

# [UNIX] Libtiff Image Decoder Parsing Flaws

**Source:** <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-10/0064.html>

---

**From:** SecuriTeam ([support\\_at\\_securiteam.com](mailto:support_at_securiteam.com))

**Date:** 10/18/04

To: [list@securiteam.com](mailto:list@securiteam.com)

Date: 18 Oct 2004 11:50:12 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

-----

Libtiff Image Decoder Parsing Flaws

---

## SUMMARY

<<http://www.libtiff.org/>> libtiff is an encoder / decoder for the TIFF image format. The TIFF image format is an incredibly rich format featuring multiple possible encodings and formats. The encodings include JPEG, LZW, ZIP, log-based encodings and many more – from many different companies such as NeXT, SGI and Pixar.

Several coding flaws were discovered in an inspection of libtiff's code. These flaws allow compromising the account used to browse the malicious TIFF files.

## DETAILS

Vulnerable Systems:

\* Libtiff version 3.8.1, possibly prior

CVE Information:

<<http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0803>>

CAN-2004-0803

Heap-based overflow during RLE decoding

## Securiteam: [UNIX] Libtiff Image Decoder Parsing Flaws

The following code is taken from `tif_next.c`:

```
off = (bp[0] * 256) + bp[1];
n = (bp[2] * 256) + bp[3];
if (cc < 4+n)
    goto bad;
_TIFFmemcpy(row+off, bp+4, n);
```

Here, `off` and `n` are arbitrary values from the TIFF. Bounds checking is performed on the data source buffer, but not the data destination buffer. A proof of concept TIFF file that can be used to demonstrate the problem can be found at [http://scary.beasts.org/misc/bad\\_next.tiff](http://scary.beasts.org/misc/bad_next.tiff)

Note: Memory is subtly corrupted and might not trigger an immediate crash.

### Heap-based overflow during RLE decoding

The following code is taken from `tif_thunder.c`:

```
case THUNDER_RUN: /* pixel run */
/*
 * Replicate the last pixel n times,
 * where n is the lower-order 6 bits.
 */
if (npixels & 1) {
    op[0] |= lastpixel;
    lastpixel = *op++; npixels++; n--;
} else
    lastpixel |= lastpixel << 4;
npixels += n;
for (; n > 0; n -= 2)
    *op++ = (tdataval_t) lastpixel;
```

Here, `n` is an arbitrary value from the TIFF, and is used to drive a copy count into the output without bounds checking. A demonstration TIFF can be found at [http://scary.beasts.org/misc/bad\\_thunder.tiff](http://scary.beasts.org/misc/bad_thunder.tiff)

Note: Memory is subtly corrupted and might not trigger an immediate crash.

### Possible overflow in `tif_luv.c`

A TIFF file to confirm this has not been crafted – but it appears that there may be heap-based overflows when doing RLE decoding:

```
for (shft = 2*8; (shft -= 8) >= 0; ) {
    for (i = 0; i < npixels && cc > 0; )
        if (*bp >= 128) { /* run */
            rc = *bp++ + (2-128);
            b = (int16)(*bp++ << shft);
            cc -= 2;
            while (rc-->0)
                [*] tp[i++] |= b;
        } else { /* non-run */
```

## Securiteam: [UNIX] Libtiff Image Decoder Parsing Flaws

```
rc = *bp++; /* nul is noop */
while (--cc && rc--)
[*] tp[i++] |= (int16)*bp++ << shft;
```

Lines marked with a [\*] exhibit a suspicious lack of checking on the size of the output buffer. Note that tif\_luv.c contains multiple versions of the RLE decoder for different image depths, and all look similarly dangerous.

### ADDITIONAL INFORMATION

The information has been provided by <mailto:chris@scary.beasts.org>  
Chris Evans.

=====

This bulletin is sent to members of the SecuriTeam mailing list.  
To unsubscribe from the list, send mail with an empty subject line and body to:  
list-unsubscribe@securiteam.com  
In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====  
=====

### DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.  
In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.