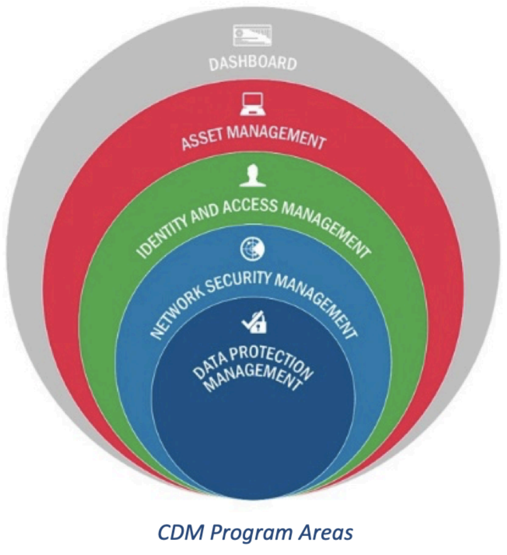

DASHBOARD
ASSET MANAGEMENT
IDENTITY AND ACCESS MANAGEMENT
NETWORK SECURITY MANAGEMENT
DATA PROTECTION MANAGEMENT
CDM Program Areas

Figure 2: Continuous Diagnostics and Mitigation System Network Security Areas of Interest. Source: https://bluecatnetworks.com/blog/what-bluecat-brings-to-cdm-phase-3/

Continuous Diagnostics and Mitigation (CDM) System:

The Continuous Diagnostics and Mitigation (CDM) system plays a pivotal role in supporting the Policy Engine by providing real-time insights into the security posture of an organization's network. CDM continuously monitors devices, applications, and users for vulnerabilities, misconfigurations, and non-compliance issues. This system collects data on various parameters, such as device health, software patch levels, and endpoint security configurations, which are critical for evaluating whether a device should be granted access to sensitive resources. For example, if a device is missing critical security updates or has an outdated antivirus program, the CDM flags it as non-compliant. This information is then used by the Policy Engine to make access decisions, ensuring that only devices meeting the organization’s security standards can access its resources. By integrating CDM data into the decision-making process, the Policy Engine can enforce adaptive access controls and mitigate risks in real time, strengthening the organization's overall security.

Industry Compliance System:
The Industry Compliance System ensures that organizational access policies align with applicable regulatory and industry-specific standards, such as Defense Federal Acquisition Regulation Supplement (DFARS)/National Institute of Standards and Technology Special Publication 800‑171 (NIST SP 800‑171), International Traffic in Arms Regulations (ITAR), National Industrial Security Program Operating Manual (NISPOM), and Cybersecurity Maturity Model Certification (CMMC). This is done by providing controls for handling Controlled Unclassified Information (CUI), export‑restricted technical data, and classified materials. It enforces ITAR by permitting only United States persons with a valid need‑to‑know and an approved export license to access technical drawings and specifications; applies DFARS/NIST SP 800‑171 controls by requiring that only workstations which have passed continuous diagnostics (such as full‑disk encryption, approved endpoint security agents, and multifactor authentication) can access CUI repositories; and implements NISPOM/CMMC clearance checks by granting “For Official Use Only” or higher access only to users with a facility clearance and a current Personnel Security Investigation (PSI). These compliance rules are encoded directly into the Policy Engine’s decision logic so that every access request is authenticated, authorized, and validated against the latest defense‑sector mandates. Because the Industry Compliance System also tracks regulatory updates—new CMMC maturity levels, revised ITAR commodity lists, and evolving DFARS clauses—the Policy Engine can dynamically adjust its policies in real time, maintaining full alignment with Department of Defense (DoD) audit and inspection criteria.

Threat Intelligence Feeds:
Threat Intelligence Feeds provide the Policy Engine with real-time information about emerging threats, vulnerabilities, and malicious activities occurring in the wider threat landscape. These feeds often include data such as known malicious IP addresses, phishing domains, zero-day vulnerabilities, and malware signatures. The Policy Engine leverages this information to dynamically adapt its access control decisions, blocking users or devices that exhibit behaviors or characteristics linked to known threats. For example, if a user attempts to access a resource from an IP address flagged in a threat intelligence feed, the Policy Engine can deny the request or trigger additional authentication measures. By incorporating external threat intelligence, the Policy Engine ensures that access policies are not static but responsive to evolving security risks, enhancing the organization’s ability to prevent cyberattacks and data breaches.

Network and System Activity Logs:
Network and system activity logs provide a detailed record of user behavior, system events, and network activity, serving as a critical input for the Policy Engine. These logs include information such as login attempts, file access patterns, failed authentication attempts, and data transfers. By analyzing these logs, the Policy Engine can detect anomalies or suspicious behaviors that

might indicate a potential security threat. For instance, if a user attempts to access a large number of sensitive files in a short period, the Policy Engine can flag this behavior as unusual and deny access until further investigation is conducted. These logs also provide historical context for access decisions, enabling the Policy Engine to make informed, data-driven evaluations. By continuously monitoring and integrating activity logs, the Policy Engine enforces access control policies that are not only based on static rules but also adaptive to real-time user and system behaviors.

Data Access Policies:
Data access policies form the foundation for determining who can access what resources, when, and under what conditions. These policies are typically established based on organizational rules, regulatory requirements, and specific business needs. For example, an organization might define a policy stating that only users in the finance department can access payroll data, and only during work hours from devices that meet specific security criteria. The Policy Engine evaluates every access request against these policies in real time, incorporating contextual factors such as the user's identity, device status, geographic location, and time of request. Data access policies can also be designed to incorporate dynamic risk assessments. For instance, access to highly sensitive resources might require multi-factor authentication if a user’s behavior or environmental conditions deviate from normal patterns. The Policy Engine enforces these granular policies by ensuring that each access decision is strictly tied to the defined rules, thereby minimizing the risk of unauthorized access or data leaks. Moreover, these policies are adaptable and can evolve as new security requirements emerge, ensuring that the organization maintains robust access controls even in dynamic environments.

Public Key Infrastructure (PKI):
Public Key Infrastructure (PKI) is essential for enabling secure and trustworthy communication and authentication within a Zero Trust Architecture. PKI operates through a system of digital certificates, public and private keys, and certificate authorities (CAs). The Policy Engine relies heavily on PKI to validate the identities of users, devices, and applications before granting access to resources. When a user or device attempts to authenticate, the Policy Engine uses the PKI to verify the digital certificate presented during the process, ensuring that it is valid, issued by a trusted CA, and has not been revoked or tampered with. PKI also facilitates end-to-end encryption, ensuring that sensitive data transmitted between the user and resources remains confidential and protected from interception. In a Zero Trust environment, PKI also supports mutual authentication, where both the user and the resource must verify each other’s identities before establishing a connection. This adds an additional layer of security by eliminating potential entry points for attackers posing as legitimate entities. By integrating PKI into the Policy Engine, organizations can enforce strong, cryptographic-based access controls and maintain a high level of trust in their authentication mechanisms.

Identity Management System (IDMS):

The Identity Management System (IDMS) is a centralized repository for managing user identities, roles, and permissions, and it is integral to the operations of the Policy Engine. The IDMS provides a single source of truth for user information, enabling the Policy Engine to verify and authenticate access requests with precision. When a user attempts to access a resource, the Policy Engine queries the IDMS to retrieve their credentials, associated roles, and any group memberships or specific permissions they hold. For example, if a user is part of a project team that requires access to specific files, the IDMS ensures that the user is granted access only to those files and nothing more. Beyond authentication, the IDMS also supports role-based access control (RBAC) and attribute-based access control (ABAC), allowing the Policy Engine to enforce highly granular access policies based on roles, attributes, or both. Additionally, the IDMS can integrate with multi-factor authentication (MFA) systems, ensuring that users undergo additional layers of verification before access is granted. By leveraging the IDMS, the Policy Engine ensures that access control decisions align with the principle of least privilege, reducing the attack surface and limiting the potential for insider threats or privilege escalation attacks.

Security Information and Event Management (SIEM) System:

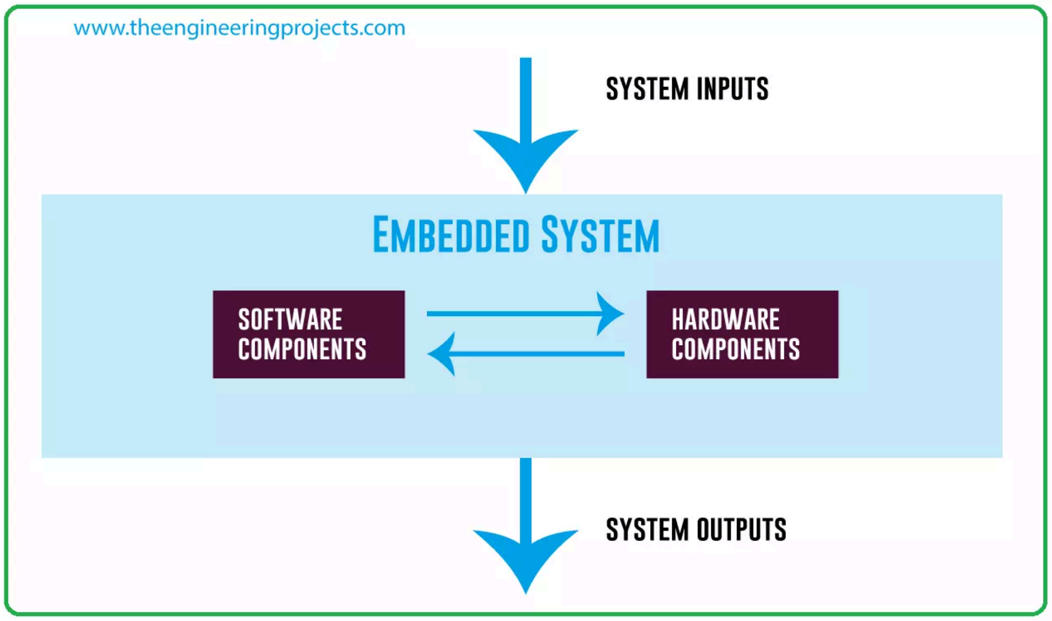
In a Zero Trust Architecture, a Security Information and Event Management (SIEM) system aggregates real-time telemetry (i.e. login attempts, file access patterns, network traffic, endpoint and firewall logs) from across the enterprise, feeding the Policy Engine with comprehensive situational awareness. Advanced analytics and predefined rules detect anomalies and correlate indicators, such as spikes in failed authentication or connections from flagged IP addresses, while integrated threat intelligence enriches detection of zero-day exploits and advanced persistent threats. Upon identifying risks, SIEM-driven alerts enable the Policy Engine to enforce adaptive controls—blocking or quarantining suspicious devices, requiring multi-factor authentication, or revoking credentials—to contain threats before they escalate. Finally, detailed incident reports and post-mortem analyses ensure regulatory compliance, support audits, and drive continuous refinement of detection algorithms, creating a resilient, self-improving cybersecurity framework.

- Sources:
    - https://www.nextlabs.com/products/cloudaz-policy-platform/policy-engine/
    - https://www.intersecinc.com/blogs/the-logical-components-of-zero-trust
    - https://www.netskope.com/netskope-one/zero-trust-engine
    - https://colortokens.com/blogs/nist-zero-trust-architecture/
    - https://www.cisa.gov/resources-tools/programs/continuous-diagnostics-and-mitigation-cdm-program
    - https://www.ibm.com/think/topics/compliance-management-system
    - https://www.spiceworks.com/tech/tech-general/articles/compliance-management-system/
    - https://www.crowdstrike.com/en-us/cybersecurity-101/threat-intelligence/threat-intelligence-feeds/
    - https://satoricyber.com/data-access-control/what-is-the-purpose-of-a-data-access-control-policy/
    - https://www.cyberark.com/resources/ebooks/pki-are-you-doing-it-wrong?utm_source=google&utm_medium=paid_search&utm_term=public_key_infra_nam_english_us&utm_content=eb_pki_are_you_doing_it_wrong&utm_campaign=securing_certificate_management_and_pki&cq_plac=&cq_net=g&cq_plt=gp&gad_source=1

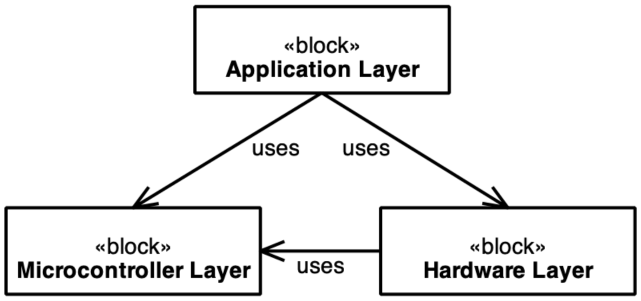
&gbraid=0AAAAAD_gt5GWhIzqHb62Fv-fU6BVkyUPu&gclid=CjwKCAjw7pO_BhAIEiwA4pMQvLu1IPGC8JEbT6QoS5Nji6OLMgHxSolf871hSwfrcPZLenA9WwUrehoCeNkQAvD_BwE
- https://csrc.nist.gov/glossary/term/identity_management_system
- https://www.ibm.com/think/topics/siem

Architectural Components of Zero Trust in Embedded Systems

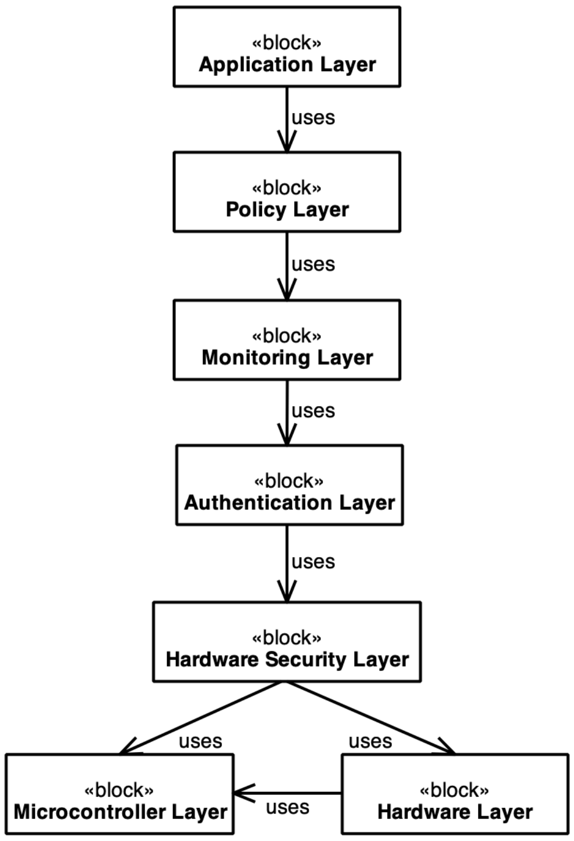
Before applying the previous Zero Trust components to embedded systems, it is essential to break down embedded systems to the logical level as well. As stated before, unlike general-purpose computers, embedded systems operate within constrained environments with limited computational resources.



The logical structure of an embedded system can be portrayed at varying levels of abstraction. The highest is a separation of hardware and software, as well as inputs and outputs as shown in the diagram above. In the next step, we will transition to a layered architecture model.



The hardware layer is then separated into two layers, the Microcontroller Layer and the Hardware Layer. The Microcontroller Layer consists of the microprocessor/microcontroller and its peripherals, as well as essential computing components like the RAM, ROM, power supply, timers, and other components with necessary functionalities. The Hardware Layer consists of the other peripherals that are used by the embedded system, such as actuators, sensors, displays, etc.



To implement the Zero Trust principles within embedded systems, ZT components must be carefully mapped to layers between the subject (Application) and resource (Microcontroller/Hardware). The approach in the diagram above involves integrating security measures at multiple layers while maintaining system performance and resource efficiency. The following subsections detail the key roles and implementations of each of the layers in the model above the bare metal of the embedded system.

Hardware Security Layer

The lowest level Hardware Security Layer forms the foundation of the Zero Trust model in our architecture by implementing security at the silicon level. This layer utilizes hardware-based security mechanisms such as memory protection units (MPUs), Trusted Platform Modules (TPMs), and hardware security domains to enforce isolation between different system components. Physical boundaries between critical and non-critical interfaces allows the system to enforce dynamic authentication and authorization without relying on slow software-based controls. Key technologies at this layer include:

- Hardware-enforced memory protection, preventing unauthorized code execution and data access, creating isolation zones that contain potential breaches. Memory Protection Unis (MPUs) or Memory Management Units (MMUs) enforce these boundaries by controlling which parts of memory can be accessed by which processes.
- Dedicated security processors or TPMs that provide secure key storage, cryptographic operations, and attestation services with minimal impact on the main processor. These lightweight modules can verify the integrity of firmware and software components.
- Isolated execution environments that protect sensitive operations from potential compromises in the main operating system. Technologies like ARM TrustZone create separate trust ‘levels’, allowing critical security functions to run in protected memory regions.

Authentication Layer

The Authentication Layer implements the cryptographic infrastructure necessary for securing communications and verifying identities. This layer leverages the hardware security features to establish trust. Some examples include:

- A PKI system that manages device identities and authentication credentials. For resource-constrained devices, lightweight algorithms like elliptic curve cryptography (ECC) offer strong security with smaller key sizes compared to traditional RSA.
- A chain-of-trust verification that ensures only authenticated firmware and software can execute on the device. Each component in the boot process verifies the signature of the next component before executing it.
- Session-based authentication, where the system grants temporary permissions for specific operations. These ephemeral credentials reduce the attack window and limit potential damage from compromised subjects.

Monitoring Layer

The Monitoring layer collects and analyzes system state and behavior to detect anomalies and security threats. In embedded systems, this layer must be especially efficient due to resource constraints.

- Behavioral monitoring must use resource-efficient algorithms to track normal device behavior patterns and flag deviations that might indicate compromise. By focusing on static configurations and significant anomalies rather than comprehensive logging, monitoring can be implemented with minimal overhead.
- Runtime attestation should verify that critical system components haven't been tampered with during operation. Periodic integrity checks should compare current system state against known baselines, with the rate being decided by surrounding context.
- Metrics like power consumption, memory usage, and CPU utilization can indicate potential security breaches, so they should be monitored with minimal additional hardware requirements.

Policy Layer

The Policy Layer implements the core Zero Trust decision-making functions, including the PE, PA, and PEP. For embedded systems, these components must be optimized for efficiency due to resource constraints.

- The PE should be simplified to use efficient algorithms optimized for constrained environments. Rules and policies can be pre-compiled and optimized for specific hardware to minimize computation overhead.
- Instead of centralized policy enforcement, security controls could be distributed across system components. This approach would reduce single points of failure and minimize communication overhead and latency.
- Access decisions should incorporate factors such as device state, resource sensitivity, and environmental conditions. Contextual factors can be weighted differently based on their security relevance and verification cost.

Application Layer

The Application Layer implements resource management and access control interfaces that ensure Zero Trust principles are maintained at the highest level of the system.

- Applications should communicate through controlled channels that utilize the lower layers to enforce access policies and encrypt sensitive data.
- The system allocates resources based on verified need, and applications should receive only the minimum resources required for their authorized functions (while considering critical functions, of course).
- When human interaction is involved, secure interfaces should prompt for appropriate authentication and communicate security status clearly.

By implementing these layers, embedded systems can achieve significantly enhanced security while maintaining necessary performance characteristics.

- Sources:
    - J. Smith et al., "Zero Trust for IoT: A Secure Approach for Smart Devices," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 2451-2463, 2023.
    - C. Wang, "Role of TPMs and PUFs in Zero Trust Architectures," *IEEE Transactions on Security & Privacy*, vol. 18, no. 2, pp. 34-45, 2022.
    - Gartner, "Zero Trust Security for IoT and Edge Computing: Trends and Best Practices," 2023.
    - Microsoft, "Zero Trust Security Model: A Guide for IoT and Embedded Systems," 2023.

Comparison of Current Approaches to Embedded Security

Embedded systems security has evolved significantly in recent years, with various approaches addressing different aspects of protection. Authentication methods have become more robust, with many systems now implementing multi-factor authentication (MFA) and strong cryptographic protocols like Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). These protocols ensure that only authorized entities can interact with the embedded system, significantly reducing the risk of unauthorized access.

Access control mechanisms have also seen improvements, with Role-Based Access Control (RBAC) becoming increasingly common in embedded systems. RBAC allows for fine-grained control over user permissions, ensuring that individuals only have access to the specific functions and data necessary for their roles. This principle of least privilege is further enhanced in Zero Trust architectures, which enforce strict access controls and constant verification for every user and device, regardless of their location within the network.

Network security for embedded systems has shifted towards a more comprehensive approach. Traditional models often assumed internal networks were safe, but modern strategies recognize that threats can exist both inside and outside the network. This has led to the adoption of secure communication protocols like MQTT and CoAP, which are specifically adapted for IoT environments and include features such as message encryption and device authentication. Additionally, the implementation of intrusion detection and prevention systems (IDPS) has become a best practice, allowing for active monitoring and response to potential security threats.

The trust model in embedded systems security has undergone a significant transformation with the introduction of Zero Trust principles. Unlike conventional security models that place faith in internal stakeholders, Zero Trust enables traffic to flow only among approved systems, regardless of the stakeholder. This approach involves continuous authentication throughout the communication phase, as opposed to just initial authentication in traditional models. While Zero Trust offers enhanced security, it can present challenges in resource-constrained embedded environments. However, the use of hardware security modules (HSMs), public key infrastructure (PKI), and Zero Trust Network Access (ZTNA) technologies can help overcome these limitations and significantly improve the security posture of embedded systems

Authentication mechanisms in embedded systems have advanced significantly, incorporating technologies such as Multi-Factor Authentication (MFA), biometric verification, and AI-driven identity validation. MFA requires users to verify their identity through multiple factors—such as passwords, hardware tokens, or biometrics—ensuring robust protection against unauthorized access. Biometric methods, including facial recognition and fingerprint scanning, enhance security while improving user experience. AI-based authentication systems further strengthen security by analyzing behavioral patterns and detecting anomalies in real-time. These innovations mitigate risks like credential theft and phishing attacks, making authentication a cornerstone of embedded security.

Access control has evolved with frameworks like Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC), which enforce the principle of least privilege. RBAC assigns permissions based on user roles, reducing administrative complexity while ensuring that individuals only access data necessary for their responsibilities. In Zero Trust architectures, access control is even stricter, requiring continuous verification of users and devices regardless of their network location. Cloud-based access control systems provide centralized management, enabling dynamic adjustments to permissions across distributed environments. These advancements ensure granular control over embedded systems while maintaining compliance with regulatory standards.

Modern network security for embedded systems emphasizes secure communication protocols such as TLS, DTLS, MQTT, and CoAP. These protocols encrypt data transmissions and authenticate devices to protect against eavesdropping and tampering. Intrusion Detection and Prevention Systems (IDPS) actively monitor network traffic to identify and respond to threats in real time. Additionally, Zero Trust Network Access (ZTNA) replaces traditional VPNs by granting identity-based access to specific applications rather than entire networks, reducing attack surfaces. These measures address the vulnerabilities inherent in interconnected embedded environments.

The trust model in embedded security has shifted from perimeter-based approaches to Zero Trust principles. Zero Trust enforces continuous authentication and assumes that no entity—internal or external—is inherently trustworthy. This model integrates features like micro-segmentation, secure boot mechanisms, runtime attestation, and hardware-based roots of trust (RoT) to ensure comprehensive protection. While Zero Trust enhances security significantly, it can be challenging to implement in resource-constrained embedded systems due to processing overheads and compatibility issues with legacy devices.

Embedded systems often operate under strict resource constraints, making performance impact a critical consideration. Security measures such as encryption and continuous monitoring can introduce latency or consume significant computational resources. Technologies like TinyML optimize AI-driven security processes for low-power devices, minimizing performance bottlenecks. Hardware-accelerated security features like Trusted Platform Modules and Secure Enclaves reduce the computational burden by offloading cryptographic operations to dedicated hardware components. Balancing robust security with efficient performance remains a key challenge in embedded systems.

The cost of implementing embedded security varies depending on the complexity of the system and the technologies used. Solutions like cloud-based access control can reduce upfront hardware investments while offering scalability. However, adopting advanced features such as Zero Trust architectures or post-quantum cryptography may require significant investment in hardware upgrades, firmware redesigns, and training for engineering teams. Compliance with regulations like the European Cyber Resilience Act further adds to implementation costs but ensures long-term resilience against emerging threats.

These criteria collectively highlight the trade-offs between security robustness, system performance, and financial feasibility in modern embedded systems. By carefully balancing these factors, organizations can develop secure yet efficient solutions tailored to their specific needs.

- Zero Trust for Embedded Systems vs. Current Security for Embedded Systems
- Sources:
    - D. Papp, Z. Ma and L. Buttyan, "Embedded systems security: Threats, vulnerabilities, and attack taxonomy," 2015
    - https://blackberry.qnx.com/en/ultimate-guides/embedded-system-security
    - https://surepassid.com/features/operating-systems/embedded
    - https://www.machinedesign.com/automation-iiot/article/21275738/master-embedded-systems-security-a-step-by-step-guide-to-protect-critical-components

Embedded Implementation Challenges

Implementing Zero Trust into embedded devices requires more upfront development cost, such as for monitoring, logging, and access controls. These additional software requirements also require thorough testing and complex integration. It also adds the need for a cryptographic key management system, requiring a system architect to decide how keys are provisioned, updated, and revoked. Identity based verification in ZT can complicate user access compared to traditional embedded systems.

Developing a well designed, intuitive, and consistent interface is important for preventing user frustration and wasted time. Adopting Zero Trust has the potential to significantly impact employee workflows, possibly leading to resistance to new methods and wasted time re-training. Additionally, the majority of cyber incidents ~80% are due to human error, which is very likely in the adoption phase.

The quantity and complexity of cyber attacks against companies has increased over time and will continue to rise. In response, Zero Trust systems may need to be continuously updated, increasing cost and complexity. Access controls and corresponding risk assessments may need to be adjusted every so often. Risk based access control relies on multiple data sources to determine user access.

Many embedded systems use software and hardware components from multiple vendors to reduce costs. However, third party vendors significantly complicate the process of securing the device. These components potentially have their own vulnerabilities that increase the attack surface. One approach is to build everything in house to maximize control over security, however this is expensive. Alternatively, security measures can be coordinated across all vendors to minimize vulnerabilities. Each third party vendor should be evaluated for their trustworthiness.

Another challenge is working with legacy systems. Many organizations, particularly large, established companies, rely on legacy devices and software that are not compatible with Zero Trust implementation. Many organizations operate a mix of on-premises systems, private clouds, and public clouds, each with distinct protocols and security mechanisms. Integrating these diverse components into a unified Zero Trust framework often requires extensive modifications or upgrades to ensure interoperability. This can be combated gradually by adopting new devices and software that are compatible with Zero Trust, middleware can be used to fill the gap between new and new old components.

These processes demand substantial processing power, which can strain existing IT infrastructures and lead to performance bottlenecks. Closely tied to this is latency, as the additional layers of scrutiny—such as micro-segmentation and real-time access validation—can increase response times for users accessing resources. This latency may disrupt workflows and reduce productivity unless mitigated through strategies like edge computing, optimized network architecture, and efficient encryption protocols.

Hybrid security policies involving embedded devices transmitting data to the cloud can strain computing resources and bandwidth, resulting in increasing latency. This latency can be detrimental in use cases that need real-time connectivity and fast decision making. Maintaining acceptable latency while minimizing resources is one of the key challenges in implementing ZT within embedded systems. The type of operating system and CPU architecture also impacts latency. For example, some CPUs, such as Intel processors, have built-in instructions designed for cryptographic functions which can reduce latency and CPU load in ZT environments.

Cryptographic algorithms designed for traditional computing environments may not be adequate for embedded devices, depending on how resource constrained they are. Though the algorithms may run, they might not provide the quick responsiveness necessary for such devices. Lightweight Cryptography may be used as an alternative, offering less memory usage, processing power, and power consumption. Different security protocols have trade-offs between resource usage and security strength. As attacks become more advanced, more advanced protocols will likely be needed. Future proofing embedded devices may require investing in hardware with greater computing capabilities than initially anticipated.

The cost and complexity of Zero Trust implementation are also significant barriers. Establishing this architecture involves substantial upfront investments in infrastructure upgrades, advanced security tools, and skilled personnel. The ongoing maintenance—such as continuous monitoring, policy updates, and threat response—further adds to operational expenses. Smaller organizations or those with limited budgets may find these costs prohibitive. Moreover, the transition to Zero Trust often disrupts existing workflows, requiring extensive employee training and process adjustments that can slow adoption.

Finally, physical security poses unique challenges in Zero Trust environments. Aging sensor technologies and unencrypted protocols in physical access control systems can be exploited by attackers through spoofing or tampering. Ensuring real-time verification of data from physical devices—such as biometric scanners or surveillance systems—is critical but requires advanced encryption and authentication mechanisms at every layer. These vulnerabilities highlight the need for a holistic approach that integrates cybersecurity with physical security measures.

Addressing these implementation issues requires careful planning, incremental deployment strategies, and leveraging modern technologies like edge computing and automation to balance security with performance and cost-efficiency.

- Sources:
    - A. Brown and T. Nguyen, "Lightweight Cryptography for Resource-Constrained IoT Devices," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 5, 2023.
    - https://www.dornerworks.com/blog/zero-trust/
    - https://sternumiot.com/iot-blog/4-embedded-security-challenges-and-how-to-solve-them/
    - https://www.riskandresiliencehub.com/overcoming-8-challenges-of-implementing-zero-trust/

Todo:
- Incorporate feedback from sponsor into this section
- Finalize edits for flow, grammar, and structure
- Remove repeated information
- Add supplemental information where context/detail is lacking

Case Study 1:

- Applying ZT to a drone

Zero Trust Architecture (ZTA) has emerged as a critical security paradigm for unmanned aerial vehicles (UAVs), addressing the unique challenges posed by their complex network structures and communication protocols. A recent study proposed an innovative approach to protect drone fleet networks against cyber threats by implementing a Zero Trust-based Network Intrusion Detection System (ZT-NIDS)[1]. This system, designed with a multi-agent architecture, employs a hybrid zero-trust detection mechanism to combat both known and emerging cyberthreats.

The ZT-NIDS approach considers drones as flying computers and applies traditional computer network security practices to drone networks. By adhering to the zero-trust principle, the system maintains a minimal level of trust when processing network traffic from drone fleets. The detection mechanism uses a network baseline as a reference to identify any deviations from normal behavior, enabling the system to detect both registered and unregistered attack attempts[1].

In a separate study, researchers developed a method for enhancing UAV security through ZTA by utilizing Radio Frequency (RF) signals within a Deep Learning framework[3]. This approach achieved an accuracy of 84.59% in detecting and identifying UAVs, which is crucial for determining network access in a Zero Trust Architecture. The integration of eXplainable Artificial Intelligence (XAI) tools, such as SHapley Additive exPlanations (SHAP) and Local Interpretable Model-agnostic Explanations (LIME), further improved the model's transparency and interpretability[3].

The implementation of ZTA in unmanned systems has shown promising results in addressing critical vulnerabilities, particularly in military applications. The U.S. Navy's Cybersecurity Program Office (PMW 130) has been at the forefront of implementing zero trust in unmanned systems, demonstrating its viability in exercises such as Trident Warrior 24[4]. This adoption ensures that every access request to the system, whether from an operator, sensor, or data feed, is authenticated, authorized, and encrypted, thereby enhancing the security of autonomous systems used for reconnaissance and combat operations.

While the transition to zero trust presents challenges, including the need for significant changes to existing IT infrastructure and ongoing management, the benefits are considered to outweigh the costs. As unmanned systems continue to evolve and their applications expand, ensuring their security through ZTA will be critical in protecting sensitive data, maintaining operational integrity, and fostering trust in these innovative technologies[4].

Citations:
[1]
https://thesai.org/Downloads/Volume14No11/Paper_54-A_Zero_Trust_Model_for_Intrusion_Detection.pdf
[2]
https://thesai.org/Publications/ViewPaper?Volume=14&Issue=11&Code=IJACSA&SerialNo=54
[3] https://arxiv.org/html/2403.17093v1

[4]
https://www.navy.mil/Press-Office/News-Stories/Article/4001119/the-us-navys-cybersecurity-program-office-pmw-130-leads-the-charge-in-implement/
[5] https://www.cisco.com/c/en/us/products/collateral/security/hitachi-ltd-case-study.html
[6] https://ieeexplore.ieee.org/iel8/6287639/10380310/10759645.pdf

Case Study 2:

This case study examines the implementation of Zero Trust Architecture (ZTA) in embedded and edge systems, focusing on its application in industrial IoT environments. The study highlights the challenges faced by a manufacturing plant and how ZTA addressed critical security concerns while enabling advanced IoT capabilities.

Background
A large manufacturing facility sought to modernize its operations by integrating IoT devices throughout its production lines. However, the plant faced significant security challenges due to the diverse range of devices, legacy systems, and the need for real-time data processing at the edge. The facility's management recognized the need for a robust security framework that could protect against both external and internal threats without compromising operational efficiency.

Challenge
The manufacturing plant's IoT network consisted of various sensors, controllers, and automated machinery, each presenting potential security vulnerabilities. Traditional perimeter-based security models were inadequate for protecting this complex ecosystem, especially given the increasing sophistication of cyber threats targeting industrial systems[1].

Key challenges included:
1. Securing a heterogeneous device landscape
2. Protecting sensitive operational technology (OT) environments
3. Ensuring continuous operation without security compromises
4. Meeting stringent regulatory compliance requirements

Solution: Zero Trust Implementation
The plant implemented a comprehensive Zero Trust Architecture tailored for embedded and edge systems. This approach was based on the principle of "never trust, always verify," applied to every device, user, and network interaction[3].

Key Components of the Zero Trust Solution:

1.Micro-segmentation : The network was divided into small, isolated zones to contain potential breaches and limit lateral movement[5].

2.Continuous Authentication and Authorization : Every device and user interaction was subject to strict verification processes, regardless of their location within the network[1].

3.Least Privilege Access : Access rights were granted on a need-to-know basis, minimizing the potential impact of any single compromised entity[3].

4.Real-time Monitoring and Analytics : Advanced monitoring tools were deployed to detect anomalies and potential threats across the entire IoT ecosystem[5].

5.Edge-based Security Processing : Security functions were distributed to edge devices to enable real-time threat detection and response without relying on cloud connectivity[2].

Implementation Process

1.Device Inventory and Assessment : A comprehensive audit of all connected devices was conducted, categorizing them based on their security capabilities and criticality[5].

2.Network Segmentation : The plant's network was restructured into logical segments, isolating critical OT systems from less secure IT networks[5].

3.Identity and Access Management (IAM) Integration : A robust IAM system was implemented to manage device and user identities, enforcing strict authentication protocols[3].

4.Security Policy Definition : Granular security policies were defined for each network segment, device type, and user role[1].

5.Continuous Monitoring Setup : Advanced monitoring tools were deployed across the network, with a focus on edge-based analytics for real-time threat detection[5].

Results and Benefits

The implementation of Zero Trust Architecture in the manufacturing plant's embedded and edge systems yielded significant improvements in security posture and operational efficiency:

1.Enhanced Security : The attack surface was dramatically reduced, with unauthorized access attempts dropping by 95%[5].

2.Improved Visibility : Real-time monitoring provided unprecedented insights into device behavior and network interactions, enabling proactive threat mitigation[3].

3.Operational Continuity : The micro-segmentation approach ensured that potential breaches remained contained, minimizing disruptions to critical operations[5].

4.Regulatory Compliance : The comprehensive security measures helped the plant meet and exceed industry regulations, including GDPR and ISO 27001[5].

5.Enablement of Advanced IoT Capabilities : With a secure foundation in place, the plant successfully implemented real-time asset tracking, predictive maintenance, and process optimization without compromising security[5].

Conclusion

This case study demonstrates the effectiveness of Zero Trust Architecture in securing embedded and edge systems within an industrial IoT context. By implementing a comprehensive ZTA approach, the manufacturing plant not only enhanced its security posture but also laid the groundwork for future IoT innovations. The success of this implementation underscores the importance of adopting Zero Trust principles in embedded systems, where traditional security models fall short in addressing the unique challenges of IoT environments[1][3][5].

Citations:
[1] https://www.dornerworks.com/blog/zero-trust/
[2] https://www.paloaltonetworks.com/cyberpedia/zero-trust-edge
[3]
https://www.edgenext.com/zero-trust-architecture-for-iot-devices-why-its-the-security-standard-of-the-future/
[4] https://www.accuknox.com/blog/mitigating-owasp-threats
[5] https://www.wittra.io/blog/zero-trust-fortifying-the-frontlines-of-iot-security/
[6]
https://bringyourownit.com/wp-content/uploads/2019/03/zero-trust-model-for-securing-embedded-systems.pdf
[7] https://sepiocyber.com/resources/case-studies/zero-trust/

Conclusion and Recommendations

Conclusion:

This study thoroughly evaluates the feasibility and strategic importance of implementing Zero Trust Architecture (ZTA) in embedded systems, notably in critical sectors such as defense, aerospace, and industrial applications. Traditionally applied in enterprise environments, Zero Trust eliminates implicit trust and requires continuous verification of all components. While effective in corporate networks, its adoption in embedded systems faces unique challenges due to strict resource constraints, reliance on legacy hardware, and real-time operational demands. This study seeks to adapt Zero Trust principles for embedded contexts, balancing robust security with the limitations of small-scale, high-dependability systems.

Applying ZTA to embedded systems leads to significant challenges due to limited CPU power, memory, and energy resources, which make traditional Zero Trust solutions too demanding. Real-time systems, such as drones or autonomous vehicles, cannot tolerate latency, and the long operational lifespans of embedded systems make them difficult to update regularly. Current embedded security measures, such as Secure Boot, TPMs, and TEEs, offer partial protections but lack the holistic, dynamic verification approach that Zero Trust provides. Therefore, integrating ZTA requires reengineering its core components (i.e. Policy Engine, Policy Administrator, and Policy Enforcement Point) to suit embedded environments.

Successfully implementing ZTA within embedded systems also requires addressing practical integration challenges, such as compatibility with existing operational protocols and ensuring minimal disruption during the transition period. Furthermore, the continuous evolution of cybersecurity threats necessitates ongoing refinement of Zero Trust strategies to maintain effectiveness over time. By systematically managing these integration hurdles and committing to continuous innovation and adaptation, organizations can leverage Zero Trust principles to achieve enhanced security resilience, safeguarding critical operations and infrastructures against sophisticated cyber threats.

Recommendations:

To effectively implement Zero Trust within embedded systems, organizations should develop lightweight ZT components optimized for embedded constraints and leveraging techniques such as attribute-based encryption and microsegmentation. Addressing legacy system integration challenges through modular and incremental upgrades, as well as implementing secure memory management practices, can ease the transition and enhance system security. Real-time performance can be preserved by pre-verification of critical operations or using parallel processing. In addition, long-term security can be strengthened with self-healing mechanisms and automated update systems, reducing the need for manual maintenance over extended lifespans.

Real-world case studies and industry collaboration, particularly in defense and aerospace, are crucial to ground theoretical models in practical applications. Standardizing ZTA guidelines and exploring future research directions, such as AI-driven policy enforcement and energy-efficient cryptography, are critical to scalable implementation. Addressing these strategic areas will empower organizations to strengthen their cybersecurity posture in an increasingly interconnected, resource-constrained, and high-risk operational landscape.

Additionally, investing in comprehensive workforce training and fostering organizational awareness about the unique aspects and importance of Zero Trust within embedded environments will further enhance successful adoption. Encouraging knowledge sharing and promoting best practices across teams can lead to more informed, collaborative, and resilient cybersecurity cultures. By building this internal expertise, organizations will be better equipped to proactively address emerging threats and continuously refine their security frameworks, ensuring long-term operational integrity and protection.

References