Zack's Kernel News

Kernel spam

In a bizarre twist, the mailing list server that hosts the linux-kernel mailing list found itself listed on Spamcop! The thing that makes linux-kernel such a great list for Linux development has often led to problems like this: the linux-kernel admins prevent non-subscribers from posting to the list.

This is the only way to ensure that users have a clear and easy path to report bugs to the kernel developers, and it requires constant vigilance on the part of the list admins, who constantly create pattern-matches for spam, so the readers of the list are not inundated. By and large, they do a good job of recognizing spam and blocking future instances of that particular message.

Unfortunately, some spam does get through, and this has resulted in the list server being listed as a spam producer by Spamcop. This led to many complaints to Spamcop, and vger.kernel.org was removed from their list within a day of the problem being noticed.

Unfortunately, the problem of what to do about spam seems to be growing exponentially. Solutions like Spamcops and other blacklists suffer from the problem of false-positives as illustrated here; while Bayesian filters and hand-tuned pattern matching are not 100% effective.

The best hope may be in modifying the SMTP protocol so that the origin of each email can be determined. Several projects are underway that pursue this idea, but so far none are in wide use. ■

Say no to binary-only

Attempts to make it easier for binary modules are being shot down early. Even the folks making the suggestions preface their remarks with, "I am not advocating binary modules."

Someone made one such proposal, suggesting mechanisms that would allow regular users to help debug modules without access to the sources. The proposal inspired Bartlomiej Zolnierkiewicz to list off his reasons for hating binary-only kernel modules:

- They indirectly slow down kernel development by generating false bug reports that can only be discarded when the existence of the binary-only module is known – a detail that does not always come out right away.
- Vendors that provide these modules claim to support Linux by virtue of providing the module, while at the same time not supporting the spirit of Linux at all.
- Users might tend to blame Linux itself and not the closed-source driver, if their system breaks.
- Since a binary-only driver is the equivalent of a modified kernel in which the modifications are secret, user's shouldn't expect to get help from the public forum where everyone gives their skill and expertise away for free.

Bartlomiej went on to suggest that someone should create a new mailing list just for discussion of binary-only modules, so no one would have to read about it on linux-kernel. ■

INFO

The Kernel Mailing List comprises the core of Linux development activities. Traffic volumes are immense 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 that take on this impossible task is Zack Brown.

Our regular monthly column keeps you up to date on the latest discussions and decisions, selected and summarized by Zack. Zack has been publishing a weekly digest, the Kernel Traffic Mailing List for several years now, reading just the digest is a time consuming task.

Linux Magazine now provides you with the quintessence of Linux Kernel activities straight from the horse's mouth.

Getting new features

New features are still going into the 2.4 kernel, with a proviso that a testing period in the 2.6 tree is pretty much required. Sam Hopkins tried to get his ATA over Ethernet (AoE) driver into the 2.4 kernel, but Marcelo Tosatti wouldn't even look at it until he'd made a formal proposal on the linux-kernel mailing list.

Sam did this, and the driver appeared to pass muster with kernel regulars like Jeff Garzik, but routing the driver through the 2.6 tree still appeared mandatory.

Sam explained that he'd started development when 2.6 had just been forked, and that he'd decided to develop under 2.4 because he wanted a fully stable system. He added that a port to 2.6 was already underway; and Marcelo told him that would be the best way to get into 2.4, possibly even in time for 2.4.27. ■

Applying limits

The relationship between the Linux kernel and the standard C library, like the relationship between the kernel and the GNU C compiler, is one of strange interdependency and occasional antagonism.

In April, Ulrich Drepper, the libc maintainer, pointed out that in recent 2.6 kernels it was possible for an unprivileged user to exhaust all the message-queue slots, so that no other user could create any. A system-wide limit did exist in the kernel already, but this would apparently not stop individual users from monopolizing the entire pool.

Marcelo Tosatti, the 2.4 kernel maintainer, backed Ulrich up on this, and pointed out that the problem extended beyond just message-queues. Apparently it was possible for unprivileged users to monopolize the list of pending signals, and a bunch of other data; however, although Marcelo agreed the problem was significant, he believed that the kernel did not support the infrastructure to limit these things on a per-user basis.

At first he suggested the old 'user-beans' project, but Andrew Morton, the 2.6 kernel maintainer, pointed out that kernel/user.c provided some infrastructure that could handle this; after that, Marcelo coded up a new RLIMIT_SIGPENDING limit, which could limit pending signals on a per-user basis. ■

Organizing Architecture

Kernel configuration can be confusing at times, but even more so for the developers organizing those options into a friendly system for users. The struggle to keep all the kernel configuration options orderly is always an uphill battle. Some decisions are made early in a project's history, before the true configuration details are known; then other projects imitate that one, while the first is later reorganized differently.

Then the question of precedent comes up, and which method represents a legitimate practice; meanwhile other projects come into the kernel with their own configuration methods, and the whole thing drifts gradually between the two polls of total chaos, and a pure ideal.

Recently the PCI MSI (Media Control Interface) configuration options were consolidated into one place by Bjorn Helgaas, removing them from the i386 and multiple other architecture-specific locations. This was on the whole well received, but because of differences between architectures, there is the possibility that the configuration options might be interpreted differently by the users of different hardware.

For example, the term 'vector' is used to identify the MSI option in the kernel configuration system, while in the ia64 that term is also used to identify external interrupts; leading some folks to believe that ia64 machines require this option to be enabled, when it is not the case.

Leaving those naming issues aside however, Bjorn still believes that there is plenty of other architecture-specific information that has wound up in the MSI code itself instead of the /arch directory where it belongs.

Tom Long Nguyen and others from Intel are working on cleaning up at least part of that problem, in order to enable MSI on the x86_64 architecture.

Back-lashing the stack

The non-executable stack is an area of an executable program that is used to store data used by that program.

One way of cracking a system has been for a hostile user to overrun a vulnerable buffer, implant some executable code in that program's stack, and then cause the program to execute that code.

The solution has been to make a program's stack non executable. Users are now requesting specific exceptions to this rule. It's already possible to allow an executable stack in a binary prior to invocation, but there are apparently also cases where you might want to change this permission during run-time.

H. J. Lu wanted to invoke a program that would access a dynamic library that had an executable stack. This failed, although partly this may have been due to a bug in some of the vendor-specific kernel modifications in H. J. Lu's Linux distribution.