LTTng-modules - Bug #1261

Missing filename field on first openat() syscall

05/05/2020 05:18 PM - Francis Deslauriers

Status:	New	**Start date:**	05/05/2020
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

We witnessed an empty filename field in the openat() syscall tracepoint on a recent Ubuntu 20.04. This openat() syscall is done a literal string hard coded in the the test binary gen-syscall-events .

```
[16:45:44.651793315] (+0.000234515) raton syscall_entry_openat: { cpu_id = 0 }, { dfd = -100, file
name = "" , flags = ( "O_RDONLY" : container = 0 ), mode = ( "S_IXOTH" : container = 1 ) }
```

Our current hypothesis to explain this behavior is that the page on which this literal string is stored is not yet in the page cache when we hit that tracepoint.
The syscall handler in the lttng kernel tracer is not allowed to trigger a page fault. The tracer will omit recording a field if doing so would result in a page fault.
We suspect that this change in behavior comes from a change in the GNU ld linker (or linker script) used to create this executable.
Using the lld or GNU gold linker fixes this issue.
We can easily link executables with different linkers using clang:

```
make CC=clang LDFLAGS="-fuse-ld=lld -L/usr/local/lib/"
```

When looking the base address of the .rodata section of the executable created by the different linkers. (using objdump -s -j .rodata gen-syscall-events)

```
GNU ld 2.34
Contents of section .rodata:
 402000 01000200 4572726f 723a204d 69737369  ....Error: Missi

GNU gold 1.16
Contents of section .rodata:
 400f00 01000200 4572726f 723a204d 69737369  ....Error: Missi

LLVM lld 10.0.0
Contents of section .rodata:
 2008d0 01000200 72656164 00636c6f 73650045  ....read.close.E
```

We see that there are differences in where the sections are placed but that's just one factor that might affect what pages are available we reach the tracepoint. Recent changes in ld seemed to have change what data is available when we reach that tracepoint.

This is not a bug as it's a limitation of the kernel tracepoint infrastructure we rely on. We **cannot** fault pages.
The clean solution would be to remove this restriction and have a safe way to fault pages in tracepoints.