



HARDENING EM LINUX

CADERNO DE ATIVIDADES

Copyright © 2018 – Rede Nacional de Ensino e Pesquisa – RNP

Rua Lauro Müller, 116 sala 1103

22290-906 Rio de Janeiro, RJ

Diretor Geral

Nelson Simões

Diretor de Serviços e Soluções

José Luiz Ribeiro Filho

Escola Superior de Redes

Diretor Adjunto

Leandro Marcos de Oliveira Guimarães

Equipe ESR (em ordem alfabética)

Adriana Pierro, Celia Maciel, Camila Gomes, Edson Kowask, Elimária Barbosa, Evellyn Feitosa, Felipe Arrais, Felipe Nascimento, Lourdes Soncin, Luciana Batista, Márcia Correa, Márcia Rodrigues, Monique Souza, Renato Duarte, Thays Farias, Thyago Alves e Yve Marcial.

Versão 0.1.0

Índice

Sessão 1: Instalação e configurações iniciais	1
1) Criação de máquina virtual no Virtualbox	2
2) Instalação do Debian Linux	6
3) Ajustes pós-instalação	16
4) Configuração do LVM	26
5) Inserção de senha no bootloader	35
6) Clonando máquinas virtuais	39
7) Operações avançadas com LVM	40
8) Criptografia de partições	47
Sessão 2: Firewall	52
1) Topologia desta sessão	52
2) Criação da VM de firewall e DNS primário	55
3) Configuração inicial do firewall	56
4) Configuração do servidor DNS primário	62
5) Configuração do DNSSEC	73
6) Automatizando assinatura DNSSEC após alterações	78
7) Reconfiguração da VM debian-template	79
8) Criação da VM de DNS secundário	82
9) Configuração do DNS secundário	83
Sessão 3: Autenticação centralizada	92
1) Criação da VM para o servidor LDAP	92
2) Configuração do servidor LDAP	94
3) Habilitando logs do LDAP	97
4) Edição de índices e permissões no LDAP	99
5) Adição de grupos e usuários no LDAP	100
6) Integração e teste do sistema de autenticação com LDAP	103
7) Configurando uma autoridade certificadora (CA) para o SSH	105
8) Configurando a SSH-CA no servidor LDAP	109
9) Automatizando a assinatura de chaves SSH de usuários	113
10) Configurando o template para funcionar com LDAP/SSH-CA	117
11) Configurando um cliente Linux	119
12) Configurando o firewall para funcionar com LDAP/SSH-CA	122
13) Restringindo login por grupos e usuários	123
14) Restringindo logins SSH apenas via chaves assimétricas	127
15) Bloqueando tentativas de brute force contra o SSH	129
Sessão 4: Controles de segurança	134
1) Requisitos de senha na base LDAP	134
2) Busca de senhas fracas	141

3) Servidor de arquivos NFS e quotas de disco	148
4) Uso de ACLs localmente	158
5) Uso de ACLs via NFS	161
6) Controle granular de permissões via sudo	165
Sessão 5: Registro e correlacionamento de eventos	178
1) Criação da máquina	178
2) Configuração do NTP	178
3) Registro de comandos digitados com SnoopyLog	181
4) Correlacionamento de eventos com o Graylog	182
Sessão 6: Gestão de configuração	186
1) Instalação do Ansible	186
2) Execução de comandos simples	186
3) Uso de roles no Ansible	187
4) Versionamento de configuração com git	190
Sessão 7: Hardening de sistemas web	191
Sessão 8: Isolamento de processos e containerização	192
Sessão 9: Módulos de segurança do kernel	193
Sessão 10: Monitoramento de vulnerabilidades	194

Sessão 1: Instalação e configurações iniciais

A segurança e o *hardening* de um sistema Linux começa desde o primeiro momento: sua instalação. Mesmo antes de iniciarmos a preparação de uma máquina ou servidor, as considerações sobre segurança devem povoar a mente do administrador de sistemas, visando reduzir a superfície de ataque, facilitar procedimentos de auditoria e garantir que as melhores práticas de configuração serão aplicadas de forma fácil e homogênea em todo o parque computacional.

Com o advento da virtualização, prevalente na maioria das organizações já há mais de dez anos, toma força o conceito de *one service per server*, ou um serviço por servidor. Nesse caso, o objetivo é que tenhamos vários servidores simples, muitas vezes com um único serviço operacional—isso facilita enormemente a administração e diminui a superfície de ataque de cada servidor, pois haverão poucos programas, bibliotecas e portas abertas a serem atacadas em cada máquina individual. O uso de *templates* é especialmente vantajoso para garantir que essa premissa seja aplicada com sucesso; construindo imagens-base sólidas e regularmente atualizadas e homologadas pela equipe de segurança da organização, é muito mais fácil e conveniente garantir que as VMs derivadas desses *templates* serão seguras.

Por outro lado, com a virtualização tivemos também o surgimento do *virtual machinel sprawl*—um número crescente (e muitas vezes aparentemente incontrolável) de máquinas virtuais sendo criadas no *datacenter*, minando as vantagens da simplicidade e facilidade de configuração apresentadas anteriormente. Processo e controles são fundamentais para garantir que VMs sejam criadas apenas quando necessário, e que possuam um ciclo de vida que considere sua implantação, operação e descontinuação quando não mais relevantes.

O primeiro passo para garantir que as várias máquinas em nosso ambiente estarão seguras é ser criterioso, portanto, com a criação dos *templates* de máquina virtual. Nesta sessão iremos tratar dos aspectos de segurança relevantes na instalação de um sistema Debian Linux a ser usado como *template* para derivação de VMs futuras a serem usadas neste curso, trabalhando aspectos relevantes da instalação de pacotes, gestão de discos e partições e criptografia de dados sensíveis.

1) Criação de máquina virtual no Virtualbox

1. Abra o *Oracle VM Virtualbox*. Para criar uma nova máquina virtual, clique em *New*. Na tela seguinte, você deverá escolher um nome, tipo e versão do sistema operacional a ser instalado na VM. Em *Name*, digite **debian-template**, em *Type* escolha **Linux** e em *Version* selecione **Debian (64-bit)**.

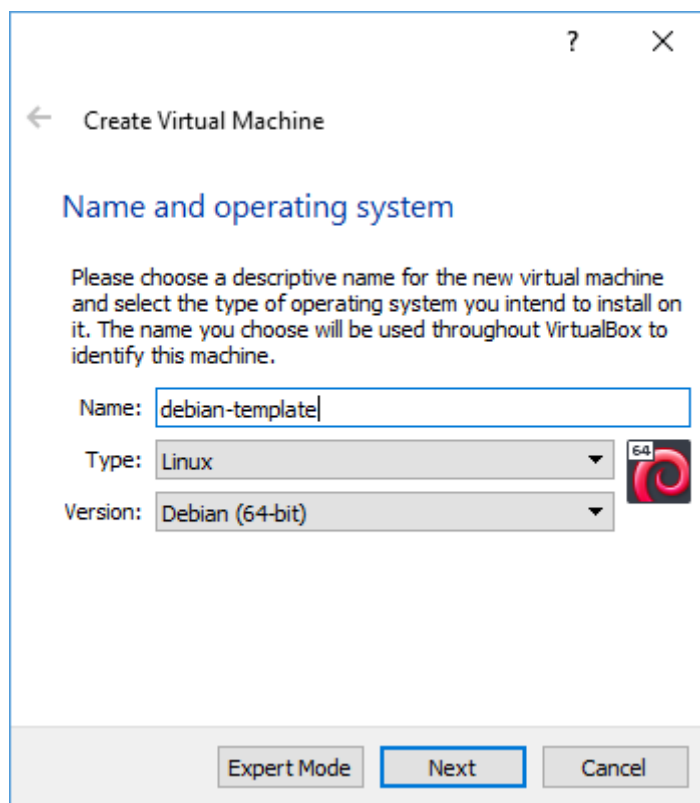


Figura 1. Criação de nova VM, parte 1

Em seguida, clique em *Next*.

2. Na tela seguinte escolheremos a quantidade de memória RAM a ser usada pelo sistema. O Debian Linux é um sistema bastante frugal, com recomendações mínimas de memória da ordem de 512 MB. Como a máquina que estamos instalando será um *template*, é interessante que ela seja bastante enxuta, e que as VMs derivadas cresçam em capacidade de acordo com o *workload* específico de cada aplicação.

Aponte **768 MB** de RAM, e em seguida clique em *Next*.

3. Agora, iremos definir se iremos criar um novo disco rígido virtual para a VM (ou usar um preexistente), e definir seu tamanho. Nesta primeira tela, mantenha a seleção-padrão *Create a virtual hard disk now*. Clique em *Create*.

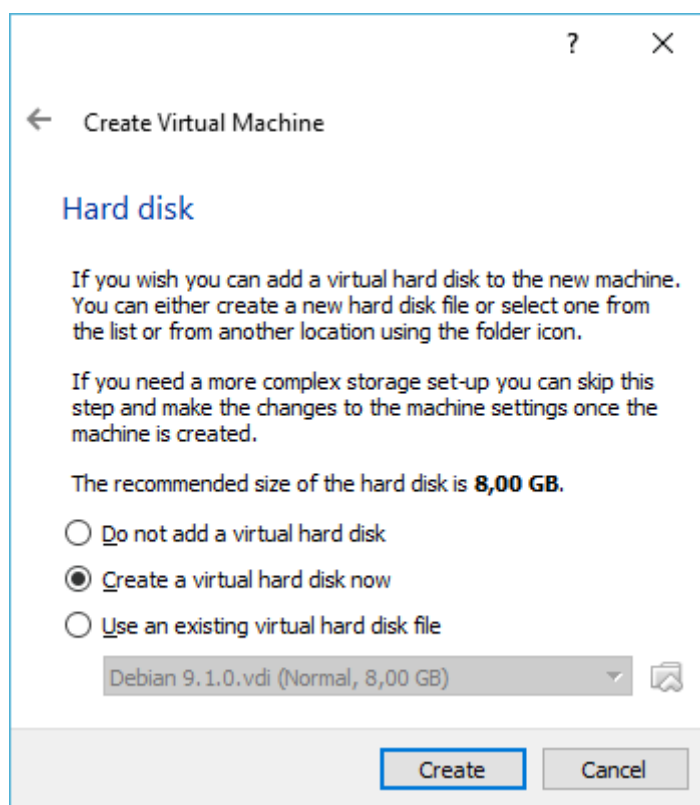


Figura 2. Criação de nova VM, parte 2

Na tela seguinte, de escolha do formato do disco virtual, mantenha a opção-padrão *VDI* (*VirtualBox Disk Image*). Em casos específicos em que se deseje interoperabilidade da VM com outros ambientes de virtualização, como VMWare ou Hyper-V, pode ser interessante escolher o formato *VMDK*. Clique em *Next*.

Agora, iremos selecionar se o espaço disco irá crescer à medida que for usado (*Dynamically allocated*), ou se será completamente alocado quando da sua criação (*Fixed size*). Em ambientes de produção, é geralmente recomendável selecionar a segunda opção, evitando que os dados do disco virtual fiquem fragmentados em pontos diferentes do disco físico, o que pode acarretar lentidão na leitura de dados, especialmente ao usar discos mecânicos. Neste exemplo, mantenha selecionada a opção *Dynamically allocated* e clique em *Next*.

Selecionaremos agora a localização do arquivo de disco virtual e seu tamanho. Não é necessário alterar a primeira opção — já para a segunda, é importante considerar que um *template* de máquina virtual será usado para criar vários tipos diferentes de servidores-alvo. Por esse motivo, é interessante que seu disco seja organizado de forma simples e seja facilmente extensível futuramente: a flexibilidade de adição de novos discos virtuais é especialmente vantajosa, pois permite que criemos uma instalação básica bastante enxuta, e a aumentemos conforme necessário.

Mantenha o valor-padrão de 8 GB para o tamanho do disco virtual, e clique em *Create*.



Figura 3. Criação de nova VM, parte 3

4. Será criada uma nova máquina virtual com o nome `debian-template`. Vamos fazer uma rápida pós-configuração antes de iniciar o processo de instalação: clique com o botão direito sobre a VM, e em seguida em *Settings*.

Selecione *Storage > Controller: IDE > Empty*, e na parte à direita da janela clique no pequeno ícone de um CD em frente à opção *Optical Drive*. Em seguida, clique em *Choose Virtual Optical Disk File...* e navegue pelo sistema de arquivos, selecionando a imagem ISO de instalação do Debian Linux como mostrado abaixo.

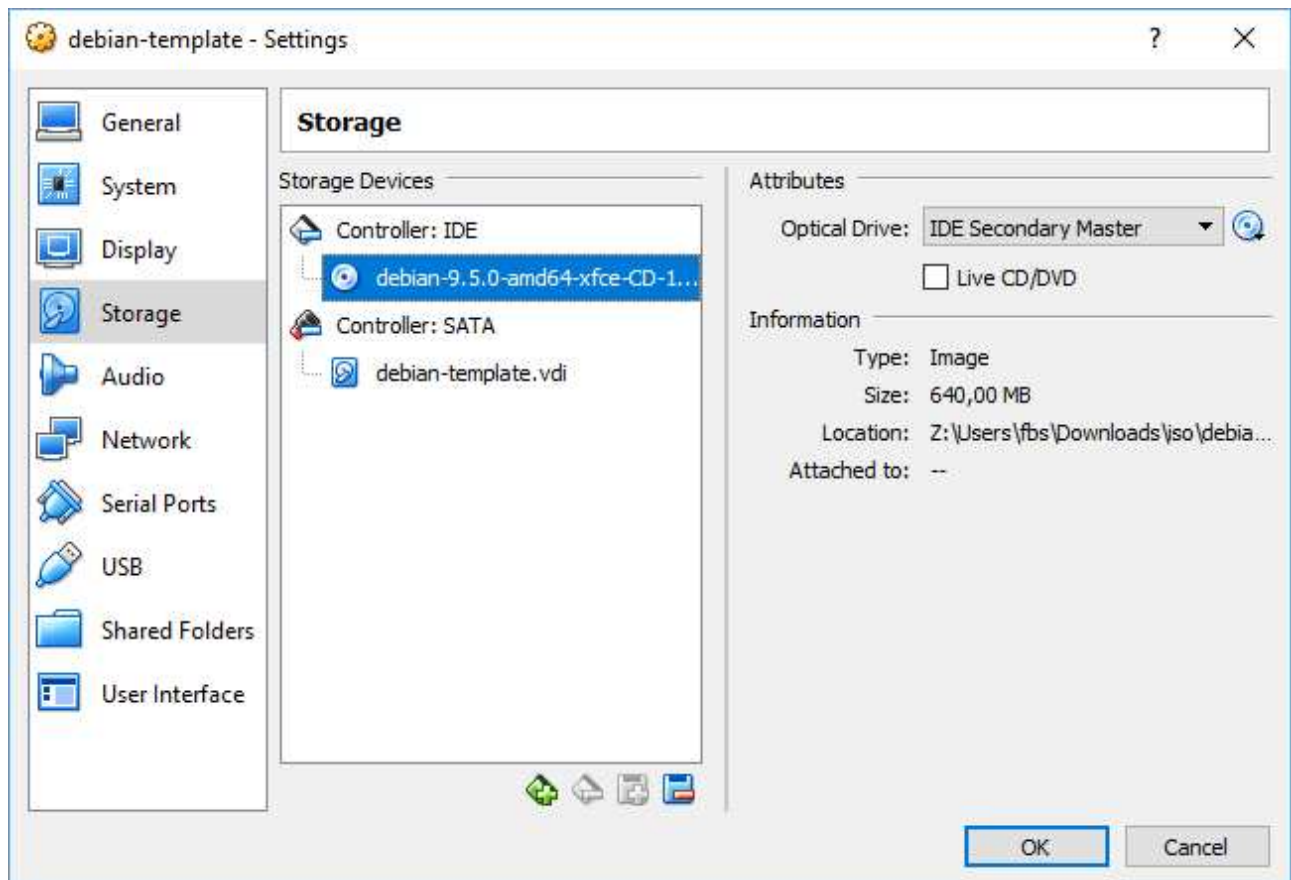


Figura 4. Configuração de nova VM

Em *Audio*, desmarque a caixa *Enable Audio*. Como iremos criar um conjunto de máquinas virtuais que atuarão exclusivamente como servidores, não há necessidade de dispor de áudio nas VMs.

Em *Network > Adapter 1 > Attached to*, altere a conexão de rede da máquina virtual para *Bridged Adapter*. Em *Name*, verifique que a placa de rede física conectada à rede externa está selecionada (isso é especialmente importante em máquinas que possuem múltiplas placas de rede ou interfaces *wireless*). Se desejar, expanda *Advanced* e clique no pequeno círculo azul à direita de *MAC Address* para randomizar um novo endereço físico para a placa de rede da máquina virtual, especialmente útil em casos de conflito de IP.

Em *USB*, marque a caixa *USB 1.1 (OHCI Controller)*. Esta configuração evita que sejam levantados erros ao iniciar a VM caso as extensões do Virtualbox não estejam instaladas na máquina hospedeira.

Finalmente, clique em *OK*.

2) Instalação do Debian Linux

1. Selecione a máquina virtual **debian-template** e clique no botão *Start* para iniciá-la. Após um curto período, você verá o menu de *boot* do Debian Linux; selecione a opção *Install* para começar o instalador no modo texto.

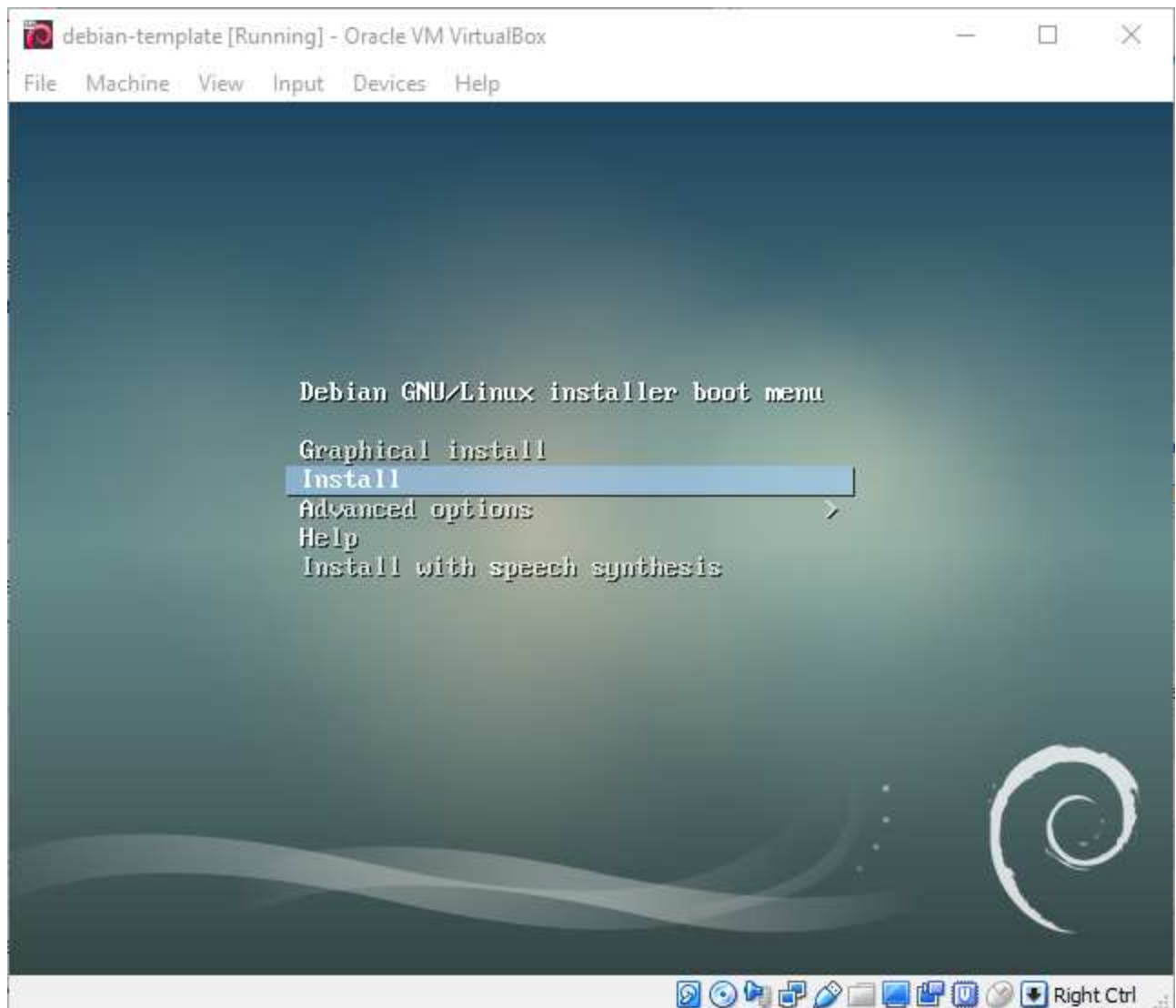


Figura 5. Instalação do Debian Linux, parte 1

2. No passo de seleção de idioma, selecione *Portuguese (Brazil)*. Em localidade, selecione *Brasil*. Para o mapa de teclado a ser usado, provavelmente será o *Português Brasileiro* (verifique se há a tecla **ç** ao lado do caractere **l**).

Os componentes do instalador serão carregados a seguir.

3. Em seguida, o instalador irá tentar autoconfigurar a rede usando DHCP. Caso esse protocolo não esteja disponível em sua rede local, consulte o instrutor sobre como proceder com a configuração manual das interfaces de rede.

Configurada a rede, iremos escolher o *hostname* da máquina. Defina o mesmo nome usado para a máquina virtual, **debian-template**.

Para o nome de domínio da rede, iremos usar a rede local fictícia **intnet** durante o curso.

4. Agora, iremos definir a senha do **root**, o superusuário em sistemas Linux. É bastante recomendado que se defina uma senha especialmente segura, já que esse usuário possui permissões totais sobre o sistema. Por simplicidade e homogeneidade no ambiente de curso, defina a senha como **rnpesr**.



Figura 6. Instalação do Debian Linux, parte 2

Na tela seguinte, confirme a senha.

5. O próximo passo é criar um usuário não-privilegiado para tarefas corriqueiras do sistema. Para o nome completo do usuário, digite **aluno** — este também será o nome de conta do usuário.

Para a senha, defina de igual forma **rnpesr**.

6. O instalador irá tentar obter a hora via Internet através do protocolo NTP. Em seguida, teremos que escolher um estado para definir o fuso horário do sistema. Escolha o estado em que você está realizando este curso.
7. Agora, faremos o particionamento do disco. Temos quatro opções: particionamento assistido usando o disco inteiro, assistido usando o disco inteiro com LVM, assistido com o disco inteiro e LVM criptografado e particionamento manual. Se tivéssemos mais de um disco virtual conectado à máquina, o instalador ofereceria também a opção de configuração assistida de RAID (*Redundant Array of Independent Disks*).

Mas, o que é LVM?

O *Logical Volume Manager* (LVM) é um sistema de mapeamento de dispositivos do Linux que permite a criação e gestão de volumes lógicos de armazenamento. As utilidades da gestão de armazenamento via volumes lógicos são muitas, destacando-se:

- Criação de volumes lógicos únicos englobando diferentes volumes físicos ou discos físicos inteiros, permitindo redimensionamento dinâmico de volumes.
- Gestão facilitada de grandes quantidades de discos físicos, permitindo que discos sejam adicionados ou substituídos sem *downtime* ou impacto à disponibilidade — especialmente útil quando combinado com hardware que suporta *hot swapping*.
- Em pequenos sistemas (como *desktops* e estações de trabalho) permite que o administrador não tenha que estimar no passo de instalação quão grande uma partição irá se tornar, permitindo redimensionamento dinâmico futuro.
- Criação de backups consistentes através de *snapshots* de volumes lógicos.
- Criptografar múltiplas partições físicas com uma mesma senha.

Em essência, o LVM traz enorme flexibilidade ao administrador de sistemas, resolvendo muitos dos problemas de particionamento que tínhamos no passado. Ele possui alguns conceitos centrais, ilustrados pela imagem a seguir:

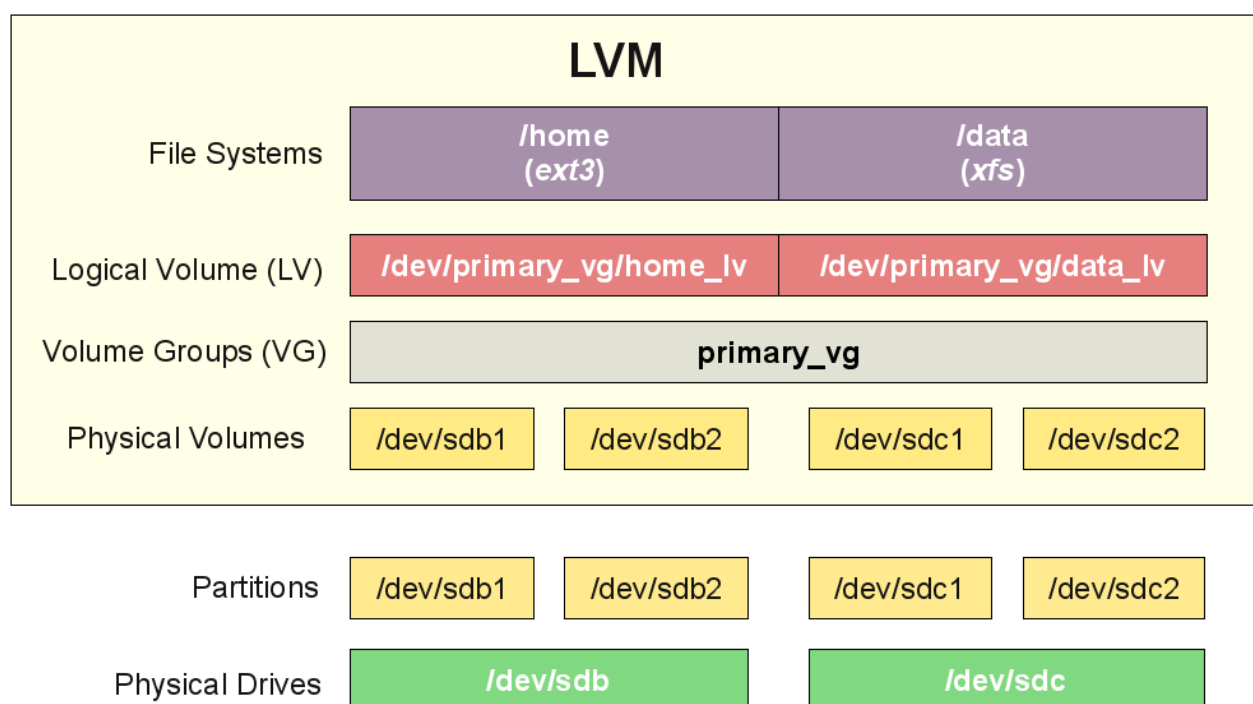


Figura 7. Organização do LVM

Na base do sistema temos os discos físicos conectados à máquina — como `/dev/sdb` ou `/dev/sdc`, por exemplo — que podem ser particionados (em formato MBR ou GPT) em múltiplas partições. Essas partições são denominadas volumes físicos (*Physical Volumes*, ou PVs). Vários PVs podem ser aglutinados para definir um grupo de volumes (*Volume Groups*, ou VGs), que é um agrupamento lógico desses PVs sob um mesmo nome. Pode-se então criar vários volumes lógicos (*Logical Volumes*, ou LVs) dentro desse VG, e finalmente formatar e montar diretórios dentro dos LVs, já no contexto do sistema de arquivos.

A explicação acima é propositalmente sucinta; iremos entrar em maior detalhe com relação ao funcionamento e operação do LVM em atividades subsequentes.

De volta ao instalador, iremos configurar um particionamento manual usando LVM. Por isso, na tela *Método de particionamento*, selecione *Manual*.

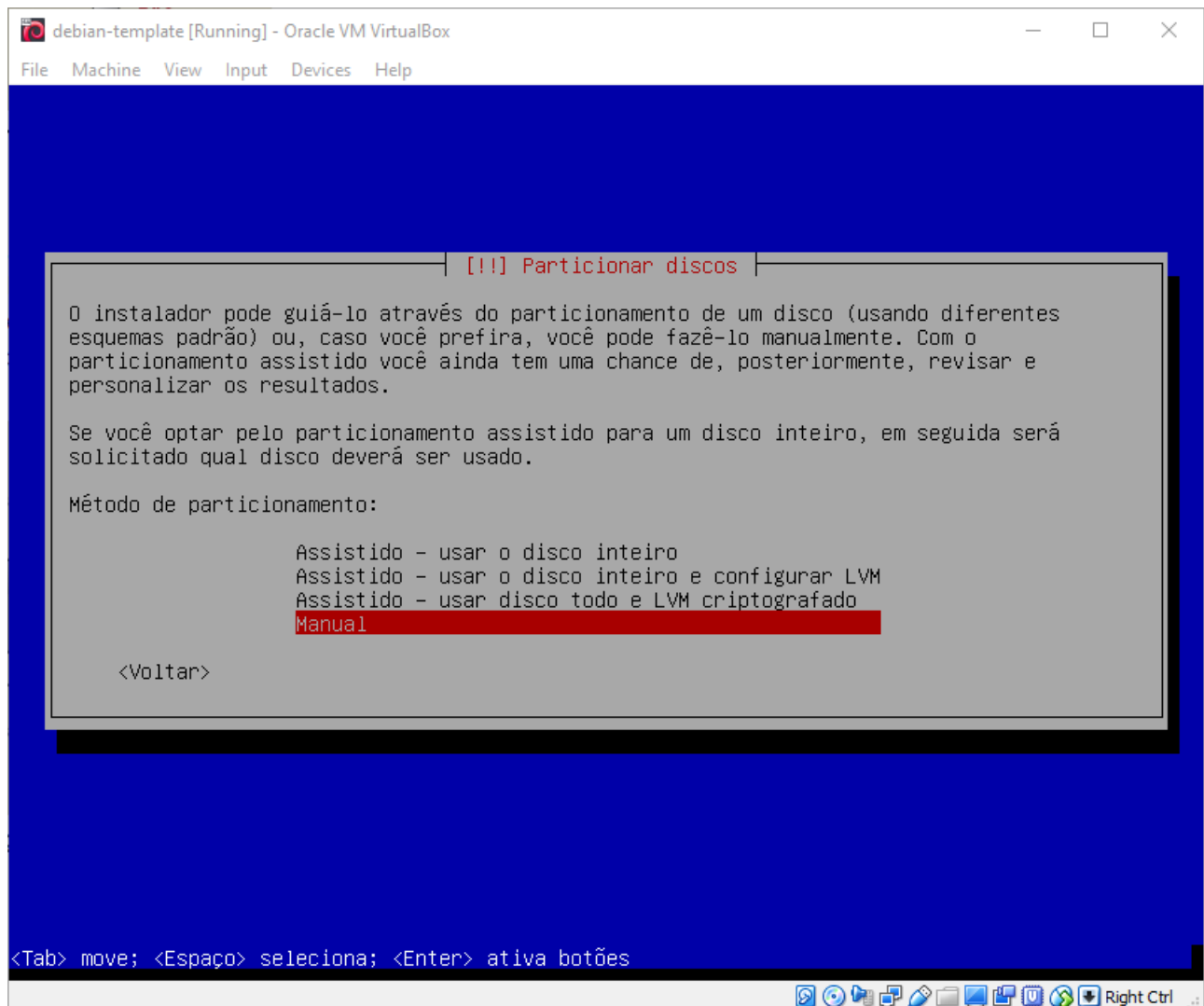


Figura 8. Instalação do Debian Linux, parte 3

8. Na tela seguinte, o primeiro passo é criar uma tabela de partições vazia no disco virtual `/dev/sda`. Coloque o cursor sobre o disco `SCSI1 (0,0,0) (sda) - 8.6 GB ATA VBOX HARDDISK` e pressione `ENTER`. Em seguida, responda *Sim* para a pergunta *Criar nova tabela de partições vazia neste dispositivo?*.
9. Agora, iremos configurar o LVM. Selecione a opção *Configurar o Gerenciador de Volumes Lógicos*, e responda *Sim* para a pergunta *Gravar as mudanças nos discos e configurar LVM?*.

10. Você verá a tela de configuração do LVM, como se segue.

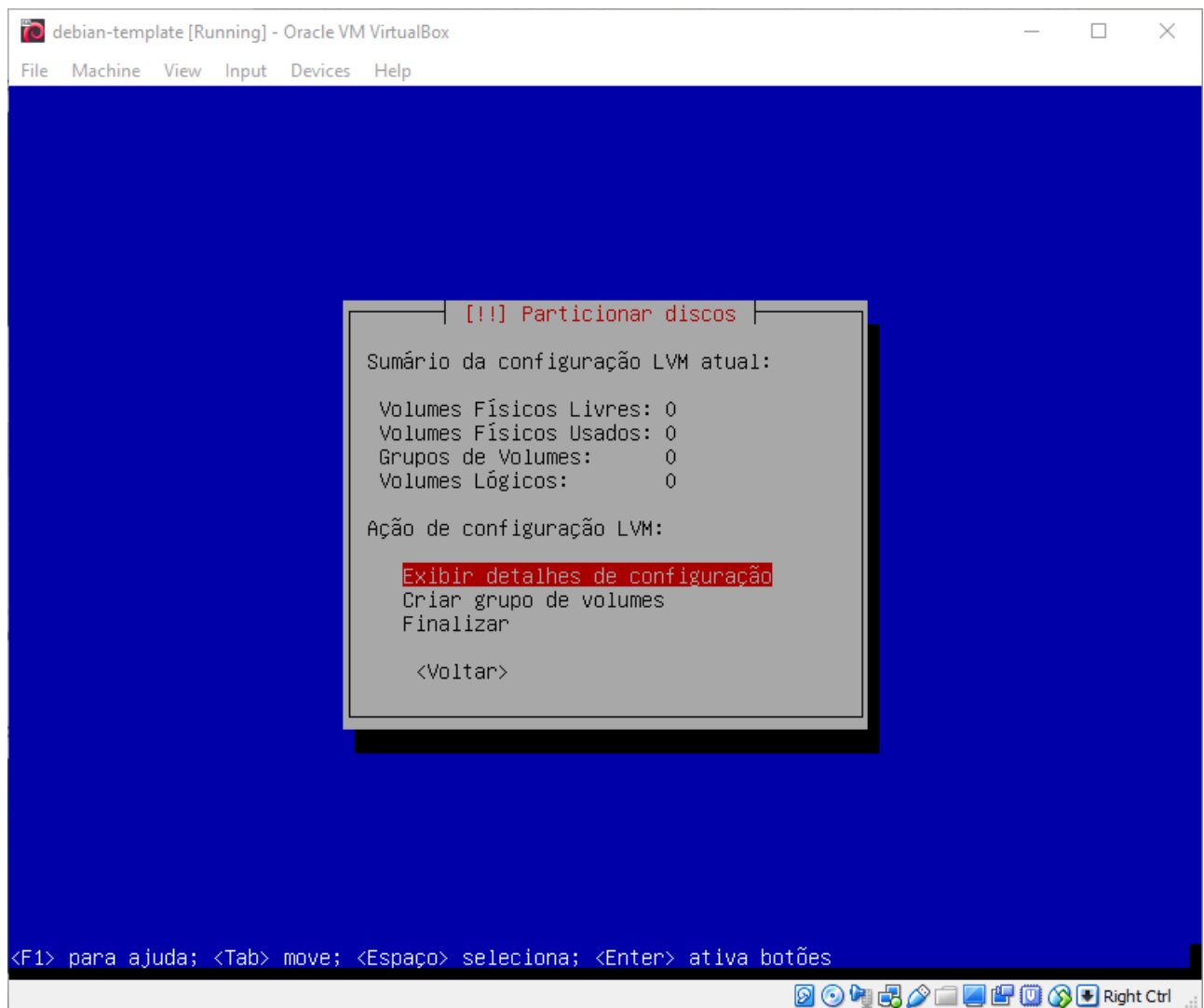


Figura 9. Instalação do Debian Linux, parte 4

A qualquer momento, você pode selecionar a opção *Exibir detalhes de configuração* para verificar o estado atual de configuração do LVM.

O primeiro passo é criar um VG, então escolher *Criar grupo de volumes*. Para o nome do grupo, digite **vg-base**. Em seguida, marque com a tecla **Espaço** os volumes físicos que integrarão esse VG (no caso, apenas o dispositivo **/dev/sda** está disponível), e finalmente responda *Sim* para a pergunta *Gravar as mudanças nos discos e configurar LVM?*. Ao exibir os detalhes de configuração após este passo, você deverá ver a tela a seguir:

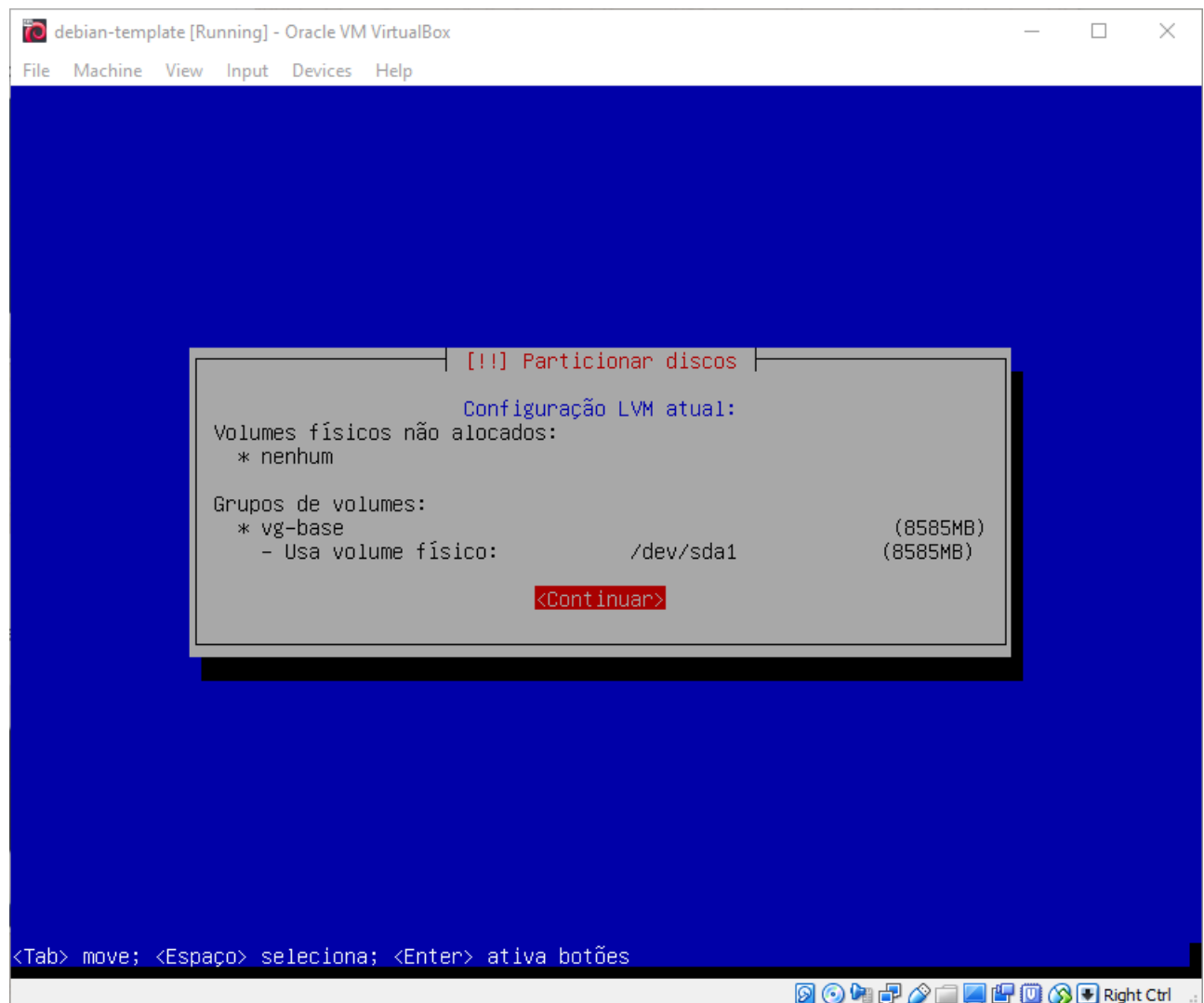


Figura 10. Instalação do Debian Linux, parte 5

11. Todos os volumes físicos (PVs) foram alocados a grupos de volumes (VGs), então agora nos resta criar volumes lógicos (LVs). Com efeito, neste momento é como se estivéssemos particionando um disco no Linux em uma instalação tradicional. Iremos criar três LVs:
 - **lv-boot**: LV que irá armazenar o diretório **/boot** do sistema, com tamanho de 256 MB. É interessante separar o **/boot** para reduzir a complexidade do sistema de arquivos em disco, bem como para aplicar configurações diferenciadas ao *filesystem*, como RAID por software, sistemas de arquivos não-usuais como ZFS, ou caso se deseje criptografar a raiz do sistema.
 - **lv-swap**: LV que irá servir como área de troca (*swap*) do SO em caso de escassez de memória física. Como idealmente não queremos chegar nesse cenário, alocaremos apenas 256 MB para essa área.
 - **lv-root**: LV que irá armazenar a raiz do sistema, **/**, com tamanho de 1536 MB. Pode parecer um valor pequeno, mas considere que iremos separar outros sistemas de arquivos posteriormente.

Imediatamente, podem surgir duas perguntas:

1. **Por que não estamos alocando a totalidade do disco?** O LVM nos permite grande flexibilidade, que será demonstrada em atividades subsequentes. Ao alocarmos $[256 + 256 + 1536] = 2048$ MB em um disco de 8 GB, deixamos (aproximadamente) 6 GB livres que poderão ser alocados de acordo com o tipo específico de uso de cada máquina. Em sentido estrito, a alocação que fizemos acima não é exatamente ideal para um *template*, mas iremos corrigir isso ao demonstrar as funcionalidades do LVM, a seguir.
2. **Por que não criamos LVs para partições como `/tmp`, `/usr` ou `/var`?** De fato, é bastante recomendável separar essas partições em servidores, como veremos a seguir. Iremos criar esses LVs brevemente, após a instalação do SO.

Para criar um LV, selecione *Criar volume lógico*. Em seguida, selecione o grupo de volumes no qual será feita a alocação (no caso, apenas `vg-base` está disponível). Para o nome do primeiro volume lógico, digite `lv-boot`, como delineado acima. Para seu tamanho, defina 256 MB.

Prossiga com a criação dos outros dois LVs, `lv-swap` e `lv-root`, com os tamanhos especificados acima. Ao exibir os detalhes de configuração após este passo, você deverá ver a tela a seguir:

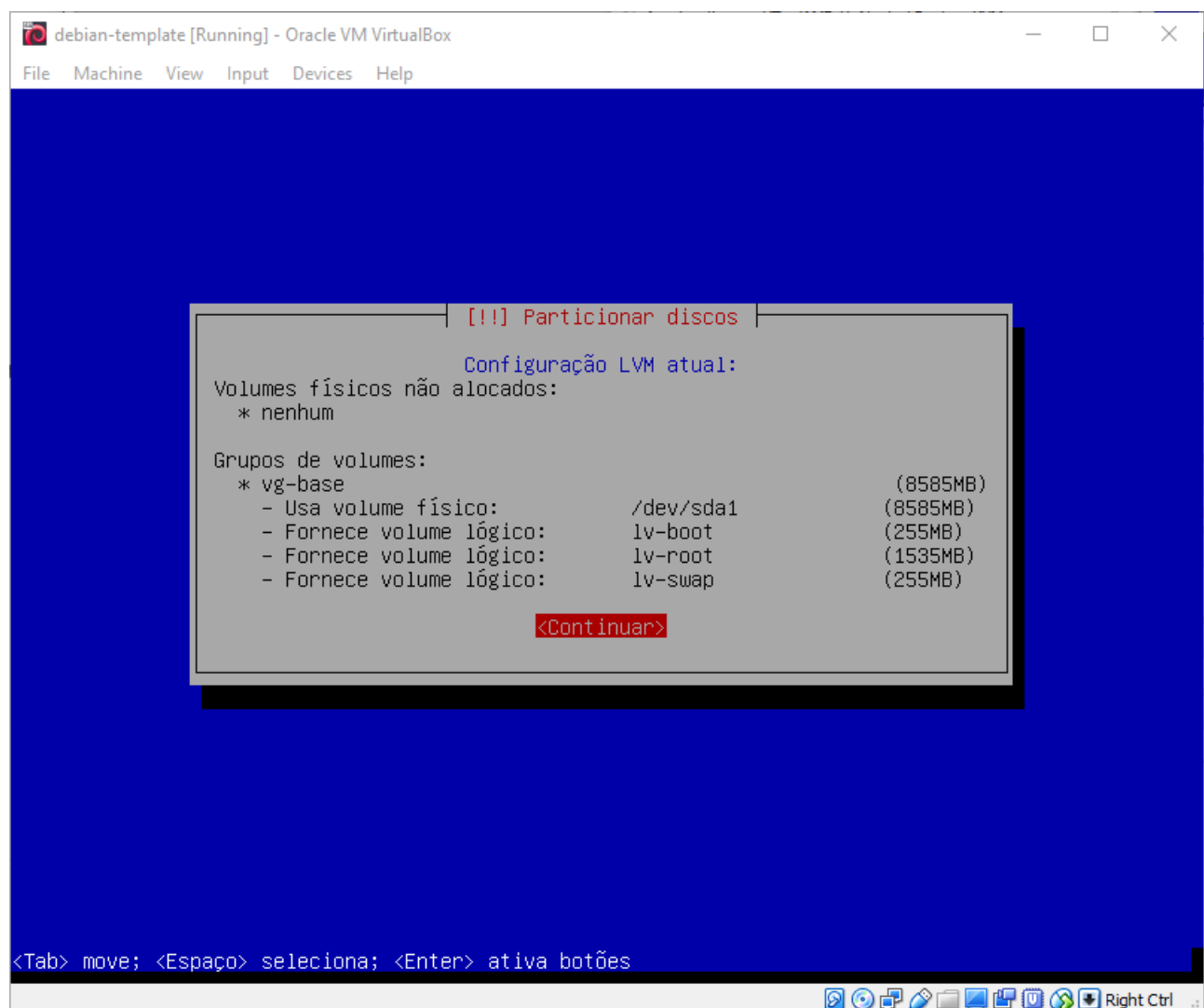


Figura 11. Instalação do Debian Linux, parte 6

Se tudo estiver a contento, selecione *Finalizar*.

12. Ainda não acabou! Neste momento, configuramos o LVM — alocamos volumes físicos, criamos um grupo de volumes agrupando esses PVs e finalmente criamos 3 volumes lógicos dentro do VG. Falta informar ao sistema quais serão os pontos de montagem desses LVs, e quais sistemas de arquivos serão usados. Usaremos a seguinte configuração:

- **lv-boot**: montado sob o diretório **/boot**, formatado em **ext2**.
- **lv-swap**: área de troca (*swap*).
- **lv-root**: montado sob o diretório **/**, formatado em **ext4**.

Para fazer as configurações acima, selecione um dos LVs indicados (por exemplo, deixe o cursor sobre a linha **#1 255.9 MB**, logo abaixo de **lv-boot**), e pressione **ENTER**. Em *Usar como*, escolha *Sistema de arquivos ext2*, e em *Ponto de montagem* selecione **/boot**. Finalmente, selecione *Finalizar a configuração da partição*.

Prossiga com a configuração dos outros dois LVs, **lv-swap** e **lv-root**, de acordo com as características especificadas acima. Ao exibir os detalhes de configuração após este passo, você deverá ver a tela a seguir:

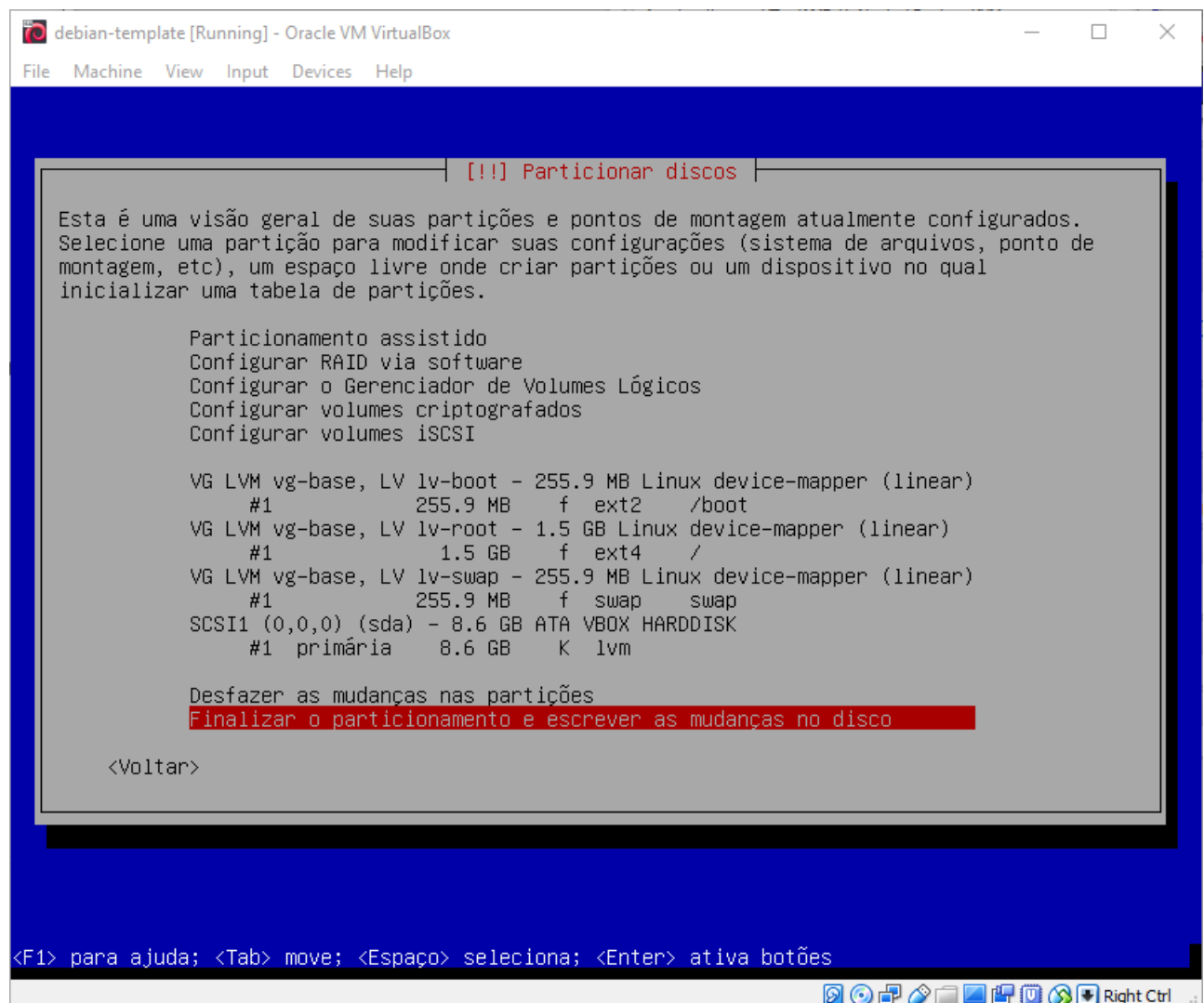


Figura 12. Instalação do Debian Linux, parte 7

Se tudo estiver a contento, selecione *Finalizar o particionamento e escrever as mudanças no disco*. Confirme a pergunta subsequente escolhendo *Sim*. Os discos serão formatados e o

sistema-base do Debian será instalado.

13. O próximo passo será a seleção e instalação de pacotes adicionais. Na pergunta *Selecionar um espelho de rede*, responda *Sim*. Em seguida, selecione *Brasil* como o país do espelho e aponte o servidor ftp.br.debian.org. Quanto à informação do proxy HTTP a ser usado, deixe em branco (a menos que o contrário seja indicado pelo seu instrutor).

O instalador irá fazer o download dos arquivos de índice do repositório de pacotes.

Após algum tempo, surgirá a pergunta *Participar do concurso de utilização de pacotes*—responda *Não*. Em seguida, o `tasksel` será invocado. Nesta tela podemos escolher quais conjuntos de software iremos instalar no disco.

Para servidores de rede, em linhas gerais, é usualmente recomendável não selecionar nada além do estritamente necessário nesta tela, e proceder com a instalação manual de pacotes posteriormente; o `tasksel` geralmente instala um conjunto de pacotes superior ao que objetivamos originalmente, aumentando a superfície de ataque e exposição do sistema. Para ambientes *desktop*, é perfeitamente razoável escolher o ambiente gráfico base e uma das opções de gerenciadores de janelas disponíveis.

Mantenha marcadas apenas as caixas *servidor SSH* e *utilitários de sistema padrão*, e selecione *Continuar*.

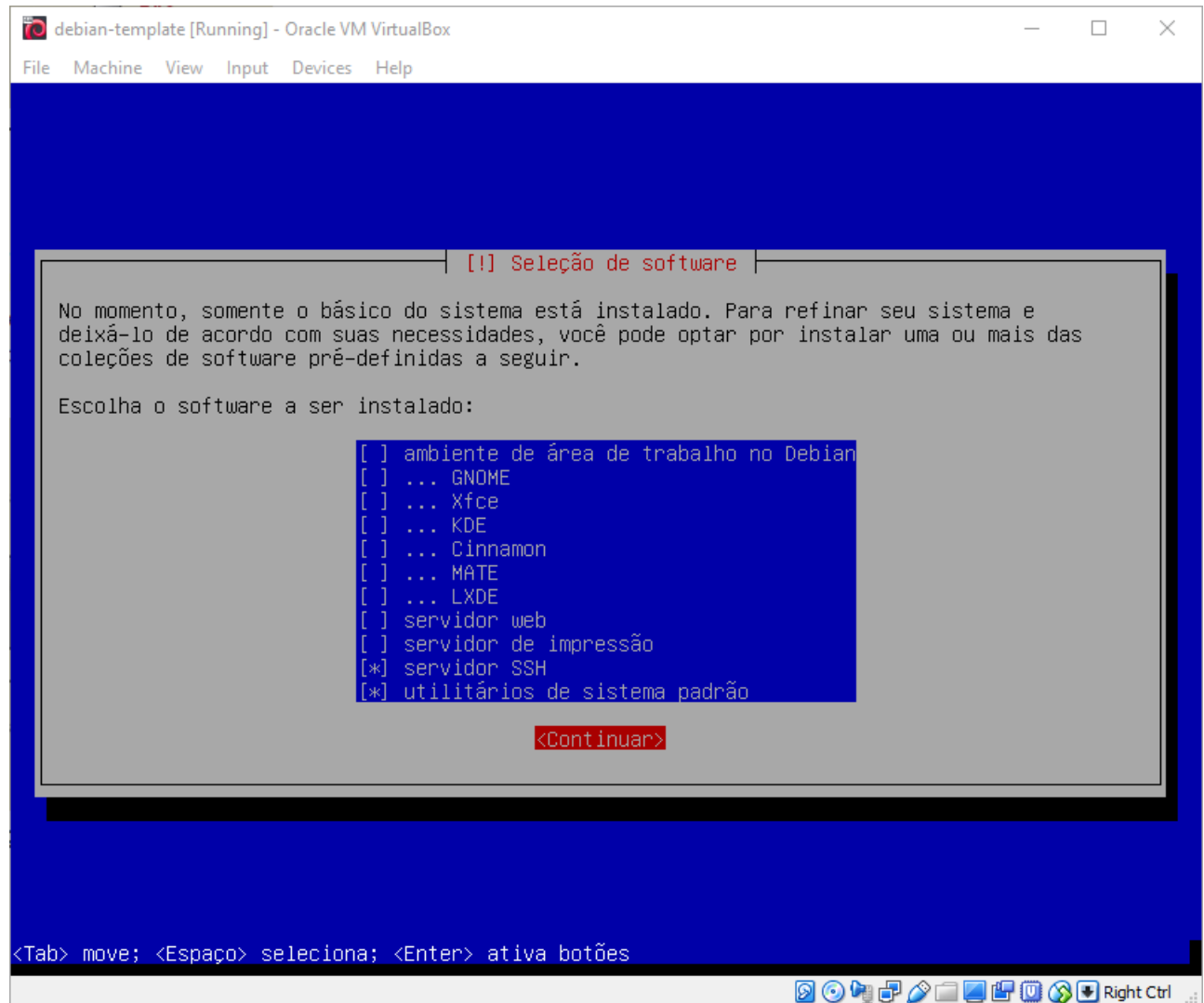


Figura 13. Instalação do Debian Linux, parte 8

O instalador irá fazer o download e instalação dos pacotes selecionados.

14. A última etapa é efetuar a instalação do carregador de inicialização, ou *bootloader*, no sistema. O Debian, assim como a maioria das demais distribuições, utiliza o GRUB (*GRand Unified Bootloader*) como *bootloader* padrão.

Responda *Sim* para a pergunta *Instalar o carregador de inicialização GRUB no registro mestre de inicialização*, e em seguida selecione o dispositivo de instalação */dev/sda* (o único disponível).

15. A instalação está concluída. Selecione *Continuar* para reinicializar a VM no novo sistema instalado.

Após o reboot, você deverá ver a tela de login abaixo:

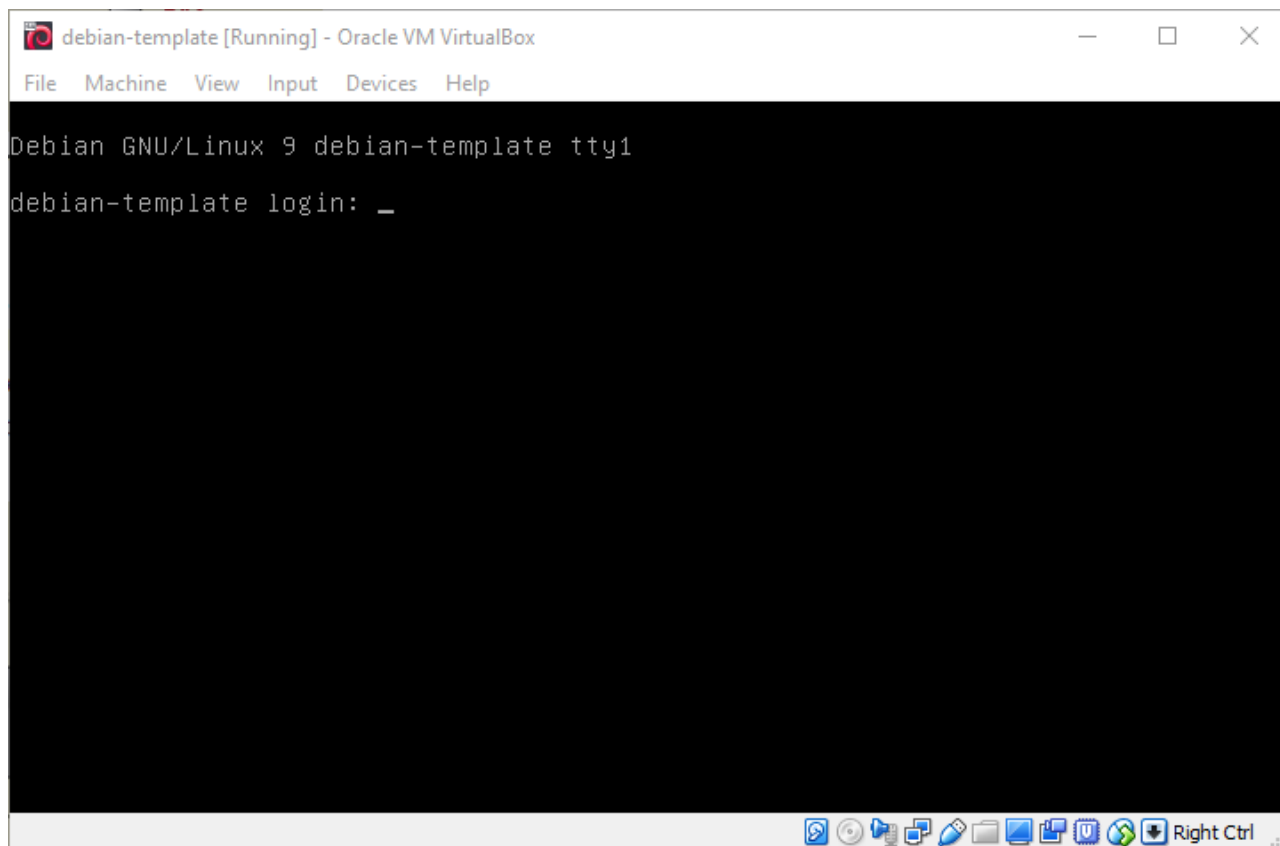


Figura 14. Instalação do Debian Linux, concluída

3) Ajustes pós-instalação

Instalado nosso *template*, iremos continuar sua configuração para torná-lo, de fato, uma imagem-base de boa qualidade para ser usada em derivações de VMs futuras. Além de corrigir a situação dos volumes lógicos (que deixamos incompleta propositalmente durante a instalação), iremos também fazer algumas configurações de base com relação aos repositórios, atualização de pacotes e contas de usuário.

1. Faça login na máquina *debian-template* como usuário *root*, usando a senha *rnpsr*. Imediatamente após o login, você verá uma mensagem parecida com a que se segue:

```
Linux debian-template 4.9.0-7-amd64 #1 SMP Debian 4.9.110-3+deb9u2 (2018-08-13)
x86_64
```

```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

```
# hostname  
debian-template
```

```
# whoami  
root
```

Essa mensagem é definida no arquivo `/etc/motd`, conhecida como *message of the day* (ou "mensagem do dia"). É interessante customizar essa mensagem para refletir o ambiente local, avisando o administrador sobre os requisitos legais para operar máquinas da organização, ou informando sobre onde encontrar documentação sobre os servidores. Lembre-se que, já que estamos mexendo no *template*, essa mensagem será copiada para todas as VMs derivadas desta imagem.

Neste curso, vamos deixar o *motd* vazio. Execute:

```
# echo "" > /etc/motd
```

Saia da sessão corrente usando `exit` ou `CTRL + D`, e logue novamente. Note que a mensagem anterior será suprimida.

2. Ao longo do curso, iremos editar vários arquivos de texto em ambiente Linux. Há vários editores de texto disponíveis para a tarefa, como o `vi`, `emacs` ou `nano`. Caso você não esteja familiarizado com um editor de texto, recomendamos o uso do `nano`, que possui uma interface bastante amigável para usuários iniciantes. Para editar um arquivo com o `nano`, basta digitar `nano` seguido do nome do arquivo a editar — não é necessário que o arquivo tenha sido criado previamente:

```
# nano teste
```

Digite livremente a seguir. Use as setas do teclado para navegar no texto, e `DELETE` ou `BACKSPACE` para apagar texto. O `nano` possui alguns atalhos interessantes, como:

- `CTRL + G`: Exibir a ajuda do editor
- `CTRL + X`: Fechar o `buffer` de arquivo atual (que pode ser um texto sendo editado, ou o painel de ajuda), e sair do `nano`. Para salvar o arquivo, digite `Y` (*yes*) ou `S` (*sim*) para confirmar as mudanças ao arquivo, opcionalmente altere o nome do arquivo a ser escrito no disco, e digite `ENTER`.
- `CTRL + O`: Salvar o arquivo no disco sem sair do editor.
- `CTRL + W`: Buscar padrão no texto.
- `CTRL + K`: Cortar uma linha inteira e salvar no `buffer` do editor.
- `CTRL + U`: Colar o `buffer` do editor na posição atual do cursor. Pode ser usado repetidamente.

Para salvar e sair do texto sendo editado, como mencionado acima, utilize `CTRL + X`.

3. Edite o arquivo `/etc/network/interfaces` como se segue, reinicie a rede e verifique o funcionamento:

```
# nano /etc/network/interfaces
(...)
```

```
# cat /etc/network/interfaces
source /etc/network/interfaces.d/*

auto lo enp0s3

iface lo inet loopback

iface enp0s3 inet dhcp
```

```
# systemctl restart networking
```

```
# ip a s | grep '^ *inet '
    inet 127.0.0.1/8 scope host lo
    inet 192.168.29.104/24 brd 192.168.29.255 scope global enp0s3
```

Desabilite a verificação de *hostnames* remotos do `ssh` para agilizar o procedimento de login. Reinicie o *daemon* posteriormente.

```
# sed -i '/UseDNS/s/^#//' /etc/ssh/sshd_config
```

```
# systemctl restart ssh
```

4. Durante as atividades deste curso iremos ter que digitar vários comandos no terminal das VMs, os quais serão mostrados nos cadernos de atividade de cada sessão. Alguns desses comandos serão bastante longos e/ou terão uma sintaxe complicada — nesse caso, o ideal é que tenhamos a possibilidade de copiá-los diretamente do caderno para a console, evitando erros de digitação.

O protocolo de login remoto SSH é ideal para solucionar essa tarefa. Em ambiente Windows, dois dos métodos mais populares para efetuar logins remotos via SSH são os programas PuTTY (<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>) ou Cygwin (<https://cygwin.com/install.html>). Vamos, primeiro, visualizar os passos necessários usando o PuTTY.

Em qualquer caso, o primeiro passo é sempre descobrir qual o endereço IP da máquina remota à qual queremos nos conectar. Para isso digite, na máquina `debian-template`:

```
# ip a s enp0s3 | grep '^ *inet ' | awk '{print $2}' | cut -d '/' -f1  
192.168.29.104
```

O uso do **PuTTY**, por se tratar de um programa *standalone* com o objetivo único de efetuar login via SSH, é mais simples. Faça o download do PuTTY em sua máquina física Windows, usando a URL informada acima. Em seguida, apenas abra o programa e digite na caixa *Host Name* o endereço IP da máquina remota descoberto acima. Em seguida, clique em *Open*.

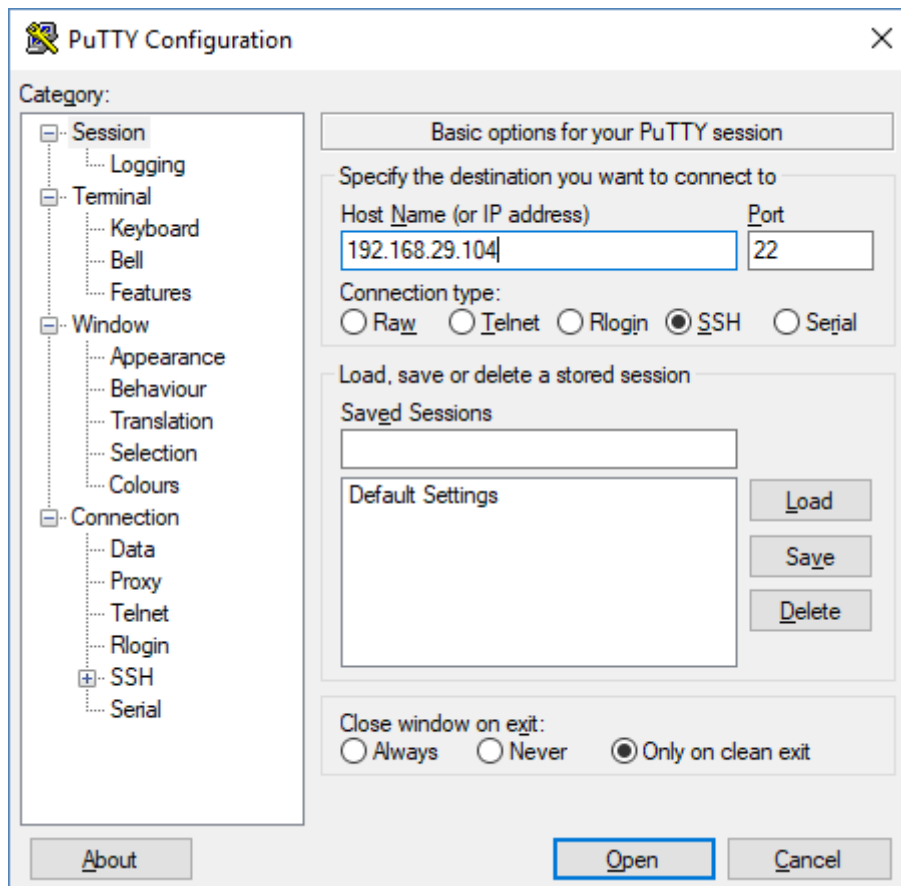


Figura 15. Login via SSH usando o PuTTY, parte 1

Será mostrado um alerta de segurança avisando que a chave do *host* remoto não se encontra na *cache* local, o que pode configurar um risco de segurança. Clique em *Yes* para prosseguir com a tentativa de login.

Em seguida, será solicitado o nome de usuário com o qual efetuar a conexão. Como, por padrão, o *daemon sshd* do Debian não permite logins remotos usando o *root*, escolha o usuário *aluno*. Para a senha, informe *rnpsr*. Em caso de sucesso, você verá a tela a seguir:

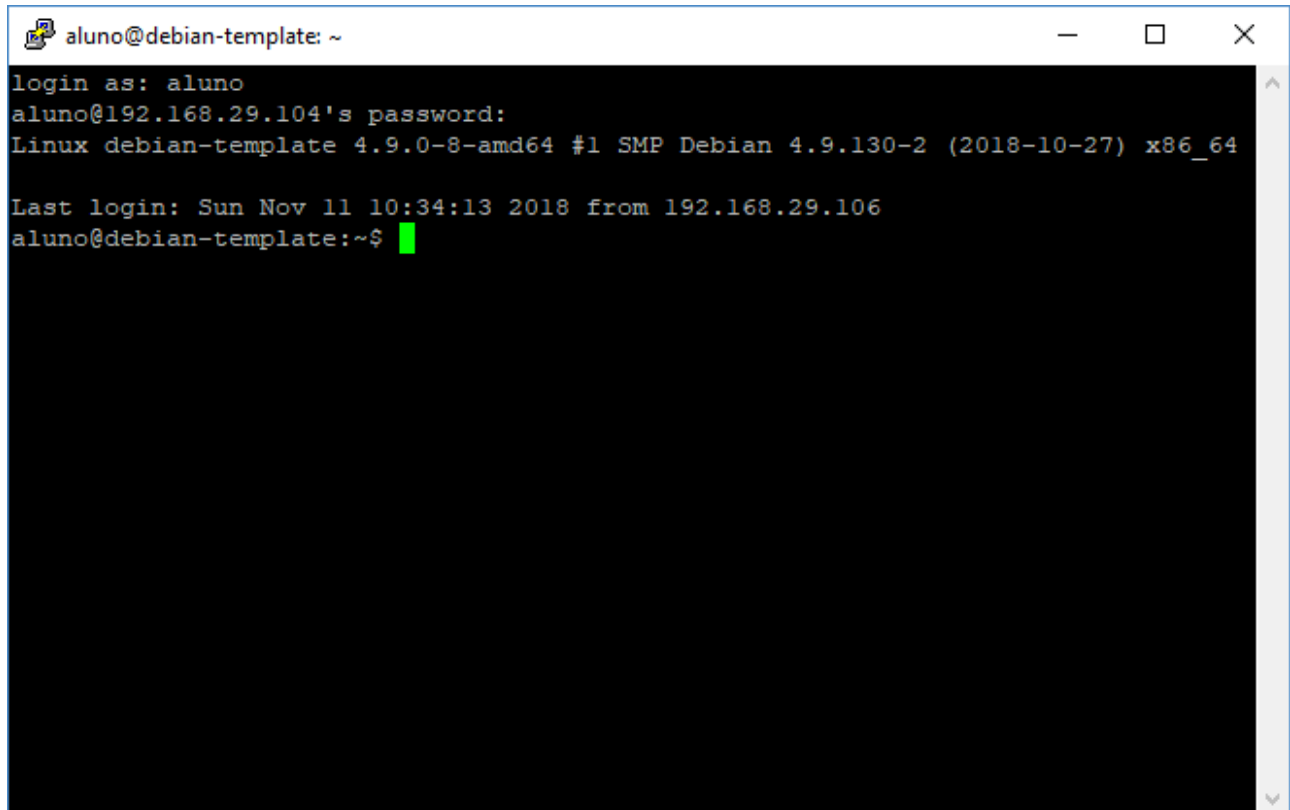


Figura 16. Login via SSH usando o PuTTY, parte 2

Para tornar-se o superusuário **root**, agora, basta executar o comando **su -** e informar a senha correta, como se segue:

```
$ su -  
Senha:
```

```
# whoami  
root
```

Para copiar/colar comandos no PuTTY, basta selecionar o texto desejado no ambiente da máquina física e digitar **CTRL + C**, e em seguida clicar com o botão direito na janela do PuTTY. O texto selecionado será colado na posição do cursor.

5. O uso do Cygwin é um pouco mais envolvido, já que seu objetivo é mais complexo: prover, em ambiente Windows, funcionalidade equivalente à que temos disponível em uma distribuição Linux. Para começar, faça o download e execute o instalador do Cygwin em sua máquina física Windows.

A instalação é, em grande parte, bastante similar à de qualquer aplicativo Windows. Na tela inicial, clique em *Next*. Em *Choose a Download Source*, mantenha marcada a caixa *Install from Internet* e clique em *Next*. Em *Select Root Install Directory*, os valores padrão estão apropriados — clique em *Next*. Na tela *Select Local Package Directory*, novamente, mantenha o valor padrão e clique em *Next*.

Agora, vamos selecionar a fonte de pacotes. Em *Select Your Internet Connection*, a menos que

haja um *proxy* na rede local (informe-se com seu instrutor), mantenha marcada a caixa *Direct Connection* e clique em *Next*. Será feito o download da lista de espelhos disponíveis para o Cygwin. Em *Choose A Download Site*, qualquer espelho irá funcionar, mas evidentemente é desejável que escolhamos um que possua maior velocidade de download—o site <http://linorg.usp.br> é provavelmente uma boa opção, nesse caso. Clique em *Next*, e o instalador irá baixar a lista de pacotes disponíveis.

Em adição ao sistema-base padrão, é necessário instalar o OpenSSH para efetuar logins remotos. Na caixa de busca *Search*, no topo da tela, digite o termo de busca *openssh*. Expanda a árvore *Net* e clique na palavra *Skip* na linha do pacote *openssh: The OpenSSH server and client programs*—ela irá alterar para a versão a ser instalada, *7.9p1-1* no caso da figura mostrada abaixo:

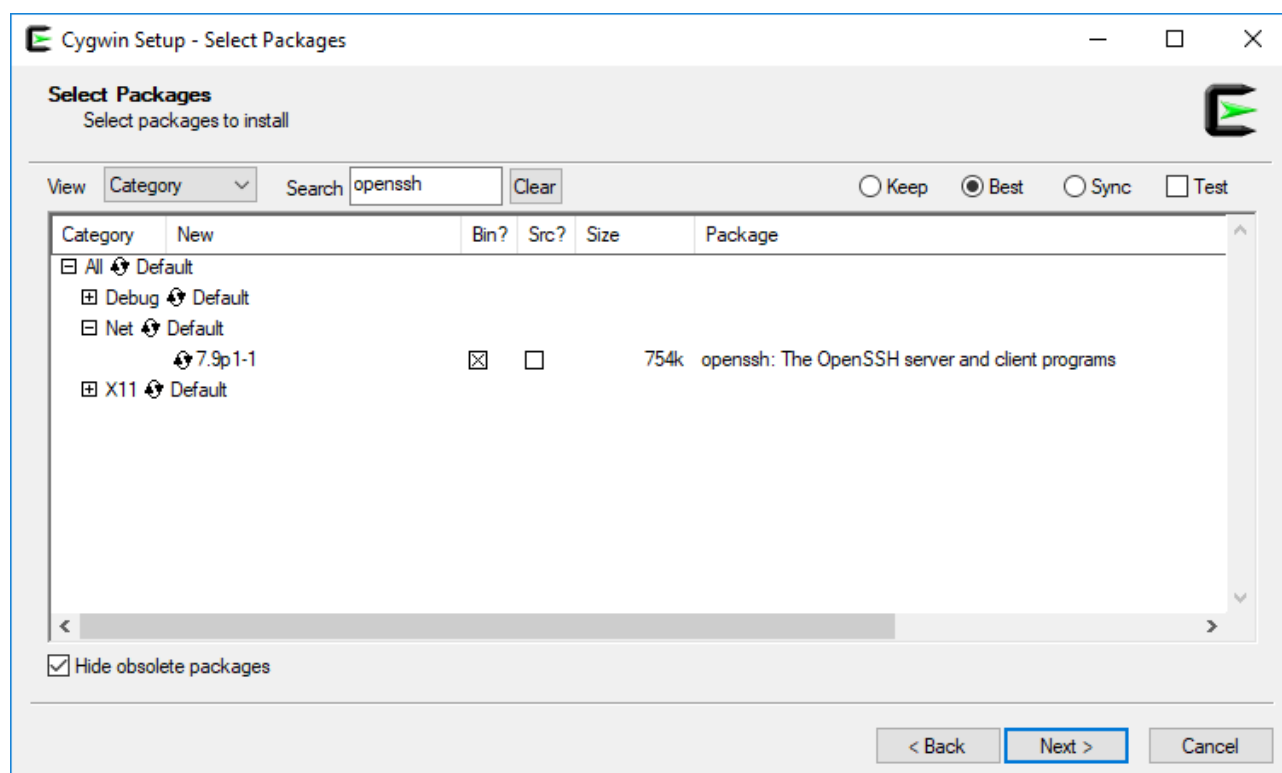
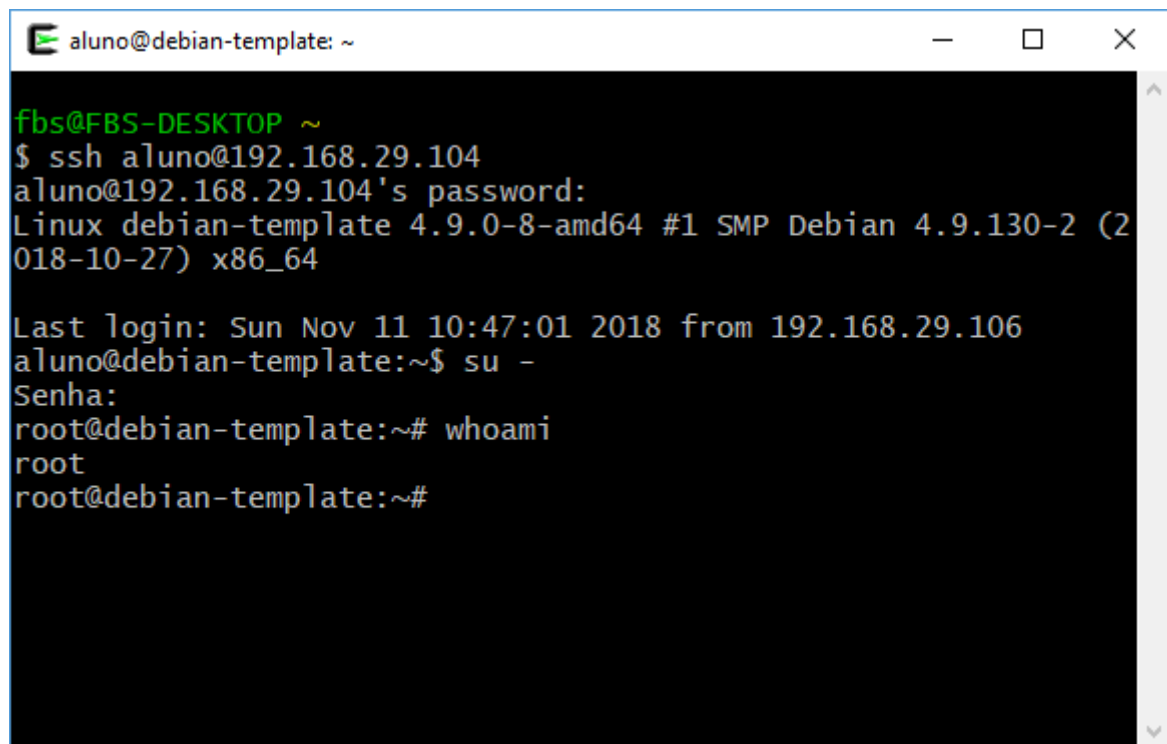


Figura 17. Instalação do OpenSSH no Cygwin

Clique em *Next*. Em *Review and confirm changes*, verifique que o Cygwin irá instalar o OpenSSH e todas as demais dependências do sistema-base Linux, como o *shell bash* ou ferramentas como o *grep*, e clique em *Next*. O instalador irá fazer o download e instalação dos pacotes selecionados.

Concluído o processo, procure pelo programa *Cygwin Terminal* no menu iniciar da sua máquina física Windows, e execute-o. Agora, tente fazer login via SSH normalmente, como se estivesse em um *shell* Linux:

A terminal window titled 'aluno@debian-template: ~' with standard window controls. The terminal output shows a user 'fbs@FBS-DESKTOP' connecting via SSH to 'aluno@192.168.29.104'. The user provides a password and is greeted with system information: 'Linux debian-template 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64'. The last login is noted as 'Sun Nov 11 10:47:01 2018 from 192.168.29.106'. The user then runs 'su -', enters a password, and becomes root. The root user runs 'whoami', which returns 'root'.

```
aluno@debian-template: ~  
fbs@FBS-DESKTOP ~  
$ ssh aluno@192.168.29.104  
aluno@192.168.29.104's password:  
Linux debian-template 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) x86_64  
  
Last login: Sun Nov 11 10:47:01 2018 from 192.168.29.106  
aluno@debian-template:~$ su -  
Senha:  
root@debian-template:~# whoami  
root  
root@debian-template:~#
```

Figura 18. Login via SSH usando o Cygwin

Para copiar/colar comandos no Cygwin, basta selecionar o texto desejado no ambiente da máquina física e digitar **CTRL + C**, e em seguida mudar o foco para a janela do Cygwin e digitar a combinação **SHIFT + Insert**. Para copiar texto a partir da janela do Cygwin, selecione-o e use a combinação de teclas **CTRL + Insert**. Para encontrar os arquivos localizados em sua máquina física, o diretório **/cygdrive/X** pode ser usado para mapear para os discos da máquina local—por exemplo, o diretório **/cygdrive/c** mapeia diretamente para o **C:** da máquina Windows.

6. A seguir, vamos checar a configuração dos repositórios de pacotes sendo usados pelo sistema. Cheque o conteúdo do arquivo **/etc/apt/sources.list**:

```
# cat /etc/apt/sources.list
#

# deb cdrom:[Debian GNU/Linux 9.5.0 _Stretch_ - Official amd64 xfce-CD Binary-1
20180714-10:25]/ stretch main

deb cdrom:[Debian GNU/Linux 9.5.0 _Stretch_ - Official amd64 xfce-CD Binary-1
20180714-10:25]/ stretch main

deb http://ftp.br.debian.org/debian/ stretch main
deb-src http://ftp.br.debian.org/debian/ stretch main

deb http://security.debian.org/debian-security stretch/updates main
deb-src http://security.debian.org/debian-security stretch/updates main

# stretch-updates, previously known as 'volatile'
deb http://ftp.br.debian.org/debian/ stretch-updates main
deb-src http://ftp.br.debian.org/debian/ stretch-updates main
```

Temos algumas linhas desnecessárias neste arquivo: primeiro, as entradas **deb cdrom** referem-se ao CD de instalação do Debian, que usamos durante a atividade (2) desta sessão; além dessas, as entradas **deb-src** referem-se a pacotes de código-fonte, que podem ser baixados com o comando **apt-get source** e posteriormente compilados pelo administrador. Via de regra, não usaremos quaisquer dessas entradas em um sistema de produção, então elas podem ser removidas com segurança.

Sobre os componentes (ou seções) do repositório, note que apenas a **main** está incluída no arquivo acima. O Debian possui três seções principais:

- **main**: contém apenas software compatível com a DFSG (*Debian Free Software Guidelines*, ou diretrizes de software livre do Debian), e não dependem de software fora desta seção para funcionar. Esses são os pacotes considerados parte integrante da distribuição de software do Debian.
- **contrib**: contém pacotes compatíveis com a DFSG, mas que possuem dependências que não estão na seção **main**.
- **non-free**: contém software incompatível com a DFSG.

Em geral, não há grandes restrições para incluir todas as três seções de software detalhadas acima em uma organização. Assim, iremos incluir as duas seções faltantes, **contrib** e **non-free**, no arquivo **/etc/apt/sources.list**. Edite-o usando o **vi** ou o **nano**:

```
# nano /etc/apt/sources.list
(...)
```

Após a edição, seu conteúdo deverá ficar assim:

```
# cat /etc/apt/sources.list
deb http://ftp.br.debian.org/debian/ stretch main contrib non-free
deb http://ftp.br.debian.org/debian/ stretch-updates main contrib non-free
deb http://security.debian.org/debian-security stretch/updates main contrib non-free
```

7. Atualize a lista de pacotes disponíveis nos repositórios remotos usando o comando:

```
# apt-get update
```

Em seguida, vamos garantir que nosso *template* está plenamente atualizado com:

```
# apt-get dist-upgrade -y
```

Caso o kernel do sistema tenha sido atualizado no processo (pacotes com o nome `linux-image-*`), será necessário reiniciar a máquina para realizar o *boot* com o novo kernel. Faça isso, se for o caso, com o comando `reboot`.

Após o reinício, logue novamente como o usuário `root`. Para remover os binários dos pacotes recentemente instalados do sistema, execute `apt-get clean`. Se quiser remover as dependências de pacotes que estão instaladas no sistema e já não são mais necessárias, rode `apt-get autoremove`.

Para remover todos os kernels antigos do sistema (isto é, todos os kernels exceto o que está em execução no momento), você pode executar:

```
# dpkg -l | egrep 'linux-image-[0-9\.-]*-amd64' | awk '{print $2}' | grep -v
$(uname -r) | xargs apt-get purge -y
```

8. Este é um bom momento para instalar pacotes que você acredita serem necessários em todas as máquinas a serem derivadas deste *template*. Pacotes como o `sudo` ou `vim` podem ser boas opções, dependendo das necessidades da sua organização.

Neste momento, iremos instalar apenas os pacotes `rsync`, `nfs-common` e `sudo`. Execute:

```
# apt-get install rsync nfs-common sudo
```

9. Pode ser interessante desabilitar a combinação de teclas `CTRL + ALT + DEL`; mesmo em um ambiente virtualizado, há casos em que o administrador se confunde e acaba enviando essa combinação de teclas para a console de um servidor aberto, causando seu *reboot* inadvertidamente.

Note que, por padrão, essa combinação de teclas aponta para o `reboot.target`, um *target* (ou alvo) do `systemd` que é responsável pelo reinício do sistema operacional.

```
# ls -ld /lib/systemd/system/ctrl-alt-del.target
lrwxrwxrwx 1 root root 13 jun 13 17:20 /lib/systemd/system/ctrl-alt-del.target ->
reboot.target
```

Para desabilitar o **CTRL + ALT + DEL**, basta executar:

```
# systemctl mask ctrl-alt-del.target
Created symlink /etc/systemd/system/ctrl-alt-del.target → /dev/null.
```

10. O *template* atual (a máquina **debian-template**) será usada como base para várias VMs futuras, como estabelecido. Ao copiar a máquina, a primeira ação a ser realizada será sempre alterar o *hostname* para o nome da nova máquina — pode ser muito interessante ter um meio para automatizar essa tarefa, como um *shell script*.

Crie um novo arquivo **/root/scripts/changehost.sh** (crie o diretório **/root/scripts** se este não existir), com o seguinte conteúdo:

```
1 #!/bin/bash
2
3
4 usage() {
5     echo " Usage: $0 HOSTNAME"
6     exit 1
7 }
8
9
10 # testar parametros
11 [ -z $1 ] && usage
12
13 # testar sintaxe valida
14 if [[ "$1" =~ [^a-z0-9] ]]; then
15     echo " HOSTNAME must be lowercase alphanumeric: [a-z0-9]*"
16     usage
17 elif [ ${#1} -gt 63 ]; then
18     echo " HOSTNAME must have <63 chars"
19     usage
20 fi
21
22 # alterar hostname local
23 chost="$( hostname -s )"
24 sed -i "s/${chost}/${1}/g" /etc/hosts
25 sed -i "s/${chost}/${1}/g" /etc/hostname
26
27 invoke-rc.d hostname.sh restart
28 invoke-rc.d networking force-reload
29 hostnamectl set-hostname $1
30
31 # re-gerar chaves SSH
32 rm -f /etc/ssh/ssh_host_* 2> /dev/null
33 dpkg-reconfigure openssh-server &> /dev/null
```

O *script* acima espera receber um único parâmetro: o novo *hostname* a ser configurado para a máquina. Após checar se o parâmetro possui sintaxe válida, o *script* irá alterar o nome da máquina nos arquivos */etc/hostname* e */etc/hosts*, reiniciar as interfaces de rede e *daemons* relevantes, e finalmente re-gerar as chaves de host do *ssh* com o novo nome da máquina.

4) Configuração do LVM

1. Vamos prosseguir com a configuração do LVM, que fizemos apenas parcialmente durante a instalação — para relembrar, configuramos três volumes lógicos (LVs), *lv-boot*, *lv-swap* e *lv-root*. Para verificar o estado dos volumes lógicos em um sistema Linux, execute o comando *lvdisplay*:

```
# lvsdisplay | grep 'Logical volume\|LV Path\|LV Name\|LV Size'
--- Logical volume ---
LV Path                /dev/vg-base/lv-boot
LV Name                 lv-boot
LV Size                 244,00 MiB
--- Logical volume ---
LV Path                /dev/vg-base/lv-swap
LV Name                 lv-swap
LV Size                 244,00 MiB
--- Logical volume ---
LV Path                /dev/vg-base/lv-root
LV Name                 lv-root
LV Size                 1,43 GiB
```

Para verificar o estado dos grupos de volumes (VGs), execute **vgdisplay**:

```
# vgdisplay
--- Volume group ---
VG Name                 vg-base
System ID
Format                  lvm2
Metadata Areas          1
Metadata Sequence No    4
VG Access                read/write
VG Status                resizable
MAX LV                  0
Cur LV                  3
Open LV                  3
Max PV                   0
Cur PV                  1
Act PV                   1
VG Size                  8,00 GiB
PE Size                  4,00 MiB
Total PE                 2047
Alloc PE / Size          488 / 1,91 GiB
Free PE / Size            1559 / 6,09 GiB
VG UUID                  h0XyJN-XA3n-i4RG-4mhJ-rDF5-edi7-xXS5f0
```

E, finalmente, para verificar o estado dos volumes físicos (PVs), execute **pvddisplay**:

```
# pvdiskdisplay
--- Physical volume ---
PV Name           /dev/sda1
VG Name           vg-base
PV Size           8,00 GiB / not usable 2,00 MiB
Allocatable       yes
PE Size           4,00 MiB
Total PE          2047
Free PE           1559
Allocated PE      488
PV UUID           ZnHMMn-Y37D-6Psd-oHei-K1Qd-wHjY-6gqFZ3
```

2. Vamos criar três novos LVs, com as configurações que se seguem:

- **lv-tmp**: armazenará o diretório **/tmp**, com tamanho de 512 MB.
- **lv-var**: armazenará o diretório **/var**, com tamanho de 1536 MB.
- **lv-usr**: armazenará o diretório **/usr**, ocupando todo o tamanho restante do VG **vg-base**.

Para criá-los, execute os comandos:

```
# lvcreate -L 512M -n lv-tmp vg-base
Logical volume "lv-tmp" created.
```

```
# lvcreate -L 1536M -n lv-var vg-base
Logical volume "lv-var" created.
```

```
# lvcreate -l 100%FREE -n lv-usr vg-base
Logical volume "lv-usr" created.
```

Vamos verificar como ficou a situação dos nossos volumes lógicos:

```
# lvsdisplay | grep 'Logical volume\|LV Path\|LV Name\|LV Size'
--- Logical volume ---
LV Path                /dev/vg-base/lv-boot
LV Name                 lv-boot
LV Size                244,00 MiB
--- Logical volume ---
LV Path                /dev/vg-base/lv-swap
LV Name                 lv-swap
LV Size                244,00 MiB
--- Logical volume ---
LV Path                /dev/vg-base/lv-root
LV Name                 lv-root
LV Size                1,43 GiB
--- Logical volume ---
LV Path                /dev/vg-base/lv-tmp
LV Name                 lv-tmp
LV Size                512,00 MiB
--- Logical volume ---
LV Path                /dev/vg-base/lv-var
LV Name                 lv-var
LV Size                1,50 GiB
--- Logical volume ---
LV Path                /dev/vg-base/lv-usr
LV Name                 lv-usr
LV Size                4,09 GiB
```

Naturalmente, o VG está ocupado em sua totalidade, agora:

```
# vgsdisplay | grep PE
PE Size                4,00 MiB
Total PE               2047
Alloc PE / Size        2047 / 8,00 GiB
Free PE / Size         0 / 0
```

O mesmo pode ser dito para o PV:

```
# pvsdisplay | grep PE
PE Size                4,00 MiB
Total PE               2047
Free PE                0
Allocated PE           2047
```

3. Apesar de termos criado os LVs para os diretórios `/tmp`, `/var` e `/usr`, nosso trabalho ainda não acabou—temos que formatar esses diretórios, copiar o conteúdo dos diretórios atuais para dentro dos novos, configurar a montagem automática via `/etc/fstab` e apagar os diretórios antigos para liberar espaço.

Primeiro, vamos formatar os LVs usando o sistema de arquivos **ext4**:

```
# mkfs.ext4 /dev/mapper/vg--base-lv--tmp
(...)
```

```
# mkfs.ext4 /dev/mapper/vg--base-lv--var
(...)
```

```
# mkfs.ext4 /dev/mapper/vg--base-lv--usr
(...)
```

Apesar de não termos feito nos exemplos acima, este seria um excelente momento para customizar aspectos do sistema de arquivos para adequá-lo aos tipos específicos de arquivos que serão armazenados ali dentro. Por exemplo, pode ser interessante escolher um tamanho de *inode* menor do que o padrão caso se deseje armazenar muitos arquivos pequenos, como é frequentemente o caso em servidores de e-mail, digamos. Para mais informações, consulte a página de manual [man 8 mke2fs](#).

Uma curiosidade: o tamanho padrão de *inodes* de novos sistemas de arquivos formatados é definido em `/etc/mke2fs.conf`; este valor pode ser customizado através da *flag* `-I` no comando `mkfs.ext*`.

4. O próximo passo é montar esses sistemas de arquivo e sincronizar o conteúdo dos diretórios atuais (que estão dentro da raiz, `/`) com os novos diretórios. Crie um *shell script*, `/root/scripts/syncdirs.sh`, com o seguinte conteúdo:

```
1 #!/bin/bash
2
3 for d in tmp var usr; do
4   [ -d /mnt/${d} ] || mkdir /mnt/${d}
5   mount /dev/mapper/vg--base-lv--${d} /mnt/${d}
6   rsync -av /${d}/ /mnt/${d}
7   umount /mnt/${d}
8   rmdir /mnt/${d}
9 done
```

O script acima irá iterar sobre os nomes `tmp`, `var` e `usr`, com o nome `DIR`. Para cada um deles, fará os passos a seguir:

1. Criar o diretório `/mnt/DIR`, se não existir.
2. Montar o volume lógico `lv-DIR` dentro do diretório `/mnt/DIR`.
3. Usando o comando `rsync`, copiar o conteúdo do diretório `/DIR` para `/mnt/DIR`.
4. Desmontar o volume lógico `lv-DIR`.
5. Remover a pasta `/mnt/DIR`, se vazia.

Execute o *script*:

```
# bash ~/scripts/syncdirs.sh

(...)

sent 551,267 bytes  received 2,023 bytes  368,860.00 bytes/sec
total size is 409,038,950  speedup is 739.28
```

5. Vamos configurar a montagem automáticas dos novos volumes lógicos. Edite o arquivo `/etc/fstab` e adicione as linhas a seguir:

```
# nano /etc/fstab
(...)
```

```
# tail -n3 /etc/fstab
/dev/mapper/vg--base-lv--tmp /tmp ext4 defaults 0 2
/dev/mapper/vg--base-lv--var /var ext4 defaults 0 2
/dev/mapper/vg--base-lv--usr /usr ext4 defaults 0 2
```

Note que estamos usando as opções de montagem `defaults`, no exemplo acima. Segundo a página de manual do comando `mount` (que pode ser acessada através do comando `man 8 mount`), essa opção equivale a `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, `async`.

Considerando o uso das partições acima, pode ser interessante do ponto de vista de *hardening* tornar a montagem um pouco mais restritiva. Considere as seguintes opções:

- `ro`: montar o sistema de arquivos em modo somente-leitura. Inviável para diretórios como `/tmp` ou `/var`, embora possa ser considerado para o `/usr`, por exemplo. O grande inconveniente dessa proteção é que a permissão de escrita terá que ser atribuída manualmente sempre que se quiser escrever no diretório (digamos, durante a instalação de um novo pacote), motivo pelo qual não faremos essa configuração neste curso.
- `nosuid`: não permitir que `bits setuid` ou `setgid` tenham efeito no sistema de arquivos. Antes de colocar em prática, é recomendável escanear o sistema de arquivos por binários desse tipo, como faremos a seguir.
- `nodew`: não interpretar dispositivos especiais de bloco ou caractere nesse sistema de arquivos. Em geral, apenas o diretório `/dev` conterá arquivos dessa natureza.
- `noexec`: não permitir execução direta de quaisquer binários no sistema de arquivos. Não é viável habilitar essa opção para o diretório `/usr`, por motivos óbvios, mas pode ser uma boa opção para o `/tmp`, por exemplo — muitos *exploits* simples de escalada de privilégio tentam escrever e executar binários a partir do `/tmp`, e esta proteção pode dificultar sua ação. Contudo, alguns *scripts* de instalação de pacotes do Debian tentam executar binários diretamente do `/tmp`, e habilitá-lo com `noexec` pode quebrar a instalação desses pacotes — assim, não iremos utilizar essa configuração neste curso.

Antes de prosseguir com a customização das opções de montagem, vamos verificar quais binários possuem o *bit suid* ativo no sistema:

```
# find / -perm -4000 -exec ls {} \; 2> /dev/null
/bin/mount
/bin/ping
/bin/umount
/bin/su
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmccrypt-get-device
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/gpasswd
```

Note que temos executáveis nos diretórios */bin* e */usr/bin* — assim, não é factível habilitar a opção *nosuid* no diretório */usr*, neste momento.

De posse do conhecimento acima, vamos editar as opções de montagem dos volumes lógicos no arquivo */etc/fstab*:

```
# nano /etc/fstab
(...)
```

```
# tail -n3 /etc/fstab
/dev/mapper/vg--base-lv--tmp /tmp ext4 defaults,nosuid,nodev 0 2
/dev/mapper/vg--base-lv--var /var ext4 defaults,nosuid,nodev 0 2
/dev/mapper/vg--base-lv--usr /usr ext4 defaults,nodev 0 2
```

6. O último passo é reiniciar o sistema e verificar se nossas configurações surtiram efeito. Antes disso, vamos registrar o tamanho ocupado por cada um dos diretórios (*tmp*, *var* e *usr*), e comparar com os tamanhos ocupados nos LVs após o *reboot*.

```
# du -sm /{tmp,var,usr}
1      /tmp
181    /var
463    /usr
```

Perfeito. Reinicie a máquina:

```
# reboot
```

Após o *reboot*, logue como o usuário *root* e verifique que os volumes lógicos estão montados

corretamente:

```
# mount | grep '/tmp\|/var\|/usr'
/dev/mapper/vg--base-lv--tmp on /tmp type ext4
(rw,nosuid,nodev,relatime,data=ordered)
/dev/mapper/vg--base-lv--var on /var type ext4
(rw,nosuid,nodev,relatime,data=ordered)
/dev/mapper/vg--base-lv--usr on /usr type ext4 (rw,nodev,relatime,data=ordered)
```

Verifique, ainda, que o espaço ocupado dentro desses volumes é bastante próximo do tamanho dos diretórios dentro da raiz, /:

```
# df -m | sed -n '1p; /\(tmp\|var\|usr\) /p'
Sist. Arq.                               Blocos de 1M Usado Disponível Uso% Montado em
/dev/mapper/vg--base-lv--tmp              488      1         452    1% /tmp
/dev/mapper/vg--base-lv--var             1480    186        1203   14% /var
/dev/mapper/vg--base-lv--usr             4059    486        3348   13% /usr
```

7. Faltou alguma coisa? Ah sim! Não apagamos o conteúdo dos diretórios `tmp`, `var` e `usr` de dentro da raiz, /. Note como o espaço ocupado ainda é bastante grande neste momento:

```
# df -m | sed -n '1p; /\$/p'
Sist. Arq.                               Blocos de 1M Usado Disponível Uso% Montado em
/dev/mapper/vg--base-lv--root            1409    900         421   69% /
```

Mas, como apagar esses diretórios? Não podemos simplesmente rodar um comando `rm -rf /usr`, pois estaríamos removendo os arquivos gravados dentro do volume lógico `/dev/mapper/vg--base-lv--usr`, e não dentro da raiz. O que fazer, então?

A opção `bind` do comando `mount` (8) permite remontar um sistema de arquivos em outro ponto da hierarquia de diretórios, tornando seu conteúdo acessível em ambos os lugares. Monte, usando a opção `bind`, a raiz do sistema dentro do diretório `/mnt`:

```
# mount -o bind / /mnt
```

O diretório raiz, /, agora está acessível também abaixo de `/mnt`, como podemos observar:

```
# ls /mnt/
bin  dev  home      initrd.img.old  lib64          media  opt   root  sbin  sys
usr  vmlinuz
boot etc  initrd.img  lib             lost+found     mnt    proc  run   srv   tmp
var  vmlinuz.old
```

O volume lógico `/dev/mapper/vg--base-lv--usr`, no entanto, está montado apenas abaixo do diretório `/usr`, e não abaixo de `/mnt/usr` — em outras palavras, a pasta `/mnt/usr` referencia

diretamente o conjunto de arquivos gravados dentro da raiz do sistema, os quais queremos apagar para liberar espaço.

Faça um teste — crie um arquivo dentro de `/usr` com o nome `teste`. Note que ele está acessível pelo caminho `/usr/teste`, mas não via `/mnt/usr/teste`:

```
# touch /usr/teste
```

```
# ls -ld /usr/teste
-rw-r--r-- 1 root root 0 out 18 15:48 /usr/teste
```

```
# ls -ld /mnt/usr/teste
ls: não foi possível acessar '/mnt/usr/teste': Arquivo ou diretório não encontrado
```

Perfeito! Apague o conteúdo dos diretórios `/mnt/tmp`, `/mnt/var` e `/mnt/usr`, que foram copiados para os volumes lógicos via `rsync` no passo (9) desta atividade e não são mais necessários. Não apague as pastas em si, pois elas são ponto de montagem desses LVs.

```
# for d in tmp var usr; do cd /mnt/${d} ; rm -rf ..?* .[!..]* *; done
```

Para referência, o comando acima irá entrar nas pastas `/mnt/tmp`, `/mnt/var` e `/mnt/usr`, e, em cada uma irá apagar todos os arquivos e diretórios:

- Não-ocultos (*)
- Ocultos, cujo primeiro caractere seja `.` e o segundo caractere seja qualquer *exceto* `.`
- Ocultos, iniciados por dois caracteres `..` e seguidos obrigatoriamente por algum outro caractere qualquer

Com efeito, a expressão regular acima irá apagar todo o conteúdo da pasta exceto os *symlinks* especiais `..` e `...`.

Verifique que o espaço ocupado dentro do diretório raiz, `/`, reduziu significativamente:

```
# df -m | sed -n '1p; /\$/p'
Sist. Arq.                               Blocos de 1M Usado Disponível Uso% Montado em
/dev/mapper/vg--base-lv--root           1409    258          1063    20% /
```

Reinicie a máquina virtual para verificar que suas alterações não causaram nenhum impacto à estabilidade do sistema.

5) Inserção de senha no *bootloader*

Um aspecto que não pode ser esquecido é o *bootloader*, que faz a carga inicial do kernel — se desprotegido, um atacante com acesso físico à máquina pode utilizá-lo para alterar a senha do usuário `root` e ter acesso irrestrito ao sistema, dentre outras possibilidades.

Como vimos durante a instalação do Debian, o *bootloader* em uso pela grande maioria das distribuições Linux atualmente é o GRUB (*G*rand *U*nified *B*ootloader). Vamos configurar uma senha de acesso ao GRUB para impedir que um atacante consiga ter acesso indevido ao sistema.

1. Usando o comando `grub-mkpasswd-pbkdf2`, vamos gerar um hash para a senha `rnpesr123`.

```
# echo -e 'rnpesr123\nrnpesr123' | grub-mkpasswd-pbkdf2 | awk '/grub.pbkdf/{print$NF}'
grub.pbkdf2.sha512.10000.E025151B0DA98A3153BADD61FCDC2A6037A0505699B7C414D046D83438
0AB53D20532441EDAFF9B1E330E8496D2C7799E6EFB43C399CC6567D0AFD8961F70109.50A159E523E8
89A805937F5BB65B4067149D0FDAB0536061015B4345647350A2E09D19580D77D51E58BFDA3432FE241
6AE61F90D7F84D1D834CFC979DCBA8F8D
```

2. Agora, vamos editar o arquivo `/etc/grub.d/40_custom` e inserir o superusuário `admin`, com senha idêntica ao hash gerado no passo anterior.

```
# echo 'set superusers="admin"' >> /etc/grub.d/40_custom
```

```
# ghash="$( echo -e 'rnpesr123\nrnpesr123' | grub-mkpasswd-pbkdf2 | awk
'/grub.pbkdf/{print$NF}' )" ; echo "password_pbkdf2 admin ${ghash}" >>
/etc/grub.d/40_custom ; unset ghash
```

```
# tail -n2 /etc/grub.d/40_custom
set superusers="admin"
password_pbkdf2 admin
grub.pbkdf2.sha512.10000.65E70B724A540A0AE79C2F6BB34AFC4397BB13952D20A9C209DD70A9F3
1FE462D301B733D8B1B308C67908A25B44AB09420CEB306EDAEEB15765905A7DEEB3BF.209A3080C89E
D4C542716B9BE14162E8338DF8E36B68F9E0146BDC8E572CF41585F6BF67C2573AFF2645F0D851A8E9D
5B6AA2E4608E4735E689FA84ECA815C14
```

3. Finalmente, vamos reconfigurar o GRUB com a nova combinação usuário/senha e reiniciar a máquina. Verifique se a configuração está funcionando.

```
# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Imagem Linux encontrada: /boot/vmlinuz-4.9.0-8-amd64
Imagem initrd encontrada: /boot/initrd.img-4.9.0-8-amd64
concluído
```

```
# reboot
```

Após o *boot* da máquina, o menu do GRUB nos apresenta a possibilidade de editar a configuração apertando a tecla **e**:

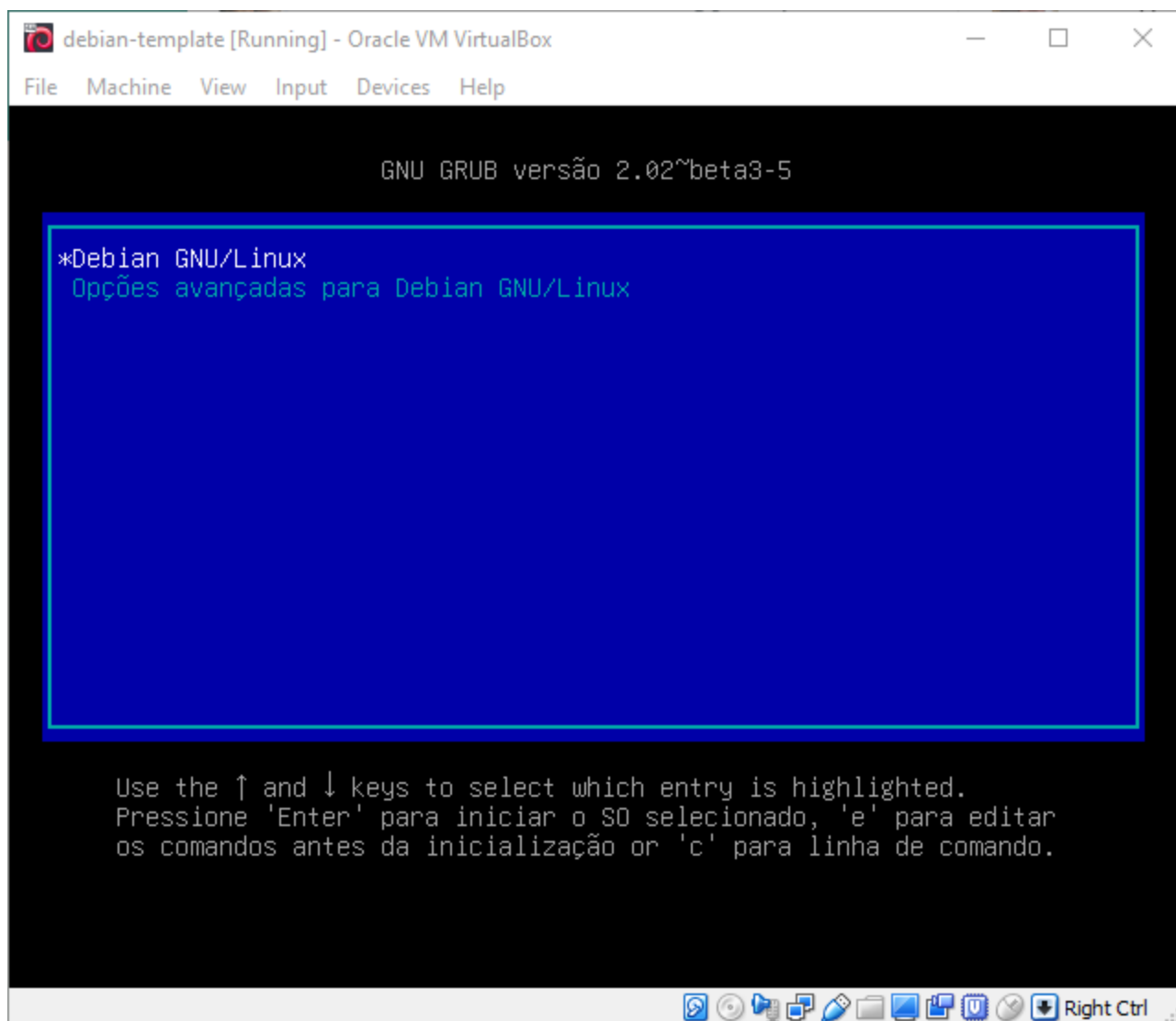


Figura 19. Edição de opções no GRUB

Apertando **e** sobre a primeira opção, imediatamente o sistema requisita a combinação usuário/senha configurada anteriormente:

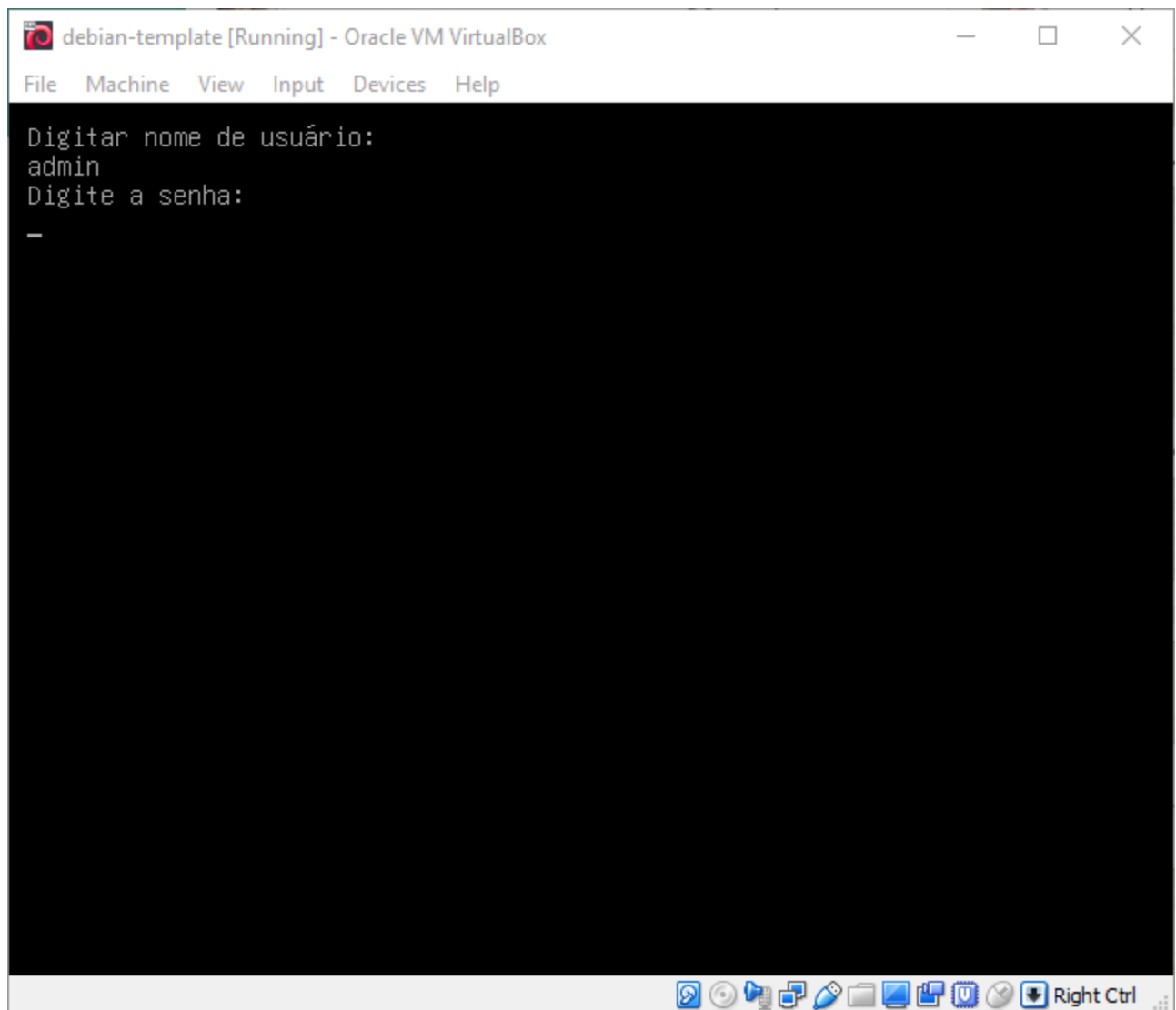


Figura 20. Inserção de usuário/senha no GRUB

Mediante a inserção da combinação correta, o menu de edição de opções de *boot* é mostrado, como se segue.

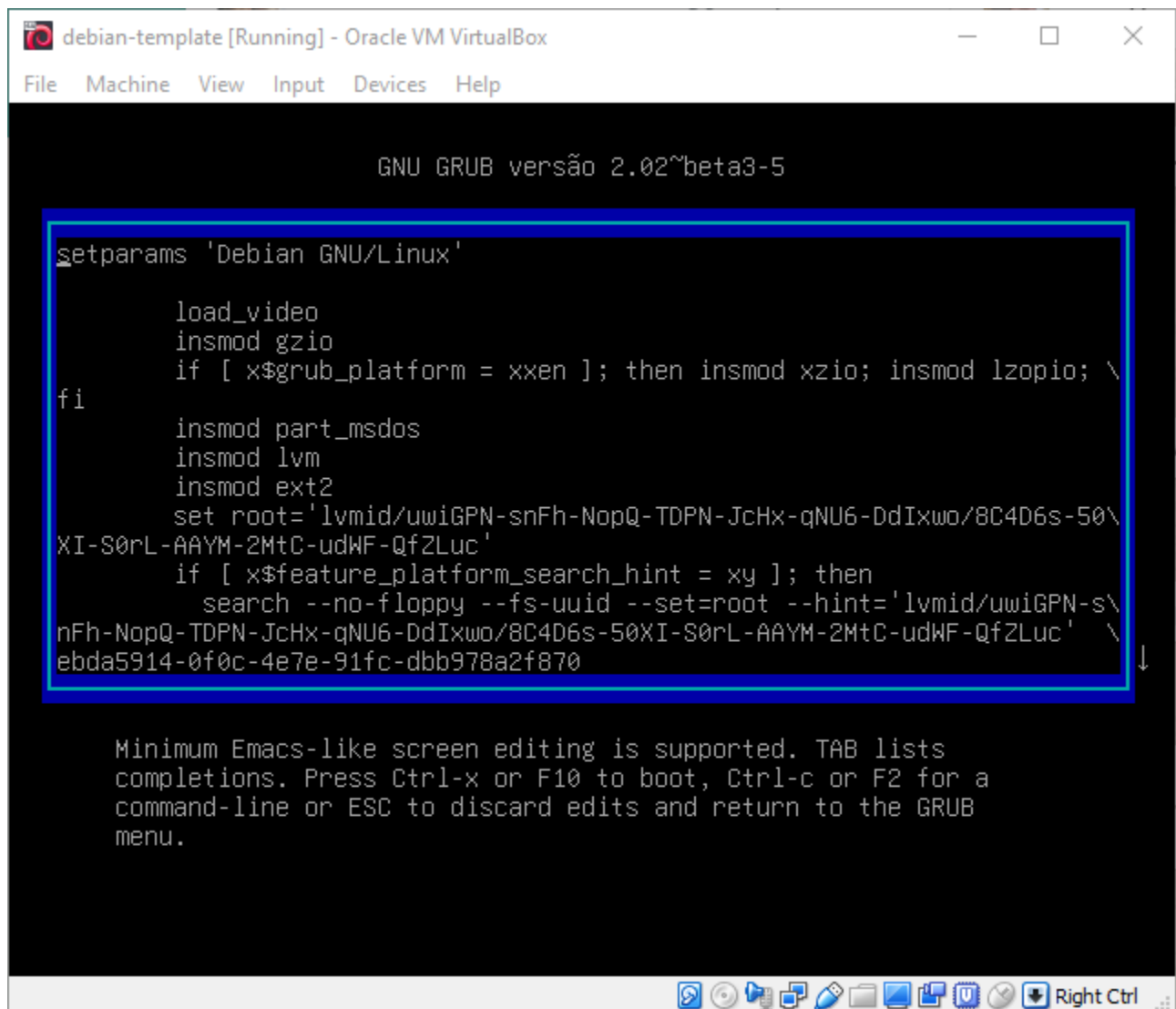


Figura 21. Edição de opções de boot no GRUB

Note ainda que, com esta configuração, o *boot* normal do sistema prossegue apenas se a combinação de usuário/senha correta for inserida no GRUB.

4. Vamos editar a configuração do GRUB para que ele solicite senha **apenas** em caso de edição de entradas do menu, e que o *boot* normal do sistema prossiga sem que haja necessidade de interação.

Para conseguir o efeito desejado, é necessário editar o arquivo `/etc/grub.d/10_linux`. Na função `linux_entry()`, iremos editar as duas linhas `echo "menuentry (...)`, inserindo a flag `--unrestricted` antes da variável `${CLASS}`.

Vamos ver um antes/depois para ficar mais claro. Veja como estão as linhas 132-134 do arquivo `/etc/grub.d/10_linux` antes da edição:

Listagem 1. /etc/grub.d/10_linux

```
132     echo "menuentry '$(echo "$title" | grub_quote)' ${CLASS}"  
\$menuentry_id_option 'gnulinux-$version-$type-$boot_device_id' {" | sed  
"s/^/$submenu_indentation/"  
133     else  
134     echo "menuentry '$(echo "$os" | grub_quote)' ${CLASS}"  
\$menuentry_id_option 'gnulinux-simple-$boot_device_id' {" | sed  
"s/^/$submenu_indentation/"
```

Após a edição, elas devem ficar assim:

Listagem 2. /etc/grub.d/10_linux

```
132     echo "menuentry '$(echo "$title" | grub_quote)' --unrestricted ${CLASS}"  
\$menuentry_id_option 'gnulinux-$version-$type-$boot_device_id' {" | sed  
"s/^/$submenu_indentation/"  
133     else  
134     echo "menuentry '$(echo "$os" | grub_quote)' --unrestricted ${CLASS}"  
\$menuentry_id_option 'gnulinux-simple-$boot_device_id' {" | sed  
"s/^/$submenu_indentation/"
```

Note a adição da flag `--unrestricted` antes de `${CLASS}` nas linhas 132 e 134.

Refaça a configuração do GRUB, reinicie a máquina e teste o funcionamento.

```
# update-grub  
Generating grub configuration file ...  
Imagem Linux encontrada: /boot/vmlinuz-4.9.0-8-amd64  
Imagem initrd encontrada: /boot/initrd.img-4.9.0-8-amd64  
concluído
```

```
# reboot
```

6) Clonando máquinas virtuais

Nosso *template*, para todos os efeitos, está preparado, atualizado e com configurações básicas de segurança aplicadas. Assim sendo, podemos utilizá-lo como base para a criação de novas máquinas virtuais durante este curso, começando a partir de agora.

Contudo, o Oracle VM Virtualbox não suporta o conceito de *templates* "a rigor", da mesma forma como interpretado em outras soluções de virtualização (como VMWare e Hyper-V). Para emular esse conceito de *templates*, sempre que necessário iremos clonar a máquina virtual `debian-template` e renomear a VM-clone, garantindo que o endereço físico (MAC) da placa de rede seja randomizado para evitar conflitos de IP.

1. Desligue a máquina **debian-template**:

```
# halt -p
```

2. Na janela principal do Virtualbox, clique com o botão direito na máquina **debian-template** e selecione a opção *Clone...*

Na tela seguinte, indique qual o nome da nova máquina virtual: para este exemplo, defina o nome **lvm-test**. Mantenha a caixa *Reinitialize the MAC address of all network cards* marcada.

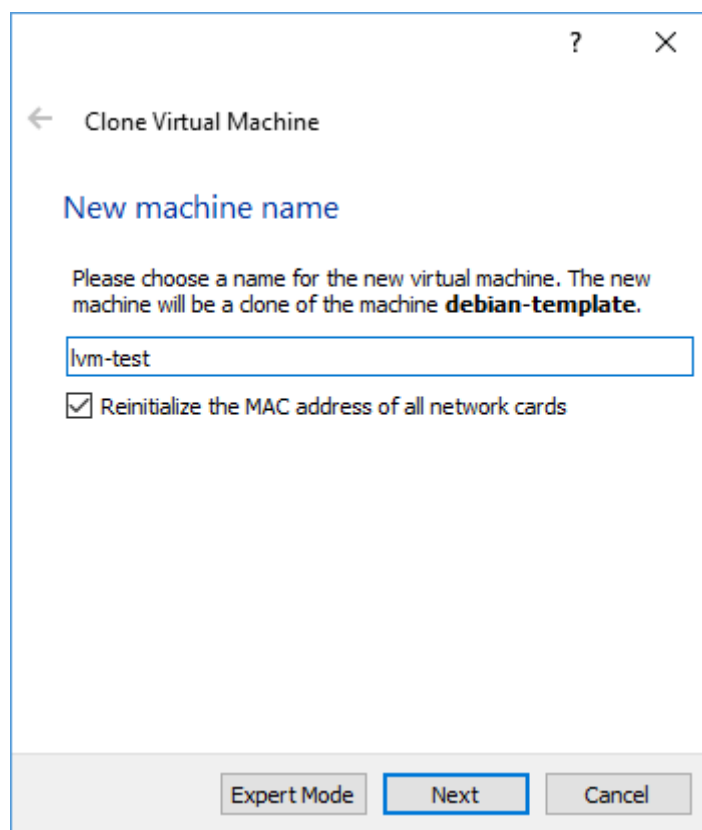


Figura 22. Clonagem de máquinas virtuais no Virtualbox

Clique em *Next*.

3. Na janela seguinte, você pode escolher se deseja fazer um clone completo (*Full clone*), ou um clone ligado (*Linked clone*). A diferença entre ambos é que no caso do clone completo é criada uma cópia separada do disco da VM original, ao passo que no clone ligado faz-se apenas um *snapshot* desse disco.

Mantenha *Full clone* marcado e clique em *Clone*.

7) Operações avançadas com LVM

Imagine que a máquina que acabamos de criar, **lvm-test**, será usada para dois propósitos: 1) atuar como um servidor de e-mail, armazenando as mensagens dos usuários da organização sob a pasta **/var/mail** e 2) armazenar dados sensíveis da organização, que devem ser acessados apenas por um número muito restrito de usuários, sob a pasta **/crypt**.

Vamos lidar com a situação (1), primeiramente.

1. Opere com a máquina recém-clonada, `lvm-test`. Na janela principal do Virtualbox, clique com o botão direito sobre a VM e depois em *Settings*.

Em *Storage > Controller: SATA*, clique no ícone com um pequeno HD com um sinal de +, com a legenda *Adds hard disk* para adicionar um novo disco à VM. Depois, clique em *Create new disk*.

Para o formato, escolha *VDI* e clique em *Next*. Mantenha a caixa *Dynamically allocated* marcada e clique em *Next*.

Para o nome do disco, digite `lvm-pv2` e mantenha o tamanho em 8 GB. Finalmente, clique em *Create*.

2. Repita o passo (1), adicionando um terceiro disco à VM. Desta vez, nomeie o disco como `lvm-crypt` e mantenha seu tamanho em 8 GB.

Ao final do processo, sua VM deverá estar com a seguinte configuração:

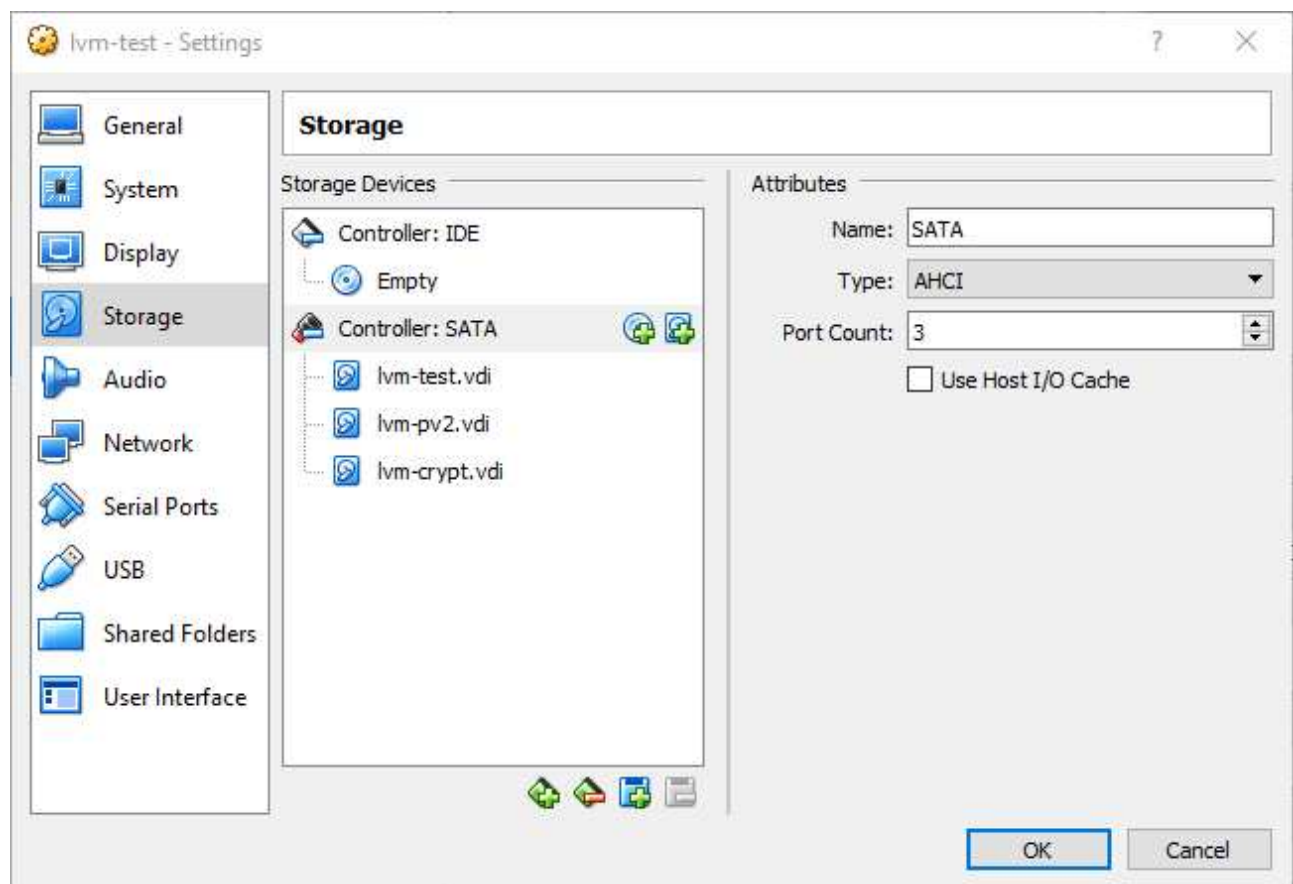


Figura 23. Discos adicionados à máquina `lvm-test`

Clique em *OK*, e ligue a máquina `lvm-test`.

3. Usando o script `/root/scripts/changehost.sh` que criamos anteriormente, renomeie a máquina:

```
# hostname  
debian-template
```

```
# bash ~/scripts/changehost.sh lvm-test
```

```
# hostname  
lvm-test
```

4. Prosseguindo com o tratamento do requisito (1), abordado no enunciado desta atividade, note que o espaço disponível no `/var` atualmente é bastante exíguo:

```
# df -h | sed -n '1p; /\var$/p'  
Sist. Arq.                               Tam. Usado Disp. Uso% Montado em  
/dev/mapper/vg--base-lv--var 1,5G 185M 1,2G 14% /var
```

Iremos utilizar o primeiro disco adicionado à VM, `lvm-pv2`, para aumentar o tamanho disponível para o diretório `/var`. Queremos, em ordem:

1. Identificar sob qual nome o disco `lvm-pv2` foi identificado pelo sistema Linux
 2. Adicionar a totalidade do disco `lvm-pv2` ao grupo de volumes `vg-base`
 3. Estender o volume lógico `lv-var` para usar o espaço extra disponível no VG `vg-base`
 4. Estender o sistema de arquivos do dispositivo `/dev/mapper/vg--base-lv--var` para utilizar o espaço extra disponível no LV `lv-var`
 5. Verificar o aumento do espaço disponível
5. Primeiramente, temos que detectar sob qual nome foi adicionado o disco virtual `lvm-pv2`. O comando `dmesg` nos mostra que três discos foram detectados durante o *boot*:

```
# dmesg | grep 'Attached SCSI disk'  
[ 1.924154] sd 2:0:0:0: [sdc] Attached SCSI disk  
[ 1.924182] sd 1:0:0:0: [sdb] Attached SCSI disk  
[ 1.936628] sd 0:0:0:0: [sda] Attached SCSI disk
```

Desses, sabemos que o dispositivo `/dev/sda` é o disco original que criamos durante a instalação do sistema, já que ele se encontra formatado e em uso pelo LVM:

```
# pvdisplay
--- Physical volume ---
PV Name           /dev/sda1
VG Name           vg-base
PV Size           8,00 GiB / not usable 2,00 MiB
Allocatable       yes (but full)
PE Size           4,00 MiB
Total PE          2047
Free PE           0
Allocated PE      2047
PV UUID           ZnHhMn-Y37D-6Psd-oHei-K1Qd-wHjY-6gqFZ3
```

Restam, então, os discos `/dev/sdb` e `/dev/sdc`. Em tese, poderíamos usar a diferença de tamanho entre os discos para intuir qual deles é o volume `lvm-pv2`, e qual é o `lvm-crypt`. Contudo, ambos possuem o mesmo tamanho, 8 GB. O que fazer, então?

Para determinar com precisão essa informação, na janela principal do Virtualbox acesse *File > Virtual Media Manager*. Na aba *Hard disks*, selecione o disco `lvm-pv2.vdi` e acesse a aba *Information*, como mostrado abaixo:

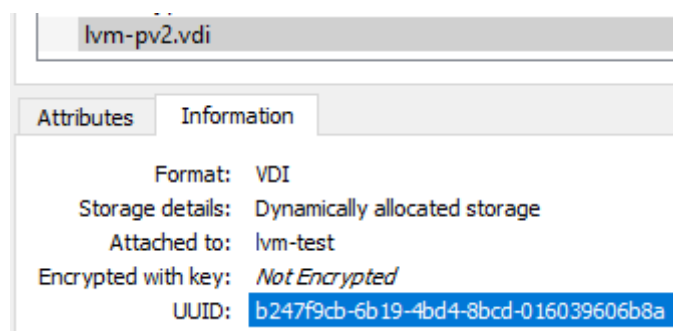


Figura 24. Identificando o UUID de um disco no Virtualbox

Note as *strings* inicial e final do campo *UUID* (Universally Unique Identifier), `b247f9cb` e `016039606b8a` respectivamente no exemplo acima.

De volta à máquina `lvm-test`, execute o comando:

```
# hdparm -i /dev/sdb | grep 'SerialNo' | cut -d',' -f3
SerialNo=VBb247f9cb-8a6b6039
```

Excluindo-se as letras `VB` do *serial* acima, note que a *string* `b247f9cb` é idêntica à que visualizamos no Virtualbox. De igual modo, a *string* `8a6b6039` é uma reversão dois-a-dois do final da *string* identificada no *Virtual Media Manager* do Virtualbox, anteriormente. Portanto, podemos afirmar com segurança que o disco `/dev/sdb` é o volume virtual `lvm-pv2`.

Faça o teste com o volume `lvm-crypt` e o disco `/dev/sdc`. As *strings* identificadoras do campo *UUID* são compatíveis?

- Identificado o disco `/dev/sdb` como nosso alvo, e como desejamos adicionar um disco inteiro ao VG `vg-base`, o primeiro passo é editar a tabela de partições do disco corretamente. Para tanto,

basta criar uma partição primária ocupando a totalidade do disco, e identificá-la como **Linux LVM**.

```
# fdisk /dev/sdb
```

Bem-vindo ao fdisk (util-linux 2.29.2).

As alterações permanecerão apenas na memória, até que você decida gravá-las.
Tenha cuidado antes de usar o comando de gravação.

Comando (m para ajuda): o

Criado um novo rótulo de disco DOS com o identificador de disco 0xe7d643f2.

Comando (m para ajuda): n

Tipo da partição

p primária (0 primárias, 0 estendidas, 4 livre)

e estendida (recipiente para partições lógicas)

Selecione (padrão p):

Usando resposta padrão p.

Número da partição (1-4, padrão 1):

Primeiro setor (2048-16777215, padrão 2048):

Último setor, +setores ou +tamanho{K,M,G,T,P} (2048-16777215, padrão 16777215):

Criada uma nova partição 1 do tipo "Linux" e de tamanho 8 GiB.

Comando (m para ajuda): t

Selecionou a partição 1

Tipo de partição (digite L para listar todos os tipos): 8e

O tipo da partição "Linux" foi alterado para "Linux LVM".

Comando (m para ajuda): w

A tabela de partição foi alterada.

Chamando ioctl() para reler tabela de partição.

Sincronizando discos.

Em ordem, executamos **fdisk /dev/sdb** para editar a tabela de partições do dispositivo e então:

- **o** para criar uma tabela de partições vazia
- **n** para criar uma nova partição
- **ENTER** para aceitar o tipo padrão de partição (primária)
- **ENTER** para aceitar o número padrão de partição (número **1**)
- **ENTER** para aceitar o primeiro setor disponível no disco (2048)

- **ENTER** para aceitar o último setor disponível no disco (16777215), maximizando o tamanho da partição
- **t** para alterar o identificador da partição
- **8e** para identificar a partição como **Linux LVM**
- **w** para gravar as alterações realizadas e sair do programa

Para visualizar o estado do disco, podemos usar **fdisk -l**:

```
# fdisk -l /dev/sdb
Disco /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 setores
Unidades: setor de 1 * 512 = 512 bytes
Tamanho de setor (lógico/físico): 512 bytes / 512 bytes
Tamanho E/S (mínimo/ótimo): 512 bytes / 512 bytes
Tipo de rótulo do disco: dos
Identificador do disco: 0xe7d643f2

Dispositivo Inicializar Início      Fim  Setores Tamanho Id Tipo
/dev/sdb1          2048 16777215 16775168      8G 8e Linux LVM
```

A seguir, usaremos o comando **pvcreate** para inicializar a partição e prepará-la para o LVM:

```
# pvcreate /dev/sdb1
Physical volume "/dev/sdb1" successfully created.
```

```
# pvdisplay /dev/sdb1
"/dev/sdb1" is a new physical volume of "8,00 GiB"
--- NEW Physical volume ---
PV Name           /dev/sdb1
VG Name
PV Size           8,00 GiB
Allocatable       NO
PE Size           0
Total PE          0
Free PE           0
Allocated PE      0
PV UUID           FdZmMJ-ZFdQ-vi0z-lc5r-9Zy2-zfbR-1uJbuj
```

O comando **pvscan** é uma opção interessante para mostrar o estado dos volumes físicos do sistema:

```
# pvscan
PV /dev/sda1   VG vg-base      lvm2 [8,00 GiB / 0    free]
PV /dev/sdb1   VG vg-base      lvm2 [8,00 GiB]
Total: 2 [16,00 GiB] / in use: 1 [8,00 GiB] / in no VG: 1 [8,00 GiB]
```


7. Agora sim, podemos adicionar o novo volume físico `/dev/sdb1` ao grupo de volumes `vg-base`. Note seu espaço atual:

```
# vgdisplay | grep 'VG Name\|VG Size'
VG Name          vg-base
VG Size          8,00 GiB
```

Expanda o VG:

```
# vgextend vg-base /dev/sdb1
Volume group "vg-base" successfully extended
```

Verificando o estado do VG `vg-base`, note que seu tamanho saltou para 16 GB, como esperado:

```
# vgdisplay | grep 'VG Name\|VG Size'
VG Name          vg-base
VG Size          15,99 GiB
```

8. A seguir, iremos estender o volume lógico `lv-var` para utilizar a totalidade do novo espaço adicionado ao VG. Note seu espaço atual:

```
# lvsdisplay /dev/vg-base/lv-var | grep Size
LV Size          1,50 GiB
```

Façamos o procedimento de expansão:

```
# lvextend -l +100%FREE /dev/vg-base/lv-var
Size of logical volume vg-base/lv-var changed from 1,50 GiB (384 extents) to 9,50 GiB (2431 extents).
Logical volume vg-base/lv-var successfully resized.
```

E em seguida, chequemos o novo tamanho disponível:

```
# lvsdisplay /dev/vg-base/lv-var | grep Size
LV Size          9,50 GiB
```

9. O passo final é redimensionar o sistema de arquivos. Como queremos simplesmente ocupar a totalidade do volume lógico, e kernels mais modernos do Linux suportam *on-line resizing* (i.e. redimensionamento sem necessidade de desmontar o sistema de arquivos), basta executar:

```
# resize2fs /dev/mapper/vg--base-lv--var
resize2fs 1.43.4 (31-Jan-2017)
Filesystem at /dev/mapper/vg--base-lv--var is mounted on /var; on-line resizing
required
old_desc_blocks = 1, new_desc_blocks = 2
The filesystem on /dev/mapper/vg--base-lv--var is now 2489344 (4k) blocks long.
```

Note, imediatamente, que o espaço disponível para o **/var** aumenta significativamente:

```
# df -h | sed -n '1p; /\var$/p'
Sist. Arq.                Tam. Usado Disp. Uso% Montado em
/dev/mapper/vg--base-lv--var 9,4G 188M 8,8G 3% /var
```

E assim, concluímos nossa seção sobre o LVM. É um sistema muito poderoso, que oferece grande flexibilidade na gestão de armazenamento no Linux. Imagine: qual teria sido a dificuldade em estender o espaço disponível para o **/var** em um sistema sem o uso do LVM?

Outros aspectos avançados do LVM, como gestão de volumes *striped* (concatenados) e *mirrored* (espelhados), provisionamento dinâmico de espaço e *snapshots* não foram trabalhados aqui. Convidamos o aluno a investigar essas capacidades, e testar suas funcionalidades em ambiente de laboratório. A documentação do Red Hat Enterprise Linux sobre o LVM é um excelente recurso para começar: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/logical_volume_manager_administration/lv_overview

8) Criptografia de partições

Retomando o requisito (2) apresentado na atividade (7), foi dito que se desejava: armazenar dados sensíveis da organização, que devem ser acessados apenas por um número muito restrito de usuários, sob a pasta **/crypt**.

Vamos, agora, implementar esse requisito na máquina **lvm-test**.

1. A solução que iremos implantar para garantir o requisito é criptografar a partição. Para tanto, precisaremos instalar os seguintes pacotes:

```
# apt-get install cryptsetup -y
```

2. Prepare o disco **/dev/sdc** da mesma forma que fizemos no passo (6) da atividade anterior. Ao final do processo, a saída do comando **pvdisk /dev/sdc1** deve ser como mostrada abaixo:

```
# pvdiskdisplay /dev/sdc1
"/dev/sdc1" is a new physical volume of "8,00 GiB"
--- NEW Physical volume ---
PV Name                /dev/sdc1
VG Name
PV Size                8,00 GiB
Allocatable           NO
PE Size               0
Total PE              0
Free PE               0
Allocated PE          0
PV UUID               JKRA1t-0sgK-d0np-3N3J-JRyE-YDbG-0cdZee
```

3. Vamos criar um novo VG para armazenar dados sensíveis. Execute:

```
# vgcreate vg-crypt /dev/sdc1
Volume group "vg-crypt" successfully created
```

4. O próximo passo é criar um volume lógico:

```
# lvcreate -l +100%FREE -n lv-crypt vg-crypt
Logical volume "lv-crypt" created.
```

Verifique que o LV foi criado corretamente:

```
# lvdiskdisplay /dev/vg-crypt/lv-crypt
--- Logical volume ---
LV Path                /dev/vg-crypt/lv-crypt
LV Name                lv-crypt
VG Name                vg-crypt
LV UUID                mMOiAc-Q7Yo-hgjN-rb2S-mzfr-youw-PnHGRC
LV Write Access        read/write
LV Creation host, time lvm-test, 2018-10-19 11:12:42 -0300
LV Status              available
# open                 0
LV Size                8,00 GiB
Current LE             2047
Segments               1
Allocation              inherit
Read ahead sectors     auto
- currently set to    256
Block device           254:6
```

5. Agora, vamos criptografar esse LV — o comando **cryptsetup** pode ser usado para este fim. Iremos criptografar o disco no formato LUKS (*Linux Unified Key Setup*), o padrão para criptografia de armazenamento no Linux. Diferentemente de outras soluções de criptografia, o

LUKS armazena todas as informações de configuração no cabeçalho da partição, permitindo ao usuário migrar seus dados de forma fácil entre diferentes distribuições Linux ou mesmo outros sistemas operacionais.

A cifra padrão de criptografia do LUKS pode ser visualizada com o comando:

```
# cryptsetup --help | grep 'LUKS1:'
LUKS1: aes-xts-plain64, Chave: 256 bits, Hash de cabeçalho LUKS: sha256,
RNG: /dev/urandom
```

Para criptografar a partição, execute:

```
# cryptsetup luksFormat /dev/mapper/vg--crypt-lv--crypt

WARNING!
=====
Isto vai sobrescrever dados em /dev/mapper/vg--crypt-lv--crypt permanentemente.

Are you sure? (Type uppercase yes): YES
Digite a senha:
Verificar senha:
```

Digite *YES* em letras maiúsculas para confirmar sobrescrita dos dados. Escolha uma senha forte para proteger os dados. Neste laboratório, recomendamos a senha **rnpesr123**, por conveniência.

6. Como saber que uma partição está criptografada? E, de fato, como saber o tipo de uma partição qualquer? O comando **lsblk** é especialmente útil nesse cenário:

```
# lsblk --fs
NAME                                FSTYPE    LABEL  UUID
MOUNTPOINT
sda
├─sda1                             LVM2_member  n3dXDL-gCma-P258-pl31-0w9A-Spcp-2mjsCG
│   └─vg--base-lv--boot            ext2        ebda5914-0f0c-4e7e-91fc-dbb978a2f870
/boot
│   └─vg--base-lv--swap            swap        fcf0e5dd-d744-4acc-aff4-65aa2219fde2
[SWAP]
│   └─vg--base-lv--root            ext4        bfebe607-ef84-4ac8-8ce0-54dded08ae9e
/
│   └─vg--base-lv--tmp             ext4        3db27de5-69ae-4db7-baba-2de5a4576942
/tmp
│   └─vg--base-lv--var             ext4        2a13673c-4266-4683-a548-d66b0c1078b1
/var
│   └─vg--base-lv--usr             ext4        5621572f-4786-4cb6-8826-2dae623b3e5d
/usr
sdb
├─sdb1                             LVM2_member  FdZmMJ-ZFdQ-vi0z-lc5r-9Zy2-zfbR-1uJbuJ
│   └─vg--base-lv--var            ext4        2a13673c-4266-4683-a548-d66b0c1078b1
/var
sdc
├─sdc1                             LVM2_member  JKRA1t-0sgK-d0np-3N3J-JRyE-YDbG-0cdZee
│   └─vg--crypt-lv--crypt         crypto_LUKS 2f10b9c6-aab3-4038-b74c-6bab83e658fe
sr0
```

Note que o LV que acabamos de criar e criptografar, **lv-crypt**, possui o tipo de sistema de arquivos **crypto_LUKS**. Note, ainda, que ele não está montado:

```
# mount | grep 'lv--crypt'
```

7. O próximo passo é formatar a partição. Contudo, para acessar partições LUKS, temos primeiro que mapeá-la sob um nome — execute:

```
# cryptsetup luksOpen /dev/mapper/vg--crypt-lv--crypt seg10-crypt
Digite a senha para /dev/mapper/vg--crypt-lv--crypt:
```

Para verificar os nomes mapeados, novamente podemos usar o **lsblk**:

```
# lsblk /dev/sdc
NAME                                MAJ:MIN RM SIZE RO TYPE  MOUNTPOINT
sdc                                8:32  0  8G  0 disk
├─sdc1                            8:33  0  8G  0 part
│   └─vg--crypt-lv--crypt         254:6   0  8G  0 lvm
│       └─seg10-crypt             254:7   0  8G  0 crypt
```

Com o dispositivo `/dev/mapper/vg—crypt-lv—crypt` mapeado para `/dev/mapper/seg10-crypt`, podemos, agora sim, formatar a partição:

```
# mkfs.ext4 /dev/mapper/seg10-crypt
mke2fs 1.43.4 (31-Jan-2017)
Creating filesystem with 2095616 4k blocks and 524288 inodes
Filesystem UUID: 5740608c-8609-4d95-ae9b-73a016765610
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

E, finalmente, montá-la:

```
# mount /dev/mapper/seg10-crypt /mnt/
```

```
# mount | grep '/mnt'
/dev/mapper/seg10-crypt on /mnt type ext4 (rw,relatime,data=ordered)
```

8. Após gravar dados na partição criptografada, temos que fazer o caminho oposto. Primeiro, desmontá-la:

```
# umount /mnt
```

E, em seguida, remover o mapeamento por nome da partição LUKS:

```
# cryptsetup luksClose seg10-crypt
```

9. Concluídas nossas atividades de exemplo com o LVM e criptografia de partições, desligue a máquina `lvm-test`:

```
# halt -p
```

Como não utilizaremos mais esta máquina no decorrer do curso, vamos removê-la. Na janela principal do Virtualbox, clique com o botão direito sobre a VM `lvm-test` e selecione *Remove*. Na nova janela, clique em *Delete all files* para remover todos os discos associados à máquina.

Sessão 2: Firewall

1) Topologia desta sessão

A figura abaixo mostra a topologia de rede que será utilizada nesta sessão, com as máquinas relevantes em destaque.

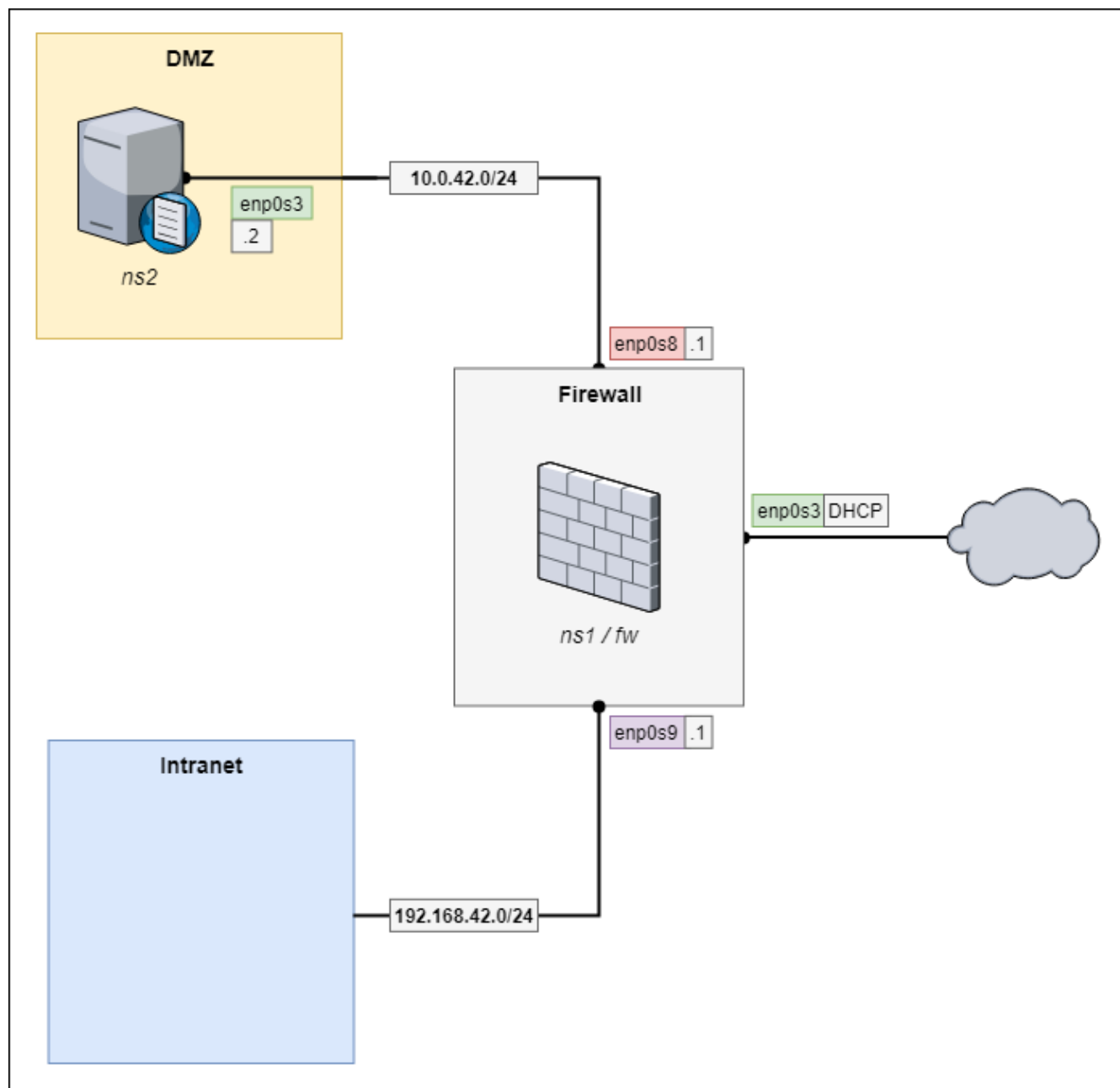


Figura 25. Topologia de rede desta sessão

Teremos apenas duas máquinas, por enquanto:

- **ns1**, atuando como firewall de rede e DNS primário, e
- **ns2**, atuando como DNS secundário e localizada na DMZ (*Demilitarized Zone*, ou zona desmilitarizada).

1. Antes de começarmos, precisamos configurar corretamente as redes virtuais no Virtualbox. Acesse o menu *File > Host Network Manager* e crie as seguintes redes:

Tabela 1. Redes host-only no Virtualbox

Rede	Endereço IPv4	Máscara de rede	Servidor DHCP
Virtualbox Host-Only Ethernet Adapter #2	10.0.42.254	255.255.255.0	Desabilitado
Virtualbox Host-Only Ethernet Adapter #3	192.168.42.254	255.255.255.0	Desabilitado

Visualmente, sua janela deve ficar parecida com o seguinte:

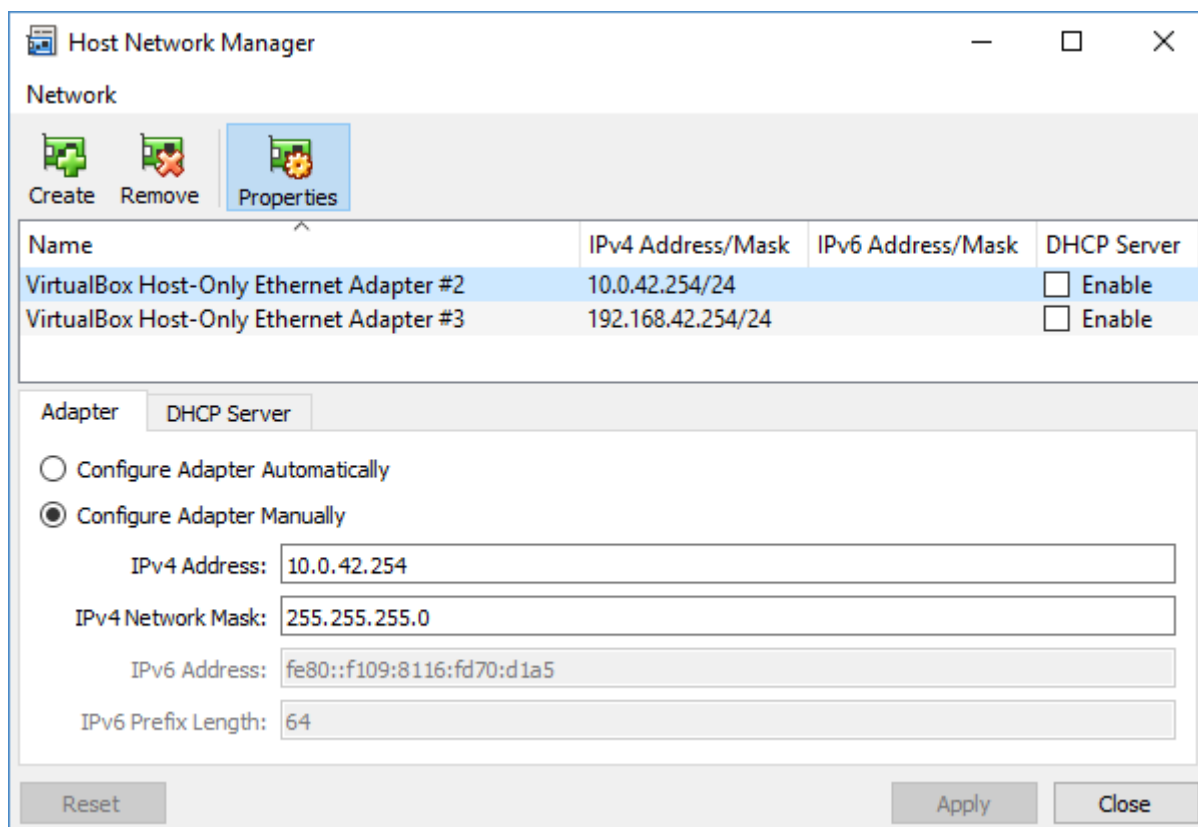


Figura 26. Redes host-only no Virtualbox

É possível que os números específicos das redes (#2 para a DMZ e #3 para a Intranet, na imagem acima) não fiquem exatamente iguais aos exemplificados. Nesse caso, faça uma anotação indicando qual número de rede *host-only* corresponde a cada uma das redes configuradas. Verifique, ainda, que o servidor DHCP interno do Virtualbox está desabilitado em ambas as redes.

2. As configurações de rede realizadas internamente em cada máquina virtual foram apresentados de forma sucinta na topologia desta sessão. Iremos detalhar as configurações logo abaixo:

Tabela 2. Configurações de rede de cada VM

VM Nome	Interface	Modo	Endereço	Gateway
ns1	enp0s3	DHCP	Automático	Automático
	enp0s8	Estático	10.0.42.1/24	n/a
	enp0s9	Estático	192.168.42.1/24	n/a
ns2	enp0s3	Estático	10.0.42.2/24	10.0.42.1

A partir do Debian 9, a nomenclatura padrão de interfaces de rede foi alterada. Ao invés de denotarmos as interfaces como `eth0`, `eth1` ou `eth2`, o `systemd/udev` utiliza, a partir da versão v197, um método de nomenclatura de interfaces usando `biosdevnames`, como documentado oficialmente em <https://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames/>. Com efeito, esse novo sistema suporta cinco meios de nomeação de interfaces de rede:

1. Nomes incorporando números de índice providos pelo firmware/BIOS de dispositivos *on-board* (p.ex.: `eno1`)
2. Nomes incorporando números de índice providos pelo firmware/BIOS de encaixes *hotplug* PCI Express (p.ex.: `ens1`)
3. Nomes incorporando localização física/geográfica do conector do hardware (p.ex.: `enp2s0`)
4. Nomes incorporando o endereço MAC da interface (p.ex.: `enx78e7d1ea46da`)
5. Nomes clássicos, usando nomenclatura não-previsível nativa do kernel (p.ex.: `eth0`)

Todas as VMs utilizadas neste curso serão derivadas da máquina `debian-template`, configurada com o Debian 9. A equivalência da nova nomenclatura de interfaces de rede, se comparada com a antiga, ficaria assim:

Tabela 3. Nomenclatura de interfaces de máquinas Debian 9

Interface antiga	Interface nova
eth0	enp0s3
eth1	enp0s8
eth2	enp0s9

Observe, por exemplo, como é feita a detecção de interfaces durante o *boot* da máquina `ns1`, que criaremos a seguir:

```
# dmesg | grep 'renamed from'
[ 1.658908] e1000 0000:00:09.0 enp0s9: renamed from eth2
[ 1.659807] e1000 0000:00:08.0 enp0s8: renamed from eth1
[ 1.660711] e1000 0000:00:03.0 enp0s3: renamed from eth0
```

2) Criação da VM de firewall e DNS primário

Iremos agora criar a primeira máquina virtual efetiva de nosso *datacenter* simulado, a máquina **ns1**. Essa máquina atuará como um firewall de borda e DNS primário da rede, como configuraremos a seguir. Por se tratar de um firewall, é necessário que ela possua ao menos duas (ou, em nosso caso específico, três) interfaces de rede interconectando redes distintas.

1. Clone a máquina **debian-template** seguindo os mesmos passos da atividade (6) da sessão 1. Para o nome da máquina, escolha **ns1**.
2. Após a clonagem, na janela principal do Virtualbox, clique com o botão direito sobre a VM **ns1** e depois em *Settings*.

Em *Network > Adapter 1 > Attached to*, mantenha escolhida a opção *Bridged Adapter*, já que esta será a interface de conexão externa da máquina.

Em *Adapter 2*, marque a caixa *Enable Network Adapter* e em *Attached to* selecione *Host-only Adapter*. O nome da rede *host-only* deve ser o mesmo alocado para a **DMZ**, como indicado na atividade (1) desta sessão. Seguindo o exemplo mostrado na figura da atividade, a rede escolhida seria portanto a **Virtualbox Host-Only Ethernet Adapter #2**.

Em *Adapter 3*, marque a caixa *Enable Network Adapter* e em *Attached to* selecione *Host-only Adapter*. O nome da rede *host-only* deve ser o mesmo alocado para a **Intranet**, como indicado na atividade (1) desta sessão. Seguindo o exemplo da figura, escolheríamos então **Virtualbox Host-Only Ethernet Adapter #3**.

Clique em *OK*, e ligue a máquina **ns1**.

3. Após o *boot*, faça login como o usuário **root**. Primeiro, vamos configurar a rede: edite o arquivo **/etc/network/interfaces** como se segue:

```
# nano /etc/network/interfaces
(...)
```

```
# cat /etc/network/interfaces
source /etc/network/interfaces.d/*

auto lo enp0s3 enp0s8 enp0s9

iface lo inet loopback

iface enp0s3 inet dhcp

iface enp0s8 inet static
address 10.0.42.1/24

iface enp0s9 inet static
address 192.168.42.1/24
```

Para garantir que nenhum endereço IP antigo, primário, se mantenha alocado às interfaces, execute o comando **flush** e em seguida reinicie a rede do sistema:

```
# ip addr flush label 'enp0s*' ; systemctl restart networking
```

Verifique que as interfaces estão com os endereços corretamente alocados:

```
# ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}'
192.168.29.104/24 enp0s3
10.0.42.1/24 enp0s8
192.168.42.1/24 enp0s9
```

4. Usando o script `/root/scripts/changehost.sh` que criamos anteriormente, renomeie a máquina:

```
# hostname
debian-template
```

```
# bash ~/scripts/changehost.sh ns1
```

```
# hostname
ns1
```

3) Configuração inicial do firewall

Para garantir a segurança da rede iremos configurar o firewall de forma extremamente restritiva, como se segue:

- Tráfego oriundo do firewall (*chain* OUTPUT) será permitido.
- Todo o tráfego na interface *loopback* será permitido.
- Serão permitidos pacotes destinados ao firewall (*chain* INPUT) ou passando pelo firewall (*chain* FORWARD) cujo estado seja relacionado ou estabelecido.
- Serão permitidos pacotes ICMP oriundos das redes DMZ e Intranet com destino ao firewall *FWGW1-G*.
- Será permitida gerência via **ssh** do firewall a partir de máquinas da Intranet.
- Será autorizado o tráfego na Internet das máquinas da DMZ e Intranet **exclusivamente** nas portas TCP/80 e TCP/443.
- Todos os demais acessos serão bloqueados.

À medida que novas regras forem necessárias, nas sessões seguintes, iremos criar regras de exceção pontualmente durante a execução das atividades, explicando os motivos das liberações. Vamos,

ponto a ponto, realizar as configurações explicitadas acima:

1. Primeiro, como em qualquer firewall de rede, devemos habilitar o repasse de pacotes entre interfaces. Para isso, edite o arquivo `/etc/sysctl.conf` e descomente a linha com a diretiva `net.ipv4.ip_forward`, como se segue:

```
# sed -i '/net.ipv4.ip_forward/s/^#//' /etc/sysctl.conf
```

Processe as alterações no arquivo com:

```
# sysctl -p
net.ipv4.ip_forward = 1
```

Observe que esse arquivo é lido durante o *boot* do sistema, o que garante que nossa configuração perdurará mesmo após reiniciarmos a máquina.

2. Agora, vamos retomar as diretivas informadas no início desta atividade: para a requisição (a) não precisamos fazer nada, já que a política da *chain* OUTPUT encontra-se em ACCEPT:

```
# iptables -L OUTPUT
Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

3. A diretiva (b) pode ser atendida com a regra que se segue:

```
# iptables -A INPUT -i lo -j ACCEPT
```

Já tratamos do caso da *chain* OUTPUT, e não faz sentido falarmos em tráfego na interface *loopback* na *chain* FORWARD.

4. Para a diretiva (c) devemos usar uma regra de estados nas *chains* INPUT e FORWARD, da seguinte forma:

```
# iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
```

```
# iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
```

5. Como a diretiva (d) diz especificamente de pacotes "com destino ao firewall", fica claro que devemos adicionar uma regra à *chain* INPUT. Note ainda que a diretiva é precisa em especificar que **apenas** o tráfego ICMP das redes DMZ e Intranet deve ser autorizado, e nenhum outro. Finalmente, não é especificado qual tipo de pacote ICMP será aceito, o que nos permite deduzir que todos serão aceitos.

Para inserir uma regra que inclua ambas as redes de origem, basta separá-las com vírgula, como se segue:

```
# iptables -A INPUT -s 10.0.42.0/24,192.168.42.0/24 -p icmp -m icmp --icmp-type any  
-j ACCEPT
```

6. A diretiva (e) é clara ao especificar que a gerência será via **ssh** (portanto, na porta 22 do protocolo TCP), a ser feita no firewall (ou seja, *chain* INPUT), e apenas a partir de máquinas da Intranet. Podemos atender a esse requisito com a seguinte regra:

```
# iptables -A INPUT -s 192.168.42.0/24 -p tcp -m tcp --dport 22 -j ACCEPT
```

7. A diretiva (f) diz que o tráfego na Internet das máquinas da DMZ e Intranet deve ser autorizado apenas nas portas TCP/80 e TCP/443. O requisito de IP de origem é bastante claro, mas o de destino não — de fato, máquinas na Internet podem ter, a princípio, qualquer endereço IP. Faz sentido, então, indicarmos a interface de saída dos pacotes, **enp0s3**.

Outro aspecto a ser observado é que os pacotes desta vez não se destinam ao firewall, mas sim passam por ele para atingir máquinas na Internet, indicando que a regra deve ser inserida na *chain* FORWARD.

Podemos ainda usar o módulo **multiport** para evitar a digitação de duas regras similares. Então, execute:

```
# iptables -A FORWARD -s 10.0.42.0/24,192.168.42.0/24 -o enp0s3 -p tcp -m multiport  
--dports 80,443 -j ACCEPT
```

Há ainda que se considerar a necessidade de realizar a tradução dos endereços de saída, pois não é possível que as máquinas da DMZ/Intranet naveguem com seus IPs em faixas privadas. Temos que criar uma regra de SNAT para permitir a navegação — levando em conta que o endereço da interface **enp0s3** é dinâmico, faz sentido usar o alvo MASQUERADE, nesse caso.

Um adendo final: podemos fazer uma regra tão restritiva quanto a que fizemos na *chain* FORWARD acima, especificando também o protocolo e porta em que o SNAT será realizado. Tenha em mente apenas que, em caso de adição de exceções futuras, será necessário adicionar o protocolo/porta de exceção em **ambas** as *chains*, **filter/FORWARD** e **nat/POSTROUTING**.

A regra fica assim:

```
# iptables -t nat -A POSTROUTING -s 10.0.42.0/24,192.168.42.0/24 -o enp0s3 -p tcp  
-m multiport --dports 80,443 -j MASQUERADE
```

8. Atender a diretiva (g) final é bastante fácil: basta alterar a política das *chains* INPUT e FORWARD para DROP:

```
# iptables -P INPUT DROP
```

```
# iptables -P FORWARD DROP
```

9. Verifique a configuração final do firewall, comparando com os requisitos iniciais. Consulte primeiro a tabela *filter*:

```
# iptables -L -vn
Chain INPUT (policy DROP 189 packets, 52988 bytes)
  pkts bytes target    prot opt in     out     source         destination
    0     0 ACCEPT    all  --  lo      *        0.0.0.0/0      0.0.0.0/0
  982 67024 ACCEPT    all  --  *       *        0.0.0.0/0      0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT    icmp  --  *       *       10.0.42.0/24    0.0.0.0/0
icmp type 255
    0     0 ACCEPT    icmp  --  *       *       192.168.42.0/24 0.0.0.0/0
icmp type 255
    0     0 ACCEPT    tcp   --  *       *       192.168.42.0/24 0.0.0.0/0
tcp dpt:22

Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source         destination
    0     0 ACCEPT    all  --  *       *        0.0.0.0/0      0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT    tcp   --  *       enp0s3  10.0.42.0/24    0.0.0.0/0
multiport dports 80,443
    0     0 ACCEPT    tcp   --  *       enp0s3  192.168.42.0/24 0.0.0.0/0
multiport dports 80,443

Chain OUTPUT (policy ACCEPT 16 packets, 1264 bytes)
  pkts bytes target    prot opt in     out     source         destination
```

E, depois, a tabela *nat*:

```
# iptables -L -vn -t nat
Chain PREROUTING (policy ACCEPT 10 packets, 1485 bytes)
  pkts bytes target    prot opt in     out     source        destination

Chain INPUT (policy ACCEPT 6 packets, 1012 bytes)
  pkts bytes target    prot opt in     out     source        destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source        destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source        destination
    0    0 MASQUERADE tcp  --  *      enp0s3  10.0.42.0/24   0.0.0.0/0
multiport dports 80,443
    0    0 MASQUERADE tcp  --  *      enp0s3  192.168.42.0/24 0.0.0.0/0
multiport dports 80,443
```

10. As configurações realizadas até aqui estão todas em memória — em caso de *reboot* da máquina *ns1*, elas serão perdidas. Para gravar as regras e integrá-las ao sistema de *init* do SO, o pacote *iptables-persistent* é uma excelente opção. Instale-o com:

```
# apt-get install iptables-persistent
```

Na instalação do pacote, quando perguntado, responda:

Tabela 4. Configurações do *iptables-persistent*

Pergunta	Resposta
Salvar as regras IPv4 atuais?	Sim
Salvar as regras IPv6 atuais?	Sim

As regras IPv4 e IPv6 serão gravadas nos arquivos */etc/iptables/rules.v4* e */etc/iptables/rules.v6*, respectivamente. Em caso de alterações futuras nas configurações do firewall, é possível gravá-las de forma fácil com o comando:

```
# /etc/init.d/netfilter-persistent save
[....] Saving netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
done.
```

Se quiser limpar todas as regras de firewall e começar do zero, execute:

```
# /etc/init.d/netfilter-persistent flush
[....] Flushing netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables flush
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables flush
done.
```

```
# iptables -L -vn
Chain INPUT (policy ACCEPT 13 packets, 1558 bytes)
  pkts bytes target    prot opt in     out     source                   destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source                   destination

Chain OUTPUT (policy ACCEPT 11 packets, 1068 bytes)
  pkts bytes target    prot opt in     out     source                   destination
```

Se cometer qualquer erro durante o processo de configuração ou simplesmente quiser recarregar o conjunto de regras gravado em `/etc/iptables/rules.v4`, execute:

```
# /etc/init.d/netfilter-persistent reload
[....] Loading netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables start
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables start
done.
```



```
# iptables -L -vn
Chain INPUT (policy DROP 70 packets, 19740 bytes)
  pkts bytes target     prot opt in     out     source            destination
    0     0 ACCEPT     all  --  lo      *        0.0.0.0/0         0.0.0.0/0
   111 5861 ACCEPT     all  --  *       *        0.0.0.0/0         0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     icmp --  *       *       10.0.42.0/24      0.0.0.0/0
icmp type 255
    0     0 ACCEPT     icmp --  *       *       192.168.42.0/24   0.0.0.0/0
icmp type 255
    0     0 ACCEPT     tcp  --  *       *       192.168.42.0/24   0.0.0.0/0
tcp dpt:22

Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source            destination
    0     0 ACCEPT     all  --  *       *        0.0.0.0/0         0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *       enp0s3   10.0.42.0/24      0.0.0.0/0
multiport dports 80,443
    0     0 ACCEPT     tcp  --  *       enp0s3   192.168.42.0/24   0.0.0.0/0
multiport dports 80,443

Chain OUTPUT (policy ACCEPT 120 packets, 12617 bytes)
  pkts bytes target     prot opt in     out     source            destination
```

4) Configuração do servidor DNS primário

Iremos agora configurar a máquina **ns1** como o servidor DNS primário da rede; para tanto, usaremos os programas NSD e Unbound.

O NSD (*Name Server Daemon*) é um servidor DNS *open-source* desenvolvido pela NLNet de Amsterdã em parceria com a RIPE NCC, o registro regional de Internet da Europa, oeste da Ásia e ex-países da União Soviética. Ele foi projetado para atuar exclusivamente como um servidor DNS autoritativo, não implementando portanto funções de *cache* recursiva como sua contraparte mais famosa, o BIND. A ideia por trás disso é aumentar a variabilidade de implementações DNS disponíveis, aumentando a resiliência do sistema DNS contra falhas de software e *exploits*.

O Unbound, por outro lado, é um *resolver* DNS com funções de validação, *cache* e recursividade também implementado pela NLNet Labs. Como se pode observar, então, ele faz as funções complementares às do NSD. Por sua percepção como sendo um software mais moderno, enxuto e seguro que seus concorrentes, o Unbound foi adotado como o *resolver* padrão do sistema-base dos sistemas FreeBSD e OpenBSD em 2014.

1. O primeiro passo, naturalmente, é instalar os pacotes dos programas mencionados:

```
# apt-get install nsd unbound dnsutils
```

2. Vamos começar configurando o **NSD**. Primeiro, pare o *daemon*:

```
# systemctl stop nsd
```

O programa pode ser controlado no terminal usando o comando **nsd-control** — para tanto, é necessário gerar as chaves TLS de controle com o comando:

```
# nsd-control-setup
setup in directory /etc/nsd
nsd_server.key exists
nsd_control.key exists
create nsd_server.pem (self signed certificate)
create nsd_control.pem (signed client certificate)
Signature ok
subject=CN = nsd-control
Getting CA Private Key
Setup success. Certificates created.
```

Verifique a criação das chaves com o comando:

```
# ls -l /etc/nsd | grep '.key\|.pem'
nsd_control.key
nsd_control.pem
nsd_server.key
nsd_server.pem
```

3. Faça um backup do arquivo de configuração original **/etc/nsd/nsd.conf**:

```
# mv /etc/nsd/nsd.conf /etc/nsd/nsd.conf.orig
```

Em seguida, crie o arquivo novo **/etc/nsd/nsd.conf** com o seguinte conteúdo:

```
1 server:
2   ip-address: 127.0.0.1
3   ip-address: 10.0.42.1
4   do-ip4: yes
5   port: 8053
6   username: nsd
7   zonesdir: "/etc/nsd/zones"
8
9   logfile: "/var/log/nsd.log"
10  pidfile: "/run/nsd/nsd.pid"
11  hide-version: yes
12  version: "intnet DNS"
13  identity: "unidentified server"
14
15 remote-control:
16   control-enable: yes
17   control-interface: 127.0.0.1
18   control-port: 8952
19   server-key-file: "/etc/nsd/nsd_server.key"
20   server-cert-file: "/etc/nsd/nsd_server.pem"
21   control-key-file: "/etc/nsd/nsd_control.key"
22   control-cert-file: "/etc/nsd/nsd_control.pem"
23
24 key:
25   name: "inkey"
26   algorithm: sha512
27   secret: "TSIGKEY"
28
29 pattern:
30   name: "inslave"
31   notify: 10.0.42.2@8053 inkey
32   provide-xfr: 10.0.42.2 inkey
33
34 zone:
35   name: "intnet"
36   include-pattern: "inslave"
37   zonefile: "intnet.zone"
38
39 zone:
40   name: "42.0.10.in-addr.arpa"
41   include-pattern: "inslave"
42   zonefile: "10.0.42.zone"
```

Abaixo, destacamos e explicamos algumas das diretivas mais relevantes do arquivo acima:

- **server/ip-address**: define os endereços de rede nos quais o NSD irá escutar — atenderemos requisições de *localhost* e do servidor DNS secundário, que será instalado na DMZ a seguir. O controle de quais *hosts* estarão autorizados a consultar o NSD será feito via diretivas **notify** e **provide-xfr**, bem como restrição via regra de firewall.

- **server/port**: define a porta em que o NSD irá escutar; como iremos instalar o Unbound na mesma máquina (escutando na porta 53/UDP), escolhemos a porta alternativa 8053 para evitar conflitos de *bind*.
- **server/username**: usuário com o qual o NSD irá operar. É possível aplicar um controle adicional além da redução de privilégios para um usuário comum, o uso de *chroot*, que não faremos nesta atividade.
- **server/zonesdir**: diretório em que serão armazenados os arquivos de zonas do NSD.
- **server/hide-version**, **server/version** e **server/identity**: os três controles objetivam mascarar o software rodando na máquina local, dificultando a tarefa de *banner grabbing* e identificação de *exploits* por parte de eventuais atacantes.
- **remote-control/control-enable** e **remote-control/interface**: define que o NSD poderá ser controlado "remotamente", mas logo depois restringe o IP de escuta para 127.0.0.1, efetivamente autorizando conexão de controle exclusivamente a partir de *localhost*.
- **key/secret**: define uma chave secreta que servidores primário e secundário deverão possuir em comum para que a transferência de zona seja feita com sucesso. Faremos a geração dessa chave e sua substituição de forma automática no arquivo a seguir.
- **pattern**: define um conjunto de diretivas que serão inseridas em diversas zonas a seguir (efetivamente, como uma *macro*). No caso, estamos indicando que o servidor 10.0.42.2 será notificado de alterações de zona (**notify**), e transferências de zona partir desse endereço serão autorizadas (**provide-xfr**).
- **zone/intnet**: define um arquivo de zona direta para o qual o servidor NSD será autoritativo, o nome de domínio **intnet**. A sintaxe do arquivo é idêntica à usada no software BIND.
- **zone/42.0.10-in-addr.arpa**: define um arquivo de zona reversa para o qual o servidor NSD será autoritativo, a faixa de endereços **10.0.42.0/24**. A sintaxe do arquivo é idêntica à usada no software BIND.

A transferência de zonas entre servidor primário e secundário é assegurada, além dos controles de interface de escuta e regras de firewall, por uma chave secreta TSIG (*Transaction SIGnature*) em formato Base64. Para inseri-la no arquivo use o comando:

```
# tsigkey_t=$( dd if=/dev/urandom of=/dev/stdout count=1 bs=32 2> /dev/null | base64
); sed -i "s|TSIGKEY|$tsigkey_t|" /etc/nsd/nsd.conf ; unset tsigkey_t
```

Verifique a inserção da chave com:

```
# grep 'secret:' /etc/nsd/nsd.conf
secret: "i7IoB5VDHVOCW9wvuOGQuFLNu8hzfAb1VAbCD1SbPL4="
```

Lembre-se que precisaremos dessa mesma chave quando estivermos configurando o servidor secundário.

4. Crie o diretório de zonas **/etc/nsd/zones**:

```
# mkdir /etc/nsd/zones
```

Agora, crie o arquivo novo `/etc/nsd/zones/intnet.zone`, com as configuração de zona direta para o domínio `intnet.`, com o seguinte conteúdo:

```
1 $TTL 86400 ; (1 day)
2 $ORIGIN intnet.
3
4 @      IN    SOA    ns1.intnet.  admin.intnet. (
5                2018111000    ;serial (YYYYMMDDnn)
6                14400          ;refresh (4 hours)
7                1800           ;retry (30 minutes)
8                1209600        ;expire (2 weeks)
9                3600           ;negative cache TTL (1 hour)
10       )
11
12 @      IN    NS     ns1.intnet.
13 @      IN    NS     ns2.intnet.
14
15 @      IN    MX     10    mx1.intnet.
16 @      IN    MX     20    mx2.intnet.
17
18 ns1    IN    A       10.0.42.1
19 ns2    IN    A       10.0.42.2
20
21 mx1    IN    A       10.0.42.91
22 mx2    IN    A       10.0.42.92
23
24 fw     IN    CNAME   ns1
```

A sintaxe é exatamente a mesma utilizada para um arquivo de zonas do BIND. Note que:

- Definimos a máquina `ns1.intnet.` como o servidor autoritativo para o domínio `intnet.`, com email de contato `admin@intnet.`
- O serial é `2018111000` — este é um número que deve identificar a versão do arquivo de zonas, e incrementado a cada atualização. O formato escolhido para esse *serial* foi `ANO-MES-DIA-VERSAO`, de forma que versões do arquivo em datas futuras terão sempre um valor superior ao de versões antigas.
- Os servidores DNS responsáveis pelo domínio são `ns1.intnet` e `ns2.intnet.`
- Os servidores de e-mail (MX — *mail exchange*) do domínio são `mx1.intnet.` e `mx2.intnet.`. Ambos são servidores fictícios, que não implantaremos neste curso, e mostrados aqui apenas para demonstrar a sintaxe desse tipo de entrada no arquivo de zonas.
- Configuram-se registros A (*address*) para as máquinas mencionadas.
- Configura-se um registro CNAME (*canonical name*) para a máquina `ns1.intnet.`, `fw.intnet.`. Com isso, os usuário poderão referir-se a essa máquina também pelo seu "apelido".

5. Vamos para o arquivo de zona reversa. Crie o arquivo novo `/etc/nsd/zones/10.0.42.zone` com o seguinte conteúdo:

```
1 $TTL 86400 ; (1 day)
2 $ORIGIN 42.0.10.in-addr.arpa.
3
4 @      IN      SOA    ns1.intnet.  admin.intnet. (
5          2018111000    ;serial (YYYYMMDDnn)
6          14400         ;refresh (4 hours)
7          1800          ;retry (30 minutes)
8          1209600       ;expire (2 weeks)
9          3600          ;negative cache TTL (1 hour)
10         )
11
12 @      IN      NS     ns1.intnet.
13 @      IN      NS     ns2.intnet.
14
15 @      IN      MX     10    mx1.intnet.
16 @      IN      MX     20    mx2.intnet.
17
18 1      IN      PTR     ns1.intnet.
19 2      IN      PTR     ns2.intnet.
20
21 91     IN      PTR     mx1.intnet.
22 92     IN      PTR     mx2.intnet.
```

Nada muito diferente neste arquivo em relação ao anterior — note apenas que os registros inseridos aqui são do tipo PTR (*pointer*), fazendo o mapeamento reverso entre endereços IP e nomes de domínio. Note, ainda, que o *ORIGIN* do arquivo é definido como `42.0.10.in-addr.arpa.`, já que somos o servidor autoritativo para a faixa 10.0.42.0/24.

6. Vamos verificar se a sintaxe dos arquivos de configuração não possui erros:

```
# nsd-checkconf /etc/nsd/nsd.conf
```

Agora, basta reiniciar o serviço e verificar sua correta operação:

```
# nsd-control start
```

```
# tail /var/log/nsd.log
[2018-11-13 17:17:26.763] nsd[2012]: notice: nsd starting (NSD 4.1.14)
[2018-11-13 17:17:26.780] nsd[2014]: notice: nsd started (NSD 4.1.14), pid 2013
```

```
# ss -tunlp | grep 8053
udp    UNCONN    0      0      10.0.42.1:8053      *:~
users: (("nsd",pid=648,fd=5),("nsd",pid=647,fd=5),("nsd",pid=646,fd=5))
udp    UNCONN    0      0      127.0.0.1:8053      *:~
users: (("nsd",pid=648,fd=4),("nsd",pid=647,fd=4),("nsd",pid=646,fd=4))
tcp    LISTEN     0      128    10.0.42.1:8053      *:~
users: (("nsd",pid=648,fd=7),("nsd",pid=647,fd=7),("nsd",pid=646,fd=7))
tcp    LISTEN     0      128    127.0.0.1:8053      *:~
users: (("nsd",pid=648,fd=6),("nsd",pid=647,fd=6),("nsd",pid=646,fd=6))
```

7. Vamos colocar o servidor à prova: teste a resolução direta de nomes usando a ferramenta **dig**. Lembre-se que o NSD está operando na porta 8053/UDP, e não na porta padrão:

```
# dig @127.0.0.1 -p 8053 fw.intnet +noadditional +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 -p 8053 fw.intnet +noadditional
+noquestion
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64874
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
fw.intnet.          86400    IN       CNAME    ns1.intnet.
ns1.intnet.         86400    IN       A        10.0.42.1

;; AUTHORITY SECTION:
intnet.             86400    IN       NS       ns1.intnet.
intnet.             86400    IN       NS       ns2.intnet.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 09:19:30 -02 2018
;; MSG SIZE rcvd: 120
```

Excelente! Todas as seções de resposta parecem corretas, tanto ANSWER quando AUTHORITY. Vamos verificar a resolução reversa:

```
# dig @127.0.0.1 -p 8053 -x 10.0.42.2 +noadditional +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 -p 8053 -x 10.0.42.2 +noadditional
+noquestion
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4272
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
2.42.0.10.in-addr.arpa. 86400    IN      PTR      ns2.intnet.

;; AUTHORITY SECTION:
42.0.10.in-addr.arpa.  86400    IN      NS       ns1.intnet.
42.0.10.in-addr.arpa.  86400    IN      NS       ns2.intnet.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 09:19:47 -02 2018
;; MSG SIZE  rcvd: 107
```

8. Agora que o NSD está configurado, vamos implementar o **Unbound**. Primeiro, pare o *daemon* se estiver rodando:

```
# systemctl stop unbound
```

Assim como no caso do NSD, o Unbound pode ser controlado via **unbound-control** — gere as chaves de controle:

```
# unbound-control-setup
setup in directory /etc/unbound
unbound_server.key exists
unbound_control.key exists
create unbound_server.pem (self signed certificate)
create unbound_control.pem (signed client certificate)
Signature ok
subject=CN = unbound-control
Getting CA Private Key
Setup success. Certificates created.
```

Faça um backup do arquivo de configuração original **/etc/unbound/unbound.conf**:


```
# mv /etc/unbound/unbound.conf /etc/unbound/unbound.conf.orig
```

Em seguida, crie o arquivo novo `/etc/unbound/unbound.conf` com o seguinte conteúdo:

```
1 server:
2   interface: 127.0.0.1
3   interface: 10.0.42.1
4   interface: 192.168.42.1
5   port: 53
6
7   access-control: 127.0.0.0/8 allow
8   access-control: 10.0.42.0/24 allow
9   access-control: 192.168.42.0/24 allow
10
11  cache-min-ttl: 300
12  cache-max-ttl: 14400
13
14  local-zone: "intnet" nodefault
15  domain-insecure: "intnet"
16
17  local-zone: "10.in-addr.arpa." nodefault
18  domain-insecure: "10.in-addr.arpa."
19
20  verbosity: 1
21  prefetch: yes
22  hide-version: yes
23  hide-identity: yes
24  use-caps-for-id: yes
25  rrset-roundrobin: yes
26  minimal-responses: yes
27  do-not-query-localhost: no
28
29 stub-zone:
30   name: "intnet"
31   stub-addr: 127.0.0.1@8053
32
33 stub-zone:
34   name: "42.0.10.in-addr.arpa."
35   stub-addr: 127.0.0.1@8053
36
37 forward-zone:
38   name: "."
39   forward-addr: 8.8.8.8
40   forward-addr: 8.8.4.4
41
42 include: "/etc/unbound/unbound.conf.d/*.conf"
```

Abaixo, destacamos e explicamos algumas das diretivas mais relevantes do arquivo acima:

- `server/interface`: define os endereços de rede nos quais o Unbound irá escutar — atenderemos requisições de *localhost* e das sub-redes DMZ e Intranet.
- `server/port`: o Unbound irá escutar na porta padrão, 53/UDP.
- `server/access-control`: define que as sub-redes declaradas (*localhost*, DMZ e Intranet, novamente) terão permissão para utilizar os serviços de resolução de nomes.
- `server/local-zone` e `server/domain-insecure`: define opções para as zonais locais `intnet.` e `42.0.10.in-addr.arpa.`, e que a cadeia de confiança DNSSEC não será verificada para esses domínios.
- `server/hide-version` e `server/hide-identity`: controles que objetivam mascarar o software rodando na máquina local, dificultando a tarefa de *banner grabbing* e identificação de *exploits* por parte de eventuais atacantes.
- `stub-zone` e `stub-zone/stub-addr`: declara zonas autoritativas locais que devem ser consultadas para os domínios especificados, para as quais o registro público DNS não será usado. Usaremos o servidor NSD rodando em *localhost*, consultando a porta 8053/UDP.
- `forward-zone` e `forward-zone/forward-addr`: define servidores nos quais o Unbound irá buscar recursão caso não consiga resolver um nome. Como está sendo declarado o nome de zona `."`, todas as pesquisas serão redirecionadas para os servidores indicados em `forward-addr`.

9. Vamos verificar o funcionamento do *daemon*. Inicie o Unbound usando o `unbound-control`:

```
# unbound-control start
```

Verifique seu funcionamento usando o `ss`:

```
# ss -unlp | grep :53
UNCONN    0      0      192.168.42.1:53          *:*
users:(("unbound",pid=2084,fd=7))
UNCONN    0      0      10.0.42.1:53           *:*
users:(("unbound",pid=2084,fd=5))
UNCONN    0      0      127.0.0.1:53           *:*
users:(("unbound",pid=2084,fd=3))
```

Reconfigure o DNS *system-wide*, no arquivo `/etc/resolv.conf`.

```
# nano /etc/resolv.conf
(...)
```

```
# cat /etc/resolv.conf
domain intnet.
search intnet.
nameserver 10.0.42.1
nameserver 10.0.42.2
```

Devido ao fato de estarmos usando DHCP na interface de saída (**enp0s3**), o arquivo **/etc/resolv.conf** poderá ser sobrescrito sempre que as interfaces de rede forem reiniciadas, ou após o *reboot* da máquina. Para evitar isso, marque o arquivo como imutável:

```
# chattr +i /etc/resolv.conf
```

10. Feito! Vamos testar a resolução de domínios internos **unbound** com a ferramenta **dig**.

```
# dig fw.intnet +noquestion +noadditional

; <<>> DiG 9.10.3-P4-Debian <<>> fw.intnet +noquestion +noadditional
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63613
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
fw.intnet.            86400    IN       CNAME    ns1.intnet.
ns1.intnet.           86400    IN       A        10.0.42.1

;; Query time: 0 msec
;; SERVER: 10.0.42.1#53(10.0.42.1)
;; WHEN: Mon Nov 12 09:39:55 -02 2018
;; MSG SIZE  rcvd: 72
```

E quanto a nomes de domínio externos, usando recursão? Vejamos:

```
# dig openbsd.org +noquestion +noadditional

; <<>> DiG 9.10.3-P4-Debian <<>> openbsd.org +noquestion +noadditional
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5694
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
openbsd.org.          21599    IN       A        129.128.5.194

;; Query time: 420 msec
;; SERVER: 10.0.42.1#53(10.0.42.1)
;; WHEN: Mon Nov 12 09:40:33 -02 2018
;; MSG SIZE  rcvd: 56
```

11. Ainda não acabou! Lembre-se que nossas regras de firewall configuradas no começo desta sessão estão extremamente restritivas, então temos que criar exceções para que a consulta de nomes funcione. Vamos ver os tipos de acesso que precisamos liberar:
- O NSD será consultado apenas pelo Unbound em *localhost* (não é necessário criar regras neste caso), e pelo servidor DNS secundário em 10.0.42.2. Como nesse caso será feita transferência de zonas entre os servidores, é necessário liberar tráfego nos protocolos TCP e UDP na porta 8053.
 - O Unbound local será consultado por todas as máquinas da DMZ e Intranet, na porta 53/UDP.
 - Os clientes da Intranet devem conseguir consultar o Unbound rodando no servidor DNS secundário, porta 53/UDP. Não é necessário criar uma regra para a DMZ, neste caso, pois as máquinas estão na mesma sub-rede e portanto seu tráfego não passa pelo firewall (na *chain FORWARD*).

Vamos ao requisito (a):

```
# iptables -A INPUT -s 10.0.42.2/32 -p tcp -m tcp --dport 8053 -j ACCEPT
```

```
# iptables -A INPUT -s 10.0.42.2/32 -p udp -m udp --dport 8053 -j ACCEPT
```

Agora, o requisito (b):

```
# iptables -A INPUT -s 10.0.42.0/24,192.168.42.0/24 -p udp -m udp --dport 53 -j ACCEPT
```

E, finalmente, o critério (c) estabelecido inicialmente:

```
# iptables -A FORWARD -s 192.168.42.0/24 -d 10.0.42.2/32 -p udp -m udp --dport 53 -j ACCEPT
```

Salve as regras na configuração do firewall local:

```
# /etc/init.d/netfilter-persistent save
[....] Saving netfilter rules...run-parts: executing /usr/share/netfilter-
persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
done.
```

5) Configuração do DNSSEC

Vamos agora configurar o DNSSEC em nosso servidor autoritativo, de forma a produzir consultas assinadas digitalmente. Evidentemente, como estamos configurando um domínio fictício não

iremos conseguir propagar a assinatura pela cadeia de confiança raiz, mas esta atividade servirá como um exemplo caso você queira fazê-lo em sua organização posteriormente.

1. Primeiro, vamos instalar as ferramentas de suporte para gestão/geração de chaves DNSSEC:

```
# apt-get install ldnsutils haveged
```

2. Vamos criar as chaves de assinatura de zona (ZSK, ou *Zone Signing Key*) e chave (KSK, ou *Key Signing Key*). O KSK é um par de chaves público/privada usado para gerar uma assinatura digital para o ZSK. O ZSK, por sua vez, é um par de chaves público/privada para gerar uma assinatura digital conhecida como RRSIG (*Resource Record Signature*) para cada um dos RRSETs (*Resource Record Sets*) da zona. O website da Cloudflare (<https://www.cloudflare.com/dns/dnssec/how-dnssec-works/>) possui uma excelente explicação sobre o sistema DNSSEC que pode ser usada para consulta.

Primeiro, entre no diretório `/etc/nsd`:

```
# cd /etc/nsd/
```

Agora, vamos gerar o ZSK/KSK e armazenar seus nomes em uma variável temporária. Os algoritmos escolhidos estão de acordo com as melhores práticas recomendadas na RFC 6944 (<https://tools.ietf.org/html/rfc6944>).

```
# export ZSK=$( ldns-keygen -a RSASHA512 -b 2048 intnet )
```

```
# export KSK=$( ldns-keygen -k -a RSASHA512 -b 2048 intnet )
```

Podemos remover o registro DS (*Delegation of Signing*) auto-gerado, já que iremos regerá-lo futuramente sob demanda.

```
# rm Kintnet.*.ds
```

Vejamos como ficaram as chaves ZSK/KSK para o domínio:

```
# ls -1 Kintnet.*
Kintnet.+010+14936.key
Kintnet.+010+14936.private
Kintnet.+010+42867.key
Kintnet.+010+42867.private
```

Para identificar qual dessas chaves é a ZSK e qual é a KSK, consulte as variáveis do *shell*:

```
# echo $ZSK  
Kintnet.+010+14936
```

```
# echo $KSK  
Kintnet.+010+42867
```

3. Agora, vamos assinar a zona **intnet.zone** com o comando **ldns-signzone**:

```
# ldns-signzone -n -p -s $(head -n 1000 /dev/urandom | sha1sum | cut -b 1-16)  
/etc/nsd/zones/intnet.zone $ZSK $KSK
```

O que ocorreu? Verifique o conteúdo do diretório **/etc/nsd/zones**:

```
# ls -l /etc/nsd/zones  
10.0.42.zone  
intnet.zone  
intnet.zone.signed
```

Vamos configurar o NSD para usar a zona assinada. Para isso, basta substituir o caminho de busca do arquivo de zona direta **/etc/nsd/zones/intnet.zone** por sua contraparte assinada:

```
# sed -i 's/intnet\.zone/intnet\.zone\.signed/' /etc/nsd/nsd.conf
```

```
# grep -B3 intnet.zone.signed /etc/nsd/nsd.conf  
zone:  
  name: "intnet"  
  include-pattern: "inslave"  
  zonefile: "intnet.zone.signed"
```

Para informar ao NSD que o arquivo de zonas mudou, basta usar o programa **nsd-control** como mostrado a seguir:

```
# nsd-control reconfig  
reconfig start, read /etc/nsd/nsd.conf  
ok
```

```
# nsd-control reload intnet  
ok
```

4. Vamos testar se nossa configuração surtiu efeito. Primeiro, pesquise pelos registros DNSKEY do domínio no NSD em **localhost**, verificando as chaves ZSK e KSK:

```
# dig -p 8053 @localhost DNSKEY intnet. +multiline +nored +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> -p 8053 @localhost DNSKEY intnet. +multiline
+nored +noquestion
; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18768
;; flags: qr aa; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
intnet.                86400 IN DNSKEY 256 3 10 (
                        AwEAAerseiW1Vud7L90iBA4IgzUUXpkw/k/6kKIHX5Qm
                        Cm1HvtLTT/dSFk9JOHVBoQ2Rpk0LQBc+WZsnqz6dQZJn
                        b9FkGzsAh39BnP8g0A1d+SfaGGkceEteQfgi3rKz5dnk
                        EWgOonWk2vGOJ5oPDvCy4aUQqfTACY9Tk0qeWZT7enfy
                        gdrg4AjaTKBdJ4kruBj2z3FKXBoLDY03HTPmosKe94Xr
                        muiLQa3uHFvFvu6k/X0mH1kCS+lb91R/OF60mG40iW4/
                        z8aLNSKPuUwJNddvZ0I14F7SwN7YGaVpoY11DAsxmbS5
                        1lGxp6Fc0fWQpAfT4WG9/mbyHj29kZnP/Oxktsk=
                        ) ; ZSK; alg = RSASHA512; key id = 14936
intnet.                86400 IN DNSKEY 257 3 10 (
                        AwEAAIo8BoUEY6ab3YBbs24FEj8Vt9aRByAFyKJk0sH
                        BiNIy9I1vXZIGB+5wLIKpZt4ZVd3oY03MDExxDYZ1HVS
                        D1magCRBzLF2oeTCWY3kLz/9ls23RV9r5IMY68aPHMg0
                        tDBJ8IZTC9Sb0+os3L9jazwpOL22Ak134YN6iIC6kXpQ
                        gN3TtNorzD8DIHkBVJ9NrpKYJjzt0g8oTyg0La3xTnc0
                        6q51Q4eVBnlcXbbJ6gquSFzAnoJ9qzq5JUnbvAB9I4yv
                        Spo3RGcX3LZFmsDvsDfnwkXShyeEPdGTr6m/mbqPrceW
                        R53QL0WnZq1KrczPwTLKjgtzgw+F7o18YDdQ0wU=
                        ) ; KSK; alg = RSASHA512; key id = 42867

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 11:13:56 -02 2018
;; MSG SIZE rcvd: 587
```

Perfeito! Aparentemente, nossas chaves foram registradas e estão sendo utilizadas com sucesso. O Unbound, no entanto, ainda possui algumas entradas na *cache* desatualizadas, prévias à configuração do DNSSEC, que foram mantidas pelas consultas anteriores. Para limpá-las, basta usar o comando `unbound-control flush_zone`:

```
# unbound-control flush_zone intnet.
ok removed 4 rrsets, 2 messages and 0 key entries
```

Agora sim, faça uma consulta via Unbound usando DNSSEC:

```
# dig fw.intnet +dnssec +multiline +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> fw.intnet +dnssec +multiline +noquestion
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47390
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; ANSWER SECTION:
fw.intnet.                86355 IN CNAME ns1.intnet.
fw.intnet.                86355 IN RRSIG CNAME 10 2 86400 (
                           20181210130559 20181112130559 14936 intnet.
                           u2DBcjAp91qYSYvspczvWiW8QuIX24aAoFTzfM8hJdUQ
                           3VcjekyJeQQfNPIM4+rE9ZdTyrADS4nKCF1/XqkHwFST
                           kitVm9So17I4GBJ+sgpKkpoKEwh6IfbUB563Mmfj9jIQ
                           SvUR0dP0ot9QV+DXWCFLD02tvXT0BSltBT9Nogx1h926
                           5lkSZ0F+5x0Ss1Law5Rpn8ZKyAAwCYS0UNJIqB8XzIqb
                           eSblG5Q4Si7Qwrccv7L16DscBbh1gjmncAYCE63XF3SN
                           s0N9SgTgfzibgVaVGw+S1SoVYjvxDVd5ZW/tpQxhx4kY
                           Wt/c+541nj7xmFT58/dnoo8t6Meo/JCKgw== )
ns1.intnet.               86297 IN A 10.0.42.1
ns1.intnet.               86297 IN RRSIG A 10 2 86400 (
                           20181210130559 20181112130559 14936 intnet.
                           bmwQPD9b/GwUX2ZvpP6ba0zyEXNW/ghcc9rqzZSWwUYn
                           VfX+UqRwzon/2LxJfY01uJh0v8no02S5+zb7saroDsCi
                           9/Oaukw+iAvwrZh6V01dHDr6WjA1S4hi9MxmfJ94NwD7
                           txB5WLG39KM0V/EpH+v4L6ser4v+oYFSer7a6EsZxCcD
                           nfcikN5L/QUWVMacduxpRM16+MI83bEf/uLC82xIthf
                           2cVVdsDA4xfZvA0cYS/IXUauYXaj9fkE2TLVoyIC1UIM
                           XE6CSbuLo2iWc+V+lJ5tkP9kctmqit3rfegWXJlQ4es3
                           4fsxR64+Weqre/iTMpVxVss/TLus7g70iA== )

;; Query time: 0 msec
;; SERVER: 10.0.42.1#53(10.0.42.1)
;; WHEN: Mon Nov 12 11:17:12 -02 2018
;; MSG SIZE rcvd: 660
```

Caso fosse desejável exportar a configuração DNS para um *registrar* hierarquicamente superior (fechando a cadeia de verificação DNS), pode-se gerar os registros DS das chaves com o comando abaixo.

```
# ldns-key2ds -n -1 /etc/nsd/zones/intnet.zone.signed && ldns-key2ds -n -2
/etc/nsd/zones/intnet.zone.signed
intnet. 86400 IN DS 42867 10 1 5b8c2c011bd011618f7e339667c075587e65ab05
intnet. 86400 IN DS 42867 10 2
424bcf7d7c09df8be1520c4230680b5c33a63598b4c6185c7884592ad3bce63b
```


6) Automatizando assinatura DNSSEC após alterações

1. É claro, seria muito inconveniente ter que realizar todos os passos que fizemos na atividade (5) a cada alteração no servidor DNS. Vamos automatizar a assinatura de zonas e reinício dos *daemons* relevantes com um *script shell*. Crie o arquivo novo `/root/scripts/signzone-intnet.sh` com o seguinte conteúdo:

```
1 #!/bin/bash
2
3 ZSK="ZSKSTUB"
4 KSK="KSKSTUB"
5 ZONE_FILE="/etc/nsd/zones/intnet.zone"
6
7 cd /etc/nsd
8
9 ldns-signzone -n \
10 -p \
11 -s $(head -n 1000 /dev/random | sha1sum | cut -b 1-16) \
12 $ZONE_FILE \
13 $ZSK \
14 $KSK
15
16 nsd-control reconfig
17 nsd-control reload intnet
18 nsd-control reload 42.0.10.in-addr.arpa
19 unbound-control flush_zone intnet.
```

Note que é necessário substituir as variáveis `ZSKSTUB` e `KSKSTUB` com os valores das variáveis `$ZSK` e `$KSK` no seu *shell* corrente, como se segue:

```
# sed -i "s/ZSKSTUB/${ZSK}/" /root/scripts/signzone-intnet.sh
```

```
# sed -i "s/KSKSTUB/${KSK}/" /root/scripts/signzone-intnet.sh
```

2. Teste a assinatura de zonas usando o *script*:

```
# bash scripts/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
ok removed 4 rrsets, 2 messages and 0 key entries
```



Ao atualizar arquivos de zona, não se esqueça que além de adicionar registros A, CNAME ou PTR conforme necessário, é também **imprescindível** incrementar o valor do *serial* do arquivo, próximo do topo no registro SOA. Se isso não for feito, o NSD não irá detectar que o arquivo de zona foi atualizado e suas modificações não serão propagadas.

7) Reconfiguração da VM *debian-template*

Agora que temos um firewall e servidores DNS na rede é interessante que alteremos a configuração padrão da VM *debian-template* para que as novas máquinas, clonadas a partir dela, já estejam corretamente ajustadas para operar em nosso *datacenter* simulado.

1. Ligue a máquina *debian-template* e acesse o terminal como o usuário *root*.

```
# hostname ; whoami
debian-template
root
```

2. Com a máquina *debian-template* ligada, acesse na janela principal do Virtualbox o menu *Settings > Network > Adapter 1 > Attached to* e altere a opção para *Host-only Adapter*. O nome da rede *host-only* escolhido deve ser o mesmo alocado para a interface de rede da máquina virtual *ns1* que está conectada à DMZ. Seguindo o exemplo mostrado no início desta sessão, portanto, a rede escolhida seria a *Virtualbox Host-Only Ethernet Adapter #2*.
3. De volta ao terminal da máquina *debian-template*, vamos configurar a rede: edite o arquivo */etc/network/interfaces* como se segue:

```
# nano /etc/network/interfaces
(...)
```

```
# cat /etc/network/interfaces
source /etc/network/interfaces.d/*

auto lo enp0s3

iface lo inet loopback

iface enp0s3 inet static
address 10.0.42.253/24
gateway 10.0.42.1
```

Para garantir que nenhum endereço IP antigo, primário, se mantenha alocado às interfaces, execute o comando *flush* e em seguida reinicie a rede do sistema:

```
# ip addr flush label 'enp0s*' ; systemctl restart networking
```

Verifique que as interfaces estão com os endereços corretamente alocados:

```
# ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}'
10.0.42.253/24 enp0s3
```

4. Agora, configure a resolução de nomes no arquivo `/etc/resolv.conf`:

```
# nano /etc/resolv.conf
(...)
```

```
# cat /etc/resolv.conf
domain intnet.
search intnet.
nameserver 10.0.42.1
nameserver 10.0.42.2
```

5. Vamos alterar o script `/root/scripts/changehost.sh` para que possamos configurar, de uma vez só, informações como endereço IP e `hostname` na máquina clonada. Apague todo o conteúdo do script original e substitua-o pelo seguinte:

```
1 #!/bin/bash
2
3
4 # exibir uso do script e sair
5 function usage() {
6     echo "  Usage: $0 -h HOSTNAME -i IPADDR -g GATEWAY"
7     echo "  Netmask is assumed as /24."
8     exit 1
9 }
10
11
12 # testar sintaxe valida de HOSTNAME
13 function valid_host() {
14     if [[ "$nhost" =~ [^a-z0-9] ]]; then
15         echo "  [*] HOSTNAME must be lowercase alphanumeric: [a-z0-9]*"
16         usage
17     elif [ ${#nhost} -gt 63 ]; then
18         echo "  [*] HOSTNAME must have <63 chars"
19         usage
20     fi
21 }
22
23
```

```
24 # testar sintaxe valida de IPADDR/GATEWAY
25 function valid_ip() {
26     local ip=$1
27     local stat=1
28
29     if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
30         OIFS=$IFS
31         IFS='.'
32         ip=($ip)
33         IFS=$OIFS
34         [[ ${ip[0]} -le 255 && \
35             ${ip[1]} -le 255 && \
36             ${ip[2]} -le 255 && \
37             ${ip[3]} -le 255 ]]
38         stat=$?
39     fi
40
41     if [ $stat -ne 0 ] ; then
42         echo " [*] Invalid syntax for $2"
43         usage
44     fi
45 }
46
47
48 while getopts ":g:h:i:" opt; do
49     case "$opt" in
50         g)
51             ngw=${OPTARG}
52             ;;
53         h)
54             nhost=${OPTARG}
55             ;;
56         i)
57             nip=${OPTARG}
58             ;;
59         *)
60             usage
61             ;;
62     esac
63 done
64
65 # testar se parametros foram informados
66 [ -z $ngw ] && { echo " [*] No gateway?"; usage; }
67 [ -z $nhost ] && { echo " [*] No hostname?"; usage; }
68 [ -z $nip ] && { echo " [*] No ipaddr?"; usage; }
69
70 # testar sintaxe de parametros
71 valid_ip $nip "IPADDR"
72 valid_ip $ngw "GATEWAY"
73 valid_host $nhost
74
```

```
75 # alterar endereco ip/gateway
76 iff="/etc/network/interfaces"
77 cip="$( egrep '^address ' $iff | awk -F'[ /]' '{print $2}' )"
78 cgw="$( egrep '^gateway ' $iff | awk '{print $NF}' )"
79 sed -i "s|${cip}|${nip}|g" $iff
80 sed -i "s|${cgw}|${ngw}|g" $iff
81 ip addr flush label 'enp0s*'
82
83 # alterar hostname local
84 chost="$( hostname -s )"
85 sed -i "s|${chost}/${nhost}|g" /etc/hosts
86 sed -i "s|${chost}/${nhost}|g" /etc/hostname
87
88 invoke-rc.d hostname.sh restart
89 invoke-rc.d networking restart
90 hostnamectl set-hostname $nhost
91
92 # re-gerar chaves SSH
93 rm -f /etc/ssh/ssh_host_* 2> /dev/null
94 dpkg-reconfigure openssh-server &> /dev/null
```

Em relação ao *script* original, algumas diferenças: adicionou-se um *loop* para leitura de opções a partir da linha de comando (*hostname*, endereço IP e *gateway* a serem utilizados pela nova máquina), verificação de sintaxe dessas opções e efetiva alteração do endereço IP e *gateway* no arquivo */etc/network/interfaces*.

Feito isso, desligue a máquina *debian-template*.

8) Criação da VM de DNS secundário

Vamos agora criar a segunda máquina de nosso *datacenter* simulado, a máquina *ns2*. Ela atuará como um servidor DNS secundário sob o endereço IP 10.0.42.2/24, dentre outras funções que serão configuradas em sessões posteriores.

1. Clone a máquina *debian-template* seguindo os mesmos passos da atividade (6) da sessão 1. Para o nome da máquina, escolha *ns2*.
2. Ligue a máquina *ns2* e, após o *boot*, faça login como o usuário *root*. Usando o script */root/scripts/changehost.sh*, efetue a configuração automática da máquina:

```
# ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}' ; hostname ;
whoami
10.0.42.253/24 enp0s3
debian-template
root
```

```
# bash ~/scripts/changehost.sh -h ns2 -i 10.0.42.2 -g 10.0.42.1
```

```
# ip addr show label 'enp0s*' | grep 'inet ' | awk '{print $2,$NF}' ; hostname ;  
whoami  
10.0.42.2/24 enp0s3  
debian-template  
ns2
```

9) Configuração do DNS secundário

A configuração do NSD como um servidor DNS secundário é bastante similar à do servidor primário, com algumas poucas diferenças na seção **pattern** do arquivo original. Já a configuração do Unbound, por outro lado, é absolutamente idêntica. Vamos ao trabalho:

1. Instale os pacotes relevantes:

```
# apt-get install nsd unbound dnsutils
```

Durante a pós-configuração do Unbound o sistema pode demorar um pouco a retornar para o *shell* — seja paciente, em alguns segundos a instalação será concluída.

2. Primeiro, o **NSD**. Queremos copiar o arquivo **/etc/nsd/nsd.conf** da máquina **ns1** e fazer as alterações necessárias — no entanto, o acesso SSH à essa máquina pela DMZ está proibido pelas configurações de firewall que fizemos no começo desta sessão. Assim sendo, vamos fazer a cópia no sentido oposto: da máquina **ns1** para a máquina **ns2**.

Logue como **root** na máquina **ns1** e copie o arquivo usando o **scp**, como se segue:

```
# hostname ; whoami  
ns1  
root
```

```
# scp /etc/nsd/nsd.conf aluno@ns2:~  
The authenticity of host 'ns2 (10.0.42.2)' can't be established.  
ECDSA key fingerprint is SHA256:jxd7SPFgwNSsaMS7ApIEMpdAmxEnWeJ83s/K4h2XV5o.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added 'ns2,10.0.42.2' (ECDSA) to the list of known hosts.  
aluno@ns2's password:  
nsd.conf  
100% 909 2.5MB/s 00:00
```

Perfeito, o arquivo foi copiado para **/home/aluno/nsd.conf** na máquina **ns2** — note que não fizemos o login remoto com o usuário **root** pois esse acesso não é permitido pela política padrão do *daemon* **sshd** no Debian.

3. De volta à máquina **ns2** como o usuário **root**, pare o NSD e gere as chaves TLS de controle:

```
# hostname ; whoami
ns2
root
```

```
# systemctl stop nsd
```

```
# nsd-control-setup
setup in directory /etc/nsd
nsd_server.key exists
nsd_control.key exists
create nsd_server.pem (self signed certificate)
create nsd_control.pem (signed client certificate)
Signature ok
subject=CN = nsd-control
Getting CA Private Key
Setup success. Certificates created.
```

Vamos fazer o backup do arquivo de configuração original:

```
# mv /etc/nsd/nsd.conf /etc/nsd/nsd.conf.orig
```

Copie o arquivo `/home/aluno/nsd.conf` para a pasta `/etc/nsd` e ajuste suas permissões:

```
# cp /home/aluno/nsd.conf /etc/nsd
```

```
# chown root. /etc/nsd/nsd.conf
```

```
# ls -ld /etc/nsd/nsd.conf
-rw-r--r-- 1 root root 909 nov 12 12:14 /etc/nsd/nsd.conf
```

Excelente. Temos agora que editar o arquivo com as configurações do servidor secundário, o que faremos através do uso do `sed` nos comandos a seguir:

```
# sed -i 's/^( *ip-address:\) 10\.0\.42\.1/1 10\.0\.42\.2/' /etc/nsd/nsd.conf
```

```
# sed -i '/^ *zonesdir:/d' /etc/nsd/nsd.conf
```

```
# sed -i 's/"inslave"/"inmaster"/' /etc/nsd/nsd.conf
```

```
# sed -i 's/^( *)notify:[0-9@. ]*(.*)/\1allow-notify: 10\0.42\1 \2/'  
/etc/nsd/nsd.conf
```

```
# sed -i 's/^( *)provide-xfr:[0-9. ]*(.*)/\1request-xfr: AXFR 10\0.42\1@8053  
\2/' /etc/nsd/nsd.conf
```

O que esses comandos fizeram, afinal? O comando **diff** pode ser usado para comparar arquivos, evidenciando as diferenças entre eles:

```
# diff -u /home/aluno/nsd.conf /etc/nsd/nsd.conf  
--- /home/aluno/nsd.conf      2018-11-12 20:51:45.040000000 -0200  
+++ /etc/nsd/nsd.conf        2018-11-12 20:55:34.156000000 -0200  
@@ -1,10 +1,9 @@  
server:  
    ip-address: 127.0.0.1  
- ip-address: 10.0.42.1  
+ ip-address: 10.0.42.2  
    do-ip4: yes  
    port: 8053  
    username: nsd  
- zonesdir: "/etc/nsd/zones"  
  
    logfile: "/var/log/nsd.log"  
    pidfile: "/run/nsd/nsd.pid"  
@@ -27,17 +26,17 @@  
    secret: "i7IoB5VDHVOCW9wvuOGQuFLNu8hzfAb1VAbCD1SbPL4="  
  
pattern:  
- name: "inslave"  
- notify: 10.0.42.2@8053 inkey  
- provide-xfr: 10.0.42.2 inkey  
+ name: "inmaster"  
+ allow-notify: 10.0.42.1 inkey  
+ request-xfr: AXFR 10.0.42.1@8053 inkey  
  
zone:  
    name: "intnet"  
- include-pattern: "inslave"  
+ include-pattern: "inmaster"  
    zonefile: "intnet.zone.signed"  
  
zone:  
    name: "42.0.10.in-addr.arpa"  
- include-pattern: "inslave"  
+ include-pattern: "inmaster"  
    zonefile: "10.0.42.zone"
```


Não se esqueça de remover o arquivo `/home/aluno/nsd.conf`:

```
# rm /home/aluno/nsd.conf
```

4. Inicie o *daemon* do NSD usando o `nsd-control`:

```
# nsd-control start
```

Note que não foi necessário criar o diretório de zonas no servidor secundário, `/etc/nsd/zones`, já que as zonas transferidas ficam mantidas no diretório `/var/lib/nsd`:

```
# ls -l /var/lib/nsd
nsd.db
xfrd.state
```

Vamos testar? Tente efetuar uma resolução direta de nomes no servidor local usando a ferramenta `dig`, também testando o funcionamento do DNSSEC:

```
# dig @127.0.0.1 -p 8053 ns1.intnet +dnssec +noadditional +noquestion +multiline
+noauthority

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 -p 8053 ns1.intnet +dnssec
+noadditional +noquestion +multiline +noauthority
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 29660
;; flags: qr aa rd; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 3
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; ANSWER SECTION:
ns1.intnet.          86400 IN A 10.0.42.1
ns1.intnet.          86400 IN RRSIG A 10 2 86400 (
                        20181210223558 20181112223558 14936 intnet.
                        TdK1Ivm1Ykdk1MBaMQSqNHANV14FQbfdBCgi0znMc7wT
                        vqZPa2TBMTe2GYvcjbWd2WzH4o88p0AzwmCxMnnAxGJ4
                        7IglUx4G1QKTG/hx/5vxpFYgczJLAKpt/HEMfrInYVyP
                        qvCFpCyUudgA7kV8w05+BvBiK4IQAWZCiH8GcC3ERFoC
                        B/ZNE/f8YnMyi2sNqdN8Mhtkgz2vMZnACiLrIfnV7h1I
                        3uvqn0fQetnDWLF/xSeHoAE4WMZ6KwMgutUT1H9lQw0g
                        Ci5PemGKn0n2Va4ARzvQeTzBdLHZlVi04X25NHIEKZTj
                        8Sx1vRnBwPfT8qlYI1oa0ozuiLSyrII3Mw== )

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 20:58:05 -02 2018
;; MSG SIZE rcvd: 985
```

Excelente. E quanto à resolução reversa?

```
# dig @127.0.0.1 -p 8053 -x 10.0.42.2 +noadditional +noquestion

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 -p 8053 -x 10.0.42.2 +noadditional
+noquestion
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44932
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; ANSWER SECTION:
2.42.0.10.in-addr.arpa. 86400    IN      PTR     ns2.intnet.

;; AUTHORITY SECTION:
42.0.10.in-addr.arpa.  86400    IN      NS      ns1.intnet.
42.0.10.in-addr.arpa.  86400    IN      NS      ns2.intnet.

;; Query time: 0 msec
;; SERVER: 127.0.0.1#8053(127.0.0.1)
;; WHEN: Mon Nov 12 20:58:17 -02 2018
;; MSG SIZE  rcvd: 107
```

5. Agora, vamos configurar o **Unbound**. Como feito anteriormente, logue como **root** na máquina **ns1** e copie desta vez o arquivo **/etc/unbound/unbound.conf** usando o **scp**, como se segue:

```
# hostname ; whoami
ns1
root
```

```
# scp /etc/unbound/unbound.conf aluno@ns2:~
aluno@ns2's password:
unbound.conf
100% 820    2.4MB/s   00:00
```

O arquivo foi copiado para **/home/aluno/unbound.conf** na máquina **ns2**, como objetivado.

6. De volta a **ns2** como o usuário **root**, pare o *daemon* do Unbound e configure as chaves de controle:

```
# hostname ; whoami
ns2
root
```

```
# systemctl stop unbound
```

```
# unbound-control-setup
setup in directory /etc/unbound
unbound_server.key exists
unbound_control.key exists
create unbound_server.pem (self signed certificate)
create unbound_control.pem (signed client certificate)
Signature ok
subject=CN = unbound-control
Getting CA Private Key
Setup success. Certificates created.
```

Faça o backup do arquivo de configuração original:

```
# mv /etc/unbound/unbound.conf /etc/unbound/unbound.conf.orig
```

Copie o arquivo `/home/aluno/unbound.conf` para a pasta `/etc/unbound` e ajuste suas permissões:

```
# cp /home/aluno/unbound.conf /etc/unbound
```

```
# chown root. /etc/unbound/unbound.conf
```

```
# ls -ld /etc/unbound/unbound.conf
-rw-r--r-- 1 root root 820 nov 12 22:15 /etc/unbound/unbound.conf
```

Edite o arquivo com as configurações do servidor secundário via `sed`, como se segue:

```
# sed -i 's/^( *interface: 10\.0\.42\.)\1/\12/' /etc/unbound/unbound.conf
```

```
# sed -i '/^ *interface: 192.*\d/' /etc/unbound/unbound.conf
```

Use o comando `diff` para verificar as diferenças entre o arquivo original e as alterações feitas com o `sed`:

```
# diff -u /home/aluno/unbound.conf /etc/unbound/unbound.conf
--- /home/aluno/unbound.conf      2018-11-12 22:13:47.276000000 -0200
+++ /etc/unbound/unbound.conf     2018-11-12 22:24:48.564000000 -0200
@@ -1,7 +1,6 @@
server:
  interface: 127.0.0.1
- interface: 10.0.42.1
- interface: 192.168.42.1
+ interface: 10.0.42.2
  port: 53

  access-control: 127.0.0.0/8 allow
```

Remova o arquivo de configuração do Unbound no diretório `/home/aluno`:

```
# rm /home/aluno/unbound.conf
```

7. Vamos proceder aos testes. Inicie o Unbound usando o comando `unbound-control`:

```
# unbound-control start
```

Teste uma resolução direta qualquer no servidor Unbound local, solicitando DNSSEC:

```
# dig @127.0.0.1 mx1.intnet +dnssec +multiline +noquestion +noadditional

; <<>> DiG 9.10.3-P4-Debian <<>> @127.0.0.1 mx1.intnet +dnssec +multiline
+noquestion +noadditional
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53007
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; ANSWER SECTION:
mx1.intnet.          86388 IN A 10.0.42.91
mx1.intnet.          86388 IN RRSIG A 10 2 86400 (
    20181210223558 20181112223558 14936 intnet.
    YPluG8EXq3WwLKILMpeXc+C1C3AZhC+P1ej+6I5ax9Aw
    xQ+Zp6lwMXz64e7nPN2mjNa1PcAVuUNc8gE77U5dohhI
    Al++b4+1mPHLzN6EY4UuSsWqkE5NacSaUngmhV52Mrwj
    cwXtmJgJm9vhSTLKnUSYhSwFWZlC2FxI96kTQhtzJuDd
    KI7SVpYsNZPGiKwj5r9F8Wr/sKpCc0PgVnseewyPmNGY
    5pJEV30x6XgzQk0qjato04IxSj3CqX6ghU+MykcU2L+n
    56LfqqRv+R6TzzVst9/bFC7UgSXjF4v9eDKvPc86mcGn
    J9giG3w+07D2+jv7Ap6RfQWnx1zBy5CWFA== )

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Nov 12 22:45:11 -02 2018
;; MSG SIZE rcvd: 349
```

Sessão 3: Autenticação centralizada

Retomando o cenário apresentado na introdução da primeira sessão, num ambiente em que a virtualização é usada em larga escala, teremos diversas máquinas virtuais operando cada qual com seus serviços alocados. Imagine, hipoteticamente, que temos centenas de VMs dentro do *datacenter*. Vários desafios podem surgir à mente, mas tente responder a seguinte pergunta: com relação à gestão de contas, o que fazer quando um novo colaborador é integrado à equipe? Ou, por outro lado, quando um funcionário é desligado da empresa?

Ora, se temos centenas de VMs, é fácil supor que teremos que logar nas diferentes máquinas que novo colaborador (ou o antigo) deverá acessar e criar uma conta de usuário para ele — além disso, adicioná-lo a grupos e editar permissões relevantes. Fazer esse procedimento inúmeras vezes é claramente um processo que pode gerar erros de configuração, então poderia-se pensar em automatizá-lo, digamos, via *shell scripts*. Uma boa solução, mas não a ideal neste caso.

Sistemas de autenticação centralizados, como NIS (*Network Information Service*) ou LDAP (*Lightweight Directory Access Protocol*) são excelentes ferramentas para facilitar a gestão em cenários como o apresentado — adicionando o usuário em um único ponto, é possível distribuir essa configuração para dezenas, ou centenas, de máquinas de forma instantânea. O gerenciamento de grupos no sistema centralizado também permite atribuir permissionamento de forma fácil, ou removê-lo quando necessário.

Nesta sessão, iremos configurar um sistema de autenticação centralizado para o nosso laboratório usando LDAP, no qual gerenciaremos usuários e grupos, e faremos a integração desse sistema de autenticação com o Linux através do PAM (*Pluggable Authentication Modules*). Em lugar de fazer o controle de senhas dos usuários diretamente via `/etc/shadow` ou no LDAP, criaremos um sistema de autoridade certificadora (*Certificate Authority*) para o SSH, com o qual os usuários farão login nos servidores usando chaves assimétricas assinadas pela CA. Finalmente, para controlar ataques de força-bruta contra os servidores, usaremos o programa Fail2Ban para realizar o bloqueio automático de atacantes no firewall de host das máquinas.

1) Criação da VM para o servidor LDAP

1. Clone a máquina `debian-template` seguindo os mesmos passos da atividade (6) da sessão 1. Para o nome da máquina, escolha `ldap`.
2. Após a clonagem, na janela principal do Virtualbox, clique com o botão direito sobre a VM `ldap` e depois em *Settings*.

Em *Network > Adapter 1 > Attached to*, escolha *Host-only Adapter*. O nome da rede *host-only* deve ser o mesmo alocado para a interface de rede da máquina virtual `fw`, configurada durante a sessão 2, que está conectada à DMZ.

Clique em *OK*, e ligue a máquina `ldap`.

3. Logue como o usuário `root` e usando o script `/root/scripts/changehost.sh` que criamos anteriormente, renomeie a máquina:

```
# hostname  
debian-template
```

```
# bash ~/scripts/changehost.sh ldap
```

```
# hostname  
ldap
```

4. Em seguida, edite o arquivo `/etc/network/interfaces` como se segue, reinicie a rede e verifique o funcionamento:

```
# nano /etc/network/interfaces  
(...)
```

```
# cat /etc/network/interfaces  
source /etc/network/interfaces.d/*  
  
auto lo enp0s3  
  
iface lo inet loopback  
  
iface enp0s3 inet static  
address 10.0.42.2/24  
gateway 10.0.42.1
```

```
# systemctl restart networking
```

```
# ip a s | grep '^ *inet '  
    inet 127.0.0.1/8 scope host lo  
    inet 10.0.42.2/24 brd 10.0.42.255 scope global enp0s3
```

Verifique o roteamento:

```
# ip r s  
default via 10.0.42.1 dev enp0s3 onlink  
10.0.42.0/24 dev enp0s3 proto kernel scope link src 10.0.42.2
```

Verifique, ainda, a configuração de DNS do sistema:


```
# nano /etc/resolv.conf
(...)
```

```
# cat /etc/resolv.conf
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Finalmente, teste o funcionamento da conexão de rede:

```
# nc -zv obsd3.srv.ualberta.ca 80
obsd3.srv.ualberta.ca [129.128.5.194] 80 (http) open
```

2) Configuração do servidor LDAP

1. Vamos, primeiramente, instalar o servidor LDAP através dos pacotes:

```
# apt-get install slapd ldap-utils ldapscripts
```

Durante a instalação, será solicitada uma senha administrativa para o LDAP, que iremos redefinir em breve. Informe **rnpesr**.

2. Agora, vamos reconfigurar o servidor LDAP para definir alguns parâmetros que não foram questionados durante a instalação via **apt-get**. Execute:

```
# dpkg-reconfigure -plow slapd
```

Informe as seguintes opções:

Tabela 5. Configurações do pacote slapd

Pergunta	Opção
Omitir a configuração do servidor OpenLDAP?	Não
Nome de domínio DNS	intnet
Nome da organização	seg10
Senha do administrador	rnpesr (repetir)
Backend da base de dados a ser usado	MDB
Você deseja que a base de dados seja removida quando o pacote slapd for expurgado ("purged")	Não
Move a base de dados antiga	Sim

Durante a configuração, será criada uma base LDAP (semi) vazia, apenas com a raiz **dc=intnet** e

o usuário administrativo `cn=admin,dc=intnet`. Iremos popular esta base brevemente.

3. Agora, vamos instalar o `nslcd`, um *daemon* LDAP local para resolução de diretivas de autenticação. Execute:

```
# apt-get install nslcd
```

Durante a instalação do pacote, informe as seguintes opções:

Tabela 6. Configurações do pacote `nslcd`

Pergunta	Opção
URI do servidor LDAP	<code>ldapi:///</code>
Base de buscas do servidor LDAP	<code>dc=intnet</code>
Serviços de nome para configurar	<code>passwd, group, shadow</code>

Note que informamos que a localização do servidor LDAP é local, já que ele está instalado na máquina corrente. Além disso, iremos configurar as bases de contas (`passwd`), grupos (`group`) e senhas (`shadow`) junto ao LDAP.

4. Vamos configurar as opções padrão dos binários de linha de comando do `ldap-utils` (como os comandos `ldapsearch` e `ldapadd`, por exemplo). Edite manualmente ou use o `sed` para editar as linhas apropriadas do arquivo `/etc/ldap/ldap.conf`:

```
# sed -i 's/^#\ (BASE\).*\/\1 dc=intnet/' /etc/ldap/ldap.conf
```

```
# sed -i 's/^#\ (URI\).*\/\1 ldapi:\/\/\/\1/' /etc/ldap/ldap.conf
```

Verifique que os valores editados estão corretos:

```
# grep -v '^#' /etc/ldap/ldap.conf | sed '/^$/d'
BASE dc=intnet
URI ldapi:///
TLS_CACERT /etc/ssl/certs/ca-certificates.crt
```

5. A seguir, vamos configurar o `ldapscripts`, um conjunto de ferramentas auxiliares que facilitam enormemente a configuração e uso de bases LDAP via linha de comando. Primeiro, edite manualmente ou use o `sed` para ajustar o arquivo `/etc/ldapscripts/ldapscripts.conf`, informando o *bind DN* do usuário administrativo na base LDAP, como se segue:

```
# sed -i 's/^#\ (BINDDN=\\).*\/\1"cn=admin,dc=intnet\\"/'
/etc/ldapscripts/ldapscripts.conf
```

Depois, informe a senha do usuário administrativo no arquivo

`/etc/ldapscripts/ldapscripts.passwd`. Execute:

```
# echo -n "rnpesr" > /etc/ldapscripts/ldapscripts.passwd
```

Como a senha do usuário administrativo do LDAP está embutida em texto claro nesse arquivo, é fundamental garantir que suas permissões estão suficientemente estritas. Verifique:

```
# ls -ld /etc/ldapscripts/ldapscripts.passwd
-rw-r----- 1 root root 6 out 29 09:16 /etc/ldapscripts/ldapscripts.passwd
```

6. Vamos inicializar a base LDAP usando o `ldapscripts`. Execute:

```
# ldapinit -s
```

7. Será que funcionou? Consulte a base LDAP sem autenticação, e liste seu conteúdo:

```
# ldapsearch -x -LLL
dn: dc=intnet
objectClass: top
objectClass: dcObject
objectClass: organization
o: seg10
dc: intnet

dn: cn=admin,dc=intnet
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

dn: ou=People,dc=intnet
objectClass: top
objectClass: organizationalUnit
ou: People

dn: ou=Groups,dc=intnet
objectClass: top
objectClass: organizationalUnit
ou: Groups

dn: ou=Hosts,dc=intnet
objectClass: top
objectClass: organizationalUnit
ou: Hosts

dn: ou=Idmap,dc=intnet
objectClass: organizationalUnit
ou: Idmap
```

Perfeito! Temos configurada a raiz `dc=intnet` e usuário administrativo `cn=admin,dc=intnet` como anteriormente, e o comando `ldapinit` se encarregou de criar DN's para armazenar usuários, grupos, *hosts* e mapeamentos de identidade na base LDAP. Podemos prosseguir.

3) Habilitando logs do LDAP

Por padrão, o *daemon* `slapd` envia logs para a *facility* `local4` do sistema. Porém, após a instalação via `apt-get`, seu *log level* (nível de criticidade dos eventos registrados) é configurado para `none` — ou seja, nenhum evento é registrado. Evidentemente, essa situação não é interessante ao configurar o servidor, pois será muito difícil identificar as fontes de problemas se não tivermos nenhum log para consultar. Vamos corrigir isso.

1. Primeiro, crie um diretório específico para o armazenamento de LDIFs. LDIFs são uma sigla para *LDAP Data Interchange Format*, arquivos de texto plano que podem ser usados para representar informações em uma base LDAP. Algo equivalente a arquivos *dump* de bases SQL,

em comparação livre.

```
# mkdir /root/ldif
```

2. Crie neste diretório um LDIF de nome `/root/ldif/slapdlog.ldif`, com o seguinte conteúdo:

```
1 dn: cn=config
2 changeType: modify
3 replace: olcLogLevel
4 olcLogLevel: stats
```

O LDIF acima irá alterar o valor do parâmetro `olcLogLevel`, que define o *log level* do `slapd`, para `stats`. Para aplicar essa configuração, execute:

```
# ldapmodify -Y external -H ldapi:/// -f /root/ldif/slapdlog.ldif
```

3. O próximo passo é informar ao `rsyslog` que as mensagens enviadas para a *facility* `local4` devem ser enviadas para um arquivo específico. Crie o arquivo novo `/etc/rsyslog.d/slapd.conf` com o seguinte conteúdo:

```
1 $template slapdtmp1,"[%$DAY%-%$MONTH%-%$YEAR% %timegenerated:12:19:date-rfc3339%]
  %app-name% %syslogseverity-text% %msg%\n"
2 local4.* /var/log/slapd.log;slapdtmp1
```

4. Em seguida, reinicie ambos os *daemons* do `rsyslog` e do `slapd`:

```
# systemctl restart rsyslog.service
```

```
# systemctl restart slapd.service
```

5. Verifique que os registros de eventos do `slapd` estão sendo, de fato, enviados para o arquivo `/var/log/slapd.log`:

```
# tail /var/log/slapd.log
[19-10-2018 18:25:46] slapd debug daemon: shutdown requested and initiated.
[19-10-2018 18:25:46] slapd debug slapd shutdown: waiting for 0 operations/tasks
to finish
[19-10-2018 18:25:46] slapd debug slapd stopped.
[19-10-2018 18:25:46] slapd debug @(#) $OpenLDAP: slapd (May 23 2018 04:25:19)
$#012#011Debian OpenLDAP Maintainers <pkg-openldap-devel@lists.alioth.debian.org>
[19-10-2018 18:25:46] slapd debug slapd starting
```

6. A rotação de logs deve ser habilitada, caso contrário os arquivos de log poderão ficar excessivamente grandes. Crie o arquivo novo `/etc/logrotate.d/slapd`, com o seguinte conteúdo:

```
1 /var/log/slapd.log {
2     missingok
3     notifempty
4     compress
5     daily
6     rotate 30
7     sharedscripts
8     postrotate
9         systemctl restart rsyslog.service
10    endscrip
11 }
```

Como o `logrotate` é invocado via `cron`, não é necessário reiniciar nenhum serviço neste caso.

4) Edição de índices e permissões no LDAP

1. Para maior performance durante as consultas ao LDAP, é recomendável aumentar os parâmetros de indexação de alguns atributos. Crie o arquivo `/root/ldif/olcDbIndex.ldif`, com o seguinte conteúdo:

```
1 dn: olcDatabase={1}mdb,cn=config
2 changetype: modify
3 replace: olcDbIndex
4 olcDbIndex: objectClass eq
5 olcDbIndex: cn pres,sub,eq
6 olcDbIndex: sn pres,sub,eq
7 olcDbIndex: uid pres,sub,eq
8 olcDbIndex: displayName pres,sub,eq
9 olcDbIndex: default sub
10 olcDbIndex: uidNumber eq
11 olcDbIndex: gidNumber eq
12 olcDbIndex: mail,givenName eq,subinitial
13 olcDbIndex: dc eq
```

O LDIF acima irá alterar o valor do parâmetro `olcDbIndex`, que define quais parâmetros serão indexados pelo `slapd` e quais tipos de busca serão suportados. Para aplicar essa configuração, execute:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f /root/ldif/olcDbIndex.ldif
```

2. É interessante permitir que usuários possam alterar seus parâmetros `loginShell` e entrada `gecos` (informações de conta do usuário) via comandos `chsh` e `chfn`. Para fazer isso, crie o arquivo novo `/root/ldif/olcAccess.ldif` com o seguinte conteúdo:

```
1 dn: olcDatabase={1}mdb,cn=config
2 changetype: modify
3 add: olcAccess
4 olcAccess: {1}to attrs=loginShell,gecos
5   by dn="cn=admin,dc=intnet" write
6   by self write
7   by * read
```

Para aplicar a configuração, execute:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f /root/ldif/olcAccess.ldif
```

5) Adição de grupos e usuários no LDAP

Agora sim, vamos começar a criar usuários e grupos em nossa base LDAP. Imagine que iremos criar um grupo de nome **sysadm**, no qual estarão todos os administradores de sistema que têm permissão para acessar servidores não-críticos. Nesse grupo, iremos criar o usuário **luke**, com senha **seg10luke**. Como proceder?

1. Primeiro, crie o grupo usando o comando **ldapaddgroup**:

```
# ldapaddgroup sysadm
Successfully added group sysadm to LDAP
```

Verifique que o grupo foi corretamente criado usando o **ldapsearch**:

```
# ldapsearch -x -LLL 'cn=sysadm'
dn: cn=sysadm,ou=Groups,dc=intnet
objectClass: posixGroup
cn: sysadm
gidNumber: 10000
description: Group account
```

2. A seguir, crie o usuário **luke**, informando seu grupo primário como **sysadm**:

```
# ldapadduser luke sysadm
Successfully added user luke to LDAP
Successfully set password for user luke
```

Novamente, consulte o **ldapsearch** para checar se o usuário foi criado com sucesso:

```
# ldapsearch -x -LLL 'cn=luke'
dn: uid=luke,ou=People,dc=intnet
objectClass: account
objectClass: posixAccount
cn: luke
uid: luke
uidNumber: 10000
gidNumber: 10000
homeDirectory: /home/luke
loginShell: /bin/bash
gecos: luke
description: User account
```

O comando **ldapid** também é uma boa opção neste cenário, fornecendo saída bastante similar à do comando **id**:

```
# ldapid luke
uid=10000(luke) gid=10000(sysadm) groups=10000(sysadm),10000(sysadm)
```

3. Vamos configurar a senha do usuário **luke** como **seg10luke**:

```
# ldapsetpasswd luke
Changing password for user uid=luke,ou=People,dc=intnet
New Password:
Retype New Password:
Successfully set password for user uid=luke,ou=People,dc=intnet
```

Para verificar, cheque que o campo **userPassword** do usuário está preenchido com um *hash* de senha. Para o comando **ldapsearch** funcionar, temos que informar um *bind DN* administrativo e senha correspondente, já que este atributo não é legível sem autenticação:

```
# ldapsearch -x -LLL -D 'cn=admin,dc=intnet' -W 'cn=luke' userPassword
Enter LDAP Password:
dn: uid=luke,ou=People,dc=intnet
userPassword:: e1NTSEF9NHdUSWZRcUhGR0o5VU5jNS9tVnhoaGJzNFVvNkFzMmE=
```

O comando **ldapfinger** também é uma boa opção para consultar as informações do usuário (e seu *hash* de senha), já que faz parte do **ldapscrip**t e utiliza as credenciais administrativas que configuramos anteriormente:


```
# ldapfinger luke
dn: uid=luke,ou=People,dc=intnet
objectClass: account
objectClass: posixAccount
cn: luke
uid: luke
uidNumber: 10000
gidNumber: 10000
homeDirectory: /home/luke
loginShell: /bin/bash
gecos: luke
description: User account
userPassword:: e1NTSEF9NEU0aFI2N3lCN6p1UHdXRHNHVEZYZTBEYm5YdGxDTUg=
```

4. Apesar de termos informado o grupo **sysadm** como grupo primário do usuário **luke**, do ponto de vista do grupo esse usuário ainda não o integra. O comando **ldapaddusertogroup** faz esse trabalho:

```
# ldapaddusertogroup luke sysadm
Successfully added user luke to group cn=sysadm,ou=Groups,dc=intnet
```

Para verificar o pertencimento, use o comando **ldapsearch**, consultando os atributos **memberUid**:

```
# ldapsearch -x -LLL 'cn=sysadm'
dn: cn=sysadm,ou=Groups,dc=intnet
objectClass: posixGroup
cn: sysadm
gidNumber: 10000
description: Group account
memberUid: luke
```

O **ldapgid**, parte da suíte **ldapscrip**ts, também é uma alternativa interessante:

```
# ldapgid sysadm
gid=10000(sysadm) users(primary)=10000(luke) users(secondary)=10000(luke)
```

5. Como já mencionado algumas vezes, o **ldapscrip**ts oferece um conjunto de programas que facilitam bastante as tarefas corriqueiras de manipulação da base LDAP via linha de comando. Apesar de termos trabalhado um bom número desses programas, listamos abaixo alguns outros que não foram utilizados. Consulte suas páginas de manual para mais informações sobre como operá-los:

- Deleção de entradas:
 - **/usr/sbin/ldapdeletgroup**
 - **/usr/sbin/ldapdeleteuser**

- `/usr/sbin/ldapdeleteuserfromgroup`
- Modificação de entradas:
 - `/usr/sbin/ldapmodifygroup`
 - `/usr/sbin/ldapmodifyuser`
- Renomear entradas:
 - `/usr/sbin/ldaprenamegroup`
 - `/usr/sbin/ldaprenameuser`
- Configurar grupo primário de um usuário:
 - `/usr/sbin/ldapsetprimarygroup`

6) Integração e teste do sistema de autenticação com LDAP

Agora, vamos integrar o sistema de autenticação do Linux com a base LDAP que configuramos anteriormente usando o PAM (*Pluggable Authentication Modules*). Felizmente, muito do trabalho de configuração já foi feito automaticamente pelos *scripts* de instalação do `apt-get` quando instalamos o pacote `nsldap`. Faltam apenas alguns poucos passos.

1. Quando um usuário do LDAP efetua login em uma máquina pela primeira vez, seu diretório *home* não existe, naturalmente. Vamos configurar o PAM para criar esse diretório automaticamente. Crie o arquivo novo `/usr/share/pam-configs/mkhomedir` com o seguinte conteúdo:

```
1 Name: Create home directory during login
2 Default: yes
3 Priority: 900
4 Session-Type: Additional
5 Session:
6         required          pam_mkhomedir.so umask=0022 skel=/etc/skel
```

Em seguida, execute:

```
# pam-auth-update
```

Durante a configuração do PAM, na pergunta "Perfis PAM para habilitar", mantenha todas as caixas marcadas e selecione OK.

Verifique que a configuração surtiu efeito pesquisando pelo termo `mkhomedir` nos arquivos de configuração do PAM, em `/etc/pam.d`:

```
# grep -ri mkhomedir /etc/pam.d
/etc/pam.d/common-session:session      required      pam_mkhomedir.so umask=0022
skel=/etc/skel
/etc/pam.d/common-session-noninteractive:session      required
pam_mkhomedir.so umask=0022 skel=/etc/skel
```

2. Reinicie os *daemons* **nsld** e **nscd** para que a *cache* de usuários, grupos e senhas do LDAP seja atualizada:

```
# systemctl restart nslcd.service
```

```
# systemctl restart nscd.service
```

3. Será que funcionou? Pesquise a lista de usuários do sistema e busque pelo usuário recém-criado **luke**:

```
# getent passwd | grep luke
luke:*:10000:10000:luke:/home/luke:/bin/bash
```

Excelente! E quando ao grupo **sysadm**?

```
# getent group | grep sysadm
sysadm:*:10000:luke
```

Finalmente, consulte se o usuário **luke** é reconhecido como membro de **sysadm** pelo sistema:

```
# groups luke
luke : sysadm
```

4. Vamos testar: tente logar via **ssh** na máquina local usando o usuário **luke**:

```
# ssh luke@localhost
luke@localhost's password:
Creating directory '/home/luke'.
```

Perfeito. Note que o diretório **/home/luke** foi criado automaticamente, como esperado. Faça as verificações pós-login de costume:

```
$ whoami
luke
```

```
$ pwd
/home/luke
```

```
$ id
uid=10000(luke) gid=10000(sysadm) grupos=10000(sysadm)
```

5. Falta testar se o usuário consegue alterar sua senha diretamente via console, sem necessidade de edição direta à base LDAP. Primeiro, verifique o *hash* da senha atual — note que iremos usar o próprio DN do usuário **luke** como *bind DN* durante a conexão LDAP, motivo pelo qual deve-se informar a senha desse usuário, e não do administrador:

```
$ ldapsearch -x -LLL -D 'uid=luke,ou=People,dc=intnet' -W 'uid=luke' userPassword
Enter LDAP Password:
dn: uid=luke,ou=People,dc=intnet
userPassword:: e1NTSEF9K29BcE55S3AwRU9XM0sreWVPeFNoZUJjdFhBbVJyVEg=
```

Use o comando **passwd** para alterar a senha para um outro valor qualquer:

```
$ passwd
(current) LDAP Password:
Nova senha:
Redigite a nova senha:
passwd: senha atualizada com sucesso
```

Verifique novamente o *hash* da senha do usuário **luke** e compare os dois valores:

```
$ ldapsearch -x -LLL -D 'uid=luke,ou=People,dc=intnet' -W 'uid=luke' userPassword
Enter LDAP Password:
dn: uid=luke,ou=People,dc=intnet
userPassword:: e1NTSEF9b0REb21VbWgvR0swMm9qaGJoZWJ2ZGtNYzFKTE1kazk=
```

Perfeito, os *hashes* são diferentes; ou seja, a alteração de senha via **passwd** funcionou normalmente. Retorne a senha do usuário **luke** ao valor anterior, para evitar confusões no futuro.

7) Configurando uma autoridade certificadora (CA) para o SSH

Até o momento, instalamos e configuramos uma base LDAP, e testamos sua integração com os sistemas de autenticação do sistema usando o **ssh**. Porém, a todo momento, tivemos que digitar as senhas dos usuários para nos autenticar — será que podemos fazer isso de uma forma mais segura?

Nesta atividade iremos configurar uma autoridade certificadora para o **ssh**, com a qual

assinaremos pares de chaves de *hosts* e de usuários. De posse dessas chaves, não será mais necessário informar senhas durante o login, tornando o processo significativamente mais seguro (desde que se tome cuidado para não perder a chave privada, como veremos).

1. Primeiro, verifique que você está logado como usuário **root** na máquina **ldap**:

```
# hostname ; whoami ; pwd
ldap
root
/root
```

2. Vamos adicionar um novo usuário à base LDAP, **sshca**, que será responsável por realizar os processos de assinaturas de chaves de *hosts* e usuários. Esse usuário irá pertencer ao grupo **setup**, um grupo especial para atividades de configuração de sistemas. Sua senha será **seg10sshca**.

Vamos por partes. Primeiro, crie o grupo:

```
# ldapaddgroup setup
Successfully added group setup to LDAP
```

Em seguida, o usuário:

```
# ldapadduser sshca setup
Successfully added user sshca to LDAP
Successfully set password for user sshca
```

Adicione o usuário ao grupo:

```
# ldapaddusertogroup sshca setup
Successfully added user sshca to group cn=setup,ou=Groups,dc=intnet
```

E, finalmente, configure a senha do usuário **sshca**:

```
# ldapsetpasswd sshca
Changing password for user uid=sshca,ou=People,dc=intnet
New Password:
Retype New Password:
Successfully set password for user uid=sshca,ou=People,dc=intnet
```

3. Faça login com o usuário recém-criado no sistema local:

```
# ssh sshca@localhost
sshca@localhost's password:
Creating directory '/home/sshca'.
```

```
$ whoami
sshca
```

```
$ pwd
/home/sshca
```

4. Vamos criar dois pares de chaves para a CA (*Certificate Authority*, ou autoridade certificadora) do **ssh**: uma para assinar chaves de *hosts*, denominada **server_ca**, e outra para assinar chaves de usuários, denominada **user_ca**. Iremos criar chaves RSA de 4096 bits, e devemos escolher uma senha bastante segura para a chave privada—já que, com ela, pode-se assinar quaisquer chaves **ssh** que autorizarão máquinas a se passarem por membros do nosso *datacenter* e usuários a logarem em qualquer servidor integrado.

Como estamos em um ambiente de laboratório, vamos escolher senhas um pouco mais inseguras para facilitar a execução das atividades. Para a chave **server_ca**, defina como senha **seg10_server_ca**, e para a chave **user_ca**, defina como senha **seg10_user_ca**.

Para criar a chave de assinatura de *hosts*, **server_ca**, execute:

```
$ ssh-keygen -f server_ca -t rsa -b 4096
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in server_ca.
Your public key has been saved in server_ca.pub.
The key fingerprint is:
SHA256:ToVgkUzfkVQsZwwdKavtkE4028U1tcaIKzsZcuSbXrg sshca@ldap
The key's randomart image is:
+---[RSA 4096]-----+
|      o=o .+Boo      |
|      .oo o+.B .      |
|      o oB + .      |
|      ..0 + +      |
|      oS= o o      |
|      ooX.=      |
|      Xo@.      |
|      . Xo.      |
|      .E.      |
+-----[SHA256]-----+
```

Verifique que o par de chaves foi criado com sucesso:

```
$ ls
server_ca  server_ca.pub
```

Para criar a chave de assinatura de usuários, `user_ca`, execute:

```
$ ssh-keygen -f user_ca -t rsa -b 4096
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in user_ca.
Your public key has been saved in user_ca.pub.
The key fingerprint is:
SHA256:AJ7kC3EAI+TXTNUJgywCUOSYUSK9A0fF4mBZu6tgnyQ sshca@ldap
The key's randomart image is:
+---[RSA 4096]-----+
|/BX+*.o+o .      |
|*8o0*= .o        |
|+====o.          |
|  +... .         |
| .. S            |
| .               |
|.E o             |
|o = .            |
|. o              |
+----[SHA256]-----+
```

Cheque que todos os quatro arquivos de chaves pública/privada foram criados:

```
$ ls
server_ca  server_ca.pub  user_ca  user_ca.pub
```

5. Dada a grande sensibilidade das chaves privadas das CAs nesse sistema de autenticação que estamos configurando, é fundamental garantir também que, além de terem uma senha forte configurada, suas permissões estejam corretamente ajustadas.

Verifique as permissões das chaves usando o comando `ls`:

```
$ ls -l *_ca*
-rw----- 1 sshca setup 1766 out 28 08:39 server_ca
-rw-r--r-- 1 sshca setup  392 out 28 08:39 server_ca.pub
-rw----- 1 sshca setup 1766 out 28 08:40 user_ca
-rw-r--r-- 1 sshca setup  392 out 28 08:40 user_ca.pub
```

8) Configurando a SSH-CA no servidor LDAP

1. Vamos configurar o servidor LDAP para interoperar com a CA `ssh` que configuramos na atividade anterior. Crie um novo arquivo, `/root/scripts/sshsign.sh` com o seguinte conteúdo:

```
1 #!/bin/bash
2
3 CA_user="sshca"
4 CA_addr="10.0.42.2"
5 SSH_OPTS="-o PreferredAuthentications=password -o PubkeyAuthentication=no"
6
7 # obter senha para sshpass
8 echo -n "(${CA_user}@${CA_addr}) Enter passphrase: "
9 read -s pass
10 echo
11
12 # escanear chave do host ssh-ca, se necessario
13 [ -d ~/.ssh ] || { mkdir ~/.ssh; chmod 700 ~/.ssh; }
14 if ! ssh-keygen -F ${CA_addr} 2>/dev/null 1>/dev/null; then
15     ssh-keyscan -t rsa -T 10 ${CA_addr} >> ~/.ssh/known_hosts
16 fi
17
18 # testar se senha correta
19 if ! sshpass -p "${pass}" ssh ${SSH_OPTS} ${CA_user}@${CA_addr} exit 0; then
20     echo "wrong password"
21     exit 1
22 fi
23
24 # iterar em todas as pubkeys SSH
25 for pkeypath in /etc/ssh/ssh_host_*.pub; do
26     pkeyname="$( echo "${pkeypath}" | awk -F '/' '{print $NF}' )"
27     certname="$( echo "${pkeyname}" | sed 's/\\(\\.pub$\\)/-cert\\1 /' )"
28
29     # copiar pubkey
30     sshpass -p "${pass}" \
31         scp ${SSH_OPTS} ${pkeypath} ${CA_user}@${CA_addr}:~
32
33     # assinar pubkey, validade [-5 min -> 3 anos]
34     identity="$(hostname --fqdn)"
35     principals="$(hostname),$(hostname --fqdn),$(hostname -I | tr ' ' ',' | sed
36 's/,,$//')"
37     echo -ne "\n(CA private key) "
38     sshpass -p "${pass}" \
39         ssh ${SSH_OPTS} ${CA_user}@${CA_addr} \
40             ssh-keygen -s server_ca \
41                 -I "${identity}" \
42                 -n "${principals}" \
43                 -V -5m:+1095d \
44                 -h \
45                 ${pkeyname}
```



```
45
46 # copiar pubkey assinada de volta
47 sshpass -p "${pass}" \
48   scp ${SSH_OPTS} ${CA_user}@${CA_addr}:${certname} /etc/ssh/
49
50 # remover temporarios do diretorio remoto
51 sshpass -p "${pass}" \
52   ssh ${SSH_OPTS} ${CA_user}@${CA_addr} \
53     rm ${pkeyname} ${certname}
54
55 # remover pubkey RSA antiga e configurar ssh para apresentar pubkey assinada
56 rm -f ${pkeypath} 2> /dev/null
57 echo "HostCertificate /etc/ssh/${certname}" >> /etc/ssh/sshd_config
58 done
59
60 # copiar pubkey da server_ca e configurar reconhecimento de chaves de host
  assinadas
61 echo "@cert-authority * $(sshpass -p "$pass" ssh ${SSH_OPTS} ${CA_user}@
  ${CA_addr} cat server_ca.pub)" > /etc/ssh/ssh_known_hosts
62
63 # copiar pubkey da user_ca e configurar reconhecimento de chaves de usuario
  assinadas
64 sshpass -p "${pass}" \
65   scp ${SSH_OPTS} ${CA_user}@${CA_addr}:~/user_ca.pub /etc/ssh/
66 echo "TrustedUserCAKeys /etc/ssh/user_ca.pub" >> /etc/ssh/sshd_config
67
68 systemctl restart sshd.service
```

Você deve estar se perguntando: o que esse *script* faz? Vamos ver:

1. (Linhas 3-4) Definimos o usuário `sshca` e o IP `10.0.42.2` (o servidor `ldap` local no qual estamos logados no momento) como a origem das chaves da CA que utilizaremos a seguir.
2. (Linhas 8-10) Solicitamos ao usuário a senha do usuário `sshca@10.0.42.2`, armazenando-a em memória na variável `$pass`. Como faremos vários logins ao longo do *script*, faz sentido que armazenemos esta senha para evitar ter que digitá-la múltiplas vezes.
3. (Linhas 13-16) Escaneamos a chave do servidor `10.0.42.2` e armazenamos no arquivo `~/.ssh/known_hosts`, evitando que o usuário tenha que confirmar que confia no *host* remoto antes de logar.
4. (Linhas 19-22) Testamos se a senha informada no passo (2) está correta usando o comando `sshpass` — instalaremos esta dependência a seguir.
5. (Linhas 25-27) Iteramos sobre todas as chaves públicas no diretório `/etc/ssh` (RSA, ECDSA e ED25519), extraindo *strings* para usar à frente.
6. (Linhas 30-31) Copiamos a chave pública do *host* sendo processadao pelo *loop* para o servidor da CA.
7. (Linhas 34-44) Assinamos a chave pública do *host* sendo processadao pelo *loop* usando a chave `server_ca`, com validade de 3 anos.
8. (Linhas 47-48) Copiamos a chave pública assinada sendo processadao pelo *loop* de volta para

a máquina local.

9. (Linhas 51-53) Removemos chaves pública não-assinada e assinada sendo processadao pelo *loop* da pasta do usuário `sshca` no servidor `10.0.42.2`, para evitar confusão em assinaturas futuras.
 10. (Linhas 56-57) Removemos a chave pública sendo processadao pelo *loop* não-assinada e mantemos apenas a assinada, configurando o servidor `ssh` para apresentá-la para clientes.
 11. (Linha 61) Configuramos a chave `server_ca.pub` como uma CA confiável para *hosts* remotos; a partir deste momento, quaisquer logins de cliente `ssh` da máquina local para *hosts* assinados não terão que confirmar relação de confiança antes de prosseguir.
 12. (Linhas 64-66) Copiamos a chave `user_ca.pub` e a configuramos como uma CA confiável para usuários; a partir deste momento, qualquer usuário que apresente uma chave assinada pela CA terá seu login autorizado sem necessitar digitação de senha.
 13. (Linha 68) Reiniciamos o servidor `ssh` para aplicar as configurações realizadas.
2. Vamos instalar o `sshpas`, dependência para que o *script* acima funcione corretamente. Logado como `root`:

```
# whoami  
root
```

Instale o pacote:

```
# apt-get install sshpas
```

3. Vamos configurar a máquina `ldap` para operar com a CA do `ssh`. Execute o *script* criado no passo (1). Na pergunta (`sshca@10.0.42.2`) Enter passphrase, responda a senha do usuário `sshca` na máquina `10.0.42.2` (que deve ser `seg10sshca`); para cada pergunta (CA private key) Enter passphrase, informe a senha da chave `server_ca` (que deve ser `seg10_server_ca`).

```
# bash ~/scripts/sshsign.sh  
(sshca@10.0.42.2) Enter passphrase:  
# 10.0.42.2:22 SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u4  
  
(CA private key) Enter passphrase: seg10_server_ca  
Signed host key ssh_host_ecdsa_key-cert.pub: id "ldap.intnet" serial 0 for  
ldap,ldap.intnet,10.0.42.2 valid from 2018-10-30T10:09:23 to 2021-10-29T10:14:23  
  
(CA private key) Enter passphrase: seg10_server_ca  
Signed host key ssh_host_ed25519_key-cert.pub: id "ldap.intnet" serial 0 for  
ldap,ldap.intnet,10.0.42.2 valid from 2018-10-30T10:09:26 to 2021-10-29T10:14:26  
  
(CA private key) Enter passphrase: seg10_server_ca  
Signed host key ssh_host_rsa_key-cert.pub: id "ldap.intnet" serial 0 for  
ldap,ldap.intnet,10.0.42.2 valid from 2018-10-30T10:09:28 to 2021-10-29T10:14:28
```

Note que não foi necessário passar quaisquer parâmetros de linha de comando para o *script*. Ele autodeteca o *hostname* e endereços da máquina local e os configura como *principals* (conjunto de endereços/hostnames válidos para uma determinada chave; linha 34 do *script*).

4. Vamos verificar que o *script* fez seu trabalho corretamente. Cheque as linhas finais do arquivo `/etc/ssh/sshd_config`:

```
# tail -n4 /etc/ssh/sshd_config
HostCertificate /etc/ssh/ssh_host_ecdsa_key-cert.pub
HostCertificate /etc/ssh/ssh_host_ed25519_key-cert.pub
HostCertificate /etc/ssh/ssh_host_rsa_key-cert.pub
TrustedUserCAKeys /etc/ssh/user_ca.pub
```

Verifique que apenas chaves públicas assinadas existem no diretório `/etc/ssh`:

```
# ls -l /etc/ssh/ssh_host_*.pub
/etc/ssh/ssh_host_ecdsa_key-cert.pub
/etc/ssh/ssh_host_ed25519_key-cert.pub
/etc/ssh/ssh_host_rsa_key-cert.pub
```

Compare o conteúdo dos arquivos de chave pública `/home/sshca/server_ca.pub` e de chave da CA confiável `/etc/ssh/ssh_known_hosts` — eles devem ser iguais:

```
# cat /etc/ssh/ssh_known_hosts
@cert-authority * ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCAQCbhWR/iPDscvZm9/RL2aiauZq3yGbbZMcoHM2eKCLabjfpk2VqEk8
rqr5cd02XsvnKL60AE69TG7FU1pY9jxKODYtdQAGEBgfycR8I8AcSZ0t1DEvvXxdA0j6rIcWm458swdGYy
SAQsjFBAxrcLqGBC/0i7JzKqdTi1r4GfuLD0Hf4T4CVkIYgWF3f40H3RvebbLS2eQHPfMX46uYCjT9xYjM8
M21ftMe5h/Jt/rio5pnN5S6LL0XuKHg0e+wBD0JLbjjXSZmaSiVDMxsYWcuJjLMZ+Ew1VSgRNayX9yyoN4f
PEAM7NOGeEerqwrk/ZSRzChLH1t0LhORDbybya8wA5+XJyGDDWYpkCMAyC1PRT3lAkR+WtJw6thJEF+mcY8
G6zzAyoAD+riy5P9pqlsu14Hj6A8HXwLGDlNT35kt/Al/4SJia8XE0zXnAHKDl0MJHBIwNsI6dG5BgIar/0
dd9tVNnhsY/gMR5kjk07Yn0giThZ+e0K6SIrugVn1mQwldSMWLZFkHiyb3Ko7zQSFWJMr13XK5aTkXxTn4
C7L3jaAGu+EgyQLANXzpN8HNNgift55s2NbbJWRcLFQ7LGQRKwMhSvDRIiZcf2o9WxHxGCV7TMq5LleBQJN
rchBDqEtJyyvrg78lfkRH+u0dWt5AxhU8xCg6ItCYMH+7FHPww== sshca@ldap
```

```
# cat /home/sshca/server_ca.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCAQCbhWR/iPDscvZm9/RL2aiauZq3yGbbZMcoHM2eKCLabjfpk2VqEk8
rqr5cd02XsvnKL60AE69TG7FU1pY9jxKODYtdQAGEBgfycR8I8AcSZ0t1DEvvXxdA0j6rIcWm458swdGYy
SAQsjFBAxrcLqGBC/0i7JzKqdTi1r4GfuLD0Hf4T4CVkIYgWF3f40H3RvebbLS2eQHPfMX46uYCjT9xYjM8
M21ftMe5h/Jt/rio5pnN5S6LL0XuKHg0e+wBD0JLbjjXSZmaSiVDMxsYWcuJjLMZ+Ew1VSgRNayX9yyoN4f
PEAM7NOGeEerqwrk/ZSRzChLH1t0LhORDbybya8wA5+XJyGDDWYpkCMAyC1PRT3lAkR+WtJw6thJEF+mcY8
G6zzAyoAD+riy5P9pqlsu14Hj6A8HXwLGDlNT35kt/Al/4SJia8XE0zXnAHKDl0MJHBIwNsI6dG5BgIar/0
dd9tVNnhsY/gMR5kjk07Yn0giThZ+e0K6SIrugVn1mQwldSMWLZFkHiyb3Ko7zQSFWJMr13XK5aTkXxTn4
C7L3jaAGu+EgyQLANXzpN8HNNgift55s2NbbJWRcLFQ7LGQRKwMhSvDRIiZcf2o9WxHxGCV7TMq5LleBQJN
rchBDqEtJyyvrg78lfkRH+u0dWt5AxhU8xCg6ItCYMH+7FHPww== sshca@ldap
```

Observer, finalmente, que os arquivos `/home/sshca/user_ca.pub` e `/etc/ssh/user_ca.pub` também devem ser idênticos:

```
# diff /home/sshca/user_ca.pub /etc/ssh/user_ca.pub
```



Note que tanto no caso de assinatura de chaves de *host* como de chaves de usuário, futuramente, estamos fazendo o acesso no sentido **máquina remota** → **servidor da CA**, que não é o ideal. Em produção, o correto seria tornar o acesso ao servidor da CA o mais controlado possível, e fazer as assinaturas de chaves apenas de forma local, ou com acessos no sentido **servidor da CA** → **máquina remota**.

9) Automatizando a assinatura de chaves SSH de usuários

1. Vamos agora fazer a segunda "perna" da configuração da CA `ssh` — assinar chaves de usuários. Na linha da atividade anterior, iremos usar um *script* para assinar chaves de usuários; porém, como o cenário de chaves de usuário é mais flexível que o de máquinas, iremos estabelecer algumas premissas para o funcionamento do *script*:

1. Deve-se estar logado com o mesmo nome do usuário com o qual se deseja assinar a chave.
2. Deve-se ter conectividade com o servidor `ldap`, no endereço `10.0.42.2`.
3. O nome de chave será fixo (`~/.ssh/id_rsa`).

Devido à primeira limitação, é conveniente que o *script* esteja localizado dentro da pasta do usuário — e, como se sabe, ao criar novos usuários o conteúdo da pasta `/etc/skel` é copiado para dentro de seu *home*. Assim, crie a pasta `/etc/skel/scripts`:

```
# mkdir /etc/skel/scripts
```

Dentro dela, crie o arquivo novo, `/etc/skel/scripts/sshsign_user.sh`, com o conteúdo que se segue:

```
1 #!/bin/bash
2
3 CA_user="sshca"
4 CA_addr="10.0.42.2"
5 SSH_OPTS="-o PreferredAuthentications=password -o PubkeyAuthentication=no"
6
7 # testar se chave ja foi assinada
8 if [ -f ~/.ssh/id_rsa-cert.pub ]; then
9     echo "key already signed"
10    exit 1
11 fi
12
```

```
13 # obter senha para sshpass
14 echo -n "(${CA_user}@${CA_addr}) Enter passphrase: "
15 read -s pass
16 echo
17
18 # escanear chave do host ssh-ca, se necessario
19 [ -d ~/.ssh ] || { mkdir ~/.ssh; chmod 700 ~/.ssh; }
20 if ! ssh-keygen -F ${CA_addr} 2>/dev/null 1>/dev/null; then
21     ssh-keyscan -t rsa -T 10 ${CA_addr} >> ~/.ssh/known_hosts
22 fi
23
24 # testar se senha correta
25 if ! sshpass -p "${pass}" ssh ${SSH_OPTS} ${CA_user}@${CA_addr} exit 0; then
26     echo "wrong password"
27     exit 1
28 fi
29
30 # gerar par de chaves RSA, se inexistentes
31 [ -f ~/.ssh/id_rsa.pub ] || ssh-keygen -f ~/.ssh/id_rsa -t rsa -b 4096 -N '' &>
/dev/null
32
33 # copiar pubkey RSA
34 sshpass -p "${pass}" \
35     scp ${SSH_OPTS} ~/.ssh/id_rsa.pub ${CA_user}@${CA_addr}:~
36
37 # assinar pubkey RSA, validade [-5 min -> 1 ano]
38 user="$( whoami )"
39 echo -ne "\n(CA private key) "
40 sshpass -p "${pass}" \
41     ssh ${SSH_OPTS} ${CA_user}@${CA_addr} \
42         ssh-keygen -s user_ca \
43         -I ${user} \
44         -n ${user} \
45         -V -5m:+1095d \
46         id_rsa.pub
47
48 # copiar pubkey assinada de volta
49 sshpass -p "${pass}" \
50     scp ${SSH_OPTS} ${CA_user}@${CA_addr}:~/id_rsa-cert.pub ~/.ssh/
51
52 # remover temporarios do diretorio remoto
53 sshpass -p "${pass}" \
54     ssh ${SSH_OPTS} ${CA_user}@${CA_addr} \
55         rm id_rsa.pub id_rsa-cert.pub
56
57 # copiar pubkey da server_ca e configurar reconhecimento de chaves de host
assinadas
58 echo "@cert-authority * $(sshpass -p "$pass" ssh ${SSH_OPTS} ${CA_user}@
${CA_addr} cat server_ca.pub)" >> ~/.ssh/known_hosts
59
60 # remover pubkey RSA antiga
```

```
61 rm -f ~/.ssh/id_rsa.pub 2> /dev/null
```

Como o *script* guarda grandes semelhanças com o anterior, iremos destacar apenas os pontos de divergência:

1. (Linhas 8-11) Testamos se a chave RSA do usuário já foi assinada anteriormente; se positivo, o programa se encerra.
 2. (Linha 31) Caso o usuário não possua um par de chaves criado, cria-se automaticamente um par RSA de 4096 bits, sem senha.
 3. (Linhas 38-46) A chave usada para assinar a chave pública do usuário desta vez é a `user_ca`. A validade é ajustada para um ano.
 4. (Linha 58) De forma similar ao que foi feito anteriormente, copia-se a chave `server_ca.pub` como uma CA confiável para *hosts* remotos; a partir deste momento, logins de cliente `ssh` deste usuário para *hosts* assinados não terão que confirmar relação de confiança antes de prosseguir.
2. Vamos testar o funcionamento do *script* com o usuário `luke`, então logue-se como este usuário:

```
$ whoami ; pwd
luke
/home/luke
```

Como o diretório `/home/luke` já existia antes de criarmos o *script* no `/etc/skel`, iremos refazer a mesma estrutura de diretórios. Crie a pasta `/home/luke/scripts`:

```
$ mkdir ~/scripts
```

E copie para dentro dela o *script* criado no passo (1) desta atividade:

```
$ cp /etc/skel/scripts/sshsign_user.sh ~/scripts
```

```
$ ls ~/scripts
sshsign_user.sh
```

3. Execute o *script*. Na pergunta (`sshca@10.0.42.2`) Enter `passphrase`, responda a senha do usuário `sshca` na máquina `10.0.42.2` (que deve ser `seg10sshca`); para a pergunta (CA private key) Enter `passphrase`, informe a senha da chave `user_ca` (que deve ser `seg10_user_ca`).

```
$ bash ~/scripts/sshsign_user.sh
(sshca@10.0.42.2) Enter passphrase:
# 10.0.42.2:22 SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u4

(CA private key) Enter passphrase: seg10_user_ca
Signed user key id_rsa-cert.pub: id "luke" serial 0 for luke valid from 2018-10-
29T09:03:00 to 2021-10-28T09:08:00
```

Note que, novamente, não foi necessário passar quaisquer parâmetros de linha de comando para o *script*. Ele utiliza o *username* do usuário informado pelo comando *whoami* como *principal* da chave.

4. Vamos verificar o funcionamento do *script*. Verifique que os arquivos de chave foram criados:

```
$ ls ~/.ssh/
id_rsa id_rsa-cert.pub known_hosts
```

Cheque o conteúdo do arquivo *~/.ssh/known_hosts*, que deve conter informações da CA *ssh* para chaves de *host* assinadas:

```
$ tail -n1 ~/.ssh/known_hosts
@cert-authority * ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCAQCbhWR/iPDscvZm9/RL2aiauZq3yGbbZMcoHM2eKCLabjfpk2VqEk8
rqr05cd02XsvnKL60AE69TG7FU1pY9jxKODYtgDQAGEBgfyC8R8I8AcSZ0t1DEvvXxdA0j6rIcWm458swdGYy
SAQsjFBAxrc1qGBC/0i7JzKqdTi1r4GfuLD0Hf4T4CVkIYgWF3f40H3RvebbLS2eQHPfMX46uYCjT9xYjM8
M21ftMe5h/Jt/rio5pnN5S6LL0XuKHg0e+wBD0JLbjjXSZmaSiVDMxsYWcuJjLMZ+Ew1VSgRNayX9yyoN4f
PEAM7NOGeEerqwrk/ZSRzChLH1t0LhORDbybya8wA5+XJyGDDWYpkCMAyC1PRT31AkR+WtJw6thJEF+mcY8
G6zzAyoAD+riy5P9pqLsu14Hj6A8HXwLGD1NT35kt/AL/4SJia8XE0zXnAHKD1oMJHBIwNsI6dG5Bg1ar/0
dd9tVNnhsY/gMR5kjk07Yn0gIiThZ+e0K6SIrugVn1mQwldSMWLZFKhiyb3Ko7zQSFWJMr13XK5aTkXxTn4
C7L3jaAGu+EgyQLANXzpN8HNNgift55s2NbbJWRcLFQ7LGQRKwMhSvDRIiZcf2o9WxHxGCV7TMq5LleBQJN
rchBDQEtJyyvrg78lfrRH+u0dWt5AxhU8xCg6ItCYMH+7FHPww== sshca@ldap
```

5. Agora sim, vamos testar. Tente logar na máquina local usando o endereço IP ou *hostname* (o endereço especial *localhost* não é registrado como um *principal* válido na chave de *host*).

```
$ ssh luke@ldap
Linux ldap 4.9.0-8-amd64 #1 SMP Debian 4.9.110-3+deb9u6 (2018-10-08) x86_64
Last login: Mon Oct 29 09:10:00 2018 from 127.0.0.1
luke@ldap:~$
```

Perfeito! Não tivemos que digitar a senha do usuário ou confirmar a relação de confiança com o servidor, como esperado.

6. Todas as características do sistema que queríamos testar estão funcionais. Claro que devemos levar em consideração que todos os testes foram feitos dentro da máquina *ldap* — não testamos o funcionamento de contas de usuário via LDAP na rede, ou a autenticação de login remoto

usando a CA do `ssh`.

Remova o diretório do usuário `luke` da máquina `ldap`. Vamos prosseguir com a configuração do *template* e de um cliente Linux para testar as funcionalidades.

```
# rm -rf /home/luke
```

10) Configurando o template para funcionar com LDAP/SSH-CA

1. Como mencionado anteriormente, iremos configurar a VM `debian-template` para funcionar com os sistemas de autenticação do LDAP e SSH-CA. Assim, todas as máquinas que forem derivadas futuramente desse *template* estarão automaticamente integradas com o sistema de autenticação do nosso *datacenter* hipotético.

Ligue a VM `debian-template` e faça login como o usuário `root`:

```
# hostname ; whoami
debian-template
root
```

2. Crie o arquivo novo `/root/scripts/sshsign.sh`, e cole dentro dele o conteúdo do *script* que discutimos no passo (1) da atividade (8).

```
# nano /root/scripts/sshsign.sh
(...)
```

```
# ls /root/scripts/
changehost.sh  sshsign.sh  syncdirs.sh
```

3. Agora, crie o diretório `/etc/skel/scripts`, e dentro dele crie o arquivo novo `/etc/skel/scripts/sshsign_user.sh`, com conteúdo idêntico ao do *script* que foi apresentado no passo (1) da atividade (9).

```
# mkdir /etc/skel/scripts
```

```
# nano /etc/skel/scripts/sshsign_user.sh
(...)
```



```
# ls /etc/skel/scripts/  
sshsign_user.sh
```

4. Instale as dependências para funcionamento dos *scripts* criados anteriormente e também para integração do sistema de autenticação PAM com o LDAP.

```
# apt-get install sshpass nslcd
```

Durante a instalação do pacote **nslcd**, informe as seguintes opções:

Tabela 7. Configurações do pacote **nslcd**

Pergunta	Opção
URI do servidor LDAP	ldap://10.0.42.2/
Base de buscas do servidor LDAP	dc=intnet
Serviços de nome para configurar	passwd, group, shadow

5. Assim como configurado antes, informe ao PAM que diretórios *home* inexistentes de usuários do LDAP devem ser criados automaticamente. Crie o arquivo novo **/usr/share/pam-configs/mkhomedir**, e cole dentro dele o conteúdo do arquivo discutido durante o passo (1) da atividade (6).

```
# nano /usr/share/pam-configs/mkhomedir  
(...)
```

```
# pam-auth-update
```

Durante a configuração do PAM, na pergunta "Perfis PAM para habilitar", mantenha todas as caixas marcadas e selecione OK.

6. Finalmente, vamos alterar o *script* **/root/scripts/changehost.sh**, criado durante a sessão (1) deste curso, para invocar automaticamente o *script* **/root/scripts/sshsign.sh** ao final e reiniciar os *daemons* do **nslcd** e **nscd**. Altere o conteúdo deste arquivo para:

```
1 #!/bin/bash
2
3 [ -z $1 ] && { echo "Usage: $0 NEWHOSTNAME"; exit 1; }
4
5 sed -i "s/debian-template/$1/g" /etc/hosts
6 sed -i "s/debian-template/$1/g" /etc/hostname
7
8 invoke-rc.d hostname.sh restart
9 invoke-rc.d networking force-reload
10 hostnamectl set-hostname $1
11
12 rm -f /etc/ssh/ssh_host_* 2> /dev/null
13 dpkg-reconfigure openssh-server &> /dev/null
14
15 bash /root/scripts/sshsign.sh
16
17 systemctl restart nslcd.service
18 systemctl restart nscd.service
```

Ao invocar este *script* após a criação de uma nova VM, iremos não apenas alterar seu *hostname* mas também integrá-la com o sistema de autenticação remota do LDAP e SSH-CA em um único comando, como veremos a seguir.

11) Configurando um cliente Linux

Nesta atividade iremos criar uma máquina cliente Linux para utilizarmos como ponto de partida para os logins *ssh* nos diferentes servidores que configuraremos durante este curso. Iremos integrá-la com os sistemas de autenticação do LDAP e SSH-CA, e testar login remoto na máquina *ldap*.

1. Clone a máquina *debian-template* seguindo os mesmos passos da atividade (6) da sessão 1. Para o nome da máquina, escolha *client*.
2. Após a clonagem, na janela principal do Virtualbox, clique com o botão direito sobre a VM *client* e depois em *Settings*.

Em *Network > Adapter 1 > Attached to*, escolha *Host-only Adapter*. O nome da rede *host-only* deve ser o mesmo alocado para a interface de rede da máquina virtual *fw*, configurada durante a sessão 2, que está conectada à Intranet.

Clique em *OK*, e ligue a máquina *client*.

3. Logue como o usuário *root*; o primeiro login irá demorar um pouco, pois o sistema irá tentar fazer o *lookup* de contas no servidor LDAP e não obterá sucesso, já que a rede ainda não está configurada.

Em seguida, edite o arquivo */etc/network/interfaces* como se segue, reinicie a rede e verifique o funcionamento:

```
# nano /etc/network/interfaces  
(...)
```

```
# cat /etc/network/interfaces  
source /etc/network/interfaces.d/*  
  
auto lo enp0s3  
  
iface lo inet loopback  
  
iface enp0s3 inet static  
address 192.168.42.2/24  
gateway 192.168.42.1
```

```
# systemctl restart networking
```

Teste a conectividade com o servidor LDAP:

```
# ping -c3 10.0.42.2  
PING 10.0.42.2 (10.0.42.2) 56(84) bytes of data.  
64 bytes from 10.0.42.2: icmp_seq=1 ttl=64 time=0.010 ms  
64 bytes from 10.0.42.2: icmp_seq=2 ttl=64 time=0.024 ms  
64 bytes from 10.0.42.2: icmp_seq=3 ttl=64 time=0.025 ms  
  
--- 10.0.42.2 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2052ms  
rtt min/avg/max/mdev = 0.010/0.019/0.025/0.008 ms
```

E também com um *host* na Internet:

```
# nc -zv obsd3.srv.ualberta.ca 80  
obsd3.srv.ualberta.ca [129.128.5.194] 80 (http) open
```

4. Usando o script `/root/scripts/changehost.sh` que criamos anteriormente, renomeie a máquina:

```
# hostname  
debian-template
```

```
# bash ~/scripts/changehost.sh client
(sshca@10.0.42.2) Enter passphrase:

(CA private key) Enter passphrase: seg10_server_ca
Signed host key ssh_host_ecdsa_key-cert.pub: id "client.intnet" serial 0 for
client,client.intnet,192.168.42.2 valid from 2018-10-30T10:28:23 to 2021-10-
29T10:33:23

(CA private key) Enter passphrase: seg10_server_ca
Signed host key ssh_host_ed25519_key-cert.pub: id "client.intnet" serial 0 for
client,client.intnet,192.168.42.2 valid from 2018-10-30T10:28:25 to 2021-10-
29T10:33:25

(CA private key) Enter passphrase: seg10_server_ca
Signed host key ssh_host_rsa_key-cert.pub: id "client.intnet" serial 0 for
client,client.intnet,192.168.42.2 valid from 2018-10-30T10:28:27 to 2021-10-
29T10:33:27
```

```
# hostname
client
```

5. Se tudo tiver funcionado corretamente, a máquina **client** já estará integrada aos sistemas de autenticação LDAP e SSH-CA. Faça login como o usuário **luke**:

```
$ hostname ; whoami ; pwd
client
luke
/home/luke
```

6. Vamos criar um par de chaves para esse usuário e assiná-las. Como o conteúdo do diretório *home* foi copiado diretamente do **/etc/skel**, temos à disposição o *script* para essa tarefa na pasta **~/scripts/sshsign_user.sh**; execute-o:

```
$ bash ~/scripts/sshsign_user.sh
(sshca@10.0.42.2) Enter passphrase:
# 10.0.42.2:22 SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u4

(CA private key) Enter passphrase: seg10_user_ca
Signed user key id_rsa-cert.pub: id "luke" serial 0 for luke valid from 2018-10-
30T10:30:59 to 2021-10-29T10:35:59
```

7. Vamos testar? Tente logar usando o usuário **luke** na máquina **ldap** (cujo endereço IP é o 10.0.42.2):

```
$ ssh luke@10.0.42.2
Linux ldap 4.9.0-8-amd64 #1 SMP Debian 4.9.110-3+deb9u6 (2018-10-08) x86_64
Last login: Tue Oct 30 10:27:01 2018 from 127.0.0.1
```

```
$ hostname ; whoami ; pwd
ldap
luke
/home/luke
```

Excelente! Conseguimos logar sem ter que confirmar a relação de confiança com o servidor e sem digitar senha, como esperado.

12) Configurando o firewall para funcionar com LDAP/SSH-CA

Suponha, agora, que queremos integrar o firewall no sistema de autenticação LDAP/SSH-CA, mas com um maior nível de restrição. Sendo uma máquina crítica, não podemos permitir que qualquer usuário faça login nessa máquina, sendo necessário implementar controles mais estritos.

O primeiro passo, naturalmente, é fazer a integração com os sistemas de autenticação. Vamos fazer isso:

1. Logue como usuário **root** na máquina **fw**:

```
# hostname ; whoami
fw
root
```

2. Instale as dependências para funcionamento dos *scripts* e integração do sistema de autenticação.

```
# apt-get install sshpass nslcd
```

Novamente, durante a instalação do pacote **nslcd**, informe as seguintes opções:

Tabela 8. Configurações do pacote **nslcd**

Pergunta	Opção
URI do servidor LDAP	ldap://10.0.42.2/
Base de buscas do servidor LDAP	dc=intnet
Serviços de nome para configurar	passwd, group, shadow

3. Configure a criação automática de diretórios, com o arquivo novo **/usr/share/pam-configs/mkhomeid**; cole dentro dele o mesmo conteúdo usado nas atividades anteriores.

```
# nano /usr/share/pam-configs/mkhomedir  
(...)
```

```
# pam-auth-update
```

Durante a configuração do PAM, na pergunta "Perfis PAM para habilitar", mantenha todas as caixas marcadas e selecione OK.

4. Crie o arquivo novo `/root/scripts/sshsign.sh` e cole o conteúdo do *script* de assinatura de chaves de *host* que utilizamos anteriormente:

```
# nano /root/scripts/sshsign.sh  
(...)
```

```
# ls /root/scripts/  
changehost.sh  sshsign.sh  syncdirs.sh
```

Execute-o:

```
# bash ~/scripts/sshsign.sh  
(sshca@10.0.42.2) Enter passphrase:  
# 10.0.42.2:22 SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u4  
  
(CA private key) Enter passphrase: seg10_server_ca  
Signed host key ssh_host_ecdsa_key-cert.pub: id "fw.intnet" serial 0 for  
fw,fw.intnet,192.168.29.105,10.0.42.1,192.168.42.1 valid from 2018-10-30T10:54:01  
to 2021-10-29T10:59:01  
  
(CA private key) Enter passphrase: seg10_server_ca  
Signed host key ssh_host_ed25519_key-cert.pub: id "fw.intnet" serial 0 for  
fw,fw.intnet,192.168.29.105,10.0.42.1,192.168.42.1 valid from 2018-10-30T10:54:03  
to 2021-10-29T10:59:03  
  
(CA private key) Enter passphrase: seg10_server_ca  
Signed host key ssh_host_rsa_key-cert.pub: id "fw.intnet" serial 0 for  
fw,fw.intnet,192.168.29.105,10.0.42.1,192.168.42.1 valid from 2018-10-30T10:54:04  
to 2021-10-29T10:59:04
```

13) Restringindo login por grupos e usuários

Agora sim, com a integração concluída, imagine o seguinte cenário: não queremos que usuários do grupo `sysadm`, do qual faz parte o usuário `luke`, possam logar no firewall. Essa permissão será dada apenas a membros do grupo `fwadm`, que criaremos a seguir. Um desses usuários é o colaborador `han`,

cuja senha será **seg10han**. Como configurar esse tipo de restrição?

1. Primeiro, vamos criar o grupo e usuário. Logue na máquina **ldap** como usuário **root**:

```
# hostname ; whoami
ldap
root
```

2. Crie o grupo:

```
# ldapaddgroup fwadm
Successfully added group fwadm to LDAP
```

Usuário:

```
# ldapadduser han fwadm
Successfully added user han to LDAP
Successfully set password for user han
```

Adicione o usuário ao grupo:

```
# ldapaddusertogroup han fwadm
Successfully added user han to group cn=fwadm,ou=Groups,dc=intnet
```

E, finalmente, configure a senha do usuário **han**:

```
# ldapsetpasswd han
Changing password for user uid=han,ou=People,dc=intnet
New Password:
Retype New Password:
Successfully set password for user uid=han,ou=People,dc=intnet
```

3. De volta ao firewall, como usuário **root**:

```
# hostname ; whoami
fw
root
```

Edite o arquivo **/etc/nslcd.conf**, configurando a opção **pam_authz_search**. Essa opção permite que sejam definidos filtros de busca para o **nslcd**, através dos quais podemos restringir que usuários e/ou grupos podem logar na máquina local. No caso, queremos que apenas membros do grupo **fwadm** possam logar, portanto adicionamos a seguinte linha ao final do arquivo:

```
# echo "pam_authz_search
(&(objectClass=posixGroup)(cn=fwadm)(memberUid=$username))" >> /etc/nslcd.conf
```

Podemos customizar o filtro acima para incluir apenas usuários específicos, ou mesmo DN's que possuam um atributo qualquer (por exemplo, e-mails com um determinado sufixo). Tome sempre cuidado para não filtrar todos os usuários disponíveis no LDAP acidentalmente — é importante, nesses casos, sempre manter uma conta local (como **aluno** ou **root**, no nosso caso específico) com acesso ao sistema.

Reinicie os serviços do **nslcd** e **nscd**:

```
# systemctl restart nslcd.service
```

```
# systemctl restart nscd.service
```

Verifique que o usuário **han** é visto como membro do grupo **fwadm**:

```
# groups han
han : fwadm
```

Ocasionalmente, reiniciar o **nscd** não é suficiente para que ele detecte novas alterações na base de usuários/grupos do LDAP. Nesse caso, podemos invalidar as *caches* das tabelas do **nscd** com o comando:

```
# nscd --invalidate TABLE
```

As tabelas disponíveis podem ser consultadas na página de manual do **nscd**, ou vistas diretamente dentro da pasta **/var/cache/nscd**:

```
# ls -1 /var/cache/nscd/
group
hosts
netgroup
passwd
services
```

Para invalidar todas as *caches* do **nscd**, podemos executar por exemplo:

```
# for table in `ls -1 /var/cache/nscd` ; do nscd --invalidate $table ; done
```

4. Vamos testar a efetividade do controle aplicado. Na máquina **client**, faça login como o usuário **luke**:


```
$ hostname ; whoami
client
luke
```

Tente logar via **ssh** na máquina **fw**, cujo endereço IP é o 10.0.42.1:

```
$ ssh luke@10.0.42.1
LDAP authorisation check failed
Authentication failed.
```

Como o usuário **luke** não pertence ao grupo **fwadm**, o acesso é negado. Observando o log de *debug* do **nsld**, podemos ver que a pesquisa com o filtro aplicado anteriormente não retorna resultados:

```
nsld: [95f874] <authz="luke"> DEBUG: trying pam_authz_search
"(&(objectClass=posixGroup)(cn=fwadm)(memberUid=luke))"
nsld: [95f874] <authz="luke"> DEBUG: myldap_search(base="dc=intnet",
filter="(&(objectClass=posixGroup)(cn=fwadm)(memberUid=luke))")
nsld: [95f874] <authz="luke"> DEBUG: ldap_result(): end of results (0 total)
nsld: [95f874] <authz="luke"> pam_authz_search
"(&(objectClass=posixGroup)(cn=fwadm)(memberUid=luke))" found no matches
```

5. Vamos fazer o mesmo procedimento com o usuário **han**. Logue-se como **han** na máquina **client**:

```
$ hostname ; whoami
client
han
```

Como é a primeira vez que estamos usando este usuário, gere um par de chaves assinadas para ele:

```
$ bash ~/scripts/sshsign_user.sh
(sshca@10.0.42.2) Enter passphrase:
# 10.0.42.2:22 SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u4

(CA private key) Enter passphrase: seg10_user_ca
Signed user key id_rsa-cert.pub: id "han" serial 0 for han valid from 2018-10-
30T11:27:43 to 2021-10-29T11:32:43
```

Tente logar via **ssh** na máquina **fw**, cujo endereço IP é o 10.0.42.1:

```
$ ssh han@10.0.42.1
Linux fw 4.9.0-8-amd64 #1 SMP Debian 4.9.110-3+deb9u6 (2018-10-08) x86_64
Last login: Mon Oct 29 11:49:38 2018 from 127.0.0.1
```

```
$ hostname ; whoami ; pwd
fw
han
/home/han
```

Observando o log de *debug* do *nsld*, podemos ver que a pesquisa com o filtro aplicado anteriormente retorna como resultado o grupo *cn=fwadm,ou=Groups,dc=intnet*:

```
nsld: [138641] <authz="han"> DEBUG: trying pam_authz_search
"(&(objectClass=posixGroup)(cn=fwadm)(memberUid=han))"
nsld: [138641] <authz="han"> DEBUG: myldap_search(base="dc=intnet",
filter="(&(objectClass=posixGroup)(cn=fwadm)(memberUid=han))")
nsld: [138641] <authz="han"> DEBUG: ldap_result(): cn=fwadm,ou=Groups,dc=intnet
nsld: [138641] <authz="han"> DEBUG: pam_authz_search found
"cn=fwadm,ou=Groups,dc=intnet"
```

14) Restringindo logins SSH apenas via chaves assimétricas

Apesar de o controle que aplicamos na atividade anterior ser interessante, ainda não resolvemos o problema completamente. Como é possível tentar login na máquina *fw* usando senha, é possível que um atacante tente login por força-bruta, adivinhando a senha do usuário *han*, até conseguir. Vamos resolver isso.

1. Primeiro, vamos constatar o problema. Logue na máquina *client* como o usuário *han*:

```
$ hostname ; whoami
client
han
```

Para evitar que o cliente *ssh* use nossa chave assinada, passe as opções abaixo para o comando. Em seguida, digite a senha correta do usuário *han*:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no han@10.0.42.1
han@10.0.42.1's password:
Linux fw 4.9.0-8-amd64 #1 SMP Debian 4.9.110-3+deb9u6 (2018-10-08) x86_64
Last login: Tue Oct 30 11:35:44 2018 from 192.168.42.2
```

```
$ hostname ; whoami
fw
han
```

Note que conseguimos fazer o login usando senha normalmente, sem usar a chave assinada pela CA.

2. Logue como **root** na máquina **fw**:

```
# hostname ; whoami
fw
root
```

Iremos aplicar o controle sobre a opção **PasswordAuthentication** do **sshd**, desativando-o. Assim, não será mais possível logar via senha, apenas via chaves assimétricas. Execute o comando abaixo:

```
# sed -i 's/^#\(\PasswordAuthentication\).*\/\1 no/' /etc/ssh/sshd_config
```

E reinicie o **sshd**:

```
# systemctl restart sshd.service
```

3. De volta à máquina **client**, como **han**:

```
$ hostname ; whoami
client
han
```

Tente novamente logar usando senha:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no han@10.0.42.1
Permission denied (publickey).
```

Note que a permissão foi negada, pois apenas o método **publickey** é aceito para autenticação. Remova as opções do **ssh** e tente novamente, desta vez usando chaves:

```
$ ssh han@10.0.42.1
Linux fw 4.9.0-8-amd64 #1 SMP Debian 4.9.110-3+deb9u6 (2018-10-08) x86_64
Last login: Tue Oct 30 11:36:09 2018 from 192.168.42.2
```

```
$ hostname ; whoami  
fw  
han
```

15) Bloqueando tentativas de brute force contra o SSH

Não podemos aplicar o mesmo tipo de controle que fizemos na máquina **fw** no servidor **ldap**—nosso *script* de assinatura de chaves de *host* e de usuário utiliza login via senha com o usuário **sshca** para operar. A alteração dos *scripts* para usarem chaves e a correspondente restrição a login usando senhas teria implicações muito negativas na segurança da máquina, neste momento. Vamos implementar um controle diferente, então: proteção contra ataques de força-bruta usando a ferramenta Fail2ban.

O Fail2ban opera através da análise de eventos de log (normalmente registrados no diretório **/var/log**) e seu processamento através de expressões regulares. Caso um evento "case" (ou seja, ocorra um *match*) com uma expressão regular configurada, o Fail2ban irá adicionar uma unidade ao contador de violações de um determinado *host* remoto. Se esse *host* ultrapassar o número de violações configuradas em um dado período, o Fail2ban irá então tomar alguma ação configurada pelo administrador (logar um evento nos logs, enviar um e-mail para o administrador ou até mesmo bloquear de forma automática o *host* no firewall local).

Várias expressões regulares já vêm pré-configuradas no Fail2ban, para as ferramentas mais populares (como o **sshd**, o servidor web Apache ou o servidor SMTP Postfix). Caso se deseje configurar expressões regulares para ferramentas customizadas, também é possível fazê-lo.

1. Logue como o usuário **root** na máquina **ldap**:

```
# hostname ; whoami  
ldap  
root
```

2. Instale a ferramenta **fail2ban**:

```
# apt-get install fail2ban
```

3. Note que, por padrão, apenas a *jail* **sshd** vem habilitada no Debian. Não teremos que fazer qualquer alteração nesse sentido, já que é justamente o serviço **ssh** que queremos proteger.

```
# cat /etc/fail2ban/jail.d/defaults-debian.conf  
[sshd]  
enabled = true
```

4. As opções padrão do Fail2ban ficam configuradas no arquivo **/etc/fail2ban/jail.conf**, seção **[DEFAULT]**. Em particular, temos interesse nas seguintes configurações:

- **findtime**: Intervalo em que o Fail2ban irá registrar violações de *hosts* remotos.
- **maxretry**: Número de violações máximo permitido dentro do período **findtime** definido acima. Caso este valor seja ultrapassado, o Fail2ban irá tomar a ação configurada pelo administrador.
- **bantime**: Período em que o *host* remoto será afetado pela ação configurada. Caso esta ação seja, por exemplo, um bloqueio no firewall local, o *host* ficará banido pelo tempo especificado aqui.

Os valores padrão para as variáveis acima são os que se seguem:

```
# cat /etc/fail2ban/jail.conf | sed -n -e '/^\[DEFAULT\]/,/^\[/p' | grep  
'^maxretry\|^bantime\|^findtime'  
bantime = 600  
findtime = 600  
maxretry = 5
```

5. Vamos configurar o seguinte cenário: caso um atacante seja detectado pelo Fail2ban com mais de 3 violações (**maxretry**) num período de dez minutos (**findtime**), então iremos bani-lo via regra no firewall local (ação **iptables-multiport**) por dez minutos (**bantime**). Como o **findtime** e o **bantime** padrão estão corretos, iremos apenas configurar as duas outras variáveis, como se segue:

```
# echo "maxretry = 3" >> /etc/fail2ban/jail.d/defaults-debian.conf
```

```
# echo "banaction = iptables-multiport" >> /etc/fail2ban/jail.d/defaults-debian.conf
```

O arquivo **/etc/fail2ban/jail.d/defaults-debian.conf** ficou assim, portanto:

```
# cat /etc/fail2ban/jail.d/defaults-debian.conf  
[sshd]  
enabled = true  
maxretry = 3  
banaction = iptables-multiport
```

6. Uma outra configuração necessária é comentar uma linha do arquivo **/etc/fail2ban/filter.d/sshd.conf** que contém uma expressão regular para detectar entradas no seguinte formato no arquivo **/var/log/auth.log**:

```
Oct 30 12:03:47 ldap sshd[6677]: pam_unix(sshd:auth): authentication failure;  
logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.42.2 user=sshca
```

Em sistemas com autenticação LDAP, como é o nosso caso, a linha acima é inserida em

tentativas de login mesmo em caso de sucesso, como reportado em <https://github.com/fail2ban/fail2ban/issues/106>. Para corrigir esse falso positivo, basta executar:

```
# sed -i 's/^\(.*pam_unix.*\)#1/' /etc/fail2ban/filter.d/sshd.conf
```

7. Reinicie o Fail2ban para aplicar as configurações que realizamos:

```
# systemctl restart fail2ban.service
```

8. O Fail2ban criará novas *chains* no firewall para inserção de regras de banimento, quando adequado. Observe que o firewall está, até este momento, sem regras de BLOCK ou REJECT:

```
# iptables -L -vn
Chain INPUT (policy ACCEPT 37 packets, 5021 bytes)
  pkts bytes target    prot opt in     out     source        destination
    26  1820 f2b-sshd  tcp  --  *      *        0.0.0.0/0      0.0.0.0/0
multiport dports 22

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source        destination

Chain OUTPUT (policy ACCEPT 13 packets, 1152 bytes)
  pkts bytes target    prot opt in     out     source        destination

Chain f2b-sshd (1 references)
  pkts bytes target    prot opt in     out     source        destination
    26  1820 RETURN   all  --  *      *        0.0.0.0/0      0.0.0.0/0
```

Agora, vamos fazer uma simulação de ataque ao **sshd**. Monitore o arquivo **/var/log/fail2ban.log**:

```
# tail -n5 -f /var/log/fail2ban.log
2018-10-30 12:13:28,292 fail2ban.filter      [6876]: INFO    Added logfile =
/var/log/auth.log
2018-10-30 12:13:28,293 fail2ban.actions   [6876]: INFO    Set banTime = 600
2018-10-30 12:13:28,293 fail2ban.filter   [6876]: INFO    Set maxlines = 10
2018-10-30 12:13:28,323 fail2ban.server   [6876]: INFO    Jail sshd is not a
JournalFilter instance 2018-10-30 12:13:28,327 fail2ban.jail      [6876]: INFO    Jail 'sshd' started
```

9. Logue na máquina **client** como o usuário **han**, por exemplo:

```
$ hostname ; whoami
client
han
```

Para disparar o filtro do Fail2ban, não poderemos usar o login via chaves assimétricas, que obterá sucesso. Faça login usando senha como mostrado no comando a seguir; digite senhas incorretas para ativar a detecção do Fail2ban:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no han@10.0.42.2
han@10.0.42.2's password:
Permission denied, please try again.
han@10.0.42.2's password:
Permission denied, please try again.
han@10.0.42.2's password:
Permission denied (publickey,password).
```

Tente logar novamente:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no han@10.0.42.2
ssh: connect to host 10.0.42.2 port 22: Connection refused
```

A máquina foi bloqueada, como esperado.

10. De volta à máquina **ldap**, como o usuário **root**:

```
# hostname ; whoami
ldap
root
```

Note que os eventos de senha incorreta foram registrados pelo Fail2ban, bem como o banimento:

```
2018-10-30 12:18:21,775 fail2ban.filter      [6876]: INFO    [sshd] Found
192.168.42.2
2018-10-30 12:18:27,825 fail2ban.filter      [6876]: INFO    [sshd] Found
192.168.42.2
2018-10-30 12:18:33,151 fail2ban.filter      [6876]: INFO    [sshd] Found
192.168.42.2
2018-10-30 12:18:33,892 fail2ban.actions     [6876]: NOTICE [sshd] Ban
192.168.42.2
```

Observe que a regra de REJECT foi inserida automaticamente pelo Fail2ban no firewall local:

```
# iptables -L f2b-sshd -vn
Chain f2b-sshd (1 references)
pkts bytes target      prot opt in      out     source        destination
  1    60 REJECT      all  --  *        *        192.168.42.2   0.0.0.0/0
reject-with icmp-port-unreachable
612 70528 RETURN     all  --  *        *        0.0.0.0/0      0.0.0.0/0
```

Para remover o banimento de um endereço IP antes que o tempo total do **bantime** tenha transcorrido, é possível usar o comando **fail2ban-client**, como mostrado a seguir:

```
# fail2ban-client set sshd unbanip 192.168.42.2
192.168.42.2
```

Note que a regra de firewall é apagada, como esperado:

```
# iptables -L f2b-sshd -vn
Chain f2b-sshd (1 references)
pkts bytes target      prot opt in      out     source        destination
395 25992 RETURN     all  --  *        *        0.0.0.0/0      0.0.0.0/0
```

11. De volta à máquina **client**, como **han**, podemos tentar o login via senha novamente — desta vez, digite a senha correta:

```
$ hostname ; whoami
client
han
```

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no han@10.0.42.2
han@10.0.42.2's password:
Linux ldap 4.9.0-8-amd64 #1 SMP Debian 4.9.110-3+deb9u6 (2018-10-08) x86_64
Last login: Tue Oct 30 11:46:50 2018 from 192.168.42.2
```

```
$ hostname ; whoami
ldap
han
```


Sessão 4: Controles de segurança

Estando configurado nosso sistema de autenticação centralizado, quais seriam os próximos passos para realizar o *hardening* do ambiente? Nesta sessão, iremos tratar de algumas configurações mais simples, num escopo particular, mas que somadas tornarão o *datacenter* muito mais resiliente contra ataques, além de mais funcional. Iremos verificar se as senhas escolhidas pelos usuários são de fato seguras, implementar *quotas* de disco em um servidor de arquivos Linux, permitir controle mais granular de permissões de arquivos através de ACLs (*Access Control Lists*), controle mais granular de autorização administrativa usando o comando `sudo` e, finalmente, registrar os comandos digitados pelos usuários em logs do sistema.

Vamos ao trabalho?

1) Requisitos de senha na base LDAP

Uma preocupação frequente dos analistas de segurança é quanto às senhas dos usuários: será que elas tem um tamanho apropriado, não utilizam palavras constantes em *wordlists*, contém caracteres especiais? Apesar de termos configurado o acesso aos nossos servidores usando chaves assimétricas via SSH-CA (e, no caso da máquina *fw*, aplicado restrição de acesso exclusivamente via chaves), não é interessante que nos despreocupemos totalmente da segurança de senhas dos usuários — afinal, os logins na máquina *ldap* ainda podem usar senhas, por exemplo.

Podemos utilizar o *policy overlay* do *slapd* (documentação em <https://www.openldap.org/doc/admin24/overlays.html> ou `man 5 slapo-ppolicy`) para implementar alguns controles no diretório LDAP para exigir aspectos mínimos de qualidade das senhas dos usuários, tais como:

- **pwdInHistory**: Histórico de senhas, mantém uma lista de senhas passadas que impede que o usuário as repita. O número de senhas mantidas em histórico é configurável.
- **pwdMaxAge**: Tempo máximo de validade da senha.
- **pwdMinAge**: Tempo mínimo de validade da senha, para evitar que o usuário circule pelo histórico rapidamente e apague o registro de uma senha que queira repetir.
- **pwdMinLength**: Tamanho mínimo da senha do usuário, em caracteres.
- **pwdMaxFailure**: Número máximo de tentativas de *bind* com senha incorreta antes que a conta do usuário seja travada.
- **pwdCheckQuality**: Define uma função externa para checagem de qualidade da senha do usuário — esta é uma extensão não-padrão da política de senhas do diretório LDAP, e não iremos configurá-la. O website <http://ltb-project.org/wiki/documentation/openldap-ppolicy-check-password> disponibiliza um software customizado que pode ser usado para implementar esse tipo de política.

1. Faça login como `root` na máquina *ldap*:

```
# hostname ; whoami
ldap
root
```

Para habilitar esses controles em nossa base LDAP, o primeiro passo é carregar o arquivo LDIF do *schema* com as informações de políticas de senhas:

```
# ldapadd -Y external -H ldapi:/// -f /etc/ldap/schema/ppolicy.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
adding new entry "cn=ppolicy,cn=schema,cn=config"
```

2. Em seguida, iremos adicionar o módulo `/usr/lib/ldap/ppolicy.la` à lista de módulos carregados pelo `slapd` em seu início. Crie o arquivo novo `/root/ldif/olcModuleLoad.ldif` com o seguinte conteúdo:

```
1 dn: cn=module{0},cn=config
2 changetype: modify
3 add: olcModuleLoad
4 olcModuleLoad: ppolicy.la
```

Para aplicar as modificações desse LDIF à base LDAP, basta executar:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f ~/ldif/olcModuleLoad.ldif
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "cn=module{0},cn=config"
```

3. A próxima etapa é configurar o *overlay* de políticas de senhas para controlar os atributos `userPassword` de nossa base `cn=intnet`. Crie o arquivo novo `/root/ldif/olcOverlayPpolicy.ldif` com o seguinte conteúdo:

```
1 dn: olcOverlay=ppolicy,olcDatabase={1}mdb,cn=config
2 objectClass: olcOverlayConfig
3 objectClass: olcPPolicyConfig
4 olcOverlay: ppolicy
5 olcPPolicyDefault: cn=passwordDefault,ou=Policies,dc=intnet
6 olcPPolicyHashCleartext: FALSE
7 olcPPolicyUseLockout: FALSE
8 olcPPolicyForwardUpdates: FALSE
```

Note que estamos indicando que o *overlay* `ppolicy` será aplicado sobre a base `{1}mdb`, que é exatamente a base com raiz em `dc=intnet`, como podemos confirmar através do comando:

```
# ldapsearch -Y external -H ldapi:/// -LLL -b 'cn=config'  
'(&(objectClass=olcDatabaseConfig)(olcSuffix=dc=intnet))' dn 2> /dev/null  
dn: olcDatabase={1}mdb,cn=config
```

Caso estivéssemos fazendo esta configuração em um ambiente que possua várias bases LDAP carregadas dentro de um mesmo *daemon* **slapd**, seria necessário determinar o número da base MDB e editar o arquivo mostrado anteriormente.

Para aplicar as modificações desse LDIF à base LDAP, execute:

```
# ldapadd -Y EXTERNAL -H ldapi:/// -f ~/ldif/olcOverlayPpolicy.ldif  
SASL/EXTERNAL authentication started  
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth  
SASL SSF: 0  
adding new entry "olcOverlay=ppolicy,olcDatabase={1}mdb,cn=config"
```

4. Agora, vamos definir a política de senhas da base **dc=intnet**. Crie o arquivo novo **/root/ldif/passwordDefault.ldif** com o seguinte conteúdo:

```
1 dn: ou=Policies,dc=intnet  
2 ou: Policies  
3 objectClass: organizationalUnit  
4  
5 dn: cn=passwordDefault,ou=Policies,dc=intnet  
6 objectClass: pwdPolicy  
7 objectClass: person  
8 objectClass: top  
9 cn: passwordDefault  
10 sn: passwordDefault  
11 pwdAttribute: userPassword  
12 pwdCheckQuality: 2  
13 pwdMinAge: 0  
14 pwdMaxAge: 2592000  
15 pwdMinLength: 8  
16 pwdInHistory: 5  
17 pwdMaxFailure: 3  
18 pwdFailureCountInterval: 0  
19 pwdLockout: TRUE  
20 pwdLockoutDuration: 0  
21 pwdAllowUserChange: TRUE  
22 pwdExpireWarning: 0  
23 pwdGraceAuthNLimit: 0  
24 pwdMustChange: FALSE  
25 pwdSafeModify: FALSE
```

Estamos, em ordem:

- Criando uma entrada `ou=Políciés,dc=intnet` para armazenar políticas da base `dc=intnet`.
- Dentro desta OU, criando o CN `cn=passwordDefault,ou=Políciés,dc=intnet` que define a política de senhas da base. Configurações mais relevantes:
 - `pwdAttribute` define o atributo que será verificado, que armazena senhas de usuários.
 - `pwdCheckQuality` ativa a checagem de qualidade de senhas; como não estamos habilitando nenhum módulo externo, apenas a checagem de comprimento será aplicada.
 - `pwdMinAge` define o tempo mínimo de validade de senhas; como queremos testar o histórico de senhas, explicado a seguir, não iremos ativar essa opção.
 - `pwdMaxAge` define o tempo máximo de validade da senha, em segundos; ajustamos esse valor para 30 dias.
 - `pwdMinLength` define o tamanho mínimo de senha, 8 caracteres.
 - `pwdInHistory` define que iremos guardar o *hash* das 5 senhas mais recentes de cada usuário, que não poderão repeti-las.
 - `pwdMaxFailure` define que usuários que errarem a senha consecutivamente mais de 3 vezes terão suas contas bloqueadas.

Para aplicar o LDIF à base LDAP temos que nos autenticar na raiz `dc=intnet`, como se segue:

```
# ldapadd -D 'cn=admin,dc=intnet' -W -f ~/ldif/passwordDefault.ldif
Enter LDAP Password:
adding new entry "ou=Políciés,dc=intnet"

adding new entry "cn=passwordDefault,ou=Políciés,dc=intnet"
```

5. Reinicie o `slapd` para aplicar as configurações:

```
# systemctl restart slapd.service
```

6. Vamos testar nossos controles — logue na máquina `client` como o usuário `luke`:

```
$ hostname ; whoami
client
luke
```

Tente alterar a senha do usuário para uma *string* menor que o tamanho exigido, como `mar-te` por exemplo:

```
$ passwd
(current) LDAP Password:
Nova senha:
Redigite a nova senha:
password change failed: Password fails quality checking policy
passwd : Erro de manipulação de token de autenticação
passwd: senha inalterada
```

O **slapd** nos informa que a senha não atende os requisitos mínimos de qualidade, nesse caso, o tamanho da senha.

7. Altere a senha para um valor aceitável, como **seg10luke2**, por exemplo:

```
$ passwd
(current) LDAP Password:
Nova senha:
Redigite a nova senha:
passwd: senha atualizada com sucesso
```

Agora, tente alterar a senha para um valor já usado anteriormente, como **seg10luke**:

```
$ passwd
(current) LDAP Password:
Nova senha:
Redigite a nova senha:
password change failed: Password is in history of old passwords
passwd : Erro de manipulação de token de autenticação
passwd: senha inalterada
```

Somos informados que a senha consta do histórico de senhas antigas. Como o LDAP implementa isso? Acesse a máquina **ldap** como usuário **root** e pesquise pelo campo **pwdHistory** do usuário **luke**:

```
# ldapsearch -LLL -D 'cn=admin,dc=intnet' -W 'uid=luke' pwdHistory
Enter LDAP Password:
dn: uid=luke,ou=People,dc=intnet
pwdHistory: 20181031133744Z#1.3.6.1.4.1.1466.115.121.1.40#38#{SSHA}OEKBo+ZPtqc
hHy1T3sKU8hk+Eb02kG4
pwdHistory: 20181031133855Z#1.3.6.1.4.1.1466.115.121.1.40#38#{SSHA}0nWgPyL1A6T
ukTeA6or6in1zTzug9w4
pwdHistory: 20181031133959Z#1.3.6.1.4.1.1466.115.121.1.40#38#{SSHA}ndljhPMAUpU
mrGEqy/lPvYeNVLgfDbxo
```

Ao informarmos uma nova senha, o **slapd** compara o seu hash com um dos *hashes* guardados no histórico do usuário (nesse caso, **luke**); se encontrada, a senha é rejeitada.

8. Vamos testar o *lockout* de contas. Como teremos que fazer logins propositalmente incorretos, pare o serviço Fail2ban na máquina **ldap** para evitar que sejamos bloqueados pelo firewall durante o teste:

```
# hostname ; whoami
ldap
root
```

```
# systemctl stop fail2ban
```

De volta à máquina **client** como **luke**, tente logar via SSH usando senha e erre propositalmente a combinação por 3 vezes consecutivas:

```
$ hostname ; whoami
client
luke
```

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no luke@10.0.42.2
luke@10.0.42.2's password:
Permission denied, please try again.
luke@10.0.42.2's password:
Permission denied, please try again.
luke@10.0.42.2's password:
Permission denied (publickey,password).
```

Agora, tente logar com a senha correta — note que seu acesso será negado:

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o
PubkeyAuthentication=no luke@10.0.42.2
luke@10.0.42.2's password:
Permission denied, please try again.
```

De volta à máquina **ldap** como o usuário **root**, vamos verificar o que aconteceu:

```
# hostname ; whoami
ldap
root
```

Execute o comando **ldapsrch** abaixo para listar todos os usuários bloqueados na base **dc=intnet**:

```
# ldapsearch -LLL -D 'cn=admin,dc=intnet' -W 'pwdAccountLockedTime=*'  
pwdAccountLockedTime  
Enter LDAP Password:  
dn: uid=luke,ou=People,dc=intnet  
pwdAccountLockedTime: 20181031121725Z
```

Como esperado, **luke** está bloqueado. Para desbloquear um usuário específico crie um arquivo LDIF novo, **/root/ldif/unlockUser.ldif** com o seguinte conteúdo:

```
1 dn: uid=luke,ou=People,dc=intnet  
2 changetype: modify  
3 delete: pwdAccountLockedTime
```

Aplique as alterações do LDIF à base com:

```
# ldapmodify -D 'cn=admin,dc=intnet' -W -f ~/ldif/unlockUser.ldif  
Enter LDAP Password:  
modifying entry "uid=luke,ou=People,dc=intnet"
```

De volta à máquina **client** como **luke**, tente logar novamente com a senha correta:

```
$ hostname ; whoami  
client  
luke
```

```
$ ssh -o PreferredAuthentications=keyboard-interactive,password -o  
PubkeyAuthentication=no luke@10.0.42.2  
luke@10.0.42.2's password:  
Linux ldap 4.9.0-8-amd64 #1 SMP Debian 4.9.110-3+deb9u6 (2018-10-08) x86_64  
Last login: Wed Oct 31 09:17:04 2018 from 192.168.42.2
```

```
$ hostname ; whoami  
ldap  
luke
```

Perfeito, nossos controles funcionaram como esperado. Na máquina **ldap**, como **root**, não se esqueça de reiniciar o Fail2ban:

```
# hostname ; whoami  
ldap  
root
```

```
# systemctl start fail2ban
```

2) Busca de senhas fracas

Simplesmente configurar um tamanho mínimo de senha, como fizemos na atividade anterior, não é garantia que os usuários escolherão senhas seguras para suas contas. Por exemplo, um usuário pode definir **12345678** como sua senha — essa *string* está dentro do tamanho mínimo exigido mas não pode, nem de perto, ser considerada uma senha segura. O que fazer?

Podemos submeter os *hashes* de senha dos usuários a testes de segurança, como ataques de força-bruta — em que testamos combinações de caracteres exaustivamente para descobrir a senha — ou de dicionário — em que usamos uma base de senhas previamente preenchida, conhecida como *wordlist*, e verificamos se a senha do usuário se encontra nessa lista. Devido ao fato de as senhas do LDAP serem armazenadas por padrão em formato SSHA (SHA-1 com *salt*), ataques do tipo *rainbow table* — em que comparamos o *hash* da senha do usuário com uma base de *hashes* previamente computados, buscando por similaridades — não são viáveis. Podemos verificar o *hash* utilizado para armazenar a senha do usuário **luke**, por exemplo, usando o comando:

```
# ldapsearch -x -LLL -D 'cn=admin,dc=intnet' -W 'uid=luke' userPassword | grep  
'^userPassword::' | awk '{print $NF}' | base64 --decode  
Enter LDAP Password:  
{SSHA}JK1/uM/9bmoWM/IzW1uIBM4b1Q4UEWd8
```

A ferramenta que iremos utilizar para realizar os ataques de dicionário e força-bruta mencionados anteriormente será o **hashcat** (<https://hashcat.net/hashcat/>). Uma das ferramentas mais rápidas para quebra de senhas disponíveis, é um programa *open source* multiplataforma que se utiliza da CPU ou GPUs (placas gráficas) de uma máquina para acelerar o processo de ataque sensivelmente, especialmente quando comparada com ferramentas mais tradicionais como o **john**.

Até a versão v3.00, o **hashcat** era dividido em duas versões, uma voltada para CPUs e outra para GPUs (esta, implementada via OpenCL ou CUDA). Com o lançamento da versão v3.00, as duas versões foram unificadas em uma única ferramenta, requerendo a biblioteca OpenCL (<https://www.khronos.org/opencl/>) como dependência.

É boa prática de segurança que instalemos apenas o estritamente necessário em servidores, a fim de reduzir a superfície de ataque disponível em uma eventual invasão. Por esse motivo, instalaremos o **hashcat** e as demais bibliotecas necessárias na máquina **client**, que é menos crítica que os servidores **ldap** e **fw**.

1. Acesse a máquina **client** como o usuário **root**, e instale o **hashcat** e suas dependências:

```
# hostname ; whoami  
client  
root
```



```
apt-get install --no-install-recommends hashcat libhwloc-dev ocl-icd-dev ocl-icd-  
opencl-dev pocl-opencl-icd
```

2. Agora, acesse como o usuário **luke**, em seu diretório *home*.

```
$ hostname ; whoami ; pwd  
client  
luke  
/home/luke
```

Altere a senha do usuário **luke** para um valor propositalmente inseguro, como **password**:

```
$ passwd  
(current) LDAP Password:  
Nova senha:  
Redigite a nova senha:  
passwd: senha atualizada com sucesso
```

O primeiro passo para testarmos a segurança das senhas dos usuários é obter seus *hashes*. Vamos fazer isso, de forma remota, usando um *script* shell mostrado a seguir. Crie o arquivo novo **/home/luke/scripts/gethashes.sh** com o seguinte conteúdo:

```
1 #!/bin/bash
2
3 TMPFILE="$( mktemp )"
4 OUTFILE="$HOME/hashtxt.txt"
5
6 rm -f ${OUTFILE}
7 touch ${OUTFILE}
8
9 ldapsearch -x \
10 -LLL \
11 -H ldap://10.0.42.2 \
12 -D 'cn=admin,dc=intnet' \
13 -w 'rnpesr' \
14 -b 'dc=intnet' \
15 'userPassword=*' \
16 cn=userPassword \
17 | grep '^cn:|^userPassword::' \
18 | awk '{print $NF}' \
19 | sed 'N;s/\n/ /' \
20 | tr ' ' ':' > ${TMPFILE}
21
22 while read l; do
23     luser="$( echo ${l} | cut -d':' -f1 )"
24     lhash="$( echo ${l} | cut -d':' -f2 )"
25
26     echo "${luser}:${( echo ${lhash} | base64 --decode )}" >> ${OUTFILE}
27 done < ${TMPFILE}
28
29 rm -f ${TMPFILE}
```

O que esse *script* faz? Vamos ver:

1. (Linhas 3-4) Criamos um arquivo temporário com o comando `mktemp`, e definimos o arquivo de saída como `~/hashtxt.txt`.
2. (Linhas 6-7) Se existente, removemos o arquivo de saída e criamos um novo, vazio.
3. (Linhas 9-20) Executamos um comando `ldapsearch` remoto na máquina `ldap`, executando o `bind` como o usuário `cn=admin,dc=intnet` e senha informada diretamente na linha de comando. Buscamos todos os DN's que possuem o campo `userPassword` não-vazio, e filtramos apenas os campos `cn` e `userPassword` na saída. Finalmente, fazemos uma junção de linhas duas-a-duas usando os comandos `awk`, `sed` e inserimos um separador usando o `tr`. Essa saída é escrita no arquivo temporário criado anteriormente.
4. (Linhas 22-27) Processamos o arquivo temporário linha-a-linha. Em cada linha, extraímos o campo 1 (`cn` do usuário) e o campo 2 (`userPassword`). O campo `userPassword` está codificado em base64, então usamos `base64 --decode` para traduzir esse campo, e escrevemos o *output* em ordem no arquivo de saída.
5. (Linha 29) O arquivo temporário é removido.

3. Vamos testar o funcionamento do *script*:

```
$ bash ~/scripts/gethashes.sh
```

```
$ cat hashes.txt
admin:{SSHA}NzQZTz7uf0xNM3PYy7cp+zV6p7bKFNcy
luke:{SSHA}46Qe8Ny+QQgDsbPcps2MODqUHGtdLX41
sshca:{SSHA}+JTtQ5+XEi+sJ4sPmWK3LZXrIHSpbcbn
han:{SSHA}BE6cC89vaJQtB/g9yEJTt008HCtRabel
```

4. Vamos, primeiramente, executar um ataque de dicionário. Um ataque de dicionário, como mencionado anteriormente, é quando obtermos um arquivo com um conjunto de senhas em texto claro, calculamos seus *hashes* usando os valores de *salt* conhecidos, e comparamos os resultados com os *hashes* dos usuários.

No caso do algoritmo SSHA implementado no OpenLDAP, para extrair o *salt* devemos decodificar o *hash* original em base64 uma vez, remover o prefixo **{SSHA}**, decodificar o *hash* resultante em base64 novamente, e extrair os últimos 4 bytes; esses 4 bytes são o *salt* da senha codificada. Para ilustrar esse conceito, o *script* Perl abaixo pode ser usado para fazer a extração:

```
1 #!/usr/bin/perl -w
2
3 my $hash=$ARGV[0];
4 # The hash is encoded as base64 twice:
5 use MIME::Base64;
6 $hash = decode_base64($hash);
7 $hash=~s/{SSHA}//;
8 $hash = decode_base64($hash);
9
10 # The salt length is four (the last four bytes).
11 $salt = substr($hash, -4);
12
13 # Split the salt into an array.
14 my @bytes = split(//,$salt);
15
16 # Convert each byte from binary to a human readable hexadecimal number.
17 foreach my $byte (@bytes) {
18     $byte = uc(unpack "H*", $byte);
19     print "$byte";
20 }
```

Vamos recuperar o *hash* de senha do usuário **luke**:

```
$ ldapsearch -x -LLL -H ldap://10.0.42.2 -D 'cn=admin,dc=intnet' -b 'dc=intnet' -w
'rnpesr' 'uid=luke' userPassword | grep '^userPassword::' | awk '{print $NF}'
e1NTSEF9NDZRZTh0eStRUWdEc2JQY3BzMk1PRHFVSEd0ZExYNDE=
```

Executando o *script* `getsalt.pl`, podemos extrair o *salt* da senha. Note que o valor de saída está em hexadecimal.

```
$ perl ~/scripts/getsalt.pl e1NTSEF9NDZRZTh0eStRUWdEc2JQY3BzMk1PRHFVSEd0ZExYNDE=5D2D7E35
```

5. De volta ao ataque de dicionário, vamos executá-lo usando o `hashcat`. Primeiro, temos que descobrir a qual código o *hash* SSHA do LDAP corresponde:

```
$ hashcat --help | grep SSHA
111 | nsldaps, SSHA-1(Base64), Netscape LDAP SSHA | HTTP, SMTP, LDAP
Server
1711 | SSHA-512(Base64), LDAP {SSHA512} | HTTP, SMTP, LDAP
Server
10300 | SAP CODVN H (PWDSALTEDHASH) iSSHA-1 | Enterprise Application
Software (EAS)
```

O código é, então, **111**. Quanto ao tipo de ataque:

```
$ hashcat --help | grep 'Attack Modes' -A8
- [ Attack Modes ] -

# | Mode
===+=====
0 | Straight
1 | Combination
3 | Brute-force
6 | Hybrid Wordlist + Mask
7 | Hybrid Mask + Wordlist
```

O ataque de dicionário, também conhecido como *straight mode* (https://hashcat.net/wiki/doku.php?id=dictionary_attack), possui código **0**.

Falta apenas obter uma *wordlist* apropriada para executar o ataque. Procurando por termos como "*wordlist*", "*password*" ou "*common*" no Google, é possível encontrar uma infinidade de páginas web dedicadas ao assunto, como por exemplo <https://github.com/danielmiessler/SecLists/tree/master/Passwords>. Iremos usar uma *wordlist* que alegadamente contém as 10 milhões de senhas mais comuns, que pode ser baixada na URL anteriormente mencionada ou solicitada ao instrutor. Note que, para um arquivo que contém apenas texto puro, seu tamanho é impressionante:

```
$ wget -q https://github.com/danielmiessler/SecLists/raw/master/Passwords/Common-Credentials/10-million-password-list-top-1000000.txt
```

```
$ du -sh 10-million-password-list-top-1000000.txt
8,2M    10-million-password-list-top-1000000.txt
```

Tudo pronto! Vamos executar o ataque:

```
$ hashcat --hash-type 111 --attack-mode 0 --username hashes.txt 10-million-
password-list-top-1000000.txt
hashcat (v3.30) starting...

(...)

Session.....: hashcat
Status.....: Exhausted
Hash.Type.....: SSHA-1(Base64), nsldaps, Netscape LDAP SSHA
Hash.Target.....: hashes.txt
Time.Started.....: Thu Nov  1 08:55:59 2018 (2 secs)
Time.Estimated...: Thu Nov  1 08:56:01 2018 (0 secs)
Input.Base.....: File (10-million-password-list-top-1000000.txt)
Input.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....: 2618.0 kH/s (0.36ms)
Recovered.....: 1/4 (25.00%) Digests, 1/4 (25.00%) Salts
Progress.....: 3999996/3999996 (100.00%)
Rejected.....: 36/3999996 (0.00%)
Restore.Point....: 999999/999999 (100.00%)
Candidates.#1....: vjq445 -> vjht008
HWMon.Dev.#1.....: N/A

Started: Thu Nov  1 08:55:53 2018
Stopped: Thu Nov  1 08:56:02 2018
```

Na máquina usada como exemplo (a velocidade pode variar de acordo com a velocidade da CPU/GPU disponível), o ataque aos quatro *hashes* disponíveis usando 10 milhões de senhas demorou... 9 segundos. Como visualizado em **Speed.Dev.#1**, a velocidade de tentativas foi de 2618 kilo-*hashes* por segundo ou, em outras palavras, 2618000 *hashes* por segundo. Foi descoberto um *digest*, que podemos visualizar emitindo o mesmo comando com a *flag* **--show**:

```
$ hashcat --hash-type 111 --attack-mode 0 --username hashes.txt 10-million-
password-list-top-1000000.txt --show
luke:{SSHA}46Qe8Ny+QQgDsbPcps2MODqUHGtdLX41:password
```

Excelente! Como era de se esperar, a senha fraca **password** do usuário **luke** foi descoberta usando o ataque de dicionário.

6. Mas, e as demais senhas? O usuário **han** e **sshca** possuem senhas relativamente mais complexas, mas sabemos que a senha do usuário **admin** é simples, **rnpsr**. Como essa *string* não consta do arquivo com 10 milhões de senhas usado no ataque anterior, ela não foi descoberta, no entanto.

Vamos executar um ataque de força-bruta contra essa senha. Para isso, alteraremos o modo de ataque do **hashcat** para 3, e definiremos uma máscara igual a **?l?l?l?l?l?l** — senhas de até seis caracteres, apenas com caracteres de **[a-z]** minúsculos. Para aprender mais sobre a sintaxe de máscaras suportadas pelo **hashcat**, consulte sua página de manual ou https://hashcat.net/wiki/doku.php?id=mask_attack.

Ao trabalho:

```
$ hashcat --hash-type 111 --attack-mode 3 --username hashes.txt ?l?l?l?l?l?l
hashcat (v3.30) starting...

(...)

[s]tatus [p]ause [r]esume [b]ypass [c]heckpoint [q]uit => s
```

Após a inicialização, a linha acima será mostrada. Podemos apertar os atalhos destacados entre colchetes para instruir o **hashcat** com ações durante o ataque. Apertando **s**, visualizamos o estado atual do ataque:

```
Session.....: hashcat
Status.....: Running
Hash.Type.....: SSHA-1(Base64), nsldaps, Netscape LDAP SSHA
Hash.Target.....: hashes.txt
Time.Started.....: Thu Nov  1 09:05:58 2018 (7 secs)
Time.Estimated...: Thu Nov  1 09:06:30 2018 (25 secs)
Input.Mask.....: ?l?l?l?l?l?l [6]
Input.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....: 27819.2 kH/s (6.17ms)
Recovered.....: 1/4 (25.00%) Digests, 1/4 (25.00%) Salts
Progress.....: 282081280/1235663104 (22.83%)
Rejected.....: 0/282081280 (0.00%)
Restore.Point....: 104192/456976 (22.80%)
Candidates.#1....: sacaxe -> xqegxe
HWMon.Dev.#1.....: N/A
```

Observando a linha **Progress**, notamos que o ataque está 22,83% concluído. Aguardamos.

```
{SSHA}NzQZTz7uf0xNM3PYy7cp+zV6p7bKFncy:rnpesr
```

Após algum tempo, a linha acima é mostrada na tela. O **hashcat** conseguiu quebrar a senha do usuário **admin**, descobrindo-a como sendo **rnpesr**. Aguardamos a conclusão do processo.

```
Session.....: hashcat
Status.....: Exhausted
Hash.Type.....: SSHA-1(Base64), nsldaps, Netscape LDAP SSHA
Hash.Target.....: hashes.txt
Time.Started.....: Thu Nov  1 09:05:58 2018 (30 secs)
Time.Estimated....: Thu Nov  1 09:06:28 2018 (0 secs)
Input.Mask.....: ?l?l?l?l?l?l [6]
Input.Queue.....: 1/1 (100.00%)
Speed.Dev.#1.....: 28006.1 kH/s (6.03ms)
Recovered.....: 2/4 (50.00%) Digests, 2/4 (50.00%) Salts
Progress.....: 1235663104/1235663104 (100.00%)
Rejected.....: 0/1235663104 (0.00%)
Restore.Point....: 456976/456976 (100.00%)
Candidates.#1....: sacxqg -> xqqfqg
HWMon.Dev.#1.....: N/A

Started: Thu Nov  1 09:05:55 2018
Stopped: Thu Nov  1 09:06:29 2018
```

Depois de 34 segundos, o ataque encontra-se 100% concluído. Um novo *digest* foi descoberto, como podemos visualizar com a *flag* `--show`:

```
$ hashcat --hash-type 111 --attack-mode 3 --username hashes.txt ?l?l?l?l?l?l --show
admin:{SSHA}NzQZTz7uf0xNM3PYy7cp+zV6p7bKFNcy:rnpesr
luke:{SSHA}46Qe8Ny+QQgDsbPcps2MODqUHGtdLX41:password
```

O `hashcat` reporta não somente a senha descoberta do usuário `admin`, bem como a senha do usuário `luke` descoberta na execução anterior. Isso ocorre porque o `hashcat` mantém o histórico de ataques realizados no diretório `~/.hashcat`:

```
$ ls -l ~/.hashcat/
hashcat.dictstat
hashcat.potfile
kernels
sessions
```

E assim, concluímos nossa busca por senhas fracas, via ataques de dicionário e força-bruta. O próximo passo, naturalmente, seria alterar o valor de senha dos usuários para um valor novo (bloqueando seu acesso), informá-los da nova senha e comunicar que devem alterar sua senha para uma combinação segura assim que possível.

3) Servidor de arquivos NFS e quotas de disco

Iremos implementar, agora, um servidor de arquivos simples para que os colaboradores da Intranet possam compartilhar arquivos e ter uma opção de backup emergencial para suas estações de trabalho, e para servidores da DMZ compartilharem configurações comuns. Como o ambiente

que estamos simulando é inteiramente baseado em Linux, não há a necessidade de configurar uma solução interoperável com outros sistemas operacionais, como o Samba. Por isso, utilizaremos o NFS (*Network File System*), que é significativamente mais fácil de ser implementado.

1. O primeiro passo, assim como fizemos antes, é clonar a máquina `debian-template` e criar uma nova, que chamaremos de `nfs`. Essa máquina estará conectada a uma única rede `host-only`, com o mesmo nome que foi alocado para a interface de rede da máquina virtual `fw`, configurada durante a sessão 2, que está conectada à DMZ. O IP da máquina será 10.0.42.3/24.

Repita os passos da atividade (11) da sessão (3), quando criamos a máquina `client`, fazendo alterações quando necessário. Após o processo estar concluído, logue como o usuário `luke`.

```
$ hostname ; whoami ; pwd
nfs
luke
/home/luke
```

```
$ ip a s enp0s3 | grep '^ *inet ' | awk '{print $2}'
10.0.42.3/24
```

Se você chegou até aqui, então a integração do servidor `nfs` com o sistema LDAP/SSH-CA está correta.

2. Vamos usar o servidor NFS para duas funções:
 1. Armazenar arquivos de configuração compartilhados entre servidores da DMZ.
 2. Armazenar arquivos de usuários da Intranet, para compartilhamento e backup.

O primeiro caso não exige muita preocupação, já que arquivos de configuração são pequenos e o diretório de armazenamento e arquivos serão *read-only*. Já no segundo caso temos o cenário em que usuários podem querer armazenar muitos arquivos (de forma acidental ou maliciosa), atrapalhando a funcionalidade de outros colaboradores e até mesmo chegando a encher a partição raiz (/) do sistema. Para evitar esse cenário, vamos criar uma partição dedicada para arquivos de usuário no `/home` do servidor `nfs` e aplicar *quotas* de disco aos usuários.

Desligue a máquina `nfs` e adicione a ela um novo disco de 10 GB, usando a interface do Virtualbox. A seguir, formate o disco e adicione-o ao sistema LVM, criando um novo *volume group* `vg-home` com um único volume lógico `lv-home`, de forma análoga ao que fizemos na atividades (7) e (8) da sessão (1). Finalmente, formate esse volume em `ext4` e ative sua montagem automática durante o *boot* da máquina `nfs` no diretório `/home` com as opções `defaults,nosuid,nodev`.

Vamos ao trabalho. Após desligar a VM e adicionar o disco de 10 GB, acessamos a máquina `nfs` como o usuário `root`:


```
# hostname ; whoami
nfs
root
```

O próximo passo é descobrir sob qual nome o disco foi detectado. Nesse caso, temos a vantagem de saber que o tamanho do disco novo, 10 GB, é diferente do disco preexistente.

```
# dmesg | grep 'GiB'
[ 1.585018] sd 1:0:0:0: [sdb] 20971520 512-byte logical blocks: (10.7 GB/10.0 GiB)
[ 1.585187] sd 0:0:0:0: [sda] 16777216 512-byte logical blocks: (8.59 GB/8.00 GiB)
```

Evidentemente, o disco `/dev/sdb` é o que acabamos de adicionar, portanto. Vamos formatá-lo:

```
# fdisk /dev/sdb
```

Bem-vindo ao fdisk (util-linux 2.29.2).
As alterações permanecerão apenas na memória, até que você decida gravá-las.
Tenha cuidado antes de usar o comando de gravação.

A unidade não contém uma tabela de partição conhecida.
Criado um novo rótulo de disco DOS com o identificador de disco 0x3bc30929.

Comando (m para ajuda): o
Criado um novo rótulo de disco DOS com o identificador de disco 0x8a16601c.

Comando (m para ajuda): n
Tipo da partição
p primária (0 primárias, 0 estendidas, 4 livre)
e estendida (recipiente para partições lógicas)

Selecione (padrão p):

Usando resposta padrão p.
Número da partição (1-4, padrão 1):
Primeiro setor (2048-20971519, padrão 2048):
Último setor, +setores ou +tamanho{K,M,G,T,P} (2048-20971519, padrão 20971519):

Criada uma nova partição 1 do tipo "Linux" e de tamanho 10 GiB.

```
Comando (m para ajuda): t
Selecionou a partição 1
Tipo de partição (digite L para listar todos os tipos): 8e
O tipo da partição "Linux" foi alterado para "Linux LVM".
```

```
Comando (m para ajuda): w
A tabela de partição foi alterada.
Chamando ioctl() para reler tabela de partição.
Sincronizando discos.
```

Agora, vamos criar o volume físico:

```
# pvcreate /dev/sdb1
Physical volume "/dev/sdb1" successfully created.
```

O grupo de volumes:

```
# vgcreate vg-home /dev/sdb1
Volume group "vg-home" successfully created
```

E, finalmente, o volume lógico:

```
# lvcreate -l +100%FREE -n lv-home vg-home
Logical volume "lv-home" created.
```

Vamos, agora, formatar o sistema de arquivos do LV:

```
# mkfs.ext4 /dev/mapper/vg--home-lv--home
mke2fs 1.43.4 (31-Jan-2017)
Creating filesystem with 2620416 4k blocks and 655360 inodes
Filesystem UUID: be99743c-7ca0-4144-8548-d7aab33a878b
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

Sincronizar os arquivos do diretório **/home** atual com o LV recém-criado:

```
# mount /dev/mapper/vg--home-lv--home /mnt/
```

```
# rsync -av /home/ /mnt/
sending incremental file list
./
aluno/
aluno/.bash_history
aluno/.bash_logout
aluno/.bashrc
aluno/.profile
aluno/.vimrc
luke/
luke/.bash_history
luke/.bash_logout
luke/.bashrc
luke/.profile
luke/scripts/
luke/scripts/sshsign_user.sh

sent 11,669 bytes  received 225 bytes  23,788.00 bytes/sec
total size is 10,787  speedup is 0.91
```

```
# umount /mnt
```

E, finalmente, configurar a montagem no `/etc/fstab` e montar o LV:

```
# nano /etc/fstab
(...)
```

```
# tail -n1 /etc/fstab
/dev/mapper/vg--home-lv--home /home ext4 defaults,nosuid,nodev 0 2
```

```
# mount -a
```

```
# df -h | sed -n '1p; /\//home/p'
Sist. Arq.                               Tam. Usado Disp. Uso% Montado em
/dev/mapper/vg--home-lv--home 9,8G  37M  9,3G  1% /home
```

3. Agora, vamos instalar os pacotes para habilitar o compartilhamento de arquivos via NFS e *quotas* de disco. Como `root`, instale os pacotes `nfs-kernel-server`, `quota` e `quotatool`:

```
# apt-get install nfs-kernel-server quota quotatool
```

4. Vamos configurar primeiro o sistema de *quotas*. Edite a entrada do diretório `/home` no arquivo

`/etc/fstab` e adicione as opções `usrquota,grpquota`, que ativam suporte a *quotas* por usuário e por grupo no sistema de arquivos.

```
# nano /etc/fstab
(...)
```

```
# tail -n1 /etc/fstab
/dev/mapper/vg--home-lv--home /home ext4 defaults,nosuid,nodev,usrquota,grpquota
0 2
```

Em seguida, reinicie o sistema.

```
# reboot
```

Após o *reboot*, acesse a máquina como `root` e verifique se o sistema de *quotas* foi habilitado na partição.

```
# mount | grep '/home'
/dev/mapper/vg--home-lv--home on /home type ext4
(rw,nosuid,nodev,relatime,quota,usrquota,grpquota,data=ordered)
```

Crie os arquivos de configuração de *quotas* usando o comando `quotacheck`:

```
# quotacheck -ugc /home/
```

Pronto, o sistema de *quotas* está configurado. Iremos editar *quotas* de usuário e testar seu funcionamento mais à frente, após a configuração do NFS.

5. O próximo passo é configurar o serviço NFS. Retomando a descrição do passo (2), queremos disponibilizar compartilhamentos para arquivos de usuário (para o qual utilizaremos o diretório `/home` recém-criado) e para arquivos de configuração comuns entre servidores — para este caso, iremos criar um diretório específico, `/config`.

```
# mkdir /config
```

Vamos configurar as exportações do NFS. Edite o arquivo `/etc/exports`, e insira o seguinte conteúdo:

```
1 /config 10.0.42.0/24(ro,async,no_subtree_check)
2 /home 192.168.42.0/24(rw,async,no_subtree_check,root_squash)
```

O que está sendo configurado?

- Estamos exportando o diretório `/config` para todas as máquinas da DMZ (faixa 10.0.42.0/24), em modo somente leitura, assíncrono e sem checagem de sub-árvores de montagem (consultar página de manual com `man 5 exports`).
- Estamos exportando o diretório `/home` para todas as máquinas da Intranet (faixa 192.168.42.0/24), em modo leitura-escrita, assíncrono (i.e. escritas ao disco do servidor remoto não precisam ter sido efetivadas para que o cliente receba confirmação), sem checagem de sub-árvores de montagem e desabilitando o mapeamento de UID do `root` da máquina remota no servidor local.

Para exportar os diretórios, basta executar:

```
# exportfs -a
```

Finalmente, para visualizar quais diretórios estão sendo exportados, execute:

```
# showmount -e
Export list for nfs:
/home 192.168.42.0/24
/config 10.0.42.0/24
```

6. Vamos testar nosso sistema de compartilhamento de arquivos via NFS e *quotas* de disco. Acesse a máquina `client` como o usuário `root`, crie um diretório `/remote` para ser o ponto de montagem NFS e monte o diretório compartilhado.

```
# hostname ; whoami
client
root
```

```
# mkdir /remote
```

```
# mount -t nfs 10.0.42.3:/home /remote
```

```
# ls -l /remote/
total 40
drwxr-xr-x 2 aluno aluno 4096 out 18 16:29 aluno
-rw----- 1 root root 8192 nov 1 12:23 aquota.group
-rw----- 1 root root 8192 nov 1 12:23 aquota.user
drwx----- 2 root root 16384 nov 1 11:58 lost+found
drwxr-xr-x 3 luke sysadm 4096 nov 1 11:46 luke
```

7. Um dos principais problemas em sistemas de compartilhamento de arquivos em ambientes Unix é o mapeamento de UIDs e GIDs—como garantir que os usuários de múltiplas máquinas

remotas possuam os mesmos identificadores que os usuários existentes no servidor de arquivos? Felizmente, nosso sistema centralizado de autenticação usando LDAP resolve esse problema de forma transparente: todos os usuários possuem um valor de UID e GID consistente em todo o *datacenter*, já que as contas são gerenciadas em um ponto único.

Senão, vejamos: como o usuário **luke**, tente criar um arquivo novo dentro do ponto de montagem **/remote/luke**:

```
$ hostname ; whoami
client
luke
```

```
$ echo test > /remote/luke/file
```

```
$ cat /remote/luke/file
test
```

Funcionou perfeitamente! Uma vez que os valores de UID e GID do usuário **luke** são consistentes entre a máquina **client** e o servidor de arquivos **nfs**, não temos problemas de permissão.

8. E quanto ao usuário **root**? Será que o **root** local da máquina **client** possui acesso irrestrito aos arquivos compartilhados?

```
# hostname ; whoami
client
root
```

```
# echo test > /remote/file
-su: /remote/file: Permissão negada
```

```
# rm /remote/aquota.user
rm: não foi possível remover '/remote/aquota.user': Permissão negada
```

Devido à utilização da opção **root_squash** no compartilhamento configurado via arquivo **/etc/exports** da máquina **nfs**, o mapeamento de UID do usuário **root** em máquinas remotas é desativado, efetivamente impedindo-o de alterar quaisquer arquivos.

9. Vamos testar o sistema de *quotas*. Na máquina **nfs**, como o usuário **root**, edite as *quotas* do usuário **luke** usando o comando **edquota**:

```
# edquota -u luke
```

O comando `edquota` irá invocar um editor (indicado pela variável de ambiente `$EDITOR`) para que as *quotas* sejam ajustadas. Vamos editar os campos *soft* e *hard* da seção *block* do arquivo, ajustando limites de 100 MB e 200 MB, respectivamente — note que os valores devem ser informados em kilobytes. Pode-se, opcionalmente, também setar um limite para *inodes* que o usuário pode criar.

```
Disk quotas for user luke (uid 10000):
  Filesystem          blocks      soft      hard      inodes      soft
hard
  /dev/mapper/vg--home-lv--home 32    100000    200000         8         0
0
```

Para verificar as *quotas* ativas em um sistema de arquivos, use o comando `repquota`:

```
# repquota -u /home
*** Report for user quotas on device /dev/mapper/vg--home-lv--home
Block grace time: 7days; Inode grace time: 7days
      Block limits
User      used  soft  hard  grace  File limits
      used soft hard  grace  used soft hard  grace
-----
root      --    20     0     0           2     0     0
aluno     --    24     0     0           6     0     0
luke      --    32 100000 200000       8     0     0
```

10. Acesse a máquina `client` como o usuário `luke`. Vamos tentar extrapolar o limite estabelecido pela *quota* no passo anterior.

```
$ hostname ; whoami
client
luke
```

O kernel do SO é um arquivo interessante a ser usado para esse teste, já que possui um tamanho razoável. Vamos copiá-lo sucessivas vezes para o diretório `/remote/luke` e verificar o que acontece:

```
$ du -sh /boot/vmlinuz-4.9.0-8-amd64
4,1M    /boot/vmlinuz-4.9.0-8-amd64
```

```
$ for i in {1..100}; do cp /boot/vmlinuz-4.9.0-8-amd64 /remote/luke/vmlinuz-$i;
done
cp: falha ao fechar '/remote/luke/vmlinuz-49': Disk quota exceeded
cp: falha ao fechar '/remote/luke/vmlinuz-50': Disk quota exceeded
cp: falha ao fechar '/remote/luke/vmlinuz-51': Disk quota exceeded
(...)
cp: falha ao fechar '/remote/luke/vmlinuz-98': Disk quota exceeded
cp: falha ao fechar '/remote/luke/vmlinuz-99': Disk quota exceeded
cp: falha ao fechar '/remote/luke/vmlinuz-100': Disk quota exceeded
```

Note que após 48 cópias de arquivo, o sistema reporta a *quota* de disco como excedida, e o usuário não pode mais escrever na partição. De fato, checando o estado da *quota* de disco com o comando `repquota` na máquina `nfs`, temos que:

```
# hostname ; whoami
nfs
root
```

```
# repquota -u /home/
*** Report for user quotas on device /dev/mapper/vg--home-lv--home
Block grace time: 7days; Inode grace time: 7days
```

User	Block limits				File limits			
	used	soft	hard	grace	used	soft	hard	grace
root	-- 20	0	0		2	0	0	
aluno	-- 24	0	0		6	0	0	
luke	+- 200000	100000	200000	6days	108	0	0	

Temos, portanto, que nosso esquema de *quotas* está funcionando como esperado. Não se esqueça de apagar os diversos arquivos `vmlinuz*` que criamos, para liberar espaço no disco novamente:

```
# rm /home/luke/vmlinuz-*
```




Observe que apenas os usuários **aluno** e **luke** possuem pastas no diretório **/home** compartilhado pela máquina **nfs**. Isso se deve ao fato de que apenas esses usuários haviam feito acesso local à máquina **nfs** até aquele momento — lembre-se que o arquivo de configuração **/usr/share/pam-configs/mkhomedir** que aplicamos ao PAM cria diretórios **home** apenas quando o usuário faz acesso à máquina pela primeira vez. Como consequência, o usuário **han**, para citar um exemplo, não possui uma pasta no servidor de arquivos.

Em produção, seria interessante que a pasta compartilhada do usuário fosse criada assim que este fosse adicionado à base LDAP, juntamente com o comando **ldapadduser**, por exemplo. Um *script* shell seria ideal para resolver essa situação. Claro, é possível que nem todos os novos usuários criados na base LDAP devam ter uma pasta nesse servidor, o que pode complicar sua configuração.

4) Uso de ACLs localmente

Imagine a seguinte situação, agora: o usuário **luke** quer criar um arquivo novo, sigiloso, e dar permissão para que **han** possa visualizá-lo. Ora, com as permissões padrão disponíveis em um sistema Linux, quais são nossas opções?

Sabemos que, ao criar o arquivo, o usuário-dono será **luke** e o grupo-dono será **sysadm**. Se **luke** altera o grupo-dono do arquivo para **fwadm** e **chmod** de **640**, apesar de a permissão objetivada para **han** ser garantida, todos os outros membros do grupo **fwadm** também poderão visualizar o arquivo, que não é o que queremos. Se garante-se a permissão de **644**, não só **han** como qualquer outro usuário pode visualizar o arquivo. Finalmente, a alternativa final que seria adicionar **han** ao grupo **sysadm** pode não ser desejável ou aceitável do ponto de vista administrativo. O que fazer?

O uso de ACLs (*Access Control Lists*) é especialmente adequado para esse tipo de situação, quando precisamos configurar permissões de arquivos e diretórios de forma granular. Com o uso de ACLs, é possível definir permissões customizadas para usuários e grupos diferentes dos donos do arquivo/diretório original, solucionando problemas de permissionamento para os quais o sistema tradicional de permissões Unix é inadequado.

1. Acesse a máquina **nfs** como o usuário **root**. Para consultar e ajustar ACLs localmente, basta instalar o pacote **acl**:

```
# hostname ; whoami
nfs
root
```

```
# apt-get install acl
```

2. Vamos testar o funcionamento de ACLs localmente, usando os usuários **luke** e **han**. Acesse a máquina **nfs** como o usuário **luke**:

```
$ hostname ; whoami ; pwd
nfs
luke
/home/luke
```

Agora, crie o arquivo novo `~/teste`, com qualquer conteúdo. Em seguida, consulte suas ACLs atuais.

```
$ echo oi > teste
```

```
$ getfacl teste
# file: teste
# owner: luke
# group: sysadm
user::rw-
group::r--
other::r--
```

3. Imaginemos que o arquivo criado na atividade anterior é especialmente sigiloso, devendo ser visualizado apenas pelo usuário `han` e seu dono, `luke`. Primeiro, retire as permissões do grupo e de outros:

```
$ chmod 600 ~/teste
```

Em seguida, use ACLs para dar permissão de leitura a `han`:

```
$ setfacl -m u:han:r ~/teste
```

Verifique as permissões Unix tradicionais—observe que ao final da coluna de permissionamento do `ls` vemos o caractere `+`, que indica que o arquivo possui permissões estendidas na forma de ACLs.

```
$ ls -ld /home/luke/teste
-rw-r-----+ 1 luke sysadm 3 nov  1 18:27 /home/luke/teste
```

Consulte novamente as ACLs do arquivo, verificando que a configuração desejada foi aplicada.

```
$ getfacl teste
# file: teste
# owner: luke
# group: sysadm
user::rw-
user:han:r--
group::---
mask::r--
other::---
```

4. Terá funcionado? Vamos ver. Como o usuário **aluno**, tente visualizar o conteúdo do arquivo **/home/luke/teste**:

```
$ whoami
aluno
```

```
$ cat /home/luke/teste
cat: /home/luke/teste: Permissão negada
```

E como **han**? Vamos ver:

```
$ whoami
han
```

```
$ cat /home/luke/teste
oi
```

5. Como **luke**, vamos remover a ACL de leitura do usuário **han** e testar:

```
$ whoami
luke
```

```
$ setfacl -x u:han ~/teste
```

```
$ su - han
Senha:
```

```
$ whoami
han
```

```
$ cat /home/luke/teste
cat: /home/luke/teste: Permissão negada
```

Perfeito! Lembre-se que também podemos configurar ACLs para grupos através do caractere **g**, o que não foi testado nesta atividade.

5) Uso de ACLs via NFS

A atividade anterior, apesar de interessante, é pouco prática quando consideramos nossa configuração atual: se ACLs podem apenas ser manipuladas localmente mas estamos mantendo nossos arquivos compartilhados via rede com NFS, então toda vez que um usuário quiser alterar ACLs ele terá que fazer um acesso local à máquina **nfs**? Não é razoável fazermos isso. De fato, tente fazer a alteração de ACLs a partir da máquina **client** como o usuário **luke**:

```
$ hostname ; whoami
client
luke
```

```
$ setfacl -m u:han:rw /remote/luke/teste
setfacl: /remote/luke/teste: Operação não suportada
```

Com efeito, ACLs POSIX não são suportadas diretamente via **setfacl** em *mounts* NFS.

Por outro lado, *mounts* NFS versão 4 possuem suporte a ACLs—de fato, a um conjunto de permissões ainda mais granulares e expressivas que as ACLs POSIX padrão. Mas primeiro, temos que responder à pergunta: nosso compartilhamento atual está em qual versão? Vamos ver:

```
$ mount | grep '/home' | grep -o 'vers=[0-9\\.]*'
vers=4.2
```

Excelente, estamos usando a versão 4.2, o que deve ser suficiente. Os comandos para visualização e edição de ACLs NFSv4 não são os mesmos que utilizamos até agora, no entanto — vamos instalá-los.

1. Acesse a máquina **client** como o usuário **root** e instale o pacote **nfs4-acl-tools**:

```
# hostname ; whoami
client
root
```

```
# apt-get install nfs4-acl-tools
```

2. Agora sim, vamos testar o funcionamento de ACLs com a pasta compartilhada via NFS. Acesse

como o usuário **luke**; para tornar o uso corriqueiro dessa pasta compartilhada mais conveniente, crie um link simbólico com o nome **remote** em seu diretório *home*.

```
$ hostname ; whoami ; pwd
client
luke
/home/luke
```

```
$ ln -s /remote/luke/ ~/remote
```

3. Consulte as ACLs NFSv4 do arquivo criado na atividade anterior:

```
$ nfs4_getfacl ~/remote/teste
A::OWNER@:rwatTcCy
A::GROUP@:tcy
A::EVERYONE@:tcy
```

O formato de representação de permissões NFSv4 é bastante diferente do que estamos acostumados — muitas opções e controles adicionais são suportados. Nesta atividade iremos trabalhar apenas com as permissões mais usuais, **rw**, mas a página de manual `man 5 nfs4_acl` possui uma documentação bastante completa sobre as possibilidades de uso desse sistema. Em especial, a seção *ACE PERMISSIONS* é recomendada para entender o formato do *output* acima.

Como um exemplo, vamos analisar em detalhe a ACE (*Access Control Entry*) **A::OWNER@:rwatTcCy**:

- **A**: tipo da ACE; pode ser **A** (*allow*), **D** (*deny*), **U** (*audit*, usada para configurar log de acessos) e **L** (*alarm*, para gerar alarmes de sistema em caso de acesso).
- **::**: o segundo campo, neste caso vazio, define as *flags* da ACE; pode ser utilizado para indicar ACEs aplicáveis a grupos, configurações de herança da ACE para diretórios e arquivos-filho, ou *flags* administrativas para controlar eventos de log e alarme.
- **OWNER@**: define o *principal* ao qual se aplica a ACE corrente; pode ser um usuário, grupo ou uma de três ACEs especiais, **OWNER@**, **GROUP@** e **EVERYONE@**, funcionalmente equivalentes às suas contrapartes POSIX.
- **rwatTcCy**: permissões definidas pela ACE; no caso, temos definidas:
 - **r**: permissão de leitura para arquivos, ou listagem de diretórios.
 - **w**: permissão de escrita para arquivos, ou criação de novos arquivos em diretórios.
 - **a**: *append* de dados em arquivos (escrever ao final), ou criar novos subdiretórios em diretórios.
 - **t**: ler atributos do arquivo/diretório.
 - **T**: escrever atributos do arquivo/diretório.
 - **c**: ler ACLs NFSv4 do arquivo/diretório.
 - **C**: escrever ACLs NFSv4 do arquivo/diretório.

- **y**: autorizar clientes a usar I/O síncrono com o servidor.

4. Vamos configurar uma ACL NFSv4 de leitura do arquivo para o usuário **han**, assim como fizemos anteriormente.

```
$ nfs4_setfacl -a A::han@intnet:rtcy ~/remote/teste
```

Vamos ver como ficaram as ACEs do arquivo:

```
$ nfs4_getfacl ~/remote/teste
A::OWNER@:rwatTcCy
A::10002:rtcy
A::GROUP@:tcy
A::EVERYONE@:tcy
```

Note que o nome de usuário **han@intnet** foi mapeado para o UID **10002** — que é consistente entre todas as máquinas do *datacenter* graças à integração com o LDAP que fizemos na sessão 3. Verifique a correspondência do UID:

```
$ getent passwd han
han*:10002:10002:han:/home/han:/bin/bash
```

5. Vamos testar? Acesse como o usuário **aluno** e tente exibir o conteúdo do arquivo **/remote/luke/teste**:

```
$ su - aluno
Senha:
```

```
$ whoami
aluno
```

```
$ cat /remote/luke/teste
cat: /remote/luke/teste: Permissão negada
```

Agora, teste com o usuário **han**:

```
$ su - han
Senha:
```

```
$ whoami
han
```

```
$ cat /remote/luke/teste  
oi
```

Excelente, tudo funcionando a contento.

6. Como remover uma ACL NFSv4? É simples:

```
$ nfs4_setfacl -x A::han@intnet:rtcy ~/remote/teste
```

```
$ nfs4_getfacl ~/remote/teste  
A::OWNER@:rwatTcCy  
A::10002:rtcy  
A::GROUP@:tcy  
A::EVERYONE@:tcy
```

Ué, não funcionou. Para deletar ACEs, temos que especificá-las **exatamente** no mesmo formato da linha reportada pelo comando `nfs4_getfacl`, ou usando o índice numérico da regra. Vamos tentar novamente:

```
$ nfs4_setfacl -x A::10002:rtcy ~/remote/teste
```

```
$ nfs4_getfacl ~/remote/teste  
A::OWNER@:rwatTcCy  
A::GROUP@:tcy  
A::EVERYONE@:tcy
```

Agora sim, perfeito. Vamos verificar que a remoção da ACE surtiu efeito:

```
$ su - han  
Senha:
```

```
$ whoami  
han
```

```
$ cat /remote/luke/teste  
cat: /remote/luke/teste: Permissão negada
```

6) Controle granular de permissões via sudo

Para todas as ações privilegiadas que precisamos tomar até aqui, sempre usamos o comando `su` para nos tornarmos o usuário `root`, e então efetuamos a instalação de pacotes, adição de usuários ou criação de arquivos de configuração. Mas, como fica essa situação em um ambiente de *datacenter* como o que estamos simulando? Seria interessante passar a senha do usuário `root` para os usuários `luke` e `han` (e outros que viermos a criar), permitindo que tomem quaisquer ações nas máquinas?

O `sudo` (*Super User DO*) é um comando que permite que usuários comuns obtenham privilégios de outro usuário, em geral o `root`, para executar tarefas específicas dentro do sistema de maneira segura e controlável pelo administrador. Assim, podemos delimitar que um determinado usuário ou grupo pode executar apenas um pequeno conjunto de comandos dentro de um servidor específico. Como o `sudo` é compatível com *hostnames* e endereços IP, é possível utilizar o mesmo arquivo em todas as máquinas do parque, facilitando tremendamente o esforço de configuração.

Para ilustrar esse cenário, vamos solucionar dois exemplos hipotéticos:

- A colaboradora `leia` acaba de se juntar à equipe de `han`, o grupo `fwadm` em nosso sistema LDAP. Imagine que ela ficará responsável por editar regras no firewall de borda, a máquina `fw`. Mas, por estar começando agora na empresa, `han` quer restringir o conjunto de comandos que `leia` pode executar na máquina, liberando apenas a edição do firewall via `iptables`. Sua senha será `seg10leia`. Nas demais máquinas (`ldap` e `nfs`) `leia` não deve ter qualquer acesso especial, apenas como um usuário regular.
- O colaborador `chewie` foi contratado para auxiliar na manutenção da base LDAP da empresa. Para desempenhar suas tarefas, iremos colocá-lo em um novo grupo `ldapadm`. Os membros desse grupo devem ter acesso aos principais comandos de edição do LDAP (criação, modificação e deleção de usuários e grupos) na máquina `ldap`. Sua senha será `seg10chewie`. Nas demais máquinas (`fw` e `nfs`) `chewie` não deve ter qualquer acesso especial, apenas como um usuário regular.
- Os usuários atuais, `luke` e `han`, terão permissão para executar qualquer comando como o usuário `root`, em qualquer máquina.
- Observe que temos controles alheios ao `sudo` já aplicados que irão restringir o acesso de certos usuários — por exemplo, apenas membros do grupo `fwadm` conseguem fazer login na máquina `fw` devido à configuração do `nslcd` que realizamos na atividade (13) da sessão (3).

Vamos solucionar esses problemas?

1. Primeiro, devemos criar os usuários e grupos — vamos começar com `leia`. Como `root`, na máquina `ldap`:

```
# hostname ; whoami
ldap
root
```



```
# ldapadduser leia fwadm
Successfully added user leia to LDAP
Successfully set password for user leia
```

```
# ldapaddusertogroup leia fwadm
Successfully added user leia to group cn=fwadm,ou=Groups,dc=intnet
```

```
# ldapsetpasswd leia
Changing password for user uid=leia,ou=People,dc=intnet
New Password:
Retype New Password:
Successfully set password for user uid=leia,ou=People,dc=intnet
```

Agora, **chewie**. Lembre-se que no caso dele temos também que adicionar um novo grupo, **ldapadm**:

```
# ldapaddgroup ldapadm
Successfully added group ldapadm to LDAP
```

```
# ldapadduser chewie ldapadm
Successfully added user chewie to LDAP
Successfully set password for user chewie
```

```
# ldapaddusertogroup chewie ldapadm
Successfully added user chewie to group cn=ldapadm,ou=Groups,dc=intnet
```

```
# ldapsetpasswd chewie
Changing password for user uid=chewie,ou=People,dc=intnet
New Password:
Retype New Password:
Successfully set password for user uid=chewie,ou=People,dc=intnet
```

Fácil, não é mesmo?

2. Vamos agora solucionar o problema de permissionamento. Queremos controlar os comandos utilizados nos servidores do *datacenter*, que até o momento são as máquinas **fw**, **ldap** e **nfs**. Como já instalamos o **sudo** na máquina **debian-template** na sessão 1, o comando deve estar disponível no **\$PATH**:

```
# which sudo
/usr/bin/sudo
```

Tecnicamente, seria possível configurar o `sudo` em cada um dos servidores — uma vez que as regras para a usuária `leia` são específica para a máquina `fw` e as do usuário `chewie` se aplicam à máquina `ldap` — mas não faremos isso. Imagine que, ao invés de três máquinas, nosso *datacenter* tivesse centenas de VMs: seria factível controlar as regras de `sudo` localmente em cada um dos servidores? É evidente que não.

Temos algumas opções para configurar o `sudo` de forma coordenada entre múltiplos servidores. A gestão de configuração (como a ferramenta Ansible, que utilizaremos na sessão 6) é uma dessas opções, um método bastante moderno e conveniente de solucionar o problema. Neste momento, no entanto, iremos empregar uma solução mais simples — usar a pasta `/config`, compartilhada via NFS, para atingir esse objetivo.

O `sudo` é configurado através de dois artefatos: o arquivo `/etc/sudoers`, e arquivos dentro do diretório `/etc/sudoers.d` (incluídos na configuração via diretiva `includedir` no arquivo padrão). Evidentemente, o diretório `/etc` é local em cada servidor que estamos trabalhando — iremos usar links simbólicos para redirecionar o `sudo` para buscar a configuração no arquivo `/config/sudoers`, renomeando o arquivo original, e mantendo a pasta `/etc/sudoers.d` para que alterações locais possam ser realizadas.

Acesse a máquina `nfs` como o usuário `root`:

```
# hostname ; whoami
nfs
root
```

Copie o arquivo `/etc/sudoers` para a pasta `/config`:

```
# cp -a /etc/sudoers /config
```

Renomeie o arquivo original:

```
# mv /etc/sudoers /etc/sudoers.old
```

Crie um link simbólico para o novo arquivo de configuração:

```
# ln -s /config/sudoers /etc/
```

Como o arquivo `/config/sudoers` terá que ser lido pelo usuário `root` em outras máquinas e estamos usando a opção `root_squash` no *mount* NFS, é necessário conceder permissão de leitura para "outros" neste arquivo:

```
# ls -ld /config/sudoers
-r--r----- 1 root root 669 nov  2 11:15 /config/sudoers
```

```
# chmod o+r /config/sudoers
```

Note que o arquivo `/config/sudoers` ainda inclui a pasta local `/etc/sudoers.d`, o que permite ao administrador realizar configurações locais independentes do compartilhamento de arquivos NFS.

```
# grep includedir /config/sudoers
#includedir /etc/sudoers.d
```

Vamos testar? Edite o arquivo `/config/sudoers` e autorize o usuário `luke` a usar o comando `/bin/grep` como o usuário `root`. Edite o arquivo com:

```
# visudo -f /config/sudoers
(...)
```

Insira a linha `luke ALL=/bin/grep` abaixo da entrada do usuário `root` na seção *User privilege specification*, como se segue:

```
# grep -A2 'User privilege specification' /config/sudoers
# User privilege specification
root    ALL=(ALL:ALL) ALL
luke    ALL=          /bin/grep
```

Como o usuário `luke`, tente usar o comando `grep` com o `sudo` para visualizar um arquivo restrito, como o `/etc/shadow`:

```
# su - luke
```

```
$ whoami
luke
```

```
$ sudo grep root /etc/shadow
root:$6$2SMIuRXP$mfXWI0HACpYqLUup.aEYcLrr4eo3WJeDmr8G8etaUC2tdNzBqn9i4yeQf0vtdMUdJ5
Y7D2ySGA72K0skF75in0:17822:0:99999:7:::
```

Agora, tente executar um comando não-autorizado, como o `cat`:

```
$ sudo cat /etc/shadow
```

Sinto muito, usuário luke não tem permissão para executar `"/bin/cat /etc/shadow"` como root em nfs.intnet.

Perfeito, nosso teste inicial funcionou com sucesso. Remova a linha referente ao usuário **luke** no arquivo `/config/sudoers`, e vamos prosseguir.

3. Vamos configurar o arquivo `/config/sudoers` de acordo com a especificação da atividade. Usando o comando `visudo -f /config/sudoers`, edite o arquivo com o seguinte conteúdo:

```

1 Defaults      env_reset
2 Defaults      mail_badpass
3 Defaults      secure_path
="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
4
5 User_Alias ADMINS      = aluno, \
6                          luke, \
7                          han
8
9 User_Alias FWUSERS      = leia
10
11 User_Alias LDAPUSERS    = %ldapadm
12
13 Host_Alias FWHOSTS      = fw
14
15 Host_Alias LDAPHOSTS    = ldap
16
17 Cmnd_Alias FWCMDs       = /sbin/iptables
18
19 Cmnd_Alias LDAPCMDs     = /usr/sbin/ldapaddgroup,          \
20                          /usr/sbin/ldapadduser,            \
21                          /usr/sbin/ldapaddusertogroup,      \
22                          /usr/sbin/ldapdeletigroup,         \
23                          /usr/sbin/ldapdeleteuser,         \
24                          /usr/sbin/ldapdeleteuserfromgroup, \
25                          /usr/sbin/ldapmodifygroup,         \
26                          /usr/sbin/ldapmodifymachine,      \
27                          /usr/sbin/ldapmodifyuser,         \
28                          /usr/sbin/ldaprenamegroup,        \
29                          /usr/sbin/ldaprenameuser,         \
30                          /usr/sbin/ldapsetpasswd,           \
31                          /usr/sbin/ldapsetprimarygroup
32
33 root          ALL=(ALL:ALL)    ALL
34
35 ADMINS        ALL=(ALL:ALL)    ALL
36
37 FWUSERS        FWHOSTS=(root)  FWCMDs
38
39 LDAPUSERS      LDAPHOSTS=(root) LDAPCMDs
40
41 #includedir /etc/sudoers.d

```

O que estamos fazendo? Vamos ver:

- Nas linhas [5-10] definimos *alias*es (apelidos) de usuários para agrupar os elementos que serão configurados para usar o **sudo**. Criamos um *alias* **ADMINS** para agrupar os usuários **aluno**, **luke** e **han**, **FWUSERS** para **leia** e **LDAPUSERS** para o **grupo** **ldapadm**. É especialmente importante manter um *alias* apontando para um usuário local, como o usuário **aluno**, caso

haja problemas com o LDAP.

- Nas linhas [12-14] definimos *aliases* para máquinas, **fw** e **ldap**. Também poderíamos usar endereços IP, se desejado.
- Nas linhas [16-30] definimos *aliases* de comandos: para a máquina **fw**, apenas o comando **/sbin/iptables** é suficiente; já para a máquina **ldap** configuramos uma lista detalhada dos comandos que o *alias* **LDAPUSERS** poderá usar.
- Nas linhas [32-38] fazemos a "amarração" dos *aliases* previamente definidos, atribuindo aos usuários/grupos em quais máquinas eles podem executar os comandos, como quais usuários, e quais são esses comandos.

4. Vamos testar o acesso de **leia** na máquina **fw**. Antes disso o primeiro passo, é claro, é criar o diretório **/config** e configurar sua montagem automática durante o *boot* via **/etc/fstab**. Acesse **fw** como **root**, crie o diretório **/config** e insira a linha a seguir no final do arquivo:

```
# hostname ; whoami
fw
root
```

```
# mkdir /config
```

```
# nano /etc/fstab
(...)
```

```
# tail -n1 /etc/fstab
10.0.42.3:/config /config nfs defaults 0 0
```

Monte o diretório e verifique seu conteúdo:

```
# mount -a
```

```
# mount | grep config
10.0.42.3:/config on /config type nfs4
(rw,relatime,vers=4.2,rsize=131072,wsiz=131072,namlen=255,hard,proto=tcp,port=0,ti
meo=600,retrans=2,sec=sys,clientaddr=10.0.42.1,local_lock=none,addr=10.0.42.3)
```

```
# ls /config/
sudoers
```

Agora, renomeie o arquivo **/etc/sudoers** e crie o link simbólico:

```
# mv /etc/sudoers /etc/sudoers.old ; ln -s /config/sudoers /etc/
```

Perfeito, agora vamos testar o funcionamento da configuração. Como **leia**, tente executar o comando **iptables** usando o **sudo**:

```
$ hostname ; whoami
fw
leia
```

```
$ sudo iptables -L
[sudo] senha para leia:
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

Excelente! E se tentarmos executar um comando não autorizado?

```
$ sudo rm /etc/shadow
Sinto muito, usuário leia não tem permissão para executar "/bin/rm /etc/shadow"
como root em fw.intnet.
```

De fato, é possível listar exatamente quais comandos um usuário está apto a executar com o comando **sudo -l**:

```
$ sudo -l
Entradas de Defaults correspondentes a leia em fw:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
```

```
Usuário leia pode executar os seguintes comandos em fw:
    (root) /sbin/iptables
```

E quanto a **han**? Ele consegue executar qualquer comando como **root**?

```
$ hostname ; whoami
fw
han
```

```
$ sudo -l
[sudo] senha para han:
Entradas de Defaults correspondentes a han em fw:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
```

Usuário han pode executar os seguintes comandos em fw:
(ALL : ALL) ALL

Perfeito! A última questão é a seguinte: e se **leia**, por qualquer motivo, conseguir obter a senha do usuário **root**? O que não é exatamente difícil, já que estamos usando **rnpesr** como senha. Nesse caso, ela terá acesso irrestrito:

```
$ hostname ; whoami
fw
leia
```

```
$ su -
Senha:
```

```
# whoami
root
```

A solução ideal, nesse caso, é desabilitar a senha do **root**. Com isso, mesmo que os usuários saibam a senha, ela não poderá ser usada para efetuar escalada de privilégios usando o **sudo**. Podemos usar o comando **passwd -l** para fazer isso:

```
# passwd -l root
passwd: informação de expiração de senha alterada.
```

```
# exit
```

```
$ whoami
leia
```

```
$ su -
Senha:
su: Falha de autenticação
```


Com a senha desabilitada, apenas aqueles usuários que tenham permissão de `sudo` para executar comandos de escalada de privilégio poderão tornar-se o usuário `root` — todos os demais, restritos a um subconjunto de comandos controlados pelo arquivo `/config/sudoers`, não conseguirão fazê-lo.

Note que mesmo o usuário `han`, que possui acesso irrestrito, não consegue executar `su` diretamente:

```
$ whoami  
han
```

```
$ su -  
Senha:  
su: Falha de autenticação
```

```
$ sudo --login
```

```
# whoami  
root
```

Apenas via `sudo su` ou `sudo --login` (que equivale a invocar um *shell* de login, como executar `sudo bash`) é possível escalar privilégio, como demonstrado.



A leitura do arquivo `/config/sudoers` a partir de um compartilhamento de rede, via NFS, traz consigo uma preocupação de segurança bastante relevante — e se a máquina `nfs` estiver indisponível? Com efeito, se isso acontecer teremos grandes problemas, já que toda a configuração de autorização do sistema local estará indisponível. Por esse motivo, é fundamental que o `sudoers` esteja acessível localmente, o que faremos na sessão 6 deste curso.

Por ora, vamos torcer para que nada catastrófico aconteça com a máquina `nfs`. Dedos cruzados.

5. Vamos para o caso do usuário `chewie`. Acesse a máquina `ldap` como o usuário `root` e:

- Crie o diretório `/config`.
- Configure sua montagem automática durante o *boot* via `/etc/fstab`.
- Configure o `sudo` para ler a configuração do `/config/sudoers`.
- Desabilite a senha do usuário `root`.
- Teste o funcionamento da configuração com os usuários `chewie` e `luke`.

Dada a semelhança dos primeiros quatro itens com o passo anterior, iremos passar diretamente para o passo final, assumindo que o aluno completou a configuração com sucesso.

Como o usuário **chewie** na máquina **ldap**, verifique quais comandos você está autorizado a executar usando o **sudo**:

```
$ hostname ; whoami
ldap
chewie
```

```
$ sudo -l
Entradas de Defaults correspondentes a chewie em ldap:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin
```

Usuário chewie pode executar os seguintes comandos em ldap:

```
(root) /usr/sbin/ldapaddgroup, /usr/sbin/ldapadduser,
/usr/sbin/ldapaddusertogroup,
    /usr/sbin/ldapdeletigroup, /usr/sbin/ldapdeleteuser,
/usr/sbin/ldapdeleteuserfromgroup,
    /usr/sbin/ldapmodifygroup, /usr/sbin/ldapmodifymachine,
/usr/sbin/ldapmodifyuser,
    /usr/sbin/ldaprenamegroup, /usr/sbin/ldaprenameuser,
/usr/sbin/ldapsetpasswd,
    /usr/sbin/ldapsetprimarygroup
```

Tente criar um novo grupo no LDAP, **sudotest**, e em seguida delete-o.

```
$ sudo ldapaddgroup sudotest
Successfully added group sudotest to LDAP
```

```
$ sudo ldapdeletigroup sudotest
Successfully deleted group cn=sudotest,ou=Groups,dc=intnet from LDAP
```

Tente executar um comando não-autorizado:

```
$ sudo reboot
Sinto muito, usuário chewie não tem permissão para executar "/sbin/reboot" como
root em ldap.intnet.
```

Como **luke**, tente logar diretamente como o **root** usando o **su**.

```
$ hostname ; whoami
ldap
luke
```

```
$ sudo su -
```

```
# whoami  
root
```

6. A máquina **nfs** já está praticamente configurada — a pasta **/config** é local, o que dispensa a montagem automática durante o *boot*, e o **/config/sudoers** já foi configurado e testado nos passos (2) e (3). Resta apenas desabilitar a senha do **root** — faça isso:

```
# hostname ; whoami  
nfs  
root
```

```
# passwd -l root  
passwd: informação de expiração de senha alterada.
```

7. Idealmente, seria interessante que novas máquinas derivadas da VM **debian-template** estivessem automaticamente integradas com o sistema de **sudo** centralizado que acabamos de configurar nesta atividade. Para isso, vamos fazer algumas alterações rápidas na máquina.

No Virtualbox, com a máquina desligada, em *Settings > Network > Adapter 1 > Attached to*, escolha *Host-only Adapter*. O nome da rede *host-only* deve ser o mesmo alocado para a interface de rede da máquina virtual **fw**, configurada durante a sessão 2, que está conectada à DMZ.

Ligue a máquina **debian-template**, e acesse como o usuário **root**.

Reconfigure a rede em **/etc/network/interfaces** para a DMZ, com o endereço IP 10.0.42.250/24:

```
# hostname ; whoami  
debian-template  
root
```

```
# nano /etc/network/interfaces  
(...)
```

```
source /etc/network/interfaces.d/*

auto lo enp0s3

iface lo inet loopback

iface enp0s3 inet static
address 10.0.42.250/24
gateway 10.0.42.1
```

Crie a pasta `/config` e configure sua montagem automática no arquivo `/etc/fstab`:

```
# mkdir /config
```

```
# echo "10.0.42.3:/config /config nfs defaults 0 0" >> /etc/fstab
```

Configure o `symlink` do arquivo `/etc/sudoers`:

```
# mv /etc/sudoers /etc/sudoers.old ; ln -s /config/sudoers /etc/
```

Finalmente, desabilite a senha do usuário `root` — usaremos o `sudo` com o usuário `aluno` para efetuar a configuração inicial das novas máquinas derivadas da VM `debian-template`:

```
# passwd -l root
passwd: informação de expiração de senha alterada.
```

Desligue a VM `debian-template`.

Sessão 5: Registro e correlacionamento de eventos

1) Criação da máquina

Clonar `debian-template` para `log`, IP 10.0.42.4/24. Alterar `hostname` e realizar integração LDAP/SSH-CA como usual.

Criar entrada para o novo *host* no DNS (não esqueça de aumentar o serial), assinar zona e reiniciar *daemons*. Testar.

```
# hostname ; whoami
fw
root
```

```
# grep log /etc/nsd/intnet.zone
log      IN      A              10.0.42.4
```

```
# grep log /etc/nsd/10.0.42.zone
4        IN      PTR              log.intnet.
```

```
# bash /etc/nsd/signzone-intnet.sh
reconfig start, read /etc/nsd/nsd.conf
ok
ok
ok
```

```
# systemctl reload unbound
```

```
# nslookup log
Server:      127.0.0.1
Address:     127.0.0.1#53

Non-authoritative answer:
Name:   log.intnet
Address: 10.0.42.4
```

2) Configuração do NTP

Logar como `root` na máquina `log`. Instalar o OpenNTPD.

```
# hostname ; whoami
log
root
```

```
# apt-get install openntp
```

Fazer o backup do arquivo original e configurar o OpenNTPD.

```
# mv /etc/openntp/ntp.conf /etc/openntp/ntp.conf.orig
```

```
# cat /etc/openntp/ntp.conf
listen on 127.0.0.1
listen on 10.0.42.4
servers pool.ntp.br
```

Pare o OpenNTPD, sincronize o relógio imediatamente.

```
# systemctl stop openntp.service
```

```
# mkdir /var/run/openntp ; ntpd -s -d -f /etc/openntp/ntp.conf -p
/var/run/openntp.pid
adjtimex returns frequency of 0.000000ppm
/var/lib/openntp/db/ntp.drift is empty
listening on 127.0.0.1
listening on 10.0.42.4
ntp engine ready
reply from 200.160.7.186: offset -0.003035 delay 0.018363, next query 8s
set local clock to Sat Nov  3 17:25:41 -03 2018 (offset -0.003035s)
reply from 200.160.7.193: offset -0.003952 delay 0.019983, next query 5s
reply from 200.160.0.8: offset -0.003851 delay 0.020152, next query 7s
reply from 200.186.125.195: offset -0.002323 delay 0.023158, next query 7s
reply from 200.20.186.76: offset -0.003652 delay 0.027538, next query 6s
```

Após a sincronização, pare o processo com **CTRL + C**, apague o diretório `/var/run/openntp` e inicie o OpenNTPD. Cheque se está escutando como esperado.

```
^C ntp engine exiting
Terminating
dispatch_imsig in main: pipe closed
```

```
# rmdir /var/run/openntp
```

```
# systemctl start openntpd.service
```

```
# ss -unlp | grep 123
UNCONN      0      0      10.0.42.4:123      *:*
users:(("ntpd",pid=1643,fd=8))
UNCONN      0      0      127.0.0.1:123      *:*
users:(("ntpd",pid=1643,fd=7))
```

Configurar todos os servidores de uma única vez: como `han@client`, crie o *script* `/home/han/scripts/install-openntpd.sh` com o seguinte conteúdo:

```
1 #!/bin/bash
2
3 # instalacao
4 DEBIAN_FRONTEND=noninteractive apt-get -yq install openntpd
5
6 # configuracao
7 mv /etc/openntpd/ntpd.conf /etc/openntpd/ntpd.conf.orig
8 echo "server 10.0.42.4" > /etc/openntpd/ntpd.conf
9
10 # reiniciar ntpd
11 systemctl restart openntpd.service
```

Executar com:

```
$ for server in fw ldap nfs; do scp ~/scripts/install-openntpd.sh ${server}::~ && ssh
han@${server} 'echo seg10han | sudo -S bash /home/han/install-openntpd.sh' && ssh
${server} rm ~/install-openntpd.sh; done
```

Verifique a correta instalação:

```
$ echo '---'; for server in fw ldap nfs; do ssh han@${server} 'hostname ; dpkg -l |
grep openntpd ; cat /etc/openntpd/ntpd.conf; ps auxwm | pgrep ntpd'; echo '---'; done
---
fw
ii openntpd                1:6.0p1-2                amd64      OpenBSD
NTP daemon
server 10.0.42.4
4950
4954
4956
---
ldap
ii openntpd                1:6.0p1-2                amd64      OpenBSD
NTP daemon
server 10.0.42.4
1354
1364
1366
---
nfs
ii openntpd                1:6.0p1-2                amd64      OpenBSD
NTP daemon
server 10.0.42.4
5192
5193
5194
---
```

3) Registro de comandos digitados com SnoopyLog

Instalar Snoopy para gerar trilha de auditoria de comandos digitados por usuários. Como `han@client`, criar script `/home/han/scripts/install-snoopy.sh` com o seguinte conteúdo:

```
1 #!/bin/bash
2
3 # instalacao
4 DEBIAN_FRONTEND=noninteractive apt-get -yq install snoopy
5
6 # configuracao
7 echo "/lib/snoopy.so" > /etc/ld.so.preload
```

Executar com:

```
$ for server in fw ldap nfs log; do scp ~/scripts/install-snoopy.sh ${server}:~ && ssh
han@${server} 'echo seg10han | sudo -S bash /home/han/install-snoopy.sh' && ssh
${server} rm ~/install-snoopy.sh; done
```


O Snoopy será instalado em todas as VMs da DMZ. Logue em uma VM, faça `sudo` e execute alguns comandos. Verifique o arquivo `/var/log/auth.log` para verificar a funcionalidade do pacote.

4) Correlacionamento de eventos com o Graylog

Desligue a VM `log`, ajuste sua memória para 4 GB e em seguida religue-a. Acesse como `root@log` e instale as dependências do Graylog.

```
# apt-get update
# apt-get install apt-transport-https openjdk-8-jre-headless uuid-runtime pwgen
dirmngr
```

Instale o MongoDB.

```
# apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5
# echo "deb http://repo.mongodb.org/apt/debian stretch/mongodb-org/3.6 main" >
/etc/apt/sources.list.d/mongodb-org-3.6.list
# apt-get update
# apt-get install -y mongodb-org
```

Inicie o serviço do MongoDB.

```
# systemctl daemon-reload
# systemctl enable mongod.service
# systemctl restart mongod.service
```

Instale o Elasticsearch.

```
# wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -
# echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" >
/etc/apt/sources.list.d/elastic-5.x.list
# apt-get update
# apt-get install elasticsearch
```

Configure o Elasticsearch, e inicie-o.

```
# sed -i 's/^#\(\cluster\.name:\).*\/\1 graylog/' /etc/elasticsearch/elasticsearch.yml
# systemctl daemon-reload
# systemctl enable elasticsearch.service
# systemctl restart elasticsearch.service
```

Instale o Graylog.

```
# wget https://packages.graylog2.org/repo/packages/graylog-2.4-repository_latest.deb
# dpkg -i graylog-2.4-repository_latest.deb
# apt-get update
# apt-get install graylog-server
```

Configure as senhas de acesso.

```
# SECRET=$(pwgen -s 96 1) ; sed -i -e 's/password_secret =.*/password_secret =
'$SECRET'/' /etc/graylog/server/server.conf ; unset SECRET
```

```
# PASSWORD=$(echo -n 'rnpesr' | shasum -a 256 | awk '{print $1}') ; sed -i -e
's/root_password_sha2 =.*/root_password_sha2 = '$PASSWORD'/'
/etc/graylog/server/server.conf ; unset PASSWORD
```

Inicie o Graylog.

```
# systemctl daemon-reload
# systemctl enable graylog-server.service
# systemctl start graylog-server.service
```

Instale o nginx para atuar como um proxy reverso.

```
# apt-get install nginx
# rm /etc/nginx/sites-enabled/default
```

Crie o arquivo `/etc/nginx/sites-available/graylog` com o seguinte conteúdo (edite o IP público da máquina `fw` onde apropriado).

```

1 server
2 {
3     listen      80 default_server;
4     listen      [::]:80 default_server ipv6only=on;
5     server_name 200.130.46.146;
6
7     location /
8     {
9         proxy_set_header    Host $http_host;
10        proxy_set_header    X-Forwarded-Host $host;
11        proxy_set_header    X-Forwarded-Server $host;
12        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
13        proxy_set_header    X-Graylog-Server-URL http://200.130.46.146/api/;
14        proxy_pass           http://127.0.0.1:9000;
15    }
16 }

```

```

# ln -s /etc/nginx/sites-available/graylog /etc/nginx/sites-enabled/
# systemctl restart nginx

```

Como **root@fw**:

```

# iptables -t nat -A PREROUTING -i enp0s3 -p tcp --dport 80 -j DNAT --to-destination
10.0.42.4
# iptables-save > /etc/iptables/rules.v4

```

Em sua máquina física, aponte o navegador para <http://200.130.46.146> ; logue como **admin:rnpesr**.
 Ajuste: System > Indices > Edit > Index Shards = 1 > Save. Ajuste: System > Inputs > Syslog UDP ;
 Selecionar node ; Title = syslog ; Bind = 10.0.42.4 ; Port = 5140 ; Save.

Como **han@client**:

```

$ for server in fw ldap nfs log; do scp ~/scripts/configure-graylog.sh ${server}::~ &&
ssh han@${server} 'echo seg10han | sudo -S bash /home/han/configure-graylog.sh' && ssh
${server} rm ~/configure-graylog.sh; done

```

Na máquina física, visualize a aba Search.

Agora, configure a autenticação. Em System > Authentication > LDAP/Active Directory ; Enable LDAP:

Server configuration: Server Address = ldap://10.0.42.2:389 ; System Username = cn=admin,dc=intnet ; System Password = rnpesr

User mapping: Search Base DN = ou=People,dc=intnet ; User Search Pattern = (&(objectClass=posixAccount)(uid={0})); Display Name Attribute = cn

Group mapping: Group Search Base DN = ou=Groups,dc=intnet ; Group Search Pattern = (&(objectClass=posixGroup)(cn=sysadm)) ; Group Name Attribute = cn

Save LDAP Settings. Teste o login como usuários **luke** e **han**.

Baixe o arquivo https://raw.githubusercontent.com/graylog-labs/graylog-contentpack-nginx/master/content_pack.json . Em System > Content Packs > Import Content Pack, aponte o arquivo. Em System > Content Packs > Web Servers > nginx > Apply Content, ative o pack.

Como **root@log**, inserir ao final da seção http no arquivo **/etc/nginx/nginx.conf**:

```
log_format graylog2_format '$remote_addr - $remote_user [$time_local] "$request"
$status $body_bytes_sent "$http_referer" "$http_user_agent" "$http_x_forwarded_for"
<msec=$msec|connection=$connection|connection_requests=$connection_requests|cache_stat
us=$upstream_cache_status|cache_control=$upstream_http_cache_control|expires=$upstream
_http_expires|millis=$request_time>';

access_log syslog:server=10.0.42.4:12301 graylog2_format;
error_log syslog:server=10.0.42.4:12302;
```

```
# systemctl restart nginx
```

No navegador, acesse System > Inputs > nginx access_log > Show received messages.

Para mais informações, acesse <https://devopsideas.com/centralized-logging-using-graylog/> .

Sessão 6: Gestão de configuração

1) Instalação do Ansible

Clonar template para **ansible**, 10.0.42.5/24. Renomear e integrar no LDAP/SSH-ca como de costume. Criar entradas no DNS direto/reverso para **ansible.intnet**.

Como **root@ldap**, crie um usuário para o Ansible, membro dos grupos **setup** e **fwadm**:

```
# ldapadduser ansible setup
# ldapaddusertogroup ansible setup
# ldapaddusertogroup ansible fwadm
```

Como **root@nfs**, permita ao usuário **ansible** executar quaisquer comandos como **root** sem digitar senha:

```
# grep ansible /config/sudoers
ansible    ALL=(ALL:ALL)    NOPASSWD: ALL
```

Como **root@ansible**, instale o Ansible no servidor:

```
# echo "deb http://ppa.launchpad.net/ansible/ansible/ubuntu trusty main" >
/etc/apt/sources.list.d/ansible.list
# apt-get install dirmngr
# apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 93C4A3FD7BB9C367
# apt-get update
# apt-get install ansible
```

2) Execução de comandos simples

Como **ansible@ansible**, assine um par de chaves para logar nos servidores integrados no sistema LDAP/SSH-CA:

```
$ bash scripts/sshsign_user.sh
(sshca@10.0.42.2) Enter passphrase:
# 10.0.42.2:22 SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u4

(CA private key) Enter passphrase: seg10_user_ca
Signed user key id_rsa-cert.pub: id "ansible" serial 0 for ansible valid from 2018-11-
06T13:25:23 to 2021-11-05T12:30:23
```

Crie um diretório **~/ansible**, e um arquivo de hosts identificando as máquinas a gerenciar.

```
$ mkdir ~/ansible
$ cd ~/ansible
```

```
$ nano ~/ansible/hosts
(...)
```

```
$ cat ~/ansible/hosts
[srv]
fw
ldap
nfs
log
ansible
```

Execute um comando simples em todas as máquinas gerenciadas pelo Ansible.

```
$ ansible -i ~/ansible/hosts srv -b --become-user=root -m shell -a 'hostname ; whoami'
ansible | CHANGED | rc=0 >>
ansible
root

fw | CHANGED | rc=0 >>
fw
root

ldap | CHANGED | rc=0 >>
ldap
root

nfs | CHANGED | rc=0 >>
nfs
root

log | CHANGED | rc=0 >>
log
root
```

3) Uso de roles no Ansible

Vamos usar roles (papéis) no Ansible para configurar o sudo de forma local, mais segura. Crie o diretório `~/ansible/roles`, e inicie o papel `sudoers`:

```
$ mkdir ~/ansible/roles
$ cd ~/ansible/roles/
```

```
$ ansible-galaxy init sudoers
- sudoers was created successfully
```

```
$ ls -R sudoers/
sudoers/:
defaults  files  handlers  meta  README.md  tasks  templates  tests  vars

sudoers/defaults:
main.yml

sudoers/files:

sudoers/handlers:
main.yml

sudoers/meta:
main.yml

sudoers/tasks:
main.yml

sudoers/templates:

sudoers/tests:
inventory  test.yml

sudoers/vars:
main.yml
```

Copie o arquivo **sudoers** do NFS para a pasta **files**:

```
$ cp /config/sudoers ~/ansible/roles/sudoers/files/
```

Observe as permissões do arquivo **sudoers** original. Com isso em mente, edite o arquivo **~/ansible/roles/sudoers/tasks/main.yml** como se segue:

```
$ ls -ld /etc/sudoers.old
-r--r----- 1 root root 669 jun  5 2017 /etc/sudoers.old
```

```
$ cat ~/ansible/roles/sudoers/tasks/main.yml
---
- name: Propagate sudoers configuration
  become: yes
  become_user: root
  copy:
    src: sudoers
    dest: /etc
    owner: root
    group: root
    mode: 0440
```

Crie o arquivo `~/ansible/srv.yml` para amarrar os hosts à nova role.

```
$ cat ~/ansible/srv.yml
---
- hosts: srv
  roles:
    - sudoers
```

Execute a role.

```
$ ansible-playbook -i ~/ansible/hosts ~/ansible/srv.yml

PLAY [srv] *****

TASK [Gathering Facts] *****
ok: [nfs]
ok: [ldap]
ok: [fw]
ok: [ansible]
ok: [log]

TASK [sudoers : Propagate sudoers configuration] *****
changed: [fw]
changed: [ldap]
changed: [nfs]
changed: [ansible]
changed: [log]

PLAY RECAP *****
ansible      : ok=2    changed=1    unreachable=0    failed=0
fw           : ok=2    changed=1    unreachable=0    failed=0
ldap         : ok=2    changed=1    unreachable=0    failed=0
log          : ok=2    changed=1    unreachable=0    failed=0
nfs          : ok=2    changed=1    unreachable=0    failed=0
```


Verifique que o arquivo `/etc/sudoers` é lido localmente, agora.

```
$ ls -ld /etc/sudoers  
-r--r----- 1 root root 1392 nov  6 13:50 /etc/sudoers
```

4) Versionamento de configuração com git

Sessão 7: Hardening de sistemas web

Clone www1, 10.0.42.6/24. Clone www2, 10.0.42.7/24. Clone db, 10.0.42.8/24. Clone lb, 10.0.42.9/24.

Configurar DNS.

Como **root@db**:

```
root@db:~# apt-get install mariadb-server --no-install-recommends

MariaDB [(none)]> create database seg10;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> grant all on seg10.* to seg10@localhost identified by 'seg10';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> grant all on seg10.* to seg10@'10.0.42.%' identified by 'seg10';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> quit
Bye
```

```
# sed -i 's/^\(bind-address[\t ]*\=\).*\/\1 0.0.0.0/' /etc/mysql/mariadb.conf.d/50-server.cnf
```

```
# iptables -A INPUT -p tcp --dport 3306 -m iprange --src-range 10.0.42.6-10.0.42.7 -j ACCEPT
# iptables -A INPUT -p tcp --dport 3306 -j DROP
```

```
# systemctl restart mariadb
```

nginx + php-fpm + MariaDB chroot manual balanceamento de carga (HAProxy)

Sessão 8: Isolamento de processos e containerização

Docker Orquestração de containers

Sessão 9: Módulos de segurança do kernel

Recompilação do kernel SELinux AppArmor Tomoyo grsecurity

Sessão 10: Monitoramento de vulnerabilidades

Pentesting automatizado Nessus OVAL/SCAP <https://wiki.debian.org/UsingSCAP>
<https://wiki.debian.org/SCAPGuide>