Colombe Academy of Technology

DESIGN DOCUMENT 2024-eCTF-CAT

This year's platform for the competition is Analog Devices' MAX78000 FTHR board, which features advanced functionality. The aim of the eCTF_2024 competition is to develop software to protect medical devices against supply chain attacks.

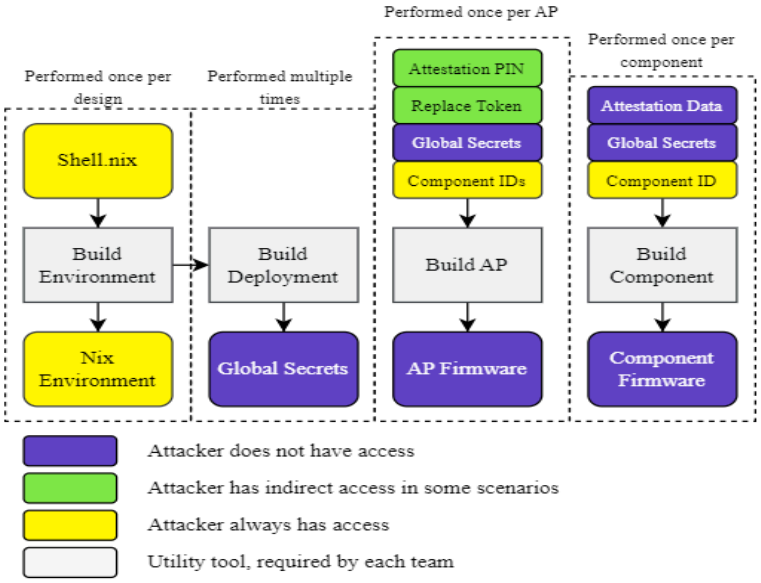
The Medical Infrastructure Supply Chain Challenge (MISC) consists of an application processor (AP), two components connected by an I2C bus, and a development board. Medical devices can be repaired in a trusted facility, where valid components from the secure production unit can replace faulty ones.

First of all, we would like to inform you about the use of the architecture proposed by the MITRE organization, of course with some of our inputs.

1. Platform and Architecture:

The Application Processor (AP) is the "brain" of the medical device. It manages all communications with the host computer, orchestrates the MISC functionalities to be implemented, and performs all the processing required to keep the device running during normal operation after start-up. As part of the MISC protocol, the access point is responsible for guaranteeing the integrity of the device

The Components represent the many additional chips that may be found on a medical device including sensors that take measurements and actuators that interact with the patient. The components rely on the AP to ensure the integrity of the device.



2. Security objectives:

The main security objectives are as follows:
Verification of the authenticity of the chips used.
Guarantee the ongoing security of devices in operation.
Prevent injection attacks, buffer overflows and other vulnerabilities.

3. Components and vulnerabilities:

3.1 Application Processor (AP):

- **Role:** Manage communications, orchestrate MISC functionality, ensure integrity.
- **Vulnerabilities:** poor token and pin management, Buffer overflows ,lack of input validation, poor error handling
- **Fixes:** Encrypt communication between AP and components , configure the global_secrets.h and crypt information about the Token and the Pin and attestation data using wolfssl. We add a delay of 5s when an attempt to replace or attestation failed once and with a bad pin or token.

3.2 Host Messaging Library:

- **Files:** inc/host_messaging.h, src/host_messaging.c
- **Role:** Format messages between AP and host tools.
- **Vulnerabilities:** Injection, buffer overflow.
- **Fixes:** replace gets' function by fgets() and fixed the exact size of variables

3.3 Board Link Library:

Files: inc/board_link.h, src/board_link.c
Role: Send arbitrary data messages between AP and components.
Vulnerabilities: unchecked packet length, unchecked I2C address, denial of service, I2C address spoofing.
Corrections : Added verifications, loop exit management.

3.4 Simple I2C Library:

- **Files:** inc/simple_i2c_controller.h, src/simple_i2c_controller.c, inc/simple_i2c_peripheral.h, src/simple_i2c_peripheral.c
- **Role:** Manages I2C communication with the MAX78000.
- **Vulnerabilities:** Unverified I2C address, missing error handling, uncontrolled access to registers, unvalidated inputs, reentrance attacks.
- **Fixes:** Added checks, error handling, limited access to registers,reentrance attacks.

3.5 Simple Flash Library:

- **Files:** inc/simple_flash.h, src/simple_flash.c
- **Role:** Interface with MAX78000 FTHR flash memory.
- **Vulnerabilities:** unvalidated inputs, insufficient error handling, unprevented buffer overflows, denial of service.
- **Corrections** : Input validation, improved error handling, overflow protection, transaction limiting.

4. Confidentiality and Integrity Solutions:

Within the realm of Medical Devices, our team has developed a secure method to ensure the storage and access of confidential information. To address Security Requirements 3 and 4, emphasizing the confidentiality of Attestation PINs, Replacement Tokens, and Component Attestation Data, we have designed two distinct scripts: pya.py for the Application Processor (AP) part and pyc.py for the Component part.

Functionalities:
Each script provides the following functionalities:
For pya.py (Application Processor) and pyc.py:

4.1 Function hash_PIN_TOKEN()

The hash_PIN_TOKEN() function improves the security of critical parameters (AP_PIN and AP_TOKEN) within the ectf_params.h file by employing SHA-1 hashing to obscure their values. It conducts a series of operations including content retrieval, component ID localization, hash computation for designated parameters, and subsequent replacement of the original file contents with the hashed values.
Regarding components.c, it serves to construct fresh content for the ectf_params.h file, integrating updated parameter values. Specifically, this process involves encrypting ATTESTATION_LOC, ATTESTATION_DATE, and ATTESTATION_CUSTOMER, thereby fortifying the security measures of the system.

4.2. Read File Content:

Reads the content of the file inc/ectf_params.h and stores it in a variable.

4.3. Replace Values with Encrypted Versions:
Initializes an AES cipher with a specified key and initialization vector.
Determines the positions of parameters within the content and encrypts their values with AES.
Replaces the original values with their encrypted versions in the content.

4.4. Create Modified File:
Creates a new file named modified_ectf_params.h to hold the modified content.

4.5. Update Parameter Values:
Replaces the original values of AP_PIN and AP_TOKEN within the content with their corresponding hashed values for the AP.
Encrypts ATTESTATION_LOC, ATTESTATION_DATE, and ATTESTATION_CUSTOMER for the component.

4.6. Write Modified Content:
Writes the updated content, containing hashed parameter values, to the modified_ectf_params.h file.

4.7. Replace Original File:
Replaces the original ectf_params.h file with the modified modified_ectf_params.h file.
Execution Flow:
The pya.py and pyc.py scripts are executed during the deployment construction. Before the deployment construction is complete, the scripts await user input and execute the appropriate functions based on the entered command.

Commands:
For pya.py script: Encrypt and save AP parameters.
For pyc.py script: Encrypt and save component parameters.
Note: The functionalities of moving the ectf_params.h files are integrated into these respective scripts for each part of the system.

Padding Functions:
Functions ensure message padding to meet encryption algorithm requirements.

Dependencies:
Utilizes 'wolfcrypt' library for encryption.
hashlib
Relies on 'subprocess' library for executing external commands.
Security Compliance:
Security Requirement 3:
Attestation PINs and Replacement Tokens are encrypted to maintain confidentiality. This prevents unauthorized extraction of secrets from devices, mitigating risks associated with unauthorized access and counterfeit part introduction.

Security Requirement 4:
Component Attestation Data is encrypted and only returned by the AP with the correct Attestation PIN. This ensures confidentiality of critical information and prevents unauthorized access, safeguarding against component recreation or modification and protecting proprietary data and patient safety.

note: the component would not be attest if it is not provisioned

5. Post Boot Secure Communication:

Post-Boot Functionality
The post-boot functionality is a crucial component of the system that ensures the security and integrity of operations performed after system boot-up. This functionality encompasses various aspects, including secure communication, component authentication, provisioned ID management.

some functions of post boot communication may be uncommented

***auth_component(const char* signature, size_t signature_size)**:
Description: This function authenticates a component using its signature.
Implementation: Loads the public key from "public.pem". Verifies the signature using SHA-256 with the wolfSSL library. Returns the verification result.

***gets_provisioned_ids (const char* signature, size_t signature_size)**:
Description: This function authenticates a component using its signature.
Implementation: Loads the public key from "public.pem". Verifies the signature using SHA-256 with the wolfSSL library. Returns the verification result.

***secure_send(uint8_t* buffer, uint8_t len)**:
Description: This function securely sends data over the I2C bus.
Implementation: Generates an encryption key. Encrypts the data using AES encryption. Sends the encrypted data over the I2C bus.

***secure_receive(uint8_t* buffer)**:
Description: This function securely receives data over the I2C bus.
Implementation: Receives the encrypted data over the I2C bus. Decrypts the data using AES encryption. Returns the decrypted data.

5.2. Key management

The attestation data is encrypted symmetrically, while its corresponding encryption key is asymmetrically encrypted. The encrypted information is not fully displayed, and the associated private key is kept in a secure location.

Security Requirement 2:

Components should only boot after being commanded to by a valid AP that has confirmed the integrity of the device.