

Squid and Novell eDirectory

Matt White

December 19, 2002

Contents

1	Why Squid and eDirectory?	3
1.1	Background	3
1.2	About Squid	4
1.3	About Novell eDirectory	4
2	Preparing the Server	5
2.1	Physical Memory Considerations	5
2.2	CPU Speed Considerations	5
2.3	File System Considerations	6
3	Installing Squid	6
3.1	Version Information	7
3.2	Installation Issues	7
3.3	A Really, Really Brief Install Guide	7
4	Novell eDirectory and LDAP	7
4.1	How Squid Uses LDAP	8
4.2	Getting Your Tree Ready For Squid	9
4.2.1	Before You Begin.	9
4.2.2	Location of the LDAP Server	9
4.2.3	Create a User For Squid to Login As	10
4.2.4	Test, test, test!	10
4.3	Non-NetWare Platforms?	10
5	The Squid LDAP Helper	10
5.1	About OpenLDAP	11
5.2	Building the LDAP Helper	11
5.3	Configuring the LDAP Helper	11
5.3.1	Configuration Options	12
5.3.2	Understanding the Search Filter	12
5.3.3	Sample Configuration	13
5.4	Testing the LDAP Helper	14
5.5	Moving the LDAP Helper	15
6	Configuring Squid	15
6.1	Instructing Squid to Use the LDAP Helper	15
6.2	Create a Basic ACL	16
7	Wrapping It All Up	17

1 Why Squid and eDirectory?

This section is not important to setting up Squid and eDirectory — its purpose is to provide some background into why Internet Tracking came about. If you're only interested in how to setup Squid and eDirectory, just skip this section altogether.

1.1 Background

In the 1990's there was a rush to wire school buildings with computer networks, and to connect those networks to the Internet. Thousands of miles of cable were run, and lots of bandwidth was purchased. One by one, schools were brought online and connected to the Internet. The rush was very successful: today it is hard to find a school that isn't connected to the Internet.

There was a major problem with the Internet rush, however. In the beginning little attention was given to the fact that not all of the content on the Internet is purely educational. As we all know today, there are a lot of sites where school-aged children *shouldn't* be going, at least not without parental supervision, and especially not at school...

...and thus, the Internet filtering industry was born. However, no filter is ever perfect — and some students will go out of their way to find sites that the filter is not yet programmed to block.

In our school district every time a case came up where a student was accused of going to a website which is against our Acceptable Usage Policy, the Technology Department would be called in to check the different computers that the student could have been using looking for evidence of the violation occurring. This took a great deal of time, per incident, and the information that was collected was normally very hard to link to an exact time and date, as well as a user name. (Not to mention that the information was located on the local hard drive of the client PC, which is in the hands of the student!)

We (the Technology Department of Mad River Local Schools) knew there had to be a better way to get a log of where students went on the Internet, so we went looking to find it. We talked to Wright State University to see how they track where students go, as well as talked to a commercial company for ideas and quotes as to what it would take to log Internet usage.

We had three main goals:

1. Track the usage of the Internet to a specific time, date, user name and client PC.
2. Use our existing NDS (eDirectory) database for login names and passwords.
3. Be able to grant us fine-grained control – down to the user account level – as who is able to access the Internet. (And when, if possible.)

One of the quotes we received was for well over \$15,000 dollars! The system that was quoted would have met all of these goals and more, but was overkill for what we wanted to do.

1.2 About Squid

Squid is an open source, free proxy server¹. Squid runs on most UNIX-like platforms (including Linux), and has been tweaked over time to be a stable and fast proxy server. Squid will cache the objects that it fetches, and store them in memory (and later to disk), so that if another client asks for the same object later, it's already in memory, and doesn't have to go out across the Internet to get it. This speeds up access times for frequently loaded pages in a busy network.

Squid can be configured to require a client to authenticate to it before it will honor it's requests, and this is the feature we use to track Internet usage, as well as ensure that only authorized accounts are actually allowed out onto the Internet. (Clients are configured to use the Squid server as their proxy server, and the rest of the ports are blocked — thus locking the client into using the proxy server and only the proxy server for access to the World Wide Web.)

1.3 About Novell eDirectory

eDirectory is the new name for Novell's directory service that first appeared in NetWare 4.0. It has been tweaked over the last ten years to be a highly scalable and fully replicated database. Novell NetWare servers use eDirectory to store all login information — if you use NetWare for your Network Operating System, you already have the user database you need for Internet Tracking built. There should be no need to maintain two separate databases, one for server login information and one for Internet authentication.

¹A proxy server is a server that a client can act to go fetch a web page on it's behalf.

Novell recently made a stand-alone version of eDirectory that doesn't require NetWare to run, but isn't common to find in schools.

2 Preparing the Server

This document is going to focus on the Linux operating system. This is not to say that Squid will not work on other operating systems — quite the opposite. It works very well on most of the BSD operating systems (FreeBSD, OpenBSD, etc), and most likely will work great on MacOS X. Mad River Local Schools has had great success with Squid on Linux, which is why I am going to focus on Linux. (Not to mention that you can't beat the price!) I have not actually done any work with Squid on Windows. It might work there, it might not. If you can find a pre-compiled binary, then you might be okay.

I will not go into the details of installing Linux here. There is a lot of documentation available all of the World Wide Web on how to install Linux, the last thing we need is another one.

This document also assumes that the GNU C compiler (`gcc`) is installed. On most Linux distributions, it is installed by default — if not, it is easy to include it when installing the operation system.

2.1 Physical Memory Considerations

If your Squid proxy is to be used for Internet Tracking purposes only, then the amount of RAM in the server is not very critical. I would guess that 128MB will do. (Maybe much less, depending on how you have the server configured.) Our proxy is also used for caching, so we our Squid server has 512MB of RAM. Squid has been configured to take up as much of that as it is wants.

2.2 CPU Speed Considerations

Our Squid server has a Pentium III 1.0GHz CPU. This is overkill, you most likely can get away with much less. If you want to enable some of the extra features that Squid can do (such as ad-blocking services), remember that these features come at the expense of CPU power.

2.3 File System Considerations

There are two issues to be consider when creating a Squid proxy server that will be used for Internet Tracking purposes.

1. Squid is a caching proxy server — it stores objects that it’s downloaded onto the local drive to speed up accessing that object the next time a client wants to see it. It will be generating lots of small files for its cache.
2. The log files will be *huge*. A typical day’s worth of logs here in Mad River runs around 100MB.

If your Squid server will be used for one purpose only, such as ours, I recommend that you create several partitions for the operation of the proxy — one for the cache files that has a file system tuned for small files (such as ReiserFS on Linux), and another for the log files. Why? In case either the log files or the cache fill the entire hard drive, the server will not crash due to running out of disk space.²

The partitioning scheme on our proxy server looks like this:

Figure 1: An excerpt from the `/etc/fstab` file on our Squid server.

```
/dev/hda2          /                reiserfs  defaults
/dev/hda1          /boot            ext2       defaults
/dev/hda7          /usr/squid/cache reiserfs  defaults
/dev/hda8          /usr/squid/log  reiserfs  defaults
/dev/hda6          /var             reiserfs  defaults
/dev/hda5          swap             swap       pri=42
```

If your Squid server is also going to serve another function (such as a web server), keep it’s disk space requirements in mind as you install Squid.

3 Installing Squid

This section is going to be pretty brief. There are also lots of Squid install “HOWTOs” available on the net.

²Squid itself will still crash, but you’ll have a much easier time recovering from that situation than if the entire volume is full.

3.1 Version Information

These directions are based on Squid v2.5-STABLE1. The official website for Squid is located at <http://www.squid-cache.org/>, and the tarball I am using is called `squid-2.5.STABLE1.tar.gz`.

The Squid website is full of installation guides for a lot of different platforms. Follow the one for the platform that you are using.

3.2 Installation Issues

Follow the installation instructions for Squid as needed — however, if you created partitions on your hard drive for different directories for Squid to use, make sure you configure it to match.

Setting up Squid to use eDirectory comes after Squid itself is installed — it is largely a configuration issue, and not an install issue.

3.3 A Really, Really Brief Install Guide

If you're familiar with how to install packages on Linux, here's the commands I used to compile and install Squid:

```
cd /usr/local/src
tar -xzvf squid-2.5.STABLE1.tar.gz
cd squid-2.5.STABLE1
./configure --prefix=/usr/squid
make
make install
```

Now I have a working copy of Squid installed on my server:

```
mwhite@rover:/usr/squid/sbin> ./squid -v
Squid Cache: Version 2.5.STABLE1
configure options:  --prefix=/usr/squid
```

4 Novell eDirectory and LDAP

eDirectory is the new name for Novell's product that was known as NetWare Directory Service, or NDS. NDS is no longer tied to NetWare, and is much more than just a simple directory, so it was renamed. More information about eDirectory is located at <http://www.novell.com/edirectory>.

LDAP stands for *Lightweight Directory Access Protocol*, and it's nothing more than a standard way to access directory services from an application. It is not a server itself, it is just a protocol. The term "LDAP Server" refers to a server that uses the LDAP protocol, not a product itself.

Novell included LDAP support in eDirectory version 8.0, and greatly enhanced it in version 8.5. We use NetWare file servers, and thus have an eDirectory (NDS) "tree". The version of eDirectory we are currently using is 8.6.2.

Squid can be configured to be an LDAP client; you can make it use LDAP to check to see if a user has a valid account, instead of having Squid maintain it's own user and password database. Since we already have an eDirectory tree, and every user in our network is already in it, linking the two together makes for a very powerful combination. It allows for every user to be able to use their "server" user name and password to access the proxy server — when they change their password in eDirectory, Squid automatically follows.

4.1 How Squid Uses LDAP

When an un-authenticated user attempts to access the Squid proxy, Squid will prompt them for their username and password. Squid then takes this username and looks for a user in the tree that matches what it was given by the user. If it finds one, it attempts to login to the tree as that user. If it works, then the user supplied the correct password and access to the proxy is granted.

In eDirectory, there are a few "gotchas" that you might not be aware of at first. The first is that Squid needs a way to log into the tree to be able to do a full search of the tree. By default eDirectory will not allow a user to do this — it could be considered a major security flaw! (If you know a valid user name, you're half way to cracking into a tree... now all you need is a password.)

By default Squid will attempt to do an anonymous bind to the tree in order to search it. (This is the [Public] user in NDS.) NDS doesn't allow for scanning the tree by default (see the security problem above), so this will fail.

To fix this problem, Squid needs a "searching user" to bind (login) to eDirectory as. This user needs to have rights to be able to search the tree, which on eDirectory means "read" and "file scan" rights. Our eDirectory tree has a user called "ApacheLDAP" just for this purpose. ApacheLDAP has been granted read and file scan rights to the root level of the tree, and

it has not been blocked further down.

Since the ApacheLDAP user is password protected, and read-only, the security impact it has is minimal.

4.2 Getting Your Tree Ready For Squid

4.2.1 Before You Begin...

There are a couple of issues you will want to consider before you allow Squid to access your tree...

1. Which eDirectory server is Squid going to connect to? Is there one close to it on the network? Can it handle a high number of LDAP queries per day?
2. Do you want to create a group that users must be in order to have rights to use the proxy server? (For example an “Internet Users” group of students that have signed the Acceptable Usage Policy.)
3. Do you have duplicate user names in different OUs in your tree? If so, that is a problem since Squid will look for the first matching user name only.

In our tree we have a group called “Internet Users” that we require users to be in before they are allowed to access the proxy. Why? Because this allows us to have fine-grained control over who’s allowed to access the proxy and who isn’t. This is a big deal because it is possible for a student to lose privileges to access the Internet, and we need a way to enforce the policy.

4.2.2 Location of the LDAP Server

On a busy Squid proxy server, it’s possible it will generate a large number of LDAP queries a second. eDirectory is perfectly capable of handling this, as long as it has enough RAM to work with. Do you need a separate server for LDAP? Most likely no. We do, but mainly because we have a Novell School License Agreement (SLA), which means we have enough licenses of NetWare to play with, and I had an old server I could install it on.

The server needs to be close enough to the Squid server that it won’t be a strain on your network. (Crossing subnets is a bad idea!) Novell has several TIDs in their knowledgebase about tuning LDAP performance on NetWare.

4.2.3 Create a User For Squid to Login As

Squid needs a user to connect to your tree as; now is the time to make that user. It can go anywhere in the tree, just as long as you know where it is. (In our tree it's in the OU that contains the administrator accounts.) Once the user is created, give this user read and file scan rights to either the O level of your tree, or to the root level, depending on where your users are located.

4.2.4 Test, test, test!

Now is the time to test out the LDAP server. The best and easiest way to do this is with one of the free LDAP browsers out there. I use one called "LDAPBrowser", which can be downloaded for free at the address <http://www.iit.edu/~gawojar/ldap/>.

Log into your NDS tree as the user you created above using the LDAP Browser. While logged into the tree via LDAP as the searching user check to see how much of the tree you can see. If you're able to look at accounts, then Squid should be able to also.

This is an important step! If you can't log into LDAP with an LDAP browser and see your tree, Squid won't be able to either. Once you have this working, the rest of the install should go smoothly.

4.3 Non-NetWare Platforms?

There's no reason this wouldn't work on non-NetWare platforms. Squid uses LDAP to talk to eDirectory, no matter what platform eDirectory is running on. Most of these instructions will work just fine on any eDirectory platform.

5 The Squid LDAP Helper

Squid uses various "helper" applications to do authentication. This is a very handy way of handling authentication, as it allows developers to write a helper application for Squid easily.

This section of this document requires some knowledge how to compile applications in Linux. I'll try to keep it fairly simple. For this section, "SRC" will refer to the directory where you uncompressed the Squid sources into. On my server that's /usr/local/src/squid-2.5.STABLE1.

5.1 About OpenLDAP

OpenLDAP is a set of libraries and client applications to allow an application to access an LDAP server. Most Linux programs that use LDAP use OpenLDAP to do it. The easiest way to check and see if OpenLDAP is installed is to just go ahead and try to build the LDAP helper and see if it works (see the next subsection!).

If the build fails with an error message like...

```
squid_ldap_auth.c:53:18: lber.h: No such file or directory
squid_ldap_auth.c:54:18: ldap.h: No such file or directory
```

...your system is missing OpenLDAP. Again, installing OpenLDAP is beyond the scope of this document — head over to the official OpenLDAP website located at <http://www.openldap.org> to download it, and find installation instructions. Grab the “stable” branch, Squid doesn’t use any of the new features in the “current” branch.

Here’s one hint — if you’re building OpenLDAP instead of installing a package for your operating system, give configure the “--disable-slapd” parameter. SLAPD is an LDAP server, and it requires a lot of things to build. (Besides, if you have eDirectory, you most likely could care less about a little LDAP server running on your Linux box...)

5.2 Building the LDAP Helper

The LDAP helper application should be located in SRC/helpers/basic_auth/LDAP. Change to this directory (cd), and just type “make” (without the quotes). If all goes well, the LDAP helper application should compile.

5.3 Configuring the LDAP Helper

When Squid is configured to use a helper application to check authentication, it starts up a number of them (that you specify in the configuration file) when it starts up. When it needs to check a password, it uses one of the running applications to do it — by giving it a username and password, and allowing the helper application to do all of the work.

If the helper application isn’t configured correctly, all Squid will know is that it’s not the right password, and you will get little information back to work with. However, the good news is that it’s really easy to fire up one of the helper applications and see if it is able to authenticate users correctly.

5.3.1 Configuration Options

The command line options for the helper application are listed in the following table:

-b	The DN of the “base” location in your tree where the helper application should search. (In our case, that’s the O level of the tree – all user accounts are in an OU below it.)
-f	Filter to use while searching for the user’s DN. (Explained in detail in the next subsection.)
-u	The name of the attribute that the username is listed under.
-s	One of “base”, “one” or “sub”. Telling the helper if it should search the location where the base DN is, or if it should traverse down the tree.
-D	The DN to login to the tree for searching. (The user you created above.)
-w	The password to use for the searching login.
-h	The hostname (or IP address) of your eDirectory server.
-p	The port number on the server to connect to. (Unless you’ve changed it, the default will work just fine!)
-P	Use a persistent LDAP connection.
-R	Do not follow referrals.
-s	One of “never”, “always”, “search” or “find”. This tells the helper when to dereference aliases.

Table 1: Squid’s LDAP Helper Command Line Arguments

Getting these parameters to the helper application correct is the most critical part of the configuration. Once you are able to get the helper configured correctly, Squid will not have a problem using eDirectory. (See our configuration below as an example.)

5.3.2 Understanding the Search Filter

Most of the configuration options are straight-forward, except for the search filter. (Command line parameter -f.) This filter is passed directly onto the LDAP server, which means that it has to be correct — but it also allows you to have a high degree of customization of the operation of the helper.

You might have noticed that there’s isn’t a command line option to configure a group that the user must be in. Why? Because it’s easy to implement with the search filter.

Here is a basic search filter that will work for most situations:

```
(&(objectClass=person)(cn=%s))
```

“%s” will be filled in by the username that the user provides when asked for their username. This tells the LDAP server to return the DNs of all the objects of class “person”, where the cn attribute is the username supplied by the user. (LDAP searches use a very LISP-like syntax!) If you do not want to require users to be in a specific group, then the above search filter is all you need.

In our network, we want to require people to be in a specified group in order to have access to the Internet – that is, if you’re not a member of the “InternetUsers” group, you can not access the Internet. How do we implement this? With a search filter.

The search filter that we use looks like this:

```
(&(&(objectClass=person)(cn=%s))(groupMembership=cn=InternetUsers,o=MRLS))
```

What is this saying? Return me a list of DNs where the cn attribute is the username that the password supplied, the object is of type “person”, and the object has the attribute “groupMembership” set to “cn=InternetUsers,o=MRLS”. (MRLS is the name the O level in our tree.) This gives us the ability to define who is able and isn’t able to access the proxy via groups in eDirectory!

This is the filter we use. You can get far more complex if you choose, the LDAP helper gives you all the tools you need.

Here’s a hint — Be careful with spaces in your group names! You will need to put a \ (backslash) in front of any spaces to tell the LDAP Worker “this isn’t a new parameter”.

5.3.3 Sample Configuration

The configuration that Mad River Local Schools uses is this:

```
./squid_ldap_auth          \
  -b o=MRLS                \
  -u cn                    \
  -s sub                   \
  -D cn=ApacheLDAP,ou=CO,o=MRLS \
  -w indians               \
```

```
-f "(&(&(objectClass=person)(cn=%s)) \
      (groupMembership=cn=InternetUsers,o=MRLS))" \
-h ldap-server.madriver.k12.oh.us
```

This tells the LDAP helper to log into the tree via LDAP server ldap-server.madriver.k12.oh.us as “cn=ApacheLDAP,ou=CO,o=MRLS” with the password of “indians³”, and to start looking at the “O=MRLS” level, and to search all available sublevels for user accounts. The backslashes are just a way to tell the UNIX shell to ignore the following carriage return and to treat what comes after it as if it’s all one one line.

5.4 Testing the LDAP Helper

Once you think you have the command line switches for the LDAP helper correct for your network, it’s time to test it. (You did test the LDAP server earlier, right?)

Run the LDAP helper with the command line switches that you believe are correct. If it doesn’t start up, read error message and correct the problem. Once you’ve gotten to where it will start up, you can attempt to do a few sample queries to see if it works. The LDAP helper, once running, will wait for your input in a loop over and over again. What it is expecting is a username and a password, with a space between then. If the password is correct it will return “OK”, otherwise it will return “ERR”. (Squid looks at the the output from the helper for these two return codes.) Press Ctrl-C to make the LDAP helper quit.

A sample run from our Squid server looks like this:

```
mwhite@rover:~> ./squid_ldap_auth \
-b o=MRLS \
-u cn \
-s sub \
-D cn=ApacheLDAP,ou=CO,o=MRLS \
-w indians \
-f "(&(&(objectClass=person)(cn=%s)) \
      (groupMembership=cn=InternetUsers,o=MRLS))" \
-h ldap-server.madriver.k12.oh.us
```

```
mwhite horse
```

³No, that’s not actually the password. What do you think I am? Stupid? :)

```
ERR
mwhite 12345
ERR
mwhite rabbit
OK
```

You can see that I tried to guess my password twice (with “horse” and “12345”), and the helper told me “ERR”, meaning that the password is incorrect. On the third try I used “rabbit” as my password, and was told that the password is correct.

Keep adjusting the configuration settings until the LDAP helper is working from the command line for you. *If you cannot get the LDAP helper to work from the command line, then Squid won't be able to either.*

5.5 Moving the LDAP Helper

Once you get the LDAP Helper configured correctly, congratulations! The hard part of the installation is done. The next step is to copy the LDAP Helper from the directory where you compiled it into a suitable location.

You can copy the LDAP Helper anywhere you'd like, but I'd recommend that you keep it in the libexec directory under the directory where you installed Squid. (On my server that's /usr/squid/libexec.) Just copy the LDAP Helper to where you'd like for it to go, and you're finished.

6 Configuring Squid

Once the LDAP Helper is working, configuring Squid itself is a fairly simple task. There are two sections to the squid.conf file that you will want to change – telling it to use the LDAP Helper for authentication, and writing an ACL⁴ forcing all clients to be authenticated before they are allowed access to the proxy.

6.1 Instructing Squid to Use the LDAP Helper

There are three parameters you need to set in the squid.conf file to have Squid use the LDAP Helper:

There is one “gotcha” on the “auth_param basic program” line: *It must be all on one line!* (This cost me several hours of work the first time...)

⁴Access Control List

auth_param basic program	This is the name of the program that Squid should call for authentication, as well as all the parameters needed to start the program. You will need the information from the previous section here.
auth_param basic children	The number of copies that Squid should start of the above program. It will pick an idle running copy when it needs to authenticate a user, so you want to ensure there are enough running copies of the LDAP Helper that it will not have to wait.
auth_param basic realm	The realm name that you want Squid to present to the client when it goes to authenticate.

Table 2: Squid's squid.conf Authentication Parameters

On our Squid server, this section of the squid.conf file looks like this:

```
# Authentication Stuff
auth_param basic program /usr/squid/libexec/squid_ldap_auth
  -b o=MRLS \
  -u cn \
  -s sub \
  -D cn=ApacheLDAP,ou=CO,o=MRLS \
  -w indians \
  -f "(&(&(objectClass=person)(cn=%s)) \
      (groupMembership=cn=InternetUsers,o=MRLS))" \
  -h ldap-server.madriver.k12.oh.us
auth_param basic children 15
auth_param basic realm MRLS Internet Gateway
```

The “auth_param basic program” line is actually on one line, but I am unable to print it that way in this document.

6.2 Create a Basic ACL

Here is a minimal ACL that will work for configuring Squid to have it authenticate users via the LDAP Helper:

```
acl all src 0.0.0.0/0.0.0.0
acl password proxy_auth REQUIRED
```

```
http_access allow password all
http_access deny all
```

The first line creates an ACL that covers the entire network called “all”. The second line creates an ACL that redirects users to the LDAP Helper. The third line tells Squid to allow anyone to use the proxy after they are able to pass the password check, and the fourth line tells Squid to not allow anyone else access to the proxy.

This is by no means a very good ACL, only a small one that will work in a pinch. The ACLs on our Squid server are actually quite involved, and only allow people from within our school buildings access to the proxy.

7 Wrapping It All Up

To summarize, the steps involved to setup eDirectory and Squid are:

1. Setup your tree for Squid. This involves creating a user for Squid to log into your tree as, and giving that user rights to scan the areas of your tree where user accounts are.
2. Building the LDAP Helper for Squid.
3. Configuring the LDAP Helper.
4. Telling Squid to use the LDAP Helper for authentication.

Using Squid, and LDAP via eDirectory can be a very powerful combination. I hope this document has been helpful to you.

Any comments, good or bad, about this procedure may be directed via e-mail to mwhite@madriver.k12.oh.us.