Exiv2 - Bug #841

exiv2 crashes on input

25 Aug 2012 05:48 - Christian Grothoff

Status:	Closed	**Start date:**	25 Aug 2012
Priority:	Normal	**Due date:**	
Assignee:	Robin Mills	**% Done:**	0%
Category:	exif	**Estimated time:**	0.00 hour
Target version:	0.24		

Description

$ exiv2 -V
exiv2 0.23 001700 (32 bit build)
Copyright (C) 2004-2012 Andreas Huggel.
$ exiv2 exiv2crash.input
terminate called after throwing an instance of 'std::bad_alloc'
what(): std::bad_alloc
Aborted (core dumped)
grothoff@niko:~$ valgrind exiv2 exiv2crash.input
2932 Memcheck, a memory error detector
2932 Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al.
2932 Using Valgrind-3.8.0 and LibVEX; rerun with -h for copyright info
2932 Command: exiv2 exiv2crash.input
2932
2932 new/new[] failed and should throw an exception, but Valgrind
2932 cannot throw exceptions and so is aborting instead. Sorry.
2932 at 0x4025BD7: VALGRIND_PRINTF_BACKTRACE (valgrind.h:4550)
2932 by 0x4027C3F: operator new[](unsigned int) (vg_replace_malloc.c:357)
2932 by 0x4121577: Exiv2::PngImage::readMetadata() (in /usr/lib/libexiv2.so.12.0.0)
2932 by 0x80587FB: ??? (in /usr/bin/exiv2)
2932 by 0x43AEE45: (below main) (libc-start.c:228)

The input file was generated by zzuf (fuzzing tool) and is intentionally corrupt.

Associated revisions

Revision 2861 - 30 Aug 2012 21:30 - Andreas Huggel

#841: Do not read past the end of the data stream (file), added test case.

Revision 2862 - 30 Aug 2012 21:31 - Andreas Huggel

#841: Additional check to prevent issues with the sign when casting uint32_t to long.

History

#1 - 27 Aug 2012 07:38 - Robin Mills

- Status changed from New to Assigned

- Assignee set to Robin Mills

- Priority changed from High to Normal

I've built exiv2 from the trunk on Windows7/64 (32 bit DebugDLL build) and Linux (Ubuntu 12.04). Both report an error. Neither crashed.

1001 rmills@rmills-ubuntu:/Windows/Users/rmills/Desktop $ exiv2 crash.png
Exiv2 exception in print action for file crash.png:
Failed to read input data

1002 rmills@rmills-ubuntu:/Windows/Users/rmills/Desktop $ which exiv2
/usr/local/bin/exiv2

1003 rmills@rmills-ubuntu:/Windows/Users/rmills/Desktop $ exiv2 -v -V
(verbose Version is a new feature for 0.24 and reports additional build and loading of shared-libraries).
exiv2 0.23 001700 (32 bit build)
Copyright (C) 2004-2012 Andreas Huggel.

This program is free software; you can redistribute it and/or

...
Software Foundation, Inc., 51 Franklin Street, Fifth Floor,
Boston, MA 02110-1301 USA
GCC=4.6.3,DEBUG=0,DLL=0,Bits=4: library=/usr/local/lib/libz.so.1
GCC=4.6.3,DEBUG=0,DLL=0,Bits=4: library=/usr/local/lib/libexpat.so.1
GCC=4.6.3,DEBUG=0,DLL=0,Bits=4: library=/lib/i386-linux-gnu/libdl.so.2
GCC=4.6.3,DEBUG=0,DLL=0,Bits=4: library=/usr/lib/i386-linux-gnu/libstdc++.so.6
GCC=4.6.3,DEBUG=0,DLL=0,Bits=4: library=/lib/i386-linux-gnu/libm.so.6
GCC=4.6.3,DEBUG=0,DLL=0,Bits=4: library=/lib/i386-linux-gnu/libgcc_s.so.1
GCC=4.6.3,DEBUG=0,DLL=0,Bits=4: library=/lib/i386-linux-gnu/libc.so.6
GCC=4.6.3,DEBUG=0,DLL=0,Bits=4: library=/lib/ld-linux.so.2
builder=GCC=4.6.3,DEBUG=0,DLL=0,Bits=4:
GCC=4.6.3,DEBUG=0,DLL=0,Bits=4: executable=/usr/local/bin/exiv2
GCC=4.6.3,DEBUG=0,DLL=0,Bits=4: date="Aug 25 2012",time=23:27:21
1004 rmills@rmills-ubuntu:/Windows/Users/rmills/Desktop $ ^C

The file is not a valid PNG (as you've said) and results in a malloc for a great deal of memory.

I'm not convinced we have a case to answer here, however I'll listen if you put forward a case for additional investigation and remediation.

#2 - 27 Aug 2012 11:23 - Christian Grothoff

Ah, that's because your system has extensive amounts of memory. If I run with 9 GB, I get this:

$ exiv2 exiv2crash.input
Exiv2 exception in print action for file exiv2crash.input:
Failed to read input data
$ ulimit -v 1024
grothoff@spec:~$ exiv2 --version
Segmentation fault

#3 - 27 Aug 2012 12:59 - Robin Mills

Christian

Thanks for the update.

I can reproduce your crash at ulimit -v 1024*1024 (on bash 4.2,24 on Ubuntu 12.04)

$ for i in 4 3 2 1 ;do echo --- $i --- ; ulimit ~~v $(($i*1024*1024)) ; exiv2 crash.png ; done~~
~~--~~ 4 ---
Exiv2 exception in print action for file crash.png:
Failed to read input data
--- 3 ---
Exiv2 exception in print action for file crash.png:
Failed to read input data
--- 2 ---
Exiv2 exception in print action for file crash.png:
Failed to read input data
--- 1 ---
terminate called after throwing an instance of 'std::bad_alloc'
what(): std::bad_alloc
Aborted (core dumped)

I'm surprised we don't catch this exception. I'll update this issue when I've investigated a little more.

Observation:
ulimit -v 1024 cannot even run ls!

$ for i in 500 200 100 50 30 10 5 3 2 1 ; do echo ~~n $i '~~> ' ; ulimit -v $(($i*1024)) ; ls | wc -l ; done
500 -> 19
...
3 -> 19
2 -> ls: error while loading shared libraries: libc.so.6: failed to map segment from shared object: Cannot allocate memory
wc: error while loading shared libraries: libc.so.6: failed to map segment from shared object: Cannot allocate memory
1 -> ls: error while loading shared libraries: libc.so.6: failed to map segment from shared object: Cannot allocate memory
wc: error while loading shared libraries: libc.so.6: failed to map segment from shared object: Cannot allocate memory
$

#4 - 27 Aug 2012 20:41 - Andreas Huggel

Thanks for looking into this, Robin.
I'm not so concerned that exiv2 dies when there is not enough memory. There is not much else we can do, other than maybe die prettier. (That's why

std::bad_alloc is not caught in the exiv2 CLI.)
But can we recognize that something is fishy and avoid attempting to allocate an unreasonable amount of memory? We do that quite extensively for JPEGs, but have much less experience with PNGs.
Besides, I don't seem to be able to download the attachment. Tried on Ubuntu with Chrome and Firefox and Windows with Chrome and Explorer. How did you get hold of it? Could you send me the attachment by email please?

Andreas

#5 - 27 Aug 2012 20:44 - Andreas Huggel

> Besides, I don't seem to be able to download the attachment.

Looks like some proxy or firewall issue here. Will try from home later.

#6 - 28 Aug 2012 00:38 - Christian Grothoff

Andreas, you're missing a key point: this is not just exiv2 crashing, but also libexiv2 crashes when it encounters this file. And libexiv2 is used by GNU libextractor, and GNU libextractor is not supposed to crash ever. Also, I don't think any library should ever contain code that just crashes -- especially if it could possibly be avoided -- after all, your lib might be used in some kind of GUI app and then all the user sees is his browser / e-mail client / etc. crashing. Very bad. So please treat this as a real issue.

#7 - 28 Aug 2012 01:38 - Andreas Huggel

Christian, I understand this is happening in the library. Also, I agree of course the library is not supposed to crash. So yes, this is a real issue.

The point I was trying to make was that we should prevent the PNG parser from trying to allocate unreasonable amounts of memory (using sanity checks where appropriate) in the first place as opposed to trying to catch and deal with an std::bad_alloc exception somewhere in the library.

And once we're doing that correctly, i.e., libexiv2 only allocates what makes sense and there is still simply not enough memory for the attempted operation, then there is nothing the library can do about it. In this case libexiv2 typically throws exceptions that the application has to deal with. So we'll just let the std::bad_alloc exception escape, there is no point catching that and translating it to an Exiv2 exception.

So yes, you can say exiv2 (the command line tool) should catch the std::bad_alloc exception and then die more gracefully, but that's secondary, it doesn't help you in libextractor or any app that uses libextractor.

Andreas

#8 - 28 Aug 2012 08:34 - Robin Mills

I think we're talking about two different things:

1) Running with no memory (ulimit -v 1024)
2) Files with corrupt data that cause the library to crash (by attempting to allocate too much memory)

I don't think we should bother with 1). It's a contrived case. Nothing runs when there's no memory. Failure is guaranteed. We're only discussing the error message on failure.

Case (2) is interesting as corrupt files can be used to attack an application. How about we add an integer to the library "maxMemoryMalloc" and ensure that's respected when "new" is needed? We can profile new() to determine a reasonable default.

#9 - 28 Aug 2012 09:30 - Christian Grothoff

Right, I had not intended to run with 1k -- ulimit -v works in kb on my system, so I gave it 1 MB, not 1kb. The issue was simply that I could easily reproduce the bug on my notebook with 1 GB RAM, but not on my system with 9 GB, so I figured I needed to restrict overall memory.

#10 - 28 Aug 2012 09:54 - Robin Mills

Thanks, Christian. I think we're all "on the same page" with this one. You're prescribing ulimit to ensure that the issue appears. The issue is the crash caused by the corrupted png demanding too much memory. As you've explained, that file could crash an application which uses the library.

#11 - 30 Aug 2012 03:55 - Andreas Huggel

I'll commit the following small patch, it replaces an existing check with a hardcoded max size with a check that makes sure we don't read beyond the end of the data source (input file in this case). I'm also adding a test case to the test-suite using the corrupted input file you provided.

```
diff --git a/src/pngimage.cpp b/src/pngimage.cpp
index 3407371..b527901 100644
--- a/src/pngimage.cpp
+++ b/src/pngimage.cpp
@@ -118,6 +118,7 @@ namespace Exiv2 {
         }
         clearMetadata();
```

```
+         const long imgSize = io_->size();
          DataBuf cheaderBuf(8);       // Chunk header size : 4 bytes (data size) + 4 bytes (chunk type).

          while(!io_->eof())
@@ -134,7 +135,8 @@ namespace Exiv2 {

              // Decode chunk data length.
              uint32_t dataOffset = Exiv2::getULong(cheaderBuf.pData_, Exiv2::bigEndian);
-             if (dataOffset > 0x7FFFFFFF) throw Exiv2::Error(14);
+             long pos = io_->tell();
+             if (pos == -1 || static_cast<long>(dataOffset) > imgSize - pos) throw Exiv2::Error(14);

              // Perform a chunk triage for item that we need.
```

#12 - 30 Aug 2012 14:19 - Robin Mills

Andreas

That's a clever trick to limit the max alloc to the size of the file.

Let's leave this issue assigned to me and I'll close this when I'm certain that the Christian's file passes our test suite on every platform. It may be some time before this can be closed because of the work associated with http://dev.exiv2.org/boards/3/topics/1249. I hope to do some work on MSVC/MSVC64 concerning the gsoc additions this weekend.

#13 - 15 Jan 2013 19:56 - Robin Mills

- Status changed from Assigned to Resolved

#14 - 24 Jul 2013 15:25 - Robin Mills

- Status changed from Resolved to Closed

Fixed in 0.24.

Files

exiv2crash.input	2.52 KB	25 Aug 2012	Christian Grothoff