2024 MITRE eCTF Document

Team Pace

Secure Communication Protocol for MISC

Table of Contents

Introduction

The medical device prioritizes the integrity and confidentiality of sensitive patient data. To achieve this, a custom Secure Communication Protocol (SCP) has been developed and integrated into the system's architecture. This protocol governs the interactions between the Application Processor (AP) and its associated components (C1, C2), ensuring the following security safeguards:

- **Robust Device Authentication**: Before any communication takes place or the system boots, the SCP rigorously validates the identities of the AP and all connected components. This mechanism prevents the intrusion of unauthorized devices.
- **Encrypted Communication**: Confidentiality is paramount. The SCP employs strong cryptographic algorithms, such as AES-128, to encrypt all sensitive data exchanged between the AP and its components, rendering it unreadable to adversaries.
- **Message Integrity Verification**: The SCP incorporates Message Authentication Codes (MACs) or cryptographic hashes to guarantee the integrity of transmitted messages. Any attempts to modify or tamper with data in transit will be detected.
- **Protection Against Replay Attacks**: The protocol utilizes techniques such as nonces or timestamps to prevent adversaries from re-transmitting previously captured messages, ensuring the freshness of communications.
- **Secure Key Synthesis**: The SCP employs a secure key synthesis method known as Diffie-Hellman key exchange. This method allows the AP and each component to generate a shared secret key without ever needing to directly exchange the key itself. It leverages mathematical operations on publicly exchanged values to establish a shared secret key that can be used for secure communication. to facilitate the generation and distribution of cryptographic keys used for encryption and authentication purposes.

This document details the design and implementation of the Secure Communication Protocol, providing a comprehensive explanation of the mechanisms employed to protect the MISC medical device and the sensitive data it handles.In the realm of medical technology, the preservation of the security and privacy of sensitive patient data assumes paramount importance. Recognizing this critical exigency, the medical device has been meticulously crafted with an unwavering emphasis on safeguarding the integrity and confidentiality of patient information. To this end, a bespoke Secure Communication Protocol (SCP) has been meticulously developed and seamlessly incorporated into the system's architectural framework.

Functional Requirements

Component Listing

The Component Listing mechanism is a fundamental aspect of the system's functionality, enabling the Application Processor (AP) to identify and interact with the various medical components connected to it. These components could include devices like sensors, monitors or actuators. The listing contains unique IDs assigned to each component, allowing the AP to recognize and communicate with them effectively. While the listing information itself may not contain highly sensitive data, such as patient information, it is still important to secure it. This is where the Secure Component Platform (SCP) comes in. The SCP acts as a gatekeeper, verifying the legitimacy of the source providing the list and ensuring that only authorized components are listed. By doing so, it prevents unauthorized or unknown components from being part of the system, thereby reducing the risk of potential vulnerabilities or attacks.

Attestation

Attestation plays a crucial role in ensuring the integrity and authenticity of the connected components. Attestation data serves as a digital fingerprint for each component, containing information about its identity, configuration, and integrity status. When the AP needs to retrieve this sensitive information, it initiates a request and provides a valid Attestation Personal Identification Number (PIN). The SCP plays a vital role in this process by encrypting both the Attestation PIN and the attestation data during transmission. This encryption ensures that even if the data is intercepted, it remains confidential and unreadable to unauthorized parties. By requiring a valid PIN, the system ensures that only authorized users or processes can access this critical information. This helps in verifying the authenticity of components before they are allowed to interact with the system, reducing the risk of unauthorized or compromised components compromising the system's security.

Replacement

Component Replacement is a process designed for scenarios where a component needs to be replaced due to failure, maintenance, or upgrade. An administrator initiates this process by providing a valid Replacement Token to the system. The Replacement Token serves as a secure authorization mechanism for the replacement operation. The SCP plays a central role in facilitating this process securely. It ensures the secure transmission of the Replacement Token and oversees the Application Processor's (AP) validation process. Once the Replacement Token is validated, the AP updates its record of authorized components, effectively replacing the old component with the new one. This mechanism ensures that only approved and authenticated components are recognized and utilized by the system, maintaining its integrity and security.

Boot

The SCP manages a secure boot process for the system. During boot, the AP and expected components engage in a mutual authentication process to verify each other's identities. Once authenticated, the components send boot confirmations to the AP, which will only boot once it receives the expected messages. This multi-layer authentication ensures that the system starts in a secure state with only authorized components active, minimizing the risk of unauthorized access or malicious activity.

Secure Send and Receive

Once the system has successfully completed the boot process and all components are verified, the Secure Component Platform (SCP) establishes a secure communication channel between the Application Processor (AP) and the connected components. This secure channel employs robust encryption algorithms to protect all subsequent data transmissions. The encryption ensures that data exchanged between the AP and the components remains confidential and secure, preventing unauthorized interception or eavesdropping. Additionally, message integrity checks are performed on the transmitted data to detect any tampering or modification during transmission. This ensures the integrity and reliability of the communication, making sure that the data received by the AP is exactly as intended by the sender component.

Security Requirements

Component Validity Check

The Application Processor (AP) is required to authenticate connected components before allowing communication or initiating boot processes. This authentication ensures that only valid, expected components are permitted to operate within the system. There are several potential implementations to achieve this. One method involves using pre-shared keys or IDs, where each component and the AP share a unique identifier or secret key. The AP can then challenge a component with a request based on this shared secret, verifying the component's correct response. Another approach is to utilize digital certificates. In this method, components possess public/private key pairs and corresponding certificates signed by a trusted authority. The AP can then verify the authenticity of components by validating their certificates.

AP Validity Check

Components must authenticate the Application Processor (AP) before responding to commands or releasing sensitive data. To achieve this, there are a few potential implementations. One method is a pre-shared key or HMAC (Hash-based Message Authentication Code) approach, similar to the Component Validity Check but in reverse. Here AP and components share a pre-shared key or secret. Components authenticate the AP by challenging it with a request based on this shared key. Another method is using digital signatures. In this approach, the AP possesses a private key and uses it to sign messages or commands. Components have the AP's corresponding public key to verify the authenticity of the signatures.

Confidential PINs & Sensitive Data

Ensuring the confidentiality of Attestation Personal Identification Numbers (PINs), attestation data, and other sensitive information during transmission and potentially at rest (stored) is crucial. There are several potential implementations to achieve this requirement. One common method is symmetric encryption, where a robust algorithm like AES-128 (Advanced Encryption Standard with a 128-bit key) is used in a suitable mode (e.g., GCM - Galois/Counter Mode) for both encryption and authentication. Another consideration is key management, involving secure key generation using a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) and potentially a Key Derivation Function (KDF). Hardware-based secure storage may also be considered for storing sensitive keys.

Secure Post-Boot Messaging

After the system successfully boots, ensuring the authenticity and integrity of all exchanged messages is paramount. Several potential implementations can achieve this. One method is utilizing Message Authentication Codes (MACs), such as HMAC-SHA256 (Hash-based Message Authentication Code with Secure Hash Algorithm 256-bit). This allows the access point (AP) and its components to verify that messages have not been tampered with during transmission. Additionally, including timestamps or nonces within messages helps prevent replay attacks, where an attacker attempts to reuse a legitimately generated message to gain

unauthorized access. To elaborate, the HMAC-SHA256 algorithm combines a cryptographic hash function (SHA-256) with a secret key to generate a message authentication code. This code is unique to each message and serves as a fingerprint that can be used to verify the message's authenticity. Additionally, timestamps or nonces provide a measure of freshness, ensuring that messages are not being replayed from a previous session.

By implementing these security measures, the post-boot communication between the AP and its components remains secure and reliable. This is crucial for maintaining the overall integrity and functionality of the system, as it prevents unauthorized access, message tampering, and replay attacks.

Key Synthesis

In systems comprised of two components, the Application Processor (AP) initiates the crucial process of key synthesis. This intricate procedure begins with the AP generating a random number, a fundamental element in achieving cryptographic security. Leveraging this random number, the AP calculates a value, which is subsequently disseminated to each component within the system. Upon receiving the information from the AP, each component independently performs an XOR operation. This operation involves combining the received value with its own locally generated random number. The result of this XOR operation is an intermediate key unique to each component.

The next phase of the key synthesis process involves the exchange of these intermediate keys between the components. This exchange ensures that both components possess both intermediate keys. Once the exchanged keys are received, each component performs a final XOR operation with another distinct value. This additional layer of XOR strengthens the security of the key synthesis process. The result of this final XOR operation is then transmitted back to the AP. Upon receiving these results from both components, the AP combines them through yet another XOR operation. This final XOR operation yields the ultimate shared encryption key, which is consistent across all devices within the system.

In scenarios involving only one component, the AP assumes a different approach to computing key shares. Instead of initiating key synthesis through random number generation, the AP directly calculates key shares using predetermined values. The solitary component, armed with this information from the AP, executes its own computations to generate an intermediate key. This intermediate key is then transmitted back to the AP. The AP amalgamates the received value from the component with its own computed value through an XOR operation. The outcome of this XOR operation results in the final shared encryption key, which is used to safeguard sensitive data within the system.

This meticulously designed algorithm, underpinned by XOR operations and random numbers, provides a robust method for generating a secure shared encryption key between the AP and the components. Once generated, these keys become instrumental in encrypting and decrypting sensitive data within the system, thereby ensuring the integrity and confidentiality of information.

Secure Communication Protocol

Our secure communication protocol utilizes a two-way handshake communication process for interactions between the Application Processor (AP) and both components. This protocol ensures that the AP will respond to information correctly only if it has received encrypted and unexpired messages from the components. The protocol is designed to support three functionalities: Boot, Attest, and Post-boot communication. All information exchanged between the AP and the components is encrypted and decrypted using a unique key constructed into the board's firmware during the Key Synthesis process. This ensures that communication is secure and valid only when all three devices (one AP and two Components) are present and valid, determined by their ability to decrypt messages within a specified time frame.

Implementation:

Code Mappings

The key synthesis process in our system is managed within the key_exchange.c module. This module handles the generation of random nonces and the derivation of shared encryption keys. For message validation and error handling, the message_handling.c module is responsible. These modules abstract the underlying processes of key synthesis and message validation, ensuring secure communication between the Application Processor (AP) and components.

Cryptographic Libraries

Our system utilizes the WolfSSL cryptographic library for encryption and hashing functions. WolfSSL provides a range of robust cryptographic algorithms to secure sensitive data transmission. The specific algorithms used, such as AES-128 for encryption and HMAC-SHA256 for message authentication, are implemented within the cryptographic functions of our system.

Hardware-Specific Details

The system benefits from hardware-based security features, including a True Random Number Generator (TRNG) integrated into the board. This TRNG enhances the security of random number generation, crucial for cryptographic operations. Utilizing the TRNG ensures the reliability and integrity of key generation and encryption processes.

Code Structure

Our codebase is structured into modules to facilitate clarity and modularity. Each module has a specific role within the system:

- key_exchange.c: Manages the key synthesis process, including random nonce generation and shared key derivation.
- message_handling.c: Handles message validation and error handling, ensuring the integrity of data exchanged between the AP and components.
- cryptographic_functions.c: Contains cryptographic operations utilizing the WolfSSL library, such as AES encryption and HMAC-SHA256 calculation.
- board_specific.c: Interfaces with hardware-specific features, such as the TRNG for secure random number generation, enhancing the security of cryptographic processes.

These modules work together seamlessly to establish a secure communication protocol within our system. The main.c file orchestrates the overall system flow, utilizing functions from the various modules as needed. This modular approach promotes a clear separation of concerns and facilitates system maintenance and expansion

Deployment and Build Process

- This section outlines the steps to securely prepare and deploy devices utilizing the Secure Communication Protocol. Begin by ensuring you have the necessary hardware (the MAX78000FTHR_RevA development board or equivalent) and software tools including the appropriate SDK, a compatible C compiler, and a tool for flashing the firmware onto the microcontroller.
- Key generation and provisioning are critical security aspects. For this project, you will likely utilize the Python scripts (import.py, ap.py, and comp.py) provided within the eCTF materials. These scripts generate the required keys for the AP and components, incorporating Masks and Final Masks from a randomly generated CSV file. Securely store the generated key files. You might consider using a secure USB drive or an encrypted file system for distribution to the respective devices.

- To build the code, refer to the project's README or the Makefile for detailed compilation instructions. The build process should incorporate the generated key files into the source file within the project. Once compiled, utilize a compatible flashing tool to load the firmware image onto the AP and component boards. Any device-specific configuration, such as component IDs, should be set accordingly, ideally using a secure setup environment.

Security Analysis

The Secure Communication Protocol (SCP) prioritizes the protection of communication within the [Project Name] medical device. This analysis examines the protocol's resilience against potential attacks with a focus on preserving data confidentiality, integrity, and device authenticity.

- Eavesdropping Prevention: The SCP employs robust encryption algorithms (e.g., AES-128) to render intercepted communications unreadable to unauthorized parties. Proper key management, ensuring keys are not leaked or easily guessed, is crucial in maintaining confidentiality.
- Tampering and Integrity Protection: The protocol utilizes Message Authentication Codes (MACs), such as HMAC-SHA256, to guarantee message integrity. Any unauthorized modification to messages in transit will be detected, preventing the injection of malicious data or commands into the system.
- Replay Attack Mitigation: The SCP incorporates mechanisms such as timestamps, nonces, or sequence numbers within messages. This ensures that old messages cannot be re-used by an attacker to disrupt operations or perform unauthorized actions.
- Device Authentication: The meticulous authentication process between the AP and its authorized components establishes device legitimacy. This could involve pre-shared keys, digital signatures, or a combination of methods. Preventing unauthorized devices from joining the system is a key security measure.

Limitations and Assumptions

The SCP is designed under the following assumptions:

- Physical Security: The hardware devices themselves are assumed to be reasonably secure from physical tampering. If an attacker has physical access to extract keys directly, the protocol might be compromised.
- Secure Key Storage: It is assumed that the cryptographic keys are stored securely on the devices, potentially utilizing hardware security features if available.

Conclusion

The Secure Communication Protocol (SCP) serves as a revolutionary breakthrough in the realm of medical device communication security. By seamlessly integrating robust encryption, message authentication, and meticulous device authentication, the SCP elevates the protection of sensitive patient data to unprecedented levels. In doing so, it addresses the burgeoning need for data confidentiality and integrity, empowering healthcare professionals to confidently leverage the *MISC* system. This unwavering commitment to data security not only safeguards the privacy of patients but also bolsters the overall safety and well-being of those under medical care.

The SCP has been meticulously designed to withstand the ever-evolving threats posed by cybercriminals. It employs state-of-the-art cryptographic algorithms to encrypt patient data in transit, ensuring that even in the event of a data breach, sensitive information remains securely protected. Furthermore, the SCP incorporates message authentication mechanisms to guarantee that the integrity of messages is preserved during transmission, precluding any unauthorized alterations or tampering.

In addition, the SCP places paramount importance on device authentication, employing rigorous methods to verify the identities of medical devices seeking access to the *Your Project Name* system. This multi-layered approach to authentication ensures that only authorized devices can communicate with the system, mitigating the risk of unauthorized access and potential data breaches.

By adopting the SCP, healthcare professionals can rest assured that the privacy and security of their patients' data are accorded the utmost priority. The SCP represents a significant milestone in the healthcare industry, setting a new standard for data protection and empowering healthcare professionals to deliver exceptional care with confidence. As technology continues to advance, the SCP stands as a testament to the unwavering commitment to safeguarding the well-being of patients and upholding the highest ethical and security standards in medical device communication.