# Microchip AT91Bootstrap Code Authentication Issues

CVE-2020-11683, CVE-2020-11684

**Type**

Information disclosure, Side channel

**Severity**

Medium

**Affected products**

Microchip AT91bootstrap

**Credits**

The issues were discovered by Dmitry Janushkevich of F-Secure Hardware Security team.

**CVE Reference**

CVE-2020-11683, CVE-2020-11684

Read more

Timeline

| | |
|---|---|
| 2020-02-07 | First contact |
| 2020-02-07 | Details provided to Microchip |
| 2020-02-25 | F-Secure requests status update |
| 2020-02-27 | Microchip informs of fixes being prepared for the next software release. |
| 2020-03-30 | Without notifying F-Secure, Microchip publishes the patches |
| 2020-03-31 | Microchip provides the patches for review |
| 2020-04-06 | F-Secure discovers the patches are public |

## Background

The AT91Bootstrap project is the 2nd level bootloader for certain families of Microchip (formerly Atmel) microprocessors (aka AT91). It provides a set of algorithms to manage the hardware initialization, download the next boot stage from specified media, authenticate it, and start it.

Two issues were identified in the code authentication functionality which allow attackers with physical access to disclose encryption keys and conduct side channel analysis attacks.

## Sensitive data remains in memory after AT91bootstrap hands control to application

One of the main tasks of the AT91bootstrap component is to authenticate and decrypt the next stage, for example, the U-Boot loader. For that, a set of hardcoded credentials is used, consisting of a key and an initialization vector (IV) for decryption as well as a CMAC key for authentication.

It was found that these sensitive keys, which are part of AT91bootstrap image, are not completely wiped from memory after authentication and decryption is complete; only a copy located in stack variables is wiped, however the original values persist as part of the code segment.

```
int secure_decrypt(void *data, unsigned int data_length, int is_signed)
{
...
  /* Init keys */
  init_keys(&key_size, cipher_key, cmac_key, iv);

  /* Init periph */
```

```
    /* Reset periph */
    at91_aes_cleanup();

    /* Reset keys */
    memset(cmac_key, 0, sizeof(cmac_key));
    memset(cipher_key, 0, sizeof(cipher_key));
    memset(iv, 0, sizeof(iv));

    return rc;
```

This allows the application to intercept these values when control is received from the bootstrap.

To verify this finding, the bootstrap was built with recognizable keys (words with equal 4-bit nibbles set to incrementing value for each word). U-Boot was used as the application and configured to provide a command console. After the boot process was stopped, it was possible to dump memory where the bootstrap code is located, and with that, extract key material. The following abbreviated console capture shows how this was done.

```
RomBOOT
Secure Boot Mode

[ REMOVED FOR BREVITY ]
AT91Bootstrap 3.8.10 (Fri Jan 31 04:59:36 UTC 2020)
[ REMOVED FOR BREVITY ]
U-Boot 2017.03-linux4sam_5.8 (Jan 26 2020 - 14:16:26 +0000)
[ REMOVED FOR BREVITY ]
Hit any key to stop autoboot:  0
=> md 00205150 14
00205150: aaaaaaaa 77777777 88888888 99999999    ....wwww........
00205160: 11111111 33333333 44444444 dddddddd    ....3333DDDD....
[REMOVED]
```

The key material can be recognized in the dumped data. Analysis showed the addresses belong to the code section; the address is provided directly in the example above for illustrative purposes.

**Impact**

An attacker able to compromise the application stage, for example by forging the CMAC value as described below, gains access to cryptographic material used to encrypt and authenticate the application stage. This results in a compromise of the expected security guarantees and the ability to forge arbitrary applications.

## Timing side channel exists when verifying CMAC

Apart from direct information leakage, for example, due to software disclosing sensitive data directly, it is possible to attack software using side channels which leak information indirectly. Timing side channel is one such kind which is caused due to time variations in certain algorithms depending on data being processed.

The AT91boostrap code uses cipher-based message authentication code (CMAC) to authenticate the next stage before handing control over. By inspecting the source code, it was found that the AT91bootstrap loader employs such a time-varying operation in security critical code which verifies CMAC validity. This allows an attacker to dramatically speed up

```
int secure_decrypt(void *data, unsigned int data_length, int is_signed)
{
...

   /* Check signature if required */
   if (is_signed) {
      /* Compute the CMAC */
      if (at91_aes_cmac(data_length, data, computed_cmac,
            key_size, cmac_key))
         goto exit;

      /* Check the CMAC */
      fixed_length = at91_aes_roundup(data_length);
      cmac = (const unsigned int *)((char *)data + fixed_length);
      if (memcmp(cmac, computed_cmac,
AT91_AES_BLOCK_SIZE_BYTE))
         goto exit;
   }
...
```

The memcmp() C library function does not provide any guarantee regarding its performance or fitness for cryptographic use, and its use in sensitive contexts is generally discouraged. The function's execution time strongly depends on how much data compares equal, allowing making precise guesses whether a given CMAC byte is correct or not. To provide an example, assume the attacker knows the first N bytes of a CMAC value for the given text; they take time measurements while trying all 256 values for the N+1'th byte. For the correct guess, execution time will be slightly longer; this value is accepted as correct and the procedure is repeated for the next byte.

#### Impact

By timing the response, an attacker with physical access is able to conduct efficient attacks against the CMAC verification step and forge CMAC values for manipulated application images. While the images are encrypted, am attack may be conducted against AES-CBC by freely manipulating one block of encrypted data using XOR.

## Affected versions

The issues were identified in AT91bootstrap version 3.9.1. Prior versions may also be affected.

## Solution

Update to version 3.9.2 when available or apply patches provided by the vendor.