



ESPILON
SECURITY RESEARCH

SECURITY ADVISORY

Out-of-bounds read in the ESP-IDF ESP-TEE secure-service wrappers

CVE-2026-45329

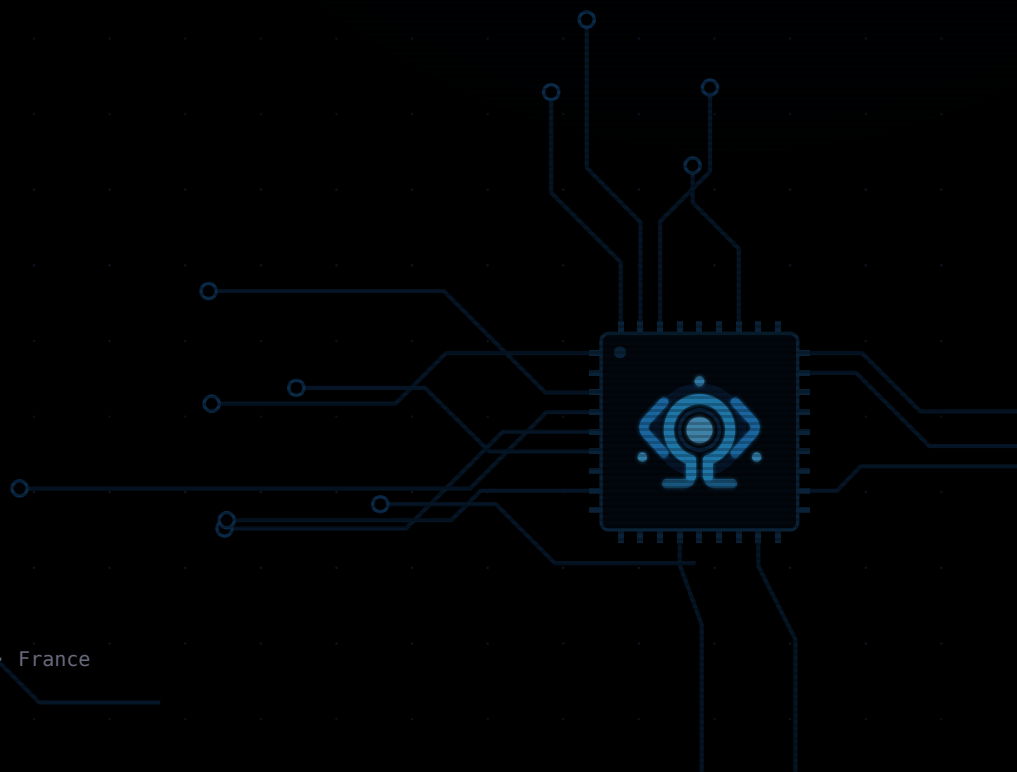
CVSS 7.1 · HIGH

Espressif · ESP-IDF · ESP-TEE

CVE PUBLISHED 10 Jun 2026

Marouane El Moufid (Eun0us)

espilon.net · Association loi 1901 · France



01 Vulnerability

7.1

CVSS 3.1 BASE SCORE · OFFICIAL (GHSA, GITHUB CNA)

AV:L / AC:L / PR:N / UI:N / S:C / C:H / I:N / A:N

HIGH

CVE	CVE-2026-45329 · GHSA-w82j-7q63-7pqm
Product	Espressif ESP-IDF, component <code>esp_tee</code> (ESP-TEE), on ESP32-C5, ESP32-C6, ESP32-C61, ESP32-H2
Location	<code>components/esp_tee/subproject/main/core/esp_secure_services.c</code> and <code>esp_secure_services_iram.c</code>
Type	Out-of-bounds read (CWE-125), improper input validation (CWE-20), information exposure (CWE-200)
Attacker	Local: code running in the REE (normal world), no TEE privilege
Published	2026-05-13 (vendor advisory) · 2026-06-10 (CVE record)
Credit	Marouane El Moufid (EunOus), Epsilon

Summary

ESP-TEE splits the chip into a trusted world (TEE, RISC-V machine mode) and the normal world (REE). The REE reaches TEE services through wrapper functions that are supposed to check every pointer it passes. Several wrappers checked only some of their pointer arguments and left input pointers unchecked. The crypto peripherals behind them (SHA, ECC) and the SPI flash path run with full address-space access, so an REE application can point an input at TEE-exclusive memory and receive a result derived from it. This is the read-side sibling of CVE-2026-45328.

Affected and fixed versions

BRANCH	AFFECTED	FIXED IN	FIX COMMIT
master			145ba4c
v6.0	v6.0	v6.0.1 (2026-04-27)	7867f4a
v5.5	v5.5.4	v5.5.5 (2026-07-17)	eebabaf

Affected and fixed releases are those of the vendor advisory; release dates and the presence of each branch commit in its fixed release were checked against the `esp-idf` repository on 2026-10-28. The same commits also fix the write-side issue, CVE-2026-45328, whose own advisory (GHSA-mmgp-73p4-92xp) was later amended: on the 6.0 line the write is fixed only in v6.0.2.

02 Technical details

The check the wrappers use

At v6.0, a wrapper validates a pointer with `esp_tee_ptr_in_ree()`, which tests one address against the REE memory ranges. A wrapper protects a buffer only if it calls this on each pointer it passes on (and, for a range, on its end). Its v6.0 body appears as context in the fix diff of section 04. Code below is quoted from the ESP-IDF v6.0 release.

SHA: no check at all

The SHA state and block services pass the caller's pointer straight to the hardware layer. `_ss_esp_sha_write_digest_state()` loads the digest state from `digest_state`, and `_ss_esp_sha_block()` feeds one message block from `data_block` into the SHA engine. Neither pointer is checked.

```
components/esp_tee/subproject/main/core/esp_secure_services.c · ESP-IDF v6.0

199 void _ss_esp_sha_read_digest_state(esp_sha_type sha_type, void *digest_state)
200 {
201     sha_hal_read_digest(sha_type, digest_state);
202 }
203
204 void _ss_esp_sha_write_digest_state(esp_sha_type sha_type, void *digest_state)
205 {
206     sha_hal_write_digest(sha_type, digest_state);
207 }
208
209 void _ss_esp_sha_block(esp_sha_type sha_type, const void *data_block, bool is_first_block)
210 {
211     esp_sha_block(sha_type, data_block, is_first_block);
212 }
```

The output pointer of `_ss_esp_sha_read_digest_state()` (line 199) is also unchecked; a pointer the TEE writes to belongs to the write side, CVE-2026-45328, and is not discussed further here.

ECC: output checked, inputs not

`_ss_esp_ecc_point_multiply()` checks the result buffer it writes, but not the point and scalar it reads. `_ss_esp_ecc_point_verify()` checks nothing and returns the verification result to the caller.

```
components/esp_tee/subproject/main/core/esp_secure_services.c · ESP-IDF v6.0

372 int _ss_esp_ecc_point_multiply(const ecc_point_t *point, const uint8_t *scalar, ecc_point_t *result,
373 bool verify_first)
374 {
375     bool valid_addr = (esp_tee_ptr_in_ree((void *)result)) &&
376                       esp_tee_ptr_in_ree((void *)((char *)result + sizeof(ecc_point_t)));
377     if (!valid_addr) {
378         return -1;
379     }
380     ESP_FAULT_ASSERT(valid_addr);
381     return esp_ecc_point_multiply(point, scalar, result, verify_first);
382 }
383
384 int _ss_esp_ecc_point_verify(const ecc_point_t *point)
385 {
386     return esp_ecc_point_verify(point);
387 }
388 }
```

SPI flash: controller structures not checked

In `esp_secure_services_iram.c` the SPI flash wrappers check the flash address range against the TEE partitions, but not the host and trans structures the TEE reads to drive the controller. At v6.0, `_ss_spi_flash_hal_common_command()` (line 285) dereferences `trans` and passes `host` on without a range check, and `_ss_spi_flash_hal_device_config()` (line 296) checks nothing.

What the attacker controls and reaches

The attacker is any code running in the REE, including an unprivileged application. It chooses the address handed to an unchecked input pointer, including addresses inside TEE SRAM, and calls the service through the normal secure-service interface. The TEE then reads that memory in machine mode on the attacker's behalf. No TEE privilege, key or authentication is required.

03 Impact

The advisory describes three outcomes depending on the wrapper: the result returned to the REE contains raw bytes of TEE memory, or a computed function of TEE memory that can be recovered through repeated calls, or a single bit per call that serves as an oracle for incremental disclosure. In the code above, `_ss_esp_ecc_point_verify()` is of the third kind: its only output is whether the bytes at the chosen address form a valid curve point. The SHA and ECC multiply services return values computed from the bytes read.

The data at stake is what the TEE exists to isolate: TEE code and data, cryptographic keys and other secrets held in the TEE. The advisory also notes that disclosing TEE addresses and internal pointers weakens the mitigations that protect the TEE and makes memory-corruption attacks against it more reliable. The scope change in the CNA vector reflects the crossing from the REE into the TEE.

Evidence

The issue was reported to Espressif as part of a set of ESP-TEE reports. On 2026-04-13 Espressif's bug bounty team confirmed that these reports were fixed by 145ba4c, and on 2026-05-12 it told us that three of them (our internal references TEE-12, TEE-16 and TEE-17) were grouped under this advisory. This paper does not include a silicon measurement of the read on v6.0 or v5.5.4, and no exploit code is published.

04 Fix and recommendation

Upstream fix

The fix routes every pointer argument through a new helper, `esp_tee_buf_in_ree()`, which rejects zero lengths and address overflow and checks both ends of the buffer, enforced with `ESP_FAULT_ASSERT()`. It also applies the check to pointers inside structure contexts.

• 7867f4a · release/v6.0 · components/esp_tee/subproject/main/include/esp_tee_memory_utils.h (abridged)

```
@@ -16,10 +18,10 @@ extern "C" {
FORCE_INLINE_ATTR bool esp_tee_ptr_in_ree(const void *p)
{
-   intptr_t addr = (intptr_t)p;
+   uintptr_t addr = (uintptr_t)p;
    return (
        (addr >= SOC_NS_IDRAM_START && addr < SOC_NS_IDRAM_END) ||
-       (addr >= (intptr_t)esp_tee_app_config.ns_drom_start &&
+       (addr >= (uintptr_t)esp_tee_app_config.ns_drom_start &&
        addr < SOC_S_MMU_MMAP_RESV_START_VADDR)
    #if SOC_RTC_MEM_SUPPORTED
        || (addr >= SOC_RTC_DATA_LOW && addr < SOC_RTC_DATA_HIGH)
@@ -27,6 +29,19 @@ FORCE_INLINE_ATTR bool esp_tee_ptr_in_ree(const void *p)
    );
}

+FORCE_INLINE_ATTR bool esp_tee_buf_in_ree(const void *p, size_t len)
+{
+   uintptr_t start = (uintptr_t)p;
+   /* Reject zero-length and overflow */
+   if (len == 0 || len > (SIZE_MAX - start)) {
+       return false;
+   }
+   return esp_tee_ptr_in_ree(p) &&
+       esp_tee_ptr_in_ree((const char *)p + len - 1);
+}
+
#ifdef __cplusplus
}
#endif
```

The ECC wrappers, as an example of how the input pointers are now covered. The SHA and SPI wrappers receive the same treatment, with buffer sizes taken from new per-algorithm helpers. Hunk line numbers refer to release/v6.0 at the time of the fix, not to the v6.0 tag quoted above.

• 7867f4a · release/v6.0 · components/esp_tee/subproject/main/core/esp_secure_services.c (abridged)

```
@@ -394,8 +501,11 @@ void _ss_esp_crypto_mpi_enable_periph_clk(bool enable)
    #if SOC_ECC_SUPPORTED
    int _ss_esp_ecc_point_multiply(const ecc_point_t *point, const uint8_t *scalar, ecc_point_t *result, bool
    verify_first)
    {
-       bool valid_addr = (esp_tee_ptr_in_ree((void *)result)) &&
-                           esp_tee_ptr_in_ree((void *)((char *)result + sizeof(ecc_point_t)));
+       bool valid_addr = (esp_tee_buf_in_ree(point, sizeof(ecc_point_t)) &&
+                           esp_tee_buf_in_ree(result, sizeof(ecc_point_t)));
+       if (valid_addr) {
+           valid_addr = esp_tee_buf_in_ree(scalar, MAX_SIZE);
+       }

        if (!valid_addr) {
            return -1;
        }

@@ -407,6 +517,13 @@ int _ss_esp_ecc_point_multiply(const ecc_point_t *point, const uint8_t *scalar,
    int _ss_esp_ecc_point_verify(const ecc_point_t *point)
    {
+       bool valid_addr = esp_tee_buf_in_ree(point, sizeof(ecc_point_t));
+       if (!valid_addr) {
+           return -1;
+       }
+       ESP_FAULT_ASSERT(valid_addr);
+       return esp_ecc_point_verify(point);
    }
    #endif
```

Recommendation

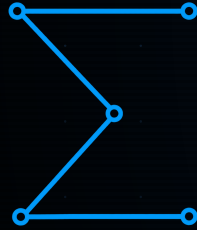
For this issue the advisory names v6.0.1 and v5.5.5. Because the sibling write, CVE-2026-45328, is fixed on the 6.0 line only in v6.0.2, upgrade to ESP-IDF v6.0.2 or later on that line, and to v5.5.5 or later on the 5.5 line. A product that cannot upgrade should cherry-pick the commits listed in both advisories. There is no runtime workaround.

05 Timeline

DATE	EVENT
2026-04-07	Fix committed to master (145ba4c); release/v6.0 follows on 2026-04-08 (7867f4a), release/v5.5 on 2026-04-09 (eebabaf)
2026-04-13	Espressif bug bounty reply: our ESP-TEE reports confirmed fixed by 145ba4c
2026-04-27	ESP-IDF v6.0.1 released with the fix
2026-05-12	Espressif reply: advisories created, three of our reports grouped under this one
2026-05-13	Advisory GHSA-w82j-7q63-7pqm published
2026-06-10	CVE-2026-45329 record published (CNA: GitHub)
2026-07-17	ESP-IDF v5.5.5 released with the fix
2026-08-31	Public write-up of CVE-2026-45328 and CVE-2026-45329 on blog.espilon.net

Credit and references

- Credit: Marouane El Moufid (Eun0us), Espilon. The vendor advisory credits “Eun0us / Espilon Worker”.
- Vendor advisory: <https://github.com/espressif/esp-idf/security/advisories/GHSA-w82j-7q63-7pqm>
 - CVE record: <https://www.cve.org/CVERecord?id=CVE-2026-45329>
 - Fix, master: [espressif/esp-idf@145ba4c](https://github.com/espressif/esp-idf/commit/145ba4c)
 - Fix, release/v6.0: [espressif/esp-idf@7867f4a](https://github.com/espressif/esp-idf/commit/7867f4a) · release/v5.5: [espressif/esp-idf@eebabaf](https://github.com/espressif/esp-idf/commit/eebabaf)
 - Sibling write: [GHSA-mmgp-73p4-92xp](https://github.com/espressif/esp-idf/commit/eebabaf) (CVE-2026-45328)
 - Write-up: <https://blog.espilon.net/posts/cve-2026-45328-esp-tee-oob-write/>



ESPILON

SECURITY RESEARCH

Embedded, IoT and OT security research

espilon.net

eun0us@espilon.net

Association loi 1901 • France