Babeltrace - Bug #1096

Babeltrace takes 100% CPU and 100% mem reading a malformed trace

04/21/2017 09:17 PM - Francis Deslauriers

Status:	New	**Start date:**	04/21/2017
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Babeltrace Version: 1.5.2
While hacking on the LTTng kernel tracer, I mistakenly wrote a ctf_sequence with 1 element but a length of 0 resulting in a malformed trace.
When reading the malformed trace, Babeltrace hangs and takes 100% of one CPU and 100% of the available memory.

I attached the trace in question.

History

#1 - 04/29/2017 07:56 AM - David Abdurachmanov

Francis Deslauriers wrote:

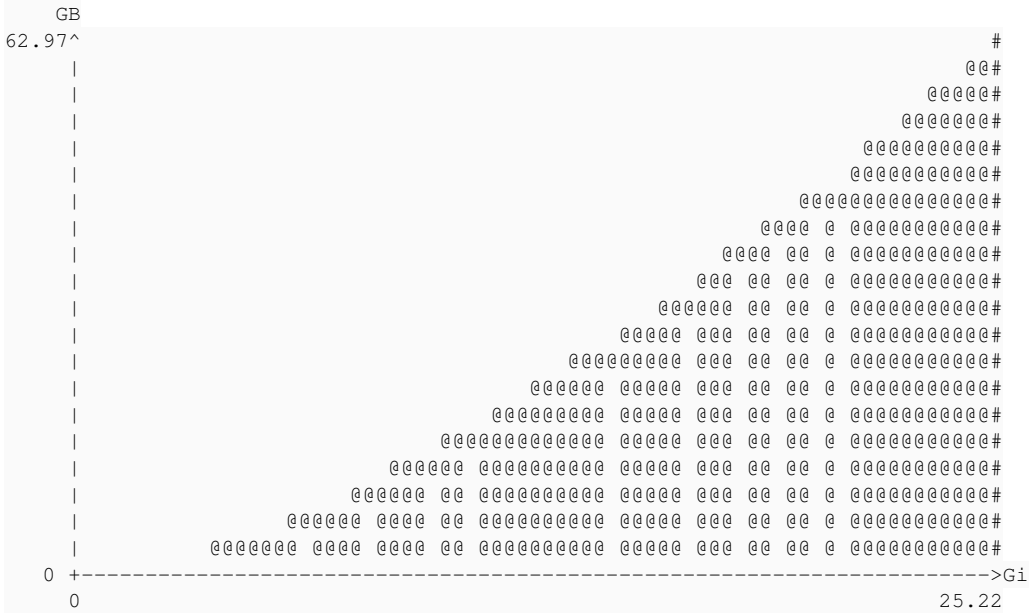
> Babeltrace Version: 1.5.2
> While hacking on the LTTng kernel tracer, I mistakenly wrote a ctf_sequence with 1 element but a length of 0 resulting in a malformed trace.
> When reading the malformed trace, Babeltrace hangs and takes 100% of one CPU and 100% of the available memory.
>
> I attached the trace in question.

Not sure if related, but I also hit an issue with ctf_sequence (or ctf_sequence_hex) where babeltrace 1.5.2 eats all available memory and is killed.

https://lists.lttng.org/pipermail/lttng-dev/2017-April/027042.html

By sequence length is from 1 to 16777215 bytes. My trace is around 1.4G, but babeltrace eats all 64G of RAM and 32G of SWAP.

Valgrind:




```
--------------------------------------------------------------------------------
  n        time(i)         total(B)   useful-heap(B) extra-heap(B)    stacks(B)
--------------------------------------------------------------------------------
 70 27,077,029,304   67,612,706,696   67,554,098,989    58,607,707            0
99.91% (67,554,098,989B) (heap allocation functions) malloc/new/new[], --alloc-fns, etc.
```

```
->99.19% (67,066,989,632B) 0x656430C: g_malloc (in /usr/lib64/libglib-2.0.so.0.4600.2)
| ->99.00% (66,936,995,840B) 0x656E001: g_quark_from_string (in /usr/lib64/libglib-2.0.so.0.4600.2)
| | ->99.00% (66,936,995,840B) 0x4E31DCA: bt_sequence_rw (in /usr/lib64/libbabeltrace.so.1.0.0)
| |   ->99.00% (66,936,995,840B) 0x4E323DC: bt_struct_rw (in /usr/lib64/libbabeltrace.so.1.0.0)
| |     ->99.00% (66,936,995,840B) 0x50441CC: ??? (in /usr/lib64/libbabeltrace-ctf.so.1.0.0)
| |       ->99.00% (66,936,995,840B) 0x4E2E4E2: ??? (in /usr/lib64/libbabeltrace.so.1.0.0)
| |         ->99.00% (66,936,995,840B) 0x4E2F12D: bt_iter_add_trace (in /usr/lib64/libbabeltrace.so.1.0.0)
| |           ->99.00% (66,936,995,840B) 0x4E2F282: bt_iter_init (in /usr/lib64/libbabeltrace.so.1.0.0)
| |             ->99.00% (66,936,995,840B) 0x5047D44: bt_ctf_iter_create (in /usr/lib64/libbabeltrace-ctf.so.1
.0.0)
| |               ->99.00% (66,936,995,840B) 0x402FCD: main (in /usr/bin/babeltrace)
| |
| ->00.19% (129,993,792B) in 1+ places, all below ms_print's threshold (01.00%)
|
->00.72% (487,109,357B) in 1+ places, all below ms_print's threshold (01.00%)
```

#2 - 04/29/2017 08:26 AM - David Abdurachmanov

I also looked into attached trace and seems the same.

```
    GB
63.39^                                                                      #
     |                                                                    @@#
     |                                                                  @@@@#
     |                                                                @@@@@@@#
     |                                                              @@@@@@@@@#
     |                                                            @@@@@@@@@@@@#
     |                                                          @@@@@@@@@@@@@@#
     |                                                        @@@@@@@@@@@@@@@@#
     |                                                    @@@ @@@@@@@@@@@@@@@@#
     |                                                 @@@@@@ @@@@@@@@@@@@@@@@#
     |                                            @@@@@ @@@ @@@@@@@@@@@@@@@@#
     |                                       @@@@@@@@@ @@@ @@@@@@@@@@@@@@@@#
     |                                    @@@@@@ @@@@@ @@@ @@@@@@@@@@@@@@@@#
     |                                @@@@@@@@@@ @@@@@ @@@ @@@@@@@@@@@@@@@@#
     |                             @@@@ @ @@@@@@ @@@@@ @@@ @@@@@@@@@@@@@@@@#
     |                          @@@@@@@ @ @@@@@@ @@@@@ @@@ @@@@@@@@@@@@@@@@#
     |                       @@@@@@ @@@@ @ @@@@@@ @@@@@ @@@ @@@@@@@@@@@@@@@@#
     |                    @@@@@ @@@@ @@@@ @ @@@@@@ @@@@@ @@@ @@@@@@@@@@@@@@@@#
     |                 @@@@@@@ @ @@@@ @@@@ @ @@@@@@ @@@@@ @@@ @@@@@@@@@@@@@@@@#
     |           @@:@@:@@@@@@@ @ @@@@ @@@@ @ @@@@@@ @@@@@ @@@ @@@@@@@@@@@@@@@@#
   0 +-------------------------------------------------------------------------->Gi
     0                                                                      26.93
```

```
--------------------------------------------------------------------------------
  n        time(i)         total(B)   useful-heap(B) extra-heap(B)    stacks(B)
--------------------------------------------------------------------------------
 82 28,913,567,139   68,066,441,744   68,008,441,450    58,000,294            0
99.91% (68,008,441,450B) (heap allocation functions) malloc/new/new[], --alloc-fns, etc.
->98.52% (67,056,030,376B) 0x656430C: g_malloc (in /usr/lib64/libglib-2.0.so.0.4600.2)
| ->98.27% (66,890,170,368B) 0x656E001: g_quark_from_string (in /usr/lib64/libglib-2.0.so.0.4600.2)
| | ->98.27% (66,890,170,368B) 0x4E33626: bt_new_definition_path (in /usr/lib64/libbabeltrace.so.1.0.0)
| |   ->98.27% (66,890,170,368B) 0x4E318CC: ??? (in /usr/lib64/libbabeltrace.so.1.0.0)
| |     ->98.27% (66,890,170,368B) 0x4E31E04: bt_sequence_rw (in /usr/lib64/libbabeltrace.so.1.0.0)
| |       ->98.27% (66,890,170,368B) 0x4E323DC: bt_struct_rw (in /usr/lib64/libbabeltrace.so.1.0.0)
| |         ->98.27% (66,890,170,368B) 0x5044150: ??? (in /usr/lib64/libbabeltrace-ctf.so.1.0.0)
| |           ->98.27% (66,890,170,368B) 0x4E2E4E2: ??? (in /usr/lib64/libbabeltrace.so.1.0.0)
| |             ->98.27% (66,890,170,368B) 0x4E2F43D: bt_iter_next (in /usr/lib64/libbabeltrace.so.1.0.0)
| |               ->98.27% (66,890,170,368B) 0x402DFF: main (in /usr/bin/babeltrace)
| |
| ->00.24% (165,860,008B) in 1+ places, all below ms_print's threshold (01.00%)
|
->01.40% (952,411,074B) in 39 places, all below massif's threshold (1.00%)
```

IIRC, g_quark_from_string copies the string into some kind of pool and allocates GQuark (which hold a number). At first look looks great if you have duplicate data. My buffers contains compressed data, thus unlikely to contain duplicates within a single event. Looks like this is done for each 64-bit chunk of data:

https://github.com/efficios/babeltrace/blob/77baf3a0d381ff3cded04cd76455be041fac177d/types/sequence.c#L69

#3 - 05/24/2019 04:08 PM - Geneviève Bastien

- File trace-fdpf-20190524-165653.tar.gz added

#4 - 05/24/2019 04:12 PM - Geneviève Bastien

This bug was discussed on IRC for a sequence whose size is too big and it tries to allocate huge memory size.

The idea, was to throw an error when the value of the sequence length "does not make sense".

#define "does not make sense" A relation between the remaining size of a packet and the size of the sequence's content. (eg remaining size / minimal sequence obj size)

Files

auto-20170421-164755.tar.gz	115 KB	04/22/2017	Francis Deslauriers
trace-fdpf-20190524-165653.tar.gz	30.9 KB	05/24/2019	Geneviève Bastien