# Linux Network Servers

## Squid (Proxy)

**Objetivo**

Otimizar a velocidade de conteúdos web. É amplamente usado em ambientes corporativos, tendo como vantagem fazer cache de conteúdo, filtrar sites indesejados, controlar acessos e diminuir a utilização da banda de internet.

Muitas pessoas usam um web proxy até mesmo sem saber, e no mundo GNU/Linux o Squid é o web proxy mais comum. Ele foi baseado no Harvest Cache Daemon desenvolvido na década de 90. E é um HTTP/1.0 proxy completo. Uma aplicação proxy qualquer trabalha como um programa intermediário, é como ver TV com um vidro na frente. Só que no caso do Squid ele é mais do que um programa que simplesmente repassa informação.

As principais aplicações do Squid são web caching, acelerador de conteúdo e distribuidor de conteúdo. Esses últimos trabalham de forma muito similar.

O Web Caching, o mais comum ou pelo menos o mais conhecido por administradores de redes. Ele permite que com um recurso de Hardware barato seja possível poupar banda de internet e ao mesmo tempo acelerar a navegação interna de uma rede. O método de cache consiste em armazenar localmente todo o conteúdo requisitado de forma que numa segunda requisição ele não precise mais buscar em outro servidor (internet). Este método também permite ajustar a quantidade de memória RAM que será disponibilizada para arquivos populares aumentando ainda mais a velocidade. Junto a esse esquema de aceleração comumente ele trabalha junto autenticação de usuários e o uso de regras de acesso (ACL) que permitem bloquear conteúdos indesejados.

O acelerador de conteúdo ou proxy reverso é, um método antigo e muito utilizado por diversos websites para amenizar a carga no seus servidores. Conteúdo que freqüentemente é acessado vai para o cache do Squid e é servido para o cliente usando apenas uma fração da carga do servidor necessária. É uma aplicação do Squid que é feita de forma rápida e simples com benefícios imediatos.

Por último temos a aplicação do Squid que é muito utilizada para provedores de conteúdo, ele permite que esses provedores comprem computadores baratos e ditribuam eles em locais estratégicos pela internet de forma a dividir a carga de seus servidores principais.

O método abordado nesse curso será o de web caching, com algumas regras de acesso. Voltando ao cache devemos ter em mente que esses arquivos serão armazenados em algum lugar e devem ser acessados do forma mais rápida possível. Além do disco rígido ágil devemos lembrar dos elementos populares que ficarão apenas na memória RAM. A memória RAM é o ponto mais importante do nosso Squid, pois para cada 1GB de dados em discos devemos ter cerca de 10MB a mais em RAM. A CPU é também uma parte importante mas como o Squid não trabalha mais com entrada e saída de dados a CPU não tem muito processamento.

# Linux Network Servers

Para a limpeza de nosso cache existem alguns algoritmos, os mais simples e fáceis de se entender são os Least Recently Used (LRU) e o Least Frequently Used (LFU), onde o LRU remove os documentos existentes por muito tempo, enquanto o LFU remove documentos menos populares. A combinação desses dois métodos forma um sistema de reciclagem de cache bastante eficiente.

Existem duas formas básicas de implementar o web chaching. A primeira consiste em configurar todas as aplicações para acessar diretamente o servidor de proxy, enquanto a segunda redirecionamos todo o tráfego de internet (porta 80 por exemplo) para o servidor de proxy Squid. Então sem perceber a pessoa estará utilizando um web proxy.
Este método é conhecido por proxy transparente ou proxy interceptador.

A segunda solução é a configuração dos programas de clientes para acessar o web proxy, assim cada aplicativo sabe qual o endereço e porta do servidor.

**Configurando um Servidor Proxy Web com Squid**

Verifique se o pacote do Squid está instalado:

Debian:

```
# dpkg -l squid
```

Red Hat:

```
# rpm -qa | grep squid
```

Primeiramente, iremos instalar o Squid, que na sua forma padrão funciona sozinho após a instalação, porém sem regras de acesso tanto no Debian quanto no Red Hat:

```
# aptitude install squid
```

# Linux Network Servers

**Configurações Mínimas**

As configurações do Squid são todas feitas em seu arquivo **squid.conf** localizado no diretório **/etc/squid**. Este arquivo possui cerca de 4350 linhas, mas não se assuste, isto se deve ao fato deste arquivo possuir para cada opção um breve resumo e a sintaxe da opção.

A recomendação do Squid quanto ao arquivo de configurações é que se uma configuração padrão está comentada, deixe assim mesmo, descomentá-las pode causar problemas na hora de execução.

A primeira configuração que iremos fazer feita por questões de segurança, se refere a qual porta ou porta e hostname/IP o Squid ir ouvir conexões. O padrão é ouvir na porta 3128 em todas as redes, vamos mudar para ouvir na nossa rede e nessa mesma porta. Localize a opção **http_port** e troque o **xxx** pelo ip da sua placa de rede:

```
# vi /etc/squid/squid.conf
http_port 192.168.200.xxx:3128
```

Adicione a linha abaixo que informará o nome que será visto nas páginas de erros do Squid. No nosso caso, iremos colocar só o nome Squid4Linux:

```
visible_hostname Squid4Linux
```

As configurações padrão permitem que o Squid rode, porém ele nega todas as requisições, como o nosso objetivo é simplesmente deixar o Squid funcionando, localize a linha que nega (deny) as conexões abaixo e troque pela que permite (allow). Essa linha permite que todos tenham acesso ao Squid independentemente da rede:

```
# http_access deny all
```

Troque por:

```
http_acces allow all
```

Pronto, restart seu squid e já temos um Squid básico mas funcional, é possível navegar utilizando este proxy agora. Para isso configure agora seu navegador web (Iceweasel), no menu **Editar**, depois em **Preferências**, em seguida abra a aba **Avançado**, agora a aba **rede**, e por último o botão **configurações**. Agore configure o proxy manual para todos os protocolos colocando o ip e a porta do Squid que seu colega configurou.

# Linux Network Servers

### Estrutura de cache

Uma das configurações mais importantes com relação ao desempenho do proxy e à otimização do tráfego da rede é a configuração dos caches, onde o Squid guarda as páginas e arquivos já acessados de forma a fornecê-los rapidamente quando solicitados novamente.

Quais são os dois tipos de cache que o Squid trabalha?

a) Cache rápido, feito usando parte da memória RAM do servidor;
b) Cache um pouco mais lento porém maior, feito no HD.

O cache na memória RAM é ideal para que?

É ideal para armazenar arquivos pequenos, como páginas html e imagens, que serão entregues instantaneamente para os clientes.

E o cache usando o disco é ideal para que?

O cache no HD é usado para armazenar arquivos maiores, como downloads, arquivos do Windows Update e pacotes baixados via aptitude.

O cache na memória RAM é quase sempre pequeno. O tamanho da memória RAM é sempre bem menor do que o espaço em disco. Logo, o cache no HD pode ser mais generoso, afinal a idéia é que ele guarde todo tipo de arquivos, principalmente os downloads grandes, que demoram para serem baixados.

Iremos especificar 512MB de cache, com 128 diretórios e 256 subdiretórios:
cache_dir ufs /var/spool/squid 512 128 256

A opção "cache_dir" é composta por quatro valores.

/var/spool/squid indica o diretório no qual o Squid armazena os arquivos do cache;

O 512 indica a quantidade de espaço no HD (em MB) que será usada para o cache.

Você pode aumentar o valor se você tiver mais espaço no disco e quiser que o Squid armazene os arquivos baixados por mais tempo.

Os números 128 e 256 indicam a quantidade de subdiretórios que serão criados dentro do diretório.

O número ideal de diretórios e subdiretórios para um melhor desempenho varia de acordo com o sistema de arquivos usado, mas esta configuração padrão é adequada para a maioria das situações.

Se a linha "cache_dir" for omitida, será usada a configuração default para a opção, que é usar o diretório "/var/spool/squid" e fazer um cache em disco de 100 MB.

# Linux Network Servers

Definindo o cache que será armazenado em memória:
cache_mem 16 MB

Você pode reservar 16 ou 32 ou 64 MB para o cache em um servidor não dedicado, que atende a apenas alguns computadores (como o servidor de uma pequena rede local) e até 1/3 da memória RAM total em um servidor proxy dedicado.

Em um servidor com 2 GB de RAM quanto eu poderia reservar para essa diretiva?

Em torno de 683 MB.

Existem também a opção "maximum_object_size_in_memory".

Ela é que determina o tamanho máximo dos arquivos que serão guardados no cache feito na memória RAM (o resto vai para o cache feito no HD).

O cache na memória é muito mais rápido, mas como a quantidade de RAM é muito limitada, é melhor deixá-la disponível para páginas web, figuras e arquivos pequenos em geral.

Para que o cache na memória armazene arquivos de até 128 KB, por exemplo, adicione a linha:

maximum_object_size_in_memory 128 KB

Se você faz download de arquivos grandes com freqüência e deseja que eles fiquem armazenados no cache, aumente o valor da opção "maximum_object_size"!

Isso é especialmente útil para quem precisa baixar muitos arquivos através do aptitude ou Windows Update em muitos computadores da rede.

Se você quiser que o cache armazene arquivos de até 256 MB, por exemplo, as linhas ficariam:

maximum_object_size 256 MB
minimum_object_size 0 KB

Você pode definir ainda a percentagem de uso do cache que fará o Squid começar a descartar os arquivos mais antigos.

Por padrão, sempre que o cache atingir 95% de uso, serão descartados arquivos antigos até que a percentagem volte para um número abaixo de 90%:

cache_swap_low 90
cache_swap_high 95

Como alteramos a estrutura de nosso cache, precisamos pedir ao squid que reconstrua esta estrutura, para isto:

```
# squid -z
```

# Linux Network Servers

E logo, para que as configurações feitas se apliquem reiniciamos o serviço:

```
# /etc/init.d/squid stop
# /etc/init.d/squid start
```

# Linux Network Servers

**Filtro de Conteúdo com ACL's**

Outro recurso interessante para o uso de um proxy-web é o conceito de filtro de conteúdo, que possibilita a filtragem do conteúdo web dos usuários. Para realizar essa configuração, podemos usar os seguintes filtros:

**src**: Filtro por rede ou endereço IP.
**time**: Filtro por hora e dia da semana.
**urlpath_regex**: Filtro de complemento de uma url.
**url_regex**: Filtro de uma string na url.
**dstdomain**: Filtro de uma url.
**proxy_auth**: Filtro por usuários autenticados.
**arp**: Filtro por MAC address.
**maxconn**: Filtro por conexões.
**proto**: Filtro por protocolos.
**port**: Filtro por porta.

**Usando ACL's e controlando os acessos**

Listas de Controle:

As listas de controle de acesso (Access Control Lists - ACL), são listas que fazem o agrupamento dos objetos que serão negados ou liberados de acordo com as regras determinadas pelo administrador.

A sintaxe de criação é simples. Veja:

```
acl nome_da_acl tipo_da_acl definição
```

Onde o nome_da_acl será um nome para referenciar futuramente a regra, o tipo_da_acl é um dos vários filtros que o Squid possui e, por ultimo a definição que pode estar em um arquivo. Caso esteja em um arquivo ele deve estar com seu caminho completo e entre aspas, e pode conter por exemplo uma lista de ips permitidos onde cada regra deve estar em uma linha.

**Vamos analisar esses exemplos práticos:**

Definindo uma ACL para tratar o acesso no horário de almoço:

```
acl almoco time 12:00-14:00
```

Definindo uma ACL para acesso exclusivo pela sua rede:

```
acl rede200 src 192.168.200.0/255.255.255.0
```

ou

```
acl rede200 src 192.168.200.0/24
```

# Linux Network Servers

**Dando permissões as ACL's**

Depois de criadas as ACL's devemos liberar ou negar acessos definidos nelas, então se temos uma regra como a acl almoco, podemos dizer que negamos ou permitimos acessos nesses horários, vamos ver alguns exemplos:

A sintaxe de controle é bem simples também:

```
# Permite (allow)
http_access allow nome_da_acl
```

```
# Nega (deny)
http_access deny nome_da_acl
```

Vamos criar o exemplo abaixo:

```
acl maquina1 src 192.168.200.1
acl maquina2 src 192.168.200.2
acl maquina3 src 192.168.200.3

acl parte_manha time 07:00-11:30
acl almoco time 12:00-14:00

http_access allow maquina1 parte_manha
http_access allow maquina2 parte_manha
http_access allow maquina2 almoco
http_access allow maquina3 almoco
```

Nesse exemplo, temos três ips de máquinas sendo definidas (linhas 1, 2 e 3) e duas ACL's de tempo chamadas parte_manha e almoco (Linhas 5 e 6).

Linha 8 - Permitindo a máquina 1 fazer acesso no período da manhã.

Linha 9 - Permitindo a máquina 2 fazer acesso no período da manhã.

Linha 10 - Permitindo a máquina 2 fazer acesso no período da manhã e no horário de almoço.

Linha 11 - Permitindo a máquina 3 fazer acessono horário de almoço.

Vejamos outros exemplos mais complicados:

Supomos que precisamos bloquear uma palavra para tentarmos filtrar algum tipo de conteúdo, por exemplo a palavra **boy**. Observe que se for feito o filtro na URL de um site fatalmente sites como www.playboy.com.br e www.sevenboys.com.br serão bloqueados.

O que faríamos numa situação dessas? Primeiramente deve ser criada uma ACL para controlar essa palavra. Mas observe que a palavra sevenboys foi bloqueada.

Se colocarmos somente uma lista negra teremos problemas, porque fatalmente isso poderá acontecer com mais palavras. Então fica impraticável trabalhar somente com uma blacklist sem ter uma whitelist.

# Linux Network Servers

Podemos criar as ACL's dessa forma:

```
acl blacklist url_regex -i "/etc/squid/blacklist"
acl whitelist url_regex -i "/etc/squid/whitelist"
```

**OBS**: Vale lembrar que as listas estando entre aspas devem representar o endereço de um arquivo no disco. A url_regex trata uma string dentro de uma url completa. O parâmetro -i diz que qualquer palavra dentro do arquivo sofrerá tratamento de case insensitive.

Com os comandos a seguir temos que dar permissões as ACL's, onde a primeira linha permite os endereços na lista branca, e a segunda nega os endereços da lista negra:

```
http_access allow whitelist
http_access deny blacklist
```

# Linux Network Servers

**Autenticação no Squid**

Quando o servidor de Proxy web Squid é configurado, podemos utilizar um recurso muito interessante que é a autenticação, ela torna o tratamento de logs mais prático. Vale lembrar que essa é uma das configurações que não funcionam em um proxy transparente. Podemos especificar um autenticador externo de acesso ao cache do Proxy, recurso muito útil pois podemos utilizar um servidor LDap por exemplo. Devemos informar ao Squid quem é o programa externo responsável pela tarefa. Iremos usar o formato ncsa_auth, por fazer parte da base do próprio Squid.

Descomente a linha abaixo e deixe-a igual a seguinte:

```
# auth_param basic program /uncomment and complete this line\
```

Por:

```
auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/passwd
```

É possível alterar o número de processos filhos (padrão é 5). O valor é baixo para dificultar o uso de programas de força bruta que tentam por meio de ataques com dicionários descobrir a senha do usuário:

```
auth_param basic children 5
```

O tempo de expiração da senha também pode ser alterado, o padrão é uma hora:

```
authenticate_ttl 1 hour
```

Definindo um ACL para ativar o recurso de autenticação:

```
acl password proxy_auth REQUIRED
```

Devemos dar permissão a essa ACL antes de qualquer outra:

```
http_access allow password
```

Devemos ter uma ACL final estratégica negando, qualquer possibilidade que não se enquadre nas ACL's anteriores. Essa diretiva já vem por padrão:

```
http_access deny all
```

# Linux Network Servers

Cadastre um usuário para testar a autenticação. Esse utilitário faz parte do pacote apache2-utils, caso não esteja instalado, instale-o:

```
# htpasswd -c /etc/squid/passwd usuario
```

Descomentado a linha abaixo, pode-se personalizar a mensagem de autenticação do Squid:

```
auth_param basic realm Squid proxy-caching web server
```

Após toda a configuração, podemos reiniciar o serviço do Squid:

```
# /etc/init.d/squid restart
```

# Linux Network Servers

**Estrutura de logs**

Atendendo à norma de segurança NBR ISO/IEC 17799, recomenda-se, no tópico 10.10.1, que sejam gerados registros de atividades dos usuários e que os mesmos sejam mantidos por tempo definido dentro da corporação, possibilitando investigações futuras e também a monitoração do controle de acesso.

Esses logs devem manter:

- Identificação dos usuários: Squid por padrão registra o IP, mas com a autenticação teremos o login do usuário;
- Data e horários de entrada: Squid por padrão mantém registrado;
- Identidade do terminal ou, quando possível, sua localização: Squid não se aplica essa diretriz;
- Registro das tentativas de acessos aceitos e rejeitados: Squid por padrão mantém registrado;
- Registro das tentativas de acesso a outros recursos e dados aceitos e rejeitados: Squid por padrão mantém registrado;

Embora seja padrão a estrutura de arquivos de logs sugerida a seguir, iremos deixar definido no squid.conf sua estrutura de logs.

Arquivo de registros de conexões HTTP de clientes no Proxy:
access_log /var/log/squid/access.log

Arquivo de informações como hora e data em que o cache foi inicializado e o que foi armazenado:
cache_log /var/log/squid/cache.log

Registro dos objetos que foram armazenados (páginas, figuras, etc):
cache_store_log /var/log/squid/store.log

**Relatórios do Squid pelo SARG**

Desenvolvido pelo brasileiro Pedro Orso, sua função é retornar um relatório HTML consolidado com as atividades dos usuários Squid.

Instalar o sarg:

```
# aptitude install sarg
```

Altere algumas configurações dentro do arquivo sarg.conf que é instalado dentro do diretório /etc/squid:

```
# vi /etc/squid/sarg.conf
language Portuguese
title "Relatório de Acessos SARG"
```

```
resolve_ip no
```

Depois de configurar o sarg.conf, basta gerar os relatórios com o comando:

```
# sarg
```

Verifique se o diretorio squid-reports foi criado em /var/www e, caso tenha um apache configurado acesse em http://localhost/squid-reports em seu navegador.

**Squid como gerenciador de banda**

Podemos usar o Squid para gerenciar banda também! O Squid pode definir o quanto cada usuário pode usar e com isso manter uma parte do link livre para os demais.

Para isso ele utiliza um recurso chamado "delay pools". Imagine, por exemplo, que você tem um link de 2 megabit para uma rede com 40 usuários. Se cada um puder baixar o que quiser, é bem provável que a rede ficará saturada em determinados horários implicando em uma navegação lenta para o resto dos usuários.

Você pode evitar isso limitando a banda que cada usuário pode usar e a banda total, que todos os usuários somados poderão usar simultaneamente. É recomendável, neste caso, que o servidor proxy (que combina todos os acessos via http) consuma um pouco menos que o total de banda disponível, de forma a sempre deixar um pouco reservado para outros protocolos.

Um link de 2 megabit (2 x 1024 kbits) corresponde a 262144 bytes por segundo.

Como que fiz essa conta?

(2 x 1024 x 1024) / 8 = 262144 bytes por segundo

1 byte = 8 bits

Nas regras do Squid, sempre usamos bytes, por isso lembre-se de fazer a conversão, dividindo o valor em kbits por 8 e multiplicando por 1024 para ter o valor em bytes.

Podemos, por exemplo, limitar a banda total usada pelo Squid a 229376 bytes por segundo, deixando 256 kbits do link livres para outros protocolos e limitar cada usuário a no máximo 32768 bytes por segundo, que correspondem a 256 kbits.

Dica importante: Você deve acompanhar o uso do link e ir ajustando o valor conforme a utilização.

Neste caso, no arquivo de configuração ficaria:

```
acl MyNetwork src 192.168.200.0/24
delay_pools 1
delay_class 1 2
```

# Linux Network Servers

```
delay_parameters 1 229376/229376 32768/32768
delay_access 1 allow MyNetwork
http_access allow localhost
http_access allow MyNetwork
http_access deny all
```

A acl "MyNetwork" está agora condicionada a três novas regras, que aplicam o uso do limite de banda.

O acesso continua sendo permitido, mas agora dentro das condições especificadas na linha "delay_parameters 1 229376/229376 32768/32768", onde vão (respectivamente) os valores com a banda total disponível para o Squid e a banda disponível para cada usuário.

Veja que nessa regra limitamos a banda apenas para a acl "MyNetwork" e não para o "localhost". Isso significa que você continua conseguindo fazer downloads na velocidade máxima permitida pelo link ao acessar a partir do próprio servidor; a regra se aplica apenas às máquinas clientes.

É possível também criar regras de exceção para endereços IP específicos, que poderão fazer downloads sem passar pelo filtro.

Nesse caso, podemos criar uma acl contendo o endereço IP da máquina cliente que deve ter o acesso liberado usando o parâmetro "src" e a colocamos antes da regra que limita a velocidade, como em:

```
acl patrao src 192.168.200.2
http_access allow patrao

acl MyNetwork src 192.168.200.0/24
delay_pools 1
delay_class 1 2
delay_parameters 1 229376/229376 32768/32768
delay_access 1 allow MyNetwork
http_access allow localhost
http_access allow MyNetwork
http_access deny all
```

Esta regra de exceção pode der usada também em conjunto com as demais regras de restrição de acesso que vimos anteriormente. Basta que a acl com o IP liberado seja colocada antes das acls com as restrições de acesso!

# Linux Network Servers

```
acl patrao src 192.168.200.2
http_access allow patrao

acl MyNetwork src 192.168.200.0/24
acl International dstdomain .com
acl Amazon dstdomain .amazon.com

http_access allow MyNetwork Amazon
http_access deny MyNetwork International
http_access allow localhost
http_access allow MyNetwork
http_access deny all
```

Continuando, sempre que fizer alterações na configuração, você pode aplicar as mudanças usando o comando:

```
# /etc/init.d/squid reload
```

O parâmetro "reload" permite que o Squid continue respondendo aos clientes e aplique a nova configuração apenas às novas requisições.