Guia do Linux/Avançado/Apache/Restrições de Acesso

Origem: Wikilivros, livros abertos por um mundo aberto.
< Guia do Linux | Avançado | Apache

Índice

- 1 Restrições de Acesso
  - 1.1 Autorização
  - 1.2 Autenticação
    - 1.2.1 Criando um arquivo de Senhas
      - 1.2.1.1 .1 htpasswd
      - 1.2.1.2 .2 htdigest e dbmmanage
    - 1.2.2 Autenticação através de usuários
    - 1.2.3 Autenticação usando grupos
  - 1.3 Usando autorização e autenticação juntos
    - 1.3.1 Acesso diferenciado em uma mesma diretiva
  - 1.4 O arquivo .htaccess
  - 1.5 Usando a diretiva SetEnvIf com Allow e Deny
  - 1.6 A diretiva <Limit>
  - 1.7 Diretiva <LimitExcept>

Restrições de Acesso

A restrição de acesso do `Apache` é feita através de *Autorização* ([#s-s-apache-acesso-restr-autor Autorização, Seção 12.7.1]) e *Autenticação* ([#s-s-apache-acesso-restr-auth Autenticação, Seção 12.7.2]). Através da *autorização*, é checado se o endereço/rede especificada tem ou não permissão para acessar a página. A *autenticação* requer que seja passado nome e senha para garantir acesso a página. Os métodos de *Autorização* e *Autenticação* podem ser combinados como veremos mais adiante.

Autorização

A restrição de acesso por autorização (controlado pelo módulo `mod_access`), permite ou não o acesso ao cliente de acordo com o endereço/rede especificada. As restrições afetam também os sub-diretórios do diretório alvo. Abaixo um exemplo de restrição de acesso que bloqueia o acesso de qualquer host que faz parte do domínio *.spammers.com.br* a URL `http://servidor/teste`:

```
<Location /teste>
 Option Indexes
 Order allow,deny
 allow from all
 deny from .spammers.com.br
</Location>
```

A opção Option foi explicada acima, seguem as explicações das outras diretivas:

- **Order**
  Especifica em que ordem as opções de acesso *allow/deny* serão pesquisadas. Caso não seja especificada, o padrão será *deny/allow*. Note que a ordem de pesquisa de *allow* e *deny* é a inversa

da especificada. A diretiva *Order* aceita os seguintes valores:

- deny,allow - Esta é a padrão, significa um servidor mais restritivo; a diretiva *allow* é processada primeiro e somente depois a diretiva *deny*. Caso nenhuma diretiva allow e deny forem especificadas ou não conferirem, **PERMITE TUDO** como padrão.
- allow,deny - Significa um servidor mais permissivo, a opção *deny* é processada primeiro e somente depois a opção *allow*. Caso nenhuma diretiva allow e deny for especificadas ou não conferirem, **BLOQUEIA TUDO** como padrão.
- mutual-failure - Somente permite o acesso se o usuário receber autorização através da opção *allow* e **NÃO** ser bloqueado pela opção *deny*, caso uma das checagens falhe, o acesso é imediatamente negado. É uma opção interessante quando você quer somente pessoas de um determinado endereço/rede acessando o seu sistema e não estejam em sua lista negra :-)

ATENÇÃO: É importante saber se a página será permissiva ou restritiva para escolher a ordem mais adequada ao seu caso, também leve em consideração a possibilidade do processamento cair na diretiva de acesso padrão, caso nem a diretiva allow e deny conferiram e estiver usando a ordem de acesso "allow,deny" ou "deny,allow". Um sistema mal configurado neste aspecto poderá trazer sérias consequências. É comum em páginas permissivas se definir a seguinte configuração:

```
Order allow,deny
allow from all
```

O motivo é que em um grande site, se forem adicionadas mais restrições nesta página (devido a alguns domínios que tem usuários mal comportados, bloqueio de acesso a rede do concorrente, potenciais atacantes, etc...), estas deverão ser lidas antes da diretiva "allow from all" e podem passar desapercebidas ao administrador e podem simplesmente não funcionar caso a opção *Order* não esteja ajustada corretamente (lembre-se, você é o administrador e a integridade do site depende de sua atenção na escolha da ordem correta das diretivas de acesso).

- **allow from**
  Especifica o endereço que terá acesso ao recurso especificado. A diretiva *allow* from aceita os seguintes valores:
  - all - O acesso é permitido a todos.
  - um endereço de domínio completo (FQDN). Por exemplo `www.debian.org.br`.
  - um endereço de domínio parcial. Qualquer computador que confira com o inicio ou fim terá o acesso permitido. Por exemplo, `.spammers.com.br`, `.debian.org`.
  - um endereço IP completo, como 192.168.1.1
  - um endereço IP parcial como 192.168.1.
  - um par rede/máscara como 10.1.0.0/255.255.0.0 ou 10.1.0.0/16, uma faixa de acesso a máquinas de uma mesma rede pode ser definida facilmente através deste método.

OBS1: É necessário reiniciar o `Apache` depois de qualquer modificação em seu arquivo de configuração (executando apachectl restart), ou recarregar os arquivos de configuração (apachectl graceful). **OBS2**: Mais de um host pode ser especificado separando com um espaço:

```
allow from 192.168. .debian.org.br
```

Permitirá o acesso de qualquer máquina que o endereço IP confira com 192.168.*.* e qualquer computador do domínio debian.org.br **OBS3**: Regras baseadas em nomes simples de hosts (como www) não conferirão! Deverá ser usado o FQDN ou IP: www.dominio.com.br **OBS4**: Caso Order não seja especificado, *deny,allow* será usado como padrão (ou seja, permitirá tudo como padrão).

- **deny from**
  Especifica os endereços que NÃO terão acesso ao recurso especificado. As explicações referentes a esta diretiva de acesso são idêntica as de *allow from*.

É recomendável o uso de endereços IP ao invés de endereços DNS e um mecanismo anti-spoofing no firewall ou código de roteamento, pois ficará mais difícil um ataque baseado em DNS spoofing, aumentando consideravelmente a segurança de seu servidor web. **ATENÇÃO**: Caso receba erros 403 (acesso negado) sem bloquear a URL nas diretivas de acesso, uma dos seguintes problemas pode ser a causa:

- O servidor Web não tem permissões para acessar/abrir o diretório da página. Certifique-se que o *dono* e *grupo* do processo `Apache` (especificado pela diretiva *User* e *Group*) possuem permissões de acesso Ã quele diretório.
- Quando quer fazer uma listagem de arquivos do diretório e não especifica a opção Option Indexes como opção de listagem.
- Quando não está usando Option Indexes para impedir a listagem de conteúdo do diretório e o não foi encontrado um arquivo de índice válido dentre os existentes na diretiva DirectoryIndex no diretório atual.

Abaixo alguns exemplos de permissões de acesso:

```
<Directory /var/www>
 Options SymLinksIfOwnerMatch Indexes MultiViews
 Order allow,deny
 allow from all
</Directory>
```

Permite o acesso a de qualquer usuário de qualquer lugar (allow from all), permite também a visualização da listagem formatada de arquivos caso nenhum arquivo especificado na diretiva *DirectoryIndex* seja encontrado (Indexes), permite negociação de conteúdo (*MultiViews*) e seguir links caso o dono do arquivo confira com o nome do link (*SymLinksIfOwnerMatch*).

```
<Directory /var/www>
 Options SymLinksIfOwnerMatch Indexes MultiViews
</Directory>
```

Tem o mesmo significado da diretiva acima por métodos diferentes; quando nenhuma opção *Order* é especificada, *deny,allow* é definido como padrão, e como nenhuma opção de acesso *allow/deny* foi especificada, o padrão "Order deny,allow" é usado e permite TUDO como padrão.

```
<Directory /var/www>
 Options Indexes
 Order deny,allow
 deny from all
</Directory>
```

Esta regra acima não tem muita lógica pois restringe o acesso de todos os usuários ao diretório `/var/www`, ao menos se esta for sua intenção...

```
<Location /focalinux>
 Options All
 Order allow,deny
 allow from all
</Location>
```

A regra acima permite o acesso a URL `http://www.servidor.org/focalinux` de qualquer host na Internet

```
<Files .htaccess>
 Order deny,allow
 deny from all
</Files>
```

Bloqueia o acesso a qualquer arquivo `.htaccess` do sistema

```
<Files ~ "leiame-(arm|alpha|m68k|sparc|powerpc)\.txt">
 Order deny,allow
 deny from all
</Files>
```

Bloqueia o acesso a qualquer arquivo `leiame-arm.txt`, `leiame-alpha.txt`, `leiame-m68k.txt`, `leiame-sparc.txt` e `leiame-powerpc.txt` fazendo uso de expressões regulares.

```
<Directory /var/www>
 Options Indexes
 Order mutual-failure
 allow from .dominio.com.br
 deny from lammer.dominio.com.br
</Directory>
```

A diretiva acima somente permite acesso ao diretório `/var/www` de máquinas pertencentes ao domínio `.dominio.com.br` desde que não seja `lammer.dominio.com.br`.

```
<Directory /var/www>
 Options Indexes MultiViews
 Order allow,deny
 deny from .com .com.br
 allow from all
</Directory>
```

Bloqueia o acesso ao diretório `/var/www` de computadores pertencentes aos domínios *.com* e *.com.br*.

```
<Directory /var/www>
 Options None
 Order deny,allow
 allow from 192.168.1. .guiafoca.org .debian.org
 deny from 200.200.123.
</Directory>
```

A regra acima permite o acesso de máquinas da rede 192.168.1.*, do domínio *.guiafoca.org e *.debian.org, o acesso de máquinas da rede 200.200.123.* é bloqueado (nada contra, peguei nesse número ao acaso :-). Note que a máquina 192.168.4.10 terá acesso LIVRE a regra acima, pois não conferirá nem com *allow* nem com *deny*, então o processamento cairá na diretiva padrão de *deny,allow*, que neste caso permite o acesso caso nem *allow* e *deny* conferiram com o padrão.

```
<Directory /var/www>
 Options None
 Order allow,deny
 allow from 192.168.1. .cipsga.org.br .debian.org
 deny from 200.200.123.
</Directory>
```

A regra acima é idêntica a anterior somente com a mudança da opção *Order*. Bloqueia o acesso de máquinas da rede 200.200.123.* e permite o acesso de máquinas da rede 192.168.1.*, do domínio *.cipsga.org.br e *.debian.org. Note que a máquina 192.168.4.10 terá acesso BLOQUEADO a regra acima, pois não conferirá nem com *allow* nem com *deny*, então o processamento cairá na diretiva padrão de *allow,deny* que neste caso bloqueia o acesso.

Autenticação

Através da *autenticação* (controlado pelo módulo `mod_auth`) é possível especificar um *nome* e *senha* para acesso ao recurso solicitado. As senhas são gravadas em formato criptografado usando *Crypto* ou *MD5* (conforme desejado). O arquivo de senhas pode ser centralizado ou especificado individualmente por usuário, diretório ou até mesmo por arquivo acessado.

Criando um arquivo de Senhas

O arquivo de senhas pode ser criado e mantido através do uso de 3 utilitários: `htpasswd`, `htdigest` e `dbmmanage`:

.1 htpasswd

Este é usado para criar o arquivo de senhas. Para criar um banco de dados com o nome `senhas` para o usuário *convidado*, é usada a seguinte sintaxe: htpasswd -c -m senhas convidado Você será perguntado por uma senha para o usuário *convidado* e para redigita-la. A opção "-c" indica que deverá ser criado um arquivo, a opção "-m" indica a utilização de senhas criptografadas usando o algoritmo *MD5*, que garante maior segurança que o método *Crypto*. A senha pode ser especificada diretamente na linha de comando através da opção "-b" (isto é um ótimo recurso para utilização em shell scripts ou programas CGI de integração com o navegador). htpasswd -b -d senhas chefe abcdef No exemplo acima, uma senha de alta segurança será introduzida no banco de dados `senhas` tornando impossível o acesso a página do usuário :-) Note que esta senha foi cadastrada usando o algoritmo de criptografia Crypto (opção -d). O algoritmo *SHA* também pode ser usado como alternativa, através da opção "-s". Para modificar a senha do usuário convidado, basta usar a mesma sintaxe (sem a opção "-c" que é usada para criar um novo arquivo): htpasswd -m senhas convidado ou htpasswd -b -m senhas convidado nova_senha Opcionalmente você pode especificar a opção "-d" para atualizar também o formato da senha para *Crypto*. Podem existir senhas de criptografias mistas (*SHA, Crypto, MD5*) no mesmo arquivo sem nenhum problema. A mudança do formato de senhas é útil quando se deseja aumentar o nível de segurança oferecido por um melhor sistema ou para manter a compatibilidade com alguns scripts/programas que compartilhem o arquivo de senhas.

.2 htdigest e dbmmanage

Estes são idênticos ao `htpasswd`, a diferença é que o `htdigest` permite criar/manter um arquivo de senhas usando a autenticação Digest, enquanto o `dbmmanage` permite manter o banco de dados de senhas em um arquivo `DB, DBM, GDBM` e `NDBM`, formatos conhecidos pelo Perl.

Autenticação através de usuários

Através deste método é possível especificar que usuários terão acesso ao recurso definido, usando senhas de acesso individuais criptografadas usando um dos utilitários da seção anterior. Para restringir o acesso ao endereço `http://servidor.org/teste`:

```
<Location /teste>
 AuthName "Acesso a página do Foca Linux"
 AuthType basic
 AuthUserFile /home/gleydson/SenhaUsuario
# AuthGroupFile /home/users/SenhaGrupo
 Require valid-user
</Location>
```

Ao tentar acessar o endereço `http://servidor/teste`, será aberta uma janela no navegador com o título *Enter username for Acesso a página do Foca Linux at servidor.org*, a diretiva *Require valid-user* definem que o usuário e senha digitados devem existir no arquivo especificado por *AuthUserFile* para que o acesso seja garantido. Uma explicação de cada opção de acesso usado na autenticação:

- **AuthName**

Será o nome que aparecerá na janela de autenticação do seu navegador indicando qual área restrita está solicitando senha (podem existir várias no servidor, bastando especificar várias diretivas de restrições).

AuthType

Especifica o método de que o nome e senha serão passados ao servidor. Este método de autenticação pode ser *Basic* ou *Digest*

- Basic - Utiliza a codificação *base64* para encodificação de nome e senha, enviando o resultado ao servidor. Este é um método muito usado e pouco seguro, pois qualquer sniffer instalado em um roteador pode capturar e descobrir facilmente seu nome e senha.
- Digest - Transmite os dados de uma maneira que não pode ser facilmente decodificada, incluindo a codificação da área protegida (especificada pela diretiva *AuthName*) que possui a sequencia de login/senha válida. A diferença deste método é que você precisará de arquivos de senhas diferentes para cada área protegida especificada por *AuthName* (também chamada de Realm).

AuthUserFile

É o arquivo gerado pelo utilitário `htpasswd` que contém a senha correspondente ao usuário

AuthGroupFile

É um arquivo texto que contém o nome do grupo, dois pontos (":") e o nome dos usuários que podem ter acesso ao recurso, separados por vírgulas. No exemplo acima ele se encontra comentado, mas a seguir encontrará exemplos que explicam em detalhes o funcionamento desta diretiva.

Require

Especifica que usuários podem ter acesso ao diretório. Podem ser usadas uma das 3 sintaxes:

- Require user usuário1 usuário2 usuário3 - Somente os usuários especificados são considerados válidos para ter acesso ao diretório.
- Require group grupo1 grupo2 grupo3 - Somente os usuários dos grupos especificados são considerados válidos para terem acesso ao diretório. Esta diretiva é útil quando deseja que somente alguns usuários de determinado grupo tenham acesso ao recurso (por exemplo, usuários do grupo admins).
- Require valid-user - Qualquer usuário válido no banco de dados de senhas pode acessar o diretório. É bem útil quando as opções de acesso especificadas por Require user são muito longas.

A opção Require deve ser acompanhado das diretivas *AuthName*, *AuthType* e as diretivas *AuthUserFile* e *AuthGroupFile* para funcionar adequadamente. **OBS**: É necessário reiniciar o `Apache` depois de qualquer modificação em seu arquivo de configuração (apachectl restart), ou recarregar os arquivos de configuração (apachectl graceful). Note que o `apachectl` é somente um shell script para interação mais amigável com o servidor web `apache`, retornando mensagens indicando o sucesso/falha no comando ao invés de códigos de saída. Alguns exemplos para melhor assimilação:

```
<Location /teste>
 AuthName "Acesso a página do Foca Linux"
 AuthType basic
 AuthUserFile /home/gleydson/SenhaUsuario
 Require user gleydson
</Location>
```

As explicações são idênticas a anterior, mas somente permite o acesso do usuário gleydson a URL `http://servidor.org/teste`, bloqueando o acesso de outros usuários contidos no arquivo *AuthUserFile*.

```
<Location /teste>
 AuthName "Acesso a página do Foca Linux"
 AuthType basic
 AuthUserFile /home/gleydson/SenhaUsuario
 Require user gleydson usuario1 usuario2
</Location>

<Location /teste>
 AuthName "Acesso a página do Foca Linux"
```

```
 AuthType basic
 AuthUserFile /home/gleydson/SenhaUsuario
 Require user gleydson
 Require user usuario1
 Require user usuario2
</Location>
```

As 2 especificações acima são equivalentes e permite o acesso aos usuários gleydson, usuario1 e usuario2 a página `http://servidor.org/teste`.

Autenticação usando grupos

Há casos onde existem usuários de um arquivo de senhas que devem ter acesso a um diretório e outros não, neste caso a diretiva *valid-user* não pode ser especificada (porque permitiria o acesso de todos os usuários do arquivo de senha ao diretório) e uma grande lista de usuários ficaria bastante complicada de ser gerenciada com vários usuários na diretiva *Require user*. Quando existe esta situação, é recomendado o uso de grupos de usuários. Para fazer uso desse recurso, primeiro deverá ser criado um arquivo quer armazenará o nome do *grupo* e dos usuários pertencente Ã quele grupo usando a seguinte sintaxe (vamos chamar este arquivo de `SenhaGrupo`):

```
admins: gleydson usuario2
usuarios: usuario1 usuario2 usuario3 gleydson
```

Agora adaptamos o exemplo anterior para que somente os usuários especificados no grupo admins do arquivo criado acima:

```
<Location /teste>
 AuthName "Acesso a página do Foca Linux"
 AuthType basic
 AuthUserFile /home/gleydson/SenhaUsuario
 AuthGroupFile /home/gleydson/SenhaGrupo
 Require group admins
</Location>
```

Agora somente os usuários pertencentes ao grupo *admins* (*gleydson* e *usuario2*) poderão ter acesso ao diretório `/teste`. **OBS1**: Verifique se o servidor Web possui acesso a leitura no arquivo de senhas de usuários e grupos, caso contrário será retornado um código "500 - Internal Server Error". Este tipo de erro é caracterizado por tudo estar OK na sintaxe dos arquivos de configuração após checagem com "apache -t" e todas as diretivas de controle de acesso apontam para os diretórios e arquivos corretos. **OBS2:**: Sempre use espaços para separar os nomes de usuários pertencentes a um grupo. **OBS3**: NUNCA coloque os arquivos que contém senhas e grupos em diretórios de acesso público onde usuários podem ter acesso via o servidor Web. Tais localizações são `/var/www`, `/home/"usuario"/public_html` e qualquer outro diretório de acesso público que defina em seu sistema. É recomendável também ocultar estes arquivos através da diretiva <Files> evitando possíveis riscos de segurança com usuários acessando os arquivos de senha e grupo. Na distribuição `Debian`, qualquer arquivo iniciando com `.ht*` será automaticamente ocultado pelo sistema, pois já existe uma diretiva <Files ~ "\.ht">. Tal diretiva pode também ser especificada no arquivo de acesso `.htaccess`. Assim um arquivo `.htsenha` e `.htgroup` são bons nomes se estiver desejando ocultar dados de olhos curiosos...

Usando autorização e autenticação juntos

Os métodos de *autorização* e *autenticação* podem ser usados ao mesmo tempo dentro de qualquer uma das diretivas de controle de acesso. As diretivas de *autorização* são processadas primeiro (mod_access) e depois as diretivas de *autenticação* (mod_auth). Segue um exemplo:

```
<Directory /var/www>
 Options Indexes
 Order deny,allow
```

```
 allow from .dominiolocal.com.br
 deny from all
 AuthName "Acesso ao diretório do servidor Web"
 AuthType basic
 AuthUserFile /var/cache/apache/senhas
 Require valid-user
</Directory>
```

Para ter acesso ao diretório `/var/www`, primeiro o computador deve fazer parte do domínio `.dominiolocal.com.br`, assim ela passa pelo teste de autorização, depois disso será necessário fornecer o login e senha para acesso a página, digitando o login e senha corretos, o teste de autenticação será completado com sucesso e o acesso ao diretório `/var/www` autorizado.

```
<Directory /var/www>
 Options Indexes
 Order mutual-failure
 allow from .dominiolocal.com.br
 deny from lammer.dominiolocal.com.br
 AuthName "Acesso ao diretório do servidor Web"
 AuthType basic
 AuthUserFile /var/cache/apache/senhas
 AuthGroupFile /var/cache/apache/grupos
 Require group admins
</Directory>
```

No exemplo acima, é usado o método de autorização com a opção *Order mutual-failure* e o método de autenticação através de *grupos*. Primeiro é verificado se o usuário pertence ao domínio `.dominiolocal.com.br` e se ele não está acessando da máquina `lammer.dominiolocal.com.br`, neste caso ele passa pelo teste de autorização. Depois disso ele precisará fornecer o nome e senha válidos, com o login pertencente ao *AuthGroupFile*, passando pelo processo de autenticação e obtendo acesso ao diretório `/var/www`.

Acesso diferenciado em uma mesma diretiva

É interessante permitir usuários fazendo conexões de locais confiáveis terem acesso direto sem precisar fornecer nome e senha e de locais inseguros acessarem somente após comprovarem **quem** realmente são. Como é o caso de permitir usuários de uma rede privada terem acesso completo aos recursos e permitir o acesso externo ao mesmo recurso somente através de senha. Isto pode ser feito com o uso da diretiva *Satisfy* junto ao bloco de *autorização/autenticação*. Vamos tomar como base o exemplo anterior:

```
<Directory /var/www>
 Options Indexes
 Order mutual-failure
 allow from .dominiolocal.com.br
 deny from lammer.dominiolocal.com.br
 AuthName "Acesso ao diretório do servidor Web"
 AuthType basic
 AuthUserFile /var/cache/apache/senhas
 AuthGroupFile /var/cache/apache/grupos
 Require group admins
 Satisfy any
</Directory>
```

Note que o exemplo é o mesmo com a adição da diretiva *Satisfy any* no final do bloco do arquivo. Quando a opção *Satisfy* não é especificada, ela assumirá "all" como padrão, ou seja, o usuário deverá passar no teste de autorização e autenticação para ter acesso. A diferença do exemplo acima em relação ao da seção anterior é se a máquina passar no teste de autorização ela já terá acesso garantido. Caso falhe no teste de autorização, ainda terá a chance de ter acesso a página passando na checagem de autenticação. Isto garante acesso livre aos usuários do domínio `.dominiolocal.com.br`. Já os outros usuários, incluindo acessos vindos de `lammer.dominiolocal.com.br` que pode ser uma máquina com muito uso, poderá ter acesso ao recurso caso tenha fornecido um nome e senha válidos para passar pelo processo de autenticação. Tenha isto em mente... este

tipo de problema é comum e depende mais de uma política de segurança e conduta interna, o sistema de segurança não pode fazer nada a não ser permitir acesso a um nome e senha válidos. Tenha cuidado com o uso da opção *Satisfy* em diretivas que especificam somente o método de autenticação:

```
<Directory /var/www>
 Options Indexes
 AuthName "Acesso ao diretório do servidor Web"
 AuthType basic
 AuthUserFile /var/cache/apache/senhas
 AuthGroupFile /var/cache/apache/grupos
 Require group admins
 Satisfy any
</Directory>
```

ATENÇÃO PARA O DESCUIDO ACIMA!: Como o método de autorização NÃO é especificado, é assumido *deny,allow* como padrão, que permite o acesso a TODOS os usuários. O bloco acima **NUNCA** executará o método de autenticação por este motivo. A melhor coisa é NÃO usar a opção *Satisfy* em casos que só requerem autenticação ou usar *Satisfy all* (que terá o mesmo efeito de não usa-la, hehehe). A falta de atenção nisto pode comprometer silenciosamente a segurança de seu sistema.

O arquivo `.htaccess`

O arquivo `.htaccess` deve ser colocado no diretório da página que deverá ter suas permissões de acesso/listagem controladas. A vantagem em relação a inclusão direta de diretivas de acesso dentro do arquivo de configuração do `Apache`, é que o controle de acesso poderá ser definido pelo próprio webmaster da página, sem precisar ter acesso direto a configuração do `Apache`, que requerem privilégios de root. Outro ponto fundamental é que não há necessidade de reiniciar o servidor Web, pois este arquivo é lido no momento de cada acesso ao diretório que controla. O nome do arquivo OverRide pode ser definido através da diretiva *AccessFileName* no arquivo de configuração do `Apache`, `.htaccess` é usado como padrão. O controle de que opções estarão disponíveis no `.htaccess` são definidas na diretiva *AllowOverride* que pode conter o seguintes parâmetros:

- None - O servidor não buscará o arquivo .htaccess nos diretórios
- All - O servidor utilizará todas as opções abaixo no arquivo `.htaccess`
- AuthConfig - Permite o uso de diretivas de autenticação (*AuthDBMGroupFile*, *AuthDBMUserFile*, *AuthGroupFile*, *AuthName*, *AuthType*, *AuthUserFile*, *Require*, etc.).
- FileInfo - Permite o uso de diretivas controlando o tipo de documento (*AddEncoding*, *AddLanguage*, *AddType*, *DefaultType*, *ErrorDocument*, *LanguagePriority*, etc.).
- Indexes - Permite o uso de diretivas controlando a indexação de diretório (*AddDescription*, *AddIcon*, *AddIconByEncoding*, *AddIconByType*, *DefaultIcon*, *DirectoryIndex*, *FancyIndexing*, *HeaderName*, *IndexIgnore*, *IndexOptions*, *ReadmeName*, etc.).
- Limit - Permite o uso de diretivas controlando o acesso ao computador (*allow*, *deny* e *order*).
- Options - Permite o uso de diretivas controlando características específicas do diretório (*Options* e *XBitHack*).

OBS: Não tem sentido usar a opção *AllowOverride* dentro da diretiva <Location>, ela será simplesmente ignorada. Para acesso ao arquivo `.htaccess` do diretório `/var/www/focalinux`, o `Apache` buscará os arquivos `.htaccess` na sequencia: `/.htaccess`, `/var/.htaccess`, `/var/www/.htaccess`, `/var/www/focalinux/.htaccess`, qualquer diretiva que não exista no `.htaccess` do diretório `/var/www/focalinux` terá seu valor definido pela diretiva dos arquivos `.htaccess` dos diretórios anteriores. Somente após esta sequencia de checagens o acesso ao documento é permitido (ou negado). Por este motivo, muitos administradores decidem desativar completamente o uso de arquivos `.htaccess` no diretório raíz e habilitar somente nos diretórios especificados pela diretiva <Directory> no arquivo de configuração do `Apache`, evitando brechas de segurança na manipulação destes arquivos (esta é uma boa idéia a não ser que se dedique 24 horas somente na administração do seu servidor Web e conheça toda sua estrutura hierárquica de segurança:

```
<Directory />
 AllowOverride none
</Directory>

<Directory /var/www>
 AllowOverride limit authconfig indexes
</Directory>
```

Na especificação acima, o arquivo `.htaccess` será procurado no diretório `/var/www` e seus sub-diretórios, usando somente opções que controlam a autorização de acesso (*limit*), autenticação e opções (*authconfig*) e de indexação de documentos (*indexes*). Alguns exemplos do uso do arquivo `.htaccess`: Para permitir o acesso direto de usuários da rede 192.168.1.* diretamente, e requerer senha de acesso para outros usuários, o seguinte arquivo `.htaccess` deve ser criado no diretório `/var/www`:

```
Order deny,allow
allow from 192.168.1.0/24
deny from all
AuthName "Acesso a página Web principal da Empresa"
AuthType basic
AuthUserFile /var/cache/apache/senhas
Require valid-user
Satisfy any
```

Note que a sintaxe é exatamente a mesma das usadas na diretivas de acesso, por este motivo vou dispensar explicações detalhadas a respeito. **ATENÇÃO**: A diretiva *Options Indexes* deverá ser especificada no *AllowOverRide* e não no arquivo `.htaccess`. Agora você já sabe o que fazer se estiver recebendo erros 500 ao tentar acessar a página (Erro interno no servidor)...

Usando a diretiva SetEnvIf com Allow e Deny

É possível especificar o acesso baseado em variáveis de ambiente usando a diretiva *SetEnvIf*, isto lhe permite controlar o acesso de acordo com o conteúdo de cabeçalhos HTTP. A sintaxe é a seguinte: SetEnvIf [*atributo*] [*expressão*] [*variável*] Isto poder ser facilmente interpretado como: Se o "atributo" especificado conter a "expressão", a "variável" será criada e armazenará o valor verdadeiro. Veja abaixo:

```
SetEnvIf User-Agent ".*MSIE*." EXPLODER
<Directory /var/www>
 Order deny,allow
 allow from all
 deny from env=EXPLODER
</Directory>
```

Se o Navegador (campo *User-Agent* do cabeçalho http) usado para acessar a página for o Internet Explorer, a variável *EXPLODER* será criada e terá o valor verdadeiro (porque a expressão de *SetEnvIf* conferiu com a expressão). Note o uso de "deny from env=VARIÁVEL". Neste caso se o navegador for o Internet Explorer, o acesso será bloqueado (pois o navegador conferiu, assim a variável *EXPLODER* recebeu o valor verdadeiro). É permitido especificar as diretivas de acesso normais junto com especificação de variáveis de ambiente, basta separa-los com espaços. Uma descrição completa dos cabeçalhos HTTP, conteúdo e parâmetros aceitos por cada um são descritos na RFC 2068.

A diretiva <Limit>

Esta diretiva é semelhante a <Directory> mas trabalha com métodos HTTP (como GET, PUT, POST, etc) ao invés de diretórios. A diretiva <Limit> pode ser usada dentro da diretiva de acesso <Directory>, <Location>, mas nenhuma diretiva de controle de acesso pode ser colocada dentro de <Limit>. Os métodos HTTP válidos são: GET, POST, PUT DELETE, CONNECT, OPTIONS, TRACE, PATCH, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK e UNLOCK. Note que os métodos são case-sensitive. Por exemplo:

```
<Directory /var/www>
 Option Indexes
  <Limit POST PUT DELETE>
     Order deny,allow
     allow from 192.168.1.0/24
     deny from all
   </Limit>
</Directory>
```

Somente permitem o uso dos métodos POST, PUT, DELETE de máquinas da rede interna. **OBS1**: Se o método GET é bloqueado, o cabeçalho HTTP também será bloqueado. **OBS2**: A diretiva de acesso <Limit> somente terá efeito na diretiva <Location> se for especificada no arquivo de configuração do servidor web. A diretiva <Location> simplesmente é ignorada nos arquivos `.htaccess`... Este abaixo é usado por padrão na distribuição `Debian` para restringir para somente leitura o acesso aos diretórios de usuários acessados via módulo `mod_userdir`:

```
<Directory /home/*/public_html>
    AllowOverride FileInfo AuthConfig Limit
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    <Limit GET POST OPTIONS PROPFIND>
        Order allow,deny
        Allow from all
    </Limit>
    <Limit PUT DELETE PATCH PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
        Order deny,allow
        Deny from all
    </Limit>
</Directory>
```

Diretiva <LimitExcept>

Esta diretiva é semelhante a <Limit>, mas atinge todos os métodos HTTP, menos os especificados.