Security Advisory

Title	Security Advisory Concerning Bypassing Secure Boot and Flash Encryption using CPA and FI attack on ESP32-C3 and ESP32-C6
Issue Date	2024/01/05
Advisory Number	AR2023-007
Serial Number	NA
Version	V1.0

Issue Summary

There is a hardware vulnerability reported on ESP32-C3 and ESP32-C6 Chips, where using side channel attack Flash Encryption feature implemented with XTS-AES algorithm support could be bypassed.

This attack uses a combination of Correlation Power Analysis (CPA), Fault Injection (FI), and a buffer overflow exploitation to take over the control of the first block on the encrypted flash. Once the first block on the flash is in the attacker's control, it is populated with attacker defined shellcode. In subsequent boot, as part of the boot sequence shellcode will be loaded into the internal memory. Before Secure Boot scheme could identify that loaded code is tampered and abort the boot, CPU will be tricked to jump and execute the shellcode. By carefully crafting this shellcode, the attacker can extract the secrets from the device.

- ### What is Side Channel Attack (SCA)?

  A side-channel attack exploits unintentional information leakage from a system to uncover secret values, typically encryption keys. The side channel can take various forms, including timing variations or the power consumption of a device.

  The side-channel attacks on ESP32-C3 and ESP32-C6 in this statement are based on Correlation Power Analysis (CPA).

- ### What is Fault Injection?

A fault-injection attack is a technique where intentional errors or faults (e.g., voltage or clock) are introduced into a system to assess its resilience and vulnerabilities. By manipulating the system's behavior through injected faults, attackers aim to identify weaknesses and exploit potential security flaws.

The fault-injection attacks on ESP32-C3 and ESP32-C6 in this statement are based on voltage glitch injection. It's an invasive attack technique where execution requires a level of expertise, precision and resources that makes it less practical for many attackers.

- **Impact Analysis**

1. Chips under this attack use XTS-AES encryption mode for Flash Encryption where a separate encryption key is used for each flash block. Using CPA technique, the attacker can extract the encryption key for the first flash block. Which could be further exploited to extract the device secrets.

2. CPA attack technique to recover key requires significant amount of power trace collection (around 600,000) and take time in the tune of 5 days. In addition, the CPA attack cannot recover the tweak key, but can only recover the tweak value of each block, which means that each attack can only break through a mere 128 bytes of data. Further, to break the XTS-AES mode required to recover both tweak values and encryption key. This makes this attack more complicated.

3. Decrypting the entire encrypted flash using the technique mentioned in point 2 above becomes impractical both in terms of effort and time required.

4. This attack also required to bypass the Secure Boot and find and employ buffer overflow using Fault Injection technique (carefully crafted voltage glitch) in ROM code to load and execute the shellcode in internal memory.

5. The complexity and time required to extract encryption keys limits the size of the shellcode attacker can load and execute on the device. This creates an additional barrier and works as deterrent for the attacker. Since each device uses a unique Flash Encryption key, this attack cannot scale to class attack.

Mitigation

At present there is no software and hardware fix available for this issue. Future products will incorporate hardware countermeasures in the chip to address these issues. The following are some recommendations to mitigate these issues.

- **Hardware Countermeasures**

SCA attack: protect the device from physical access by enclosing it with a tamper resistant mechanism which could not be broken without detection, to effectively avoid the implementation of fault-injection. Device should respond to tamper

detection as per the predetermined action, e.g., reset the device, clear-out the secret information on the device.

- **Application Countermeasures**

  Long lived encryption keys that are common between the devices or manufacturing batch should be avoided at all costs.

  These attacks need significant effort, skill, expensive and sophisticated lab equipment to be carried out successfully on a device. If each device is provisioned with a unique secret tied to that specific device identity, then the attacker cannot scale it to an entire class of devices, making this attack less attractive. In addition, we recommend that chip users enable Flash Encryption and Secure Boot at the same time, which can minimize the risk of attacker rewriting with the firmware.

  Several Espressif products are available in System-in-Package (SiP) form-factor with flash pins terminated internally. These SiP (such as ESP32-PICO-V3) can protect against this type of attack better. This prevents usage of any external flash emulator or monitoring of flash pins as was used in the Flash Encryption related attack discussed in this advisory.

Other Espressif Products

CPA attacks can theoretically be applicable to all chips containing XTS-AES, which includes ESP32-C2, ESP32-C3, ESP32-S2, ESP32-S3, ESP32-C6, ESP32-H2, ESP32-P4 and ESP32-C5. In ESP32-C5's XTS-AES, a pseudo-round mechanism has been added to resist side-channel attacks, significantly increasing the difficulty of implementing CPA attacks. However, the successful execution of the complete attack also hinges on the viability of the second step, Fault Injection (FI), which is intricately linked to the ROMs within each series. In subsequent chip series (such as ESP32-H2, ESP32-P4, ESP32-C5 etc.), the hardware-integrated glitch detection circuit can identify the glitch used in this attack. Once such a glitch is detected, the chip will automatically reset.

ESP32, including Chip Revision v3.0 and v3.1, does not feature XTS-AES mechanism. The impact of this combined CPA and FI attack on ESP32 is not discussed here.

Credits

We would like to thank **Kévin Courdesses**, an exceptional hardware and firmware engineer, for reporting this vulnerability and following up on responsible disclosure.