CyberSecurity Club at San Francisco State University

Initial Design Document

Ethan Hanlon[a], Michael Petrossian[a]

[a]San Francisco State University, 1600 Holloway Avenue, San Francisco, CA 94132, USA

Abstract

This research focuses on designing and implementing a secure MISC device that will be built to adhere to the standards of the functional and security requirements. Our design will aim to withstand all attack scenarios presented by the MITRE documentation, as well as provide a new approach to embedded system programming with the popular and recent memory-safe Rust programming language.

1. Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed posuere interdum sem. Quisque ligula eros ullamcorper quis, lacinia quis facilisis sed sapien. Mauris varius diam vitae arcu. Sed arcu lectus auctor vitae, consectetuer et venenatis eget velit. Sed augue orci, lacinia eu tincidunt et eleifend nec lacus. Donec ultricies nisl ut felis, suspendisse potenti. Lorem ipsum ligula ut hendrerit mollis, ipsum erat vehicula risus, eu suscipit sem libero nec erat. Aliquam erat volutpat. Sed congue augue vitae neque. Nulla consectetuer porttitor pede. Fusce purus morbi tortor magna condimentum vel, placerat id blandit sit amet tortor.

2. Functional Requirements

We plan to meet the functional requirements by doing x, y, and z.

3. Security Requirements

3.1. Security Requirement 1

Security Requirement 1 requires us to ensure the Application Processor will not boot unless expected components are present and valid. Our plan for this involves a two-pronged approach which uses public/private key encryption and a secure cryptographic nonce.

During the build process, we will generate a public/private key pair. The public key will be flashed onto the components, and the private key will be stored in the application processor. Additionally, a random seed will be generated using /dev/urandom and stored in the application processor. This seed will be used to generate a cryptographic nonce which will be sent to the components. The components will use the public key to encrypt the nonce along with their own unique identifier and send it back to the application processor. The application processor will use the private key to decrypt the nonce and verify that it matches the original nonce. If the nonces match, the application processor will know that the components are valid and will boot. If no message is received, or the nonces do not match, the application processor will not boot.

Once the application processor

References

[1] S. Scholes, Discuss. Faraday Soc. No. 50 (1970) 222.

[2] O.V. Mazurin and E.A. Porai-Koshits (eds.),

Acknowledgements

This template was originally created by the Partnership for Advanced Computing in Europe (PRACE). It was modified for use by the San Francisco State University eCTF team.

Make sure to include acknowledgements for people who helped in the project, including professors, graduate students, and other team members.