Heap out-of-bounds read in SNMP supply-level polling leaks stack memory to authenticated users

Moderate **zdohnal** published **GHSA-6wpw-g8g6-wvrv** last week

Package

cups

Affected versions

<= 2.4.16

Patched versions

2.4.17

Description

Summary

A network-adjacent attacker can send a crafted SNMP response to the CUPS SNMP backend that causes an out-of-bounds read of up to 176 bytes past a stack buffer. The leaked memory is converted from UTF-16 to UTF-8 and stored as printer supply description strings, which are subsequently visible to authenticated users via IPP `Get-Printer-Attributes` responses and the CUPS web interface.

Details

Root Cause

In `cups/snmp.c`, function `asn1_decode_snmp()`, line 1017:

```
case CUPS_ASN1_OCTET_STRING :
case CUPS_ASN1_BIT_STRING :
case CUPS_ASN1_HEX_STRING :
    packet->object_value.string.num_bytes = length;  // Set to raw ASN.1 length
    asn1_get_string(&bufptr, bufend, length,
                    (char *)packet->object_value.string.bytes,
                    sizeof(packet->object_value.string.bytes));
    break;
```

The `num_bytes` field is set to the raw ASN.1 length value **before** `asn1_get_string()` is called. `asn1_get_string()` clamps the actual copy to `min(length, sizeof(string.bytes) - 1)` = 1023 bytes, but `num_bytes` is never updated to reflect the clamped value.

In `backend/snmp-supplies.c`, function `backend_walk_cb()`, lines 897-900:

```
case CUPS_TC_csUTF16BE :
case CUPS_TC_csUTF16LE :
    utf16_to_utf8((cups_utf8_t *)supplies[i - 1].name,
                  packet->object_value.string.bytes,
                  packet->object_value.string.num_bytes,  // Uses inflated value
                  sizeof(supplies[0].name), charset == CUPS_TC_csUTF16LE);
```

When the charset is UTF-16, `utf16_to_utf8()` reads `num_bytes` bytes from the 1024-byte `string.bytes` buffer, reading up to 176 bytes past the buffer into adjacent stack memory (the `num_bytes` field itself, struct padding, and stack variables from the calling function).

Data Flow of Leaked Memory

1. `utf16_to_utf8()` interprets OOB bytes as UTF-16 code points, converts to UTF-8
2. Result stored in `supplies[i-1].name` (supply description string)
3. `backend_init_supplies()` outputs `ATTR: marker-names=...` on stderr
4. `cupsd` parses this and sets `printer-supply-description` IPP attribute
5. Leaked data (garbled as UTF-8) visible via:
   - IPP `Get-Printer-Attributes` response
   - CUPS web UI supply names at `https://localhost:631/printers/<name>`

Struct Layout

```
cups_snmp_t (2344 bytes total):
  string.bytes:     offset 1312, size 1024
  string.num_bytes: offset 2336, size 4
  struct end:       offset 2344

OOB read with num_bytes=1200:
  Reads bytes 1312..2511 (1200 bytes)
  = 176 bytes past string.bytes[]
  = 168 bytes past end of cups_snmp_t struct (into stack)
```

PoC

Prerequisites

- A CUPS system with an SNMP-discovered printer on the local network

- Attacker on the same LAN segment (same broadcast domain)

Exploit

The attacker listens for SNMP GetRequest broadcasts from the CUPS backend and responds with a crafted GetResponse containing:

1. `prtLocalizationCharacterSet` = 1015 (`CUPS_TC_csUTF16BE`) -- sets UTF-16 charset
2. `prtMarkerSuppliesDescription` with ASN.1 OCTET_STRING length field = 1200 (but only 1023 bytes of actual data follow)

```
#!/usr/bin/env python3
"""SNMP OOB Read PoC - responds to CUPS supply queries with inflated OCTET_STRING"
import socket, struct

def build_snmp_response(request_id, oid_suffix, length=1200):
    """Build GetResponse with OCTET_STRING claiming 'length' bytes"""
    # ASN.1 OCTET_STRING with 2-byte length encoding
    value = b'\x04\x82' + struct.pack('>H', length) + b'\x41' * min(length, 1023)
    # ... (full packet construction with OID, request-id, etc.)

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(('0.0.0.0', 161))
while True:
    data, addr = sock.recvfrom(1472)
    # Parse request, respond with crafted prtMarkerSuppliesDescription
    sock.sendto(build_response(data), addr)
```

Full working PoC with SNMP dialog implementation available at:

- `poc_snmp_e2e.c` -- C program with full ASan crash
- `poc_snmp_oob_read.py` -- Python SNMP responder for live testing

ASan Output

```
==66078==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x78e6c2300b08
READ of size 1 at 0x78e6c2300b08 thread T0
    #0 test_utf16_to_utf8 poc_snmp_e2e.c:59
    #1 main poc_snmp_e2e.c:485

[480, 2824) 'packet' <== Memory access at offset 2824 overflows this variable
```

Impact

Information disclosure. A network-adjacent attacker without authentication can cause the CUPS SNMP backend to read 176 bytes of stack memory past the `cups_snmp_t` struct. The leaked memory (which may contain return addresses, local variables, or pointers) is garbled through UTF-16-to-UTF-8 conversion and stored as printer supply names. These names are visible to authenticated users querying printer attributes.

The OOB read can also cause crashes in hardened builds (ASan/Valgrind), constituting a denial-of-service of the supply monitoring function.

Who is impacted: Any CUPS deployment where:

- SNMP printer discovery is enabled (default: yes, via `snmp.conf`)
- An attacker is on the same LAN segment as the CUPS host
- The CUPS host has at least one SNMP-discovered printer

Affected Products

- **Ecosystem:** OpenPrinting CUPS
- **Package:** cups
- **Affected versions:** All versions with SNMP backend support
- **Patched versions:** None (unpatched)

Severity

CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:L

Score: 5.4 (Medium)

- Attack Vector: Adjacent Network (SNMP uses UDP broadcast on LAN)
- Attack Complexity: Low (respond to broadcast query)
- Privileges Required: None
- User Interaction: None (auto-triggered during printer discovery/supply polling)
- Confidentiality: Low (stack memory leaked, garbled by UTF-16 conversion)
- Availability: Low (can crash supply monitoring in hardened builds)

Weaknesses

- **CWE-125**: Out-of-bounds Read
- **CWE-200**: Exposure of Sensitive Information to an Unauthorized Actor

Suggested Fix

In `cups/snmp.c`, after the `asn1_get_string()` call at line 1018-1020, clamp `num_bytes`:

```
packet->object_value.string.num_bytes = length;
asn1_get_string(&bufptr, bufend, length,
                (char *)packet->object_value.string.bytes,
                sizeof(packet->object_value.string.bytes));
// Clamp num_bytes to actual buffer size
if (packet->object_value.string.num_bytes >= sizeof(packet->object_value.string.bytes))
    packet->object_value.string.num_bytes = sizeof(packet->object_value.string.bytes) -
```

Fixes:
master `d7fe0f5` Limit num_bytes for SNMP string values.

2.4.x `b7c2525` Limit num_bytes for SNMP string values.

Severity

Moderate 4.3 / 10

CVSS v3 base metrics

Attack vector	Adjacent
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchanged
Confidentiality	Low
Integrity	None
Availability	None

Learn more about base metrics

CVSS:3.1/AV:A/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

CVE ID

CVE-2026-41079

Weaknesses

- CWE-125
- CWE-200

Credits

Tomer-PL Reporter