Kernel: ABI, GNU C Library, Core Scheduling, Linux 5.4, Randomness, Kernel Concurrency Sanitizer (KCSAN)

By *Roy Schestowitz*
Created *03/10/2019 - 12:00pm*
Submitted by Roy Schestowitz on Thursday 3rd of October 2019 12:00:09 PM Filed under Linux [1] Google [2] Security [3]

- **Monitoring the internal kernel ABI [4]**

  As part of the Distribution Kernels microconference at Linux Plumbers Conference 2019, Matthias Männich described how the Android project monitors changes to the internal kernel ABI. As Android kernels evolve, typically by adding features and bug fixes from more recent kernel versions, the project wants to ensure that the ABI remains the same so that out-of-tree modules will still function. While the talk was somewhat Android-specific, the techniques and tools used could be applied to other distributions with similar needs (e.g. enterprise distributions).

  Männich is on the Google Android kernel team, but is relatively new to the kernel; his background is in build systems and the like. He stressed that he is not talking about the user-space ABI of the kernel, but the ABI and API that the kernel exposes to modules. The idea is

to have a stable ABI over the life of an Android kernel. He knows that other distributions have been doing this "for ages", but the Android kernel and build system are different so it made sense to look at other approaches to this problem.

- **System-call wrappers for glibc [5]**

  The GNU C Library has long had a reputation for being hostile to the addition of wrappers for new Linux system calls; that has resulted in many system calls being unsupported by the library for years. That situation is changing, though. During the Toolchain microconference at the 2019 Linux Plumbers Conference, Maciej Rozycki talked about glibc's new attitude toward system-call wrappers, but also served notice that there is still significant work to do for the addition of any new system call.
  Rozycki, who put together the talk with Dmitry Levin, is not the person doing most of this work. He was, instead, "delivering a message from Florian Weimer", who was unable to attend the event.

  For those who might appreciate a bit of background: applications running in user space do not call directly into the kernel; instead, they will call a wrapper function that knows how to invoke the system call of interest. If nothing else, the wrapper will place the system-call arguments in the right locations and do whatever is necessary to invoke a trap into kernel mode. In some cases, the interface implemented by the wrapper can be significantly different from what the kernel provides.

- **Many uses for Core scheduling [6]**

  Some new kernel features are welcomed by the kernel development community, while others are a rather harder sell. It is fair to say that core scheduling, which makes CPU scheduling harder by placing constraints on which processes may run simultaneously in a core, is of the latter variety. Core scheduling was the topic of (at least) three different sessions at the 2019 Linux Plumbers Conference. One of the most interesting outcomes, perhaps, is that there are use cases for this feature beyond protection from side-channel attacks.

- **5.4 Merge window, part 1 [7]**

  As of this writing, 9,632 non-merge changesets have been merged for the 5.4 kernel. This merge window is thus off to a strong start. There has been a wide range of changes merged across the kernel tree, including vast numbers of cleanups and fixes.

-

Following Buggy AMD RdRand, The Linux Kernel Will Begin Sanity Checking Randomness At Boot Time [8]

> The Linux kernel will begin doing a basic sanity check of x86_64 CPUs with the RdRand instruction to see if it's at least returning "random looking" data otherwise warn the user at boot time. This stems from a recent issue where AMD's RdRand behavior with some hardware (particularly, buggy motherboards) could have borked RdRand issues.

- **Google Is Uncovering Hundreds Of Race Conditions Within The Linux Kernel [9]**

> One of the contributions Google is working on for the upstream Linux kernel is a new "sanitizer". Over the years Google has worked on AddressSanitizer for finding memory corruption bugs, UndefinedBehaviorSanitizer for undefined behavior within code, and other sanitizers. The Linux kernel has been exposed to this as well as other open-source projects while their newest sanitizer is KCSAN and focused as a Kernel Concurrency Sanitizer.

- **Kernel Concurrency Sanitizer (KCSAN) [10]**

```
We would like to share a new data-race detector for the Linux kernel:
Kernel Concurrency Sanitizer (KCSAN) --
https://github.com/google/ktsan/wiki/KCSAN  (Details:
https://github.com/google/ktsan/blob/kcsan/Documentation/dev-tools/kcsa
To those of you who we mentioned at LPC that we're working on a
watchpoint-based KTSAN inspired by DataCollider [1], this is it (we
renamed it to KCSAN to avoid confusion with KTSAN).
[1] http://usenix.org/legacy/events/osdi10/tech/full_papers/Erickson.po
In the coming weeks we're planning to:
* Set up a syzkaller instance.
* Share the dashboard so that you can see the races that are found.
* Attempt to send fixes for some races upstream (if you find that the
kcsan-with-fixes branch contains an important fix, please feel free to
point it out and we'll prioritize that).
There are a few open questions:
* The big one: most of the reported races are due to unmarked
accesses; prioritization or pruning of races to focus initial efforts
to fix races might be required. Comments on how best to proceed are
welcome. We're aware that these are issues that have recently received
attention in the context of the LKMM
(https://lwn.net/Articles/793253/).
* How/when to upstream KCSAN?
Feel free to test and send feedback.
Thanks,
-- Marco
```

Linux Google Security

Source URL: http://www.tuxmachines.org/node/128832

Links:
[1] http://www.tuxmachines.org/taxonomy/term/63
[2] http://www.tuxmachines.org/taxonomy/term/120
[3] http://www.tuxmachines.org/taxonomy/term/59
[4] https://lwn.net/Articles/800452/
[5] https://lwn.net/Articles/799331/
[6] https://lwn.net/Articles/799454/
[7] https://lwn.net/Articles/799425/
[8] https://www.phoronix.com/scan.php?page=news_item&px=Linux-RdRand-Sanity-Check
[9] https://www.phoronix.com/scan.php?page=news_item&px=Google-KCSAN-Sanitizer
[10] https://lkml.org/lkml/2019/9/20/394