Team Spartans

Michigan State University

Felipe Marques Allevato, Udbhav Saxena, Charles Selipsky Riley Cook, Aashish Harishchandre, Aditya Chaudhari, Fatima Saad, Samay Achar, Krishna Patel, Ramisa Anjum, Radhe Patel

Design Overview

The goal of the competition was to create a secure supply chain solution for a medical device, including the deployment infrastructure as well as the firmware for an *Application Processor* and *Components* that it controlled.
Our design modeled a public key infrastructure where the manufacturer had a signing keypair which created certificates for legitimate APs and Components. This was modeled after PKI of the internet, where Certificate Authorities (CAs) sign certificates to verify the identity of web server.

Defensive Highlight

In addition to the manufacturer keypair, each AP and Component has its own keypair which is signed by the manufacturer.

Our design involves a handshake between the Application Processor and Components prior to boot where each party verifies two things:

1) The AP/Component has a valid certificate signed by the host attesting to its public key
2) The AP/Component owns the public key included in the certificate (this is proved by signing a randomly generated challenge sent during the handshake)

After the handshake, both parties exchange an ephemeral keypair that is discarded after the session is ended to ensure *forward secrecy* of encrypted communications.

To ensure authenticity and integrity of post-boot communication between the MISC components, *authenticated encryption* is used through the ChaCha20-Poly1305 algorithm, using a counter as associated data to prevent replay attacks.

Another defensive measure we included was the use of flash memory to protect against brute force attacks. When we implemented a delay for PIN and Component ID attempts, we were concerned that teams may be able to automate resetting the board somehow and circumvent the delay. We essentially wrote a "have a delay" flag to a flash entry, and proceeded to write it to the flash before an attempt was even made to prevent all reset scenarios. Upon a successful attempt or complete reflash of the board, the flag was set to false. In the end, we couldn't figure out how to automate this reset-spam process on other teams due to the configuration of the attack boards, so we're not sure how effective the strategy was in practice. If any other teams were able to figure that out, we are confident this method protected us from brute force attacks.

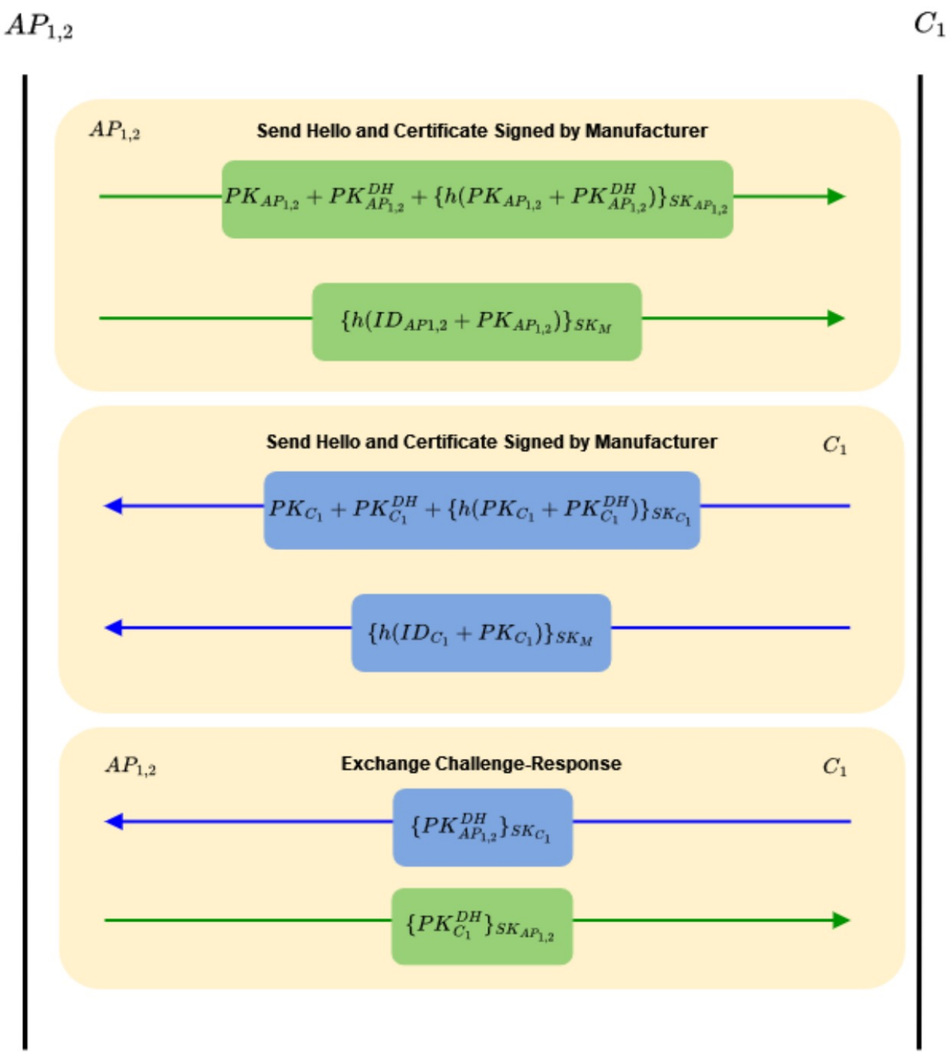

Figure 1: Custom TLS Handshake Diagram

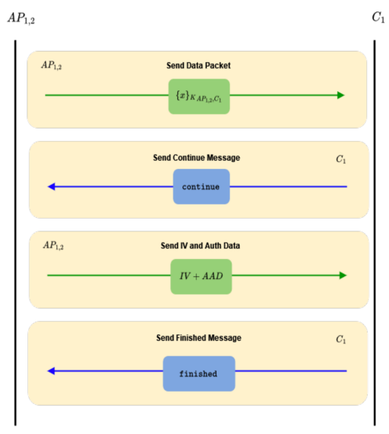

Figure 2: AP to Component Secure Communication Diagram

Figure 3: Component to AP Secure Communication Diagram

Offensive Highlights

Some common design flaws we exploited for attacks were:

1) Systems that shared a single common "password" across all components and exchanged it to verify themselves, which could be exploited by building a fake component to obtain the secret.
2) A design that involved a shared key derivation but had no certificates to verify the source of the component, meaning it would communicate with any device that took part in the key exchange.

These attacks were made possible due to the lack of a complete chain of trust in the public key infrastructure, which could be fixed by making sure that the design allowed for the manufacturer to attest to a genuine component through something like a signed certificate.

References

- "1. Introduction - WolfSSL Manual." *Www.wolfssl.com*, www.wolfssl.com/documentation/manuals/wolfssl/. Accessed 19 Apr. 2024.
- Boeyen, Sharon, et al. "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile." *IETF*, 1 May 2008, datatracker.ietf.org/doc/html/rfc5280.
- "Common Cryptographic Architecture (CCA): CSNDEDH." *Www.ibm.com*, www.ibm.com/docs/en/linux-on-z?topic=keys-ec-diffie-hellman-csndedh.
- "MAX78000FTHR Evaluation Board | Analog Devices." *Analog.com*, 2023, www.analog.com/en/resources/evaluation-hardware-and-software/evaluation-boards-kits/max78000fthr.html#eb-documentation. Accessed 19 Apr. 2024.
- Rescorla, Eric, and Tim Dierks. "The Transport Layer Security (TLS) Protocol Version 1.2." *IETF*, 1 Aug. 2008, datatracker.ietf.org/doc/html/rfc5246.