Multi-processor SoC systems are widely used nowadays. Symmetric multiproducting (SMP) systems are a popular architecture for multiprocessor systems, which classifies the processors into a bootstrap processor (BSP) and multiple application processors (APs). The authenticity of APs is under-examined since the APs are trusted by default. However, it turns out to be vulnerable when introducing the supply chain attack aiming to replace or counterfeit the APs.

In this paper, we systematically investigate critical stages of SMP system boot process, and discover a fundamental and non-trivial attack due to the lack of AP authentication. To mitigate the new attack, we propose PA-Boot, the first formally verified processor authentication protocol for secure boot of SMP systems. PA-Boot proved to be functionally correct, and is guaranteed to detect different adversarial behaviors, e.g., replacement of AP, tampering with certificates, and man-in-the-middle attacks. The fine-grained formal specification and its fully mechanized security proofs are carried out in the theorem prover Isabelle/HOL with 305 lemmas/theorems and ~7,000 LOC. We also implement PA-Boot in C as a prototype, and the performance and security evaluations show that it can identify boot process attacks efficiently and improve the security of SMP systems.

I. INTRODUCTION

Attacks during boot process are arguably the most difficult to defend against because at this stage in a device’s lifecycle, traditional defences such as firewalls and anti-viruses are not in place, and attacks are difficult to detect [1]. A common defence against boot attacks is authenticated or secure boot [2]. Recent works have shown a growing interest in secure boot verification both in academia and industry. However, despite extensive research focusing on boot process for uniprocessors, little attention has been paid to whether and how these verification techniques can be applied to symmetric multiproducting (SMP) systems. SMP refers to the computer architecture where multiple identical processors are interconnected to a single shared main memory, with full accessibility to all the I/O devices. Nowadays, given the increasing popularity of SMP systems, it is imperative to guarantee the security of their boot process.

In this paper, we focus on a fundamental but neglected problem for SMP secure boot, which is a specific type of attack stemming from the authentication of the application processors (AP). Specifically, while all the processors in a compliant system are functionally identical, SMP specification classifies them into two types: the bootstrap processor (BSP) and the application processors (AP). In the boot process, each processor first executes boot code in its own bootROM. Then, the BSP is responsible for initializing the system and booting the operating system; APs are activated only after the operating system is up and running with the same privilege as the BSP.

To meet the requirement for stronger security in such a scenario, we dig into the SMP system boot process and develop a processor authentication protocol, called PA-Boot, to validate the authenticity of APs. We also specify and verify PA-Boot based on formal techniques, providing a strong assurance of its correctness and security. In addition, guided by principles of practicality and performance, we implement PA-Boot as a prototype. It offers the user a trusted and efficient way to ensure whether the processor is genuine or has been manipulated. To the best of our knowledge, this is the first study that systematically verifies the authenticity of APs in SMP boot process based on formal methods.

In detail, the technical contributions of our work are as follows:

- **Systematical investigation:** We systematically investigate critical stages of SMP system boot process and discover a novel attack in AP authentication.
- **Formally verified PA-Boot:** To address the above problem, we propose a processor authentication protocol, named PA-Boot, that can satisfy security goals for secure boot in the presence of an active attacker executing simultaneously with the protocol. We provide a mechanically checked formal specification for PA-Boot.
TABLE I: Security statistics for three types of attacks.

<table>
<thead>
<tr>
<th>Category</th>
<th>Tamper with Packets in Shared Memory</th>
<th>Tamper with Certificates in Flash</th>
<th>Replace the AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack</td>
<td>$P_{Cert}$</td>
<td>$P_{Chall}$</td>
<td>$P_{Resp}$</td>
</tr>
<tr>
<td>Total Time</td>
<td>2,502</td>
<td>2,453</td>
<td>2,418</td>
</tr>
<tr>
<td>Time</td>
<td>1 1 1 5 1 1.272 1,222</td>
<td>6 6 12 14 4 2,411</td>
<td>1 3 8 12 6 783 1,605</td>
</tr>
<tr>
<td>Error Code</td>
<td>❶ ❷ ❸ ❹ ❼ ❾ ❽</td>
<td>❶ ❷ ❸ ❹ ❼ ❾ ❽</td>
<td>❶ ❷ ❸ ❹ ❼ ❾ ❽</td>
</tr>
</tbody>
</table>

- **Implementation and evaluation:** We implement PA-Boot in C based on a code-to-spec review and validate its performance and security by extensive evaluations. The results show that it offers adequate performance in practice and can improve the security of SMP systems.

II. WORKFLOW

Figure 1 depicts PA-Boot’s workflow. First, we find vulnerabilities in SMP processor authentication process, and design PA-Boot as a mitigation approach. Based on it, a threat model is explicitly described with respect to the attacker’s fundamental target, security assets and three potential attacks. Then, security goals are defined coarsely in terms of integrity, authenticity and confidentiality. Afterward, following the informal threat model and security goals, we formally derive $M_A$ in Isabelle/HOL [4], a high-level functional specification of PA-Boot. In addition, the security goals are formally specified into high-level security properties and are verified to be satisfied by $M_A$. Next, to describe a more detailed design of PA-Boot, we specify $M_C$, which is an instance of $M_A$. It succinctly captures the key attributes and behaviors of various components in the threat model. Based on $M_C$, we extend each of the high-level security properties into a low-level one that is stronger and more precise. $M_C$ is proved to hold all of them based on refinement relation. Afterward, based on human review and logical equivalence transformation from $M_C$, a practical processor authentication protocol is implemented in C. As it follows the formally verified protocol design, it is guaranteed to meet our security goals. Finally, we evaluate the protocol in terms of performance and security, and the results indicate its high efficiency and security.

III. IMPLEMENTATION AND EVALUATION

We implemented our PA-Boot with about 3000 lines of C code as a prototype, along with roughly 1000 lines for performance and security tests. Specifically, we measure the running time of the certificate generation stage and the processor authentication stage 10,000 times each. The average running time is around 28ms for certificate generation, and around 110ms for processor authentication. For security evaluation, we simulate three types of attacks, i.e., man-in-the-middle attacks, flash compromise, and AP replacement, 10,000 times each. We then record the return value to analyze if our system can discover the attacks as expected (see Table I). In summary, the results show that PA-Boot offers adequate performance and completely complies with our security goals.

IV. CONCLUSION

The security of boot process depends on the weakest link in the authentication chain. This paper address this point by proposing a formally verified PA-Boot to mitigate the new boot attack resulting from AP replacement in SMP system. This paper intends to rethink the security design of SMP boot process and motivate the researchers/developers to draw more attention to the “known-safe” or “assumed-safe” components in the existing systems. Therefore, this work is of great value for identifying boot process attacks and improving the boot ecosystem’s overall security.

REFERENCES

1. Contribution and Workflow

- **Discovery of a new attack.** After a systematic investigation in SMP system secure boot, find a fundamental but neglected vulnerability due to the lack of AP.

- **Proposing a mitigation method.** Define a threat model describing the attacker’s fundamental target, security assets, and three types of attacks. Based on this, define security goals in terms of integrity, authenticity and confidentiality. Design a processor authentication protocol, called PA-Boot, that can satisfy the security goals in the presence of an active attacker executing concurrently with the protocol.

- **Formal Specification and Verification.** Formalize the protocol and verify it against the security properties. The mechanical verification is performed in Isabelle/HOL, a state-of-the-art interactive theorem prover.

- **Implementation and evaluation.** Implement the protocol in C and validate its effectiveness, scalability and security by extensive evaluations.

![Workflow](image1)

2. Motivation and Threat Model

- **Fig. 3:** Security assets in threat model.

- **Fig. 2:** Adversarial behaviors.

3. Protocol Design

**Certificates Validation.** The immutable hash value of root certificate is stored in bootROM, serving as a chain of trust. Use this to help validate the legality of certificates.

**Challenge-Response.** Obtains the other processor’s public key from the validated certificate for encrypted communication. After authentication, use DH algorithm to generate shared session key for subsequent secret communications between the two processors.

![Protocol Design](image2)

4. Formal Specification and Verification

- **Formalize PA-Boot as a state machine, consisting of states and events.** Formalize security goals as three security properties.

- **Prove if the model satisfies the security properties.**

**Security Properties Covering all Execution Traces**

- **Security Property under Normal Execution:** Starting from initial state where the initial configuration has not been tampered, and there is no adversary behaviors during execution, then the final state must be an ideal state.

  \[
  \Gamma \vdash \text{ops} \Rightarrow \text{P(ops)} \text{ and } R(t) \Rightarrow \forall \text{V(t)}
  \]

- **Security Property under Tampered Configuration:** Starting with compromised initial configuration, the system should terminate in a bad state.

  \[
  \Gamma' \vdash \text{ops} \Rightarrow t \text{ and } R(t) \Rightarrow \neg \forall \text{V(t)}
  \]

- **Security Property under Adversarial behaviors:** Any adversary behaviors during system execution should be detected, resulting in a bad final state.

  \[
  \Gamma' \vdash \text{ops} \Rightarrow t \Rightarrow P(\text{ops}) \text{ and } R(t) \Rightarrow \neg \forall \text{V(t)}
  \]

**Security Properties Covering all Execution Traces**

- **Performance:** Measure the running time of the certificate generation stage and the processor authentication stage, respectively.

- **Security:** Simulate three types of attacks, i.e., man-in-the-middle attacks, flash compromise, AP replacement, 10,000 times each. Record the return value to analyze if our system can discover the attacks as expected.

![Performance](image3)

5. Implementation and Evaluations

**Implement PA-Boot in C, and Evaluate It in Two-fold**

- **Security Property under Normal Execution:**

  \[
  \Gamma \vdash \text{ops} \Rightarrow \text{P(ops)} \text{ and } R(t) \Rightarrow \forall \text{V(t)}
  \]

- **Security Property under Tampered Configuration:**

  \[
  \Gamma' \vdash \text{ops} \Rightarrow t \text{ and } R(t) \Rightarrow \neg \forall \text{V(t)}
  \]

- **Security Property under Adversarial behaviors:**

  \[
  \Gamma' \vdash \text{ops} \Rightarrow t \Rightarrow P(\text{ops}) \text{ and } R(t) \Rightarrow \neg \forall \text{V(t)}
  \]

**Security Properties Covering all Execution Traces**

- **Performance:** Measure the running time of the certificate generation stage and the processor authentication stage, respectively.

- **Security:** Simulate three types of attacks, i.e., man-in-the-middle attacks, flash compromise, AP replacement, 10,000 times each. Record the return value to analyze if our system can discover the attacks as expected.

![Performance](image4)

**Table 2:** Security statistics for three types of attacks.

<table>
<thead>
<tr>
<th>Category</th>
<th>Tamper with Flash in Shared Memory</th>
<th>Tamper with Certificate in Flash</th>
<th>Tamper with Signature Error</th>
<th>Replace the AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack</td>
<td>Power2s</td>
<td>Power2s</td>
<td>Power2s</td>
<td>Time</td>
</tr>
<tr>
<td>Total</td>
<td>2,832</td>
<td>2,326</td>
<td>2,326</td>
<td>2,627</td>
</tr>
<tr>
<td>Time</td>
<td>1.5 sec</td>
<td>1.2 sec</td>
<td>1.2 sec</td>
<td>1.2 sec</td>
</tr>
</tbody>
</table>

**Error Code:**
- **Power2s** means the attack is successful.
- **Power2s** means the attack is not successful.
- **Time** means the attack time is longer than 10 minutes.
- **Replace the AP** means the attack replaces the AP.

**Conclusion:** PA-Boot offers adequate performance and completely complies with our security goals.

Contact: Zhuoruo Zhang (zhangzhuoruo@zju.edu.cn), Rui Chang (crix1021@zju.edu.cn), Qinming Dai (qinm_dai@zju.edu.cn) and Kui Ren (kuiren@zju.edu.cn).