SquidProxyServer < Webmin < TWiki

Configuring the Squid Proxy Server

This article explains what an HTTP or FTP proxy server is, and then explains how Webmin can be used to configure the popular Squid proxy server.

Introduction to Proxying and Squid

An HTTP proxy server is basically a program that accepts requests from clients for URLs, fetches them on behalf of the client, and returns the results to the client. Proxies are used on networks where clients do not have direct access to the Internet but still need to be able to view web pages. A proxy is also used for caching commonly requested pages so that if more than one client wants to view the same page it only has to be downloaded once.

Many companies and organizations have their firewalls set up to block all incoming and outgoing traffic by systems on internal LANs. This may be done for security reasons, or to limit what employees can access on the Internet. Because being able to view web pages is extremely useful, a proxy is often set up so that websites can be accessed through it.

Large organizations and ISPs with many client PCs accessing the web may also want to run a proxy server to reduce the load on their networks. Because one of the main tasks of a proxy is caching pages requested by clients, any page asked for more than once will be returned from the cache instead of being fetched from the originating server. For this reason clients systems are often configured or forced to use a caching proxy to access the web.

A proxy is only useful if client browsers are configured to use it instead of connecting to web sites directly. Fortunately, every web browser in existence, and almost all programs that download files via HTTP for various purposes, can be configured to use a proxy. This tells them to make a special proxy HTTP connection to the proxy server instead, specifying the complete URL to download.

Proxies are not just for HTTP - they can also support FTP and Gopher protocol requests from clients, which they service by making a FTP or Gopher connection to the actual requested server. Even encrypted SSL connections can be handled by a proxy, even though it cannot decrypt the request. Instead, the proxy simply forwards all data from the client to the destination server and back again.

Squid is the most popular proxy server for Unix/Linux systems. It is open source and is freely available for download from www.squid-cache.org, and is included as a standard package with all Linux distributions and many other operating systems. Squid supports both proxying, caching and HTTP acceleration, and has a large number of configuration options to control the behavior of these features.

Squid reads its configuration from the text file squid.conf, usually found in or under the /etc directory. This file consists of a series of directives, one per line, each of which has a name and value. Each directive sets some option, such as the TCP port to listen on or a directory to store cached files in. Webmin's Squid module edits this file directly, ignoring any comments or directives that it does not understand.

Many versions of Squid have been released over the years, each of which has supported different

configuration directives or assigned different meanings to the same directives. This means that a squid.conf file from version 2.0 may not be compatible with Squid 2.5 - and one from Squid 2.5 certainly will not work with version 2.0. Fortunately, Webmin knows which directives each release supports and only allows editing of those that are known to the running version of Squid.

Cached web pages are stored in files in a multi-level directory structure for increased filesystem performance. Squid can be configured to use multiple separate cache directories, so that you can spread files over different disks to improve performance. Every time a cacheable page is requested it is stored in a file, so that when a subsequent request for the same page arrives the file can be read and the data served from it. Because some web pages change over time (or are even dynamically generated), Squid keeps track of the last-modified and expiry dates of web pages so that it can clear data from the cache when it is out of date.

The actual program that handles client requests is a permanently running server process called squid. It may also start several other sub-processes for tasks such as DNS lookups or client authentication, but all the actual HTTP protocol processing is done in the single master process. Unlike other similar servers such as Apache or Sendmail, Squid does not start or use sub-processes to handle client requests.

Squid can be compiled on all the flavors of Unix that Webmin supports, and works almost identically on all of them. This means that the Webmin module's user interface is the same across operating systems as well, with the exception of the default paths that it uses for the Squid programs and configuration files.

The Squid Proxy Server module

If you want to set up or configure Squid from within Webmin, you will need to use the Squid Proxy Server module, found under the Servers category. When its icon is clicked on, the page shown in the screenshot below will appear, assuming that Squid is installed and configured correctly. As you can see, the main page consists only of a table of icons, each of which can be clicked on to bring up a form for editing settings in that category.

The Squid module main page

If you have not configured or started Squid on your system before, the cache directory has probably not been set up yet. The module will detect this and display a message like *Your Squid cache directory /var/spool/squid has not been initialized* above the table of icons. To initialize the cache, follow these steps :

1. If you are unhappy with the displayed cache directory, now is the time to change it. Follow the instructions in the **Adding cache directories** section to define your own directories before continuing.
2. In the **as Unix user** field enter the name of the user who will own the cache files and who the daemon process will run as. Typically this will be a special squid user created for the purpose (and the field will default to squid if such a user exists), but in fact any user will do. I recommend using the Users and Groups module (covered in chapter 4) to create a user called squid whose home directory is the cache directory if needed though.
3. Click the **Initialize Cache** button. The Squid configuration will be updated to use your chosen username, and the command `squid -z` will be run to set up the cache directories. All output that it produces will be displayed so that you can see how the initialization is progressing.
4. When the process is complete, return to the module's main page and the error message should have disappeared.

If Squid is not installed at all on your system (or installed in a different location to the one Webmin expects), an error message like **The Squid config file /etc/squid.conf does not exist** will appear on the main page instead of the table of icons. If you really do have it installed, read the *Configuring the Squid Proxy Server module* section for instructions on how to change the paths the module uses. On the other hand, if it really is not installed you should use the Software Packages module (covered on SoftwarePackages) to install the squid package from your Linux distribution CD or website.

If no such package exists for your operating system, you will need to download, compile and install the latest version of Squid from www.squid-cache.org. As long as you have a compiler installed on your system, this is a relatively simple process with no dependencies.

Once the server is installed, if you want to make use of it in the long term you should arrange to have it started at boot time, using the Bootup and Shutdown module (which chapter 9 explains how to use). All Linux packages include a bootup action script for Squid, although it may be disabled by default thus requiring you to enable it in that module. Otherwise you will need to create an action that runs a command like =/usr/local/squid/bin/squid -sY=, assuming that you have Squid installed in /usr/local/squid.

Once Squid has been installed and initialized, you can start using this module. When Squid running, every page has two links at the top - **Apply Changes** which forces the current configuration to be re-read, and **Stop Squid** which shuts down the proxy server. If the server is not running, those links are replaced with *Start Squid* instead, which as the name suggests attempts to start it. If it is not yet running, you will probably want to start it now.

Because each version of Squid has introduced new configuration directives, this module's user interface will appear differently depending on the version of Squid that it detects on your system. All of the instructions in this chapter are written for Squid 2.4 as it is currently the most widely deployed version. If you are running an older or newer release, different fields may appear on the forms or have

more or fewer options. For example, each new version has introduced different ACL types, and authentication has been handled in three different ways through the history of the program. However, the basic concepts have always been the same.

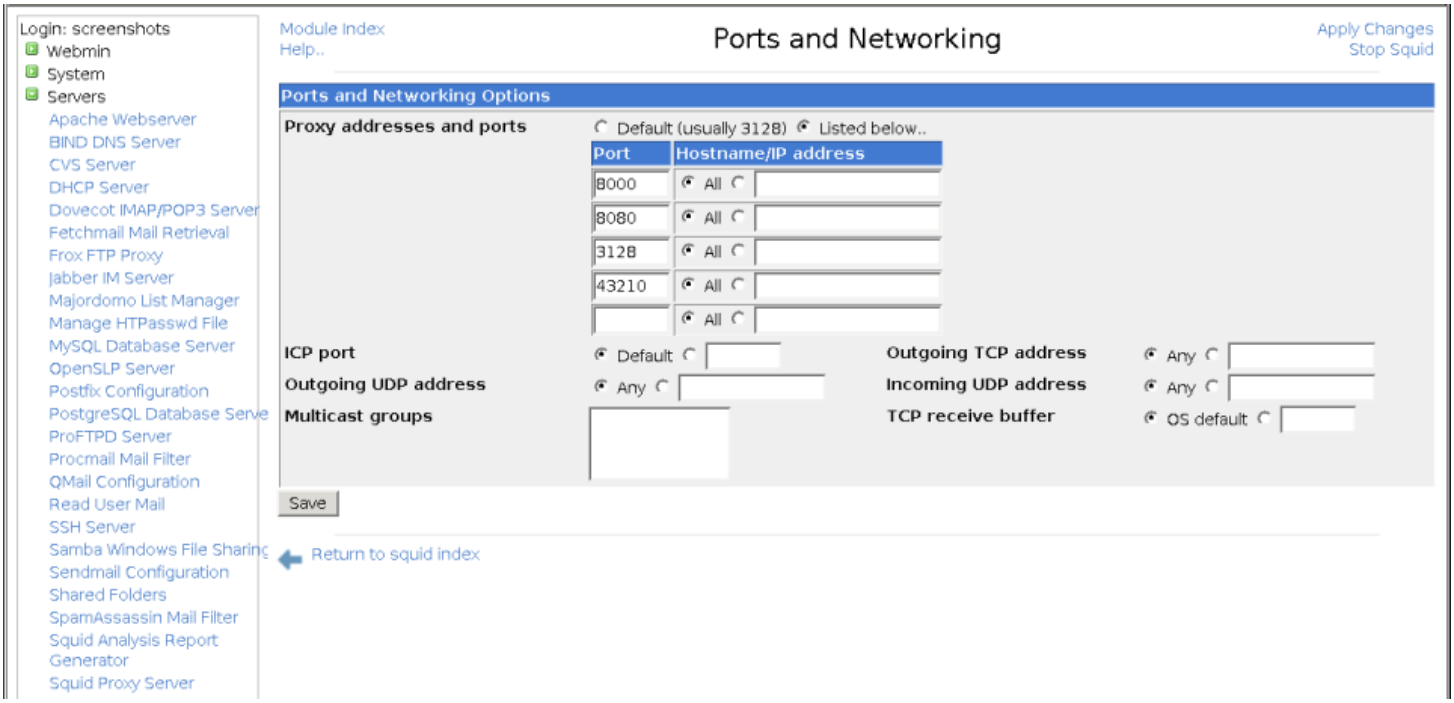
When you are using this module, make sure your browser is configured not to use the Squid proxy to access your Webmin server. Otherwise you run the risk of cutting off your own access to the module if you make a configuration mistake or shut down the server process. All browsers that can use a proxy have a field for listing hosts to connect to directly, into which you can enter the hostname of your Webmin server.

Changing the proxy ports and addresses

By default, Squid listens for proxy requests on TCP port 3128 on all of your system's IP addresses. Because this is not the usual port that proxies are run on (8000 and 8080 seem to be the most common), you may want to change it. You might also want to edit the listening address so that only clients on your internal network can connect, if your system has more than one network interface.

To specify the ports that Squid uses, follow these steps :

1. On the module's main page, click on the **Ports and Networking** icon to bring up the form shown in Figure 44-2.
2. In the **Proxy addresses and ports** table, select the **Listed below** option. In the table below, each row defines a listening port and optionally an address to bind to. Any existing ports and addresses will be listed, followed by a single blank row for adding a new one. In the first empty field in the **Port** column, enter a port number like *8000* or *8080*. In the **Hostname/IP address** column, either select **All** to accept connections on any of your system's interfaces, or the second option to enter an IP address in the adjacent text box. Using this table, Squid can be configured to listen on as many ports as you like. However, because only one blank row appears at a time you will need to save and re-open the form to add more than one new port.
3. ICP is a protocol used by Squid to communicate with other proxies in a cluster. To listen on a port other than the default of 3130 for ICP, fill in the **ICP port** field. This is not generally necessary though, as only other proxies ever use this protocol.
4. Squid will normally accept ICP connections on any IP address. To change this, select the second radio button in the **Incoming UDP address** field and enter one of your system's interface IPs into its text field. This can be useful if all of the other proxies that your server might want to communicate with are on a single internal LAN.
5. Click the **Save** button at the bottom of the page to update the configuration file with your new settings, then click the **Apply Changes** link back on the main page to activate them.



The ports and networking form

Adding cache directories

In its usual default configuration, Squid uses a single directory for storing cached pages. At most 100 MB of data will be stored in this directory, which is not likely to be enough if serving a large number of active clients. If your system has more than one hard drive, it makes sense to spread the cache across multiple disks to improve performance. This can be done by specifying multiple directories, each with its own maximum size.

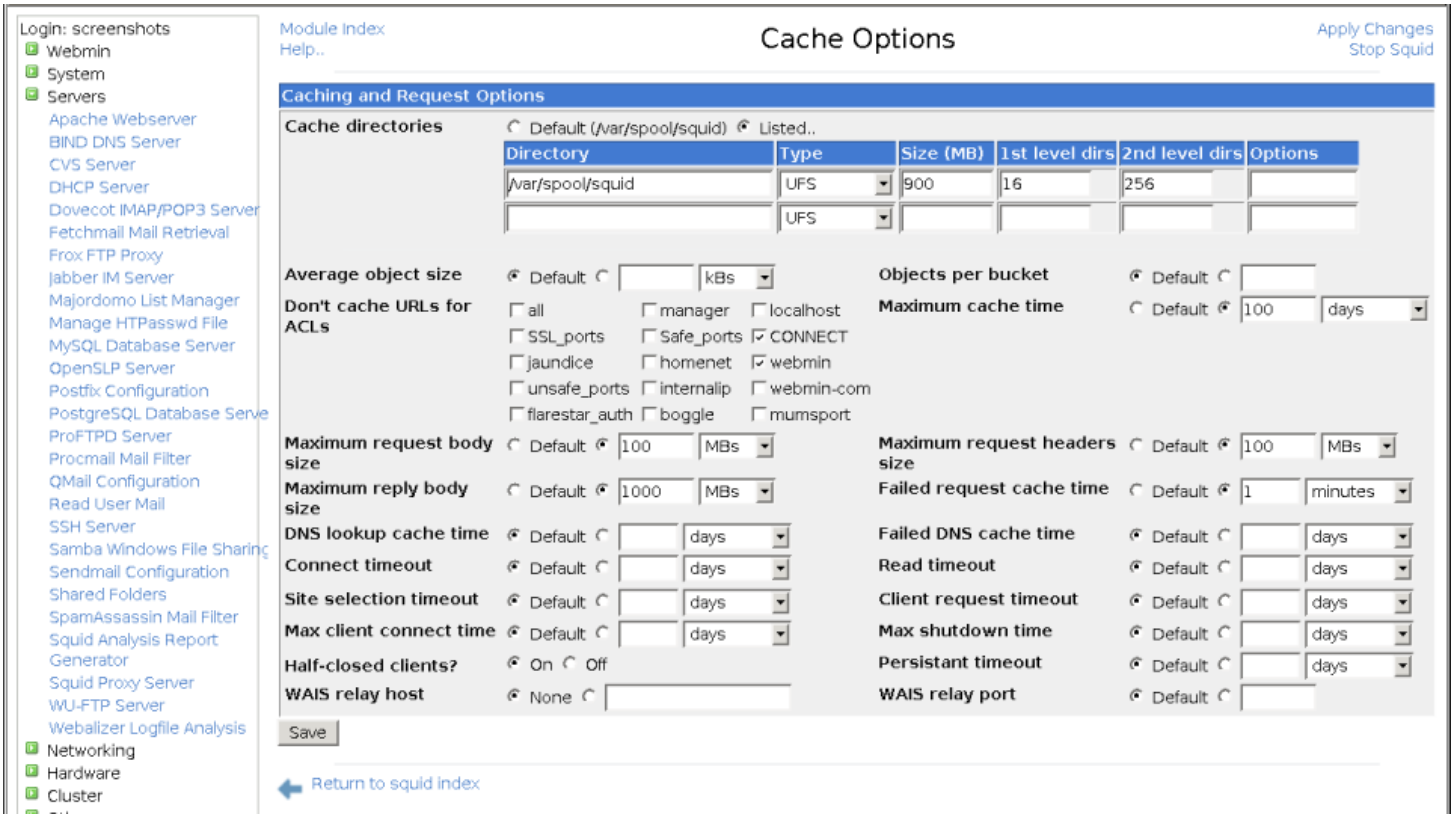
On a system that is dedicated to running a proxy server, the maximum amount to cache in each directory should be about 90% of the available space. It is unwise to configure or allow Squid to use up all free disk space, as many filesystems suffer reduced performance when nearly full. Furthermore, disk space may be used by log files and user data as well. If Squid fills up your entire hard drive, problems may occur because other programs are unable to create temporary files or write to logs.

To add a new cache directory and specify the maximum size for the existing one, follow these steps :

1. Click on the **Cache Options** icon on the module's main page to bring up the form shown in the screenshot below.
2. In the **Cache directories** field, select the **Listed** option. If **Default** was chosen before, Squid will have been using the single compiled-in default cache directory displayed in brackets. If you want to continue using this directory, it must be explicitly entered into the table. The default size is 100 MB, and it uses 16 1st level and 256 2nd level directories. Each row in the table specifies a single cache directory. Any existing directories (apart from the default) will be listed so that you can edit them, followed by a single blank row. Each row has fields under the following columns :
    - **Directory** The full path to the top-level cache directory, such as */var/spool/squid* or */disk2/cache*. This directory must already exist and be owned by the use that Squid runs as (usually called squid) - the module will not create it for you.

- **Type** The storage type used in the directory. You should always select **UFS** here.
- **Size (MB)** The maximum amount of data that it will contain, in megabytes. Once this limit is reached, the oldest un-requested files will be replaced with new ones.
- **1st level dirs** The number of subdirectories that will be created under the cache directory. The default of 16 is usually fine, but you may want to increase this for very large caches.
- **2nd level dirs** The number of subdirectories that will be created under each first-level directory. You should just enter 256 unless your cache is going to be very large.
- **Options** Leave this field blank - it is only used for other directory types. If you are wondering why Squid needs to create two levels of subdirectories under each cache directory, the reason is the poor performance of many filesystems when a directory contains a large number of files. Because every single cached HTML page or image is stored in a separate file, the number of files on a busy proxy system can be huge. Spreading them across multiple directories solves this problem.

3. After adding a directory, click the **Save** button at the bottom of the page. If you want to add more than one you will need to click on the **Cache Options** icon again to re-display the table with a new empty row.
4. When you are done defining directories, return to the module's main page. If a new one has been added, an error message like **Your Squid cache directories have not been initialized** will be displayed. Click the **Initialize Cache** button to have Squid create all the sub-directories in any new cache directories. The server will be shut down during the process, and re-started when it is complete.
5. After initialization is complete, click on the **Apply Changes** link on any page to start using your new directories.



The cache options form

Editing caching and proxy options

Squid has numerous settings that limit the size of cached objects, the size of client requests and the types of pages to cache. They can be used to stop the server storing enormous files (such as downloaded ISO images), to limit the size of files that clients can upload or download, and to prevent the cache of pages that change frequently (such as those generated by CGI scripts). The defaults will generally work fine though, with the possible exception of the maximum upload size which is only 1 MB.

To edit caching options, follow these steps :

1. Click on the **Cache Options** icon on the main page to display for form show above again.
2. To set the maximum size of uploaded files, select the second option in the **Maximum request body size** field, enter a number into the text box and select some units from the menu. 10 or
3. MB should be more than enough for anyone.
4. To stop clients downloading large files, fill in the **Maximum reply body size** field in the same way. This can be used by prevent the abuse of your network by clients downloading huge movies or ISO files, but can often be subverted by downloading a large file in pieces.
5. If you want to set an upper limit on the file that a page can be stored in the cache, fill in the **Maximum cache time** field instead of leaving it set to **Default**. Otherwise data will be cached for up to a year, or until it the expiry date set by the originating server.
6. As well as caching downloaded files, Squid will remember error messages from servers and return them to clients that request the same page. You can change the amount of time that errors are cached for by entering a number and selecting units in the **Failed request cache time** field. If **Default** is chosen, errors will be cached for 5 minutes. Even this can be annoyingly long if you have just fixed an error on a web site though.
7. Squid will cache the responses to hostname lookups to reduce the amount of DNS activity, regardless of the TTLs that the DNS servers supply. If **Default** is selected in the **DNS lookup cache time** field, responses will be remembered for 6 hours. If this seems to long for you, select the second radio button and enter your own cache time instead.
8. The **Don't cache URLs for ACLs** field can be used to completely prevent caching for certain URLs, web servers or clients. Any request that matches one of the ACLs checked in this field will never be cached, and thus will always be fetched directly. You can use this feature to block the caching of dynamically generated pages by creating a **URL Path Regexp** ACL for *.cgi* or *cgi-bin* and selecting it here. See the **Using access control lists** section for more details on how ACLs work and can be defined.
9. Hit the **Save** button at the bottom of the page to return to the main menu. Because some additional caching options are on the memory and disk usage form, click on the **Memory Usage** icon to display it
10. To limit the amount of memory that Squid will use, fill in the **Memory usage limit** field. Note that this limit only effects the maximum memory used for storing in-transit and frequently accessed files, and negative responses. Because Squid uses memory for other purposes, it will certainly consume more than whatever you enter here. If **Default** is selected, a limit of 8 MB will be enforced, which is probably too low for a busy server.

11. To prevent the caching of huge files, fill in the **Maximum cached object size** field. The default is only 4 MB, so if you have plenty of disk space it should definitely be increased.

12. Hit the **Save** button at the bottom of the form and then the **Apply Changes** link on the main page to activate all of your new settings.

Introduction to access control lists

ACLs (access control lists) are possibly Squid's most powerful feature. An ACL is simply a test that is applied to a client request to see if it matches or not. Then, based on the ACLs that each request matches you can choose to block it, prevent caching, force it into a delay pool, or hand it off to another proxy server. Many different types of ACL exist - for example, one type checks a client's IP address, another matches the URL being requested, while others check the destination port, web server hostname, authenticated user and so on.

The most common use of ACLs is blocking connections from clients outside your network. If you run a proxy server that is connected to and accessible from the Internet, hosts outside your local network should not be allowed to use it. Malicious people often use other proxies to launder connections used for hacking, sending spam or accessing web sites that they shouldn't be allowed to.

Because the special CONNECT proxy request can be used to connect to any port, an ACL is often used to block its use for any ports other than 443 (the SSL default). This stops users from using your proxy to connect to servers other than web servers, such as AIM, ICQ or MSN. Similarly, an ACL can be set up to block normal HTTP requests to ports like 22, 23 and 25 which are normally used for SSH, telnet and SMTP.

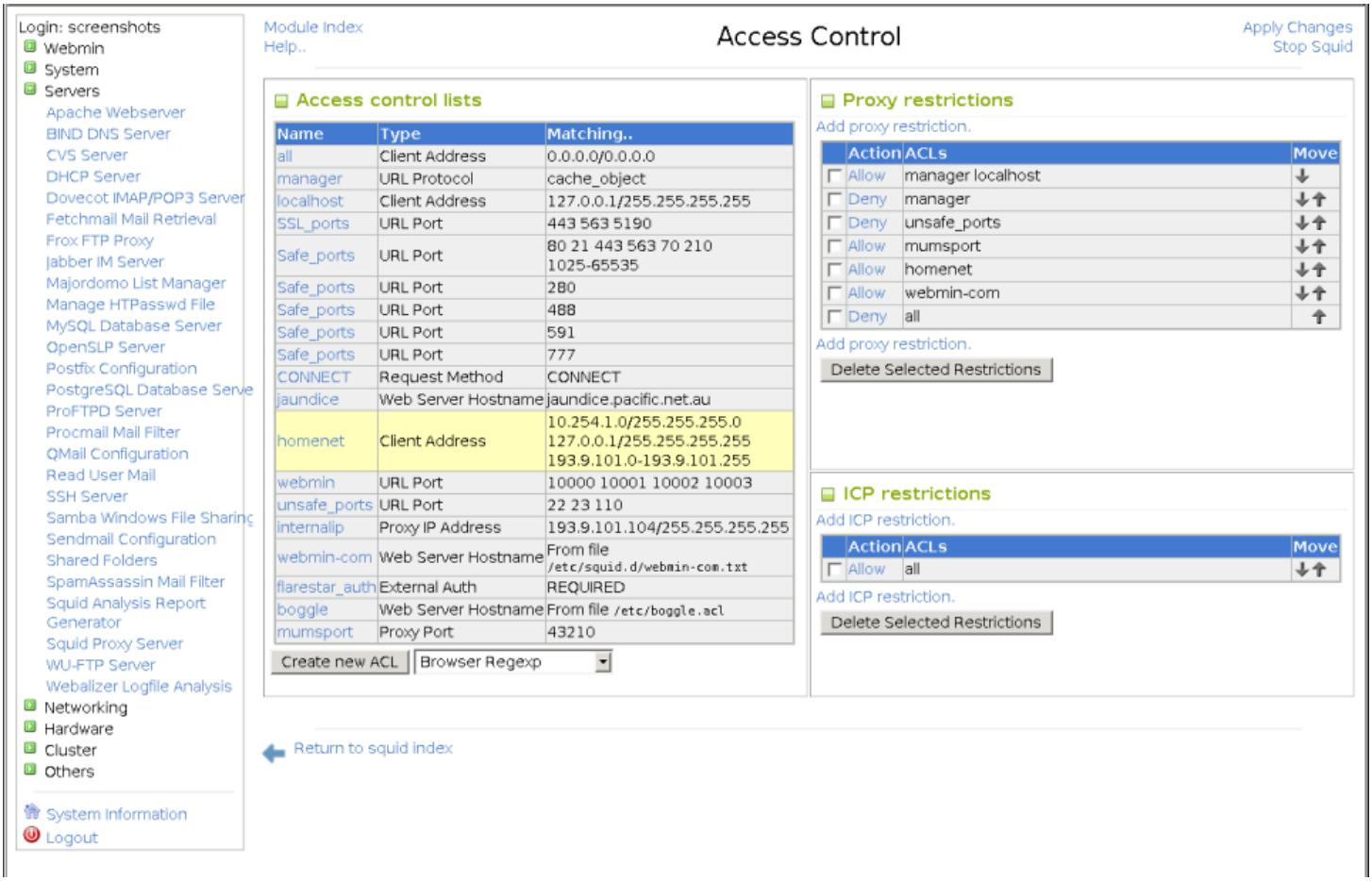
Just defining an ACL in the Squid configuration does not actually do anything - it must be applied in some way to have any effect. This section explains how to use them to control which requests to your server are allowed or denied. Other sections explain how they relate to caching and accessing other servers.

When it receives a request, Squid first determines which ACLs it matches. It then compares this list of matches against a list of proxy restrictions, each of which contains one or more ACLs and an action to perform (either Allow or Deny). As soon as a restriction is found that matches the ACLs for the request, its action determines whether the request is allowed or denied. If no restrictions match, the opposite of the last action in the list is applied. For this reason, the final action in most Squid configurations is **Allow all** or **Deny all**.

ICP requests from other proxies are also checked to see which ACLs they match, and compared against a similar but different list of ICP restrictions to see if they will be allowed or not. See the **Connecting to other proxies** section later for a more complex explanation of what ICP is and when it is used.

The typical default Squid configuration includes several ACLs and proxy restrictions. For security reasons, all requests from anywhere are denied by default. This means that you will need to change the restrictions list before anyone can use your proxy. Read on to find out how.

To view the lists of defined ACLs, proxy restrictions and ICP restrictions, click on the **Access Control** icon on the module's main page. As the image below shows, a table of ACLs showing their names, types, and matches is displayed on the left. To the right are tables of proxy and ICP restrictions showing their actions and the ACLs that they match. The restriction tables have up and down arrows next to each entry to move them in the list, because their order matters.



The access control lists page

Before clients can use your proxy you will need to configure it to allow access from some addresses. The steps to do this are :

1. On the access control page, select **Client Address** from the menu below the list of existing ACLs. When you click the **Create new ACL** button, a form for entering matching addresses will appear.
2. In the **ACL name** field enter a short name such as *yournetwork*.
3. In the empty field under **From IP** enter the starting IP address in the range to allow, such as 192.168.1.1.
4. If the field under **To IP** enter the ending address in the range, such as *192.168.1.100*. Only clients that fall within this range will match the ACL. Do NOT enter anything in the **Netmask** field.
5. Alternately, you can specify an IP network by entering the network address in the **From IP** field, and the netmask (like *255.255.255.0*) into the **Netmask** field. To enter more than one, you will need to save and re-edit this ACL so that new blank fields appear.
6. Click the **Save** button to add the ACL and return to the access control page on which your new ACL will be listed.
7. Click on **Add proxy restriction** below the **Proxy restrictions** table.
8. On the form that appears, select **Allow** from the **Action** field.

9. In the **Match ACLs** list, select your new *yournetwork* ACL.
10. Click the **Save** button on this form to go back to the access control page again. The new restriction will be displayed at the bottom of the table, most likely below the **Deny all** entry.
11. Click the up arrow next to your new restriction to move it above **Deny all**. This tells Squid to allow connections from your network, and deny everyone else.
12. Finally, click the **Apply Changes** link at the top of the page. The proxy will now be usable by clients on your internal network, but no-one else!

These instructions assume that you are starting with the default Squid configuration. If the proxy has already been configured to allow access from anywhere (by changing the **Deny all** restriction to **Allow all**), you should change it back again to block clients from outside your network. To learn more about the types of ACL available and how to use them, read the next two sections.

Creating and editing ACLs

Before you can block or allow requests from some address, to some server or for some page you will need to create an appropriate ACL. The basic steps to do this are :

1. Select the type of ACL to create from the drop-down menu below the **Access control lists** table and click the* Create new ACL* button.
2. On the form that appears, enter a name for your new ACL in the **ACL name** field. If more than one has the same name, it will be treated as matched if any ACL with that name matches. The name should consist of only letters and numbers, with no spaces or special characters.
3. Fill in the rest of the form as explained in the table below.
4. In the **Failure URL** field, enter a complete URL that clients who are denied by this ACL will be redirected to. This allows you to define custom error pages to be displayed instead of the default Squid responses.
5. Click the **Save** button at the bottom of the form.

Once an ACL has been created you can edit it by clicking on its name in the list, changing the fields and clicking **Save**. Or your can delete it (if it is not in use by some proxy or ICP restriction) with the **Delete** button. As usual, the **Apply Changes** link must be used to activate any changes that you make.

Squid has an amazing number of ACL types, although not all are available in all versions of the server. The table below lists those that you can create for Squid 2.4, and explains what they do and what the fields on the creation form for an ACL of each type mean :

Many types of ACL are inappropriate for certain situations. For example, if a client sends a CONNECT request the URL path is unavailable, and thus a **URL Path Regexp** ACL will not work. In cases like this the ACL is automatically assumed not to match.

Creating and editing proxy restrictions

Once you have created some ACLs, they can be put into use by creating, editing and moving around proxy restrictions. Squid will compare every request to all defined restrictions in order, stopping when it finds one that matches. The action set for that restriction then determines if the request is allowed or denied. This processing system combined with the power of ACLs allows you to set up some incredibly

complex access control rules - for example, you could deny all access to sites with *quake* in the URL between 9 AM and 5 PM Monday to Friday, except for certain client addresses.

To create a proxy restriction, follow these steps :

1. Click on the **Access Control** icon on the module's main page to bring up the page shown in the screenshot above.
2. Click on **Add proxy restriction** below the list of existing restrictions to go to the creation form.
3. From the **Action** field select either **Allow** or **Deny** depending on whether you want matching requests to be processed or not.
4. The **Match ACLs** list can be used to select several ACLs that if all are matched will trigger the action. Similarly, the **Don't match ACLs** field can be used to select ACLs that must not match for the action to be triggered. It is perfectly valid to make selections from both lists to indicate that the action should be triggered only if all ACLs on the left match and if those on the right do not. In its default configuration Squid has an ACL called **all** that matches all requests. It can be useful for creating restrictions that allow or deny everyone, one of which usually exists by default.
5. Click the **Save** button to create the new restriction and return to the access control page.
6. Use the arrows next to it in the **Proxy restrictions** table to move it to the correct location. If your list ends with a **Deny all** entry, you will need to move it off the bottom for it to have any effect. If the list has an entry that allows all clients from your network and you have just added a restriction to deny access to some sites, you will need to move it above that **Allow** entry as well for it to be used.
7. When you are done creating and positioning restrictions, hit the **Apply Changes** link at the top of the page to make them active.

After a proxy restriction has been created you can edit it by clicking on the link in the **Action** column for its row in the table. This will bring up an editing form identical to the one used for creating the restriction, but with **Save** and **Delete** buttons at the bottom. The former will save any changes that you make to the action or matching ACLs, while the latter will remove the restriction altogether. Again, the **Apply Changes** link must be used after updating or deleting a restriction to make the change active. If for some reason you delete all the proxy restrictions, Squid will allow all requests from all clients, which is probably not a good idea.

Also on the access control page is a table for editing and creating restrictions that apply to ICP requests. As the *Connecting to other proxies* section explains, ICP is a protocol used by Squid proxies in a cluster or hierarchy to determine what URLs other servers have cached. You can add to and edit entries in the **ICP restrictions** table in exactly the same way as you would for proxy restrictions. If you really are running a cluster of proxies, it may make sense to block ICP requests from sources other than your own network. If not, the default setup that allows all ICP packets is fine.

Setting up proxy authentication

Even though it is possible to configure Squid to allow access only from certain IP addresses, you may want to force clients to authenticate themselves to the proxy as well. This might make sense if you want to give only certain people access to the web, and cannot use IP address validation due to the use of dynamically assigned addresses on your network. It is also handy for keeping track of who has requested what through the proxy, as usernames are recorded in the Squid logs.

All browsers and programs that can make use of a proxy also support proxy authentication. Browsers

will pop up a login window for entering a username and password to be sent to the proxy the first time it requests them, and automatically send the same information for all subsequent requests. Other programs (such as wget or rpm) require the username and password to be specified on the command line.

Each login and password received by Squid is passed to an external authentication program which either approves or denies it. Typically this program checks against a separate users file, but it is possible to write your own programs that use all sorts of methods of validating users - for example, they might be looked up in a database, or an LDAP server, or the Unix user list. Webmin comes with a simple program that reads users from a text file in the same format as is used by Apache, and this module allows you to edit users in such a file.

The steps to turn on authentication for your Squid proxy are :

1. On the module's main page, click on the **Access Control** icon to bring up the form shown in Figure 44-4.
2. Select **External Auth** from the menu below the ACL table and hit the **Create new ACL** button.
3. In the form that appears, enter *auth* for the **ACL name** and select **All users** in the *External