Zack's Kernel News

Chronicler Zack Brown reports on the latest news, views, dilemmas, and developments within the Linux kernel community.

By Zack Brown

Premium Real-Time Mode Support

Vernon Mauery submitted his IBM Real-Time "SMI Free" mode driver for inclusion in the official kernel. The driver would provide a SysFS interface to switch the entire system into and out of IBM Premium Real-Time Mode (PRTM) on x86 hardware that supports it. This functionality would allow user code to disable System Management Interrupts (SMIs), thus allowing the code to decouple itself from the operating system's normal security and resource allocation controls and interact directly with the hardware in otherwise impossible ways.

This feature can do lots of fun things. It can provide an environment for real-time applications to run without fear of being interrupted; it can allow debuggers to do neat stuff; and it can allow really spectacular security breaches on the system.

Vernon's code was accepted without complaint, although Matthew Garrett did have some technical suggestions. Presumably the code will have its own security features to prevent the SysFS interface from being just a gaping security wound in the side of any system that uses it.

Android's Alarm

In other real-time clock (RTC) news, John Stultz noticed an Android alarm driver in the Android source tree that behaved differently from the RTC POSIX clock interface. The Android driver would set an RTC alarm to wake the system out of suspended mode if necessary when the timer expired. The RTC POSIX clock interface, which John had written himself, would also wake the system, but it used a different approach that had some benefits and drawbacks relative to the Android version. The Android driver simplified the relationship to the RTC, at the expense of making it more difficult for user code to do wonky things with the RTC on its own.

John didn't think those drawbacks outweighed the added simplicity of the Android driver, so he thought the driver should be submitted to the official kernel tree. However, the Android driver implemented some technical details in ways that seemed to John to be a bit more invasive into the rest of the kernel. So, he reimplemented what he thought were the key elements of the Android driver, using the more standard interfaces that already existed in the kernel, and submitted the patch for review.

No one had any objection, though Richard Cochran offered some technical feedback. The Android developers might prefer to work with John to incorporate his changes into their code. More likely, however, John's version will just go into the official kernel tree, and the Android developers will either switch to that or keep maintaining their own driver. Time will tell.

Module Auto-Loading Vulnerability

Dan Rosenberg thought one serious security issue for Linux was that unprivileged users could cause kernel modules that were not currently loaded to be loaded automatically by the system.

The existing solutions provided by the various Linux distributions, Dan thought, were not elegant and involved things like blacklists that had to be maintained over time. He wanted to short-circuit the entire vulnerability; his solution was to replace the `modules_disable` sysctl with a new `modules_restrict` sysctl that had more refined features.

The new interface would give the user three options: The first option would be to keep the current behavior. The second would prevent wholesale all unprivileged users from being able to trigger loading a module. The third option would prevent any new kernel modules from loading at all until the system was rebooted.

Overall, the response to Dan's patch was good. Kees Cook in particular, from the Ubuntu Security Team, thought it was great, though he acknowledged it might cause some transition pain for a very small minority of users.

But, Alan Cox vetoed the patch. He said the modules_disable sysctl was an established application binary interface (ABI), which simply couldn't be broken. He insisted that Dan simply extend the `modules_disable` sysctl to include new functionality without breaking the old.

Alan added that he didn't think anyone would ever use `modules_restrict` in the real world and that Dan's patch actually harmed security, rather than enhancing it. Unperturbed, Dan responded that he'd rework the patch as Alan suggested to be an extension of existing functionality.

Dan's patch will probably be accepted eventually, although with Alan's ringing unendorsement, it will probably be reworked a bit to satisfy him.

ZACK BROWN

The Linux kernel mailing list comprises the core of Linux development activities. Traffic volumes are immense, often reaching 10,000 messages in a week, and keeping up to date with the entire scope of development is a virtually impossible task for one person. One of the few brave souls to take on this task is **Zack Brown**.

Bringing the Real-Time Clock to All

One amazing thing about Linux is all the hardware to which it's been ported. This accomplishment was amazing in the old days and, with the modern proliferation of architectures, it's all the more so now. Porting Linux in not just a question of compiling the C source into machine code that can run on each different setup. Often, each piece of hardware has different features, and the kernel has to wrangle all those discrepancies into a consistent interface that gives userland a predictable terrain.

The real-time clock (RTC) is a good example. The clock is often a battery-powered device that keeps time even when the system is turned off. Some RTCs just keep track of the seconds as they pass, whereas others keep a calendar. Some can be used to trigger alarms, in the form of interrupt requests, whereas others can't. Some can trigger alarms but only up to a day in advance, whereas yet others can schedule an alarm up to a century ahead of time.

John Stultz recently submitted a patch to enhance the RTC kernel code, which involved navigating all those intricate conflicts. His specific enhancement addressed the problem that only one process could use the RTC at any given time. The problem was that a given process would have to lock the `/dev/rtc` file so that no other process would be monitoring it when the awaited alarm arrived. John's idea was to create a new abstraction layer between the RTC and userland. The abstraction layer, essentially a separate process, would handle all the RTC monitoring and keep track of the various processes waiting for various alarms. When each alarm arrived, the new layer would feed it to the process that was waiting for it.

Given the hardware variations under consideration and that John had only really tested the code on x86 hardware, he expected at least some criticism from folks. But the response was mild and promising. So, the real-time clock might soon no longer require locking in order to use.

Documenting Staging

Andres Salomon submitted a patch to document some of the development processes associated with the staging trees. The document essentially contains information that Andres would have enjoyed knowing about when he submitted his first staging patch to Greg Kroah-Hartman. As Andres said in the doc, `drivers/staging` is where incomplete drivers live, so they can be used and tested by the widest possible set of users, while still being essentially unfinished and optional. Once the drivers have stabilized, they can be moved out of `drivers/staging` and into their own driver directory.

Of course, that's only part of the story. Andres's doc is useful for anyone writing a new driver or for anyone who wants to know how to get it into the kernel without having to meet the rigorous standards associated with other kernel patch submissions. In practice, however, the `drivers/staging` directory is used in a number of different ways, some of which are more controversial than others. In some cases, drivers existing elsewhere in the kernel are moved into the staging tree as a way to alert their maintainer that something had better be done, and done quick, or that driver will be removed entirely.

In other cases, drivers in the staging directory are virtually essential to the system, but their developers want to retain the option of changing their interfaces without worrying about backward compatibility. This latter case in particular can result in large-scale flame-fests. Users come to depend on the APIs, whereas the developers desperately need to break those APIs. The interaction makes for very exciting times.

Andres's documentation, however, isn't intended to elucidate any of those subtleties. It's just a nice, solid explanation of what to expect if you're writing a driver and want to get it into the kernel quickly.