

# ZACK'S KERNEL NEWS

## SECURITY ISSUES IN OPEN DEVELOPMENT

An interesting puzzle confronts anyone developing open source software: given that development takes place on open mailing lists, how can security problems be found and fixed before exploits are created and distributed? The problem boils down to one of culture. In the Linux world, some developers favor total openness. Let all exploits be published immediately, and the fixes will follow! Other developers see this as a dangerous approach, and feel that security problems should be kept secret to some extent while fixes are developed behind closed doors.

The question came up on the linux-kernel mailing list this past January, and it was the subject of a huge debate. Apparently there's a mailing list, called vendor-sec, which has been used for a lot of kernel security reports. Originally, as the name implies, it was intended to be used for general distribution security, but its focus shifted toward the kernel out of necessity, because no other kernel-specific place existed (discounting the main linux-kernel list itself). On vendor-sec, folks reporting the problems

would be able to set *embargos* on disclosure, to give them time to prepare fixes. These embargos could go on for days or weeks (perhaps even months), and the timing of disclosure itself began to have a political flavor.

Linus Torvalds wanted no part of this type of security list. In fact, as he's said, he would prefer total openness about security flaws in the kernel. But, on the urging of big-time folks like Alan Cox and Marcelo Tosatti, he agreed to compromise a bit; and Chris Wright has created the security@kernel.org mailing list for this purpose. The list itself is effectively invitation-only, but no disclosure embargos will exist. As Linus has said, there might be some delays in disclosure, but these would be measured at the most in days and would wait only for a fix to be ready, not for the convenience of any other party. In fact, as I write this, the true rules of the list are still in flux.

So in the case of the Linux kernel, it seems there will be a closed area of development in which security problems are dealt with more secretly than elsewhere, but this could change over time.

## DOSFS

Linux support for DOSFS has traditionally been governed by the various design mistakes left by Microsoft; and in recent years, by the threat that Microsoft would alter the format in such a way as to make Linux support more difficult. NTFS is an example of this – its mount-points are still effectively read-only under Linux, after years of development.

H. Peter Anvin is trying to navigate these treacherous waters in order to add support for reading and setting FAT filesystem attribute bits. He created a set of new ioctls to accomplish this. One might ask, as Nicholas Miell did, why H. Peter used the strongly deprecated ioctl interface for this when an xattr would seemingly be more appropriate. The reason, apparently, is that Microsoft might add NTFS-style xattrs to FAT in the future, making continued Linux support difficult if Linux has already overloaded the xattr mechanism.

The issue of DOSFS support is inherently controversial, involving the choice between various evils all for the sake of the poorly designed and extremely limited DOSFS.

The Linux kernel mailing list comprises the core of Linux development activities. Traffic volumes are immense, often reaching ten thousand messages in a given 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.



Our regular monthly column keeps you abreast of the latest discussions and decisions, selected and summarized by Zack. Zack has been publishing a weekly online digest, the Kernel Traffic newsletter for over five years now. Even reading Kernel Traffic alone can be a time consuming task. Linux Magazine now provides you with the quintessence of Linux Kernel activities, straight from the horse's mouth.

## THE KERNEL DEVELOPMENT MODEL

A new favorite topic of conversation among kernel developers appears to be how kernel development should proceed, now that the even/odd stable/development ideas have been abandoned. Many developers are unhappy with the current situation, saying it is impossible to trust recent kernels; while others are extremely pleased, saying it's great to be able to continue contributing new ideas and not wait for a development series.

Andrew Morton, clearly one of the folks making deci-

sions about development strategy, has acknowledged the difficulties of the current system. Few people, it seems, are testing the 2.6-rc releases, while point releases are tested heavily. Point releases are therefore less stable than release candidates, because bugs discovered in each point release are fixed during the candidate cycle, even as new features are added or enhanced.

One possibility that seems to be bubbling closer to the surface is that the 2.6.x.y numbering scheme could be used to stabilize point releases, even as development continues toward the next point release. Clearly this is an idea with some support, although it's still too early to say for sure.