Impacto do robustecimento de sistemas no desempenho de serviços Internet

Tiago Pedrosa, Rui Pedro Lopes, Nuno Rodrigues, José Rufino

{riftman,rlopes,nuno,rufino}@ipb.pt

Instituto Politécnico de Bragança, Campus Santa Apolónia - Apartado 1038
5301-854 Bragança, Portugal

Resumo

O recurso a medidas preventivas, no âmbito da Segurança de Sistemas Informáticos, tem uma importância cada vez maior. Neste contexto, os mecanismos do tipo *Role-Based Access Control* (RBAC) representam uma das abordagens mais promissoras. Todavia, para que sejam largamente aceites, é desejável que i) tenham um impacto reduzido no desempenho dos sistemas e aplicações, ii) sejam escaláveis e iii) sejam facilmente administráveis.

Para além da abordagem RBAC, existem outros mecanismos complementares, para a detecção e contenção de erros. Por exemplo, mecanismos como o PaX intervêm ao nível da gestão de memória efectuada pelo núcleo do sistema operativo, tornando um sistema virtualmente imune a situações do tipo *buffer overflow*.

Neste artigo descreve-se uma aplicação (e respectiva avaliação) do mecanismo RBAC da plataforma *grsecurity*[1], baseado em listas de controle de acesso (ACLs), em conjunção com a plataforma PaX. Em traços gerais, e para a gama de testes utilizada, concluímos que, para serviços nos quais a memória é partilhada entre os vários fios de execução que fazem o atendimento dos pedidos, o impacto do *hardening* no desempenho dos serviços é mínimo; já em situações em que a memória não é partilhada, a intervenção dos mecanismos de validação PaX torna-se evidente, prejudicando o desempenho global dos serviços.

1 Introdução

A sociedade actual encontra-se cada vez mais dependente de aplicações e de sistemas que requerem conectividade permanente. Neste sentido, as preocupações com a segurança têm vindo a crescer, uma vez que o número de ataques tem aumentando todos os anos.

Todavia, do ponto de vista da disponibilização de serviços, a preocupação principal tem sido, quase exclusivamente, a optimização do desempenho das aplicações e do sistema em geral. Por este motivo, poucas empresas estão dispostas a investir com o objectivo de aumentar os níveis de segurança, acabando, até, por negar a própria existências de falhas de segurança.

Além disso, a complexidade acrescida que será necessário efectuar para tornar o sistema mais robusto, desde a compreensão das vulnerabilidades até à implementação de mecanismos de robustecimento[2] de sistemas, acaba, por vezes, por desmotivar a sua adopção pelos técnicos.

Devido à enorme diversidade e número de ataques, lidar com cada um deles, como um caso isolado é impraticável.

A alternativa é adoptar medidas mais gerais de prevenção, detecção ou contenção, o que poderá, ainda assim, representar uma tarefa complicada, devido ao constante aparecimento de novos tipos de ataques e de novas soluções.

De forma a simplificar esta tarefa, alguns sistemas tentam prever, detectar e conter todas as situações suspeitas. Estes sistemas baseiam-se no conhecimento do funcionamento normal das aplicações, pelo que tudo o que caia fora desse padrão é considerado uma

[1] `http://www.grsecurity.net`
[2] Do inglês: *hardening*

anomalia. Por exemplo, algumas destas soluções incluem um método automático para, após todo o sistema estar configurado e os serviços a correr, construir uma base de dados com os padrões normais do sistema, dos processos e utilizadores.

A inclusão destes mecanismos de protecção pode, no entanto, afectar o desempenho do sistema. Este artigo compara o desempenho de vários serviços internet, nomeadamente, HTTP, SMTP, POP e LDAP entre duas configurações semelhantes, uma instalada com sistemas de *hardening* e outra sem qualquer protecção adicional.

A secção seguinte faz uma pequena introdução ao *hardening* de sistemas, encontrando-se na secção 3 a configuração de testes. A secção 4 apresenta os resultados dos ensaios. O artigo termina com algumas conclusões.

2 *Hardening*

A presença na Internet é praticamente obrigatória para as empresas e para as organizações modernas. Tipicamente, esta medida passa pela instalação de um servidor de HTTP, a partir do qual se respondem aos pedidos e às ligações feitas por intermédio de clientes como o Firefox, Internet Explorer ou outros.

Dependendo da dimensão e, por vezes, da natureza da empresa, o servidor HTTP é complementado com outros serviços como, por exemplo, de correio electrónico e LDAP.

A facilidade de acesso associada às vulnerabilidades encontradas no software fazem com que os serviços possam ser alvo de ataques de segurança, com o objectivo de tirar contrapartidas, obter informação confidencial ou lesar o funcionamento do sistema alvo.

Por estes motivos, é importante dotar os servidores de características que lhes permita sobreviver, na medida do possível, a ataques de segurança. Esta capacidade não é óbvia, dado que existe uma variedade de possibilidades. Na melhor das hipóteses, o sistema deverá tentar prever, detectar e conter as possíveis falhas de segurança.

Uma possibilidade é fazer com que o sistema consiga distinguir o comportamento normal do sistema, baseado no funcionamento espectável dos serviços e das aplicações, do comportamento adoptado quando um ataque está a decorrer.

A caracterização do funcionamento normal de um sistema pode ser difícil de descrever pelo administrador, pelo que um sistema de *hardening* deve permitir detectar, de forma automática, o padrão de funcionamento normal do sistema. A solução deverá permitir construir um padrão de funcionamento global, incluíndo a totalidade do sistema, ou o padrão de funcionamento de um processo específico.

O *hardening* de sistemas baseia-se, tipicamente, na implementação de medidas que permitam levantar restrições de acesso. Estas podem envolver instalar sistemas de protecção de memória (como o PaX[3]), fechar portas de rede, instalar sistemas de detecção de intrusão, instalar *firewalls*, entre outras.

Técnicas baseadas em DAC (*Discretionary Access Control*) [2], utilizadas por alguns sistemas operativos, associam as permissões de acesso a utilizadores. Este sistema baseia-se na restrição de leitura, escrita e execução a objectos conhecidos, sendo, para isso, necessário autenticar o utilizador. Por vezes, as técnicas baseadas em DAC são complementadas ou substituídas por mecanismos de MAC (*Mandatory Accesss Control*) [1]. Mais recentemente, estas abordagens têm vindo a ser substituídas por novas soluções, como, por exemplo, o RBAC (*Role-Based Access Control*) [4, 5].

O RBAC, tal como o DAC, permite restringir o acesso a utilizadores autorizados. No entanto, utiliza o conceito de *roles* para regular a execução de determinadas operações. Adicionalmente, apresenta mecanismos que impedem um utilizador de, quer inadvertidamente quer propositadamente, conceder mais permissões que as definidas pelo administrador. No âmbito do trabalho apresentado neste artigo foi usado o projecto *grsecurity*.

[3] `http://pax.grsecurity.net/`

Existem outros mecanismos de controlo de acesso, como, por exemplo, o LBAC (*Lattice Based Access Control*) [3], baseado no RBAC mas mais complexo.

Uma abordagem que reconhece (ou tenta reconhecer) o comportamento normal do sistema e dos processos permite prever, detectar e conter falhas desconhecidas. Apresenta algumas vantagens que complementam o ciclo de procura de falhas, criação e aplicação de *patchs*, pois o sistema estará sempre sobre controlo mesmo quando um falha existir no sistema ou nos seus processos. Esta abordagem permite que o sistema fique protegido entre o instante de descoberta da falha e o instante de aplicação do *patch* de segurança.

Estas soluções encontram-se disponíveis de raiz em algumas distribuições de *Linux*, sendo as mais conhecidas o *Adamantix*[4] e o *Hardened Gentoo*[5].

2.1 *grsecurity*

O projecto *grsecurity*[6] define vários mecanismos de segurança para o sistema *Linux*, com o grande objectivo de implementar soluções que permitem prever, detectar e conter as possíveis falhas de sistema. Os principais objectivos do projecto incluem ser auto-configurável, fornecer protecção absoluta contra todas as formas de *bugs* de modificação de espaço de endereçamento, reconhecer ACLs ricas e um bom sistema de auditoria onde passam a ser registadas, sempre que possível, as informações do *inode*, o número de dispositivo e informações relativa ao processo pai.

O *grsecurity* inclui os mecanismos definidos no projecto PaX[7], que fornece protecção contra toda a classe de *exploits* de *bugs* de espaço de endereçamento, incluindo o *buffer overflow*. Estes funcionam com base no controlo por software da execução de páginas de memória. Desta forma, mesmo as arquitecturas que não prevêem a *flag* executável encontram-se protegidas. Outro artifício seguido consiste em é usar aleatoriadade para todo o espaço de endereçamento em binários do tipo `ELF`.

A interpretação da ACL é feita em *userspace*, com interacção com o *kernel* atráves de uma entrada na `/proc`. A construção da ACL é efectuada por um mecanismo de aprendizagem que reflecte o funcionamento normal do sistema. Este método de aprendizagem pode ser global ou para um processo particular.

O suporte ao *sysctl* permite que o utilizador possa abilitar ou desabilitar funcionalidades do *grsecurity* sem a necessidade da recompilação do *kernel*. Este suporte permite a integração do *grsecurity* com qualquer distribuição *Linux*.

O componente *grsecurity* fornece muitas das características de segurança do *OpenBSD*. Implementa o `Trusted Path Execution` (TPE)[8], implementa um módulo para o *netfilter*, que nega todas as conexões `TCP/UDP` com destino a portos sem serviços associados. Além disso, foram robustecidas algumas *syscalls*, nomeadamente: `Chroot, Ptrace, Nmap, Link/Symlink, Sysctl`.

3 Cenário de testes

Para ter uma ideia do impacto que um sistema de *hardening* pode ter num servidor de rede foi definido um cenário concreto, em que a única variável foi o *grsecurity*. Neste sentido, utilizou-se uma máquina servidora com *dual-boot*: uma opção de *boot* arranca com o *kernel* normal (*vanilla*) e outra opção com o *patch* do *grsecurity*.

[4] `http://www.adamantix.org`

[5] `http://www.gentoo.org/proj/en/hardened/`

[6] `http://www.grsecurity.com`

[7] `http://pax.grsecurity.net/`

[8] Normalmente um directório é considerado de confiança se o directório pai pertence ao utilizador `root` e não existem permissões de escrita para o groupo nem para outros.

A distribuição utilizada foi o debian unstable, com o *kernel* 2.6.17.7. Os serviços presentes no servidor utilizados para testes foram:

- apache2,
- qmail com pop3,
- slapd.

Em termos de topologia, foi utilizado um *switch* de 100Mbps para a ligação do servidor, 3 máquina clientes (c1-0, c1-1, c1-2) responsáveis por gerar carga e o controlador dos clientes de carga e de recolha dos dados (Figura 1).

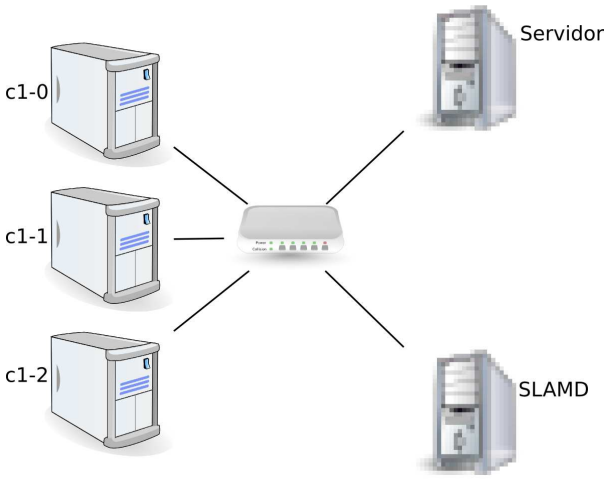

Figura 1: Topologia da configuração de testes.

Configuração das máquinas:

- Servidor de testes: 2x AMD Athlon a 1.5GHz, 1 GB de RAM a 333MHz, 80GB IDE 7200 RPM.
- Servidor slamd: 2x AMD Athlon a 1.5GHz, 1 GB de RAM a 333MHz, 80GB IDE 7200 RPM.
- Máquina cliente: Pentium 4 a 3GHz, 1GB de RAM a 400MHz, 80 GB SATA 7200 RPM.

A carga no servidor é gerada com auxílio à ferramenta SLAMD[9], para os protocolos HTTP, LDAP, SMTP e POP3.

4 Resultados

Todos os testes foram realizados nas mesmas condições, tendo sido seguida uma metodologia que especifica as condições de início, de paragem e o número de clientes.

4.1 Metodologia

Inicialmente, foi necessário definir as regras para o *grsecurity*. Como já referido, o *grsecurity* vem munido de uma ferramenta que permite que o sistema obtenha, de forma automática, o padrão normal de funcionamento e, consequentemente, gerar a ACL do sistema. Neste

[9] `http://www.slamd.com/`

sentido, deixou-se o sistema em modo de aprendizagem durante vários dias, para incluir dados sobre o funcionamento dos processos iniciados pela `cron`.

Para manter o estado da máquina idêntico para todos os testes, a máquina foi reiniciada entre cada novo teste, e, só após a carga média do último minuto da máquina ser aproximadamente 0 (período de estabilização de serviços), é que se iniciava novo teste.

Cada teste foi efectuado com recurso ao SLAMD, para gerar carga no servidor. Este possui um utilitário que permite monitorizar a utilização dos recursos nos clientes usados nos testes. Foi decidido que, se a utilização do processador ultrapassasse os 80% seria necessário inserir um novo nó para gerar carga. Por outras palavras, cada novo nó só iniciaria a geração de carga quando o anterior excedesse os recursos disponíveis.

Para cada serviço foi efectuado o teste, aumentando o número de *threads*, ou clientes, conforme a utilização dos recursos da máquina cliente. O aumento de clientes seguiu uma distribuição 2^n (duplicação do número de pedidos em cada novo teste). A condição de paragem foi não haver um ganho superior a 5%.

Para o caso do POP3 foi necessário seguir um passo adicional. O servidor POP3 utilizado só executa tarefas de inicialização (*caching*, testes de integridade ou outras) após receber o primeiro pedido. Nota-se na carga do processador que, ao receber o primeiro pedido, o servidor fica ocupado com várias tarefas, demorando um tempo proporcional ao número de mensagens na *maildir*. Por este motivo, foi feito um pedido inicial e, após a carga média do servidor atingir valores na ordem dos 0% é que se iniciaram os testes.

Os testes foram efectuados em primeiro lugar sobre o *kernel* com o *grsecurity* e só depois sobre o *kernel* normal. Em termos de protocolo, a sequência de testes foi HTTP, LDAP *search rate*, SMTP, e POP3.

Cada teste teve a duração de 2 minutos, ao final do qual foi calculada a média da carga do servidor no último minuto:

```
watch -n 60 'w | grep load'
```
(1)

Após terminado o teste, este commando também permitia saber quando efectuar o novo teste.

Os valores totais dos pedidos processados eram registados na base de dados do SLAMD, os valores eram fornecidos pelos clientes da respectiva plataforma de teste.

HTTP

O teste de HTTP consitiu em diversos *threads* clientes descarregarem uma página HTML simples. Cada *thread* fez os pedidos ciclicamente, sempre aguardando a recepção da página em cada ciclo.

Os resultados obtidos podem ser vistos na figura 2. No eixo horizontal pode-se ver o número de *threads* cliente. De notar que, ao aumentar de 8 para 16 *threads*, não houve um aumento superior a 5% no número de pedidos satisfeitos, pelo que, de acordo com a metodologia acima indicada, constitui a condição de paragem.

Até 2 *threads* a diferença entre as configurações é desprezável. A partir de 4 *threads* ambos os servidores começam a quebrar, notando-se uma diferença de cerca de 7% entre as configurações analisadas.

LDAP

No caso do LDAP, foi inicializado o servidor OpenLDAP com 14648 entradas. Cada *thread* cliente fez uma pesquisa do tipo:

```
(ObjectClass=*)
```
(2)

O número máximo de resultados da pesquisa foi limitada a 500, dado ser este o valor configurado no servidor.

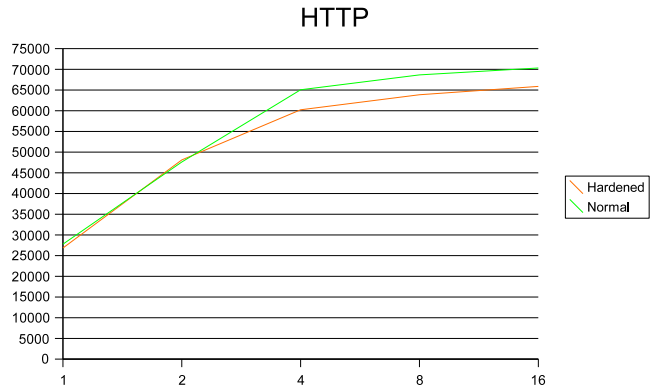

Figura 2: Pedidos HTTP.

Os resultados obtidos podem ser vistos na figura 3. No eixo horizontal encontra-se rep-

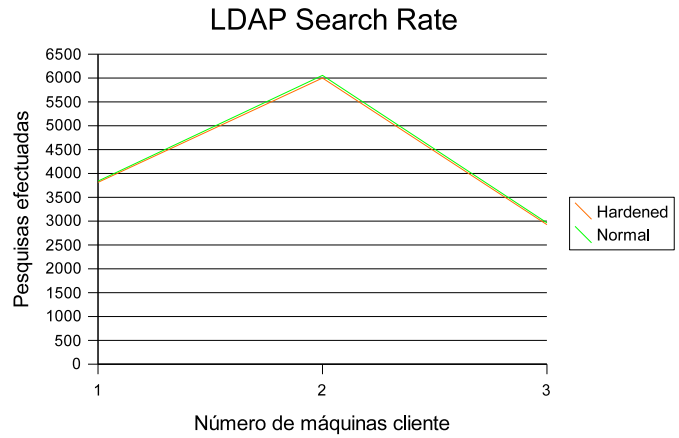

Figura 3: Pesquisas LDAP.

resentado o número de máquinas cliente. Neste caso, um único *thread* satura o processador (80%) da máquina cliente, pelo que não foi possível iniciar novos *threads*. Foi necessário usar mais nós computacionais para albergar novos clientes.

A quebra acontece ao terceiro cliente, não se notando diferenças significativas entre as duas configurações.

SMTP

Para o teste de SMTP, cada *thread* envia um correio electrónico para um endereço local ao servidor. A dimensão da mensagem é de 1024 bytes (Figura 4).

No eixo horizontal pode-se ver o número de *threads* cliente. De notar que, ao aumentar de 32 para 64 *threads* não houve um aumento superior a 5% no número de pedidos atendidos o que constitui a condição de paragem.

Até 8 *threads* a diferença entre as configurações é desprezável. A partir de 16 *threads* ambos os servidores começam a quebrar, notando-se uma maior capacidade de serviço ao *kernel* normal, como esperado.

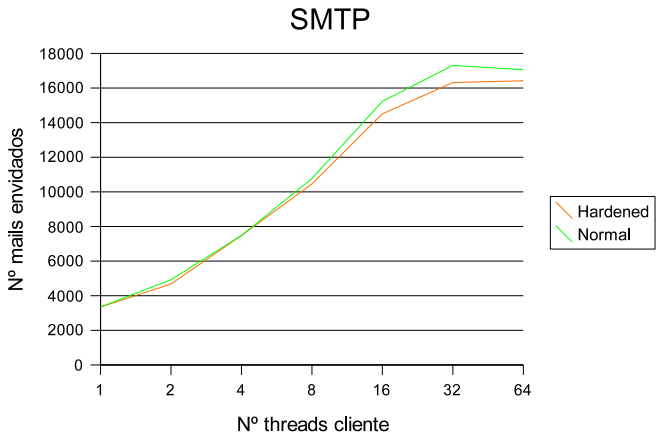

Figura 4: Número de mensagens enviadas por STMP.

POP

O teste de POP consitiu em usar diversos *threads* clientes a descarregar mensagens do servidor. De notar que se encontrava na caixa apenas uma mensagem. Após receber a mensagem, o *thread* iniciava nova ligação e recomeçava o processo (Figura 5).

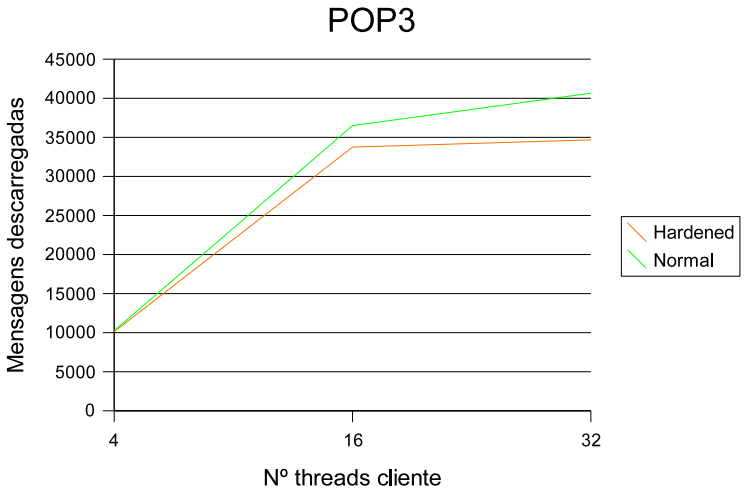

Figura 5: Número de mensagens descarregadas com POP.

Neste caso, a diferença entre o *kernel* normal e o *kernel* robustecido é superior aos casos anteriores, provavelmente devido à necessidade de realizar mais operações de alocação de memória que nos casos anteriores.

5 Conclusões

Os sistemas servidores encontram-se permanentemente expostos a ataques ao seu normal funcionamento, tendo em vista a obtenção (ilícita) de informação privilegiada, ou o comprometimento da eficácia dos serviços, colocando em risco o normal funcionamento das organizações. Assim, e dependendo do valor estratégicos da informação e bens envolvidos,

uma organização poderá ter a necessidade de associar recursos consideráveis à protecção dos seus sistemas de informação.

A variedade de ataques possíveis e a sua constante evolução tornam impraticável a aplicação de medidas paliativas específicas para cada ataque. Uma alternativa será o recurso a contra-medidas automáticas (em que o intervenção do factor humano é menos solicitada) e de carácter mais genérico. Neste contexto, técnicas baseadas em RBAC, como as disponibilizadas pela plataforma *grsecurity*, previnem muitos ataques. Estas técnicas operam ao nível do sistema operativo, em mecanismos como os de gestão de memória, controlo de acesso, gestão de permissões, etc.

Neste artigo procuramos investigar o impacto da aplicação destas técnicas no desempenho dos serviços Internet mais comuns - HTTP, LDAP, SMTP e POP. Para o efeito, efectuamos testes de carga em dois cenários distintos, correspondentes à utilização de um núcleo (*kernel*) normal e de outro modificado pela plataforma *grsecurity*.

Em traços gerais, observamos que o desempenho dos serviços HTTP, SMTP e LDAP (este no que diz respeito a operações de pesquisa) é pouco afectado, o que não acontece com o serviço POP. Mais especificamente: i) os serviços que utilizam memória partilhada intensivamente (HTTP, SMTP e LDAP) parecem sofrer menor quebra de desempenho; ii) a utilização do PaX parece ser o factor responsável pela degradação (ainda que diferenciada) do desempenho em todos os serviços, devido à sobrecarga introduzida ao nível da gestão de memória [6].

Neste sentido, pode-se concluir que, em máquinas de produção, é conveniente a opção por implementações de serviços que usem, sempre que possível, memória partilhada. Quando tal não for possível, é necessário dar atenção especial à configuração do PaX, para que não haja uma grande degradação do serviço.

Futuramente, pretendemos extender este estudo a outras arquitecturas e ambientes, tais como arquitecturas de 64 bits e máquinas virtuais, com o objectivo de adquirir conhecimento complementar que permita a administradores de sistemas e outros agentes decisores a escolha das soluções mais adequadas em termos do binómio custo/benefício.

Referências

[1] J.K. Guo, S. Johnson, D. Braun, and I.-P. Park, *Applicability of low water-mark mandatory access control security in linux-based advanced networked consumer electronics*, Consumer Communications and Networking Conference, 2004. CCNC 2004. First IEEE, 5-8 Jan. 2004, pp. 364–369.

[2] C.J. McCollum, J.R. Messing, and L. Notargiacomo, *Beyond the pale of mac and dac-defining new forms of access control*, Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on, 7-9 May 1990, pp. 190–200.

[3] Ravi S. Sandhu, *Lattice-based access control models*, IEEE Computer **26** (1993), no. 11, 9–19.

[4] Brad Spengler, *Grsecurity - acl documentation - v1.5*, Tech. report, grsecurity, 2003.

[5] ———, *Increasing performance and granularity in role-based acesss control systems*, (2004).

[6] Pedro Venda, *Pax performance impact (http://www.pjvenda.org/linux/doc/pax-performance/)*.