Message-ID: <20260403032437.GB23840@brightrain.aerifal.cx>
Date: Thu, 2 Apr 2026 23:24:37 -0400
From: Rich Felker <dalias@...c.org>
To: oss-security@...ts.openwall.com
Subject: Re: [libc musl] - Algorithmic complexity DoS in iconv
 GB18030 decoder

On Thu, Apr 02, 2026 at 08:45:57PM -0400, Rich Felker wrote:
> On Thu, Apr 02, 2026 at 10:27:38PM +0200, Jens Jarl Nestén Hansen-Nord wrote:
> > ==
> > libc musl Security Advisory: April 2, 2026
> > ==
> > Description:
> > The GB18030 4-byte decoder in musl libc's iconv() implementation
> > contains a gap-skipping loop that performs a full linear scan of the
> > gb18030126 lookup table (23,940 entries) on each iteration of an
> > outer loop whose iteration count is input-dependent. For 4-byte
> > sequences whose linear index falls just below the dense CJK Unified
> > Ideographs range, the outer loop executes approximately 20,905
> > times, resulting in approximately 500 million comparisons per input
> > character.
> > Classification:
> > Inefficient Algorithmic Complexity (CWE-407)
> > Impact:
> > This allows a remote attacker to cause denial of service via CPU
> > exhaustion by sending a crafted GB18030 payload to any network
> > service that uses musl's iconv() for character encoding conversion.
> > Measured on musl 1.2.6 and 1.2.5: a single 4-byte input character
> > (bytes 0x82 0x35 0x8F 0x33) takes approximately 260ms to decode,
> > compared to approximately 13 microseconds for a benign character — a
> > 19,000x slowdown. A payload of 40kB will take ~43 minutes to decode.
> >
> > Versions affected:
> > musl 0.8.0 to 1.2.6
> >
> > Status:
> > The issue has been confirmed and fixed by maintainer, Rich Felker.
> > A CVE has been requested and is pending assignment.
> >
> > Reported by:
> > Jens Jarl Nestén Hansen-Nord
> >
> > Upstream fix:
> > Iconv-gb18030-fix.diff
> >
> > diff --git a/src/locale/iconv.c b/src/locale/iconv.c
> > index 52178950..e559aa4c 100644
> > --- a/src/locale/iconv.c
> > +++ b/src/locale/iconv.c
> > @@ -74,6 +74,10 @@ static const unsigned short gb18030[126][190] = {
> > #include "gb18030.h"
> > };
> >
> > +static const unsigned short gb18030utf[][2] = {
> > +#include "gb18030utf.h"
> > +};
> > +
> > static const unsigned short big5[89][157] = {
> > #include "big5.h"
> > };
> > @@ -224,6 +228,8 @@ static unsigned uni_to_jis(unsigned c)
> >   }
> > }
> >
> > +#define countof(a) (sizeof (a) / sizeof *(a))

```
> > +
> >  size_t iconv(iconv_t cd, char **restrict in, size_t *restrict inb, char **restrict out, size_t
*restrict outb)
> >  {
> >     size_t x=0;
> > @@ -430,16 +436,14 @@ size_t iconv(iconv_t cd, char **restrict in, size_t *restrict inb, char **restri
> >                  d = *((unsigned char *)*in + 3);
> >                  if (d-'0'>9) goto ilseq;
> >                  c += d-'0';
> > -                c += 128;
> > -                for (d=0; d<=c; ) {
> > -                    k = 0;
> > -                    for (int i=0; i<126; i++)
> > -                        for (int j=0; j<190; j++)
> > -                            if (gb18030[i][j]-d <= c-d)
> > -                                k++;
> > -                    d = c+1;
> > -                    c += k;
> > +                for (int i=0; i<countof(gb18030utf); i++) {
> > +                    if (c<gb18030utf[i][1]) {
> > +                        c += gb18030utf[i][0];
> > +                        break;
> > +                    }
> > +                    c -= gb18030utf[i][1];
> >                  }
> > +                c += 0x10000;
> >                  break;
> >              }
> >              d -= 0x40;
> >
> >
> >
> >
>
> The above patch was a proposal for testing. It should mitigate the
> extreme slowness for characters encoded in GB18030's UTF, but it does
> not work correctly and has not been confirmed not to have other
> problems. I will follow up with a correct patch.

The attached patch has now been tested to work. Compared to the
previous version above, it corrects one value in the table, an
erroneous += 0x10000 above, and missing logic for characters U+10000
and up.

Rich
```

View attachment "0001-fix-pathological-slowness-incorrect-mappings-in-icon.patch" **of type** "text/plain" **(7502 bytes)**