Capítulo 7 Administração de usuários e permissões – Parte 1

7.1. Objetivos :

Entender como funciona o sistema de usuários, grupos, permissões.

7.2. Introdução

Um dos destaques do GNU/Linux é o seu poder em restringir usuários no sistema. É necessário primeiro aprender conceitos de administração de usuários no sistema.

Os 3 tipos de usuários que temos no sistema:
Usuário root
Usuários comuns
Usuários de sistema

Usuários e Funções:

•root – administrador do sistema e tem poder total sobre ele;

•comum - acesso restrito e tem o diretório home;

•sistema - são usuários que não se logam, apenas controlam serviços, exemplos: web, ftp etc.

O usuário que dá mais dor de cabeça é o usuário comum! Por isso temos as permissões que veremos mais adiante para reduzir a dor! Os usuários de sistema são usuários de serviços. Por exemplo: se você tem um apache (servidor web) instalado, ele vai ter um usuário para controlá-lo. Se você usa banco de dados, terá um usuário para controlar o banco. Controlar está relacionado com o controle de processos e das requisições no serviço .

Processo é algo que esteja em execução no sistema e ocupa área na memória RAM e que usa o processamento da máquina. Teremos uma aula de Processos no Linux ainda.

Em geral não é necessário preocupar-se com os usuários de sistema, pois os mesmos são usuários normais e quase sempre não tem shell para logar-se na máquina, não tem diretório home e não tem senha! Claro que devemos sempre deixar apenas os usuários que estão sendo usados pelos serviços da máquina.

Falando em usuários, cada usuário tem um número para identificá-lo. Esse número chamamos de ID ou UID (identificação do usuário). O nome do usuário para o sistema é só “perfumaria”, ou seja, o que vai realmente importar é o seu ID. Podemos comparar o ID como o seu CPF. Esse ID é como se fosse um CPF, ou seja, não posso ter 2 usuários com o mesmo ID. O que podemos ter são vários logins com o mesmo ID, mas para o sistema é um usuário só com vários apelidos diferentes. Mas você só se loga com o apelido (login) e não com o ID.

Para saber o ID de um usuário faça (exemplo):

```
# id maria 1001
```

Esse ID, assim como outras informações, são armazenadas no arquivo /etc/passwd. O arquivo /etc/passwd é a "base" onde ficam armazenados todos os usuários da máquina! Visualizando Login e ID dos usuários:

```
# cat /etc/passwd | cut -d: -f1,3
```

Assim fica mais legível. Vamos explicar o que esse comando faz: O cat lista o arquivo passwd e joga a saída para o cut que vai cortar o arquivo a cada ":" -d: significa delimitador, e vai apenas me mostrar a 1 e a 3 coluna do arquivo cortado. Até mesmo a seqüência numérica dos IDs tem uma lógica:

Relação de ID:

0 - ROOT
1 - 999 - Usuários de Sistema
>= 1000 Usuário normal

Isso é para que o sistema possa organizar as restrições! Sendo assim, não importa o nome do seu usuário, quando ele logar-se no sistema o GNU/Linux irá gerenciá-lo pelo ID.

Então, estou dizendo aqui, que o usuário root tem sempre que ter ID igual 0, que significa poder total, pode até ter um outro nome, mas o seu ID tem que ser 0 para ter poderes de superusuário. O ID é o que importa para o sistema. É através desse número que o sistema irá saber o que você pode ou não fazer.

Se tiver dois usuários com o mesmo número de ID, para o sistema você terá apenas um usuário com dois nomes diferentes, mas na prática ele sempre irá assumir que você é o primeiro que está listado no /etc/passwd.

Podemos perceber pelo comando que esse arquivo possui os campos separados por " : ":

```
# cat /etc/passwd
```

Cada : temos um campo com uma informação importante sobre um usuário.

Dica LPI: Qual é a seqüência dos campos do arquivo passwd??? Temos 7 campos no arquivo.

Relação dos campos:

1 - Nome do usuário no sistema (login)

2 - Senha do usuário (Obs: se tiver x aponta a senha para o /etc/shadow)

3 - ID do usuário

4 - ID do grupo primário do usuário

5 - Informações Pessoais do Usuário como Nome, Telefone etc

6 - Diretório Home do Usuário

7 - Shell que o usuário vai usar na hora do Login

Lembrando que, no campo 2, se estiver habilitado senhas no /etc/shadow sempre terá um x indicando que a senha não está nesse arquivo! Caso contrário a criptografia da senha estará no campo 2 do arquivo /etc/passwd. E, se ainda não tem nada, o usuário não tem senha! A maioria das distribuições Linux já habilitam shadow (senha sombra) automaticamente.

A função de existir o arquivo de senhas, é porque o arquivo /etc/passwd precisa ter permissão de leitura para os usuários, pois é lá que ele pega as informações referentes a ele, tais como diretório home, shell, ID etc.

Então, como todos os usuários podem ver o conteúdo do arquivo /etc/passwd, eles também poderiam pegar a criptografia das senhas e tentar quebrar com alguns programas. Aí foi criado um arquivo apenas para guardar as senhas, que só o root precisa ter acesso de leitura.

Quando um usuário é criado, a única coisa que acontece é que uma linha é inserida no arquivo /etc/passwd e no /etc/shadow.

Usa-se os comandos useradd ou adduser para criar usuários no sistema. Mas qual a diferença entre eles? O useradd sem parâmetros apenas cria um usuário na máquina, sem senha, sem diretório pessoal etc. Agora o adduser é um script que pode variar em cada distribuição. Esse script vai perguntar para você informações para a criação do usuário (senha, home etc).

Caso precise criar um usuário sem home, sem shell é melhor usar o comando useradd (sem parâmetros). Caso for criar um usuário normal para usar o sistema, tem a opção de usar o adduser pois ele é bem completo.

Criando um usuário simples:

```
# useradd maria
```

Ele nem tem home:

```
# ls /home
```

Nem senha:

```
# cat /etc/shadow | grep  maria
```

Criando um usuário completo:

```
# adduser linus
```

Ele tem home:

```
# ls /home  linus
```

Sua senha :

```
# cat /etc/shadow | grep linus
linus: $1$k4G05HcG$eI81FGaGsA5mPVveusl6X/ :12973:0:99999:7:::
```

Adicionando um usuário com esses comandos, uma linha é inserida no /etc/passwd, e uma linha no /etc/shadow com a senha do usuário!

Todo o usuário no GNU/Linux tem um grupo e isso serve para criar permissões para grupos. Um usuário é obrigado a ter um grupo primário e ele pode pertencer a vários outros grupos adicionais também.

Por padrão o GNU/Linux adota que o grupo primário de um usuário criado será um grupo com o próprio nome do usuário.

A idéia de grupo é assim: cria-se um grupo e coloca-se vários usuários, assim determina-se tudo que esse grupo pode fazer falando em arquivos que poderão acessar.

Damos permissões para 3 usuários no sistema: Uma permissão e a mais importante é a permissão de DONO, ou seja, quem manda no arquivo ou diretório, quem criou. A outra permissão é a de GRUPO , ou seja, todo mundo que pertence ao mesmo grupo poderá ou não entrar no arquivo ou diretório. E a última permissão seria a permissão para todo o resto do sistema que não seja o DONO ou GRUPO . Os grupos ficam armazenados no arquivo /etc/group.

Na prática para ver o conteúdo desse arquivo:

```
# cat /etc/group
```

Para adicionar um grupo no sistema:

```
# groupadd selecaobrasileira
```

Perceba que irá ser acrescentada uma linha no arquivo /etc/group para o grupo selecaobrasileira:

```
# cat /etc/group
```

Seguindo a mesma lógica do usuário, cada grupo tem um número, que chamamos de GID. A partir do momento que o grupo existe já posso colocar usuários nele para que eu possa definir políticas de acesso para esse grupo! Para incluir um usuário no grupo criado, usamos o comando abaixo:

```
# gpasswd -a romario selecaobrasileira
```

Onde -a é de add, romario é o usuário do sistema, e selecaobrasileira o grupo que já criamos. Assim romario terá o grupo dele padrão para as permissões de seu home, e o grupo selecaobrasileira que eu acabei de colocar. Verificando isso, podemos novamente dar um cat no /etc/group

```
# cat /etc/group | grep selecaobrasileira
selecaobrasileira:x:1006:romario
```

E veremos que o grupo selecaobrasileira agora tem o usuário romario! O mesmo posso fazer para o usuário kaka que também criamos:

```
# gpasswd -a kaka selecaobrasileira
```

E podemos usar o comando:

```
# groups kaka
kaka : kaka selecaobrasileira
```

Que irá listar todos os grupos que kaka pertence!

Sendo que o grupo primário sempre irá constar no /etc/passwd (Coluna 4), e os demais grupos, apenas no /etc/group. Quando cria-se um usuário pode-se perceber que alguns arquivos são criados no HOME desse usuário automaticamente.

Por exemplo:

```
# adduser maria
# cd /home/maria
# ls -a
-rw-r--r-- 1 maria maria 1312 Mar 7 18:32 .bash_logout
-rw-r--r-- 1 maria maria 703 Mar 7 18:20 .profile
-rw-r--r-- 1 maria maria 1312 Mar 7 18:32 .bashrc
```

```
# cd /etc/skel
# ls -a
.
..
.profile
.bashrc
.bash_logout
```

Esses arquivos vem de um diretório chamado skel que fica no /etc. Esse diretório traz um padrão de arquivos e diretórios que serão criados no home do usuário, quando o mesmo for criado. Pode-se ver que nesse diretório tem o mesmos arquivos que no home do usuário:

```
# ls -a /etc/skel
# ls -a /home/maria
.
..
.profile
.bashrc
.bash_logout
```

A função deles que é:

- .bashrc - Arquivo de alias, para personalizar o shell do usuário
- .profile - Arquivo de execução no login, posso colocar qualquer comando nele, que o mesmo será executado quando o usuário logar
- .bash_logout - Arquivo que guarda comandos que serão executados no momento do logout Isso é muito bom para criar um padrão para o /home!

O Windows tem algo desse tipo! Pois todo Home de usuário tem diretórios padrões:

- Meus Documentos
- Imagens
- Musicas

Se quisermos deixar que nossa distribuição Linux também crie esses arquivos por padrão, como devemos fazer? Podemos fazer isso no GNU/Linux com o diretório /etc/skel!

```
# mkdir /etc/skel/Documentos
# mkdir /etc/skel/Imagens
# mkdir /etc/skel/Musicas
```

Depois que coloco tudo isso no skel, adiciono um novo usuário para ver se o home dele terá isso:

```
# adduser novo
# cd /home/novo
# ls
Documentos Imagens Musicas
```

O comando usermod tem a função de modificar as informações armazenadas no /etc/passwd do usuário! Então dando um:

```
# usermod --help
```

Veja as opções para alterar qualquer informações do passwd (home, ID, shell, etc). Assim não precisa ficar editando o arquivo com um editor de textos. Vamos supor que quero mudar a localização do home dele:

```
# mkdir /mnt/leo
# usermod -d /mnt/leo leo
```

Aqui, estou mudando o home do usuário leo, que antes era /home/leo para /mnt/leo. Mas isso não significa que esse diretório /mnt/leo será criado automaticamente.

Você terá que ajustar as coisas, ele apenas muda no arquivo e não faz milagre.

Na prática não usamos muito esse comando, mas é importante saber que ele existe por causa da LPI.

Com o chfn preenche-se as informações adicionais do usuário. E com o comando finger listamos as informações.

Exemplo de chfn:

 # chfn leonardo

Modificando as informações de usuário para leonardo Informe o novo valor ou pressione ENTER para aceitar o padrão

Nome Completo [Leonardo Afonso]:
Número da Sala [X]:
Fone de Trabalho [xXxX-xXxX]:
Fone Doméstico [XxXx-XxXx]:
Outro []: outro

Para ver as informações adicionais, veja o arquivo /etc/passwd:

 # cat /etc/passwd | grep leonardo

leonardo:x:1000:1000:LeonardoAfonso,X,xXxX-xXxX,XxXx XxXx,outro:/home/leonardo:/bin/bash

Vejam que as informações no 5° Campo estão separadas por " , " (vírgula)

Exemplo de finger:

```
# finger leonardo
Login:leonardo Name: Leonardo Afonso
Directory: /home/leonardo
Shell: /bin/bash
Office: X, xXxX-xXxX
Home Phone: XxXx-XxXx
On since Fri Nov 12 09:02 (BRST) on tty1 6 hours 58 minutes idle(messages off)  On since Fri Nov 12 09:11 (BRST) on pts/0 from :0.05 minutes 5 seconds idle  On since Fri Nov 12 09:11 (BRST) on pts/1 from :0.0
No mail.
No Plan.
```

Acabamos para exemplificar os comandos, criando uma porção de usuários. Já que não serão usados, para deletá-los usamos o comando:

```
# userdel romario
# userdel -r romario
```

Onde a opção -r é para remover o diretório pessoal. Então na prática se quero remover o usuário e seu home tenho que colocar -r.

7.3. Permissões no Linux

No GNU/Linux, como em outros sistemas Unix, cada arquivo tem uma permissão. As permissões são atributos dos arquivos que especificarão se ele pode ser:

- lido (r) - 4
- escrito (w) - 2
- executado (x) - 1

Estas permissões são o que vai definir o que um usuário pode fazer ou não. Define-se permissões para 3 pessoas como já tinha comentado.

- dono (u) - dono do arquivo ou diretório
- grupo (g) - que pertence ao mesmo grupo do dono
- outros (o) - todo o resto do sistema
- todos (a) – dono + grupo + outros

OBS: *É importante falar que o fato de você acessar o diretório, ainda sim você terá que ter permissão no arquivo dentro dele! Ou seja, posso dar permissão de acesso a um diretório e bloquear o acesso em alguns arquivos do mesmo. Então tenho que dar permissão no diretório e no arquivo que está dentro do diretório.*

Para dar permissão para um arquivo ou diretório uso o comando chmod. Existem duas maneiras para setar uma permissão com o comando chmod.

Revisando:

Damos permissão para 3 pessoas:
(u , g , o) user, group, other
E temos 3 tipos de permissão:
(r , w , x)read, write, executable

Então, apenas é setado o que se quer para cada um dos 3 usuários:

```
# cd /tmp
```

Exemplo:

```
# touch arquivo

# ls -l arquivo
-rw-r--r-- 1 root root 0 2007-07-18 21:36 arquivo
```

E setar as 3 permissões para as 3 pessoas:

```
# chmod u=rwx,g=rw,o=r arquivo
```

Ou seja, aqui neste comando o dono (u) terá permissão total, leitura(r), gravação(w) e execução (x). O grupo (g) apenas terá permissão de Leitura e gravação e o resto do povo (o) apenas leitura.

Falando dos sinais temos:

```
= Aplique exatamente assim
+ Adicionar mais essa
- Tirar essa
```

Exemplificando:

```
# ls -l arquivo
-rw-r--r-- 1 root root 0 2007-07-18 21:36 arquivo

# chmod u=rwx,g=r,o=r arquivo
# ls -l
-rwxr--r-- 1 root root 0 2007-07-18 21:36 arquivo
```

Ou seja exatamente assim!

Agora:

```
# ls -l
-rw-r--r-- 1 root root 0 2007-07-18 21:36 arquivo
# chmod u+x,g+w arquivo

# ls -l
-rwxrw-r-- 1 root root 0 2007-07-18 21:36 arquivo
```

Aqui, apenas foi adicionado o x para dono e w para grupo. As demais permissões permaneceram as mesmas.

O comando ls com a opção -l sempre irá me mostrar o dono o o grupo.

Então, o resultado do comando ls -l sempre será, falando nas colunas:

```
# ls -l leo.txt
- rw- r-- r-- 1 root root 30 2004-11-12 16:26 leo.txt
```

Onde:

- = É a identificação de Arquivo que pode ser:
- arquivo
d diretório
l link
rw- = Permissão do Dono
r-- = Permissão do Grupo
r-- = Permissão dos outros
1 = Indicando ser um arquivo único (não possui links em outro lugar)
root = Dono do Arquivo
root = Grupo do Arquivo
30 = Tamanho do Arquivo
Data do Arquivo
leo.txt = Nome do Arquivo

É exatamente com esse comando que é analisado o dono, grupo e permissões do arquivo ou diretório.

Mas é importante lembrar que a permissão x (execução) em um arquivo é para que o mesmo seja executado, como por exemplo shell script. Em diretório é para que eu possa entrar nesse diretório! Sendo assim, só poderei dar o comando cd no diretório caso o mesmo tenha a permissão x.