Appendix B
Squid ACL Primer

This is a crash course on Squid ***Access Control Lists*** (***ACLs***) in Squid. For more background on how to use Squid in your organisation, see chapter six, **Performance Tuning**.

There are two types of ACL objects in Squid: ***ACL elements*** and ***ACL rules***. Both are used together to implement access control on your Squid cache. ACLs are specified in your **`squid.conf`** file. When you are finished making changes to your ACLs, you should run **`squid -k reconfigure`**. This causes Squid to reload the config file and use your new settings.

ACL elements

ACL elements are the basic building blocks of the access control features of Squid. They let you define elements that identify a request, such as IP addresses, port numbers, host names or user names.

The basic syntax of a ACL element is:

```
acl name type value1 value2 value3 ...
```

It is important to note that elements use OR logic to process the values. This means that as soon a match is found, the element will return true. It is possible to refer to the same element name on multiple lines, as well as on the same line. For example:

```
acl myinternalip 10.1.0.0/255.255.0.0 10.2.0.0/255.255.0.0
```

...is the same as:

```
acl myinternalip 10.1.0.0/255.255.0.0
acl myinternalip 10.2.0.0/255.255.0.0
```

ACL elements can also be contained in external files. This makes it easy to manage a long list of values.

```
acl myinternalip "/etc/mynets"
```

This directs Squid to read **`/etc/mynets`** to populate the **`myinternalip`** element. Each value should be on its own line in the **`/etc/mynets`** file.

Some other useful ACL elements are:

- **`src`**. This will match the IP address of the client making the request.

```
acl aclname src [ip-address/netmask]
```

- **`dst`**. This refers to the web server's IP address. When using this element in an ACL, Squid performs a DNS lookup on the host specified in the HTTP request header. This can add a delay to ACL processing as Squid waits for the DNS response.

```
acl aclname dst [ip-address/netmask]
```

- **`dstdomain`**. This matches the domain of the site requested by the client. A leading **`.`** before the domain will cause all subdomains to be matched as well.

```
acl aclname dstdomain [domain-name]
```

- **`dstdom_regex`**. This is similar to **`dstdomain`**, but uses a regular expression to match the domain.

```
acl aclname dstdom_regex [pattern]
```

- **`time`**. This matches the time and day of the week.

```
acl aclname time [day] [hh:mm-hh:mm]
```

The day is a one-letter abbreviation as follows: **`S`** (Sunday), **`M`** (Monday), **`T`** (Tuesday), **`W`** (Wednesday), **`H`** (Thursday), **`F`** (Friday), or **`A`** (Saturday). The last parameter specifies a range of time, and is specified in 24-hour format. For example, this would match Monday through Friday, 8:00am to 6:00pm:

```
acl business_hours time MTWHF 8:00-18:00
```

- **url_regex**. This searches the entire URL for the regular expression specified. Note that these regular expressions are case-sensitive by default. To make them case-insensitive, use the **-i** option just before the pattern.

```
acl aclname url_regex [pattern]
```

- **urlpath_regex**. This performs regular expression pattern matching on the URL, but without the protocol and host name. Note that as with **url_regex**, these regular expressions are case-sensitive by default unless you use the **-i** switch.

```
acl aclname urlpath_regex [pattern]
```

- **port**. This matches the destination port address used by the web server.

```
acl aclname port [number]
```

- **proxy_auth**. Matches an authenticated proxy user. Note that **proxy_auth** requires an external authentication program (which is specified by the **authenticate_program** tag). Multiple user names are separated by spaces.

```
acl aclname proxy_auth [user names]
```

  You can use the magic term **REQUIRED** instead of an explicit user name to accept any valid user name.

- **proxy_auth_regex**. Similar to the proxy_auth element, but uses a regular expression to match the user name. Remember to use the -i switch if you wish to ignore case when matching.

```
acl aclname proxy_auth_regex [pattern]
```

ACL rules

ACL rules combine ACL elements to control access to certain features of Squid. It is important to remember is that the ACL rules use AND logic. This means that all elements on line need to evaluate to true in order for the rule to execute. The rules are processed from the top down. As soon as a rule matches, the rule is executed and all subsequent rules using the same rule type will be ignored.

It is a very good idea to keep ACL rules of the same type together. It can be very easy to get confused if you have, for example, your **http_access** rules scattered all over your **squid.conf** file.

There are several ACL rule types. Here are some of the most commonly used rules.

- **`http_access`**. Allow or deny http access based on a previously defined element.

```
http_access [allow|deny] [aclname]
```

You can precede any element with a **`!`** to match anything that is **not** on the list. If none of the access lines match, then the default is to do whatever is the opposite of the last line in the list. It is a good idea to insert a **`deny all`** or **`allow all`** entry at the end of your access lists to avoid confusion.

- **`icp_access`**. If your Squid cache is configured to serve ICP replies (when using hierarchical caches), you should use the **`icp_access`** list. In most cases, you should only allow ICP requests from your neighbor caches.

```
icp_access [allow|deny] [aclname]
```

- **`redirector_access`**. This access list determines which requests are sent to one of the redirector processes. If you do not specify a **`redirector_access`** line, all requests go through the redirectors. You can use this list to prevent certain requests from being rewritten.

```
redirector_access [allow|deny] [aclname]
```

- **`delay_access`**. This access list rule determines if a client should be sent to a delay pool. Note that the client is directed to the first delay pool that matches.

```
delay_access [delaypoolnumber] [allow|deny] [aclname]
```

Examples

Here are some examples of how to perform common ACL tasks in Squid.

Allow only local clients

Almost all Squid installations need to restrict access based on the client's IP address. This is one of the best ways to protect your cache from abuse and bandwidth theft. This example will only permit clients from **`MyNet`** to access the cache.

```
acl MyNet src 10.2.1.0/24 10.2.2.0/24
http_access allow MyNet
http_access deny All
```

Deny a list of sites

Add the list of sites you wish to block to the file specified by **BadSites**, with one site per line.

```
acl BadSites dstdomain "/usr/local/squid/etc/badsites"
http_access deny BadSites
http_access allow MyNet
http_access deny All
```

Block a few clients by IP address

```
acl BadPc src 10.1.2.4
http_access deny BadPc
http_access allow MyNet
http_access deny All
```

Allow access to the bad sites only after hours

```
acl workhours acl workhours time MTWHF 08:00-17:00
acl BadSites dstdomain "/usr/local/squid/etc/badsites"
http_access deny workhours BadSites
http_access allow MyNet
http_access deny All
```

Block certain users regardless of their IP address

```
acl Authenticated proxy_auth REQUIRED
acl BadUsers proxy_auth Richard John
http_access deny BadUsers
http_access allow MyNet
http_access deny All
```

Direct certain users to a delay pool

```
acl Authenticated proxy_auth REQUIRED
acl SlowUsers proxy_auth Richard John
http_access allow MyNet
http_access deny All

delay_access 2 allow SlowUsers
delay_access 2 deny # This so no other users match this pool
delay_access 1 allow all
```

For more information about ACLs, see the online Squid documentation at *http://www.deckle.co.za/squid-users-guide/.*