Embedded Systems Cyber Security VIP
Final Paper for eCTF GT1(BuzzHack)

Lindsay Estrella
Georgia Institute of Technology
Atlanta, Georgia
lestrella7@gatech.edu

Shayan Aqeel
Georgia Institute of Technology
Atlanta, Georgia
saqeel3@gatech.edu

Jacob Devane
Georgia Institute of Technology
Atlanta, Georgia
jdevane7@gatech.edu

Tracy Guo
Georgia Institute of Technology
Atlanta, Georgia
tguo72@gatech.edu

John Zhang
Georgia Institute of Technology
Atlanta, Georgia
jzhang3213@gatech.edu

Adith Devakonda
Georgia Institute of Technology
Atlanta, Georgia
adevakonda3@gatech.edu

Scott Snow
University of North Georgia
Dahlonega, Georgia
srsnow3840@ung.edu

1. Introduction

The 2024 MITRE eCTF competition focuses on developing and attacking microcontroller medical devices. In this report the design decisions and implementations of the secure firmware are outlined. These solutions emphasize security measures like symmetric encryption and hashing algorithms. The build environment includes hosting necessary tools, packages, and dependencies using Nix and Poetry. The Medical Infrastructure Supply Chain (MISC) architecture consists of an Application Processor (AP) and two Components implemented through the MAX78000FTHR development boards.

The MISC functionality includes commands for listing Component IDs, obtaining Attestation Data, replacing failed Components, performing integrity checks, and enabling secure communications. The security requirements include verifying the boot process, authorizing boot-up sequences, maintaining confidentiality, and establishing secure communications post-boot. This report captures the work team members have done in the competition, along with a timeline of events and goals completed.

1.1. Build Environment and Setup

The build environment will host all of the required tools, packages, dependencies, and other development tools necessary for the device to run and operate. Nix will be the main way that these dependencies will be added and used in the design; Nix is a package manager tool that helps manage dependencies, allowing for a consistent and reproducible way of generating environments. The design environment is created by using a ”*nix-shell*” based on the requirements set forth in the ”*shell.nix*” and the ”*pyproject.toml*” files that were provided by the competition organizers.

The dependencies installed are *Make, Python3.9, gcc-arm-embedded, Poetry, Cacert, Minicom, Analog Devices MSDK, OpenOCD, PySerial, ArgParse, Loguru, and GDBUI.* After installing the required dependencies through Nix and Poetry the Application Processor and Components need to be generated, and updated onto the MAX78000FTHR boards to produce a global secrets file, which is needed to build the design.

2. Functionality

The functional requirements of the design focus on the two stages of building and utilizing the Medical Infrastructure Supply Chain (MISC). It is important to emphasize that the security requirements should not interfere with the functional requirements of the medical device.

2.1. Architecture

The architecture of the medical device consists of an Application Processor (AP) and two Components, all three of which are implemented through Analog Device MAX78000FTHR [1] development boards. The Application Processor is the center of the MISC system as it handles all communications with the host computer and uses the I2C bus to interact with the Components to perform the necessary tasks for the medical device. The Components are

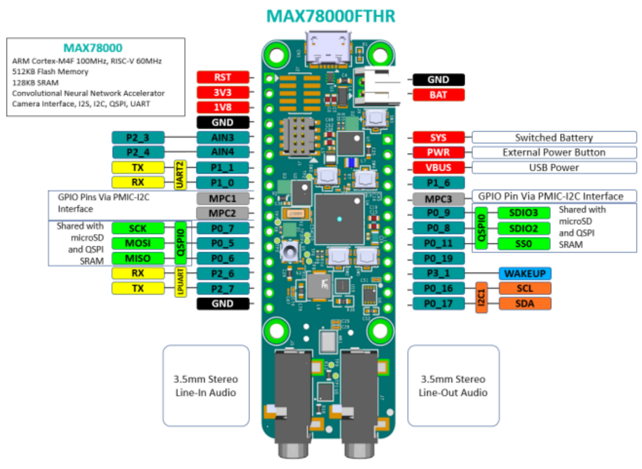
MAX78000FTHR
MAX78000
ARM Cortex-M4F 100MHz, RISC-V 60MHz
512KB Flash Memory
128KB SRAM
Convolutional Neural Network Accelerator
Camera Interface, I2S, I2C, QSPI, UART
RST
3V3
1V8
GND
P2_3
AIN3
P2_4
AIN4
TX
RX
UART2
P1_1
P1_0
GPIO Pins Via PMIC-I2C Interface
MPC1
MPC2
Shared with microSD and QSPI SRAM
SCK
MOSI
MISO
QSPI0
P0_7
P0_5
P0_6
RX
TX
LPUART
P2_6
P2_7
GND
GND
BAT
SYS
Switched Battery
PWR
External Power Button
VBUS
USB Power
P1_6
MPC3
GPIO Pin Via PMIC-I2C Interface
P0_9
P0_8
P0_11
QSPI0
SDIO3
SDIO2
SS0
Shared with microSD and QSPI SRAM
P0_19
P3_1
WAKEUP
P0_16
P0_17
I2C1
SCL
SDA
3.5mm Stereo Line-In Audio
3.5mm Stereo Line-Out Audio

Figure 1. Pinout Diagram

the elements of the device that provide useful functionalities through features such as sensors and actuators.

2.2. Building the Application Processor and Components

After properly setting up the build environment with Nix and Poetry, the Build Deployment Phase is executed. During this phase, the global secrets file is created in order to generate and maintain relevant key and other cryptography data for building the Application Processor and Components. From here, the Components for the Medical Device can be made. Each Component requires attestation data, a valid ID and information from the global secrets to be built. After creating the desired number of Components, one Application Processor for the device is created by providing the Component IDs for the device, a valid Attestation PIN, a valid Replace Token, and information from the global secrets to maintain security. If these steps are carried out correctly, the build phase for the MISC system can move on to creating the Medical Device.

2.3. Creating the Medical Device

After the firmware for AP and two Components are built, a Medical Device can be created. Using the update tool, the firmware is flashed onto the three MAX78000FTHR boards. The Medical Device is powered by the connecting a USB from the boot board to the host computer. This allows the device to take commands from the host to the AP.

2.4. MISC Functionality

There are five commands that the MISC is required to respond to. The first is the List Components command, where the MISC lists the Component IDs of the Components installed on the device. The next command is the Attest command, where the MISC allows authorized users with valid Attestation PINs to get Attestation Data from Components at the build process. Another command is the Replace command, where an authorized user with a valid Replacement Roken is allowed to replace a failed Component with a valid Component.

Additionally there is an important command being the Boot command. The boot command runs an integrity check on the Medical Device and the Components. If it passes the check, it prints a boot message and allows software to run the device. Otherwise, it terminates the boot process. Finally, the secure send and receive command allows the AP and Components to have secure communications with each other through a secure channel provided by MISC.

3. Security Requirements

This section will detail the needed requirements and how the team will achieve them for the final secure design. These requirements are put into place to protect critical data and code paths that may be used to gain unauthorized access to client information. The requirements to be listed are: the Application Processor (AP) must only initiate the boot process if all expected Components are present and valid.

Components on the Medical Device should only boot upon receiving a command from a valid AP that has confirmed the device's integrity. The Attestation PIN and Replacement Token must be kept confidential. Component Attestation Data must be kept confidential. Ensure the integrity and authenticity of messages sent and received using the post-boot MISC secure communications functionality.

3.1. Security Requirement 1: AP Boot Verification

This requirement is implemented in order to ensure that the Application Processor (AP) only boots if all expected Components are present and valid. This security check will be implemented by developing a comprehensive verification mechanism within the AP firmware itself to check the presence of all expected Components and their validity.

To first verify the connected Components, a routine in the AP firmware queries for unique signatures assigned to each Component to ensure proper identification. Then to ensure validity, a secret token in the global secrets file is hashed by each Component using SHA256 hashing and sent to the AP. The AP then hashes this token itself to ensure that each Component sent a matching hashed value. If all Components are present and valid through this method, the AP can successfully initialize a boot for the medical device. Otherwise, the boot fails until another request to boot is attempted, from which the verification and validation process repeats.

Additionally, these two processes will be developed in order to align with the Global Secrets generated during the build process. Should any of the above checks throw errors, robust systems for not only handling but logging these events will be developed. This system will work in tandem with the Component Boot Authorization detailed in security requirement 2 in order to authorize Component boot-up sequences.

3.2. Security Requirement 2: Component Boot Authorization

This requirement is put in place so the Component will only boot after being commanded to by the valid AP that has confirmed the devices integrity. This will be met by implementing a secure communication protocol between the AP and the Components. The use of encryption and authentication mechanisms will ensure that the valid AP can command the Components.

A system will be developed that authorizes the tokens generated by the AP for each single Component. These tokens should be required by both Components during the boot phase to ensure that the system is valid. On top of requiring the AP and Components to check for valid tokens the build process will need this implemented as well. These tokens could be associated with the Global Secrets to maintain a secure link between the AP and the Components. Lastly, integrating a secure communication channel between the AP and the Components using crytographic protocols to prevent unauthorized interception of commands will be implemented.

3.3. Security Requirement 3: Confidentiality of Attestation PIN and Replacement Token

This requirement is meant to keep the confidentiality of the Attestation PIN and the Replacement Tokens. With this approach the objective is to satisfy the requirement. First, a secure storage mechanism will need to be put in place by implementing the mechanism within the AP and Components to store the PINs and Tokens. An access policy will be used to control changing/accessing the PINs and Tokens. Only authorized processes should have access to these secrets. An encryption should be used on the PINs and Tokens during the storage and transmission of both. This can be done by using strong encryption algorithms and key management practices.

The team's design achieved confidentiality through hashing of each when they are passed in such that only the hash of each exists in memory and still be used to compare with hashed inputs to verify correctness of inputted PINs and tokens.

3.4. Security Requirement 4: Confidentiality of Component Attestation Data

The objective of this requirement is to keep Component Attestation Data confidential. The first step would be to implement a secure storage retrieval mechanism used to gather the Component Attestation Data. Using encryption and access controls to ensure only authorized processes can retrieve the sensitive information. Secondly, restricting access to the Attestation Data to a need-to-know basis for other processes by implementing a least privilege principle to decrease exposure.

Encryption of the I2C channel through RSA encryption and Cipher Block Chaining ensures confidentiality of the Attestation Data during transfer from the Component to the Application Processor.

3.5. Security Requirement 5: Secure Communications Post-Boot

The last requirement's objective is to ensure the integrity and authenticity of messages sent and received during the post-boot by using secure communication channels. Implementing a secure communication protocol between the AP and Components by utilizing encryption, message authentication codes (MACs), and secure key exchange mechanisms would solve this issue. Also implementing mechanisms such as checksums or cryptographic hashes, to verify messages during transmission and that will also reject corrupted messages. Lastly, creating a process for authenticating Components during the post-boot communication and using digital signatures or other authentication mechanisms to ensure that commands come from legitimate Components to check for authentic Components that have not been tampered with.

This requirement is fulfilled through the use of Cipher Block Chaining in conjunction with RSA encryption. RSA encryption adds additional security and is the first layer of encryption applied to messages exchanged on the I2C channel. RSA keys are exchanged during validation and used at every point after. Cipher Block Chaining XORs the most recently message with the outgoing message before encrypting it, which ensures integrity and authenticity of messages by requiring knowledge of the entire history of communications to correctly encrypt and decrypt messages.

4. Known Vulnerabilities

4.1. Application Processor

Within the Application Processor, MD5 hashing is used by the default hash method within the cryptography file. This immediately stands out as a poor design choice because of potential collisions with hashes. In order to remedy this, another hashing technique such as SHA256 could be used that would help to mitigate collision risk. Using WolfSSL, as recommended by Mitre, this is a straightforward replacement. The Application Processor also validates Components by ID, which is information that can be readily available to attackers. Instead, components can have unique keys generated alongside the IDs, which will be checked by the Application Processor for validity. The Attestation PIN and replacement token are also not encrypted, which could lead attackers to gain sensitive Attestation Data and to replace Components with false ones. Lastly, sending and receiving data is not encrypted, authenticated, or checked for corruption, which can lead to exploitation in the post boot phase.

4.2. Components

Components in the reference design are able to be booted without proper validation from the Application Processor, which could lead to false Components being injected into the medical device. This can lead to sensitive data being obtained and malicious actions being performed by attackers through communications with the Application Processor.

5. Individual Work

This section highlights what each individual has done during the competition. It consist of multiple detailed explanations of the different sub teams such as the development team, red team, documentary team, and etc.

5.1. Tracy Guo

Tracy Guo is a part of the red team, focusing on attacking other teams' secure design during the attack phase. She successfully found the design document flag on the MITRE website and debugged, configured, and updated hardware boards to attempt to obtain the boot flag and debugger flag. Guo has advanced her expertise with tools such as Nix shell, Kali Linux VM, Ghidra, and CyberChef, and has embarked on deepening her knowledge in red teaming. Tracy has taken initiative to overcome technical challenges, notably adapting to different software architectures by transitioning from OllyDBG to GDB on a Kali Linux machine due to compatibility issues. Preparing for the attack phase, she helped her red team devise an attack plan for each of the attack challenges as well as introducing John the Ripper to crack passwords. Her proficiency with this tool allowed her to contribute significantly to her understanding and exploitation of cryptographic weaknesses. Moreover, Tracy's introduction to GDB has further broadened her skill set, enabling her to debug and analyze programs more effectively. She quickly adapted to GDB's functionalities, using it to inspect the behavior of binaries.

5.2. Scott Snow

Similar to Tracy, Scott Snow is a part of the red team which focuses on attacking opposing teams secure designs during the attack phase. Scott is the team's documenter making sure to update and revise the design documentation based on the development team's efforts and changes made to the secure design by the development team. This document is submitted along with the secure design during the hand-off portion of the competition. Scott made the teams first advancements by finding the first flag along with the other flags in more recent challenges. Scott also helps create the red sub team's attack plan to tackle the second part of the MITRE competition, being the attack phase. He has also completed a lot of individual study when it comes to learning how to use reverse engineering tools like NSA's Ghidra and other tools like OllyDBG and GDB. He has added the resources that he has learned from in his GTRI VIP notebook for next years cohort to study and implement in their attack plan. He has also undertook learning about malware development as the attack phase's third and fourth attack scenario included the use of creating counterfeit parts to exfiltrate data from the opposing teams secure medical device. He shared the materials that he was learning from with the rest of his red sub team, along with mentioning and listing them in his VIP notebook.

5.3. John Zhang

John Zhang is a member of the development team, working through the reference design to understand what elements of the security requirements are missing or need modification. John has helped organize the team's meetings and approach as well as discover insight into the system. As a developer, he has integrated RSA encryption into the secure send and secure receive function, implemented generation and exchange of cryptography keys, and integrated CBC encryption from the modified simple file with the RSA encryption to ensure the confidentiality, integrity, and authenticity of messages exchanged between the Application Processor and its Components across the I2C channel. He has also added a check in the Component's code to prevent its booting without validation, encrypted the Attestation PIN and Replacement Token to ensure their confidentiality, and debugged the Application Processor.

5.4. Adith Devakonda

Adith Devakonda is a member of the development team which makes sure the secure design submission is compliant with the five security requirements. Until now, Adith has successfully set up the reference and insecure examples. He began work on the insecure example along with the rest of the team, identifying key areas of improvement in the Application Processor files. Some key work of note is: finding vulnerable structs that could carry extra data when booting and validating Components, and working on a rough cryptographic method that employs a CBC encryption mode as a tool for the developer team to use throughout the five security requirements. After completing system-wide encryption for the design, Adith helped the team debug and hopefully compile the secure design. He continued to help through all the roadblocks, meeting every Tuesday with the developer team and sometimes on Fridays in person at GTRI. One major roadblock was during compilation, when building with WolfSSL; multiple issues were encountered which ultimately led to the team not submitting the design and moving to attack phase. Adith learned collaborative development on an embedded system, which taught him the importance of effective communication, teamwork, and version control. It provided him with experience in navigating complex projects with multiple files that were all connected with each other. Something else Adith learned was attention to detail. The system the team used for the competition was larger than anything he had worked on previously, so lots of code review became a regular practice. This process not only ensured the quality and integrity of the team's codebase but also made it so that the design team had a mindset of continuous improvement. Each line of code was collaboratively worked on, enhancing Adith's understanding of the intricacies of software development and also enhancing his precision when debugging.

In the future, Adith plans to implement strategies to further his learning based on what he has learned throughout this class and competition. Some strategies he will use are:

1) Collaboration Techniques: Adith plans to explore more sophisticated version control workflows and enhance his communication skills.
2) Architecture and Design Patterns: Adith has already learned basic architectural code designs but he plans to learn specific ones tailored to embedded systems for better scalability and collaboration.
3) Automated Debugging: Adith plan to implement automated debugging via tests so that in the future, he does not hit a roadblock as the design team did with the WolfSSL issues.
4) Enhanced Debugging: Adith plans to learn debugging tools that can help him trace code back in a time efficient manner.
5) Continuous Learning: Adith plans to stay updated on new technologies and industry trends.

5.5. Lindsay Estrella

Along with Tracy and Scott, Lindsay is a member of the red team. After a late start with the team Lindsay worked to catch up with all group members and their work. After struggling with setting up the required environment Lindsay has gained more experience with virtual machines and many of the resources offered by Mitre. Due to the requirements of the secure design Lindsay along with the other red team members devised a plan on how to approach the Attack Phase. The team decided to continue their efforts in learning how to use debuggers such as OllyDbg and DBG and reverse engineering software like Ghidra. Lindsay in the process developed a proficient grasp on these tools. The team also concluded that malware development would come into use to aid in retrieving sensitive data or causing components to malfunction. This led to the Lindsay learning more about malware development but due to time constraints was not able to use these newly established skills in the Attack Phase.

5.6. Jacob Devane

Jacob Devane is also a member of the development team, focusing on building off of the insecure design in order to produce a secure system. Devane has helped to keep the development team organized with meetings and worked collaboratively to dissect the Application Processor code provided to by Mitre. Jacob has helped to improve the Application Processor's firmware by implementing SHA256 hashing for cryptographic uses. When the design was largely complete, Devane focused on code review for the other design team members. This process involved attempting to build both the Application Processor and the Component files, then reporting the build errors to the team and attempting to implement fixes. This also required researching new things like RSA and CBC encryption in order to understand the code the team had implemented.

5.7. Shayan Aqeel

Shayan Aqeel is a part of the development team, ensuring that the design submitted adheres to the five security requirements. Initially, Shayan has helped initialize the overall team's progress by booting the reference design and successfully utilizing the debugger, getting the debugger flag in the process. As a part of the development team, Shayan has found weaknesses in the system of validating and transferring data within the Application Processor where data encryption and validation keys could be implemented. From here, he implemented validation techniques for both the Application Processor and the Components by having both hash a secret message (using SHA256 hashing) and send it to the other which validates that the hash matches the actual hashed value. Lastly, Shayan implemented changes in the make file to incorporate these secrets in a safe manner into the shared global secrets file. Through this, security requirements 1 and 2 were fulfilled.

6. Conclusion

This research paper has covered in detail the design, implementation, and security requirements of the secure firmware, as it pertains to the 2024 MITRE eCTF competition. The paper has outlined the architecture of the MISC, the build environment, and the functionality, including the commands and security requirements put forth by the MITRE competition organizers. The build environment, was implemented with Nix shells and the Python Poetry library, ensuring consistent and reproducible development environments. The MISC architecture, comprised of the Application Processor (AP) and two Components, along with the process of generating and updating firmware onto the MAX78000FTHR boards. Functionality was outlined, including commands such as listing Component IDs, obtaining Attestation Data, replacing failed Components, performing integrity checks, and enabling secure communications. Security requirements were addressed, emphasizing AP boot verification, Component boot authorization, confidentiality of Attestation PINs and Replacement Tokens, confidentiality of Component Attestation Data, and secure communications post-boot. Vulnerabilities were identified and suggestions for improvements were provided, such as replacing MD5 hashing with SHA256, encrypting sensitive data, and implementing secure communication protocols. Individual contributions during the competition were highlighted, demonstrating the efforts in developing, red teaming, and documenting.

References

[1] MITRE Embedded Capture the Flag (eCTF) Website, https://ectfmitre.gitlab.io/ectf-website/index.html

[2] Maxim Integrated Products, MAX78000FTHR Technical Documentation Data-sheet, Pinout Diagram, https://www.analog.com/media/en/technical-documentation/data-sheets/MAX78000FTHR.pdf