Nitish Kumar
mr.nitish.kumar@gmail.com
published on http://nitishkumar.wordpress.com

Making Squid Server from Scratch: The Dummies Manual

Most of us should have heard of **Squid,** mostly while discussing requirements of **restricting Internet Usages** among clients. Although a requirement for **Squid** may arise for any few of the following reasons or anything else:

1- **To limit bandwidth usages:** Squid optimizes data flow between client and server to improve performance and caches frequently-used content to save bandwidth (*As data is being accessed locally not through ISP for further requests*).

   Moreover, Organizations might have limited bandwidth or expensive over some threshold value, so management cannot permit employees to download inappropriate material as it usages precious bandwidth *(there are even options to limit the download size through Squid Server, which might be handy for such a scenario).*

2- **Due to Organizational Policy:** Sometimes, organizations might have very strict internet policies regarding offensive materials. For this and for other reasons like controlling distractions, they don't want their employees gaining access to inappropriate sites.

3- **To limit usages as per defined hours:** Sometimes, organizations might need to provide internet access to employees during certain working days/ hours only.

4- **Monitoring site access patterns:** Sometimes, in place of restricting or in addition of restricting internet access, the purpose might be monitoring the usages patterns for further steps to optimize or restrict.

Most special point about **Squid** is its being open source and vast availability of information and tweaks through forums and blogs. That's why it's most preferable solution for any such scenario.

Here I am providing the **Step By Step Dummies Manual for implementing a Squid Proxy Server** for layman like me, which should be sure helpful for many of us (including myself).

Step-by-Step with the implementation:

1- **Base Machine:** For my deployment, I chosen **CentOS** as the Linux installation due to availability and reliability of update sources for the OS itself (*its replica OS to Redhat Enterprise versions with almost all features*). The Configuration for the machine was 2.66 GHz Core 2 Duo Processor, 1 GB RAM and 160 GB HDD space.

   Installation was customized to have **2 GB swap** partition, **200 MB** boot partition, **Squid** package checked, **Web Server** packages checked, **SendMail** related packages (*Squid may be configured to send reports on mail*), **MySQL/ PHP** packages checked (*not required for Squid itself, but might be required for reporting software's later on*).

2- **Setting Up the services:** We need just one service specially Squid, but I will recommend to keep the same server up as an Apache Web Server as well, so that could customize Squid Error Messages with pics or logos.

Here is the basic way:

```
# chkconfig squid on
# chkconfig httpd on
```

The above commands will set up the services **squid** and **httpd** ON on startup. For later dealing with **Squid Service,** you can always use the following commands:

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

Although I will come up with firewall and **iptables** stuff at the later part of this manual itself (as integrating **squid** and **iptables** is kind of necessary for any production environment), but for people, who wish to keep them minimal with **squid,** here is what minimum needed to do with firewall. First check whether **port 3128** is opened or not

```
# netstat –tulpn | grep 3128
```

If not, then next part would be

```
# vi /etc/sysconfig/iptables
```

And append the following line to open up the **port 3128** for squid:

```
-A RH-Firewall-1-INPUT -m state --state NEW,ESTABLISHED,RELATED -
m tcp -p tcp --dport 3128 -j ACCEPT
```

And finally, restart of **iptables** service (Firewall service)

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

3- **Configuring Squid:** Till here, you got **Squid** services are up and running and now the next and major part remaining is setting up configurations, defining ACLs and setting Access Groups for getting a basic squid configuration running. Except creating a few files for storing domain names to allow/ deny or to store keywords to deny, now most of the part has to be done by editing Squid configuration file **squid.conf**

```
# vi /etc/squid/squid.conf
```

The starting step of playing with **squid.conf** is setting a hostname for Squid, which is essential for its working. Need to find out *visibal_hostname* and setting it by putting a name.

```
visible_hostname squidproxy
```

Now, first we need to understand the basic requirements and then have to design a policy according to that. So, **what your general requirements might be?**

1- You may require groups of IP Addresses (different sets), which will have customized web access per requirements/ policy.

2- You may require that few groups might be restricted to only few mentioned sites, few groups might require access for most of the sites (even not documented ones) and few inappropriate ones blocked either **domain-based** or **keyword-based**.

3- You may require set of user names/ passwords to access the web along with rules including the above two. (*I am not taking this specific one as my case for simplicity reasons*).

Although there are numerous **Use-Case-Scenario** for **Squid,** but I guess the above ones cover most of the corporate scenarios for basic security administration. So, I am starting with this.

For documentation and readability purpose, you need to name/ remember the various requirement groups first **like**.*IT, Management, Team1, Team2 etc.* and then we will proceed further to configure policy for each of the group.

Rest all is about **Access Control List definitions.** One can limit user's ability to browse the internet through ACLs. Each ACL defines a particular type of activity, such as an access time or source network, then all ACL statements are linked to ***http_access*** statement that tells squid that whether or not to deny or allow the traffic that matches particular ACL.

Squid matches each web access request it receives by checking the ***http_access*** list from top to bottom. If it finds a match, it enforces allow or deny statement and stop reading further (*that's why you need to be careful not to put a deny statement above similar allow statement*).

Note: The last ***http_access*** statement denies all access that's why we need to keep all of our customization above the same line.

Making Internet Access Policy: First set of rules (template): First you need to start from Access Controls section. At first you need to name a group of IP Addresses and then have to define ACLs for domain-based/ keyword-based site access blocking. I am taking the case of IT Support Web Access, where ***we need to block a selected list of sites and have to keep rest of the web opened.*** Although format is given in **squid.conf** itself, but I am

putting the format here as well. There might be two ways to define the address range as given below:

`# acl aclname src ip-address/netmask` or `# acl aclname src addr1-addr2/netmask`

In next step, it's better to keep everything allowed/ denied network, denied sites, denied keywords, so that later updating could be done without touching the **squid.conf** itself, moreover, backing up configuration would involve backing up those files and squid.conf itself that would be much cleaner and readable than usually **squid.conf** ended up to be.

Here I am taking first case of **management network** (*just an example for use case*).

Requirement is, we have to allow some specific IPs to access internet, some specific sites like orkut, facebook etc might be needed to be blocked, some specific keywords like port, xxx might be needed to be blocked and even you might have some machines in the same IP range that should not be given any internet access at all.

The following snip-set of configuration shows how to do it (**acl** names itself enough to explain).

```
# ACLs to define Management Network
#-----------------------------------------
acl management_network src "/usr/local/etc/squid/management/management_network"
acl management_deny_network src "/usr/local/etc/squid/management/management_deny_network"
acl management_deny_sites dstdomain "/usr/local/etc/squid/management/management_deny_sites"
acl management_deny_keywords url_regex -i
"/usr/local/etc/squid/management/management_deny_keywords"
#-----------------------------------------
```

Now, the next and final set of configuration entries would be selected domains and keywords denying first and then allowing rest of the web (*squid scans top to bottom*).

```
# Allow/deny web access to Management Network
#-----------------------------------------
http_access deny management_deny_network
http_access deny management_deny_sites
http_access deny management_deny_keywords
http_access allow management_network
#-----------------------------------------
```

Now, most importantly, you need to create these files at respective locations and putting required entries in them.

The profit for this approach is, any newbie could maintain the squid as usual maintenance works asks for adding/ removing IPs and adding/ removing sites and keywords for denying. It will save squid.conf from being messed up again and again by simple requirements, moreover, will keep it clean and readable.

In this way, all the files would be kept outside squid directory for keeping other IT staff not messing with actual **squid.conf** itself in case of any short term requirement. Now, there is a folder **/usr/local/etc/squid** and I'll make folders inside this folder with the names of access groups as required (like in above case, I made a folder named **management**).

management_network will keep IP addresses to allow. Syntax might be one IP in each line or range like 172.16.1.25-172.16.1.50 or 172.16.11.0/24

management_deny_network will keep IP addresses that should not get any internet access.

management_deny_sites will keep domains to be denied (one domain in each line)

management_deny_keywords will keep keywords, which if are contained in any url then the whole URL should be blocked (like **xxx**).

More Restrictive Policy for another group of IPs: Second set of rules: Now, consider a requirement, where you have to allow only provided set of domains/ websites and have to restrict rest of the web access **i.e.** just company mail site/ website.

Again, you will be needed to pick another range of IP addresses and then defining the rules in following way (*on the above pattern*). Say the network would be **MIS network**:

```
# Permission set defined for MIS Network - Nitish Kumar
# ------------------------------------------------------------
acl mis_network src "/usr/local/etc/squid/mis/mis_network"
acl mis_deny_network src "/usr/local/etc/squid/mis/mis_deny_network"
acl misGoodSites  dstdomain "/usr/local/etc/squid/mis/misGoodSites"
# ------------------------------------------------------------
```

Now, the next and final set of configuration entries would be selected domains and keywords denying first and then allowing rest of the web (*squid scans top to bottom*).

```
# Defining web access for MIS Network - Nitish Kumar
# ------------------------------------------------------------
http_access deny mis_deny_network
http_access allow mis_network misGoodSites
http_access deny mis_network
# ------------------------------------------------------------
```

Explanation for file names are similar as was in last case. Here **misGoodSites** file contain the names of those domains, which will be allowed and rest all will be restricted.

In this way, the second kind of requirement is done to restrict the web access in aggressive way, where only intimated sites would be allowed.

Note: In this scenario, you would be receiving request about site not opening in proper manners and of skipping frames/ pics etc. The reason of such issues would be third party

domain embedded in the domains we allowed. So, obviously, the frames and pics are being blocked as they are from not mentioned domain. In such a case, you need to find out these third party domains and allowing them in Good site list.

So, here is **the simplistic configuration for squid**. There might be many use cases and many on-the-fly custom issues as per scenario, which could be worked out easily on the basis of extensive support provided through blogs and forums all over the web.

Rest part of the Squid Management belongs to Internet Connection and Log Management. If Internet Connection is working over Squid server, then it should work over client after configuring proxy configuration IP/PORT in internet options.

As about directories and logs, then cache directory location is **/var/spool/squid** and log directory location is **/var/log/squid** and the important log files, while will be needed to be managed later on are **store.log, access.log, users.log** and **cache.log** Note that squid can handle maximum size of a log file as **2GB** only and after the same **squid service** will be terminated, so have to take care of that. Although fortunately, logrotate program automatically takes care of purging the data.

Now, with the above part anybody could easily configure a working Proxy Server and happily live with it later on more easier than other squid configuration manuals suggest.

For people asking for more, here are a few more tips and recommendations

Blocking MSN/ Yahoo/ Gtalk Messengers

Sure, most of you will come across such a requirement and trouble with that is leading messenger know that they would face proxy at some places so they already come with ways to bypass the proxy itself, which makes the job a bit difficult. Here is how to accomplish the same task.

First define the list of IP addresses that some smart messengers like MSN or Yahoo could use (like 64.4.13.0/24 , 207.46.104.0/24). The below section will go to network definition section.

```
acl bannedips dst "/usr/local/etc/squid/bannedip"
```

Now, how to use the rules to block messenger traffic

```
# No Messenger
# ----------------------------------------------------
acl stopmsn req_mime_type ^application/x-msn-messenger$
```

```
acl msngw url_regex -i gateway.dll
http_access deny stopmsn
http_access deny msngw
http_access deny bannedips
# ------------------------------------------------------
```

No Cache for selected sites in Squid

Caching is good for sites with mostly static content, but it could create lots of session related troubles around sites with more dynamic contents and it might be a better option to choose not caching any data for a particular set of sites. Here is how to implement it:

```
# Defining list to preventing caching for sites
# ————————————————————————-
acl prevent_cache dstdomain "/usr/local/etc/squid/No_Cache_Sites"
acl prevent_cache_file url_regex -i "/usr/local/etc/squid/No_Cache_Ext"
# ————————————————————————-
```

The above part needs to put, where network ranges are defined (above other custom rules) and the below part has to be placed where rest of **http_access** statements are placed (above other custom rules):

```
# Preventing caching for particular sites
# ————————————————————————-
no_cache deny prevent_cache
no_cache deny prevent_cache_file
# ————————————————————————-
```

And now we need to put the domains, which needs not to be cached in **No_Cache_Sites** file and File extensions not to be cached in **No_Cache_Ext** file and Squid server will stop caching for mentioned domain/ file extensions after restarting the Squid.

Need pics/ logo in squid error messages?

What if you wish to customize the error message screen you get from Squid? Sure, you have to reach the error file named **ERA_ACCESS_DENIED** somewhere in /usr/share/…. and then have to edit with normal HTML. Lots of things could be done with this, but what many people wish to do first, is trying to put some gif or logo in the same error message.

Although I don't favour putting images in error message as it make it a little heavier than originally it is, but here is the work-around.

Putting the image in same directory as ERA_ACCESS_DENIED file doesn't work and what you require is making Squid itself a Web Server (*that's why **I suggested** to keep an installation of Apache over same server*) and then referencing the image required through some

web-path of the same Squid Server. Also notice that you also needs to allow Squid Server Access to all those PCs, where this error message is expected to appear otherwise, you will get error page without any gif or pics over it.

All Network range could be allowed to access Squid server in the following way

```
# Permission set defined for Complete Network
# ————————————————————————-
acl all_network src 172.16.0.0/16
acl GoodSites url_regex -i "/usr/local/etc/squid/GoodSites"
# ————————————————————————-
```

And as per convention, I followed throughout, the above lines will go around section for ACLs defining Network range and the lines given below will go along with rest of **http_access** statements.

```
# Defining web access for All Network
# ————————————————————————-
http_access allow all_network GoodSites
# ————————————————————————-
```

Outlook and Squid Solved: Requirement of iptables (Firewall)

Why my Outlook not working behind Squid?
How can we use Outlook express or any other mail client behind Squid?
Squid running fine and filtering traffic for http access, but how to use SMTP/POP3 with Squid?

It's very easy to find people coming up with such queries. I wish to make a clear statement here **"Squid has nothing to do with Outlook or SMTP/ POP3 access".** Squid is nothing but a HTTP proxy, which could intercept requests coming over http ports only, not these POP3/SMTP ports.

Disappointed? Don't be.

Even if it's not the case of Squid, you could make use of **iptables** (In built Linux Firewall), which will not only solve the above issue, but will add up more security for your squid.

What is needed to be done with iptables is as given below:

1. First of all, the Linux Box **should act as a router** to forward all requests coming on port 25 and 100 to outside means **IP forwarding required**.

2. In next part, as IP forwarding is enabled and any request coming to Box, is going outside, so **all ports needs to be secure and controlled**.

3. Need to **redirect all requests coming to port 80 to port 3128**, where squid rules will govern internet access.

4. Need to allow **only required ports open** on Squid (like 22, 3128, 25, 110, 995, 467).

5. Could be defined that which workstations could be able to make use SMTP/ POP3 through same server.

6. Could be defined that only a few workstations could be able to do **ssh** to Squid server.

For allowing SMTP/ POP3 connections, your Linux Box (Squid Installation) needs to act as a gateway, which will be entered in **Default Gateway** entry of client PC. For doing so, one needs to enable IP Forwarding on the same.

It's disabled by default. For checking the same, you may type the following:

```
cat /proc/sys/net/ipv4/ip_forward
```

If output is **1**, then nothing to do and if output is **0**, then it needs to be ON.

For permanently putting IP Forwarding as ON, you need to change the value of `net.ipv4.ip_forward` to **1** from **0** in the file

`/etc/sysctl.conf`. The changes could take affect by either a reboot or by the command

```
Sysctl -p /etc/sysctl.conf
```

Once you have enabled it, the immediate step is to redirect all traffic of port 80 to port 3128, securing other ports, allowing required ports, allowing ICMP ping, allowing ssh etc. Edit **/etc/sysconfig/iptables** file and put the following in that.

```
*nat
:PREROUTING ACCEPT [631:109032]
:POSTROUTING ACCEPT [276:26246]
:OUTPUT ACCEPT [276:26246]
-A PREROUTING -i eth0 -p tcp -m tcp --dport 80 -j REDIRECT --to-ports
3128
-A PREROUTING -i eth0 -p tcp -m tcp --dport 80 -j REDIRECT --to-ports
3128
COMMIT
*filter
:INPUT DROP [490:62558]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [10914:7678585]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 3128 -j ACCEPT
```

```
-A INPUT -i eth0 -p tcp -m tcp --dport 25 -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 110 -j ACCEPT
-A INPUT -i eth0 -p udp -m udp --dport 25 -j ACCEPT
-A INPUT -i eth0 -p udp -m udp --dport 110 -j ACCEPT
-A INPUT -d 172.16.8.10 -p tcp -m tcp --sport 1024:65535 --dport 80 -m
state --state NEW,ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 10051 -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 10050 -j ACCEPT
-A INPUT -d 172.16.8.10 -p icmp -m icmp --icmp-type 8 -m state --state
NEW,RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -s 172.16.8.10 -p tcp -m tcp --sport 80 --dport 1024:65535 -m
state --state ESTABLISHED -j ACCEPT
-A OUTPUT -s 172.16.8.10 -p icmp -m icmp --icmp-type 0 -m state --state
RELATED,ESTABLISHED -j ACCEPT

COMMIT
```

In the above, I have enabled ports 22, 25, 110, **10051**, **10050** (zabbix), also have allowed **ICMP ping and web server (***as I will use SARG for reporting of Squid Access***) for all.**

Now, after this, if you use Squid Server's IP Address as **Default Gateway,** then you will be governed by all Squid rules (*without putting Squid's IP Address in proxy setting*) and also would be able to sent-receive emails in Outlook (*Note that currently, everyone is allowed over port 110, port 22 for all sites*).

Task: Enable or allow ICMP ping incoming client request

For people looking for enabling ICMP ping only, use following three command in order.

Rule to enable ICMP ping incoming client request (Assuming that default iptables policy is to drop all INPUT and OUTPUT packets)

```
SERVER_IP="IP_Address"
iptables -A INPUT -p icmp --icmp-type 8 -s 0/0 -d $SERVER_IP -m state --state
NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type 0 -s $SERVER_IP -d 0/0 -m state --state
ESTABLISHED,RELATED
```

Task: Allow SSH from given IP Addresses only

Rule to allow SSH from one given IP Address only (Assuming that default iptables policy is to drop all INPUT and OUTPUT packets on SSH port)

Although there are many other ways to do it, but I am putting the **iptables** way here

```
iptables -A INPUT -p tcp -m state --state NEW,ESTABLISHED -s
172.16.12.0/24 --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp -m state --state NEW,ESTABLISHED -d
172.16.12.0/24 --sport 22 -j ACCEPT
```

It will allow only IP Address of 172.16.12.0/24 series to SSH the box. Similarly individual IP Address and range could be allowed.

I hope I have provided a complete info for anyone wishing to start with Squid. Requesting you all to put your queries, so that I could make this manual better and covering more and more aspects. Although work perfectly, but **iptables** part is little messy in my manual. I would welcome, if someone suggest some **more flexible ways** (*preferably file based rules*) with easy conventions.

I also recommend using **SARG for daily/ weekly/ monthly** online reporting as its effective and very easy to use. Here is how to implement it.

So, Enjoy a Happy Safe Browsing by SQUID.