Zack's Kernel News

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

By Zack Brown

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**.

make MODSIG=1

Mimi Zohar introduced a patch to support ephemeral module signing. The idea is that if you use a private key to sign modules, the kernel can use a public key to ensure that it only loads modules signed by you. Anyone trying to crack into your system by loading a hostile module would find the way blocked.

The problem is that if they do get a certain level of access to your system, they might locate your private key, sign their hostile module with it, and thus crack deeper into your system anyway.

Mimi's code reduces this danger by generating a new public and private key at build time and then discarding the private key after your modules have been built. Anyone poking around for your private key won't find it because there's nothing there to find. To gain access to that particular key, the attacker would have had to have been sniffing around at the time you gave your make command, which is a much more difficult attack to engineer.

The drawback to this approach is that, having once discarded your private key, you'll be in the same boat as the attacker – you won't be able to load any modules you didn't build at the start. So, Mimi's patch is only really useful to people who know in advance that they won't want to modify their system much once it's set up. Anyone building their own server might fall into that category. Ordinary users might decide to use this feature as well, if they don't tend to add new hardware very often or play around with experimental features.

Rusty Russell accepted the patch, although he felt some improvements still could be made. One of Mimi's ideas was to have the key generation be tied to a particular build target rather than a temporary file, but Rusty tends to build his kernels as the root user, so he'd have root-owned files sitting around in his tree after the build had completed.

David Howells also had certain technical objections to Mimi's patch, although not significant enough to prevent the patch from being accepted into the kernel. David thought that some of Mimi's makefile code would confuse the dependencies and cause the Make program to think certain files had already been handled. David also pointed out that the GNU debugger might become confused at a certain point and stop being able to debug the running kernel.

Strange issues. However, it seems that the overall patch is to Rusty's liking, and these other technical problems might end up being solved down the road.

Minimal Toolchain Requirements

The issue of what tools are required to build a Linux kernel is a significant one. The kernel developers want to make sure their code will build with the oldest and most reliable versions of their tool chain that they can manage. If each new version of the kernel relied heavily on bleeding edge features of the latest release of Make or GCC or of some other tool along the chain, users might view it as a little risky.

Bleeding edge features can cause bleeding; they can have undetected bugs, and they can have drawbacks that result if they're removed from the tool in the next release. In this case, all of a sudden you'd have a version of Linux that would *only* build with this particular broken version of the required tool.

Bad scene. The kernel developers want the kernel to be buildable on the very broadest array of systems that exist in the world. Just as there are innumerable ports of Linux to every conceivable piece of hardware, so the kernel folks want the kernel to build on all the innumerable setups that might exist on those innumerable platforms.

The Linux kernel documentation, for example, says that the kernel will build with the decade-old GCC version 3.2, as well as all later GCC versions. This is a very broad base of support. Pretty much any system with a GCC compiler can build pretty much any version of the Linux kernel.

Rob Landley, however, noticed that GCC version 4.2.1 (released in 2012) would not successfully build the 3.7 kernel. He posted the error messages produced by the attempt and suggested updating the kernel documentation to reflect the new lower bounds of GCC versions.

Shaun Ruffell responded, saying a Jan Beulich patch would fix the incompatibility and allow the kernel to compile with the old version of GCC again. It was too late to fix in Linux 3.7, but Shaun was hopeful that the fix would get into the tree before Linux 3.8 came out. Jan's patch only changed about three

lines of code and seemed to Shaun like an obvious fix. Rob liked the patch and thought it was preferable to abandoning a decade's worth of GCC versions.

Rob also said that the stable 3.7 kernel might benefit from it as well. He CC'd the stable maintainers on the email, but Greg Kroah-Hartman replied, saying that they could only take patches that had made it into Linus Torvalds's tree already; otherwise, the risk was too great that a fix would go into the stable tree and not the development tree. However, Greg asked Rob to let him know when the fix made it into Linus's tree, and he'd apply it then.

So, it looks as though GCC 3.2 will continue to be supported, at least for most kernel versions.

The Quest for Fire

The ability to "hot plug" devices into and out of a running system has been a goal of Linux development for many years. Of course, the hardware has to be capable of such a thing, so certain platforms are highly unlikely to support full-fledged hot-plugging; however, others do support it in theory, and the kernel continues to inch its way toward supporting it in practice as well.

Toshi Kani recently submitted a series of patches to support hot-plugging system devices implementing the ACPI (Advanced Configuration and Power Interface) specification.

The problem with hot-plugging system devices such as CPUs and RAM chips is that they're typically needed at the very start of the boot sequence. Without them, there's almost no real system to boot.

Toshi's approach was to create a complicated infrastructure to queue up hot-plugging requests and allow them to be performed one at a time. Each request would arrange to power down the particular device and make it ready for removal or make the system ready to have the device plugged in and started up to join the running kernel.

Rafael J. Wysocki had a lot of technical questions for Toshi. To begin, he asked why Toshi's approach was limited to system devices only. It seemed to be applicable in more general-purpose ways.

Toshi replied that his framework could be extended to non-system devices but that the existing methods of hot-plugging non-system devices was actually more flexible than his approach because those methods didn't have to contend with the difficulties of a boot sequence.

Rafael had a number of other technical questions, several of which involved the order in which components had to be removed from the running system and the relationship of that order to other aspects of the system that might have their own effect on any ultimate order of removal. His final question addressed the issue of what problems Toshi's approach was really supposed to solve. Why use this type of framework at all?

Toshi replied that, among other things, his approach targeted a variety of race conditions that could otherwise occur. His framework would ensure that a series of components could be safely removed, without any one of them inadvertently starting up again before the user could remove it.

Another reason Toshi liked his approach was that it handled ACPI failure modes properly. Existing ACPI drivers used `.add` and `.remove` to bring devices online and offline. However, those operations actually have to do with the physical connection of each device into the system, rather than turning them off and on.

In most cases, those two things corresponded, but if the operation failed for any reason, the driver wouldn't be able to detect that fail-

ure and would happily let the user remove a device that was still in use by the system. Toshi's approach wouldn't allow that.

Clearly the discussion will continue, if for no other reason than that Toshi's code would replace other code that's already in the kernel. Can that existing code be fixed up to answer some of the reasons why Toshi felt his approach would be better, or is the real solution to ditch the existing code and go with Toshi's approach, whole hog?

Freeing Unused Memory Pages

Minchan Kim posted a patch adding two new `mvolatile()` and `mnovolatile()` system calls to the kernel. The idea is to allow user programs to alert the kernel to memory pages it no longer needs. The kernel could then free the memory instead of swapping it out to disk.

The issue is complicated by the presence of the `madvise()` system call that performs a similar function, so in his initial post, Minchan had to justify adding two new system calls.

As he pointed out, `madvise()` really only gives hints to the kernel, so the kernel can use read-ahead and caching techniques to speed things up. On the other hand, `mvolatile()` and `mnovolatile()` allow the identified memory to be discarded entirely and made available to the rest of the system.

Taras Glek and John Stultz had some technical questions and comments. John in particular wasn't convinced that a whole new system call would be needed for what Minchan wanted to accomplish, and because there's no urgency to these features, the debate could linger for some time before being decided yea or nay.

Ongoing 3.5 Stabilization

Herton Krzesinski announced the intention of the Ubuntu kernel team to maintain the 3.5 Linux kernel as a stable tree. He also announced the first release, Linux 3.5.7.1. Herton invited anyone to use this kernel tree for anything they wanted and to contribute fixes to the Ubuntu team. He also gave a link to their wiki [1], which said they expected to drop support of this kernel by the end of March 2014.

Arkadiusz Miskiewicz suggested making this tree an official stable tree, rather than just hosting it on the Ubuntu website. He thought it'd be cool to see it hosted on kernel.org, with tarballs, a Git tree, and the whole nine yards, but as of this writing, this hasn't happened.

Architecture Ports and Naming Issues

Vineet Gupta posted a series of patches to port Linux to the ARC700 processor family. As he described it, "ARC700 is [a] highly configurable and power efficient 32-bit RISC core with MMU. It is embedded in SoCs deployed in TV Set Top boxes, Digital Media Players, all the way to Network-on-Chips." He also announced – or at least hinted at – a forthcoming Linux distribution intended to run on these chips.

Arnd Bergmann offered some encouragement, saying he liked the patches and saw only a couple of issues that might slow down their inclusion into the official tree.

Arnd's first objection was that Vineet's code relied on legacy system calls that had since been replaced. Vineet would have to update his code to use the current interfaces.

Arnd also pointed out that Vineet's code didn't use dynamic hardware detection but instead relied on configuring the entire hardware setup at compile time. Arnd said, "new ports these days are normally able to run on all kinds of hardware and detect the differences by looking at configuration registers (e.g., PCI), asking firmware (Open Firmware, ACPI, …) or by interpreting a device tree that is passed by the boot loader (most embedded systems)."

Gilad Ben-Yossef had some encouraging comments. He said that he'd been using this port for "Ezchip running on top of FPGA in 4 way SMP configuration (support for which will no doubt follow later) and have subjected it to various stress tests (hackbench and such) and it is working up quite well over all." He added that he "wanted to attest the nature of its good behavior and performance on real hardware."

At this point, the conversation completely imploded. Pavel Machek noticed that the "ARC" name – used by Vineet's code – was already used by the Advanced RISC Computing Specification, which dated back to the early 1990s. He also pointed out that `/arch/arc` was a bit too similar to `/arch/arm` and could confuse developers. Pavel suggested some different names for Vineet's project, but Vineet pointed out that the code was named after the actual architecture, and he had no intention of deviating from that. He suggested that everyone's time would be better used if they'd focus on the technical aspects of the code, and not on the naming scheme.

Pavel didn't like this response, and retorted, "Yeah, and it is best use of reviewers time to confuse them with one-letter difference to very popular architecture … and use three letter acronym that was already taken."

Arnd defended Vineet at this point, saying, "A lot of people are familiar with the ARC name, it's been around for decades and has sold billions of CPUs under that name, likely more than x86 or mips (still dwarfed by ARM of course). They've just kept a low profile so far and never tried to upstream their Linux port (unlike their gcc port, which was merged in 1997). I'm sure there are a lot of bad things one can say about Synopsys, but they are doing the right thing here, and calling their CPU architecture by a different name is not going to be helpful to their users."

Pavel was not mollified and took a hard line against doing any more reviews for Vineet's code or helping it get merged in any way. Vineet said he didn't understand what the fuss was about. He said, "I won't go around asking ARC name to be changed because someone's Tab completion doesn't work anymore."

The discussion ended at that point, without any real clarity. It seems as though Vineet's code is not a problem, and it's unclear how much of an issue the naming scheme will turn out to be.

With big-time kernel people like Arnd and Pavel disagreeing on the importance of that particular conflict, it's hard to say what will happen eventually. Undoubtedly, Vineet's employers will resist abandoning their brand. But the open source world isn't known for caring much about corporate branding in their source code. ■■■

INFO

[1] Ubuntu wiki, ExtendedStable: *https://wiki.ubuntu.com/Kernel/Dev/ExtendedStable*