Kernel: LWN and Phoronix Articles About Latest Discussions and Linux Developments

By *Rianne Schestowitz*
Created *27/02/2020 - 6:53am*
Submitted by Rianne Schestowitz on Thursday 27th of February 2020 06:53:18 AM Filed under Linux [1]

- **Filesystem UID mapping for user namespaces: yet another shiftfs [2]**

  The idea of an ID-shifting virtual filesystem that would remap user and group IDs before passing requests through to an underlying real filesystem has been around for a few years but has never made it into the mainline. Implementations have taken the form of shiftfs and shifting bind mounts. Now there is yet another approach to the problem under consideration; this one involves a theoretically simpler approach that makes almost no changes to the kernel's filesystem layer at all.

  ID-shifting filesystems are meant to be used with user namespaces, which have a number of interesting characteristics; one of those is that there is a mapping between user IDs within the namespace and those outside of it. Normally this mapping is set up so that processes can run as root within the namespace without giving them root access on the system as a whole. A user namespace could be configured so that ID zero inside maps to ID 10000 outside, for example; ranges of IDs can be set up in this way, so that ID 20 inside would be 10020 outside. User namespaces thus perform a type of ID shifting now.

  In systems where user namespaces are in use, it is common to set them up to use non-overlapping ranges of IDs as a way of providing isolation between containers. But often complete isolation is not desired. James Bottomley's motivation for creating shiftfs was to allow processes within a user namespace to have root access to a specific filesystem. With the current patch set, instead, author Christian Brauner describes a use case where multiple containers have access to a shared filesystem and need to be able to access that filesystem with the same user and group IDs. Either way, the point is to be able to set up a mapping for user and group IDs that differs from the mapping established in the namespace itself.

- **Keeping secrets in memfd areas [3]**

  Back in November 2019, Mike Rapoport made the case that there is too much address-space sharing in Linux systems. This sharing can be convenient and good for performance, but in an era of advanced attacks and hardware vulnerabilities it also facilitates security problems. At that time, he proposed a number of possible changes in general terms; he has now come back with a patch implementing a couple of address-space isolation options for the memfd mechanism. This work demonstrates the sort of features we may be seeing, but some of the hard work has been left for the future.
  Sharing of address spaces comes about in a number of ways. Linux has traditionally mapped the kernel's address space into every user-space process; doing so improves performance in a number of ways. This sharing was thought to be secure for years, since the mapping doesn't allow user space to actually access that memory. The Meltdown and Spectre hardware bugs, though, rendered this sharing insecure; thus kernel page-table isolation was merged to break that sharing.

  Another form of sharing takes place in the processor's memory caches; once again, hardware vulnerabilities can expose data cached in this shared area. Then there is the matter of the kernel's direct map: a large mapping (in kernel space) that contains all of physical memory. This mapping makes life easy for the kernel, but it also means that all user-space memory is shared with the kernel. In other words, an attacker with even a limited ability to run code in the kernel context may have easy access to all memory in the system. Once again, in an era of speculative-execution bugs, that is not necessarily a good thing.

- **Revisiting stable-kernel regressions [4]**

  Stable-kernel updates are, unsurprisingly, supposed to be stable; that is why the first of the rules for stable-kernel patches requires them to be "obviously correct and tested". Even so, for nearly as long as the kernel community has been producing stable update releases, said community has also been complaining about regressions that make their way into those releases. Back in 2016, LWN did some analysis that showed the presence of regressions in stable releases, though at a rate that many saw as being low enough. Since then, the volume of patches showing up in stable releases has grown considerably, so perhaps the time has come to see what the situation with regressions is with current stable kernels.
  As an example of the number of patches going into the stable kernel updates, consider that, as of 4.9.213, 15,648 patches have been added to the original 4.9 release ? that is an entire development cycle worth of patches added to a "stable" kernel. Reviewing all of those to see whether each contains a regression is not practical, even for the maintainers of the stable updates. But there is an automated way to get a sense for how many of those stable-update patches bring regressions with them.

  The convention in the kernel community is to add a Fixes tag to any patch fixing a bug introduced by another patch; that tag includes the commit ID for the original, buggy patch. Since stable kernel releases are supposed to be limited to fixes, one would expect that almost

every patch would carry such a tag. In the real world, about 40-60% of the commits to a stable series carry Fixes tags; the proportion appears to be increasing over time as the discipline of adding those tags improves.

- **Finer-grained kernel address-space layout randomization [5]**

  The idea behind kernel address-space layout randomization (KASLR) is to make it harder for attackers to find code and data of interest to use in their attacks by loading the kernel at a random location. But a single random offset is used for the placement of the kernel text, which presents a weakness: if the offset can be determined for anything within the kernel, the addresses of other parts of the kernel are readily calculable. A new "finer-grained" KASLR patch set seeks to remedy that weakness for the text section of the kernel by randomly reordering the functions within the kernel code at boot time.

- **Debian discusses how to handle 2038 [6]**

  At this point, most of the kernel work to avoid the year-2038 apocalypse has been completed. Said apocalypse could occur when time counted in seconds since 1970 overflows a 32-bit signed value (i.e. time_t). Work in the GNU C Library (glibc) and other C libraries is well underway as well. But the "fun" is just beginning for distributions, especially those that support 32-bit architectures, as a recent Debian discussion reveals. One of the questions is: how much effort should be made to support 32-bit architectures as they fade from use and 2038 draws nearer?

  Steve McIntyre started the conversation with a post to the debian-devel mailing list. In it, he noted that Arnd Bergmann, who was copied on the email, had been doing a lot of the work on the kernel side of the problem, but that it is mostly a solved problem for the kernel at this point. McIntyre and Bergmann (not to mention Debian as a whole) are now interested in what is needed to update a complete Linux system, such as Debian, to work with a 64-bit time_t.

  McIntyre said that glibc has been working on an approach that splits the problem up based on the architecture targeted. Those that already have a 64-bit time_t will simply have a glibc that works with that ABI. Others that are transitioning from a 32-bit time_t to the new ABI will continue to use the 32-bit version by default in glibc. Applications on the latter architectures can request the 64-bit time_t support from glibc, but then they (and any other libraries they use) will only get the 64-bit versions of the ABI.

  One thing that glibc will not be doing is bumping its SONAME (major version, essentially); doing so would make it easier to distinguish versions with and without the 64-bit support for 32-bit architectures. The glibc developers do not consider the change to be an ABI break, because applications have to opt into the change. It would be difficult and messy for Debian to change the SONAME for glibc on its own.

- **UEFI Boot Support Published For RISC-V On Linux [7]**

  As we've been expecting to happen with the Linux EFI code being cleaned up before the introduction of a new architecture, the RISC-V patches have been posted for bringing up UEFI boot support.

  Western Digital's Atish Patra sent out the patch series on Tuesday for adding UEFI support for the RISC-V architecture. This initial UEFI Linux bring-up is for supporting boot time services while the UEFI runtime service support is still being worked on. This RISC-V UEFI support can work in conjunction with the U-Boot bootloader and depends upon other recent Linux kernel work around RISC-V's Supervisor Binary Interface (SBI).

- **Linux Kernel Seeing Patches For NVIDIA's Proprietary Tegra Partition Table [8]**

  As an obstacle for upstreaming some particularly older NVIDIA Tegra devices (namely those running Android) is that they have GPT entry at the wrong location or lacking at all for boot support. That missing or botched GPT support is because those older devices make use of a NVIDIA proprietary/closed-source table format. As such, support for this proprietary NVIDIA Tegra Partition Table is being worked on for the Linux kernel to provide better upstream kernel support on these consumer devices.

  NVIDIA Tegra devices primarily rely on a special partition table format for their internal storage while some also support traditional GPT partitions. Those devices with non-flakey GPT support can boot fine but TegraPT support is being worked on for handling the upstream Linux kernel with the other devices lacking GPT support or where it's at the wrong sector. This issue primarily plagues Tegra 2 and Tegra 3 era hardware like some Google Nexus products (e.g. Nexus 7) while fortunately newer Tegra devices properly support GPT.

- **Intel Continues Bring-Up Of New Gateway SoC Architecture On Linux, ComboPHY Driver [9]**

  Besides all the usual hardware enablement activities with the usual names by Intel's massive open-source team working on the Linux kernel, one of the more peculiar bring-ups recently has been around the "Intel Gateway SoC" with more work abound for Linux 5.7.

  The Intel Gateway SoC is a seemingly yet-to-be-released product for high-speed network packet processing. The Gateway SoC supports the Intel Gateway Datapath Architecture (GWDPA) and is designed for very fast and efficient network processing. Outside of Linux kernel patches we haven't seen many Intel "Gateway" references to date. Gateway appears to be (or based on) the Intel "Lightning Mountain" SoC we were first to notice and bring attention to last summer when patches began appearing for that previously unknown codename.

Linux

Source URL: http://www.tuxmachines.org/node/134543

Links:
[1] http://www.tuxmachines.org/taxonomy/term/63
[2] https://lwn.net/Articles/812504/
[3] https://lwn.net/Articles/812325/
[4] https://lwn.net/Articles/812231/
[5] https://lwn.net/Articles/812438/
[6] https://lwn.net/Articles/812767/
[7] https://www.phoronix.com/scan.php?page=news_item&px=UEFI-RISC-V-Linux-Patches
[8] https://www.phoronix.com/scan.php?page=news_item&px=Linux-Kernel-TegraPT-Patches
[9] https://www.phoronix.com/scan.php?page=news_item&px=Intel-Gateway-SoC-ComboPHY