Exiv2 - Bug #888

(near-)infinite loop in video decoders

10 Mar 2013 16:02 - Alyssa Milburn

Status:	Closed	**Start date:**	10 Mar 2013
Priority:	Normal	**Due date:**	
Assignee:	Abhinav Badola	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	0.24		

Description

If I hand RiffVideo::nikonTagsHandler() data with a size value <4, then it subtracts 4 from it (riffvideo.cpp:745 at the time of writing) and will loop for a very long time. I guess you could cause a similar effect by messing with the size value subtracted on line 751.

The attached file can be used as a testcase.

You can do a similar trick with AsfVideo::decodeBlock (which will loop forever if given size==0).

I don't know if you care about these kind of bugs. If you do, at a glance it looks like might be some other candidates for similar issues, at least on 32-bit platforms, such as jpgimage.cpp:187 (by overflowing the position if long is 32 bits), and psdimage.cpp:241/242 (by using a large value for resourceSize).

History

#1 - 10 Mar 2013 16:04 - Abhinav Badola

- Assignee set to Abhinav Badola

#2 - 10 Mar 2013 16:10 - Abhinav Badola

Thank you Alyssa for reporting the crash.

Indeed it is a quite a notorious bug.
I will look into the matter asap and patch the video code accordingly.

#3 - 11 Mar 2013 01:19 - Abhinav Badola

Hi Alyssa,
I looked into the problem, and this is what I observed. Please correct me if I may have missed some point.

The file that you provided with the issue seems to have been generated manually. (**I really appreciate the effort.**)
But it seems that the file doesn't follow the RIFF standard specification.

I am saying this, because when I run the exiv2 utility while debugging the Video File, this is what I got as the size.

Tag = NCDT
Size = 153428472

The size is not turning out to be zero, as such.
I guess this is the reason for the long run of the loop that you specified, "and will loop for a very long time".
This simply means that it is trying to subtract 4 recursively from the size, and will definitely take time.

I guess this problem should not occur, if the file has proper fields according to the RIFF standards.

Please can you send a link or upload a proper RIIF or ASF file which would produce a similar effect.

#4 - 11 Mar 2013 02:38 - Alyssa Milburn

Sorry, if you don't care about bugs caused by invalid data, these bugs are irrelevant. That's why I said "I don't know if you care about these kind of bugs". Various projects are using libexiv2 right now for reading metadata on files which may have been downloaded from the internet, for example, so trusting input files sounds like a bad idea.

#5 - 11 Mar 2013 07:27 - Andreas Huggel

Thanks for reporting this issue, Alyssa. We do care about such things; making sure Exiv2 behaves well for any kind of input is a challenge that we have accepted. We have many checks to avoid issues with corrupted (or 'doctored') images in the more mature code and we'll be happy to add more. The video metadata support on the other hand is relatively new and has not been exposed to as many problematic files, so it may not be quite as stable.

Abhinav, see e.g., tiffvisitor.cpp for examples of sanity checks and how they are typically handled. Generally, if we detect an anomaly, we issue a warning, make sure the process doesn't fail (e.g., by skipping the suspicious section) and continue parsing as much of the remaining metadata as possible. Below is an example from this file. Without having looked at the reported case in detail, it sounds like a similar check might be possible to avoid processing unreasonably large values and avoid the reported issue(s). As you program more, it will become a reflex to consider "what if" scenarios and document your assumptions (e.g., using assert) in your code routinely, so that many such cases (never all) will be taken care of from the beginning.

```
        // Sanity check with an "unreasonably" large number
        if (n > 256) {
#ifndef SUPPRESS_WARNINGS
            EXV_ERROR << "Directory " << groupName(object->group()) << " with "
                      << n << " entries considered invalid; not read.\n";
#endif
            return;
        }
```

#6 - 11 Mar 2013 08:21 - Abhinav Badola

Thank you Alyssa and Andreas for your valuable feedback.
I will fix the errors asap.

#7 - 13 Mar 2013 14:54 - Abhinav Badola

- % Done changed from 0 to 50

Hi Alyssa,

I have applied some patches in the latest revision **R2996** and **R2997**.
I have tested thoroughly on my local machine, and now the code seems to be working fine.

Whenever you get time, please confirm if the bug is solved at your end as well.
Once I have your confirmation, I will be closing this issue and marking it as solved.

Thank you for your patience and support in making Video Code better.

#8 - 05 May 2013 04:45 - Abhinav Badola

- Status changed from New to Resolved

- % Done changed from 50 to 100

#9 - 24 Jul 2013 15:09 - Robin Mills

- Target version set to 0.24

Fixed in 0.24.

#10 - 24 Jul 2013 15:09 - Robin Mills

- Status changed from Resolved to Closed

Fixed in 0.24.

Files

infiniteloop.riff	64 KB	10 Mar 2013	Alyssa Milburn