O slackware é a distribuição linux mais antiga ainda em atividade. Tendo sido criada por Patrick Volkerding em 1993, a partir da SLS.

Em todos esses anos, a distro conquistou ardorosos utilizadores, principalmente graças à sua filosofia de simplicidade e estabilidade.

Um produto de extrema qualidade para usuários com esta mesma característica. E este zine é de slacker para slacker.

slackware zine

17 de Novembro de 2006 – Edição #15

Editorial

Eeeeee! Não estamos atrasados! Pela primeira vez em muito tempo lançamos um zine dois meses depois do anterior. Conseguimos voltar à nossa periodicidade correta. Agradecimentos especiais aos nossos colaboradores!

O Ricardo Leite mandou dois artigos interessantes, um sobre como adicionar CBL no sendmail e outro para os corajosos que desejam testar um kernel novo em servidores a 500km de distância -;).

Duas matérias sobre assuntos “bem pedidos” completam a edição. Uma básica sobre como configurar um servidor de DNS primário, do Fabiano Silva e outra do Júlio mostrando a facilidade de se montar uma VPN com o OpenVPN (prestem atenção no arquivo de configuração muito bem comentado).

Além disso, gostaríamos de agradecer a todo o pessoal que passou no nosso estande durante o CONISLI (e pegou a edição #14.5). E, é claro, lembrar a todos que tem SlackShow dia 18 de Novembro!

Se você estiver em SP/SP ou proximidades, venha assistir palestras técnicas, feitas por técnicos e para técnicos. É quem faz mostrando o que sabe -;)

Até mais e Boa Leitura!

Piter PUNK

PS> Ah! Aproveitando o finalzinho da página, já que voltamos à periodicidade normal, estamos precisando dos seus artigos! -;)

Índice

slack users

Implementando lista negra CBL no sendmail sem alterar o arquivo sendmail.cf

Recentemente, com o aumento de adwares que infectam máquinas de usuários windows (que são geralmente alienados a questões como segurança em desktops). Ocorreu um aumento absurdo de máquinas usadas como "spam zombies", ou seja: máquinas que acabam sendo usadas para enviar centenas de e-mails por segundo sem que seus usuários saibam disso.

Uma das piores consequências do spam é que ele gasta o link do MTA, memória e processamento, podendo até se comportar como um verdadeiro DDoS (Distributed Deny Of Service) em alguns casos.

Geralmente quando se utiliza apenas filtros de conteudo como o spamassasin e similares, cada e-mail que chega tem que ser descarregado pelo servidor e analizado para somente depois bloquear a entrega ou não. Com isso os recursos do MTA serão onerados, pois mesmo quando se trata de um e-mail vindo de um IP comprovadamente "spammer" este será baixado pelo MTA.

Para resolver este problema, existem várias listas negras públicas de IPs que são atualizadas diariamente.

Uma das listas mais usadas é a CBL (Composite Bloking List, disponível na URL: http://cbl.abuseat.org/), esta lista pode ser obtida em forma de arquivo texto através de rsync. Mas para isso é necessário cadastrar o ip que irá fazer o download no site (http://cbl.abuseat.org/rsync-signup.html). Eles não cobram pelo serviço, mas pedem que o arquivo seja baixado no máximo 02 vezes por dia

Poderíamos fazer um script para bloquear todos os IP's da lista negra criando regras com o iptables, mas se acontecer um falso positivo (ou seja se o ip for bloqueado, mas o e-mail for um e-mail desejado, um e-mail HAM) o originador da mensagem não iria receber uma mensagem de erro informando o motivo do bloqueio. E além disso, o SysAdmin do MTA de origem pode nem saber que seu servidor está provavelmente sendo usado para enviar lixo, ele deveria ser avisado para acertar este problema! Um outro motivo para não se utlizar o iptables seria a quantidade gigantesca de regras que iriam onerar consideravelmente o desempenho do MTA. Só para se ter uma idéia o tamanho do arquivo (no momento em que eu escrevo este artigo) tem cerca de 42,2MB contendo 3.165.734 IPs cadastrados (caramba !).

Vamos criar um script para pegar a lista e coloca-la em um formato ágil para através de um simples e eficiente grep consultar sem perda de tempo:

```
#!/bin/bash
# Arquivo /usr/local/pegalista.sh

rsync -Cravzp rsync:// \
      rsync.cbl.abuseat.org/cbl/ \
      list.txt /var/cache
cat /var/cache/list.txt | grep -v "#"\
      | grep -v ":"| tr '.' 'x' > \
      /var/cache/cbl.txt
```

Este script pode ser executado uma vez por dia pelo crontab (basta digitar crontab -e para editar o crontab). Adicione no final o seguinte:

```
1 4 * * * /usr/local/pegalista.sh
```

Agora todo dia às 4:01 da manhã este script será executado. Note que eu substitui os pontos dos números de ip por caracteres 'x' porque o grep usa o ponto '.' como uma espécie de curinga e isto causaria erros na hora de comparar alguns IPs. (Pode crer em mim, me deu o maior problema isso !)

É possível implementar o CBL através do milter no sendmail, mas teriamos que recompilar o sendmail (o que seria muito chato na hora de atualizar usando os pacotes oficiais de **`www.slackware.org`**) e para dificultar um pouco mais, o sendmail possui um arquivo de configuração "frágil", o sendmail.cf. Ele é tão sensivel que trocar um simples <TAB> por um espaço pode detonar o funcionamento do sendmail. Claro que o pessoal do sendmail inclusive não recomenda que o sendmail.cf seja editado diretamente.

Eles inventaram uma forma (bastante inusitada) para se editar o arquivo de configuração do sendmail que é criar um arquivo de texto em outro formato (sendmail.m4) e depois converter no sendmail.cf com o utilitário m4. Como não queremos complicação (e sim uma implementação no melhor estilo **KISS**) vamos ver uma forma de intermediar uma conexão ao nosso servidor sendmail sem ter que "mexer no time que está ganhando".

Existe uma maneira de se intermediar uma conexão a um daemon de rede que aceita conexões TCP/UDP: usando o super server inetd que vem no **slackware**. O inetd pode ser usado para "levantar" um servidor pop, um samba, ou até um apache e um servidor de smtp.

Para executar um script no lugar do sendmail, poderiamos colocar no `/etc/inetd.conf` a seguinte linha:

```
smtp stream tcp nowait root \
      /usr/sbin/tcpd
/usr/local/sbin/sendmail_cbq.sh
```

Mas para este mesmo fim eu preferi utilizar no lugar do inetd tradicional o xinetd que é uma reimplementação do inetd com umas "features" que ajudam a combater situações de DoS pois ele pode ter limitação de conexões simultâneas por IP e limitação de quantidade de conexões por serviço por exemplo.

É importante lembrar que quando o xinetd não permitir a um determinado MTA conectar-se não chega a ser um grave problema porque os MTAs devem re-enviar os e-mails caso eles não consigam conectar ao MTA receptor segundo a RFC 821 (Simple Mail Transfer Protocol) e RFC 2505 (Anti-Spam Recommendations for SMTP MTAs). Vamos instalar o xinetd:

```
#> wget http://www.xinetd.org/     \
      xinetd- 2.3.14.tar.gz
#> tar xzvf xinetd-2.3.14.tar.gz
#> cd xinetd-2.3.14
#> ./configure
#> make
#> su
# make install
```

Neste ponto caso fosse utilizado anteriormente o inetd para outros serviços além do smtp, pode-se converter o arquivo de configuração dele (/etc/inetd.conf) no novo arquivo de configuração (/etc/xinetd.conf) com o utilitário xconv.pl. Se este for o caso tecle ainda como root:

```
#> cat /etc/inetd.conf | xconv.pl > \
      /etc/xined.conf
```

Abaixo transcrevo meu arquivo `/etc/xinetd.conf` devidamente editado e comentado:

```
defaults
{
# Número máximo de conexões por um
# serviço controlado:
# instances   = 25
# O tipo de registro de log.  Este
# log será feito a um arquivo sem
# utilizar o daemon syslogd.
# Outra opção seria: SYSLOG
syslog_facility [Nível_do_syslog]
log_type = FILE /var/log/servicelog

# Aqui vc pode determinar o que é
# logado quando ocorre a conexão.
# PID vai logar o pid do processo de
# requisição.
# HOST vai logar o enredeço remoto do
# host que conectou
# USERID vai logar o usuário remoto
# (segundo a RFC 1413)
# EXIT vai logar o status de término
# do runlevel do processo servidor.
# (0 sem errors)
# DURATION vai logar a duração da
# conexão.
log_on_success = HOST PID

# Aqui vc pode determinar o que é
# logado quando ocorre falha na
# conexão.
#(Utiliza-se os mesmo parâmetros da
# veriável anterior)
log_on_failure = HOST

# O número maximo de conexões a um
# específivo endereço de IP conectado
# a um específico serviço.
per_source  = 15
}

service smtp
{
  disable = no
  socket_type   = stream
  protocol      = tcp
  wait          = no
  user          = root

# Pode-se desativar a lista CBL mas
# permanecer com restrições anti DoS
# do xinetd ativando-se as proximas
# 2 linhas e comentando a linha que
# chama nosso script.

# server_args   = -C \
      /etc/mail/sendmail.cf -bs

# server = /usr/sbin/sendmail
```

```
# Este é o nosso script que será
# chamado como se ele fosse
# um servidor de SMTP:
server = /usr/local/sbin/  \
      sendmail_cbl.sh

# Em casos de DDoS causado por uma
# corrente de e-mails pode-se querer
# restringir um pouco mais as coisas:
# Pode-se restringir o numero de
# conexoes e instancias por ip
# instances = 300
# nice = 10

# Pode-se expecificar que este smtp
# somente pode ser conectado por
# determinado IP/rede only_from
# += 0.0.0.0 pode-se bloquear
# Ips  e redes:
# no_access += 129.22.122.84 \
      204.0.224.254
# Veja mais opções em \
      http://www.xinetd.org
}
```

Agora que já temos um xinetd configurado precisaremos de um script que será chamado pelo xinetd para intermediar a conexão com o sendmail:

```
#!/bin/bash
# Arquivo: \
      /usr/local/sbin/sendmail_cbl.sh

# A variável REMOTE_HOST retorna o
# IP do HOST que conectou no serviço
# chamado pelo xinetd   vamos
# converter o IP do host que conectou
# para o formato da nossa lista
# (colocando 'x' no lugart dos '.')
ip=$(echo $REMOTE_HOST | tr "." "x")

# Estou considerando que os usuários
# de nossa rede não devem ser
# bloqueados pelo nosso MTA

if ["X$(grep ^$ip$ /etc/iplocal.txt)"\
      = "X" ] ; then
      if [ "X$(grep ^$ip$ \
      /var/cache/cbl.txt)" != "X" ] ;\
      then
echo "550 TOO MUCH SPAM (BLOQUEADO \
      POR MANDAR SPAM)- \
      http://cbl.abuseat.org/ \
      lookup.cgi?ip=$REMOTE_HOST \
      &.submit=Lookup"
 exit
 fi
fi

# este é o pulo do gato que irá chamar
# o sendmail dentro do script caso
# esteja tudo OK
/usr/sbin/sendmail -C /etc/mail/  \
      sendmail.cf -bs
```

Não devemos esquecer de criar uma lista (em /etc/iplical.txt) com nossos IPs da rede local, afinal não queremos que nossos proprios clientes não consigam enviar e-mail. (Na verdade seria interesante ver se os ips de nossa rede estão na lista CBL e retira-los dela corrigindo o problema.) Esta lista de ips pode ser gerado com um comando semelhante a este:

```
#> for i in $(seq 1 255); do echo \
     192x168x0x$i >> /etc/iplocal.txt;
done
```

Este comando irá gerar uma lista com os ips de 192.168.0.1 até 192.168.0.255 (note que no lugar de '.' eu novamente coloquei 'x' para facilitar a procura sem erros do grep) Pode-se adicionar qualquer faixa de IPs reais com esse método. Finalmente convém alterar o antigo /etc/rc.sendmail para que ele chame o xinetd em vez do sendmail em modo daemon:

```
#!/bin/sh

sendmail_start() {
if [ -x /usr/sbin/sendmail ]; then
echo "Starting sendmail MTA daemon:
# /usr/sbin/sendmail -L sm-mta -bd \
            -q25m"
# /usr/sbin/sendmail -L sm-mta -bd \
       -q25m
# echo "Starting sendmail MSP queue \
      runner:  /usr/sbin/sendmail -L \
      sm-msp-queue -Ac -q25m"
# /usr/sbin/sendmail -L sm-msp-queue \
      -Ac -q25m
/usr/local/sbin/xinetd
echo "Ok"
  fi
}

sendmail_stop() {
  killall sendmail
  killall xinetd
}

sendmail_restart() {
  sendmail_stop
  sleep 1
  sendmail_start
}

case "$1" in
  'start') sendmail_start
  ;;
  'stop') sendmail_stop
  ;;
  'restart') sendmail_restart
  ;;
  *)
    echo "usage $0 start|stop|restart"
  ;;
esac
```

Um toque final seria colocar no crontab mais uma linha para re-enviar e-mals que tenham ficado pendentes na queue, já que o sendmail não está mais cuidando disso como um daemon (digite crontab -e):

```
*/5 * * * * /usr/sbin/sendmail -q
```

Pronto, agora esta implementação ira diminuir bastante o processamento de mensagens no MTA, pricipalmente se vc tiver o spamassasin instalado ou o amavis por exemplo.

Com este mesmo método pode-se também fazer outras verificações como por exemplo se o host que conectou o MTA possui DNS reverso e bloquea-lo caso não esteja em conformidade com esta especificação e de quebra enviar uma outra mensagem de erro indicando este problema. As mensagens de erro devem possuir no máximo 256 caracteres em uma única linha e devem ter no início o código 550 segundo a RFC 821.

Referências:

RFC's
http://www.faqs.org/rfcs/rfc821.html
http://www.faqs.org/rfcs/rfc2505.html

CBL - Composit Bloking List
http://cbl.abuseat.org/

Página anti-spam do Comitê Gestor da Internet no Brasil
http://antispam.br

Ferramentas de verificação de DNS e black lists (veja o Spam database lookup)
http://www.dnsstuff.com

Xinetd (um substituto para o inetd com ênfase na segurança)
http://www.xinetd.org

Ricardo Leite Gonçalves
<material@asbyte.com.br>

Autores

Fabiano Silva de Carvalho, é Analista de Sistemas pela Universidade Salgado de Oliveira e Mestre em Engenharia Elétrica e de Computação pela UFG. Atualmente é Professor Assistente da Universidade Salgado de Oliveira, Analista de Sistemas da Companhia Energética de Goiás e Professor Universitário da Faculdade de Tecnologia Senai de Desenvolvimento Gerencial. Tem experiência na área de Ciência da Computação, atuando principalmente nos seguintes temas: Sistemas Multiagentes, Aprendizado por Reforço, Data Warehouse, Implementação e Integração. Publicou 3 trabalhos em anais de eventos. Trabalha com Linux desde 1996.

Julio Cezar Estrella, é doutorando em Ciência da Computação e Matemática Computacional no ICMC-USP em São Carlos. É usuário slackware desde 1999.

Ricardo Leite Gonçalves a.k.a. Ricleite, é metido a administrador de redes, intrometido em programação asm de microcontroladores PIC e processadores Z80 e i386. Convencido em programação Basic dos principais computadores da old (80's) school, COBOL, Pascal, Xbase (Clipper), C, Bash, louco colecionador de hardware (VG e Computadores) dos anos 80, fução em eletrônica digital e doente em slackware.

Resultados da pesquisa br-linux.org

Distro
Servidor: Slackware (2o. Lugar)
Desktop: Slackware (2o. Lugar)

Grupo de usuários:
GUS-Br (3o. Lugar)

Personalidade Internacional:
Patrick Volkerding (5o. Lugar)

Personalidades Nacionais:
Piter Punk (5o. Lugar)
Sulamita Garcia (7o. Lugar)

Ótimos resultados!!! Mas vamos tentar melhorá-los para 2007!

Keep Slacking!

Implementando VPN's com o OpenVPN

Introdução:

VPN's, ou Virtual Private Networks, são túneis virtuais que podem ser estabelecidos sob uma rede pública, geralmente a internet, onde dois pontos são interligados com segurança, formando dessa forma, uma rede única.

As VPN's são amplamente utilizadas por empresas que necessitam comunicar matrizes com a filial. A idéia é que os dados trafegados pela Internet sejam transparentes para as entidades comunicantes, ainda que estas pertençam a redes com faixa de IP distintas.

Os dados são encriptados antes de entrar no túnel e apenas uma extremidade conhece a chave para decriptar o mesmo, criando um canal de comunicação relativamente seguro.

A utilização do OpenVPN tem se destacado nos últimos anos pela sua simplicidade e pela facilidade de manutenção. Além disso, possui uma outra característica relevante em relação a abordagens baseadas em IPSec que é o suporte a redes com NAT.

Este tutorial:

Este tutorial se destina aos iniciantes na instalação e configuração de OPENVPN. Os arquivos de configuração foram elaborados de acordo com o padrão descrito no manual oficial do OPENVPN. Detalhes adicionais, como suporte a TLS/SSL, serão abordados em um próximo tutorial. Vale ressaltar o tutorial é dividido em duas etapas. A primeira, aborda a instalação do OPENVPN em servidores distintos e a segunda permite que compartilhamentos SAMBA presentes em uma rede remota possam ser acessados através da VPN.

Para a elaboração desse artigo, foi utilizado o seguinte cenário e a seguinte configuração:

Sistema Operacional: **slackware** 11.0, mas o tutorial também funciona na versão 10.2.

Rede da Matriz: Faixa de IP - 192.68.0.0/27
ip da interface eth0 do servidor: 192.168.0.12
ip da interface tun0 do servidor: 10.0.0.1

Rede da Filial: Faixa de IP: 192.68.2.0/27
ip da interface eth0 do servidor: 192.168.2.1
ip da interface tun0 do servidor: 10.0.0.2

Instalação:

Baixando os arquivos necessários:

```
wget http://openvpn.net/release/\
      openvpn-2.0.9.tar.gz

wget http://www.oberhumer.com/\
      opensource/lzo/download/\
      lzo-2.02.tar.gz
```

Instalando a biblioteca LZO (necessária para compactação):

```
tar -xzvf lzo-2.02.tar.gz
cd lzo-2.02
./configure
make
make install
```

Instalando o OPENVPN:

```
tar -xzvf openvpn-1.5.0.tar.gz
cd openvpn-1.5.0
./configure
make
make install
```

Configurando o servidor da matriz:

Inicialmente, vamos criar um usuário e um grupo chamados openvpn:

```
groupadd openvpn
useradd -g openvpn -s /dev/null \
      openvpn >/dev/null 2>&1
passwd -l
```

Em seguida, vamos criar o diretório onde iremos armazenar os arquivos de configuração da VPN na matriz:

```
mkdir /etc/openvpn-matriz
```

O próximo passo é gerar a chave criptográfica estática:

```
openvpn --genkey -secret /etc/\
      openvpn-matriz/chave
```

Agora, vamos criar o script matriz.conf com o seguinte conteúdo:

```
pico /etc/openvpn/matriz.conf
```

```
# Usar como interface o driver TUN
dev tun
# 10.0.0.1 ip da matriz
# 10.0.0.2 ip remoto, da filial
ifconfig 10.0.0.1 10.0.0.2
# Entra no diretório onde se encontram
# os arquivos de configuração
cd /etc/openvpn-matriz

# Indica que esse túnel possui uma
# chave criptográfica
secret chave

# OpenVPN usa a porta 5000/UDP por
# padrão. Cada túnel do OpenVPN deve
# usar uma porta diferente.
port 5000

# Usuário e grupo que irão executar o
# daemon do OpenVPN
user openvpn
group openvpn

# Especifica uma MTU comum entre as
# partes - MATRIZ e FILIAL
tun-mtu 1500
tun-mtu-extra 32

# Usa a biblioteca lzo para
# compactação dos dados que irão
# trafegar no túnel
comp-lzo

# Envia um ping via UDP para a parte
# remota a cada 15 segundos para
# manter a conexão de pé em firewall
# statefull.
ping 15

# Nível de log
verb 3
```

Iniciar a conexão no servidor da matriz, executando o comando abaixo:

```
openvpn --config /etc/openvpn/\
      matriz.conf -daemon
```

Para verificar se a interface `tun0` foi inicializada com sucesso:

```
ifconfig tun0
```

Para configurando o servidor da filial, repita o processo de download e instalação e configuração exatamente da mesma forma como foi feito na matriz. Em seguida, copie a chave criptográfica da matriz para a filial com o seguinte comando:

```
scp /etc/openvpn-matriz/chave  \
      ip_filial:/etc/openvpn-filial
```

Por fim, crie o script `filial.conf` com o conteúdo **idêntico** ao do arquivo `matriz.conf`, alterando **somente** as seguintes linhas:

```
ifconfig 10.0.0.2 10.0.0.1

# NOTA: ESCOLHA UMA DAS POSSIBILIDADES
# ABAIXO. NUNCA AS DUAS!!!!
remote 201.43.56.5 (se for IP fixo)
remote domain.com.br (IP dinâmico)

# Entra no diretório onde se encontram
# os arquivos de configuração
cd /etc/openvpn-filial
```

Iniciar a conexão no servidor da filial, executando o comando abaixo:

```
openvpn --config /etc/openvpn-filial/\
      filial.conf -daemon
```

Para verificar se a interface `tun0` foi carregada com sucesso:

```
ifconfig tun0
```

Na filial, faça o teste com um ping:

```
ping 10.0.0.1
```

2ª Etapa:

Se a matriz possui um serviço samba rodando, alguns passos são necessários para que as máquinas da filial possam acessar os seus compartilhamentos. Vamos a eles:

Adicione as seguintes linhas ao arquivo `/etc/samba/smb.conf`:

```
# rede do servidor da matriz e rede do
# servidor da filial
hosts allow = 192.168.1. 192.168.2.
interfaces  = 192.168.0.0/27  \
      192.168.2.0/27
```

Reinicie o serviço samba, com os comandos:

```
/etc/rc.d/rc.samba stop
/etc/rc.d/rc.samba start
```

Faça alguns testes, acessando o compartilhamento através do Windows Explorer ou usando o comando abaixo no prompt de cliente da rede interna:

```
net use z: \\10.0.0.1\usuario_samba\
      /USER:usuario_do_samba ou ainda,

net use z: \\192.168.0.12\usuario\
      _samba /USER:usuario_do_samba
```

O processo de automatização é trivial, bastando inserir as seguintes entradas no arquivo `/etc/rc.d/rc.local`:

```
# Na matriz.
nohup /usr/local/sbin/openvpn –config  \
      /etc/openvpn-matriz/matriz.conf  \
      --daemon >/dev/null 2>&1
route add -net 192.168.2.0/27 gw 10.0.0.2

# Na filial.
nohup /usr/local/sbin/openvpn –config  \
      /etc/openvpn-filial/filial.conf  \
      --daemon >/dev/null 2>&1
route add -net 192.168.0.0/27 gw 10.0.0.1
```

Julio Cezar Estrella
<juliocest@hotmail.com>

Instalando um servidor DNS primário no slackware

Um servidor DNS é um serviço de fundamental importância que também pode ser implementado em sua rede local, vejamos o processo de implementação de um servidor DNS primário utilizando o bom e velho **slackware**.

Primeiramente, verifique se o pacote bind está instalado:

```
# ls /var/log/packages/bind*
```

Se o pacote estiver instalado existe um arquivo com o nome do pacote dentro do diretório /var/log/packages .

Caso o pacote não esteja instalado, baixe o pacote no site http://www.slackware.com ou na mídia de instalação do **slackware** e instale o pacotecom o comando:
`# installpkg bind-versao.tar.gz`

Modifique o arquivo /etc/named.conf adicionando as entradas zones (direta e reversa) para o domínio que se deseja configurar.

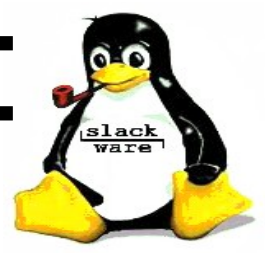
```
zone "seudominio.com.br" IN {
        type master;
        file "seudominio.com.br.zone";
        allow-update { none; };
};

zone "0.168.192.in-addr.arpa" IN {
        type master;
                                file
"seudominio.com.br.reverse";
        allow-update { none; };
};
```

Veja como ficou o arquivo após as alterações:

```
options {
      directory "/var/named";
      forwarders{192.168.0.1;};
    // query-source address * port 53;
};

zone "." IN {
      type hint;
      file "caching-example/named.ca";
};

zone "localhost" IN {
      type master;
      file "caching-  \
      example/localhost.zone";
      allow-update { none; };
};

zone "0.0.127.in-addr.arpa" IN {
      type master;
      file "caching-  \
      example/named.local";
      allow-update { none; };
};

zone "seudominio.com.br" IN {
      type master;
      file "seudominio.com.br.zone";
      allow-update { none; };
};

zone "0.168.192.in-addr.arpa" IN {
      type master;
      file    \
      "seudominio.com.br.reverse";
      allow-update { none; };
};
```

Verifique a sintaxe do arquivo /etc/named.conf com o utilitário:

```
# named-checkconf
```

Se o comando não retornar nada o arquivo está correto. Se houver algum erro a linha e o erro serão mostrados.

Crie o arquivo `.zone`, cujo formato é: `seudominio.com.br.zone`, com as informações do domínio que está sendo configurado. Pode-se utilizar como modelo o arquivo `/var/named/caching-example/localhost.zone` , dessa forma:

```
# cp /var/named/\
      caching-example/localhost.zone \
      /var/named/\
      seudominio.com.br.zone
```

Em seguida, modifique-o com o editor de texto de sua preferência. Veja como ficou o arquivo após as alterações:

```
$TTL    86400
$ORIGIN seudominio.com.br.
@ 1D IN SOA root.seudominio.com.br. (
     271006001 ; serial (d. adams)
     3H        ; refresh
     15M       ; retry
     1W        ; expiry
     1D )      ; minimum
     IN NS    ns1.seudominio.com.br.
     IN NS    ns2.seudominio.com.br.
     IN MX    mail.seudominio.com.br.

ns1  IN A     192.168.0.201
ns2  IN A     192.168.0.202
mail IN A     192.168.0.203
www  IN A     192.168.0.204
ftp  IN A     192.168.0.205
pop  IN CNAME mail.seudominio.com.br.
smtp IN CNAME mail.seudominio.com.br.
```

Verifique a sintaxe do arquivo `/var/named/seudominio.com.br.zone` com o utilitário `named-checkzone`

```
# named-checkzone seudominio.com.br \
      /var/named/\
      seudominio.com.br.zone
```

Se o comando retornar OK o arquivo está correto. Se houver algum erro a linha e o erro serão mostrados.

Crie o arquivo `.reverse`, cujo formato é: `seudominio.com.br.reverse`, com as informações de resolução reversa do domínio que está sendo configurado.

Pode-se utilizar como modelo o arquivo `/var/named/caching-example/named.local`

Para copiar o modelo:

```
# cp /var/named/caching-\
      example/named.local \
      /var/named/\
      seudominio.com.br.reverse
```

Em seguida, edite o arquivo utilizando o editor de texto de sua preferência:

```
# vi /var/named/seudominio.com.br.reverse

$TTL    86400
@ IN SOA seudominio.com.br.\
       root.seudominio.com.br.  (
       271006001 ; Serial
       28800     ; Refresh
       14400     ; Retry
       3600000   ; Expire
       86400 )   ; Minimum

       IN NS    ns1.seudominio.com.br.
       IN NS    ns1.seudominio.com.br.

201    IN PTR   ns1.seudominio.com.br.
202    IN PTR   ns2.seudominio.com.br.
203    IN PTR   mail.seudominio.com.br.
204    IN PTR   www.seudominio.com.br.
205    IN PTR   ftp.seudominio.com.br.
```

Verifique a sintaxe do arquivo recém-criado `/var/named/seudominio.com.br.reverse` com o utilitário `named-checkzone`:

```
# named-checkzone \
      0.168.192.in-addr.arpa \
      /var/named/\
      seudominio.com.br.reverse
```

Se o comando retornar OK o arquivo está correto. Se houver algum erro a linha e o erro serão mostrados. Inicie o bind:

```
# sh /etc/rc.d/rc.bind start
```

Se preferir tornar automática a inicialização bind a cada novo boot dê permissão de execução para o script:

```
# chmod +x /etc/rc.d/rc.bind
```

Para testar o funcionamento do servidor DNS utilize o utilitário `nslookup`.

Fabiano Silva de Carvalho
<fscarvalho@gmail.com>

Testando um kernel novo remotamente no slackware

É muito perigoso alterar um kernel em uma máquina remota. Já que isso envolve algumas operações que não poderemos ver e se estas derem errado podem travar o sistema remoto obrigando-nos a se mover ao local físico do servidor.

Mas podem acontecer situações onde é inevitável alterar imediatamente o kernel e impossível estar de "corpo presente". Ou simplesmente você pode não querer ir na sala de servidores e ficar lá espremido(a) só para testar um kernel novo. De qualquer forma alguns métodos e cuidados podem permitir que seja possível realizar remotamente até mesmo essa operação perigosa.

Portanto arregasse as mangas e seja você um "cabra macho" ou uma destemida Linux chix ou mesmo um doido varrido como eu e parta para o desafio de fazer uma "cirurgia por telepresença no cérebro do pinguim".

Testar um kernel novo envolve "rebootar" a máquina e torcer que nada trave ou não saia do controle (como um "kernel panic" ou coisa pior). Pois se der um "kernel panic" o sistema trava e fica congelado até alguém aparecer e reiniciar a máquina, certo ? Não exatamente! Existe um parâmetro que pode ser colocado no lilo.conf (ou mesmo passado no prompt do lilo) que diz ao kernel do Linux que caso ele "trave" a maquina deve ser rebootada em 'n' segundos. Isso pode parecer inútil pois o Linux iria rebootar e carregar de novo o kernel que travou e assim ficar indefinidamente obrigando-nos a ir ao "local do crime". Bem, existe um outro parâmetro do lilo que diz a ele que SOMENTE no próximo boot ele deve carregar determinado kernel e/ou determinadas opões de kernel:

```
#> lilo -R kernel_teste \
      append="panic=3"
```

O comando acima diz ao lilo para que na próxima vez (e somente na próxima vez) que houver um reboot seja carregado o kernel identificado como kernel_teste e além disso caso aconteça um "kernel panic" a maquina não ficará congelada mas será rebootada depois de 3 segundos. Se a máquina rebootar por causa do "kernel panic" ela ira carregar o kernel padrão do sistema e não mais o kernel_teste novamente. Em outras palavras, se nosso kernel de teste der problema o sistema deve rebootar com o kernel padrão.

Transcrevo a seguir um exemplo de lilo.conf usado para testar um kernel novo:

```
# Arquivo: /etc/lilo.conf
boot=/dev/hda
prompt
timeout=50
lba32
vga=normal
root=/dev/hda1
read-only
menu-title=" Slackware "
#
# Aqui coloca-se o label do kernel
# padrão que funciona comprovadamente.
default=Linux

# Aqui ficam as configurações do
# kernel padrão de instalação
image=/boot/vmlinuz
label=Linux
root=/dev/hda1
read-only

# Aqui está definido o novo kernel de
# teste
image=/usr/src/linux-
2.6.16/arch/i386/boot/bzImage
label=kernel_teste
root=/dev/hda1

# Esta opção permite ao linux rebootar
# em 3 segundos caso ocorra um "kernel
# panic"
append="panic=3"
read-only
```

Infelizmente algumas coisas inesperadas ainda podem dar errado sem que isso provoque um "kernel panic". Por exemplo: as placas de rede podem "mudar de ordem", a placa que era para ser eth0 se torna eth1 e vice versa. Este problema aconteceu comigo ao atualizar um kernel de 2.4.x para o 2.6.x.

Para resolver estes e outros imprevistos o que nós devemos ter em mente é que deve ser possível voltar a se conectar ao servidor depois do boot com o novo kernel.Uma possível solução é um script de verificação pós boot colocado no final de `/etc/rc.d/rc.local` que irá verificar se tudo o que precisa funcionar realmente está funcionando.

São basicamente 3 coisas que TEM de funcionar sem erros:

1) O Driver de leitura do HD e do filesystem usado tem que funcionar e estar compilado built-in e não como módulo (Apesar de ser possível compilar como módulo e utilizar a `initrd`, built-in é mais simples e quanto mais simples melhor).

2) O driver da placa de rede tem que funcionar e estar carregado (Fique esperto no caso raríssimo de drivers mudarem ligeiramente de nome entre versões de séries de kernel, ou serem unificados vários drivers de placas semelhantes em um único módulo). Teoricamente o santo `/etc/rc.d/rc.hotplug` descobre os drivers corretos. Todo cuidado é pouco, e colocar o driver de placa de rede built-in pode ser uma medida aceitavel em uma situação dessas.

(Aproveite pra compilar como módulo também alguns outras marcas mais populares de placas de rede. Melhor prevenir, do que ter que no futuro recompilar o kernel todo porque compilamos drivers somente pra uma placa de rede que não se acha mais no mercado.)

3) O processo `sshd` precisa estar ativo a permitindo conexões remotas. Cuidado com as regras de firewall que tem como alvo o ip e a porta do ssh, pode acontecer que depois do reboot e justo nessa hora a sua conexão ADSL cai e ao autenticar de novo você descubra que possui um novo ip e fique sem poder conectar por conta de uma regra de iptables mais nervosa.

Uma abordagem que deve funcionar na maioria dos casos é também colocar no nosso "script de checagem pós boot" uma contagem regressiva de tempo que reboota o servidor de qualquer maneira caso não consigamos conectar nele.

Convém desativar essa contagem regressiva assim que consigamos conectar ou caso o "timeout" ocorra. Se o servidor rebootar por esse "timeout" irá voltar ao kernel antigo (se tiver sido aplicada a dica anterior do `lilo`).

O motivo de se programar o "timeout" para ele se desative sozinho caso dispare é que isso evita que nosso servidor fique rebootando eternamente (já pensou se nossa rede local cair ou se ficarmos impossibilitados de conectar de novo no servidor pra terminar o serviço ?).

Vejamos um script ideal:

```
#!/bin/bash
# Arquivo: /root/timeout.sh
# tem gente que prefere mudar a
# porta do ssh, então fica mais fácil
# mudar aqui:

sshport=22
log=/root/timeout.log
reboot=0

# se o script já tenha sido
# executado sai

if [ -f /root/delete.me ]; then
    exit
else
   touch /root/delete.me

   # obtém o gateway para testar
   # a rede:
   GATEWAY=$(route -n | grep  \
      ^0\.0\.0\.0  | cut -c17-  \
      | cut -d" " -f1)
  # outra forma de pegar o gateway é
  # descomentando a linha a seguir:
  #. /etc/rc.d/rc.inetd1.conf

  # É importante que o gateway
  # responda ao ping. Conforme o caso
  # abra na regra de firewall do
  # gateway para responder aos pings.
  internet=$(ping $GATEWAY -c 5 -W 1 \
      | grep "100% packet loss" )

  if [ "X$internet" != "X" ] ; then \
      echo "GATEWAY $GATEWAY não \
      responde ao ping" >> $log
      reboot=1
  fi

  # testa se o sshd está conectando
  # no  localhost
  if ["X$(nmap localhost -p $sshport \
      | grep open) = "X"]; then
      echo "sshd está morto" >> $log
      reboot=1
  fi

  if [ "$reboot" -eq "1" ]; then
      /sbin/reboot
  fi

  # Agenda o shutdown para daqui a 5
  # min e não checa o filesystem no
  # próximo boot
```

```
  shutdown -r -f 5
  while [ 1 ]; do
    # checa se alguém se conectou
    # via ssh
    if ["X$(who|grep pts/ )" != "X"] \
      ; then
      shutdown -c  # cancela shutdown
      # determina qual o terminal ssh
      # deve enviar os avisos
      n=$(who | grep "pts/" | \
          cut -f2 -d"/" | \
          cut -f1 -d" ")
      # Envia mensagem ao terminal
      # quando se conecta via ssh
      echo "Benvindo, \
            Time-bomb desativada" \
            >/dev/pts/$n
      echo "CUIDADO" >/dev/pts/$n
      echo "Se for fazer novo teste \
            de kernel," >/dev/pts/$n
      echo "não esqueça de apagar: \
            /root/delete.me" \
            >/dev/pts/$n
      echo "CUIDADO" >/dev/pts/$n
      exit
    fi
  done
fi
```

Este script deve ser chamado no final de `/etc/rc.d/rc.local` e é bom que ele tenha um & no final (para que ele seja executado em background). Vejamos como ficaria dentro de um `/etc/rc.d/rc.local` padrão:

```
#!/bin/sh
#
# /etc/rc.d/rc.local:  Local system
# initialization script.
# Put any local setup commands in
# here:

# Para não esquecer, coloquei aqui o
# chmod para tornar executável o
# script de timeout
chmod +x /root/timeout.sh
/root/timeout.sh &
```

Você também pode dar o direito de execução ao script via linha de comando, sem problemas, ok?

Você pode compilar localmente o kernel e somente fazer upload dele ao servidor para testar. Essa prática é perfeitamente possível e recomendada mesmo se o processador e hardware da máquina alvo seja diferente da máquina local.

Somente um cuidado é de suma importância: se você estiver compilando a mesma versão do kernel que já está rodando no servidor é perigoso sobre-escrever o kernel padrão com o novo por ele ter o mesmo nome.

Para resolver isso é essencial que seja editado o `Makefile` do diretório `/usr/src/linux-x.x.x` e adicionado um valor à variável chamada `EXTRAVERSION`.

Dessa forma o novo kernel de teste não irá sobrescrever o anterior.

Logo no inicio do arquivo `Makefile` encontramos a variável `EXTRAVERSION`, é só colocar depois do '=' o valor que você quiser, procure não colocar espaços nem caracteres acentuados.

Agora para se produzir um pacote .tgz do kernel é muito simples.

Se for kernel série 2.4.x após compilar (e em vez de digitar "`make install`" e "`make modules_install`"), digite::

```
#> make rpm
#> rpm2tgz \
   /usr/src/rpm/RPMS/i386/\
      kernel-*.rpm
```

Ou se for um kernel da série 2.6.x, após compilar, digite:

```
#> make targz-pkg
#> mv linux-2.6.x.tar.gz \
      linux-2.6.x.tgz
```

Mesmo tomando esses pequenos cuidados ainda pode acontecer algum desastre durante o boot.

O kernel testado poderia por exemplo danificar o filesystem por algum motivo que só Murphy conseguiria entender, então a dica final é que você tenha um pouco de fé e se possível alguém lá no local pra via telefone seguir suas instruções se tudo der errado.

Essas dicas funcionam muito bem, mas nada substitui a experiência pessoal e o bom senso.

Ah sim, e antes que eu me esqueça:

"Apesar dessas dicas e do script terem sido testadas e funcionarem no meu caso, não me responsabilizo por qualquer dano ocorrido durante a operação. Não irei indenizar ninguém por qualquer erro contido neste texto, nem por erros contidos nos scripts ou procedimentos. Qualquer um que for corajoso (ou louco) de tentar estes procedimentos deve ser também corajoso (ou louco) para se responsabilizar pelas conseqüências."

Be a man! (or a brave chix) !

Ricardo Leite Gonçalves
<material@asbyte.com.br>