

Table of Contents

Overview.....	2
A definition of proxy.....	2
Additional benefits.....	2
Basic setup.....	2
Setting up Access List (ACL).....	3
Setting up access.....	3
A short explanation.....	3
Working proxy.....	4
Setting up browser.....	4
Transparent Proxy.....	6
Squid configuration.....	6
Is that really all?.....	6
TCP redirection.....	7
Content filtering.....	8
Lets get it working.....	8
Lists in the wild.....	9
Squid configuration finally.....	10
Summary.....	10



Overview

This paper will provide an overview of proxy servers, and an introduction to the popular open source web proxy called Squid. It will also aim to explain, in simple terms, the benefits of using a proxy server, the basic configuration of Squid, and will touch on some of the more advanced functions and administration tools.

A definition of proxy.

When a user (or 'client') types an address, such as "www.google.com", in the address bar of their web browser, it will begin searching for the IP address of the web server responsible for the domain. If it finds it, it will then connect to the web server and request the content of the web page. The content is then loaded into the client's web browser. This process involves a lot of communication between the requesting computer and the web server, as well as any servers the request routes through in order to connect to the web server. In an office, or similar situation, where several computers from an internal network may be attempting to access the same site, there will be a lot of duplicated traffic. If this duplication can be reduced, it will increase the speed of the response to the user's browser, and reduce the load on the network.

This is the sort of task the proxy server excels at. All requests for web sites are first directed to the proxy server, which makes the request on the client machines behalf. When the results are returned, they are saved in the proxy server's cache (a file on the server's hard drive), and then passed on to the machine originating the request. When the same site is requested again, from a machine behind the proxy server, the previously saved data is read from the proxy's cache, instead of being requested from open internet, speeding up the response to the user's browser and reducing internal network bandwidth.

In the early days of the internet, when whole offices could be using a single dial-up connection, proxying was essential to maintain a usable connection to the internet.

Additional benefits

Today, with fast Internet connections, the requirement for caching web sites is no longer so essential. However, there are many additional benefits provided by additional functions which can be desirable to a business. For example, businesses can log all Internet traffic, providing detail on which sites are being visited and by who. And access to unauthorised sites, such as social networking sites, can be blocked during office hours.



Basic setup.

Squid is probably the most effective and popular open-source proxy server software, so we will look at a basic setup of it below. Most Linux distributions will already have a pre-compiled package available, allowing Squid to be installed using your package manager, but in other cases it must be built from source.

Once installed, with only the default options enabled, it will provide a working proxy almost out of the box. In most cases, the only part to consider is setting up the ACL (Access Control Lists) for your network, and changing the Squid default port, if necessary.

Configuration changes.

All configuration options for Squid are contained in one central configuration file: `squid.conf`. In a Red Hat/CentOS installation this file will be located in the `/etc/squid/` directory. Open the file in text editor of your choice and look for the following sections within the configuration file:

Setting up Access List (ACL)

Search for string “# TAG: *acl*”

The `squid.conf` file contains extensive help text within it, including an explanation of creating valid ACLs. Lists can be created based on source or destination IP, source or destination domain name, browser type, port and more. We'll look at a simple setup here, assuming Squid is acting as a proxy for small internal network. In this situation our internal hosts have IP's in private network range. Therefore, first and most effective way of creating ACL would be using IP range. Such ACL would look like that :

```
acl office src 192.168.0.0/255.255.255.0
```

The ACL is called `office` and covers the subnet listed.

Setting up access

Search for string “# TAG: *http_access*”

Now, after creating ACL, we need to tell Squid what action we want to associate with the ACL. By default Squid will deny all actions from all interfaces apart from `localhost`. In the `http_access` section we have few lines allowing http access for several access lists. The most important thing to remember here is that order of rules is critical. If a rule is applied, based on the ACL, that request isn't processed any further. So, make sure rules denying access are placed before your rule granting access, otherwise they will not access with already have been allowed and processing stopped, so the deny rules will never be applied.

A rule to allow access from our 'office' ACL would look like that :

```
http_access allow office
```

As long as this line is placed before any uncommented line denying all access, it will work.

A short explanation.

Here, in more detail, is an explanation of what we've done so far – which can also be found in the Squid online documentation and the squid.conf file itself.

First line, ACL creation have four main parts :

`<acl> <name> <method> <list>`

Part	Meaning
<code><acl></code>	is key word for squid to understand what we doing in this line
<code><name></code>	is name of access list, so we can refer to it later
<code><method></code>	is one of methods to create access list. Can be for example “src” for source ip based acl
<code><list></code>	is the actual list of entries to make access rule. Have to be in type of chosen method

Second line, allowing access to proxy from given acl, have three main parts :

`<http_access> <rule> <acl_name>`

Part	Meaning
<code><http_access></code>	is key word informing squid that line is defining access level
<code><rule></code>	is setting up allow or deny rule for following acl
<code><acl_name></code>	is name of acl to which rule needs to be applied

Working proxy.

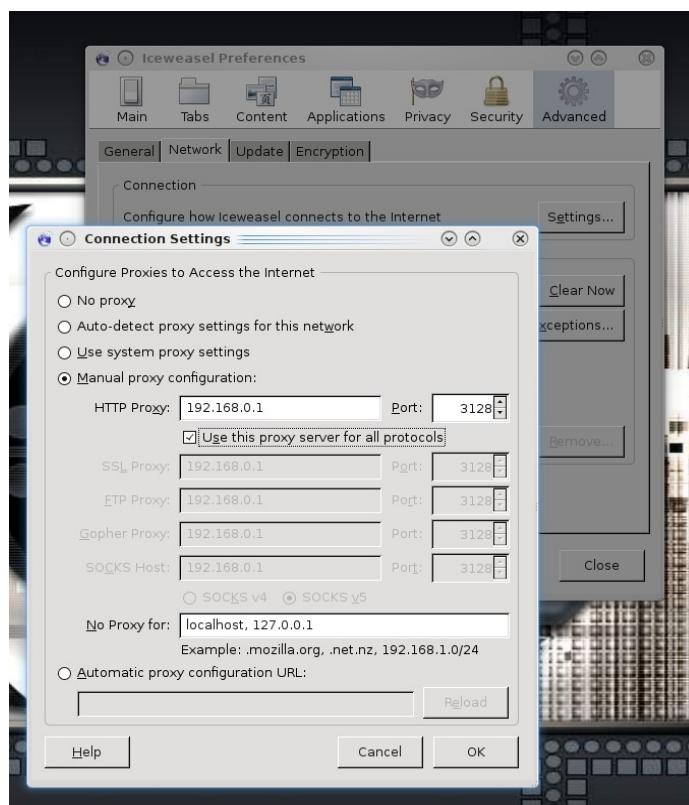
With just these simple steps, we already have a basic proxy, ready to start working. All you need now is to start it and configure the browser on client machines to use it. For example on CentOS you need to run this command :

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

On non Red Hat compatible systems, please consult your services start up documentation.

Setting up browser

Now the Proxy server is running, you need to configure the user's browser to use it. It obviously depends on the browser you using, but in Firefox/Iceweasel you need to go to Preferences (Edit->Preferences from main menu), then click on Advanced icon, choose Network tab and press Settings :



There is few possibilities there on how proxy is set up in browser. As we have one proxy server for now, we can fill only “HTTP Proxy” and tick box to use it for all other protocols, as shown on picture above. If you have any internal services served, add your network to “No Proxy” box, next to localhost interface there.

Transparent Proxy

There are some situations that setting up a proxy on a client by client basis can be problematic and work intensive. Imagine a company with 200 – 300 employees, and now imagine setting up all of their browsers by hand. If something was later changed on the proxy, it might then be necessary to manually edit these all again, which is not a very effective use of work time.

In addition, as the change are made on the clients machine, you have no way of enforcing this setting, as each user can change it or switch off proxy in browser. However, there are ways to force users to use a specific proxy. One such method is transparent proxy:

Squid configuration

Open squid.conf file with a text editor.

Search for string : “# TAG: http_port”

Under a commented out explanation you will find this default line :

```
http_port 3128
```

This line sets the proxy to listen on all available interfaces on port 3128 in normal mode. As this is not what we want, change that line to read :

```
http_port 192.168.0.1:3128 transparent
```

This sets up proxy to listen only on interface with given IP and enable transparent mode. The port has been left as default. Restarting squid will make this change effective.

Is that really all?

Transparent mode was added to squid to make network administrator life easier. It forces users to use the proxy, and you don't have to go to every machine on the network and configure each browser. However, there are two trade-offs: 1) You have to redirect all web traffic to proxy on TCP level, 2) and transparent proxy can be only used for http connections, https traffic has to be set up in conventional way, or simply allowed. This is due to how secure connections work, where transparency (where forced redirection, that is not visible to the user) can't be differentiated from a man-in-the-middle attack.



TCP redirection

If your proxy is working on your network gateway, you can use iptables to allow transparent proxy to work using just a one line rule :

```
iptables -A PREROUTING -i eth1 -p tcp -m tcp --dport 80 -j REDIRECT --to-ports 3128
```

This rule intercepts all TCP traffic from eth1 (which should be internal interface on gateway) with destination port 80 (default http port) and redirects it to port 3128, where our proxy is listening. If the proxy in your network is on server different than gateway, configuration is bit more complicated, as you need to redirect all traffic from gateway port 80 to proxy and then redirect locally to port 3128 where proxy listens.

As you can see transparent proxy is not an ideal solution. It may be that it is sufficient in your network, but it may prove useless. The obvious downside is the need for https traffic to be left untouched.



Content filtering

One useful feature that Squid offers is use of third-party software, called helpers, such as redirectors, authenticators and monitors. A redirector takes a requested URL and, depending on its pre-set rules, may seamlessly rewrite the request to a different URL. It's often used by ISPs to filter out offensive content.

One popular example is Squidguard. Here's an example of a basic configuration:

Lets get it working

Installation in CentOS with rpmforge repository enabled is achieved by just running the command :

```
yum install squidguard
```

If you have non-RedHat distribution, or installing from source, you may need to refer to the online documentation. Once Squidguard is installed, we need to add a rule list we can filter against. Depending on the method used for installation, the config file for Squidguard can be in different place, or might not even don't exist at all. Luckily, the default file is very short so can be easily created :

The content of squidguard.conf file :

```
dbhome /var/lib/squidguard
logdir /var/log/squid

acl {
    default {
        pass none
        redirect http://www.google.com
    }
}
```

Above is all that is needed for a simple config file. First two lines are declaring where filter database (lists) will be stored, and where application will log (make sure that directory exists). Next is the default access list defining redirection to google.com page for all traffic.

To make an actual list and use it, we need two things. First, create file with list of URL's or IP's to filter :

```
vi /var/lib/squidguard/blacklists/list
```

Note that file is created in side of dbhome directory.

After that, add three config directives to the squidguard.conf file :

```
src office {
    ip          192.168.0.1-192.168.0.254
}

destination mylist{
    domainlist  blacklists/list
    urllist     blacklists/url_list
}

acl {
    office {
        pass !mylist any
        redirect 302:http://internal.corporation.com/error.html
    }
    default { .... }
}
```

The first directive defines source addresses. This range will be then used in the ACL of same name at the end, so filtering applies only to certain hosts.

The second directive defines our filtering list. Note that filenames are relative to the previously defined dbhome directory. You can define what to block using specific URL's, or by domain name which causes it to matches everything from that domain.

The third directive is where the job is actioned. For a source list who's name matches the name of this ACL ('office' in our example) specific rules will be used. Note that the office ACL have to be inside acl directive brackets. This is NOT a separate acl directive.

Lists in the wild

There can be many approaches to filtering. Some will block all traffic and create whitelist of allowed sites. Others will want to restrict certain content, but allow everything else. A second approach makes everything a bit complicated, because you need to create list of all sites you want to block. This can be an incredibly complicated task but, luckily, there are some prepared lists available. One of them, still free for corporate usage (List is freely available for all users, however corporate users have to register. Commercial usage is allowed with an annual fee.), is shallalist (<http://www.shallalist.de>).

Here is how to integrate it into your Squidguard configuration :



Download shallalist into appropriate location:

```
cd /var/lib/squidguard/  
wget http://www.shallalist.de/Downloads/shallalist.tar.gz
```

Untar the list and make them available for squidguard

```
tar zxvf shallalist.tar.gz  
mv BL blacklists  
chown -R squid:squid blacklists
```

Squid configuration finally

Add this line on very end of squid.conf file :

```
redirect_program /usr/bin/squidguard -c /etc/squid/squidguard.conf
```

After restarting squid, which will apply the changes just made, our proxy is now able to filter content based on the lists provided. Every thematic list has to be manually added to the SquidGuard config file to make use of it, but this is still much quicker than manually creating the lists themselves. All lists are located in blacklists directory created previously.

Summary

What we have covered here some fairly simple examples of proxy configuration. But these simple tasks can be combined together to make filtering rules that are much more powerful and versatile. Squid has something to offer every network administrator, from a simple home network with basic content filtering, to networks spread across multiple buildings where authentication is essential, and all proxy servers need to have consistent rules applied on them.

