Exiv2 - Bug #1248

floating point exception / crash on malformed input

21 Oct 2016 21:17 - Hanno Böck

Status:	Closed	**Start date:**	21 Oct 2016
Priority:	Normal	**Due date:**	
Assignee:	Robin Mills	**% Done:**	100%
Category:	not-a-bug	**Estimated time:**	5.00 hours
Target version:	0.26		

Description

The attached file will cause a floating point exception with "exiv2 print".

Here's a stack trace:

```
18792ERROR: AddressSanitizer: FPE on unknown address 0x7f50fa97325b (pc 0x7f50fa97325b bp 0x7ffdcda38c80 sp
0x7ffdcda38b90 T0)
#0 0x7f50fa97325a in Exiv2::ValueType<std::pair<int, int> >::toLong(long) const /f/exiv2/trunk/include/exiv2/value.hpp:1666:32
#1 0x7f50fab94125 in std::ostream& Exiv2::Internal::printTag<27, Exiv2::Internal::exifFlash>(std::ostream&, Exiv2::Value const&,
Exiv2::ExifData const*) /f/exiv2/trunk/src/tags_int.hpp:229:44
#2 0x7f50fa94d256 in Exiv2::Exifdatum::write(std::ostream&, Exiv2::ExifData const*) const /f/exiv2/trunk/src/exif.cpp:230:16
#3 0x55fdcd in Action::Print::printTag(Exiv2::ExifData const&, std::string const&, std::string const&) const
/f/exiv2/trunk/src/actions.cpp:497:13
#4 0x5485a2 in Action::Print::printSummary() /f/exiv2/trunk/src/actions.cpp:360:13
#5 0x544a58 in Action::Print::run(std::string const&) /f/exiv2/trunk/src/actions.cpp:244:44
#6 0x4fe07f in main /f/exiv2/trunk/src/exiv2.cpp:170:19
#7 0x7f50f920378f in __libc_start_main (/lib64/libc.so.6+0x2078f)
#8 0x421eb8 in _start (/r/exiv2/exiv2+0x421eb8)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: FPE /f/exiv2/trunk/include/exiv2/value.hpp:1666:32 in Exiv2::ValueType<std::pair<int, int>
>::toLong(long) const
18792ABORTING
```

Related issues:

Related to Exiv2 - Bug #1247: out of bounds read access in Exiv2::Image::setI...	**Closed**	**21 Oct 2016**

History

#1 - 21 Oct 2016 23:00 - Robin Mills

Hanno

What are you doing here? This is another malformed file. It's very helpful to have those files, however I can't deal with an avalanche of illegal files at the moment. I'm in the end game of completing v0.26. The team will meet on Sunday to discuss RC1 v0.26:
http://clanmills.com/exiv2/Exiv2v0-26RC1.pdf

How about proceeding as follows:

1) We change the title of #1247 to "Exceptions/Crashes due to malformed input files"
2) When you find another file, update #1247 with the file and symptoms.

Find all that you can and we'll deal with them in v0.27.

Are you aware of the term "fuzzing"? I believe this is the art of creating illegal files to put software under stress. I am aware that libexiv2 can be exploited in this way, however I have never had time to work on this topic. Images are read by the function readMetadata() in the image handlers src/tiffimage.cpp, src/jpgimage.cpp/etc. It's probably necessary to go through those functions line-by-line and devise files to break the code. Then engineer a fix in the code. And that won't be the end of the project - far from it. We can modify metadata/delete metadata which could also put the code under great stress. Such a project is probably a very considerable undertaking.

Dealing with this by dumping files on me one-at-a-time is very inefficient use of my time. The fix you provided earlier to #1247 was incorrect and generated compiler warnings on GCC. And I added your test file to our test harness. So, you've taken up 2 hours of my day. 2 hours that I would prefer to use on the bug hunt for v0.26 #1230.

If you'd like to take on the challenge of finding ***and*** fixing all of this, I will happily assist/mentor you. However I'd like to avoid a very discouraging avalanche of bug reports.

#2 - 21 Oct 2016 23:18 - Robin Mills

- Category set to image format

- Target version set to 0.28

#3 - 22 Oct 2016 11:02 - Hanno Böck

These files are actually a result of fuzzing. I'm using american fuzzy lop in combination with address sanitizer. See here [1] [2] for some docs.

If you feel it's more appropriate I can add them all to one bug.

[1] https://fuzzing-project.org/tutorial2.html
[2] https://fuzzing-project.org/tutorial3.html

#4 - 22 Oct 2016 12:51 - Robin Mills

Hanno

Thanks for doing this. I did more work on #1247 last night and again this morning (I'm in England). I've thought of a way to cure/hide this by adding a function ***Image::lint()*** which would have signal handlers. ***Image::lint()*** would "sniff" files by running readMetadata() and printStructure() in a safe context.

I have closed #1247. As you discover additional issues, please update this issue #1248.   I'll add the ***Image::lint()*** function in v0.27 and this will make the code immune to crashes on opening files. ***Image::lint()*** will make the library safer. I will also look at small changes in readMetadata() and printStructure() on a case-by-case basis.

We are having a Team Meeting tomorrow at 13:00 UTC 2016-10-23. You are welcome to join us if you wish.
http://dev.exiv2.org/boards/3/topics/2767

Robin

#5 - 22 Oct 2016 14:31 - Robin Mills

- Status changed from New to Assigned

- Assignee set to Robin Mills

- Estimated time set to 100.00 h

#6 - 22 Oct 2016 20:52 - Hanno Böck

- File exiv2-fpe-print0x9207-printTag.jpg added

Attached a file causing a different floating point exception (may be the same underlying bug).

Stack trace:
```
15656ERROR: AddressSanitizer: FPE on unknown address 0x000000736534 (pc 0x000000736534 bp 0x0c0800001924 sp 0x7ffd491c0430 T0)
#0 0x736533 in Exiv2::ValueType<std::pair<int, int> >::toLong(long) const /f/exiv2/trunk/include/exiv2/value.hpp:1666:32
#1 0x885227 in std::ostream& Exiv2::Internal::printTag<9, Exiv2::Internal::exifMeteringMode>(std::ostream&, Exiv2::Value const&, Exiv2::ExifData const*) /f/exiv2/trunk/src/tags_int.hpp:229:44
#2 0x8627b9 in Exiv2::Internal::print0x9207(std::ostream&, Exiv2::Value const&, Exiv2::ExifData const*) /f/exiv2/trunk/src/tags.cpp:2733:16
#3 0x7093e4 in Exiv2::Exifdatum::write(std::ostream&, Exiv2::ExifData const*) const /f/exiv2/trunk/src/exif.cpp:230:16
#4 0x610fcd in Action::Print::printTag(Exiv2::ExifData const&, std::string const&, std::string const&) const /f/exiv2/trunk/src/actions.cpp:497:13
#5 0x5fb699 in Action::Print::printSummary() /f/exiv2/trunk/src/actions.cpp:405:9
#6 0x5f5c58 in Action::Print::run(std::string const&) /f/exiv2/trunk/src/actions.cpp:244:44
#7 0x5af24f in main /f/exiv2/trunk/src/exiv2.cpp:170:19
#8 0x7f99f63e778f in __libc_start_main (/lib64/libc.so.6+0x2078f)
#9 0x4d3088 in _start (/r/exiv2/exiv2+0x4d3088)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: FPE /f/exiv2/trunk/include/exiv2/value.hpp:1666:32 in Exiv2::ValueType<std::pair<int, int> >::toLong(long) const
15656ABORTING
```

#7 - 22 Oct 2016 21:01 - Hanno Böck

- File heapoverflow-printIFDStructure-382.jpg added

- File heapoverflow-byteSwap4-printIFDStructure-429.jpg added

Attached are two files causing (different) heap buffer overflows (one writing and one reading) in exiv2.

I have to add that these don't crash exiv2 for me. This is not uncommon for memory safety bugs, it often depends on system, compiler behavior or memory layout whether these cause an issue or not. You can reliably see these bugs with address sanitizer, which is a feature of gcc and clang and can be enabled by passing "-fsanitize=address" to the compiler flags.

Address Sanitizer output for heapoverflow-printIFDStructure-382.jpg:
```
12115ERROR: AddressSanitizer: heap-buffer-overflow on address 0x60200000edd3 at pc 0x00000076004c bp 0x7fffe4bdfb90 sp 0x7fffe4bdfb88
```

```
WRITE of size 4 at 0x60200000edd3 thread T0
#0 0x76004b in Exiv2::Image::printIFDStructure(Exiv2::BasicIo&, std::ostream&, Exiv2::PrintStructureOption, unsigned int, bool, char, int)
/f/exiv2/trunk/src/image.cpp:382:17
#1 0x761168 in Exiv2::Image::printTiffStructure(Exiv2::BasicIo&, std::ostream&, Exiv2::PrintStructureOption, int, unsigned long)
/f/exiv2/trunk/src/image.cpp:494:13
#2 0x89c0f9 in Exiv2::TiffImage::printStructure(std::ostream&, Exiv2::PrintStructureOption, int) /f/exiv2/trunk/src/tiffimage.cpp:348:9
#3 0x894d98 in Exiv2::TiffImage::readMetadata() /f/exiv2/trunk/src/tiffimage.cpp:191:9
#4 0x5f658c in Action::Print::printSummary() /f/exiv2/trunk/src/actions.cpp:289:9
#5 0x5f5c58 in Action::Print::run(std::string const&) /f/exiv2/trunk/src/actions.cpp:244:44
#6 0x5af24f in main /f/exiv2/trunk/src/exiv2.cpp:170:19
#7 0x7fc2bc7b278f in __libc_start_main (/lib64/libc.so.6+0x2078f)
#8 0x4d3088 in _start (/r/exiv2/exiv2+0x4d3088)

0x60200000edd3 is located 2 bytes to the right of 1-byte region [0x60200000edd0,0x60200000edd1)
allocated by thread T0 here:
#0 0x5abd40 in operator new[](unsigned long) (/r/exiv2/exiv2+0x5abd40)
#1 0x7589fa in Exiv2::DataBuf::DataBuf(long) /f/exiv2/trunk/include/exiv2/types.hpp:204:46
#2 0x7589fa in Exiv2::Image::printIFDStructure(Exiv2::BasicIo&, std::ostream&, Exiv2::PrintStructureOption, unsigned int, bool, char, int)
/f/exiv2/trunk/src/image.cpp:381
#3 0x761168 in Exiv2::Image::printTiffStructure(Exiv2::BasicIo&, std::ostream&, Exiv2::PrintStructureOption, int, unsigned long)
/f/exiv2/trunk/src/image.cpp:494:13

Address Sanitizer output for heapoverflow-byteSwap4-printIFDStructure-429.jpg:
31059ERROR: AddressSanitizer: heap-buffer-overflow on address 0x62700001b930 at pc 0x00000075fee6 bp 0x7ffc51df1a90 sp 0x7ffc51df1a88
READ of size 1 at 0x62700001b930 thread T0
#0 0x75fee5 in Exiv2::Image::byteSwap4(Exiv2::DataBuf&, unsigned long, bool) /f/exiv2/trunk/src/image.cpp:265:16
#1 0x75fee5 in Exiv2::Image::printIFDStructure(Exiv2::BasicIo&, std::ostream&, Exiv2::PrintStructureOption, unsigned int, bool, char, int)
/f/exiv2/trunk/src/image.cpp:429
#2 0x761168 in Exiv2::Image::printTiffStructure(Exiv2::BasicIo&, std::ostream&, Exiv2::PrintStructureOption, int, unsigned long)
/f/exiv2/trunk/src/image.cpp:494:13
#3 0x89c0f9 in Exiv2::TiffImage::printStructure(std::ostream&, Exiv2::PrintStructureOption, int) /f/exiv2/trunk/src/tiffimage.cpp:348:9
#4 0x894d98 in Exiv2::TiffImage::readMetadata() /f/exiv2/trunk/src/tiffimage.cpp:191:9
#5 0x5f658c in Action::Print::printSummary() /f/exiv2/trunk/src/actions.cpp:289:9
#6 0x5f5c58 in Action::Print::run(std::string const&) /f/exiv2/trunk/src/actions.cpp:244:44
#7 0x5af24f in main /f/exiv2/trunk/src/exiv2.cpp:170:19
#8 0x7f94d781178f in __libc_start_main (/lib64/libc.so.6+0x2078f)
#9 0x4d3088 in _start (/r/exiv2/exiv2+0x4d3088)

0x62700001b930 is located 0 bytes to the right of 12336-byte region [0x627000018900,0x62700001b930)
allocated by thread T0 here:
#0 0x5abd40 in operator new[](unsigned long) (/r/exiv2/exiv2+0x5abd40)
#1 0x7589fa in Exiv2::DataBuf::DataBuf(long) /f/exiv2/trunk/include/exiv2/types.hpp:204:46
#2 0x7589fa in Exiv2::Image::printIFDStructure(Exiv2::BasicIo&, std::ostream&, Exiv2::PrintStructureOption, unsigned int, bool, char, int)
/f/exiv2/trunk/src/image.cpp:381
#3 0x761168 in Exiv2::Image::printTiffStructure(Exiv2::BasicIo&, std::ostream&, Exiv2::PrintStructureOption, int, unsigned long)
/f/exiv2/trunk/src/image.cpp:494:13
```

#8 - 22 Oct 2016 21:31 - Robin Mills

Thanks for working on this, Hanno. I've thought of a serious limitation with my idea of the function ***Image::lint()***. It has to modify and restore signal handlers. If signal handlers are process wide, messing with them in a single function can never be thread-safe without a lock. I never want to add a hidden lock to libexiv2. None of this misery occurs with clang where the code throws an Exiv2 exception. The host application doesn't crash and can inform user that the file cannot be read.

#9 - 23 Oct 2016 08:06 - Robin Mills

I'm wondering if there is any point in working on this issue. It isn't an issue when you build with clang. The problem is 100% due to elderly compilers such as GCC and CL. It does not make sense to pollute the Exiv2 code with checks to remedy deficiencies in the build tools.

The idea/proposal to write ***Image::lint()*** can never be implemented in a thread safe manner.
http://stackoverflow.com/questions/5282099/signal-handling-in-pthreads

I feel this issue and #1247 are "not a bug" and should be closed.

#10 - 23 Oct 2016 08:32 - Robin Mills

More proof that this is a build tools issue. I test last night's MSVC/2015 build. No problem:

```
C:\temp\dist\2015\x64\dll\Release\bin>exiv2 -vV | grep -e svn -e date -e time -e compiler
compiler=MSVC
date=Oct 23 2016
time=02:46:49
svn=4655
have_gmtime_r=0
have_timegm=0
```

```
xmlns=mediapro:http://ns.iview-multimedia.com/mediapro/1.0/

C:\temp\dist\2015\x64\dll\Release\bin>exiv2 -pa \Users\rmills\gnu\exiv2\trunk\test\data\exiv2-bug1247.jpg
Exiv2 exception in print action for file \Users\rmills\gnu\exiv2\trunk\test\data\exiv2-bug1247.jpg:
Not a valid ICC Profile

C:\temp\dist\2015\x64\dll\Release\bin>exiv2 -pR \Users\rmills\gnu\exiv2\trunk\test\data\exiv2-bug1247.jpg
STRUCTURE OF JPEG FILE: \Users\rmills\gnu\exiv2\trunk\test\data\exiv2-bug1247.jpg
 address | marker       |  length | data
       0 | 0xffd8 SOI
       2 | 0xffe2 APP2  |      16 | ICC_PROFILE.... chunk 0/0
      18 | 0xffffffffff É"ÿ·Exiv2 exception in print action for file \Users\rmills\gnu\exiv2\trunk\test\data\
exiv2-bug1247.jpg:
This does not look like a JPEG image

C:\temp\dist\2015\x64\dll\Release\bin>
```

Two simple questions:

1) Why are we working on this?
2) Is there any reason I should not close this issue?

#11 - 24 Oct 2016 10:25 - Robin Mills

- Category changed from image format to not-a-bug

- Status changed from Assigned to Closed

- Target version changed from 0.28 to 0.26

- % Done changed from 0 to 100

- Estimated time changed from 100.00 h to 5.00 h

#12 - 31 Oct 2016 12:25 - Hanno Böck

I'm not sure why you come to the conclusion these are not bugs....

You say these don't affect clang, I can't reproduce that. All four issues also happen with clang (latest version 3.9.0, on Linux). I'm also not entirely sure if your comments are only related to the fpe issue or about all issues (but you asked me to put unrelated bugs into one bugreport...)

The floating point exception issues are "only" crashes, so I might understand you say that you don't want to fix them, but the buffer overflows are security issues. If it's not your intention to fix those then this is very problematic and at the very least you need to clearly state that exiv2 is not suitable for untrusted input.

#13 - 31 Oct 2016 12:50 - Robin Mills

I'm very unwilling to stick lots of code into Exiv2 to deal with illegal files. It will be a very considerable undertaking which might have unbounded scope. I'll build it with clang 3.9 and see what happens. If it throws an exception, I don't want to dig deeper.

I have to point out that I deal almost alone with Exiv2. I have to say "No" to some requests. I am not under any obligation to you to say that it is untrusted or anything else for that matter. If you had the courtesy to say "Thank You for investigating this Robin and being willing to commit 100 hours to this matter", I would feel more enthusiasm for your request. However you have been shown no appreciation for my cooperation.

#14 - 31 Oct 2016 12:54 - Robin Mills

One more comment. If you think this is a very important matter, the best way to have it resolved is for you to join the team and work on the issue.

#15 - 31 Oct 2016 14:33 - Robin Mills

You are correct. It does crash on clang 3.9 on Linux.

```
750 rmills@rmillsmbp-ubuntu:~/gnu/exiv2/trunk $ exiv2 -vVg compiler -g version -g date -g time
exiv2 0.25 001900 (64 bit build)
compiler=Clang
version=4.2.1 Compatible Clang 3.9.0 (tags/RELEASE_390/final)
date=Oct 31 2016
time=14:25:26
id=$Id: version.cpp 4590 2016-09-30 16:45:54Z robinwmills $
have_gmtime_r=1
have_timegm=1
xmlns=mediapro:http://ns.iview-multimedia.com/mediapro/1.0/
751 rmills@rmillsmbp-ubuntu:~/gnu/exiv2/trunk $ exiv2 -pa test/data/exiv2-bug1247.jpg
Exiv2 exception in print action for file test/data/exiv2-bug1247.jpg:
Not a valid ICC Profile
752 rmills@rmillsmbp-ubuntu:~/gnu/exiv2/trunk $ exiv2 -pR test/data/exiv2-bug1247.jpg
STRUCTURE OF JPEG FILE: test/data/exiv2-bug1247.jpg
```

```
 address | marker       |  length | data
       0 | 0xffd8 SOI  
       2 | 0xffe2 APP2  |      16 | ICC_PROFILE.... chunk 0/0
Segmentation fault (core dumped)
753 rmills@rmillsmbp-ubuntu:~/gnu/exiv2/trunk $ 
```

However, I'm not working on this. If I have a ***GSoC*** student to work on #992 in 2017, I will ask him to investigate this.

#16 - 31 Oct 2016 19:08 - Robin Mills

Apple Clang does not crash. It throws an exception. Here's the clang version info for the current Xcode (8.1 8B62) on Sierra 10.12.1

```
558 rmills@rmillsmbp:~/gnu/exiv2/trunk $ clang --version
Apple LLVM version 8.0.0 (clang-800.0.42.1)
Target: x86_64-apple-darwin16.1.0
Thread model: posix
InstalledDir: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin
559 rmills@rmillsmbp:~/gnu/exiv2/trunk $ 
```

Perhaps this has something to do with the Apple Sandbox. We need to understand more to decide how best to deal with this. Hammering nails into the Exiv2 image handling code to protect against any/every possible image spec violation sounds like the road to nowhere. Clearly, we could launch an external image-lint program (which could be exiv2 with a signal handler). This adds considerable overhead to opening images - however it would help. By testing the "safety" of the image in a separate process, we avoid the need for a lock in libexiv2.

I know nothing about hacking and exploiting buffer overflows. Perhaps we need more caution about them. This is a mind-boggling issue. The code to defend our image read/write code could be larger than the current library. It appears that Apple have generated code which catches the crash. We need more research on how to handle this within a library. And preferably a method that doesn't require us to add thousands of ***if*** statements.

#17 - 01 Nov 2016 08:49 - Robin Mills

Here is a tool that we could investigate. http://safecode.cs.illinois.edu/index.html It's a little elderly, however I found a September 2015 version: https://github.com/jtcriswell/safecode-llvm37/tree/master/cmake The LLVM web site promotes SAFECode as a current project http://llvm.org For sure that isn't working in clang 8.0 on MacOS-X:

```
579 rmills@rmillsmbp:~/gnu/exiv2/trunk $ clang --version
Apple LLVM version 8.0.0 (clang-800.0.42.1)
Target: x86_64-apple-darwin16.1.0
Thread model: posix
InstalledDir: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin
580 rmills@rmillsmbp:~/gnu/exiv2/trunk $ ./configure "CXXFLAGS=-g -fmemsafety" 
checking for g++... g++
checking whether the C++ compiler works... no
configure: error: in `/Users/rmills/gnu/exiv2/trunk':
configure: error: C++ compiler cannot create executables
See `config.log' for more details
581 rmills@rmillsmbp:~/gnu/exiv2/trunk $ 
```

Nor Linux with clang 3.9 on Ubuntu:

```
827 rmills@rmillsmbp-ubuntu:~/gnu/exiv2/trunk $ which clang
/usr/bin/clang
828 rmills@rmillsmbp-ubuntu:~/gnu/exiv2/trunk $ clang --version
clang version 3.9.0-1ubuntu1 (tags/RELEASE_390/final)
Target: x86_64-pc-linux-gnu
Thread model: posix
InstalledDir: /usr/bin
829 rmills@rmillsmbp-ubuntu:~/gnu/exiv2/trunk $ ./configure "CXXFLAGS=-g -fmemsafety"checking for g++... g++
checking whether the C++ compiler works... no
configure: error: in `/home/rmills/gnu/exiv2/trunk':
configure: error: C++ compiler cannot create executables
See `config.log' for more details
830 rmills@rmillsmbp-ubuntu:~/gnu/exiv2/trunk $ 
```

Alien Skin Software are an Exiv2 User. I have been discussing with them to add options to ***ImageFactory::open*** for something in which they are interested. http://dev.exiv2.org/issues/1245#note-2

Currently, we have an option bool bUseCurl.

```
    class EXIV2API ImageFactory {
    ...
    public:
    ...
        static BasicIo::AutoPtr createIo(const std::string& path, bool useCurl = true);
    ...
        static Image::AutoPtr open(const std::string& path, bool useCurl = true);
    ...
    };
```

bool useCurl could be replaced with an enum bitmask. We could add an option ***checkSafe*** which would invoke exiv2(.exe) in a separate process. We have code in src/version.cpp to detect the location of the libexiv2.vv.so (.dylib, .dll) library at run-time. We can take advantage of that to determine the path to exiv2(.exe).

Returning to Hanno's request that we say that Exiv2 is ***"not suitable for untrusted input"***. I will not do that. The default (for all open source software) is that the user must determine suitability for his/her purposes. We do not claim that Exiv2 is safe for all files. I doubt if we could ever make such a claim even if we pepper our code with checks for malicious images. I don't believe in 2016 that it is possible to guarantee the safety and reliability of any software.

Files

exiv2-fpe-printTag.jpg	65 Bytes	21 Oct 2016	Hanno Böck
exiv2-fpe-print0x9207-printTag.jpg	65 Bytes	22 Oct 2016	Hanno Böck
heapoverflow-printIFDStructure-382.jpg	220 Bytes	22 Oct 2016	Hanno Böck
heapoverflow-byteSwap4-printIFDStructure-429.jpg	18 Bytes	22 Oct 2016	Hanno Böck