

Squid: Master this Proxy Server

Squid is an excellent and mature open source Web caching proxy package, but it requires plenty of tuning to achieve the kind of performance seen in commercial proxies. Here are some useful ideas for tuning a Web caching system, and effectively using the server in a corporate environment.

While implementing the Squid proxy server in a real-life scenario, we need to have a server-end machine with a good amount of RAM and a high-end hard disk. Typically, an SCSI hard drive (if possible with Raid) is preferred. This is because Squid can be easily crippled by disks that are not performing up to the specifications. Web caching is an 'I/O-bound' application, meaning that it can only go as fast as it gets the data onto and off the storage media. So the first thing necessary is to ensure you have the right hardware for caching. Squid in its default configuration, as found in RPMs or in a standard compile, does not need a very fast CPU to support one or two disks. However, a large performance boost can be achieved by using a threaded version of Squid. The threads will use a lot of CPU power. So, if you are going

to use threads as documented below, consider using a rather fast processor.

While choosing the flavour of Linux, the kernel version plays an important role. As our implementation of the Squid will be threaded in nature, the minimum kernel version should be 2.4x. We can do with 2.2.x also, but we need to patch the kernel a lot and the procedure becomes complicated.

SIMPLE CONFIGURATION OF SQUID PROXY SERVER

Get Squid from your distribution CD or search it in <http://www.rpmfind.net/>. I am using Red Hat Linux 7.1 here.

```
Install squid by using
[root@linux /root]# rpm -ivh squid-2.3.STABLE4-10.i386.rpm
```

The configuration files for squid are in the /etc/squid directory.

To configure Squid, edit the squid.conf configuration file. I will go here with a basic configuration. I have removed the comments included in the configuration file.

Backup your original Squid configuration file and make a new one if you are new to it.

Add these entries on this new squid.conf file for a basic setup.

```
maximum_object_size 200 KB

#This creates 100 MB disk space with 16
#first level sub-directories and 256 second
#level subdirectories.
```

```
cache_dir ufs /var/spool/squid 100 16 256
cache_access_log /var/log/squid/access.log
cache_log /var/log/squid/cache.log
```



```
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl lan src 192.168.0.0/255.255.255.0 #specify a name for your
network
acl SSL_ports port 443 563
acl Safe_ports port 80 21 443 563 70 210 1025-65535
acl Safe_ports port 280 # http-mgmt
acl Safe_ports port 488 # gss-http
acl Safe_ports port 591 # filemaker
acl Safe_ports port 777 # multiling http
acl CONNECT method CONNECT

http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow lan
http_access deny all
icp_access allow all
miss_access allow all
cache_mgr root@yourdomain.com
visible_hostname you.yourdomain.com
unique_hostname you.yourdomain.com

#We will run squid with accelerator on
httpd_accel_host virtual
httpd_accel_port 80
httpd_accel_with_proxy on
httpd_accel_uses_host_header on

http_port 3128
```

Now you can start the squid proxy server

```
[root@mail /root]# /etc/rc.d/init.d/squid start
```

Squid, when started for the first time, will create the cache directories by itself and start the proxy server on port 3128, which can only be accessed by the local network segment 192.168.0.0/255.255.255.0 without any restrictions.

Now open your browser on the internal network and modify the LAN connection settings and set it to the server's IP address and the port as 3128. Try browsing.

This is the simple way of making Squid work. Let us now try and configure Squid to do some complicated tasks, and tune it to have the desired or high performance.

TUNING SQUID PROXY FOR HIGH PERFORMANCE

For a high performance, as discussed earlier, the following are needed:

1. Server platform with good processing speed, good amount of memory, and a fast hard drive
2. Latest kernel or minimum 2.4x
3. ReiserFS file system
4. Squid version 2.2.STABLE5

Simply switching your cache directories to ReiserFS will increase Squid performance by about 20 per cent. ReiserFS is significantly faster than ext2 when dealing with thousands of small files. Since small files are pretty much all that a Web cache deals with, we're going to use ReiserFS on our cache

partitions. You can find ReiserFS at < <http://devlinux.com/projects/reiserfs/> > . An added benefit of a journalled FS is that in the event of an abnormal shutdown, your system will recover faster and without as many lost files as ext2.

We've chosen to use a somewhat older version of Squid, version 2.2.STABLE5, plus the latest patch snapshot from Henrik Nordstrom. There are two reasons for this. First, squid-2.3.STABLE2 is not as stable as 2.2.STABLE5+ Henrik's patches. Second, squid-2.3.STABLE2 is not as fast as squid-2.2.STABLE5+ Henrik's patches.

Download the Squid version 2.2.STABLE5 from <http://www.squid-cache.org/Versions/v2/2.2/squid-2.2.STABLE5-src.tar.gz> or from <http://www.linuxforu.com/editorial/may03.htm>.

So you'll need Henrik's patch snapshot. This can be found at this address:

<http://squid.sourceforge.net/hno/> or from <http://www.linuxforu.com/editorial/may03.htm>. The file name is squid-2.2.STABLE5-hno.20000202.snapshot.

Untar the source package

```
[root@mail misc]# tar xvfz squid-2.2.STABLE5-src.tar.gz
[root@mail misc]# zcat squid-2.2.STABLE5-hno.20000202.snapshot
> hnopatch
[root@mail misc]# cd squid-2.2.STABLE5
[root@mail squid-2.2.STABLE5]# patch -p1 < ../hnopatch
```

One more thing you need to do before compiling Squid is raise the number of file descriptors on your system. First you need to edit this line in the `/usr/include/bits/types.h` file:

```
#define __FD_SETSIZE 8192
```

It is 1024 by default. 1024 is not quite enough for even a single disk cache after all of these optimisations. Next, you need to raise the limit in the shell so the configure script will see the added descriptors:

```
ulimit -HSn 8192
```

This raises the hard and soft limits to 8192. Now you are ready to compile Squid in a high-load environment. Squid wants to be compiled with the `async-io` configure option enabled. Further, if you want to increase the number of threads Squid is given, the default is 16. This number depends heavily on the speed of your processor and the number of disks.

```
[root@mail squid-2.2.STABLE5]# CFLAGS="-DNUMTHREADS=30" ./
configure --prefix=/usr/local/squid \
> -enable-async-io -enable-async-io -enable-delay-pools
[root@mail squid-2.2.STABLE5]# make
[root@mail squid-2.2.STABLE5]# make install
```

Once you have the complete build, you need to define the cache structure or the file system to be used by the cache. Earlier, we have mentioned ReiserFS is to be incorporated so

as to have a faster cache. In our installation we have dedicated a partition of 10 GB for the cache. Fortunately, under kernel 2.4x, we have the ReiserFS module in-built. It can be checked out at

```
[root@mail /root]# modprobe -l | grep reiserfs
/lib/modules/2.4.2-2smp/kernel/fs/reiserfs/reiserfs.o
```

Now you can convert your Squid directories to ReiserFS in this way (X is your planned cache partition):

```
[root@mail /root]# mkreiserfs/dev/hdaX
```

Add the following line to your `/etc/fstab`:

```
/dev/hdaX /cache reiserfs notail,noatime 0 0
```

You can then mount your new cache directory:

```
[root@mail /root]# mount /cache
```

The no-tail option above tells ReiserFS not to pack tails, which is a way for the FS to save file space. Since we need speed much more than a few megabytes of saved space, we'll disable it. And the no-time option should be used regardless of what file system you use for a Web cache. It disables access time updates on files, saving one write for every read.

Now, after all of these things, you will have a Squid that is capable of handling a much higher load than the default compile. But, there are a few more things we need to do before we have a Web cache running full tilt. Thankfully, there will be no more patching or compiling. The first thing to do is configure the `squid.conf` file located in `/usr/local/squid/etc`

The important configuration options for performance are given below:

cache_mem32: This is one of the top two options as far as performance impact goes. The default value here is 8 MB, which is a safe option for just about any system size. But since your box is not lacking in memory, it can safely be raised to 32 MB, and even 48 MB, if you have stripped all other services off of your server. If your box is lacking in memory, go back to the hardware section and re-read the part about memory... you need a portion of memory for a fast Web cache.

You should be aware that `cache_mem` does not limit the process size of Squid. This sets how much memory Squid is allowed to set aside for 'hot objects', which are the most recently used objects. Having said all that, keep in mind that the buffer cache in Linux is also quite good, so the gains you'll see by raising this are very small, but still measurable.

cache_dir: This is where you set the directories you will be using. You should have already `mkreiserfs'd` your cache

directory partitions, so you'll have an easy time deciding the values here. First, you will want to use about 60 per cent or less of each cache directory for the Web cache. If you use any more than that you will begin to see a slight degradation in performance. Remember that cache size is not as important as cache speed, since for maximum effectiveness your cache needs only to store about a week's worth of traffic. You'll also need to define the number of directories and sub-directories. The formula for deciding that is this:

```
x=Size of cache dir in KB (i.e. 10GB=~10,000,000KB) y=Average
object size
(just use 13KB z=Number of directories per first level
directory
(((x / y) / 256) / 256) * 2 = # of directories
```

As an example, I use 10 GB of cache size

```
10,000,000 / 13 = 769230.7 / 256 = 3004.8 / 256 =11 * 2 = 22
```

So my `cache_dir` line would look like this:

```
cache_dir /cache 10000 22 256
```

Those are really the two important ones. You may wish to turn off the store log since there isn't much one can do with it anyway:

```
cache_store_log none
```

If, after all this, you still find your Squid being overloaded at times, try setting the `max_open_disk_fds` to some low number. For a single disk cache, 900 is a good choice. For a dual disk cache, try 1400. To get the perfect number for your cache you'll need to keep an eye on the info page from the Cache Manager during a particularly heavy load period. Watch the file descriptor usage, and then set this number about 10 per cent below it. This can help your cache ride out the heaviest loads without getting too bogged down.

Also, the following may be set to improve performance marginally:

```
half_closed_clients off maximum_object_size 1024 KB
```

Ok, that's all for Squid configuration. Now there's one last thing to do to complete the picture. You need to edit your Squid startup script to include the following two lines:

```
ulimit -HSn 8192 echo 1024 32768 > /proc/sys/net/ipv4/
ip_local_port_range
```

This will raise the file descriptor limit for the Squid process, and will open up a ton of IP ports for use by Squid. It won't need quite this many, but better safe than sorry.

That's it! You now have the fastest Squid proxy server on your block.

TIPS AND TRICKS

1. Specifying which port to listen on:

Edit the file: `/etc/squid/squid.conf`

The control of external access to the local LAN should be managed by the Firewall.

To be safer, set the restriction given below on where the Squid server is listening.

```
# http_port 3128
http_port internal_nic1:3128
http_port internal_nic2:3128
```

Normally, Squid starts up and listens to 3128 on all network devices. The above just ensures that it is listening on port 3128 only for the internal network. Our firewall can further block port 3128 requests from coming through from the outside (but our ACLs should be handling any further problems.)

2. Timing restrictions:

This sample discusses how the LAN network can be denied use of the proxy server during non-office hours.

```
# After Hours Settings
acl TIMEafterhoursMORN time MTWHF 00:00-08:00
acl TIMEafterhoursAPT time MTWHF 16:30-24:00

http_access deny lan TIMEafterhoursMORN TIMEafterhoursAPT
```

3. Blocking sites or the pre-defined URL:

```
acl block_advertisers url_regex -i "/etc/squid/
block_advertisers.txt"
acl block_entertainment url_regex -i "/etc/squid/
block_entertainment.txt"
acl block_webmail url_regex -i "/etc/squid/
block_webmail.txt"
acl block_porn url_regex -i "/etc/squid/
block_porn.txt"
```

In the specified file, mention the URL of the relevant topic. It should be one line per URL. Then define the access methods

```
http_access deny block_advertisers
http_access deny block_entertainment
http_access deny block_porn
http_access deny block_webmail
```

TRANSPARENT PROXYING

In ‘ordinary’ proxying, the client specifies the hostname and port number of a proxy in his Web browsing software. The browser then makes requests to the proxy, and the proxy forwards them to the origin servers. This is all fine and good, but sometimes one of several situations can arise. Either

- You want to force clients on your network to use the proxy, whether they want to or not.

- You want clients to use a proxy, but don’t want them to know they’re being proxied.
- You want clients to be proxied, but don’t want to go to all the work of updating the settings in hundreds or thousands of Web browsers.

This is where transparent proxying comes in. A Web request can be intercepted by the proxy, transparently. That is, as far as the client software knows, it is talking to the original server itself, when it is really talking to the proxy server. (Note that the transparency only applies to the client.)

Once you have kernel 2.4x on Red Hat 7.x or 8.x up and running, you may need to enable IP forwarding. IP forwarding allows your computer to act as a router. To check, do `cat /proc/sys/net/ipv4/ip_forward`. If you see ‘1’, you’re doing well. Otherwise, do `echo ‘1’ > /proc/sys/net/ipv4/ip_forward`. You will then want to add that command to your appropriate boot-up scripts like `/etc/rc.d/rc.local`.

In case of transparent proxy, rebuild the Squid with `–enable-linux-netfilter` additionally. Now, we need to edit the default `squid.conf` file (installed to `/usr/local/squid/etc/squid.conf`). Find the following directives, uncomment them, and change them to the appropriate values as below:

- `httpd_accel_host virtual`
- `httpd_accel_port 80`
- `httpd_accel_with_proxy on`
- `httpd_accel_uses_host_header on`

The rest of the directive can be according to the policies of the corporate. To activate the transparent nature of the proxy, you need to know two things — the interface that the to-be-proxied requests are coming in on (I’ll use `eth0` as an example) and the port Squid is running on (I’ll use the default of 3128 as an example).

Now, the magic words for transparent proxying:

```
• iptables -t nat -A PREROUTING -i eth0 -p tcp -dport 80 -j
REDIRECT -to-port 3128
```

Please make sure Iptables is activated (as in case of most Red Hat distributions, `ipchains` is activated) by

```
[root@mail /root]#modprobe -r ipchains
[root@mail /root]# modprobe ip_tables
```

You will want to add the above commands to your appropriate boot-up script under `/etc/rc.d/rc.local`. The main drawback of using transparent proxy is that there is no support for authentication and HTTPS proxying.

AUTHENTICATION AND MANAGEMENT OF USERS

The Squid source code comes with a few authentication processes. These include:

- *LDAP*: Uses the Lightweight Directory Access Protocol
- *NCSA*: Uses an NCSA-style user name and password

file

- *MSNT*: Uses a Windows NT authentication domain
- *PAM*: Uses the Linux Pluggable Authentication Modules scheme
- *SMB*: Uses a SMB server like Windows NT or Samba
- *getpwnam*: Uses the old-fashioned Unix password file

In order to authenticate users, you need to compile and install one of the supplied authentication modules. In our implementation, we will be using NCSA-based authentication modules to demonstrate. The basic procedure is to first compile the NCSA source module and make an executable file. Then integrate this feature in the squid.conf file to have the client first authenticate on to the proxy.

```
Compile NCSA by
[root@mail squid-2.2.STABLE5]# cd auth_modules/NCSA/
[root@mail NCSA]# make; make install
```

This will generate *nsc_auth* in */usr/local/squid/bin* directory.

Now you have to tell Squid to check for user authentication. You need to modify */usr/local/squid/etc/squid.conf* file...

```
# authenticate_program example
authenticate_program /usr/local/squid/bin/nsc_auth /usr/
local/squid/etc/passwd

authenticate_ttl 900
authenticate_ip_ttl 60
acl name proxy_auth REQUIRED
http_access allow lan name
http_access deny all
```

Where *lan* is the acl defined earlier. Now create a file */usr/local/squid/etc/passwd* to store user names and the passwords. *htpasswd* is the utility that comes along with Apache and will do the required action for us.

```
[root@mail /root]# htpasswd -c /usr/local/squid/etc/passwd
biswajit Enter password:
```

This will create a file */usr/local/squid/etc/passwd* with user name Biswajit and an encrypted password. Create other

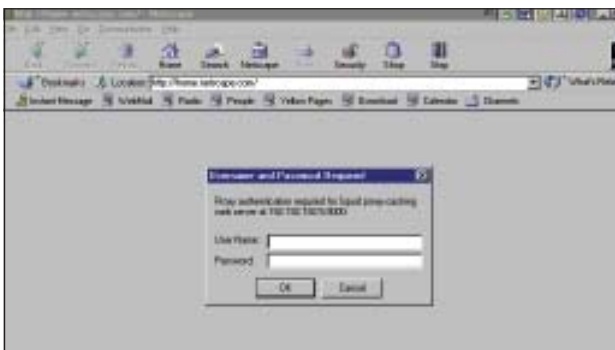


Figure 1

users with the same command without *-c* option; the file will look like

```
biswajit:P0z0V41ZY4xZI
tetra:hszXF.a02Bomc

[root@mail /root]# chmod 755 /usr/local/squid/etc/passwd
```

Restart your Squid proxy server with */usr/local/squid-2.2stable5/bin/squid -k reconfigure* and point your browser to any external site. You will be asked to authenticate as shown in Figure 1.

Once the correct user name and password is given, the user will be allowed to pass through and to browse.

If the list of users is long enough to manage, we can have some sort of Web interface to control these users in terms of creation, deactivation, modification, etc, by the system administrators. In our next section we are going to discuss how to implement and manage users of proxy by a Web interface. Please note this interface is user-centric and not for tuning any of the Squid proxy parameters. Administrators can use *webmin* (www.webmin.org) as a tool to define parameters of proxy. But the more you are familiar with the text environment, the better it is for the administrator. I personally believe the easy interfaces are meant for repetitive and manual work as the user management, but tuning of Squid proxy is more or less a one-time job, which needs clear understanding. We will use an *Admuser* package to do so. With *Admuser*, you can handle your Squid or Web users and passwords using your browser. *Admuser* allows you to create, change, remove, disable and enable users online. You can download the file (*admuser-2.1.tar.gz*) from <http://web.onda.com.br/orso/> or <http://www.linuxforu.com/editorial/may03.htm>.

```
[root@mail misc]# tar xvfz admuser-2.1.tar.gz
[root@mail misc]# cd admuser-2.1
[root@mail admuser-2.1]# ./configure --enable-cgidir=/var/www/
cgi-bin
```

Our *httpd*'s *cgi-bin* directory is */var/www/cgi-bin*

Create a file */usr/local/etc/admuser/pwd_files* with entry



Figure 2

```
/usr/local/squid/etc/passwd:Squid Password file

[root@mail admuser-2.1]# htpasswd -c /usr/local/etc/admuser/
pwauth admuser
New password:
Re-type new password:
Adding password for user admuser
```

Remember, `/usr/local/squid/etc/passwd` file must be owned by `httpd` user, or `chmod 777`(dangerous)
 Make sure your `http` service is up, point your browser to `http://server_ip/cgi-bin /admuser.cgi` and give user name as `Admuser` and a respective password. Thrown on a screen (Figure 2) where the admin can create, modify and disable users.

In a certain scenario, the administrator would like to provide the password changing facility to the user itself. In our next implementation, we will like to do so. This is done by implementing a Web-based interface given to the user, so that the user can change his own password as per his requirement.

We choose to implement `CHPASSWD`. This utility allows your users to change his/her Squid or Web password using the browser. This was adapted from `htpasswd` for use with Squid Cache Proxy.

Download `chpasswd` utility (`chpasswd-2.2.tar.gz`) from <http://web.onda.com.br/orso/> or <http://www.linuxforu.com/editorial/may03.htm>.

```
bash# tar xvfz chpasswd-2.2.tar.gz
bash# cd chpasswd-2.2
bash# ./configure --enable-cgidir=/var/www/cgi-bin
bash# make; make install
```

Now the admins are ready to give the URL `http://server_ip/cgi-bin/chpasswd.cgi` and users can now change their passwords as shown in Figure 3.

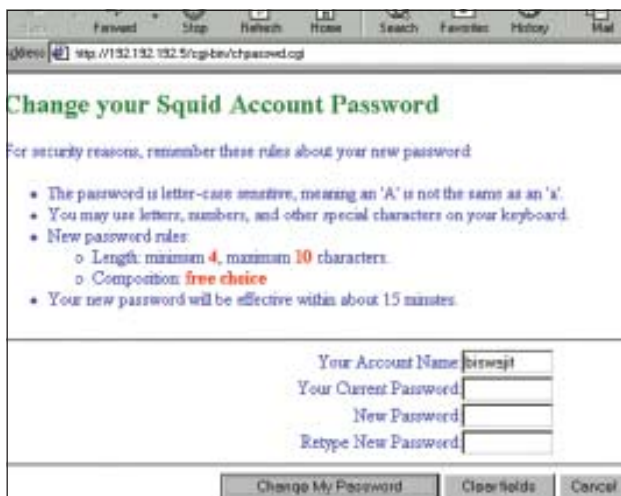


Figure 3

CONTROL PANEL FOR USER MANAGEMENT WITH DATABASE INTEGRATION

Let us try to see how we can integrate a database like MySQL to authenticate users and manage them with the following features:

1. Authentication for Squid users via MySQL database.
2. Traffic limitations, by setting maximum daily, weekly and monthly downloads for each user.
3. Monthly-generated detailed statistics, with cache and real online time used.
4. Reports of current month statistics for each user in any time.
5. Users properties management, including real name, phone, e-mail, and password.

We will use `Squid2MySQL` to do so which requires pre-installed Linux (any version), `squid` (version 2.x and higher), Apache with PHP and MySQL support and PERL with DBI module, which shipped with `Squid2MySQL`.

Download `squid2mysql-1.0.0.tar.gz` from <http://evc.fromru.com/squid2mysql/download.html> or <http://www.linuxforu.com/editorial/may03.htm>

```
[root@mail misc]# tar xvfz squid2mysql-1.0.0.tar.gz
[root@mail misc]# cd squid2mysql-1.0.0
Make sure Mysql server is started. Modify the squid2mysql.sql
Replace INSERT INTO usernames
VALUES('squidroot','','','',''); to
INSERT INTO usernames VALUES('squidroot','','','','');
And comment all lines containing "DROP".
[root@mail squid2mysql-1.0.0]# ./install_squid2mysql
```



Figure 4

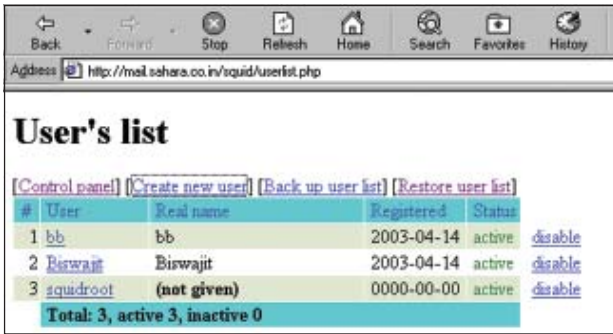


Figure 5

This will run and make the relevant links and files. Remove Squid log file and create named pipe file with name of Squid log file (“mknod /var/log/squid/access.log p”). Put squid2mysql perl script and sqauth sh script into Squid directory with the ownership of Squid. Configure squid.conf file as...

```
authenticate_program /usr/local/squid/bin/sqauth
authenticate_children 5
authenticate_ttl 900
authenticate_ip_ttl 60
```

Connect to the MySQL server

```
#mysql -p
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 331 to server version: 3.23.36

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql> connect squidlog

Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -A

Connection id: 332
Current database: squidlog

mysql> GRANT Select, Insert, Update, Delete, Index, Alter,
Create, Drop,
-> References ON *.* TO 'squidroot'@'localhost' IDENTIFIED
BY 'sqroot'
-> ;
Query OK, 0 rows affected (0.01 sec)
```

Modify /var/www/html/squid/include.php to have English as language from “tconst.koi8r.php” to “tconst.eng.php”

Point your browser to http://serverip/squid/ and authenticate with user name as ‘squidroot’ and password ‘sqroot’ as defined earlier. Figures 4, 5 and 6 give the screen shot of the interface.

SQUID GRAPHS AND PERFORMANCE MONITORING

For larger sites, it is mandatory to have graphical analytic tools that identify the kind of traffic getting through, the



Figure 6

performance of the Squid server, and so on. There are many tools that are available to analyse the traffic in graphical as well as detailed reports.

For our analysis we will implement a Squid Graph. This is a powerful log file analysis tool for native Squid version 2 log files. It generates HTML reports and graphs of accesses, transfers and transfer durations, somewhat similar to MRTG.

Download squid graph from http://squid-graph.securlogic.com/files/stable/squid-graph-3.0.tar or http://www.linuxforu.com/editorial/may03.htm.

Extract the Squid Graph tarball file after you have downloaded it. Those with Red Hat Linux (or other similar

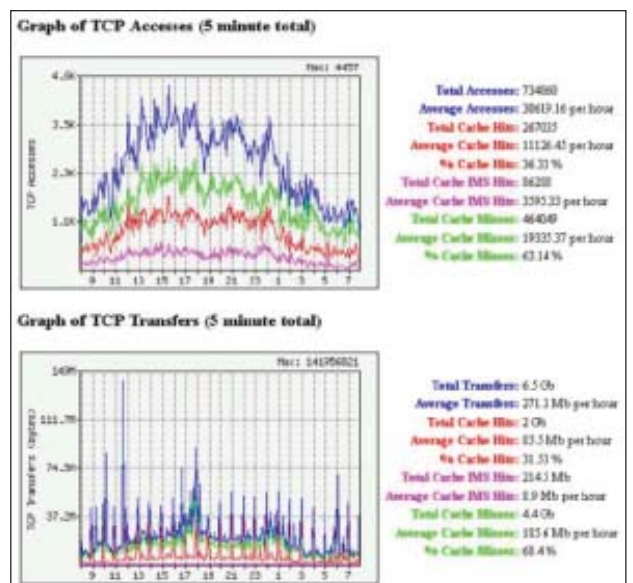


Figure 7

distributions) can do this:

As of version 3.0, Squid Graph requires the GD perl module. You can download it from <http://stein.cshl.org/WWW/software/GD/> or you can use the included GD-1.3.3.tar.gz file in the extras/ directory. Follow the instructions in the GD perl module to get it installed correctly before you proceed.

```
[root@mail misc]# tar xvf squid-graph-3.0.tar
[root@mail misc]# cd squid-graph-3.0
[root@mail misc]# mv ../squid-graph-3.0 /usr/local/squid-graph
[root@mail squid-graph-3.0]# cd /usr/local/squid-graph/bin
[root@mail squid-graph-3.0]# mkdir /var/www/html/reports/
[root@mail squid-graph-3.0]# ./squid-graph --output-dir=/var/www/html/reports \
< /usr/local/squid/logs/access.log
```

Point your browser to <http://serverip/reports/index.html>. It will display the comprehensive graph shown in Figure 7.

To implement the graph facility on a regular basis (realtime) create a cron job as:

```
#crontab -e
*/5 * * * * /usr/local/squid-graph/bin/squid-graph --output-dir=/var/www/html/reports \
< /usr/local/squid/logs/access.log
```

You can manage log files by editing the /etc/cron.daily/squid.rotate file and add the file lines:

```
if [ -x /usr/local/squid/bin/squid -a -f /usr/local/squid/logs/squid.pid ]; then
    /usr/local/bin/squid -k rotate
fi
```

CONTENT MANAGEMENT THROUGH A PROXY SERVER

To have maximum productivity, Web content needs to be analysed and passed to the users. URL filtering can sort out the issues to a certain extent, but content filtering is the right solution. The Squid-guard or DansGuardian does the job for us. I found DansGuardian to be more effective and faster than Squid-guard.



Figure 8

DansGuardian is a Web content filtering proxy that uses Squid to do all the fetching. It filters using multiple methods including, but not limited to, phrase matching, file extension matching, MIME type matching, PICS filtering, and URL/domain blocking. It has the ability to switch off filtering by certain criteria, including username, domain name, source IP, etc. The configurable logging produces a log in an easy-to-read format. It has the option to only log text-based pages, thus significantly reducing redundant information (such as every image on a page).

Download the source package *DansGuardian-2.2.5-1.source.tar.gz* from <http://www.linuxforu.com/editorial/may03.htm>

```
[root@mail]# tar xvfz DansGuardian-2.2.5-1.source.tar.gz
[root@mail]# cd DansGuardian-2.2.5
[root@mail DansGuardian-2.2.5]# ./configure --cgidir=/var/www/cgi-bin
[root@mail DansGuardian-2.2.5]# make; make install
```

```
Modify the /etc/dansguardian/dansguardianconf to suit your configurations
accessdeniedaddress = 'http://YOURSERVER.YOURDOMAIN/cgi-bin/dansguardian.pl'
```

Change *YOURSERVER.YOURDOMAIN* to your server ip or the name

Also note that our Squid, by default, runs on 3128, which needs to be pointed in the dansguardian.conf and dansguardian run on 8080. Therefore, the port 3128 needs to be blocked by a firewall rule so that users cannot bypass and go to 3128. Also, note now the user's browser must point to the proxy port 8080 instead of 3128.

```
#!/sbin/iptables -t filter -A INPUT -p tcp -d 0/0 -i lo - destination-port 3128 -j ACCEPT
#!/sbin/iptables -t filter -A INPUT -p tcp -d 0/0 - destination-port 3128 -j DROP
#!/sbin/iptables -t filter -A INPUT -p tcp -d 0/0 -destination-port 8080 -j ACCEPT
```

This means the lo interface will accept on 3128 requested by the DansGuardian module and reject any request on port 3128. So the user will not be allowed to access Squid proxy server directly but will be allowed access only through the content filter.

Squid is so flexible in nature that you can practically implement any kind of corporate policies without losing on performance. There is so much to talk about Squid that you actually cannot cover everything in a single go. But I sincerely feel that this article should be able to give an idea of how Squid can help your organisation to have a high-performance, stable and flexible Web content management proxy. **LFY**

The article is written by Mr Biswajit Banerjee, Director, Tetra Information Services Pvt. Ltd, who are into Linux corporate solutions. He can be contacted at biswajit@tetra.in