Notícias do Kernel

Problemas de segurança no modelo aberto de desenvolvimento

A esfinge do mal apresenta uma charada interessante a qualquer um que enverede pelo caminho do desenvolvimento de software com o código aberto: uma vez que esse desenvolvimento se dá na verdadeira "feira livre" que é a Internet, com sites abertos a todos e listas de discussão públicas, como evitar que *exploits* sejam criados e distribuídos antes que as falhas de segurança no software sejam corrigidas? Afinal, a discussão é aberta a todos – até mesmo aos mal-intencionados. O ponto crucial do problema não é outro senão cultural. No mundo do Linux, a maioria dos desenvolvedores tem como bandeira e missão a transparência total: "Deixemos que os *exploits* sejam publicados imediatamente, pois logo atrás deles vêm as correções!" Mas uma minoria, na verdade não muito pequena, de desenvolvedores encara essa abordagem como sendo de alto risco e acha que os problemas de segurança devem ser mantidos em segredo até que as correções possam ser elaboradas – de preferência a portas fechadas!

A questão surgiu na lista de discussão do kernel do Linux (LKML – *Linux Kernel Mailing List*), durante o mês de janeiro e foi objeto de intenso e acalorado debate. Aparentemente há uma outra lista de discussão chamada *vendor-sec*, que vem sendo usada para divulgação de inúmeros relatórios sobre falhas de segurança no kernel. Originalmente, como o nome indica, a lista era destinada a discussões mais gerais sobre a segurança das distribuições (*vendors*). Percebe-se hoje que seu foco foi desviado e seus assinantes discutem mais sobre a segurança e as falhas do kernel do que sobre qualquer outro assunto – tudo motivado pela necessidade, já que além da LKML nenhum outro fórum existe para se discutir o assunto. Na *vendor-sec*, os camaradas que revelarem falhas têm o poder de "*embargar*" sua divulgação ao público, ganhando assim tempo suficiente para preparar as correções necessárias. Como esses "embargos" podem durar desde poucos dias até muitas semanas (às vezes meses), o período de veto passou a vagar sem direção, ao sabor das ondas da politicagem.

Linus Torvalds nunca quis nenhum tipo de censura na divulgação de qualquer coisa e sempre execrou esse tipo de lista. Não é nenhum segredo o fato dele preferir transparência total quando o assunto são falhas no kernel, incluindo as de segurança. Entretanto, sob os apelos de peixes grandes do porte de Alan Cox e Marcelo Tosatti, Torvalds cedeu um pouquinho. Chris Wright, então, criou a lista *security@kernel.org* para esse propósito. Nem tente se inscrever: a lista é um clube exclusivo, é preciso um convite de um participante para poder entrar. Por outro lado, nenhum tipo de censura ou "embargo" será tolerado. Linus quer assegurar que, mesmo que uma falha de segurança demore alguns dias para ser divulgada ao grande público, a revelação espere *apenas* pelas correções necessárias e nada mais – evitando, dessa forma, que esse tempo seja manipulado por terceiros para satisfazer a seus próprios interesses, nem sempre honestos.

Portanto, no caso do kernel Linux, parece que finalmente haverá uma área restrita de desenvolvimento na qual os problemas de segurança serão tratados com absoluto sigilo, coisa inexistente em qualquer outro lugar até hoje. Mas como tudo no mundo Linux, esse comportamento deve mudar ao longo do tempo. A lista *vendor-sec* vai continuar a existir e provavelmente será a mais popular das duas, mesmo porque é pública. Se isso realmente acontecer, a lista *security@kernel.org* deve lentamente morrer ou, ainda, ficar cada vez mais parecida com a *vendor-sec*. ■

O modelo de desenvolvimento do kernel

Outro assunto que já virou a "novela das oito" entre os desenvolvedores do kernel parece ser como o desenvolvimento do kernel deve ser conduzido, já que o modelo de numeração tradicional (versões ímpares para séries de desenvolvimento e pares para a versão estável) foi abandonado – leia os "capítulos anteriores" nas edições passadas da Linux Magazine, nesta coluna. Muitos desenvolvedores estão infelizes e insatisfeitos com a situação atual, dizendo que essa decisão torna impossível confiar em kernels mais recentes. Enquanto isso, outros estão extremamente empolgados. Segundo essa facção, é muito bom poder contribuir com novas idéias para o kernel sem ter que esperar uma nova série ímpar de desenvolvimento.

Andrew Morton, um dos que claramente tomam decisões a respeito da estratégia de desenvolvimento, reconheceu as dificuldades do sistema atual. Poucas pessoas estão testando as versões *2.6-rc*, enquanto as versões finais são pesadamente testadas. Absurdamente, isso está levando a um paradoxo: as versões "ponto X" estão ficando menos estáveis que os *release candidates*, porque as falhas descobertas nas versões finais são corrigidas no ciclo de desenvolvimento da próxima versão – mesmo com novos recursos sendo incluídos ou aprimorados.

Uma das possibilidades que estão surgindo é usar um esquema de numeração do tipo 2.6.x.y (já não vimos isso antes?) até que a versão final se estabilize, deixando as novas adições e melhoramentos para a versão *release candidate* (também chamada *RC*). William Lee Irwin III já avisou que adotará arbitrariamente essa estratégia se ninguém mais o fizer. Podemos dizer que é, claramente, uma idéia que conta com algum apoio, embora seja muito cedo para termos qualquer certeza. ■

DOSFS

O suporte ao DOSFS no Linux sempre foi norteado pelos diversos erros de projeto cometidos pela Microsoft e, em época mais recente, pela possibilidade da gigante de Redmond alterar o formato de maneira a dificultar ou mesmo impedir que o Linux entenda seus sistemas de arquivos. O NTFS é um bom exemplo – mesmo depois de anos de desenvolvimento, volumes NTFS só podem ser confiavelmente montados para leitura.

Peter Anvin está tentando navegar por essas águas traiçoeiras a fim de permitir a leitura e a alteração dos atributos de arquivo no sistema de arquivos FAT. Para isso, criou um conjunto de novos *ioctls* (*Input/Output Controls*) para dar cabo da tarefa. Você deve estar se perguntando, como Nicholas Miell já fez, por que Peter usou a interface *ioctl* – não só não-recomendada como já bastante obsoleta – quando um *xattr* teria sido muito mais apropriado. A razão, aparentemente, é que a Microsoft deve adicionar *xattrs* como os do NTFS ao FAT num futuro não muito distante. Isso tornaria mais difícil a manutenção desse sistema de arquivos no Linux, pois o mecanismo *xattr* já está sobrecarregado.

Contudo, parece que há maneiras de contornar os problemas de nomes com o *xattr*. Ocorre que Peter Anvin acha o mecanismo de *xattr* uma "bela porcaria", um erro levado adiante. O problema é bastante controverso por exigir a escolha de um caminho a tomar, mesmo havendo aspectos ruins em qualquer deles – afinal, o DOSFS, além de ser extremamente limitado, foi desenvolvido sem qualquer compromisso com a qualidade. É natural que, à procura por uma maneira de lidar com esse assunto espinhoso, cada um veja as coisas à sua maneira. ■

DebugFS

O *DebugFS* parece já ter seu assento no parlamento kerniano, embora sua entrada no processo de desenvolvimento tenha sofrido uma viravolta interessante. Uma das muitas razões pelas quais projetos como o *SysFS* e o *Udev* são tão populares é o fato de tratar das complexidades do sistema sem a intervenção do usuário – ou, pelo menos, esconder dele essa dificuldade. Resumindo, o *SysFS* e o *Udev* substituem com vantagens as implementações complexas de idéias semelhantes mais antigas. A grande massa de arquivos gerados por *ioctls* no `/proc` está entre as outras velharias obsoletas herdadas dos ancestrais sistemas Unix. O *SysFS* e similares são vistos como uma maneira de escapar desses lodaçais.

Mas algo vai contra a maré. Greg Kroah-Hartman declarou que "a anarquia governa o *DebugFS*!", encorajando seus camaradas a usá-lo onde quiserem e não se importarem por entupir o espaço de nomes de outra pessoa (embora reconheça que essa "pessoa" talvez precise autorizar a bagunça em alguns casos). Greg acha que não há possibilidade de haver colisões no *DebugFS*, nunca.

Só podemos esperar e ver se sua postura continuará a mesma daqui a um ano ou dois. Deve ser empolgante defender a absoluta anarquia em um sistema de arquivos depois de anos de restrições e obsolescências. Greg está ciente disso tudo e dos perigos que podem se abater sobre todos – ele foi um dos principais desenvolvedores do *Udev*, uma tentativa bastante popular de reinar nas pradarias sem lei nem ordem do diretório `/dev`. Talvez seja mesmo seguro usar o *DebugFS* de acordo com as paixões de cada um, mas também há o risco de que os tempos vindouros transformem a tão falada liberdade num monstro grande, feio e cabeludo. Veremos. ■

A situação do *iswraid* na árvore 2.4

O driver *iswraid* (*Intel Software RAID*) teve uma vida dura nos últimos tempos. Em outubro passado, Alan Cox e Jeff Garzik tinham declarado o driver pronto para o kernel 2.4 e sua inclusão na árvore de código era iminente. Entretanto, por motivos não muito claros, ele ficou de fora das versões oficiais e até hoje não figura como driver oficial.

Em janeiro seu autor, Martins Krikis, pediu a Marcelo Tosatti, mantenedor da série 2.4, uma posição definitiva. Em resposta, Tosatti disse que não havia muita chance de aceitar o driver, pois apenas correções importantes de falhas na série 2.4 seriam aceitas a partir de então. Novamente Jeff interveio, dizendo que o patch já estava pronto há algum tempo, e Marcelo cedeu. Entretanto parece que, depois de tudo, Martins quer fazer alguns ajustes antes de enviar o driver; portanto, o *iswraid* deve entrar no kernel apenas na versão 2.4.30 – a encarnação final e definitiva da série 2.4.

Marcelo vem tentando fechar a porta desse kernel desde que foi dada à luz a versão 2.6.0, com sucesso parcial. Com a árvore 2.6 em rápido desenvolvimento, talvez Marcelo não tenha outra alternativa senão liberar novas versões do 2.4 para o pessoal que precisa dos recursos do 2.6 "transplantados" para o kernel 2.4. Mas esse é um mar de águas traiçoeiras pelo qual Marcelo claramente gostaria de não ter de navegar. ■

Ajuste fino do kernel com algoritmo genético?

No desenvolvimento do kernel, é difícil prever quais idéias serão recebidas de braços abertos e quais serão enxotadas sob gargalhadas. Entretanto, eu apostaria todas as minhas fichas no fato de que a proposta de Jake Moilanen não seria levada a sério. Bem, perdi a aposta. O código de Jake é uma biblioteca de algoritmos genéticos no kernel que procura ajustar o *scheduler* (software que distribui os recursos do sistema entre os processos) do sistema – por sinal, uma área em que tradicionalmente os desenvolvedores humanos são mais qualificados.

É verdade que nenhum desenvolvedor "da pesada" deu apoio quando Jake postou seu *patch*, mas tampouco tentaram escorraçar a idéia da mesma forma que fariam com, por exemplo, um interpretador Lisp integrado ao kernel. Pelo contrário: muitos assinantes da lista sugeriram coisas bastante interessantes. Parece que essa idéia tem futuro. ■

Sobre o autor

*A lista de discussão **linux-kernel** é o centro do desenvolvimento do kernel Linux. O volume de tráfego é imenso e se manter em dia com todo o processo é uma tarefa humanamente impossível. Uma das poucas pessoas corajosas o suficiente para aceitá-la é Zack Brown, que já publica um "resumão semanal" das discussões, na forma da lista **kernel-traffic**.*

Esta coluna mensal manterá você informado sobre as últimas novidades e decisões relativas ao kernel, selecionadas direto da fonte e resumidas pelo próprio Zack.