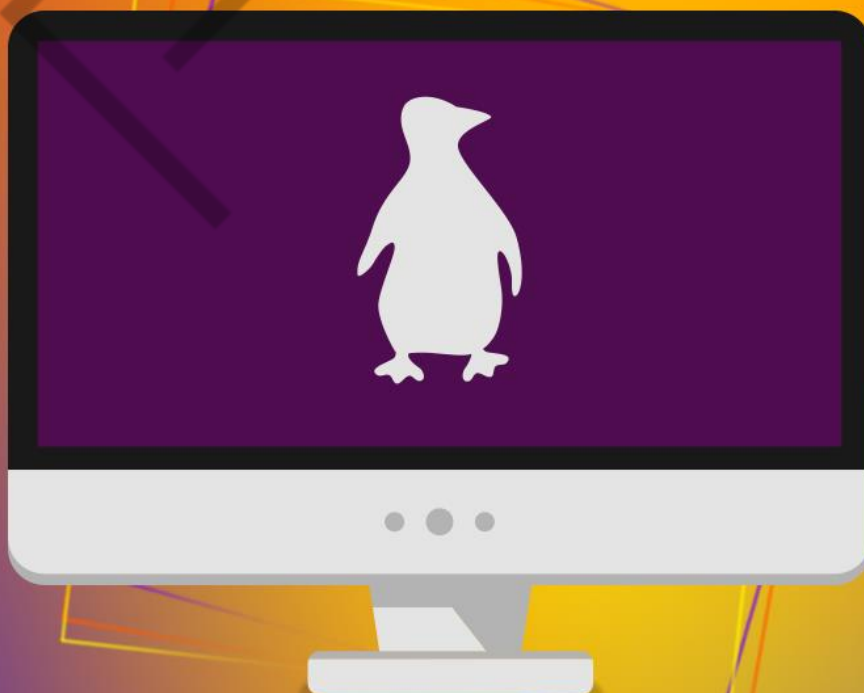


LINUX FUNDAMENTOS

RECURSOS AVANÇADOS

HENRIQUE POYATOS



3

LISTA DE FIGURAS

Figura 3.1– Criar um novo volume no Oracle VirtualBox.....	7
Figura 3.2 – Diferentes partições de sistemas operacional em um mesmo volume ..	8
Figura 3.3 – Procurar por “sdb” no syslog e diretório /dev.....	10
Figura 3.4 – Exibir as partições de /dev/sda com fdisk.....	11
Figura 3.5 – Procurar por sda1 no arquivo /etc/mstab	12
Figura 3.6 – Criar duas partições com o comando fdisk.....	13
Figura 3.7 – Exibir as partições e salvando a tabela com o comando fdisk	14
Figura 3.8 – Criar um sistema de arquivos commkfs.....	15
Figura 3.9 – Montagem de volume usando o comando mount.....	16
Figura 3.10 – Desmontagem de volume com o comando umount	17
Figura 3.11 – O arquivo /etc/fstab	18
Figura 3.12 – Montar /dev/sdb1 automaticamente em /etc/fstab.....	20
Figura 3.13 – Verificar um sistema de arquivos com fsck	21
Figura 3.14 – Rodar o comando date antes e dentro do comando tar	22
Figura 3.15 – Agendar o backup utilizando o comando crontab.....	23
Figura 3.16 – Conteúdo do arquivo /etc/crontab	24
Figura 3.17 – Fluxo de Processo do iptables	26
Figura 3.18 – Colocar o firewall no ar.....	28
Figura 3.19 – Acessar o servidor web localmente com o comando links	29
Figura 3.20 – Instalar o serviço knockd.....	31
Figura 3.21 – Verificar e testando acesso na porta 22 do ssh.....	34
Figura 3.22 – Port-knocking realizado com sucesso	35

LISTA DE QUADROS

Quadro 3.1 – Opções de montagem de partições	19
Quadro 3.2 – Correntes de regraspadrão no iptables	25
Quadro 3.3 – Ações que podem ser realizadas no iptables	26
Quadro 3.4 – Principais parâmetros do comando iptables	27

EMANIP

LISTA DE CÓDIGOS-FONTE

Código-fonte 3.1 – Arquivo /etc/firewall.sh, primeira versão.....	28
Código-fonte 3.2 – Arquivo /etc/firewall.sh, segunda versão.....	30
Código-fonte 3.3 – O arquivo /etc/firewall.sh.....	32
Código-fonte 3.4 – O arquivo /etc/knockd.conf com uma ligeira alteração.....	33
Código-fonte 3.5 – O arquivo /etc/default/knock alterado.....	33
Código-fonte 3.6 – Shell script que realiza o port-knocking.....	34

EXEMPLO

SUMÁRIO

3 RECURSOS AVANÇADOS.....	6
3.1 Gerenciamento de volumes.....	6
3.1.1 Um pouco sobre partições.....	7
3.1.2 Um pouco sobre sistema de arquivos	8
3.1.3 Gerenciamento partições – fdisk	9
3.1.4 Criar um sistema de arquivos – mkfs	14
3.1.5 Montar volumes – mount	15
3.1.6 Desmontar volumes – umount.....	17
3.1.7 Montagem automática de volumes.....	17
3.1.8 Verificar um sistema de arquivos – fsck	20
3.2 Agendamento de tarefas	21
3.2.1 Agendando tarefas – cron, anacron e crontab	21
3.3 Política de <i>firewall</i>	25
3.3.1 Filtragem de pacotes – iptables.....	27
3.3.2 Port-knocking	31
3.4 Considerações finais	35
REFERÊNCIAS	37
GLOSSÁRIO	38

3 RECURSOS AVANÇADOS

Conforme prometido em nosso último capítulo sobre os comandos Linux, vamos abordar agora a possibilidade de gerenciar volumes, agendar tarefas e filtrar pacotes (a instalação de *firewall*, incluindo *port-knocking*) no sistema operacional do pinguim. Venha conosco!

3.1 Gerenciamento de volumes

O gerenciamento de volumes para armazenamento é um conhecimento importante para qualquer administrador Linux. Cada disco rígido, seja HDD ou SSD, *pendrive*, HD externo ou disco óptico (CD-ROMs, DVD-ROMs ou Blurays) são tratados e identificados pelo sistema como volumes e, diferente do que estamos acostumados em um sistema operacional Windows, do qual, cada um dos volumes é identificado com uma letra e possui sua própria raiz de arquivos e diretórios, em sistemas do tipo **nix*, esta raiz é única e os diferentes volumes serão “pendurados” nesta árvore no ponto que achamos mais conveniente.

Com o intuito de similar esta situação, vamos criar uma unidade de armazenamento na instância Debian em nosso *Oracle VirtualBox*. Com a instância desligada, basta clicar em “Configurações”, escolher “Armazenamento” e clicar no botão do disco rígido com o sinal de mais e, na sequência, seguir escolhendo as opções pertinentes no assistente que se abre. Procurei criar um disco no padrão VDI (próprio do VirtualBox), dinamicamente alocado de tamanho máximo de 8GB, mas você pode variar as escolhas conforme julgar mais adequado (Vide Figura “– Criar um novo volume no Oracle VirtualBox”).

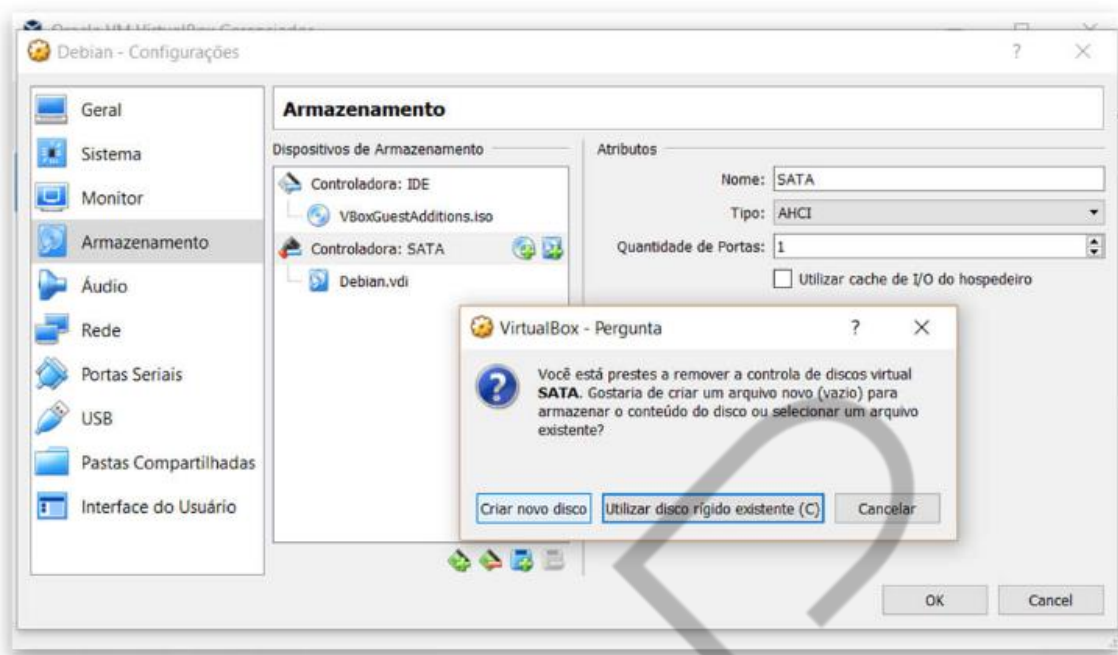


Figura 3.1– Criar um novo volume no Oracle VirtualBox
Fonte: Elaborado pelo autor (2018)

3.1.1 Um pouco sobre partições

Vamos falar um pouco de como estas unidades de armazenamento são divididas. Um único dispositivo físico como um disco rígido ou *pendrive* pode ser dividido em várias partes do ponto de vista lógico, ou seja, por *software*; as razões para isso podem ser diversas: maior organização, maior segurança (pois é possível estabelecer regras de acesso distintas para diferentes partições) ou mesmo *performance* (algumas operações realizadas pelos sistemas de arquivos são mais velozes em partições menores do que maiores).

Outra boa razão para dividir um disco rígido em diferentes partições é a possibilidade de se instalar sistemas operacionais diferentes no mesmo *hardware*, ou seja, se temos Windows e Linux em um mesmo equipamento (e disco rígido) instalados diretamente no equipamento físico (não estamos falando de máquinas virtuais neste caso), é graças à possibilidade de particionar volumes de armazenamento.



Figura 3.2 – Diferentes partições de sistemas operacional em um mesmo volume
Fonte: Google Imagens (2018)

3.1.2 Um pouco sobre sistema de arquivos

Uma vez criada(s) a(s) partiç(ões) de um volume de armazenamento, faz-se necessário gerenciar como esta informação ficará armazenada. Por exemplo, em um disco rígido tradicional, formado por discos empilhados e agulhas de leitura como pode ser visto na Figura “Diferentes partições de sistemas operacional em um mesmo volume”, temos cilindros, setores e cabeças de leitura diferentes; dependendo de onde a informação está armazenada, e velocidade de acesso ou mesmo gravação da informação será diferente. Além disso, se desejo gravar um arquivo grande de, digamos, 5GB e não há 5GB continuamente livres no disco, este arquivo será quebrado em partes que ocuparão lugares diferentes neste disco.

Quem organiza a estratégia de leitura e gravação em um disco, além de criar os conceitos de arquivos e diretórios para organizar as informações é um componente importante do sistema operacional chamado sistema de arquivos. Sim, o sistema de arquivos é parte de um sistema operacional e trata-se desta camada que é instalada nas partições, transformando arquivos e diretórios em blocos de informação e vice-versa.

Existem dezenas de sistemas de arquivos diferentes. No caso do Windows, os mais comuns são o FAT32 (sigla para File AllocationTable ou Tabela de Alocação de Arquivos), seguido pelo sucoessorex FAT (Extended File AllocationTable ou Tabela de Alocação de Arquivos Estendida ou ainda FAT64) e o NTFS (New Technology

File System). O NTFS é o sistema de arquivos da Microsoft mais seguro entre as três opções e as versões mais atuais do Windows serão sempre instaladas em uma participação com NTFS. Embora possua uma série de desvantagens e restrições à segurança, o FAT32 é um sistema de arquivos fácil de ser implementado e embarcado, sendo reconhecido pelos mais diferentes periféricos como SmartTVs, sons automotivos e outros dispositivos que façam leitura de pendrives através de suas portas USB.

No sistema operacional Linux ou ext4, é a quarta versão de sistema de arquivos e a mais utilizada pelo sistema. No entanto, temos outras possibilidades como o ReiserFS, Btrfs, XFS, ZFS entre dezenas de opções.

3.1.3 Gerenciamento partições – fdisk

Criamos um volume de armazenamento em nosso Oracle VirtualBox que simulará a instalação física de um novo disco rígido, vamos aprender a criar partições, este volume que se encontra virgem para gravação.

Como foi dito, qualquer periférico detectado pelo sistema operacional Linux se torna um pseudo arquivo no diretório **/dev**. Geralmente, discos rígidos são identificados como **/dev/sd***, sendo **/dev/sda** o primeiro disco encontrado, **/dev/sdb** o segundo, assim por diante.

Ao buscar por “sdb” no **/var/log/syslog** ou no diretório **/dev**, tenho os seguintes resultados (Vide Figura “Procurar por “sdb” no syslog e diretório /dev”):

```

hpoyatos@debian:~$ sudo grep sdb /var/log/syslog*
/var/log/syslog.1:Apr 14 09:53:04 debian kernel: [ 2.180180] sd 1:0:0:0: [sdb] 16777216 512-byte logical
blocks: (8.59 GB/8.00 GiB)
/var/log/syslog.1:Apr 14 09:53:04 debian kernel: [ 2.180204] sd 1:0:0:0: [sdb] Write Protect is off
/var/log/syslog.1:Apr 14 09:53:04 debian kernel: [ 2.180206] sd 1:0:0:0: [sdb] Mode Sense: 00 3a 00 00
/var/log/syslog.1:Apr 14 09:53:04 debian kernel: [ 2.180217] sd 1:0:0:0: [sdb] Write cache: enabled, rea
d cache: enabled, doesn't support DPO or FUA
/var/log/syslog.1:Apr 14 09:53:04 debian kernel: [ 2.181407] sd 1:0:0:0: [sdb] Attached SCSI disk
/var/log/syslog.1:Apr 15 12:39:23 debian kernel: [ 1.754041] sd 1:0:0:0: [sdb] 16777216 512-byte logical
blocks: (8.59 GB/8.00 GiB)
/var/log/syslog.1:Apr 15 12:39:23 debian kernel: [ 1.754102] sd 1:0:0:0: [sdb] Write Protect is off
/var/log/syslog.1:Apr 15 12:39:23 debian kernel: [ 1.754109] sd 1:0:0:0: [sdb] Mode Sense: 00 3a 00 00
/var/log/syslog.1:Apr 15 12:39:23 debian kernel: [ 1.754135] sd 1:0:0:0: [sdb] Write cache: enabled, rea
d cache: enabled, doesn't support DPO or FUA
/var/log/syslog.1:Apr 15 12:39:23 debian kernel: [ 1.755655] sd 1:0:0:0: [sdb] Attached SCSI disk
hpoyatos@debian:~$
hpoyatos@debian:~$ ls -la /dev/sd*
brw-rw---- 1 root disk 8, 0 abr 15 12:39 /dev/sda
brw-rw---- 1 root disk 8, 1 abr 15 12:39 /dev/sda1
brw-rw---- 1 root disk 8, 2 abr 15 12:39 /dev/sda2
brw-rw---- 1 root disk 8, 5 abr 15 12:39 /dev/sda5
brw-rw---- 1 root disk 8, 16 abr 15 12:39 /dev/sdb
hpoyatos@debian:~$

```

Figura 3.3 – Procurar por “sdb” no syslog e diretório /dev
 Fonte: Elaborado pelo autor (2018)

Repare meu segundo disco rígido, identificado com sucesso pelo sistema Linux, mostrando até mesmo o tamanho máximo (8GB) no **syslog**. Outro fator interessante é o resultado da listagem de /dev/sd*: o primeiro disco (sda) está presente em **/dev/sda**, além de outros três arquivos: **/dev/sda1**, **/dev/sda2** e **/dev/sda5**. Mas o que seriam estes três?

Pois bem, estes três arquivos representam as três partições do primeiro disco. Sim, quando instalei o Linux pela primeira vez nesta instância do Oracle VirtualBox, dividi o primeiro volume em três partições distintas. A numeração “salta” de 2 para 5 pois as partições enumeradas com “1” e “2” são consideradas partições primárias, enquanto **/dev/sda5** é a primeira partição estendida.

Antes de dividir o segundo volume (sdb) em partições com o comando **fdisk**, utilizarei este mesmo comando para mostrar as partições existentes no primeiro volume (sda). Basta digitar **sudofdisk /dev/sda** e a letra “p” (de *print table* ou imprimir tabela de partições):

```
hpyatos@debian:~$ sudo fdisk /dev/sda

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): p
Disco /dev/sda: 20 GiB, 21474836480 bytes, 41943040 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: 0x3082ec8e

Dispositivo Inicializar  Início      Fim      Setores Tamanho Id Tipo
/dev/sda1  *                2048 33554431 33552384    16G 83 Linux
/dev/sda2                33556478 41940991 8384514      4G  5 Estendida
/dev/sda5                33556480 41940991 8384512      4G 82 Linux swap / Solaris

Comando (m para ajuda): q

hpyatos@debian:~$
```

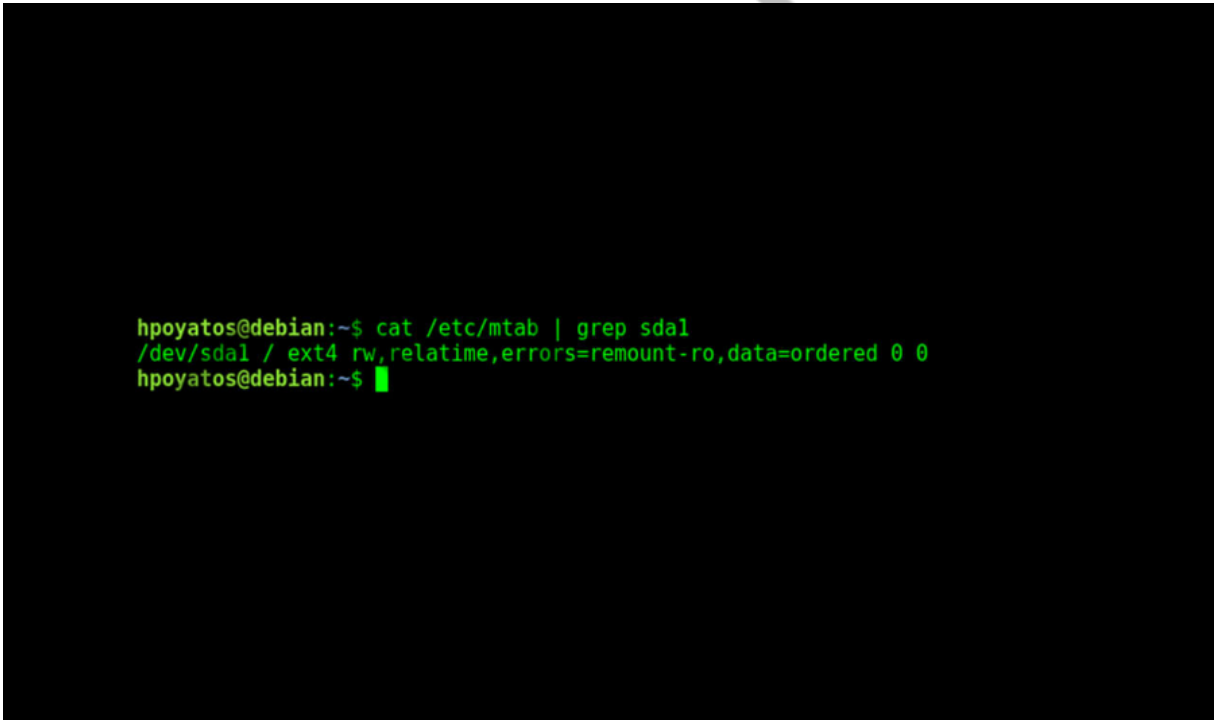
Figura 3.4 – Exibir as partições de /dev/sda com fdisk
Fonte: Elaborado pelo autor (2018)

Podemos reparar nas colunas três e quatro (respectivamente “Início” e “Fim”) o bloco inicial e final de cada partição, e destacamos que:

- A primeira partição nunca começa no primeiro setor; inicia sempre em 2048; os mais de dois mil setores anteriores guardam a própria tabela de partições em si, além de instruções que indiquem em qual partição estão sistema operacional. Quando temos mais um sistema operacional presente no disco, os menus que nos permitam escolher qual queremos usar também ficam nestes setores iniciais que são chamados de MBR ou *Masterboot*.
- As partições **/dev/sda2** e **/dev/sda5** são do mesmo tamanho e ocupam praticamente os mesmos setores (sda5 começa dois setores depois). Isso acontece, pois, sda2 é uma partição estendida, que é uma partição que permite ser subdivida em outras partições. Ou seja, sda5 fica dentro de sda3. Na prática, temos apenas duas partições em uso.
- **/dev/sda1** é a partição principal onde o Linux está instalado, enquanto /dev/sda5 é uma partição de swap; este tipo de partição entra em ação

quando a memória RAM do equipamento está completamente tomada por processos e, para abrir novos processos, o sistema operacional começa a descer informações da memória secundária (RAM) para a memória terciária (HD), e subi-los novamente quando são necessários, realizando trocas ou *swapping*. Do contrário, o sistema operacional se recusaria a abrir novos programas por falta de memória.

Caso queira saber qual sistema de arquivos o Linux está usando, o arquivo **/etc/mtab** mostra a tabela de todos os volumes de armazenamento ativos neste momento, juntamente com seus sistemas de arquivos e regras. Na Figura “*Procurar por sda1 no arquivo /etc/mtab*” investiguei e concluí que **/dev/sda1** está usando **ext4**:



```
hpoyatos@debian:~$ cat /etc/mtab | grep sda1
/dev/sda1 / ext4 rw,relatime,errors=remount-ro,data=ordered 0 0
hpoyatos@debian:~$
```

Figura 3.5 – Procurar por sda1 no arquivo /etc/mtab
Fonte: Elaborado pelo autor (2018)

Logo depois do dispositivo (**/dev/sda1**) temos o que é chamado de ponto de montagem, ou seja, onde aquele volume está ativo na raiz de arquivos; a resposta para **/dev/sda1** é **/**, ou seja, é a raiz de nossa árvore e, por esta razão, é o que deve ser montado primeiro; todos os outros dispositivos serão “pendurados” ou montados na árvore iniciada por **/dev/sda1**.

Agora que sabemos mais sobre partições, vamos usar o comando `fdisk` para criar duas partições em `/dev/sdb`. Para tal, comece com o comando `sudo fdisk /dev/sdb` e no menu do comando, digite “n” (de *new*, para criar uma partição) e seguir as instruções (Figura “Criar duas partições com o comando *fdisk*”):

```
hpoyatos@debian:~$ sudo 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 0xfe7deaf6.

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): p
Número da partição (1-4, padrão 1): 1
Primeiro setor (2048-16777215, padrão 2048):
Último setor, +setores ou +tamanho{K,M,G,T,P} (2048-16777215, padrão 16777215): +4G

Criada uma nova partição 1 do tipo "Linux" e de tamanho 4 GiB.

Comando (m para ajuda): n
Tipo da partição
  p primária (1 primárias, 0 estendidas, 3 livre)
  e estendida (recipiente para partições lógicas)
Selecione (padrão p): p
Número da partição (2-4, padrão 2): 2
Primeiro setor (8390656-16777215, padrão 8390656):
Último setor, +setores ou +tamanho{K,M,G,T,P} (8390656-16777215, padrão 16777215):

Criada uma nova partição 2 do tipo "Linux" e de tamanho 4 GiB.

Comando (m para ajuda):
```

Figura 3.6 – Criar duas partições com o comando `fdisk`
Fonte: Elaborado pelo autor (2018)

As duas partições foram criadas como primárias (Letra “p” e tipo da partição, podemos ter quatro partições primárias). Como esperado, o primeiro setor disponível para a primeira partição era 2048 (os setores anteriores ficam reservados ao MBR) e, no lugar do último setor, ao invés de pensarmos o tamanho de partição em setores, o `fdisk` permite informar seu tamanho usando `+tamanho{K,M,G,T,P}`, ou seja, `+4G` significa o desejo de criar uma partição de 4 gigabytes, enquanto `+500M` criaria uma de 500 megabytes, `+1T`, um terabyte, e assim por diante.

Por padrão, as duas partições foram criadas como tipo “Linux” assim, espera-se que se use sistemas de arquivo como o `ext4` ou `ReiserFS`. Caso você queira criar partições para outros fins ou sistemas operacionais, a letra “t” seguida da letra “L” permite mudar o tipo facilmente.

Finalizaremos com a letra “p” para listar as partições recém-criadas e a letra “w” (de *write*) para gravar a nova tabela de partições e sair (a letra “q” abandona o comando fdisk sem realizar qualquer alteração):

```
Comando (m para ajuda): p
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: 0xfe7deaf6

Dispositivo Inicializar  Início      Fim Setores Tamanho Id Tipo
/dev/sdb1          2048    8390655 8388608      4G 83 Linux
/dev/sdb2          8390656 16777215 8386560      4G 83 Linux

Comando (m para ajuda): w
A tabela de partição foi alterada.
Chamando ioctl() para reler tabela de partição.
Sincronizando discos.

hpoyatos@debian:~$ ls /dev/sdb*
/dev/sdb /dev/sdb1 /dev/sdb2
hpoyatos@debian:~$
```

Figura 3.7 – Exibir as partições e salvando a tabela com o comando fdisk
Fonte: Elaborado pelo autor (2018)

3.1.4 Criar um sistema de arquivos – mkfs

Embora o volume sdb possua duas partições de 4GB do tipo Linux, estas não possuem nenhum sistema de arquivos que nos permita criar arquivos ou diretórios. Para tal, podemos usar o comando **mkfs** (abreviação de *makefilesystem* ou criar sistema de arquivo):


```
hpoyatos@debian:~$ sudo mkfs -t ext4 /dev/sdb1
mke2fs 1.43.4 (31-Jan-2017)
Creating filesystem with 1048576 4k blocks and 262144 inodes
Filesystem UUID: 10ca8b3a-c814-4de9-aebd-b346b4f83bd7
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

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

hpoyatos@debian:~$ sudo mkfs.ext4 /dev/sdb2
mke2fs 1.43.4 (31-Jan-2017)
Creating filesystem with 1048320 4k blocks and 262144 inodes
Filesystem UUID: 7a6218d1-b02a-476f-a2c9-c9c90588c4ff
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

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

hpoyatos@debian:~$ sudo mkfs.
mkfs.bfs    mkfs.exfat  mkfs.ext3   mkfs.fat    mkfs.msdos  mkfs.vfat
mkfs.cramfs mkfs.ext2   mkfs.ext4   mkfs.minix  mkfs.ntfs
hpoyatos@debian:~$ sudo mkfs.
```

Figura 3.8 – Criar um sistema de arquivos com mkfs
Fonte: Elaborado pelo autor (2018)

Como pode ser visto na Figura “Criar um sistema de arquivos com mkfs”, o sistema de arquivo desejado pode ser informado com o parâmetro “-t” ou com um atalho; no caso do ext4, o atalho “**mkfs.ext4**” seguido da partição é o suficiente para criar o sistema de arquivos. Ao digitar `sudo mkfs.` e a tecla [TAB] podemos ver as opções de sistemas de arquivo disponíveis no Linux que estamos.

3.1.5 Montar volumes – mount

O comando `mount` é utilizado para montar as partições na raiz de diretórios. Para tal, basta indicar a partição e o ponto de montagem, ou seja, a partir de qual diretório queremos que o volume atue.

```

hpooyatos@debian:~$ sudo mkdir /media/novoDisco
hpooyatos@debian:~$ sudo mount /dev/sdb1 /media/novoDisco/
hpooyatos@debian:~$ df -h
Sist. Arq.      Tam. Usado Disp. Uso% Montado em
udev           2,0G      0  2,0G   0% /dev
tmpfs          396M    6,0M  390M   2% /run
/dev/sdal       16G    4,8G   11G  32% /
tmpfs          2,0G      0  2,0G   0% /dev/shm
tmpfs          5,0M    4,0K   5,0M   1% /run/lock
tmpfs          2,0G      0  2,0G   0% /sys/fs/cgroup
tmpfs          396M    32K  396M   1% /run/user/117
tmpfs          396M    28K  396M   1% /run/user/1000
/dev/sr0        56M    56M     0 100% /media/cdrom0
/dev/sdb1       3,9G    16M   3,7G   1% /media/novoDisco
hpooyatos@debian:~$ █

```

Figura 3.9 – Montagem de volume usando o comando mount
Fonte: Elaborado pelo autor (2018)

Repare na Figura “*Montagem de volume usando o comando mount*”, que um novo diretório foi criado (/media/novoDisco) para servir de ponto de montagem; o comando **mount** realizou a montagem neste diretório com sucesso, e isso pode ser verificado tanto ao olhar o conteúdo do arquivo /etc/mtab descrito anteriormente quanto com o comando **df** que, com seu parâmetro “-h”, permite visualizar o tamanho dos volumes de uma maneira mais amigável.

Podemos observar também que, embora a partição /dev/sdb1 de 4GB nunca tenha sido usada, esta começa com alguma ocupação (16M ou dezesseis megabytes) e nos permite a utilização de 3,7 gigabytes. Isso acontece, pois, o próprio sistema de arquivos precisa de um percentual desta partição para funcionar adequadamente, e por esta razão que nunca conseguiu usar todo o espaço que a embalagem que seu *pendrive* diz que tem à disposição.

O comando *mount* conta com um parâmetro “-o” da qual é possível parametrizar a montagem do dispositivo. Vide o Quadro “*Opções de montagem de partições*” para saber as opções.

3.1.6 Desmontar volumes – umount

O comando **umount** (que vem da palavra *unmount* ou desmontagem) permite desmontar um volume da raiz de diretórios. Para tal, basta informar a partição ou ponto de montagem, como pode ser visto na Figura “Desmontagem de volume com o comando umount”:

```

hpooyatos@debian:~$ df -h
Sist. Arq. Tam. Usado Disp. Uso% Montado em
udev      2,0G    0  2,0G    0% /dev
tmpfs     396M    6,0M  390M    2% /run
/dev/sda1  16G    4,8G   11G   32% /
tmpfs     2,0G    0  2,0G    0% /dev/shm
tmpfs     5,0M    4,0K  5,0M    1% /run/lock
tmpfs     2,0G    0  2,0G    0% /sys/fs/cgroup
tmpfs     396M   32K  396M    1% /run/user/117
tmpfs     396M   28K  396M    1% /run/user/1000
/dev/sr0   56M    56M    0 100% /media/cdrom0
/dev/sdb1  3,9G   16M  3,7G    1% /media/novoDisco
hpooyatos@debian:~$
hpooyatos@debian:~$
hpooyatos@debian:~$ sudo umount /media/novoDisco
hpooyatos@debian:~$
hpooyatos@debian:~$ df -h
Sist. Arq. Tam. Usado Disp. Uso% Montado em
udev      2,0G    0  2,0G    0% /dev
tmpfs     396M    6,0M  390M    2% /run
/dev/sda1  16G    4,8G   11G   32% /
tmpfs     2,0G    0  2,0G    0% /dev/shm
tmpfs     5,0M    4,0K  5,0M    1% /run/lock
tmpfs     2,0G    0  2,0G    0% /sys/fs/cgroup
tmpfs     396M   32K  396M    1% /run/user/117
tmpfs     396M   28K  396M    1% /run/user/1000
/dev/sr0   56M    56M    0 100% /media/cdrom0
hpooyatos@debian:~$
  
```

Figura 3.10 – Desmontagem de volume com o comando umount
Fonte: Elaborado pelo autor (2018)

3.1.7 Montagem automática de volumes

Embora o comando **mount** nos permita a montagem de partições com facilidade, na maioria dos casos precisamos que esta montagem aconteça automaticamente.

Quando queremos que certas partições sejam montadas no boot do sistema operacional, basta informá-la no arquivo **/etc/fstab**:

```

hpooyatos@debian:~$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=74b2f273-f38e-4dff-8598-184636436508 / ext4 errors=remount-ro 0 1
# swap was on /dev/sda5 during installation
UUID=5b47c985-4e5d-41e1-9f2f-b82a99ff79da none swap sw 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
hpooyatos@debian:~$

```

Figura 3.11 – O arquivo /etc/fstab
Fonte: Elaborado pelo autor (2018)

Sendo:

- **Primeira coluna – partição a ser montada:** a primeira informação é a partição que será montada. Em nosso caso, pode ser /dev/sdb1; repare que os dois exemplos no arquivo não estão usando a partição como sendo /dev/sda1, e sim seu UUID. É possível usar também o rótulo (LABEL) do volume. A razão pela qual o Linux está usando o UUID é que antes de /dev/sda1 ser montado, não existe raiz de diretórios e, por coerência, também não existe um diretório /dev. Indicar a partição como ela aparece na raiz seria impossível.
- **Segunda coluna – ponto de montagem:** assim como no comando mount, indicamos onde queremos que o volume atue.
- **Terceira coluna – tipo:** aqui informamos qual sistema de arquivos estamos utilizando; em nosso caso, ext4.
- **Quarta coluna – opções:** conforme dito, a partição pode ser montada de várias formas diferentes, conforme Quadro “Opções de montagem de partições”:

Opção	Descrição
defaults	Usar opções padrão: rw, suid, dev,exec e async
rw	Partição montada para leitura e escrita
ro	Partição montada em somente leitura
exec	Permitir execução de binários na partição montada
noexec	Não permite execução de binários na partição montada
suid	Permite o uso de um bit no arquivo ou diretório para identificação de usuário
noauto	Não montá-la automaticamente no ato do boot (com mount -a)
user	Permitir que um usuário monte a partição
owner	Permitir que o dono do dispositivo possa montá-lo
nofail	Não reporte caso a montagem da partição apresente erro

Quadro 3.1 – Opções de montagem de partições
 Fonte: Elaborado pelo autor (2018)

- **Quinta coluna – dump:** esta coluna pode ser usada para indicar ao comando **dump** quais partições deve fazer *backup*; o comando **dump** faz *backup* em um nível de bloco. A opção padrão é “0” que significa “não faça o dump”.
- **Sexta coluna – ordem de verificação:** esta coluna pode ser usada para indicar ao comando **fsck** a ordem que deve verificar as partições no ato do *boot*. Exemplo: se defino uma partição como “1”, as partições que estão com “0” são verificadas antes da minha e assim por diante.

Sendo assim, com alguns poucos comandos conforme mostrados na Figura “Montar /dev/sdb1 automaticamente em /etc/fstab”, podemos montar /dev/sdb1 automaticamente no *boot* do sistema:

```

hpoyatos@debian:~$ su -
Senha:
root@debian:~# echo "/dev/sdb1 /media/novoDisco ext4 defaults 0 0" >> /etc/fstab
root@debian:~# exit
sair
hpoyatos@debian:~$ cat /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sdal during installation
UUID=74b2f273-f38e-4dff-8598-184636436508 / ext4 errors=remount-ro 0 1
# swap was on /dev/sda5 during installation
UUID=5b47c985-4e5d-41e1-9f2f-b82a99ff79da none swap sw 0 0
/dev/sr0 /media/cdrom0 udf,iso9660 user,noauto 0 0
/dev/sdb1 /media/novoDisco ext4 defaults 0 0
hpoyatos@debian:~$

```

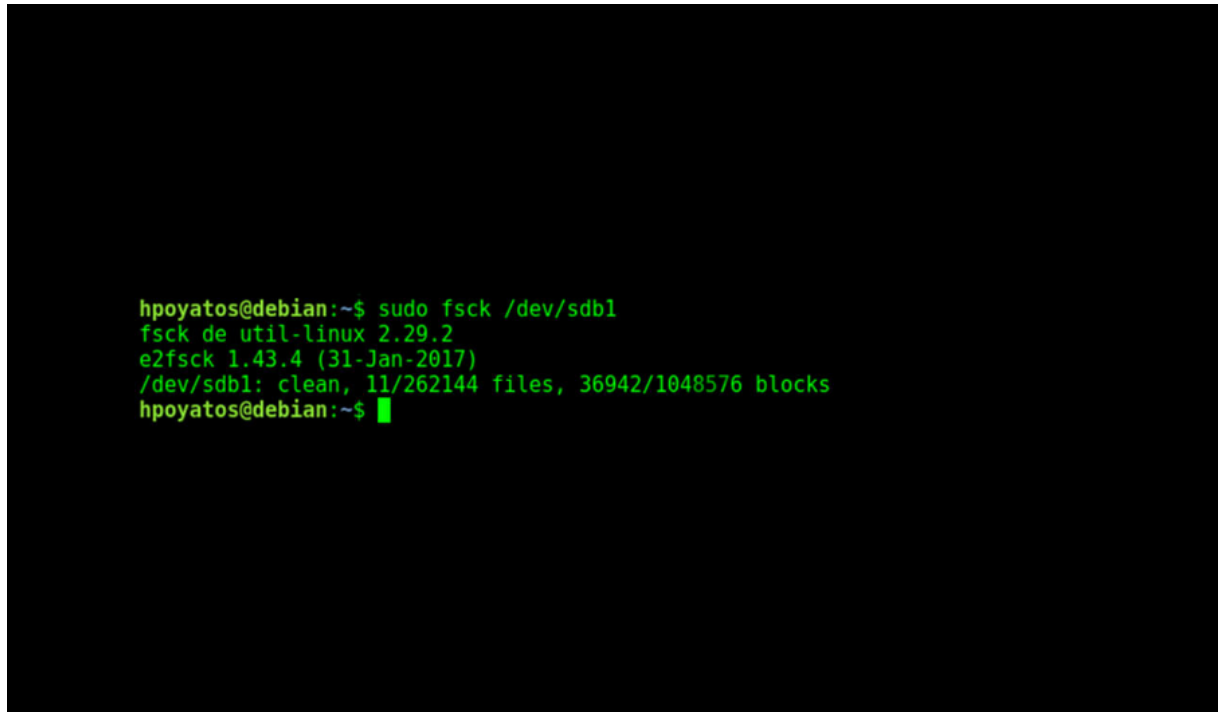
Figura 3.12 – Montar /dev/sdb1 automaticamente em /etc/fstab
 Fonte: Elaborado pelo autor (2018)

Para casos como discos ópticos e *pendrives*, o Linux conta com um poderosíssimo gerenciador dinâmico de dispositivos do Linux conhecido como **udev**, que teve como antecessores **DEVFS** e **hotplug** que foram combinados para formar este novo gerenciador. O udev é o componente que permite o *plug and play* no Linux, e no caso de dispositivos de armazenamento, algumas regras podem ser definidas em **/etc/udev/rules.d** para automatizar seu uso. Geralmente, regras básicas vêm instaladas em qualquer distribuição Linux moderna, como montar o *pendrive* automaticamente em /media/<label do pendrive>. No entanto, se queremos que um *pendrive* em específico monte em outro lugar e/ou execute algum aplicativo automaticamente, é com estas regras que isso será feito.

3.1.8 Verificar um sistema de arquivos – fsck

O comando **fsck** (abreviação de *filesystemcheck* ou verificação de sistema de arquivos) permite uma verificação “na saúde” do sistema de arquivos presente na partição. Este deve ser usado sempre que o sistema de arquivos esteja corrompido por alguma razão, e a partição não pode ser montada e, portanto, utilizada. A execução de fsck em uma partição com sistema de arquivos corrompido resolverá o

problema e permitirá a montagem. Este também é executado automaticamente no *boot* do sistema operacional Linux para verificar as partições de forma preventiva.

A terminal window with a black background and green text. The text shows a user named 'hpoyatos' at a 'debian' prompt running the command 'sudo fsck /dev/sdb1'. The output shows the version of fsck (2.29.2), the version of e2fsck (1.43.4), and the results of the check: '/dev/sdb1: clean, 11/262144 files, 36942/1048576 blocks'. The prompt returns to the user.

```
hpoyatos@debian:~$ sudo fsck /dev/sdb1
fsck de util-linux 2.29.2
e2fsck 1.43.4 (31-Jan-2017)
/dev/sdb1: clean, 11/262144 files, 36942/1048576 blocks
hpoyatos@debian:~$
```

Figura 3.13 – Verificar um sistema de arquivos com fsck
Fonte: Elaborado pelo autor (2018)

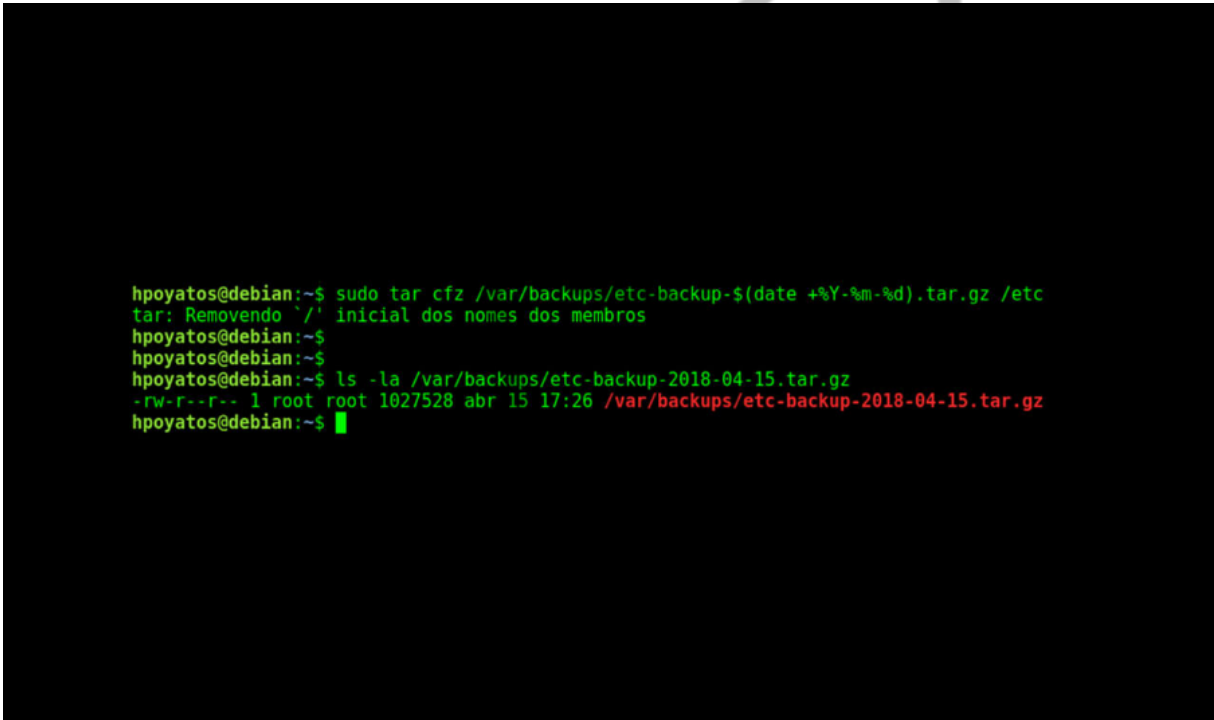
3.2 Agendamento de tarefas

É uma necessidade comum agendar tarefas em um sistema operacional, que precisem ser executadas no futuro ou regularmente de tempos em tempos. Quando isso acontecer, pode utilizar o **cron**, criado no Unix e portado para o Linux exatamente para este fim.

3.2.1 Agendando tarefas – cron, anacron e crontab

Para fins de exemplo, imagine que precisemos realizar *backups* diários do diretório **/etc** (responsável pelas configurações gerais do sistema) em um arquivo do tipo *tar.gz* ou *.tgz*. Para tal, precisaremos do comando *tar*, *gzip* e um lugar para armazenar estes arquivos e, para este caso, usaremos o **/var/backups** que existe na árvore de diretórios do Ubuntu.

Outro pré-requisito de nosso exemplo é que desejo acumular estes backups; não basta ter um backup do que o diretório **/etc** tinha ontem, se quisermos arquivos de uma semana, três semanas ou 23 dias atrás exatamente, será possível. Para tal, o jeito mais fácil e organizado é colocar a data do *backup* no nome do arquivo, só que isso precisa ser feito de forma dinâmica. Na Figura “Rodar o comando *date* antes e dentro do comando *tar*” testamos esta necessidade e o “truque” é executar o comando **date** ANTES e DENTRO do comando **tar**, bem no ponto que queremos a data:



```
hpoyatos@debian:~$ sudo tar cfz /var/backups/etc-backup-$(date +%Y-%m-%d).tar.gz /etc
tar: Removendo '/' inicial dos nomes dos membros
hpoyatos@debian:~$
hpoyatos@debian:~$
hpoyatos@debian:~$ ls -la /var/backups/etc-backup-2018-04-15.tar.gz
-rw-r--r-- 1 root root 1027528 abr 15 17:26 /var/backups/etc-backup-2018-04-15.tar.gz
hpoyatos@debian:~$
```

Figura 3.14 – Rodar o comando *date* antes e dentro do comando *tar*
Fonte: Elaborado pelo autor (2018)

Agora, basta agendarmos o comando de forma com que rode todos os dias. A primeira possibilidade é usar o comando `sudocrontab -u root -e` para criar uma tabela de agendamentos cron com a permissão do superusuário (root). O comando abrirá um arquivo com o vim ou nano e nele digitaremos o que está na Figura “Agendar o backup utilizando o comando *crontab*”:

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m. every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
30 2 * * * tar cfz /var/backups/etc-backup-$(date +%Y-%m-%d).tar.gz /etc
```

Figura 3.15 – Agendar o backup utilizando o comando crontab
Fonte: Elaborado pelo autor (2018)

Cada uma das colunas de tabela **cron** tem um significado, sendo:

- **Primeira coluna – minutos:** Informe arquivo o qual minuto o comando rodará. Números como “0” ou “00” farão o comando rodar em horas cheias, enquanto “30” fará rodar no minuto trinta e assim por diante. Se quisermos que o comando rode a cada 5 minutos, usamos “*/5”, por exemplo.
- **Segunda coluna – hora:** Informe a hora da qual o comando deve acontecer no formato 0-23. Ou seja, “00 5” fará com que o comando rode às cinco horas da manhã, enquanto “45 21” fará com que rode às nove e quarenta e cinco da noite. O uso do asterisco faz com que rode em todas as horas, ou seja, “10 * * * *” faria o comando rodar em todas as horas de minuto 10, enquanto “*/20 * * * *” faria o comando rodar a cada vinte minutos.
- **Terceira coluna – dia do mês:** Permite informar o dia do mês em que o comando deverá rodar. Por exemplo, “30 23 10 * *” faria o comando rodar às 23h30 todo dia 10 de cada mês.

- **Quarta coluna – mês:** Permite informar o mês em que o comando deverá rodar, sendo assim, “00 2 8 8 *” faria o comando rodar todo dia oito de agosto às 2 da manhã.
- **Quinta coluna – dia da semana:** Permite informar o dia da semana em que o comando deverá rodar, sendo “0” para domingo e “6” para sábado sendo assim, “15 4 * * 3” faria o comando rodar toda quarta-feira às quatro e quinze da manhã.
- **Sexta coluna – comando:** Na sequência, deve ser informado o comando propriamente dito.

Assim sendo, o *backup* do diretório **/etc** foi marcado para todos os dias às duas e meia da manhã, bastando sair do vi com o comando “:X”.

Outra possibilidade mais simples é a criação de *scripts* que podem ser armazenados ou vinculados aos diretórios **/etc/cron.hourly**, **/etc/cron.daily**, **/etc/cron.weekly** e **/etc/cron.monthly**, permitindo rodar o comando de hora em hora, diariamente, semanalmente e mensalmente, respectivamente. Os horários de execução podem ser vistos ou modificados no arquivo **/etc/crontab**, como pode ser visto na Figura “Conteúdo do arquivo /etc/crontab”:

```
hpoiyatos@debian:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
hpoiyatos@debian:~$
```

Figura 3.16 – Conteúdo do arquivo /etc/crontab

Fonte: Elaborado pelo autor (2018)

Atente-se apenas ao fato que em **/etc/crontab** deve ser informado o usuário da qual o comando rodará com os privilégios.

3.3 Política de *firewall*

Na última parte de nosso capítulo, mas não menos importante, vamos abordar como implementar algumas regras de *firewall* que permitam a filtragem de pacotes.

O *firewall* mais utilizado do Linux, o iptables, é capaz de filtrar qualquer pacote de dados que entre ou saia por qualquer uma das interfaces de rede do equipamento.

Temos as seguintes correntes de regras como padrão:

Nome da corrente	Descrição
INPUT	Regras a serem aplicadas para qualquer pacote que tenha como destino o sistema em que estamos; chegam ao equipamento.
OUTPUT	Regras a serem aplicadas para qualquer pacote que tenha como origem o sistema em que estamos; partam do equipamento.
FORWARD	Regras a serem aplicadas para qualquer pacote que passe pelo sistema em que estamos. Chegam por uma interface do equipamento e saem por outra.
PREROUTING / POSTROUTING	Disponível para casos específicos quando o equipamento faz NAT (<i>Network Address Translation</i> ou traduções de endereços de rede) PREROUTING é usado para SNAT (<i>Source NAT</i> , quando se muda o endereço de origem) e POSTROUTING é usado para DNAT (<i>Destination NAT</i> , quando se muda o endereço de destino), operações comuns em <i>gateways</i> de rede.

Quadro 3.2 – Correntes de regras padrão no iptables
Fonte: Elaborado pelo autor (2018)

A Figura “*Fluxo de Processo do iptables*” ilustra o fluxo de pacotes e a possibilidade de que passem por cada uma das correntes:

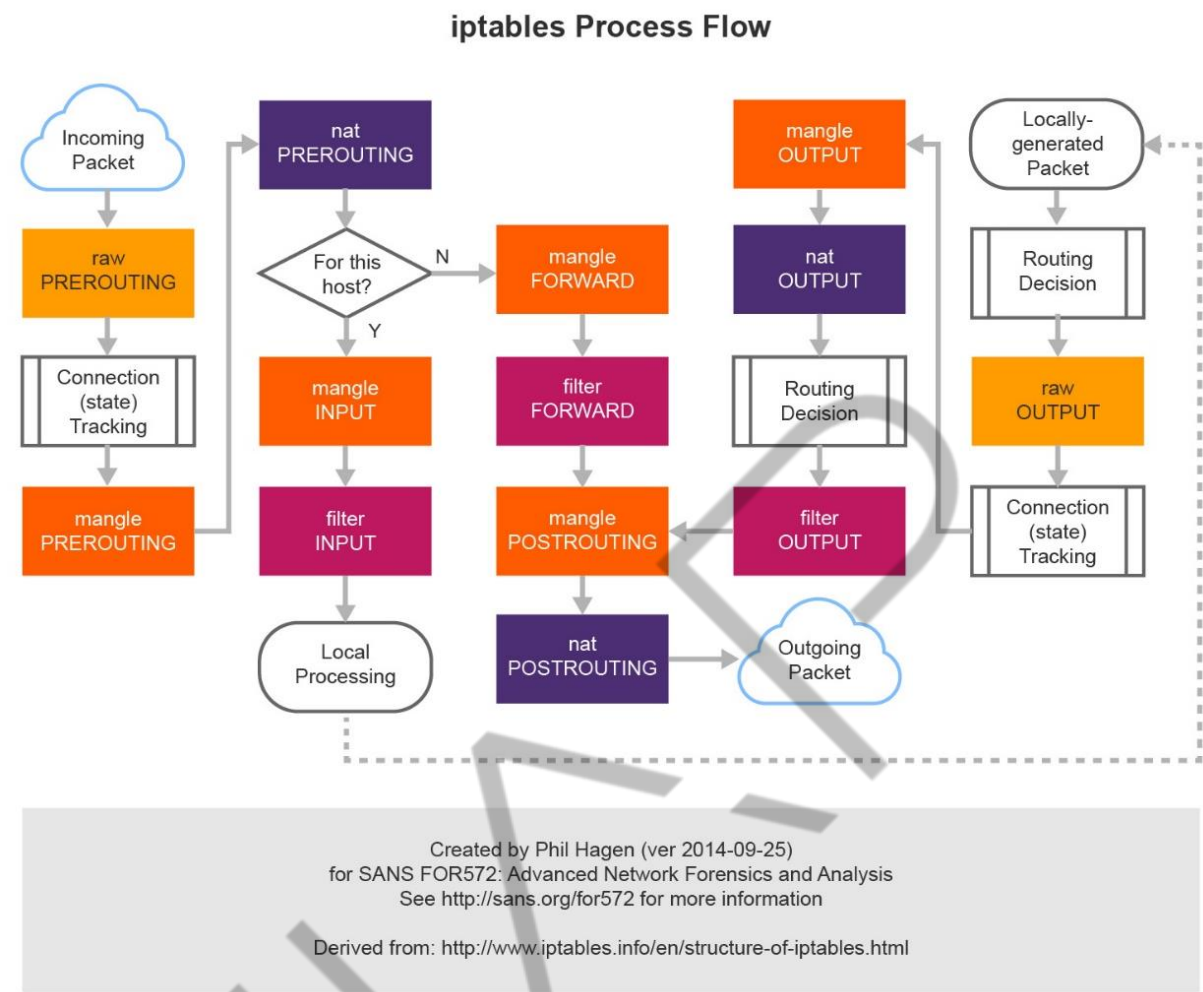


Figura 3.17 – Fluxo de Processo do iptables
Fonte: Phil Hagen (2014)

Quanto às ações que podem ser tomadas, temos:

Nome da ação	Descrição
ACCEPT	Aceita o pacote, liberando-o para entrada, saída ou passagem pelo equipamento.
REJECT	Rejeita o pacote devolvendo uma resposta ao remetente; se for uma conexão do tipo UDP, acontece um ICMP do tipo 3, enquanto uma do tipo TCP retorna um TCP reset.
DROP	Descarte o pacote silenciosamente, sem retornar qualquer tipo de resposta ao remetente.

Quadro 3.3 – Ações que podem ser realizadas no iptables
Fonte: Elaborado pelo autor (2018)

3.3.1 Filtragem de pacotes – iptables

O comando que será utilizado para a filtragem de pacotes é o iptables. No quadro “*Principais parâmetros do comando iptables*”, esclarecemos o que fazem os principais parâmetros disponíveis:

Nome da ação	Descrição
-A	Adiciona uma nova regra ao final da corrente
-D	Remove uma ou mais regras de uma corrente
--dport	Permite informar a porta específica em que a regra se aplica
-F	Remove todas as regras do <i>firewall</i> ou de uma corrente em específico
-i	Permite definir a qual interface se aplica a regra
-j	Permite definir a ação que será aplicada pela regra (ACCEPT REJECT DROP)
-I	Adiciona uma regra no topo da corrente
-L	Lista as regras do <i>firewall</i> ou de uma corrente em específico
-P	Estabelece a política padrão de uma determinada corrente
-p	Permite informar o tipo de pacote (tcp udp)
-s	Permite informar qual o IP de origem do pacote
-X	Remove todas as regras de uma corrente

Quadro 3.4 – Principais parâmetros do comando iptables

Fonte: Elaborado pelo autor (2018)

Para que a demonstração a seguir seja realmente relevante, utilizei o host que possuo na Digital Ocean, assim sendo, estou estabelecendo regras de firewall para um equipamento que está exposto à Internet e pode se beneficiar muito destas regras. Pode usar quaisquer duas máquinas para testar as regras, podendo ser até mesmo a relação entre o Windows de uma máquina real e o Linux virtualizado com o Oracle VirtualBox, tudo depende do tipo de rede que foi estabelecido entre as máquinas e as regras que deseja testar.

Para começar nossa demonstração, criaremos um arquivo no *host* da Digital Ocean chamado **/etc/firewall.sh** com o comando `sudo vi /etc/firewall.sh` e digitaremos inicialmente as seguintes linhas (Vide Código-fonte “Arquivo */etc/firewall.sh*, primeira versão”):

```
#!/bin/bash
```

```
# Limpa as regras de todas as correntes
iptables -F
iptables -X

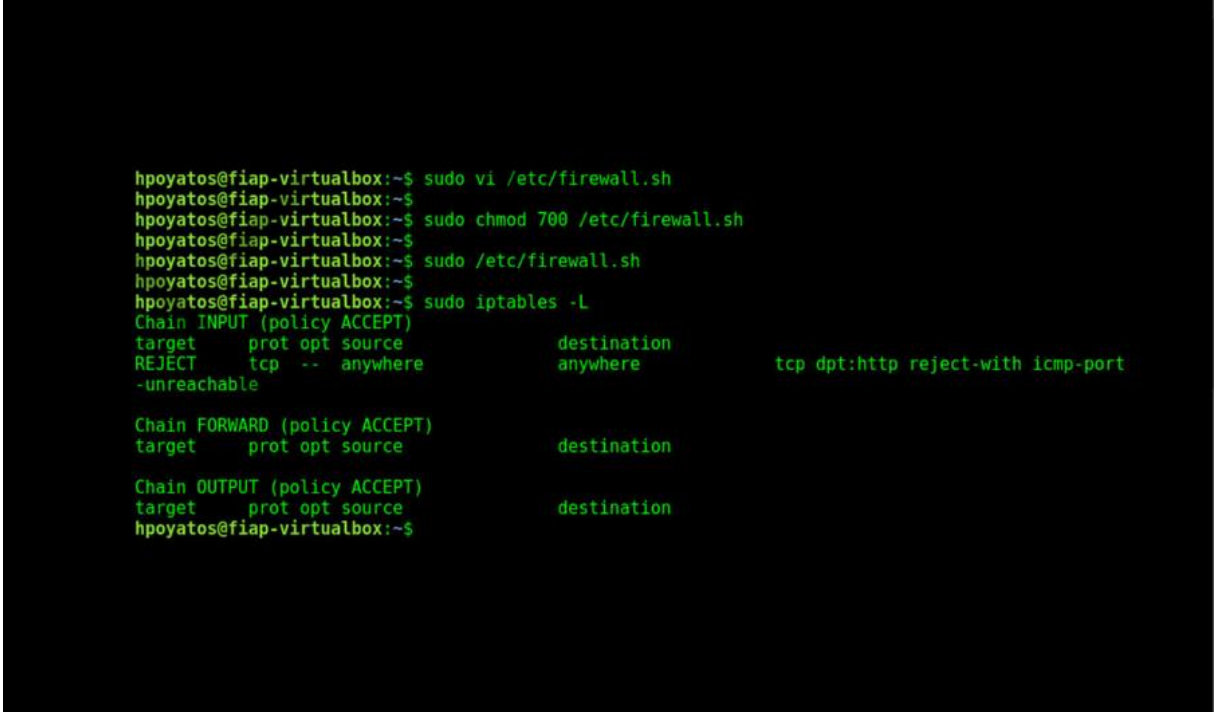
# Estabelece as políticas padrão das correntes
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT

# Rejeita quaisquer conexões na porta 80
iptables -A INPUT -i eth0 -p tcp --dport 80 -j REJECT
```

Código-fonte 3.1 – Arquivo /etc/firewall.sh, primeira versão

Fonte: Elaborado pelo autor (2018)

Com estas poucas linhas, rejeitaremos todas as requisições que chegam na porta 80, ou seja, a porta do serviço HTTP da qual se encontra o Apache. Ao salvar o arquivo, digitarei o comando `sudo chmod 700 /etc/firewall.sh` para que o comando seja lido, modificado e executado pelo dono do arquivo, o root ($4+2+1 = 7$), e não possa ser acessado por mais ninguém “0”. Na Figura “Colocar o firewall no ar”, podemos ver o momento em que as regras passam a vigorar:



```
hpoyatos@fiap-virtualbox:~$ sudo vi /etc/firewall.sh
hpoyatos@fiap-virtualbox:~$
hpoyatos@fiap-virtualbox:~$ sudo chmod 700 /etc/firewall.sh
hpoyatos@fiap-virtualbox:~$
hpoyatos@fiap-virtualbox:~$ sudo /etc/firewall.sh
hpoyatos@fiap-virtualbox:~$
hpoyatos@fiap-virtualbox:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination            tcp dpt:http reject-with icmp-port
-unreachable

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
hpoyatos@fiap-virtualbox:~$
```

Figura 3.18 – Colocar o firewall no ar

Fonte: Elaborado pelo autor (2018)

O comando `sudo /etc/firewall.sh` é o suficiente para rodar as regras, pois o arquivo é executável ao root graças ao `chmod` e o interpretador de comandos foi definido na primeira linha do arquivo (`#!/bin/bash`), ou seja, criamos nosso primeiro *bash script*.

Com o comando `sudo iptables -L`, conseguimos ver as três correntes (INPUT, FORWARD e OUTPUT) e as políticas aplicadas a cada uma delas. Além disso, nossa primeira regra de rejeição está listada na corrente INPUT e, portanto, ativa.

Neste momento, é impossível acessar o Apache do servidor através do navegador web, pois o firewall está bloqueando a entrada do pacote. O serviço está rodando normalmente e “escutando” a porta 80, mas os pacotes simplesmente não chegam.

Uma prova que o serviço está funcionando é acessá-lo localmente utilizando o navegador web de modo texto **links**. A Figura “Acessar o servidor web localmente com o comando **links**” prova que o Apache funciona, e o firewall não barrou este pacote, pois quando criamos a regra estipulamos que os pacotes rejeitados deveriam passar pela interface de rede **eth0**, enquanto o navegador web **links** acessa com `http://localhost`, passando pela interface de rede **lo**.

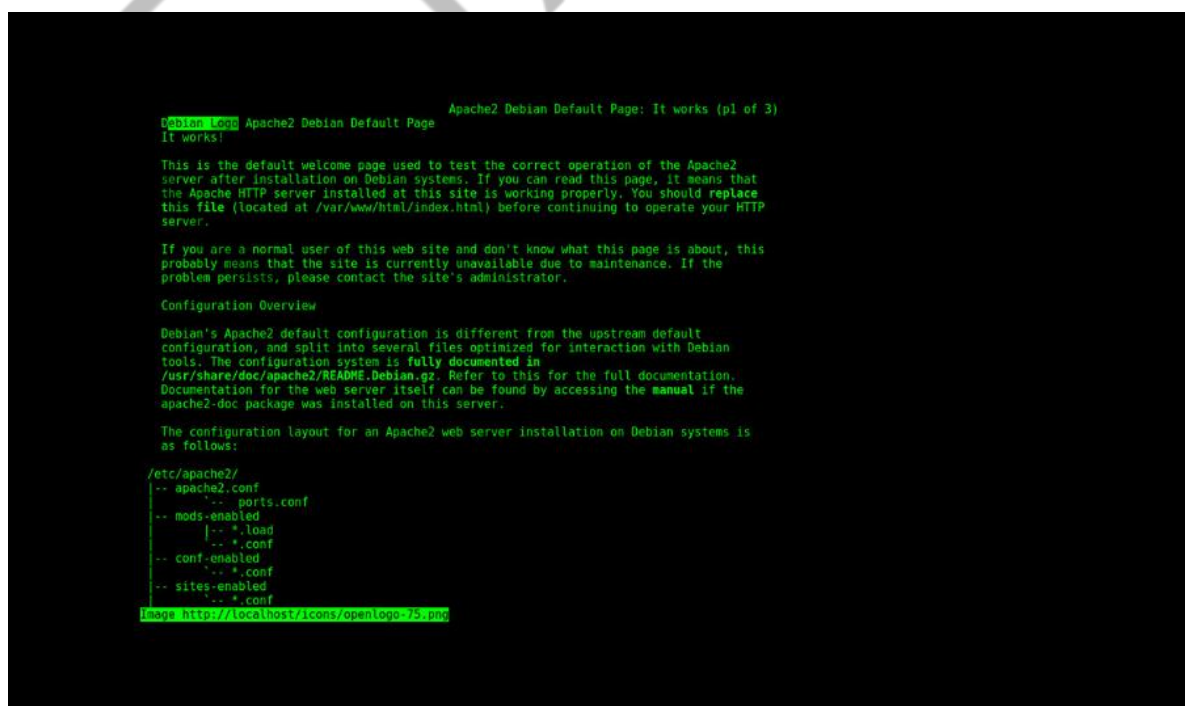


Figura 3.19 – Acessar o servidor web localmente com o comando **links**

Fonte: Elaborado pelo autor (2018)

Ao acessar o servidor com ssh, descobrimos que o meu IP de saída é 179.209.93.204, ou seja, nossa conexão cabeada está usando este IP no momento. Com uma nova regra, decidimos aceitar conexões oriundas do meu IP, rejeitando com as demais. Além disso, mudemos a política padrão de FORWARD para REJECT, o que não altera nossa demonstração (Vide Código-fonte “Arquivo /etc/firewall.sh, segunda versão”):

```
#!/bin/bash

# Limpa as regras de todas as correntes
iptables -F
iptables -X

# Estabelece as políticas padrão das correntes
iptables -P INPUT ACCEPT
iptables -P FORWARD REJECT
iptables -P OUTPUT ACCEPT

# Rejeita quaisquer conexões na porta 80
iptables -A INPUT -i eth0 -s 179.209.93.204 -p tcp --dport 80
-j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport 80 -j REJECT
```

Código-fonte 3.2 – Arquivo /etc/firewall.sh, segunda versão

Fonte: Elaborado pelo autor (2018)

Repare que a regra de *aceite* foi colocada ANTES da regra de *rejeição*, e isso não é por acaso. Se fosse colocada após, a primeira regra da corrente INPUT rejeitaria todas as conexões, incluindo a nossa, a regra de aceite jamais seria alcançada e nossa conexão nunca seria liberada. É por isso que as regras são agrupadas em correntes, possuem uma ordem de execução.

Rodar o script novamente com `sudo /etc/firewall.sh` é o suficiente, pois criamos o arquivo para sempre limpar as regras e carregá-las todas novamente. A alternativa seria rodar o comando `sudo iptables -I INPUT -i eth0 -s 179.209.93.204 -p tcp --dport 80 -j ACCEPT`; repare que o “-I” é usado no lugar do “-A” para que a regra seja colocada antes da regra de rejeição vigente.

Agora consigo acessar normalmente o servidor web com meu navegador, embora eu seja o único a poder fazer isso. Trata-se de uma regra que não pode ser permanente, pois não estou utilizando um IP dedicado, e sim um IP dinâmico; no

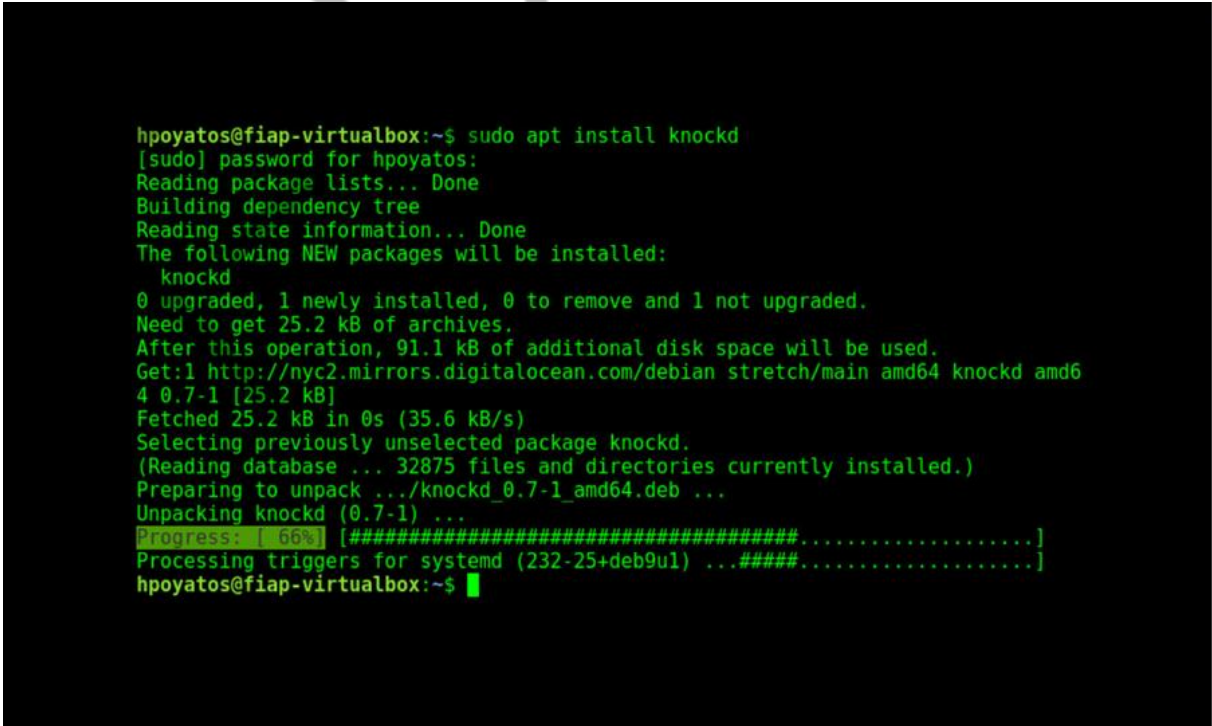
momento em que meu equipamento fosse reiniciado, um novo IP seria destinado, fazendo com que deixasse de acessar o host pelo navegador *web* novamente.

3.3.2 Port-knocking

As portas de serviço de um equipamento não precisam ficar 100% do tempo disponíveis e vulneráveis a ataques. Um recurso que pode aumentar muito a segurança é implementar o *port-knocking* (traduzido para “batidas na porta”) que remete àqueles filmes antigos em que a porta misteriosa seria aberta por alguém dentro do estabelecimento, se o personagem batesse na porta na sequência esperada.

É exatamente isso que acontece, mas a porta aberta é uma porta de serviço liberada por uma regra de *firewall* disparada, se alguns pacotes de rede específicos atinjam algumas portas na sequência certa.

Para facilitar, vamos instalar o serviço chamado **knockd** usando o comando `sudo apt install knockd` (Vide Figura “*Instalar o serviço knockd*”):



```
hpyatos@fiap-virtualbox:~$ sudo apt install knockd
[sudo] password for hpyatos:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  knockd
0 upgraded, 1 newly installed, 0 to remove and 1 not upgraded.
Need to get 25.2 kB of archives.
After this operation, 91.1 kB of additional disk space will be used.
Get:1 http://nyc2.mirrors.digitalocean.com/debian stretch/main amd64 knockd amd6
4 0.7-1 [25.2 kB]
Fetched 25.2 kB in 0s (35.6 kB/s)
Selecting previously unselected package knockd.
(Reading database ... 32875 files and directories currently installed.)
Preparing to unpack .../knockd_0.7-1_amd64.deb ...
Unpacking knockd (0.7-1) ...
Progress: [ 66%] [#####.....]
Processing triggers for systemd (232-25+deb9u1) ...####[.....]
hpyatos@fiap-virtualbox:~$
```

Figura 3.20 – Instalar o serviço knockd
Fonte: Elaborado pelo autor (2018)

Vamos realizar algumas mudanças de regras de firewall no arquivo `/etc/firewall.sh`. Repare a presença de uma regra baseada no parâmetro **-ctstate** que permite que conexões *entrantes* pré-existentis continuem aceitas e não sejam derrubadas. Encerramos as regras com uma que rejeita todas as conexões entrantes além das estabelecidas (Vide Código-fonte “O arquivo `/etc/firewall.sh`”):

```
#!/bin/bash

# Limpa as regras de todas as correntes
iptables -F
iptables -X

# Estabelece as políticas padrão das correntes
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT

# Rejeita quaisquer conexões na porta 80
iptables -A INPUT -i lo -j ACCEPT

# Aceita e mantém conexões já estabelecidas
iptables -A INPUT -m conntrack --ctstateRELATED,ESTABLISHED -j ACCEPT

# Aceita conexões na porta 80 livremente
iptables -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT

# Rejeita outras conexões entrantes
iptables -A INPUT -j DROP
```

Código-fonte 3.3 – O arquivo `/etc/firewall.sh`

Fonte: Elaborado pelo autor (2018)

Ao confirmar as regras com o comando `sudo /etc/firewall.sh`, a conexão ssh atual não é derrubada, graças à regra de aceite mencionada anteriormente.

O arquivo `/etc/knockd.conf` nos é entregue praticamente pronto, exigindo apenas uma alteração: a regra para abrir a porta 22 para meu IP deve ser criada com `iptables -I INPUT -s %IP% -p tcp --dport 22 -j ACCEPT` e não `iptables -A INPUT -s %IP% -p tcp --dport 22 -j ACCEPT`, a nova regra de firewall com o parâmetro “-I” ficaria após a regra de DROP e jamais seria alcançada. Sua correção como parâmetro “-I” para que a regra seja executada

ANTES do DROP, resolvendo o problema (Vide Código-fonte “O arquivo */etc/knockd.conf* com uma ligeira alteração”):

```
[options]
UseSyslog

[openSSH]
sequence      = 7000,8000,9000
seq_timeout   = 5
command       = /sbin/iptables -I INPUT -s %IP% -p tcp -
-dport 22 -j ACCEPT
tcpflags      = syn

[closeSSH]
sequence      = 9000,8000,7000
seq_timeout   = 5
command       = /sbin/iptables -D INPUT -s %IP% -p tcp
--dport 22 -j ACCEPT
tcpflags      = syn
```

Código-fonte 3.4 – O arquivo */etc/knockd.conf* com uma ligeira alteração
Fonte: Elaborado pelo autor (2018)

Cabe uma última alteração no arquivo */etc/default/knockd*, trocando o parâmetro **START_KNOCKD** de "0" para "1" (Vide Código-fonte “O arquivo */etc/default/knock* alterado”):

```
# control if we start knockd at init or not
# 1 = start
# anything else = don't start
# PLEASE EDIT /etc/knockd.conf BEFORE ENABLING
START_KNOCKD=1

# command line options
#KNOCKD_OPTS="-i eth1"
```

Código-fonte 3.5 – O arquivo */etc/default/knock* alterado
Fonte: Elaborado pelo autor (2018)

Basta agora subir o serviço **knockd** com o comando **sudo /etc/init.d/knockd start** e o *port-knocking* está devidamente implementado.

Cabe testar a segurança, inicialmente realizando um escaneamento na porta 22 que indica uma filtragem ali; a tentativa de conexão direta na porta resulta em um *timeout* (Vide Figura “Verificar e testando acesso na porta 22 do ssh”):

```
hpoyatos@debian:~$ nmap 67.205.132.93 -p 22

Starting Nmap 7.40 ( https://nmap.org ) at 2018-04-16 22:18 -03
Nmap scan report for digitalocean (67.205.132.93)
Host is up (0.14s latency).
PORT      STATE      SERVICE
22/tcp    filtered  ssh

Nmap done: 1 IP address (1 host up) scanned in 1.55 seconds
hpoyatos@debian:~$
hpoyatos@debian:~$ ssh hpoyatos@67.205.132.93
ssh: connect to host 67.205.132.93 port 22: Connection timed out
hpoyatos@debian:~$
```

Figura 3.21 – Verificar e testando acesso na porta 22 do ssh
Fonte: Elaborado pelo autor (2018)

Conforme visto em Código-fonte “O arquivo */etc/knockd.conf* com uma ligeira alteração”, o serviço espera por pacotes nas portas 7000, 8000 e 9000 respectivamente, e o próprio comando *nmap* pode ser usado para isso. Em Código-fonte “Shell script que realiza o *port-knocking*”, temos um pequeno *shell script* que dispara os pacotes nas portas 7000, 8000 e 9000, realizando o *port-knocking* e permitindo a conexão *ssh* na sequência. Caso queira realizar, troque o *ip67.205.133.93* pelo host em questão:

```
for x in 7000 8000 9000; do nmap -Pn --host_timeout 201 --max-retries 0 -p $x 67.205.133.93; done
```

Código-fonte 3.6 – Shell script que realiza o *port-knocking*
Fonte: Elaborado pelo autor (2018)

```
hpoyatos@debian:~$ for x in 7000 8000 9000; do nmap -Pn --host_timeout 201 --max-retries 0 -p $x 67.205.132.93; done

Starting Nmap 7.40 ( https://nmap.org ) at 2018-04-16 23:00 -03
Warning: 67.205.132.93 giving up on port because retransmission cap hit (0).
Nmap scan report for digitalocean (67.205.132.93)
Host is up.
PORT      STATE      SERVICE
7000/tcp  filtered  afs3-fileserver

Nmap done: 1 IP address (1 host up) scanned in 1.06 seconds

Starting Nmap 7.40 ( https://nmap.org ) at 2018-04-16 23:00 -03
Warning: 67.205.132.93 giving up on port because retransmission cap hit (0).
Nmap scan report for digitalocean (67.205.132.93)
Host is up.
PORT      STATE      SERVICE
8000/tcp  filtered  http-alt

Nmap done: 1 IP address (1 host up) scanned in 1.10 seconds

Starting Nmap 7.40 ( https://nmap.org ) at 2018-04-16 23:00 -03
Warning: 67.205.132.93 giving up on port because retransmission cap hit (0).
Nmap scan report for digitalocean (67.205.132.93)
Host is up.
PORT      STATE      SERVICE
9000/tcp  filtered  cslistener

Nmap done: 1 IP address (1 host up) scanned in 1.11 seconds
hpoyatos@debian:~$ ssh hpoyatos@67.205.132.93
hpoyatos@67.205.132.93's password: █
```

Figura 3.22 – Port-knocking realizado com sucesso
Fonte: Elaborado pelo autor (2018)

Conforme pode ser visto em Código-fonte “O arquivo `/etc/knockd.conf` com uma ligeira alteração”, o serviço knockd pode fechar a porta se os pacotes forem disparados na sequência contrária (portas 9000, 8000 e 7000), cabendo uma ligeira modificação em Código-fonte “Shell script que realiza o port-knocking” para tal.

Para aumentar a segurança, altere todas as portas do exemplo mostrado aqui. Comece alterando o arquivo `/etc/ssh/sshd_config` para que o serviço ssh suba em uma porta diferente da 23. Depois, altere `/etc/knockd.conf` para que a regra do firewall libere a mesma porta que o ssh subirá. Além disso, modifique as portas que abrirão e fecharão o ssh, o *port-knocking* deve ocorrer em portas fora do padrão, pois as primeiras que um atacante tentará bater são justamente 7000, 8000 e 9000, pois são o padrão predefinido por knockd. Suba os serviços de sshd e knockd e sintá-se mais seguro do que antes.

3.4 Considerações finais

Esperamos que tenha gostado de conhecer mais sobre o Linux. Estes três capítulos que abordam os comandos deste sistema operacional não tinham qualquer pretensão de ser um “guia completo e definitivo” ou a última palavra em segurança;

na verdade, esperamos que os conhecimentos abordados aqui tenham lhe inspirado a conhecer mais sobre este sistema e sua segurança, em uma jornada de conhecimento que apenas começou.

Até a próxima!

EMSE

REFERÊNCIAS

DIGITAL OCEAN. **How To Use Port Knocking to Hide your SSH Daemon from Attackers on Ubuntu.** 2014. Disponível em: <<https://www.digitalocean.com/community/tutorials/how-to-use-port-knocking-to-hide-your-ssh-daemon-from-attackers-on-ubuntu>>. Acesso em: 13 jul. 2020.

FERREIRA, Rubem E. **Linux Guia do Administrador do Sistema.** 3.ed. São Paulo: Novatec, 2008.

EXEMPLO

GLOSSÁRIO

Partição primária	São as primeiras partições de um disco rígido. São permitidas até quatro partições primárias e este era o máximo de divisões permitidas no passado.
Partição estendida	Recurso que permitia que uma partição primária fosse subdividida, possuindo sua própria tabela de partições para seu controle. Quando isso é realizado, a partição primária da qual isso acontece é conhecida como partição estendida.