eCTF 2024 Duke Design Document

Conducted By:
Duke University's Team

Team Members	Emails
Ana Stanisavljevic	ana.stanisavljevic@duke.edu
Isaac Martinez	isaac.martinez.espejo@duke.edu
Jack Steel	jack.steel@duke.edu
Oliver Wang	oliver.wang@duke.edu
Tyler Bletsch (mentor)	tyler.bletsch@duke.edu
Miroslav Pajic (mentor)	miroslav.pajic@duke.edu

Table of Contents

1 Report Overview

1.05 Disclaimer

Due to time constraints we were unable to implement the design we had planned in the beginning of the design phase. For the design we submitted, we created a secure send and a secure receive that encrypts the key when the application processor communicates with the component. The function encrypts the first 16 bytes of our secure send and then decrypts the bytes in the receive function. The following document consists of what we hope to complete in the near future to further secure our system.

1.1 Executive Summary

Our team of seven representing Duke University are competing in the 2024 MITRE eCTF embedded systems competition. This year's competition is focused on the implementation of a security solution for a given medical device. The goal is to ensure the confidentiality of device data while maintaining the integrity and authenticity of the device. This particular document outlines our plan for the design phase of the eCTF competition. Our design document details the implementation of the security and functional requirements of our embedded system security solution, as specified in the MITRE eCTF rulebook.

1.2 Scope of Defense

The AP has four pre-boot required functional elements that directly relate to the host tools: list components, boot board, replace component, and attestation. All of the functional requirements must align with the timing requirements. The AP's post-boot functionality requirements no longer need to be activated while the secure_send and secure_receive functions must be active. The boot board functionality must handle booting all provisioned components and the AP. After printing boot messages, the AP and Components should boot their respective main functionality. The replace component function must handle replacing a provisioned component with a new component. The Component firmware can communicate with the AP through all of the pre-boot required functional elements except for the Replace Component. We seek to implement all of the above in a manner that preserves all security requirements: confidentiality, integrity, and authenticity of the attestation PIN/data, replacement token, global secrets, boot process, and communications.

2 Competition Requirements

2.1 Security Requirements

The following sections review the security requirements as specified in the 2024 MITRE eCTF competition. We seek to make our embedded system design as impenetrable as possible, to avoid leaks during the attack phase of the competition. Our approach to the following security requirements, as drafted below, will work to avoid enemy penetration.

2.11 Security Requirement One

The Application Processor (AP) should only boot if all expected Components are present and valid.

In order to pass security check number one, we need the AP to verify that the components are valid as expected. We will do this by having the attestation data, which is present in each component, be signed by the vendor- the public key for which will be put into the firmware by the vendor. The AP can request this data using the existing attestation process.

2.12 Security Requirement Two

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

This requirement is needed to ensure mutual integrity confirmation between the AP and component. Currently, we are planning on using the attestation pin to confirm that the AP is valid. However, it is important to note that this will likely not be our final solution to this requirement, as in some cases the attacker has indirect access to the attestation PIN. Ideally, we need to ensure that the piece of data used for AP validation is not ever given to the attacker.

2.13 Security Requirement Three

The Attestation PIN and Replacement Token should be kept confidential.

To meet this requirement, we plan to encrypt our attestation pin and replacement token using a shared global secret key. This will serve to slow down our attackers and prevent the secrets from being simply extracted, but we plan to add on to this strategy later using a level of hash chaining or possibly implementing Salted Challenge Response Authentication Mechanism (SCRAM) for pin exchange.

2.14 Security Requirement Four

Component Attestation Data should be kept confidential.

In order to keep the attestation data confidential, it will be encrypted symmetrically with a key derived from the attestation pin. That derivation of the attestation pin will occur from chain hashing, which will be done in an amount that takes just under five seconds, so that the maximum wait time is implemented for attackers in both online and offline attacks.

2.15 Security Requirement Five

The integrity and authenticity of messages sent and received using the post-boot MISC secure communications functionality should be ensured.

AEAD, or Authenticated Encryption with Associated Data, is the process of using keys to symmetrically encrypt and decrypt messages being sent from one individual or system to another. The encryption of these messages using secret keys ensures confidentiality of what is being sent. As well, verifying that the sender has access to the shared secret key ensures the authenticity and integrity of the messages being sent and received using the post-boot MISC secure communications functionality. In our embedded system, we intend on encrypting our message into ciphertext using a shared secret key – which is stored in global secrets that the attacker does not have access to – alongside providing associated data, or AD. When received by the receiver, the associated data used to encrypt the message is verified by comparing it to the associated data used to decrypt the message. If this data matches, the ciphertext can be decrypted using the shared secret key. This converts the message to text that can be read by the receiver. This plan of implementing AEAD preserves both the confidentiality of the message, as only the sender and receiver can see the content of the message, along with the authenticity and integrity of the message, as the sender and receiver can verify the source of the message.

2.2 Functional Requirements

The following sections review the security requirements as specified in the 2024 MITRE eCTF competition. We seek to make our embedded system design as impenetrable as possible, to avoid leaks during the attack phase of the competition. Our approach to the following security requirements, as drafted below, will work to avoid enemy penetration.

2.21 Functional Requirement One

List Components

The MISC must be able to list the Component IDs of the Components currently installed on the Medical Device. This command is not authenticated and may be initiated by anyone, as the Component IDs are not sensitive data.

Nothing too extravagant will be done here- since the AP knows the component IDs we will simply scan to see if they are present and then print them out.

2.22 Functional Requirement Two

Attest

To debug and validate the integrity of the Components, the MISC must allow an authorized user to retrieve the Attestation Data that was stored on the Components during the build process. This should only occur if the user is able to provide a valid Attestation PIN (see Security Requirements)

The attestation data will be encrypted symmetrically with a key derived from the attestation PIN. The key derivation process is done by using chain hashing which will take just under 5 seconds. If the user does not have the key, then they can not read the attestation data even if they pass through the wall created by the PIN validation step. Chain hashing can be used to increase the computational effort required to crack the passwords. Instead of hashing the PIN once, the hashing process is repeated multiple times. This makes offline brute-force attacks more difficult and time-consuming because an attacker must apply the hash function many times for each PIN guess.

2.23 Functional Requirement Three

Replace

In the case that one of the components fails, the MISC should allow an authorized user to replace it with a new, valid component. This should only occur if the user is able to provide a valid replacement token (see Security Requirements).

To allow authorized users to replace components, they must provide a valid replacement token. We will do this by noting every component switch and updating any data structure involved to the AP. During this removal, we will mutually validate AP and the old component in a manner similar to boot validation, discussed previously.

2.24 Functional Requirement Four

Boot

One of the most important functionalities of the MISC is to boot the Medical Device. During this process, the MISC must first ensure the integrity of the device and the Components on it (see Security Requirements). If this integrity check fails, the boot process will be aborted. Otherwise, the MISC should print a boot message and hand off control of the AP and Controllers to the target software that will then run the Medical Device.

We will use a similar implementation to the validation step, using attestation. When the handshake between the AP and the components is complete, and both have validated each other, only then will the boot occur.

2.25 Functional Requirement Five

Secure Send & Receive

After a successful boot, the MISC must provide a secure communications channel for the AP and Components to use. This channel will allow the AP and Components to securely send and receive messages (see Security Requirements).

By using AEAD and a global shared secret key, communications channels will be secure. This will be like the secure communication we use for boot, except it will be applied to the send and receive calls.

3 Conclusion

Going forward, our strategy is to begin securing our system and ensuring its confidentiality, integrity, authenticity, and availability using symmetric encryption techniques and hash chaining. The main goal from here is to maintain the utility outlined in the functionality requirement and met by the reference design, while adding obfuscation and security to protect our system secrets. After initially covering the security requirements, we will trace back through with more advanced techniques to further strengthen our system.