Comandos

Para Permissão de Acesso a Arquivos e Diretórios

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Prof. Edwar Saliba Júnior
Agosto / 2017

Permissão de Acesso

- As permissões de acesso protegem o sistema de arquivos GNU/Linux de acessos indevidos de pessoas ou programas não autorizados;

- As permissões de acesso do GNU/Linux também impede que um programa mal intencionado, por exemplo, apague um arquivo que não deve, envie arquivos especiais para outras pessoas ou forneça acesso da rede para que outros usuários invadam o sistema.

Donos, Grupos e Outros Usuários

- A ideia básica da segurança no sistema GNU/Linux é definir o acesso aos arquivos por donos, grupos e outros usuários:

- `dono`

  – É a pessoa que criou o arquivo ou o diretório. O nome do dono do arquivo/diretório é o mesmo do usuário usado para entrar no sistema GNU/Linux. Somente o dono pode modificar as permissões de acesso do arquivo.

  – As permissões de acesso do dono de um arquivo somente se aplicam ao dono do arquivo/diretório. A identificação do dono também é chamada de `user id` (`UID`).

  – A identificação de usuário ao qual o arquivo pertence é armazenada no arquivo `/etc/passwd` e do grupo no arquivo `/etc/group`. Estes são arquivos textos comuns e podem ser editados em qualquer editor de texto, mas utilize preferencialmente os comandos `vipw` e `vigr` que executa procedimentos adicionais de checagem de `UID's` e grupos após a alteração. Tenha cuidado para não modificar o campo que contém a senha do usuário encriptada (que pode estar armazenada no arquivo `/etc/passwd` caso não estiver usando senhas ocultas).

Donos, Grupos e Outros Usuários

- `grupo`
  - Permite que vários usuários diferentes tenham acesso a um mesmo arquivo (já que somente o dono poderia ter acesso ao arquivo). Cada usuário pode fazer parte de um ou mais grupos e então acessar arquivos que pertençam ao mesmo grupo que o seu (mesmo que estes arquivos tenham outro dono);
  - Por padrão, quando um novo usuário é criado e se não foi especificado nenhum grupo, ele pertencerá ao grupo de mesmo nome do seu grupo primário (este comportamento é controlado pelo parâmetro `USERGROUPS=yes` do arquivo `/etc/adduser.conf`). A identificação do grupo é chamada de `GID` (`group id`);
  - Um usuário pode pertencer a um ou mais grupos.

Donos, Grupos e Outros Usuários

- `outros`
  - É a categoria de usuários que não são donos ou não pertencem ao grupo do arquivo;
  - Cada um dos tipos acima possuem três tipos básicos de permissões de acesso: `r`, `w` e `x`.

Tipos de Permissões de Acesso

- Quanto aos tipos de permissões que se aplicam ao dono, grupo e outros usuários, temos 3 permissões básicas:

  `r` - Permissão de leitura para arquivos. Caso for um diretório, permite listar seu conteúdo (através do comando `ls`, por exemplo);

  `w` - Permissão de gravação para arquivos. Caso for um diretório, permite a gravação de arquivos ou outros diretórios dentro dele. Para que um arquivo/diretório possa ser apagado, é necessário a permissão de gravação;

  `x` - Permite executar um arquivo (caso seja um programa executável). Caso seja um diretório, permite que seja acessado através do comando `cd`.

- As permissões de acesso a um arquivo/diretório podem ser visualizadas com o uso do comando `ls -la`.

- As 3 letras (rwx) são agrupadas da seguinte forma:

```
-rwxr-xr--   gleydson   users  teste
```

Tipos de Permissões de Acesso

- As 3 letras (rwx) são agrupadas da seguinte forma:

  `-rwxr-xr-- 1 gleydson user 8192 nov 4 16:00 teste`

  explicando cada parte, da esquerda para a direita, para entendermos o que quer dizer as 10 letras acima:

  - A primeira letra (em vermelho) diz qual é o tipo do arquivo. Caso tiver um "`d`" é um diretório, um "`l`" um link a um arquivo no sistema, um "`-`" quer dizer que é um arquivo comum, etc.
  - Da segunda a quarta letra (`rwx`) dizem qual é a permissão de acesso ao dono do arquivo. Neste caso `gleydson` ele tem a permissão de ler (`r - read`), gravar (`w - write`) e executar (`x - execute`) o arquivo `teste`;
  - Da quinta a sétima letra (`r-x`) diz qual é a permissão de acesso ao grupo do arquivo. Neste caso todos os usuários que pertencem ao grupo `users` tem a permissão de ler (`r`), e também executar (`x`) o arquivo `teste`;
  - Da oitava a décima letra (`r--`) diz qual é a permissão de acesso para os `outros usuários`. Neste caso todos os usuários que não são donos do arquivo `teste` têm a permissão somente para ler o arquivo;

- Veja o comando `chmod`, para detalhes sobre a mudança das permissões de acesso de arquivos/diretórios.

Tipos de Permissões de Acesso

- Outro exemplo, desta vez com um diretório:

```
drwxr-x--- 2 gleydson user 1022 nov 4 17:55 exemplo
```

  - A primeira letra (d) determina o tipo de arquivo: "d" é um diretório, um "l" um link para um arquivo no sistema, um "-" quer dizer que é um arquivo comum, etc.

  - Da segunda a quarta letra (rwx) dizem qual é a permissão de acesso ao dono do diretório. Neste caso gleydson ele tem a permissão de listar arquivos (r), gravar arquivos (w) e entrar no diretório (x) exemplo;

  - Da quinta a sétima letra (r-x) diz qual é a permissão de acesso dos usuários que pertencem ao grupo users e que têm permissão de listar arquivos (r) e também entrar (x) no diretório exemplo;

  - Da oitava a décima letra (---) diz qual é a permissão de acesso para os outros usuários. Neste caso todos os usuários que não são donos do diretório exemplo não têm nenhuma permissão.

Tipos de Permissões de Acesso

- OBSERVAÇÕES:
  - O usuário `root` não tem nenhuma restrição de acesso ao sistema;
  - Se você tem permissões de gravação no diretório e tentar apagar um arquivo que você não tem permissão de gravação, o sistema perguntará se você confirma a exclusão do arquivo apesar do modo leitura. Caso você tenha permissões de gravação no arquivo, o arquivo será apagado por padrão sem mostrar nenhuma mensagem de erro (a não ser que seja especificada a opção `-i` com o comando `rm`).
  - Por outro lado, mesmo que você tenha permissões de gravação em um arquivo mas não tenha permissões de gravação em um diretório, a exclusão do arquivo será negada.

A Conta `root`

- A conta `root` é também chamada de super usuário, este é um login que não possui restrições de segurança. A conta `root` somente deve ser usada para fazer a administração do sistema, e usada o menor tempo possível;

- Qualquer senha que criar deverá conter de 6 a 8 caracteres (em sistemas usando criptografia) ou até frases inteiras (caso esteja usando MD5, que garante maior segurança), e também poderá conter letras maiúsculas e minúsculas, e também caracteres de pontuação. Tenha um cuidado especial quando escolher sua senha `root`, porque ela é a conta mais poderosa. Evite palavras de dicionário ou o uso de qualquer outros dados pessoais que podem ser adivinhados.

A Conta `root`

- Se qualquer um lhe pedir senha `root`, seja extremamente cuidadoso. Você normalmente nunca deve distribuir sua conta `root`, a não ser que esteja administrando um computador com mais de um administrador do sistema;

- Utilize uma conta de usuário normal ao invés da conta `root` para operar seu sistema. Porque não usar a conta `root`? Bem, uma razão para evitar usar privilégios `root` é por causa da facilidade de se cometer danos irreparáveis como `root`. Outra razão é que você pode ser enganado e rodar um programa Cavalo de Troia -- que é um programa que obtém poderes do super usuário para comprometer a segurança do seu sistema sem que você saiba.

Comando `chmod`

- Muda a permissão de acesso de um arquivo ou diretório. Com este comando você pode escolher se usuário ou grupo terá permissões para ler, gravar, executar um arquivo ou arquivos. Sempre que um arquivo é criado, seu dono é o usuário que o criou e seu grupo é o grupo do usuário.

```
chmod [opções] [permissões] [diretório/arquivo]
```

- Onde:

`diretório/arquivo`

Diretório ou arquivo que terá sua permissão mudada;

`opções`

`-v, --verbose`

Mostra todos os arquivos que estão sendo processados;

`-f, --silent`

Não mostra a maior parte das mensagens de erro.

Comando chmod

- Continuando...
- Onde:

  `-c, --change`
  Semelhante a opção `-v`, mas só mostra os arquivos que tiveram as permissões alteradas;

  `-R, --recursive`
  Muda permissões de acesso do `diretório/arquivo` no diretório atual e subdiretórios;

  `ugoa+-=rwxXst`
  `ugoa` - Controla que nível de acesso será mudado. Especificam, em ordem, usuário (`u`), grupo (`g`), outros (`o`), todos (`a`).
  `+-=`
  + coloca a permissão,
  – retira a permissão do arquivo e
  = define a permissão exatamente como especificado.
  `rwx`
  `r` permissão de leitura do arquivo,
  `w` permissão de gravação,
  `x` permissão de execução (ou acesso a diretórios).

- `chmod` não muda permissões de links simbólicos, as permissões devem ser mudadas no arquivo alvo do link. Também podem ser usados códigos numéricos octais para a mudança das permissões de acesso a `arquivos/diretórios`;

- DICA: É possível copiar permissões de acesso do `arquivo/diretório`, por exemplo, se o arquivo `teste.txt` tiver a permissão de acesso `r-xr-----` e você digitar `chmod o=u`, as permissões de acesso dos outros usuários (`o`) serão idênticas ao do dono (`u`). Então a nova permissão de acesso do arquivo `teste.txt` será `r-xr--r-x`

Comando chmod

- Exemplos de permissões de acesso:

```
chmod g+r *
```

  - Permite que todos os usuários que pertençam ao grupo dos arquivos (`g`) tenham (`+`) permissões de leitura (`r`) em todos os arquivos do diretório atual;

```
chmod o-r teste.txt
```

  - Retira (`-`) a permissão de leitura (`r`) do arquivo `teste.txt` para os outros usuários (usuários que não são donos e não pertencem ao grupo do arquivo `teste.txt`);

```
chmod uo+x teste.txt
```

  - Inclui (`+`) a permissão de execução (`x`) do arquivo `teste.txt` para o dono e outros usuários do arquivo;

```
chmod a+x teste.txt
```

  - Inclui (`+`) a permissão de execução (`x`) do arquivo `teste.txt` para o dono, grupo e outros usuários;

```
chmod a=rw teste.txt
```

  - Define a permissão de todos os usuários exatamente (`=`) para leitura (`r`) e gravação (`w`) do arquivo `teste.txt`.

Comando `chgrp`

- Muda o grupo de um arquivo/diretório.

```
chgrp [opções] [grupo] [arquivo/diretório]
```

- Onde:
  - `grupo`
    - Novo grupo do `arquivo/diretório`.
  - `arquivo/diretório`
    - `Arquivo/diretório` que terá o grupo alterado.
  - `opções`
    - `-c, --changes`
      - Somente mostra os `arquivos/grupos` que forem alterados;
    - `-f, --silent`
      - Não mostra mensagens de erro para arquivos/diretórios que não puderam ser alterados;
    - `-v, --verbose`
      - Mostra todas as mensagens e arquivos sendo modificados;
    - `-R, --recursive`
      - Altera os grupos de `arquivos/subdiretórios` do diretório atual.

Comando `chown`

- Muda dono de um `arquivo/diretório`. Opcionalmente pode também ser usado para mudar o grupo.

```
chown [opções] [dono.grupo] [diretório/arquivo]
```

- onde:

`dono.grupo`

Nome do `dono.grupo` que será atribuído ao `diretório/arquivo`. O grupo é opcional;

`diretório/arquivo`

`Diretório/arquivo` que o `dono.grupo` será modificado.

`opções`

`-v, --verbose`

Mostra os arquivos enquanto são alterados;

`-f, --supress`

Não mostra mensagens de erro durante a execução do programa.

Comando `chown`

- Continuação...
- onde:

  `-c, --changes`
  Mostra somente arquivos que forem alterados;

  `-R, --recursive`
  Altera dono e grupo de arquivos no diretório atual e subdiretórios.

- O `dono.grupo` pode ser especificado usando o nome de grupo ou o código numérico correspondente ao `grupo` (`GID`).

- Você deve ter permissões de gravação no `diretório/arquivo` para alterar seu `dono/grupo`:

  `chown gleydson teste.txt` - Muda o dono do arquivo `teste.txt` para `gleydson`;

  `chown gleydson.foca teste.txt` - Muda o dono do arquivo `teste.txt` para `gleydson` e seu grupo para foca;

  `chown -R gleydson.focalinux *` - Muda o `dono/grupo` dos arquivos do diretório atual e subdiretórios para `gleydson/focalinux` (desde que você tenha permissões de gravação no diretórios e subdiretórios).

Modo de Permissão Octal

- Ao invés de utilizar os modos de permissão `+r`, `-r`, etc, pode ser usado o modo octal para se alterar a permissão de acesso a um arquivo. O modo octal é um conjunto de oito números onde cada número define um tipo de acesso diferente.

- É mais flexível gerenciar permissões de acesso usando o modo octal ao invés do comum, pois você especifica diretamente a permissão do `dono`, `grupo`, `outros` ao invés de gerenciar as permissões de cada um separadamente. Abaixo a lista de permissões de acesso octal:
  - `0` - Nenhuma permissão de acesso. Equivalente a `-rwx`;
  - `1` - Permissão de execução (`x`);
  - `2` - Permissão de gravação (`w`);
  - `3` - Permissão de gravação e execução (`wx`). Equivalente a permissão `2+1`
  - `4` - Permissão de leitura (`r`);
  - `5` - Permissão de leitura e execução (`rx`). Equivalente a permissão `4+1`
  - `6` - Permissão de leitura e gravação (`rw`). Equivalente a permissão `4+2`
  - `7` - Permissão de leitura, gravação e execução. Equivalente a `+rwx` (`4+2+1`).

Modo de Permissão Octal

- O uso de um deste números define a permissão de acesso do dono, grupo ou outros usuários. Um modo fácil de entender como as permissões de acesso octais funcionam, é através da seguinte tabela:

  `1 = Executar`

  `2 = Gravar`

  `4 = Ler`

  * Para `Dono` e `Grupo`, multiplique as permissões acima por x100 e x10.

- Basta agora fazer o seguinte:

  Somente permissão de execução, use `1`

  Somente a permissão de leitura, use `4`

  Somente permissão de gravação, use `2`

  Permissão de leitura/gravação, use `6` (equivale a `2+4` / `Gravar+Ler`).

  Permissão de leitura/execução, use `5` (equivale a `1+4` / `Executar+Ler`).

  Permissão de execução/gravação, use `3` (equivale a `1+2` / `Executar+Gravar`).

  Permissão de leitura/gravação/execução, use `7` (equivale a `1+2+4` / `Executar+Gravar+Ler`).

Modo de Permissão Octal

- Alguns exemplos:

```
"chmod 764 teste"
```

- Os números são interpretados da direita para a esquerda como permissão de acesso aos `outros usuários` (`4`), `grupo` (`6`), e `dono` (`7`). O exemplo acima faz os `outros usuários` (`4`) terem acesso somente `leitura` (`r`) ao arquivo `teste`, o `grupo` (`6`) ter a permissão de `leitura` e `gravação` (`rw`), e o `dono` (`7`) ter permissão de `leitura`, `gravação` e `execução` (`rwx`) ao arquivo `teste`.

- Outro exemplo:

```
"chmod 40 teste"
```

- O exemplo acima define a permissão de acesso dos `outros usuários` (`0`) como nenhuma, e define a permissão de acesso do `grupo` (`4`) como somente `leitura` (`r`). Note que foi usado somente dois números e então a permissão de acesso do dono do arquivo não é modificada (leia as permissões de acesso da direita para a esquerda!).

```
"chmod 751 teste"
```

- O exemplo acima define a permissão de acesso dos `outros usuários` (`1`) para somente `execução` (`x`), o acesso do `grupo` (`5`) como `leitura` e `execução` (`rx`) e o acesso do `dono` (`7`) como `leitura`, `gravação` e `execução` (`rwx`).

Comando `umask`

- A `umask` (`user mask`) são 3 números que definem as permissões iniciais do `dono`, `grupo` e `outros usuários` que o `arquivo/diretório` receberá quando for criado ou copiado para um novo local. Digite `umask` sem parâmetros para retornar o valor de sua `umask` atual;

- A `umask` tem efeitos diferentes caso o arquivo que estiver sendo criado for binário (um programa executável) ou texto.

Comando `umask`

- Veja a tabela a seguir para ver qual é a mais adequada a sua situação:

UMASK	ARQUIVO		DIRETÓRIO
	Binário	Texto	
0	r-x	rw-	rwx
1	r--	rw-	rw-
2	r-x	r--	r-x
3	r--	r--	r--
4	--x	-w-	-wx
5	---	-w-	-w-
6	--x	---	--x
7	---	---	---

Comando `umask`

- Um arquivo texto criado com o comando `umask 012;touch texto.txt` receberá as permissões `-rw-rw-r--`, pois `0` (`dono`) terá permissões `rw-`, `1` (`grupo`), terá permissões `rw-` e `2` (`outros usuários`) terão permissões `r--`. Um arquivo binário copiado com o comando `umask 012;cp /bin/ls /tmp/ls` receberá as permissões `-r-xr--r-x` (confira com a tabela no *slide* anterior);

- Por este motivo é preciso atenção antes de escolher a `umask`, um valor mal escolhido poderia causar problemas de acesso a arquivos, diretórios ou programas não sendo executados. O valor padrão da `umask` na maioria das distribuições atuais é `022`. A `umask` padrão no sistema Debian é a `022`;

- A `umask` é de grande utilidade para programas que criam `arquivos/diretórios` temporários, desta forma pode-se bloquear o acesso de outros usuários desde a criação do arquivo, evitando recorrer ao `chmod`.

Referências

- GUIA FOCA GNU/Linux. Iniciante. Disponível em: <http://www.guiafoca.org/cgs/guia/iniciante/ch-perm.html>. Acesso em: 06 ago. 2017.