



**ESPILON**  
SECURITY RESEARCH

SECURITY ADVISORY

# Out-of-bounds write in ArduinoBLE's ATT write-request handler

**CVE-2026-47773**

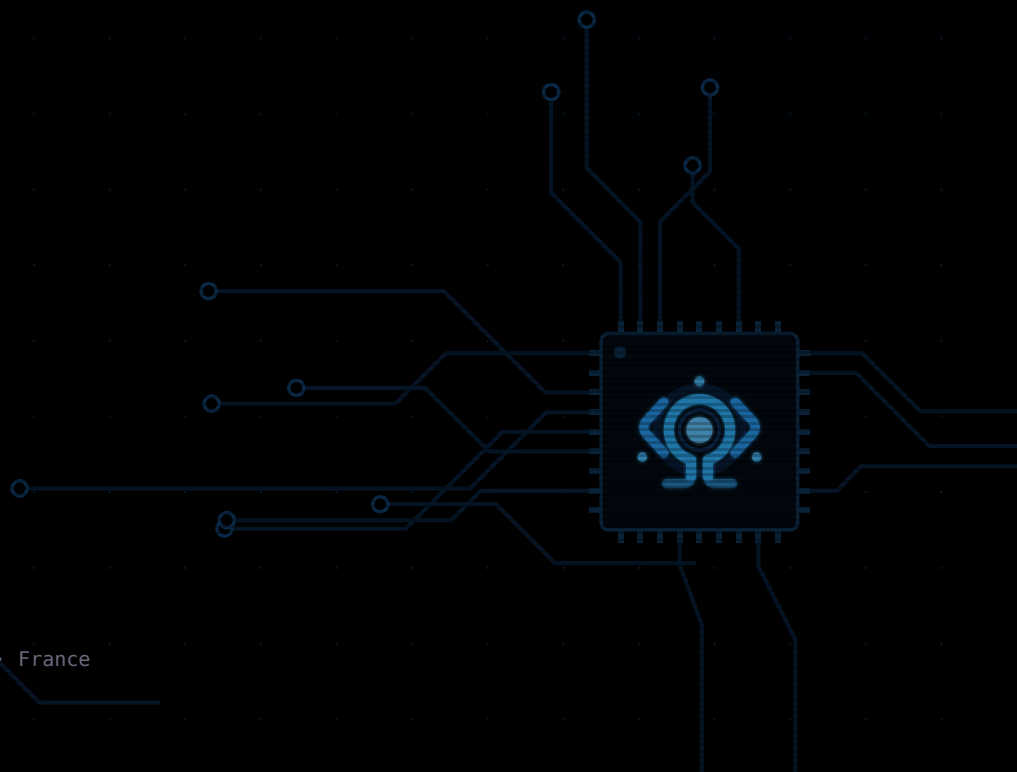
CVSS 7.2 · HIGH

Arduino · ArduinoBLE

CVE PUBLISHED 11 Sep 2026

Marouane El Moufid (Eun0us)

espilon.net · Association loi 1901 · France



# 01 Vulnerability

## 7.2

CVSS 4.0 BASE SCORE · GITHUB (CNA)

AV:A / AC:L / AT:N / PR:N / UI:N / VC:N / VI:H / VA:H / SC:N / SI:N / SA:N

HIGH

|           |                                                                                                |
|-----------|------------------------------------------------------------------------------------------------|
| CVE       | CVE-2026-47773                                                                                 |
| Component | ArduinoBLE, ATTClass::writeReqOrCmd (src/utility/ATT.cpp); writeBuffer[64] (src/utility/ATT.h) |
| Affected  | ArduinoBLE <= 2.0.1                                                                            |
| Fixed     | ArduinoBLE 2.0.2 (commit 1460e1a, PR #431)                                                     |
| Type      | Out-of-bounds write (CWE-787) + incorrect buffer-size calculation (CWE-131)                    |
| Advisory  | <a href="#">GHSA-77v6-cw9f-9whg</a>                                                            |
| Credit    | Epsilon · Marouane El Moufid (Eun0us)                                                          |

### Issue

ArduinoBLE is the Bluetooth Low Energy stack for Arduino boards (Nano 33 BLE, Nano RP2040 Connect, and others). At its core is a single global `ATTClass` object that parses incoming ATT PDUs. When a client writes to a characteristic that requires encryption (the `BLEEncryption` permission) but the link is not yet AES-encrypted, ArduinoBLE stashes the pending write into a fixed 64-byte `writeBuffer` inside that global object, to replay it once encryption is up. It copies the attacker-supplied value into that buffer with no length check. A remote, unauthenticated BLE client sends an oversized write and overflows the buffer with content it fully controls, corrupting whatever the firmware placed after it in the global object. The flaw sits on the pre-encryption path, the first one an unauthenticated attacker reaches.

### Timeline

| DATE       | EVENT                                   |
|------------|-----------------------------------------|
| 2026-05-12 | ArduinoBLE 2.0.2 released (fix 1460e1a) |
| 2026-06-03 | Advisory GHSA-77v6-cw9f-9whg published  |
| 2026-09-11 | CVE-2026-47773 record published         |

### Mitigation

Update to ArduinoBLE 2.0.2, which adds the missing bounds check before the copy, rejecting an oversized pending write with `INSUFF_RESOURCES` instead of copying it. The fix is four lines (commit 1460e1a, PR #431).

## 02 Technical analysis

### Root cause

The stash buffer is a fixed 64 bytes, a plain member of the global `ATTClass` object. Inside `writeReqOrCmd()`, the length and pointer of the value come straight from the incoming PDU, and the hold-for-encryption branch fills `writeBuffer` by hand, with no check that the value fits:

• `src/utility/ATT.cpp, ATTClass::writeReqOrCmd()` (vulnerable, <= 2.0.1, abridged)

```
uint8_t valueLength = dlen - sizeof(handle); // attacker-controlled ATT PDU length
uint8_t* value      = &data[sizeof(handle)];
if (holdResponse) { // BLEEncryption required, peer not yet encrypted
    writeBufferSize = 0;
    memcpy(writeBuffer, &handle, 2);           writeBufferSize += 2;
    writeBuffer[writeBufferSize++] = _peers[i].addressType;
    memcpy(&writeBuffer[writeBufferSize], _peers[i].address, 6); writeBufferSize += 6;
    writeBuffer[writeBufferSize] = valueLength;   writeBufferSize += 1;
    // no check that writeBufferSize + valueLength fits in writeBuffer
    memcpy(&writeBuffer[writeBufferSize], value, valueLength); // out-of-bounds write
}
```

The header written before the value is 10 bytes (handle 2, addressType 1, address 6, length 1), leaving 54 bytes of headroom. But `valueLength` is derived from the incoming ATT PDU length, which ArduinoBLE reassembles into a 255-byte L2CAP buffer, so an attacker can push it up to roughly 248 bytes. The handler never checks it against the buffer, nor even against the negotiated MTU, so it fires at the default 23-byte MTU. A single oversized write runs the copy well off the end of the 64-byte buffer, with chosen content, into the rest of the global object.

### Impact

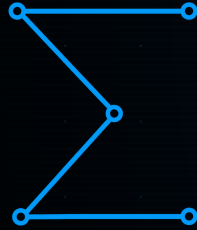
We did not build a proof-of-concept. What follows is reasoned from the source above, not demonstrated exploitation. The primitive is a remote, unauthenticated, no-interaction out-of-bounds write with attacker-chosen content, starting at a fixed offset inside the 64-byte buffer:

| TIER                              | ASSESSMENT                                                                                    |
|-----------------------------------|-----------------------------------------------------------------------------------------------|
| remote DoS (expected)             | portable: one oversized write corrupts the live BLE stack, crash or wedged radio until reboot |
| state corruption (conditional)    | chosen content overwrites ATT bookkeeping, peer table, pointers; depends on build layout      |
| control-flow hijack (speculative) | only if a reachable function pointer sits in range on a given image; none identified          |

The portable, defensible impact is a remote, pre-authentication denial of service. Anything beyond that is target-specific and would need a per-firmware exploit. The CVE record scores it CVSS 4.0 7.2 (High); Arduino's advisory lists CVSS 4.0 6.1 (Medium, AC:H).

### Reachability

Everything the attacker needs is on the pre-encryption path, so no pairing, bonding or key is involved. The target exposes at least one writable characteristic carrying the `BLEEncryption` permission; the attacker connects over BLE and sends a write to that handle with a value longer than the 54 bytes of headroom, fragmented over L2CAP if needed. No MTU exchange is required, and no user interaction.



# ESPILON

SECURITY RESEARCH

Embedded, IoT and OT security research

[espilon.net](https://espilon.net)

[eun0us@espilon.net](mailto:eun0us@espilon.net)

Association loi 1901 • France