jmacd / **xdelta** Public

Code | Issues 113 | Pull requests 5 | Actions | Projects | Security | Insights

Add appheader tests; fix buffer overflow in main_get_appheader

Browse files

release3_1_apl
v3.1.0 ··· v3.0.9

jmacd committed on Oct 12, 2014 — 1 parent 7b6ff92 — commit ef93ff74203e030073b898c05e8b4860b5d09ef2

Showing **2 changed files** with **108 additions** and **28 deletions**.

Unified | Split

5 xdelta3/xdelta3-main.h

```
2810 2810
2811 2811     if (appheadsz > 0)
2812 2812       {
     2813 +       const int kMaxArgs = 4;
2813 2814         char *start = (char*)apphead;
2814 2815         char *slash;
2815 2816         int   place = 0;
2816      -       char *parsed[4];
     2817 +       char *parsed[kMaxArgs];
2817 2818
2818 2819         memset (parsed, 0, sizeof (parsed));
2819 2820
2820      -       while ((slash = strchr (start, '/')) != NULL)
     2821 +       while ((slash = strchr (start, '/')) != NULL && place < (kMaxArgs-1))
2821 2822       {
2822 2823         *slash = 0;
```

```
2823  2824        parsed[place++] = start;
```

131 xdelta3/xdelta3-test.h

```
...   ...   @@ -1,5 +1,5 @@
1     1       /* xdelta 3 - delta compression tools and library Copyright (C) 2001,
2           - * 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012.
      2     + * 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012.
3     3       * Joshua P. MacDonald
4     4       *
5     5       *  This program is free software; you can redistribute it and/or modify
54    54        /* only MSBs of the array mt[].                          */
55    55        /* 2002/01/09 modified by Makoto Matsumoto               */
56    56        mt->mt_buffer_[i] =
57          -   (1812433253UL * (mt->mt_buffer_[i-1] ^
      57    +   (1812433253UL * (mt->mt_buffer_[i-1] ^
58    58                (mt->mt_buffer_[i-1] >> 30)) + i);
59    59      }
60    60    }
69    69        int kk;
70    70
71    71        for (kk = 0; kk < MT_LEN - MT_IA; kk++) {
72          -       y = (mt->mt_buffer_[kk] & UPPER_MASK) |
      72    +       y = (mt->mt_buffer_[kk] & UPPER_MASK) |
73    73      (mt->mt_buffer_[kk + 1] & LOWER_MASK);
74          -       mt->mt_buffer_[kk] = mt->mt_buffer_[kk + MT_IA] ^
      74    +       mt->mt_buffer_[kk] = mt->mt_buffer_[kk + MT_IA] ^
75    75      (y >> 1) ^ mag01[y & 0x1UL];
76    76        }
77    77        for (;kk < MT_LEN - 1; kk++) {
78          -       y = (mt->mt_buffer_[kk] & UPPER_MASK) |
      78    +       y = (mt->mt_buffer_[kk] & UPPER_MASK) |
79    79      (mt->mt_buffer_[kk + 1] & LOWER_MASK);
80          -       mt->mt_buffer_[kk] = mt->mt_buffer_[kk + (MT_IA - MT_LEN)] ^
```

```
                80 +       mt->mt_buffer_[kk] = mt->mt_buffer_[kk + (MT_IA - MT_LEN)] ^
81              81      (y >> 1) ^ mag01[y & 0x1UL];
82              82        }
83              -       y = (mt->mt_buffer_[MT_LEN - 1] & UPPER_MASK) |
                83 +     y = (mt->mt_buffer_[MT_LEN - 1] & UPPER_MASK) |
84              84          (mt->mt_buffer_[0] & LOWER_MASK);
85              -       mt->mt_buffer_[MT_LEN - 1] = mt->mt_buffer_[MT_IA - 1] ^
                85 +     mt->mt_buffer_[MT_LEN - 1] = mt->mt_buffer_[MT_IA - 1] ^
86              86          (y >> 1) ^ mag01[y & 0x1UL];
87              87        mt->mt_index_ = 0;
88              88      }
166             166     {
167             167       stream->msg = "abnormal command termination";
168             168     }
169             -         return XD3_INTERNAL;
                169 +       return ret;
170             170       }
171             171     return 0;
172             172   }
257             257   static int
258             258   test_make_inputs (xd3_stream *stream, xoff_t *ss_out, xoff_t *ts_out)
259             259   {
260             -   usize_t ts = (mt_random (&static_mtrand) % TEST_FILE_MEAN) + TEST_FILE_MEAN / 2;
261             -   usize_t ss = (mt_random (&static_mtrand) % TEST_FILE_MEAN) + TEST_FILE_MEAN / 2;
                260 +   usize_t ts = (mt_random (&static_mtrand) % TEST_FILE_MEAN) +
                261 +     TEST_FILE_MEAN / 2;
                262 +   usize_t ss = (mt_random (&static_mtrand) % TEST_FILE_MEAN) +
                263 +     TEST_FILE_MEAN / 2;
262             264     uint8_t *buf = (uint8_t*) malloc (ts + ss), *sbuf = buf, *tbuf = buf + ss;
263             265     usize_t sadd = 0, sadd_max = (usize_t)(ss * TEST_ADD_RATIO);
264             266     FILE  *tf = NULL, *sf = NULL;
409             411     {
410             412       if (obuf[i] != rbuf[i])
411             413         {
```

```
412       -          XPR(NT "byte %u (read %u @ %"Q"u) %d != %d\n",
          414 +      XPR(NT "byte %u (read %u @ %"Q"u) %d != %d\n",
413       415            (int)i, (int)oc, offset, obuf[i], rbuf[i]);
414       416            diffs++;
415       417            return XD3_INTERNAL;
421       423
422       424        fclose (orig);
423       425        fclose (recons);
424       -        if (diffs != 0)
          426 +    if (diffs != 0)
425       427          {
426       428      return XD3_INTERNAL;
427       429          }
428       430        return 0;
429       431    }
430       432
431       433    static int
432       -  test_save_copy (const char *origname)
          434 +  test_copy_to (const char *from, const char *to)
433       435    {
434       436      char buf[TESTBUFSIZE];
435       437      int ret;
436       438
437       -    snprintf_func (buf, TESTBUFSIZE, "cp -f %s %s", origname, TEST_COPY_FILE);
          439 +    snprintf_func (buf, TESTBUFSIZE, "cp -f %s %s", from, to);
438       440
439       441      if ((ret = system (buf)) != 0)
440       442        {
444       446      return 0;
445       447    }
446       448
          449 + static int
          450 + test_save_copy (const char *origname)
          451 + {
```

```
                452 +   return test_copy_to(origname, TEST_COPY_FILE);
                453 + }
                454 +
447             455   static int
448             456   test_file_size (const char* file, xoff_t *size)
449             457   {
499             507     inp = buf->base;
500             508     max = buf->base + buf->next - trunto;
501             509
502             -   if ((ret = xd3_read_uint32_t (stream, & inp, max, & rval)) !=
                510 +   if ((ret = xd3_read_uint32_t (stream, & inp, max, & rval)) !=
503             511         XD3_INVALID_INPUT ||
504             512         !MSG_IS (msg))
505             513       {
1654            1662    if ((buf = (uint8_t*) malloc (TWO_MEGS_AND_DELTA)) == NULL) { return ENOMEM; }
1655            1663
1656            1664    memset (buf, 0, TWO_MEGS_AND_DELTA);
1657            -   for (i = 0; i < (2 << 20); i += 256)
                1665 +   for (i = 0; i < (2 << 20); i += 256)
1658            1666       {
1659            1667         int j;
1660            1668         int off = mt_random(& static_mtrand) % 10;
1661            -        for (j = 0; j < 256; j++)
                1669 +        for (j = 0; j < 256; j++)
1662            1670     {
1663            1671       buf[i + j] = j + off;
1664            1672     }
1683            1691       }
1684            1692
1685            1693    /* Test transfer of exactly 32bits worth of data. */
1686            -   if ((ret = test_streaming (stream,
1687            -                 buf,
1688            -                 buf + (1 << 20),
1689            -                 buf + (2 << 20),
```

```
1690      -                 1 << 12)))
     1694 +   if ((ret = test_streaming (stream,
     1695 +                 buf,
     1696 +                 buf + (1 << 20),
     1697 +                 buf + (2 << 20),
     1698 +                 1 << 12)))
1691 1699       {
1692 1700         goto fail;
1693 1701       }
1889 1897       }
1890 1898
1891 1899     /* First encode */
1892      -   snprintf_func (ecmd, TESTBUFSIZE, "%s %s -f %s %s %s %s %s %s %s",
     1900 +   snprintf_func (ecmd, TESTBUFSIZE, "%s %s -f %s %s %s %s %s %s %s",
1893 1901         program_name, test_softcfg_str,
1894 1902         has_adler32 ? "" : "-n ",
1895 1903         has_apphead ? "-A=encode_apphead " : "-A= ",
1910 1918     snprintf_func (recmd, TESTBUFSIZE,
1911 1919         "%s recode %s -f %s %s %s %s %s", program_name, test_softcfg_str,
1912 1920         recoded_adler32 ? "" : "-n ",
1913      -       !change_apphead ? "" :
     1921 +       !change_apphead ? "" :
1914 1922             (recoded_apphead ? "-A=recode_apphead " : "-A= "),
1915 1923         recoded_secondary ? "-S djw " : "-S none ",
1916 1924         TEST_DELTA_FILE,
2361 2369     return 0;
2362 2370   }
2363 2371
     2372 + /* This tests that the default appheader works */
     2373 + static int
     2374 + test_appheader (xd3_stream *stream, int ignore)
     2375 + {
     2376 +   int i;
     2377 +   int ret;
```

```
2378 +   char buf[TESTBUFSIZE];
2379 +   char bogus[TESTBUFSIZE];
2380 +   xoff_t ssize, tsize;
2381 +   test_setup ();
2382 +
2383 +   if ((ret = test_make_inputs (stream, &ssize, &tsize))) { return ret; }
2384 +
2385 +   snprintf_func (buf, TESTBUFSIZE, "%s -q -f -e -s %s %s %s", program_name,
2386 +         TEST_SOURCE_FILE, TEST_TARGET_FILE, TEST_DELTA_FILE);
2387 +   if ((ret = do_cmd (stream, buf))) { return ret; }
2388 +
2389 +   if ((ret = test_copy_to (program_name, TEST_RECON2_FILE))) { return ret; }
2390 +
2391 +   snprintf_func (buf, TESTBUFSIZE, "chmod 0700 %s", TEST_RECON2_FILE);
2392 +   if ((ret = do_cmd (stream, buf))) { return ret; }
2393 +
2394 +   if ((ret = test_save_copy (TEST_TARGET_FILE))) { return ret; }
2395 +   if ((ret = test_copy_to (TEST_SOURCE_FILE, TEST_TARGET_FILE))) { return ret; }
2396 +
2397 +   if ((ret = test_compare_files (TEST_TARGET_FILE, TEST_COPY_FILE)) == 0)
2398 +     {
2399 +       return XD3_INVALID;  // I.e., files are different!
2400 +     }
2401 +
2402 +   // Test that the target file is restored.
2403 +   snprintf_func (buf, TESTBUFSIZE, "(cd /tmp && %s -q -f -d %s)",
2404 +         TEST_RECON2_FILE,
2405 +         TEST_DELTA_FILE);
2406 +   if ((ret = do_cmd (stream, buf))) { return ret; }
2407 +
2408 +   if ((ret = test_compare_files (TEST_TARGET_FILE, TEST_COPY_FILE)) != 0)
2409 +     {
2410 +       return ret;
2411 +     }
```

```
2412 +
2413 +   // Test a malicious string w/ entries > 4 in the appheader by having
2414 +   // the encoder write it:
2415 +   for (i = 0; i < TESTBUFSIZE / 4; ++i)
2416 +     {
2417 +       bogus[2*i] = 'G';
2418 +       bogus[2*i+1] = '/';
2419 +     }
2420 +   bogus[TESTBUFSIZE/2-1] = 0;
2421 +
2422 +   snprintf_func (buf, TESTBUFSIZE,
2423 +         "%s -q -f -A=%s -e -s %s %s %s", program_name, bogus,
2424 +         TEST_SOURCE_FILE, TEST_TARGET_FILE, TEST_DELTA_FILE);
2425 +   if ((ret = do_cmd (stream, buf))) { return ret; }
2426 +   // Then read it:
2427 +   snprintf_func (buf, TESTBUFSIZE, "(cd /tmp && %s -q -f -d %s)",
2428 +         TEST_RECON2_FILE,
2429 +         TEST_DELTA_FILE);
2430 +   if ((ret = do_cmd (stream, buf)) == 0)
2431 +     {
2432 +       return XD3_INVALID;  // Impossible
2433 +     }
2434 +   if (!WIFEXITED(ret))
2435 +     {
2436 +       return XD3_INVALID;  // Must have crashed!
2437 +     }
2438 +
2439 +   return 0;
2440 + }
2441 +
2364 2442   /*********************************************************************
2365 2443    Source identical optimization
2366 2444    *********************************************************************/
2603 2681         default: CHECK(0);
```

```
2604  2682          }
2605  2683
2606        -       snprintf_func (rptr, rbuf+TESTBUFSIZE-rptr, "%d/%d",
      2684  +       snprintf_func (rptr, rbuf+TESTBUFSIZE-rptr, "%d/%d",
2607  2685                inst->pos, inst->size);
2608  2686        rptr += strlen (rptr);
2609  2687
2848  2926      DO_TEST (force_behavior, 0, 0);
2849  2927      DO_TEST (stdout_behavior, 0, 0);
2850  2928      DO_TEST (no_output, 0, 0);
      2929  +   DO_TEST (appheader, 0, 0);
2851  2930      DO_TEST (command_line_arguments, 0, 0);
2852  2931
2853  2932    #if EXTERNAL_COMPRESSION
```

0 comments on commit `ef93ff7`