gphoto / **libgphoto2** Public

Code | Issues 441 | Pull requests 16 | Actions | Projects | Wiki | Sec

# Commit 1817ece

**msmeissn** committed 3 days ago · ✓ 4 / 4

```
Fixed EOS ImageFormat/CustomFuncEx Parsers Lack Length Parameter

ptp_unpack_EOS_ImageFormat() and ptp_unpack_EOS_CustomFuncEx() accept
const unsigned char** data but no length/size parameter. They perform
unbounded reads via dtoh32o calls (up to 36 bytes for ImageFormat,
up to 1024 bytes for CustomFuncEx). Callers in ptp_unpack_EOS_events()
have xsize available but never pass it.

 CVE-2026-40333

Reported-By: Sebastián Alba <sebasjosue84@gmail.com>
```

master

1 parent c385b34 commit 1817ece

## 1 file changed +44 -9 lines changed

Top

Filter files...

camlibs/ptp2

ptp-pack.c

## 1 file changed +44 -9 lines changed

Search within code

camlibs/ptp2/ptp-pack.c

```
@@ -1459,7 +1459,7 @@ ptp_unpack_Canon_EOS_FE (PTPParams *params, const
unsigned char* data, unsigned
1459  1459
1460  1460
1461  1461   static inline uint16_t
1462       - ptp_unpack_EOS_ImageFormat (PTPParams* params, const unsigned char** data )
```

```
        1462 + ptp_unpack_EOS_ImageFormat (PTPParams* params, const unsigned char** data,
             +  unsigned int *size )
1463    1463   {
1464    1464       /*
1465    1465         EOS ImageFormat entries look are a sequence of u32 values:
             @@ -1503,30 +1503,57 @@ ptp_unpack_EOS_ImageFormat (PTPParams* params,
             const unsigned char** data )
1503    1503
1504    1504       const uint8_t* d = *data;
1505    1505       uint32_t offset = 0;
1506        - -    uint32_t n = dtoh32o (d, offset);
        1506 +     uint32_t n;
1507    1507       uint32_t l, t1, s1, c1, t2 = 0, s2 = 0, c2 = 0;
1508    1508
        1509 +     if (*size < sizeof(uint32_t)) {
        1510 +         ptp_debug (params, "parsing EOS ImageFormat property failed 1 (size
             %d)", *size);
        1511 +         return 0;
        1512 +     }
        1513 +     n = dtoh32o (d, offset);
        1514 +     *size -= sizeof(uint32_t);
        1515 +
1509    1516       if (n != 1 && n !=2) {
1510    1517           ptp_debug (params, "parsing EOS ImageFormat property failed (n != 1
             && n != 2: %d)", n);
1511    1518           return 0;
1512    1519       }
1513        - -
        1520 +     if (*size < sizeof(uint32_t)) {
        1521 +         ptp_debug (params, "parsing EOS ImageFormat property failed 2 (size
             %d)", *size);
        1522 +         return 0;
        1523 +     }
1514    1524       l = dtoh32o (d, offset);
        1525 +     *size -= sizeof(uint32_t);
        1526 +
1515    1527       if (l != 0x10) {
1516    1528           ptp_debug (params, "parsing EOS ImageFormat property failed (l !=
             0x10: 0x%x)", l);
1517    1529           return 0;
```

```
1518  1530          }
1519  1531
      1532  +       if (*size < 3*sizeof(uint32_t)) {
      1533  +           ptp_debug (params, "parsing EOS ImageFormat property failed 3 (size
                      %d)", *size);
      1534  +           return 0;
      1535  +       }
1520  1536          t1 = dtoh32o (d, offset);
1521  1537          s1 = dtoh32o (d, offset);
1522  1538          c1 = dtoh32o (d, offset);
      1539  +       *size -= 3*sizeof(uint32_t);
1523  1540
1524  1541          if (n == 2) {
      1542  +           if (*size < sizeof(uint32_t)) {
      1543  +               ptp_debug (params, "parsing EOS ImageFormat property failed 4
                          (size %d)", *size);
      1544  +               return 0;
      1545  +           }
1525  1546              l = dtoh32o (d, offset);
      1547  +           *size -= sizeof(uint32_t);
      1548  +
1526  1549              if (l != 0x10) {
1527  1550                  ptp_debug (params, "parsing EOS ImageFormat property failed (l !=
                          0x10: 0x%x)", l);
1528  1551                  return 0;
1529  1552              }
      1553  +           if (*size < 3*sizeof(uint32_t)) {
      1554  +               ptp_debug (params, "parsing EOS ImageFormat property failed 5
                          (size %d)", *size);
      1555  +               return 0;
      1556  +           }
1530  1557              t2 = dtoh32o (d, offset);
1531  1558              s2 = dtoh32o (d, offset);
1532  1559              c2 = dtoh32o (d, offset);
          ⇣           @@ -1695,12 +1722,20 @@ ptp_unpack_EOS_FocusInfoEx (PTPParams* params,
          ⇡           const unsigned char** data, uint3
1695  1722
1696  1723
1697  1724      static inline char*
1698      -   ptp_unpack_EOS_CustomFuncEx (PTPParams* params, const unsigned char** data )
```

```
                1725 + ptp_unpack_EOS_CustomFuncEx (PTPParams* params, const unsigned char** data,
                       unsigned int *size )
1699    1726    {
1700            -       uint32_t s = dtoh32a( *data );
1701            -       uint32_t n = s/4, i;
                1727 +     uint32_t s, n, i;
1702    1728            char    *str, *p;
1703    1729
                1730 +     if (*size < sizeof(uint32_t))
                1731 +         return strdup("bad length");
                1732 +
                1733 +     s = dtoh32a( *data );
                1734 +     n = s/4;
                1735 +
                1736 +     if (*size < 4+s)
                1737 +         return strdup("bad length");
                1738 +
1704    1739            if (s > 1024) {
1705    1740                ptp_debug (params, "customfuncex data is larger than 1k / %d...
                       unexpected?", s);
1706    1741                return strdup("bad length");
        ⋮       ⋮   @@ -2013,7 +2048,7 @@ ptp_unpack_EOS_events (PTPParams *params, const
        ⋮       ⋮   unsigned char* data, unsigned in
2013    2048                case PTP_DPC_CANON_EOS_ImageFormatExtHD:
2014    2049                    /* special handling of ImageFormat properties */
2015    2050                    for (j=0;j<dpd_count;j++) {
2016            -                   dpd->FORM.Enum.SupportedValue[j].u16 =
                       ptp_unpack_EOS_ImageFormat( params, &xdata );
                2051 +             dpd->FORM.Enum.SupportedValue[j].u16 =
                       ptp_unpack_EOS_ImageFormat( params, &xdata, &xsize );
2017    2052                        ptp_debug (params, INDENT "prop %x option[%2d] ==
                       0x%04x", dpc, j, dpd->FORM.Enum.SupportedValue[j].u16);
2018    2053                    }
2019    2054                    break;
        ⋮       ⋮   @@ -2318,15 +2353,15 @@ ptp_unpack_EOS_events (PTPParams *params, const
        ⋮       ⋮   unsigned char* data, unsigned in
2318    2353                case PTP_DPC_CANON_EOS_ImageFormatSD:
2319    2354                case PTP_DPC_CANON_EOS_ImageFormatExtHD:
2320    2355                    dpd->DataType = PTP_DTC_UINT16;
```

```
2321      -                 dpd->DefaultValue.u16 = ptp_unpack_EOS_ImageFormat( params, &xdata );
     2356 +                 dpd->DefaultValue.u16 = ptp_unpack_EOS_ImageFormat( params, &xdata, &xsize );
2322 2357                   dpd->CurrentValue.u16 = dpd->DefaultValue.u16;
2323 2358                   ptp_debug (params, INDENT "prop %x value == 0x%04x (u16)", dpc, dpd->CurrentValue.u16);
2324 2359                   break;
2325 2360               case PTP_DPC_CANON_EOS_CustomFuncEx:
2326 2361                   dpd->DataType = PTP_DTC_STR;
2327 2362                   free (dpd->DefaultValue.str);
2328 2363                   free (dpd->CurrentValue.str);
2329      -                 dpd->DefaultValue.str = ptp_unpack_EOS_CustomFuncEx( params, &xdata );
     2364 +                 dpd->DefaultValue.str = ptp_unpack_EOS_CustomFuncEx( params, &xdata, &xsize );
2330 2365                   dpd->CurrentValue.str = strdup( (char*)dpd->DefaultValue.str );
2331 2366                   ptp_debug (params, INDENT "prop %x value == %s", dpc, dpd->CurrentValue.str);
2332 2367                   break;
```

## Comments 0

Please sign in to comment.