Closed Issue created 6 months ago by **Adam Doupe**

matroska: segfault / potential heap overflow in zlib decoding

Describe the vulnerability

Integer overflow while decoding zlib encoded data in `gst_matroska_decompress_data`.

The given crashing POC zlib-decode-overflow-1.mkv has a block that when zlib decompressed is `0x100000000`.

This overflows the `new_size` integer here:
https://gitlab.freedesktop.org/gstreamer/gstreamer/-/blob/main/subprojects/gst-plugins-good/gst/matroska/matroska-read-common.c#L117

One potential root cause is that the `zstream` data counters are 64-bit:
https://gitlab.freedesktop.org/gstreamer/gstreamer/-/blob/main/subprojects/gst-plugins-good/gst/matroska/matroska-read-common.c#L105

while 32-bit integers are used to store the size in `gst_matroska_decompress_data`
https://gitlab.freedesktop.org/gstreamer/gstreamer/-/blob/main/subprojects/gst-plugins-good/gst/matroska/matroska-read-common.c#L80

Expected Behavior

Not segfault.

Observed Behavior

Segfault

Setup

- **Operating System:** Ubuntu 20.04.4
- **Device:** Computer
- **GStreamer Version:** 1.16.2
- **Command line:** `gst-play-1.0 ./zlib-decode-overflow-1.mkv`

Steps to reproduce the bug

1. Download zlib-decode-overflow-1.mkv
2. Run `gst-play-1.0 ./zlib-decode-overflow-1.mkv` (takes about 8 seconds to trigger on my system)

How reproducible is the bug?

Always

Impact

Depending on the libc used, and the underlying OS capabilities, it could be just a segfault or a heap overwrite.

If the libc uses mmap for large chunks, and the OS supports mmap, then it is just a segfault (because the realloc before the integer overflow will use mremap to reduce the size of the chunk, and it will start to write to unmapped memory).

However, if using a libc implementation that does not use mmap, or if the OS does not support mmap while using libc, then this would likely result in a heap overwrite.

Additional Information

Not sure if this qualifies for a CVE, not sure what this project does in the past. Happy to discuss.

Thanks!

Edited 6 months ago by Adam Doupe

Drag your designs here or click to upload.

Tasks 0

No tasks are currently assigned. Use tasks to break down this issue into smaller parts.

Linked items 0

Related merge requests 1

matroskademux, qtdemux: Fix integer overflows in zlib/bz2/etc decompression code

!2610 1.21.1

When this merge request is accepted, this issue will be closed automatically.

Activity

Adam Doupe changed the description 6 months ago

Sebastian Dröge @slomo · 6 months ago Owner

> One potential root cause is that the `zstream` data counters are 64-bit: https://gitlab.freedesktop.org/gstreamer/gstreamer/-/blob/main/subprojects/gst-plugins-good/gst/matroska/matroska-read-common.c#L105

The counters seem to be 32 bit here. `avail_in` is `uInt` which is a typedef to `unsigned int` on my system at least. The `total_in` counter is 64 bit (or rather: `unsigned long` so still 32 bit on 32 bit platforms and 64 bit Windows).

So in addition to making sure the local variables can fit enough, we also need a) overflow checks anyway, b) put at most 2**32 - 1 bytes into the stream per iteration or otherwise `avail_in` will overflow.

We don't support blocks larger than `MAX_BLOCK_SIZE` (15MB at this time) so it should be sufficient to error out if the input is too large, and then handle decompression going to the limits accordingly.

There is also another bug where `avail_out` is always the size of the full buffer and not the size that is remaining from `next_out`. That can potentially cause out of bounds writes, and is also fixed here.

In addition I also limited the maximum decompressed size to something less than `G_MAXSIZE` as that's rather pathological and would likely cause out of memory handling.

The same bug is also in the bzip2/lzo decompression code below and there's also a potential overflow in the headerstrip decompression. I've fixed those too here the same way.

This should fix all these issues: 0001-matroskademux-Fix-integer-overflows-in-zlib-bz2-etc-.patch

Please give it a thorough review :) And again, thanks for the analysis and reporting.

Tim-Philipp Müller added Security label 6 months ago

Tim-Philipp Müller @tpm · 6 months ago Owner

We'll probably merge these patches closer to the next stable bug-fix release %1.20.3 in ca. 2 weeks time or so.

It would be great if you could give us some indication whether you expect there to be more issues forthcoming in the near future (e.g. if you're running a lab at the moment that's still actively identifying problems), so we don't do a new bug-fix release and then 10 new issues come in the day after :)

Adam Doupe @adamdoupe · 6 months ago Author

> The counters seem to be 32 bit here. `avail_in` is `uInt` which is a typedef to `unsigned int` on my system at least. The `total_in` counter is 64 bit (or rather: `unsigned long` so still 32 bit on 32 bit platforms and 64 bit Windows).

My apologies, I found some of these bugs awhile ago, so my memory of the underlying root cause was a bit rusty.

> The same bug is also in the bzip2/lzo decompression code below

Ah, the target that I analyzed had these disabled, so I ignored the bzip2/lzo decompression.

> a potential overflow in the headerstrip decompression

This one I did find, and I was going to file a separate issue for it. I don't think it's exploitable through `matroska-emux`, because of the size check, but it can be triggered through `matroskaparse` (although I'm not sure if that component is used).

Ok, I looked at the patch, the only thing I see is that the size check in LZO decompression:

```
+
+    if (size > G_MAXINT) {
+      GST_WARNING ("too large compressed data buffer.");
+      ret = FALSE;
+      goto out;
+    }
```

Seems unnecessary with the earlier check that is added at the start of `gst_matroska_decompress_data`:

```
+  if (size > G_MAXUINT32) {
+    GST_WARNING ("too large compressed data buffer.");
+    ret = FALSE;
+    goto out;
+  }
```

I also verified that my POC no longer works.

Sebastian Dröge @slomo · 6 months ago Owner

> Ok, I looked at the patch, the only thing I see is that the size check in LZO decompression: [...] Seems unnecessary with the earlier check that is added at the start of `gst_matroska_decompress_data`:

`G_MAXINT` has a smaller maximum value than `G_MAXUINT32` so that's unfortunately needed here. The LZO functions work with plain signed integers.

Edited by Sebastian Dröge 6 months ago

Sebastian Dröge @slomo · 6 months ago Owner

Same bugs also exist in `qtdemux.c:qtdemux_inflate()`. I'll fix those on Monday and check if there's more of this.

Adam Doupe @adamdoupe · 6 months ago Author

@slomo very cool find! I feel a bit silly that I didn't search for all instances of zlib or decompression/inflating in the code base, but I'm glad you did and that you found it!

For the other issues, Red Hat issued the following CVEs:

- "DOS / potential heap overwrite in mkv demuxing using zlib decompression": CVE-2022-1922
- "DOS / potential heap overwrite in mkv demuxing using bzip decompression": CVE-2022-1923
- "DOS / potential heap overwrite in mkv demuxing using lzo decompression": CVE-2022-1924
- "DOS / potential heap overwrite in mkv demuxing using HEADERSTRIP decompression": CVE-2022-1925

@slomo let me know if you'd like me to request CVEs for the other issues that you found in `qtdemux.c`. I have a thread going with them and would be happy to do that and put them here.

Edited by Adam Doupe 6 months ago

Sebastian Dröge @slomo · 5 months ago Owner

0001-qtdemux-Fix-integer-overflows-in-zlib-decompression-.patch

For the `qtdemux` issue. Literally the same thing

> very cool find! I feel a bit silly that I didn't search for all instances of zlib or decompression/inflating in the code base, but I'm glad you did and that you found it!

I actually found the `qtdemux` one by accident, fortunately :) But these were the only two places where zlib is used, and bz2 is used in another place but that doesn't have the same problem.

> I have a thread going with them and would be happy to do that and put them here.

@adamdoupe Sure that would be great, thanks!

Sebastian Dröge @slomo · 5 months ago Owner

@adamdoupe Any news on the CVE ID?

Tim-Philipp Müller changed milestone to %1.20.3 5 months ago

Tim-Philipp Müller @tpm · 5 months ago Owner

@adamdoupe any news on the CVE for qtdemux?

Adam Doupe @adamdoupe · 5 months ago Author

Sorry folks I was traveling last week. I just wrote up the info and sent it to Red Hat, I'll post here when they have a CVE for the qtdemux issue.

Tim-Philipp Müller @tpm · 5 months ago Owner

> I just wrote up the info and sent it to Red Hat, I'll post here when they have a CVE for the qtdemux issue.

Still no CVE number? :)

Adam Doupe @adamdoupe · 5 months ago Author

No update yet, I just sent them another ping. For the prior ones it took roughly a week, and it's been 8 days so far. Hopefully they respond to the ping I just sent.

Please register or sign in to reply

Sebastian Dröge mentioned in commit tpm/gstreamer@e0ae11fa 5 months ago

Sebastian Dröge mentioned in commit tpm/gstreamer@de132490 5 months ago

Sebastian Dröge mentioned in commit tpm/gstreamer@e423dcb9 5 months ago

Sebastian Dröge mentioned in commit tpm/gstreamer@cdf04c3f 5 months ago

Tim-Philipp Müller mentioned in merge request !2610 (merged) 5 months ago

Sebastian Dröge mentioned in commit tpm/gstreamer@d2dfd5d1 5 months ago

Sebastian Dröge mentioned in commit tpm/gstreamer@0ebb5b21 5 months ago

Sebastian Dröge mentioned in commit tpm/gstreamer@a167ac25 5 months ago

Sebastian Dröge mentioned in commit tpm/gstreamer@b9bbd287 5 months ago

Sebastian Dröge mentioned in commit tpm/gstreamer@e23f9c31 5 months ago

Sebastian Dröge mentioned in commit tpm/gstreamer@499061a4 5 months ago

Sebastian Dröge mentioned in commit tpm/gstreamer@869e670f 5 months ago

Sebastian Dröge mentioned in commit tpm/gstreamer@839e48d4 5 months ago

Sebastian Dröge mentioned in commit tpm/gstreamer@a68e693e 5 months ago

Sebastian Dröge mentioned in commit tpm/gstreamer@d3414d6c 5 months ago

Sebastian Dröge mentioned in commit `tpm/gstreamer@fafb0281` 5 months ago

Sebastian Dröge mentioned in commit `tpm/gstreamer@92b5eb1d` 5 months ago

Sebastian Dröge mentioned in commit `tpm/gstreamer@ad601215` 5 months ago

Sebastian Dröge mentioned in commit `tpm/gstreamer@14d306da` 5 months ago

Sebastian Dröge closed via commit `ad601215` 5 months ago

Sebastian Dröge closed via commit `14d306da` 5 months ago

Tim-Philipp Müller made the issue visible to everyone 5 months ago

Adam Doupe @adamdoupe · 5 months ago — Author

@tpm Red Hat just responded and assigned CVE-2022-2122 to the "DOS / potential heap overwrite in qtdemux using zlib decompression" vulnerability.

Tim-Philipp Müller @tpm · 5 months ago — Owner

Thanks, will update the web page.

Sebastian Dröge mentioned in commit `wtaymans/gstreamer@9747b805` 2 weeks ago

Sebastian Dröge mentioned in commit `wtaymans/gstreamer@143dfa14` 2 weeks ago