

Directory Services: Part 2

This month's OPEN For Business! is the second in our LDAP mini-series. The series is about using directory services to make the administration of your network easier, and potentially, how to replace every bit of proprietary software you may have in the network with better Open Source software, plus how to tie it all together. Last month we covered what a directory is, and hinted at what you could do with it. This month we will look at the creation of a single user account base and single sign-on initially across all of your non-Windows systems (Linux, Solaris, MacOSX, FreeBSD, HP-UX, etc).

But first you need your LDAP directory!

The Open Source world is blessed with the simply fantastic OpenLDAP project. Led by Kurt Zeilenga, the author of most of the RFCs that actually define what LDAP is and how it works, the OpenLDAP server and related tools are one of our best-kept secrets. So it's about time we corrected that...

The OpenLDAP server is a full implementation of the LDAP definition, and has such a vast range of features that are soooo technically wonderful I could go on and on and on for ever. Let's just say it's lightning fast, superbly robust, and does simply everything you could ever need a directory server to do - but without the simply obscene price tag the proprietary vendors seem to enjoy attaching to these things. Of course it's Open Source too, which gives it transparency, extensibility, unlimited customisation potential, and it's guaranteed to never deviate from LDAP's defining standards - not like some other Directory products I could actively mention!

So we've got our OpenLDAP server. Let's get on with populating it!

As I mentioned last month, an LDAP Directory enables information about objects to be held in a tree-like organisational structure (a quick tip at this point - the structure you create to hold your users can be anything you like, we recommend you create one that reflects your actual organisational structure). You can then associate key pieces of information with these objects

Time to make this real. The objects that we are interested in creating are users. If I asked you what your users consist of in IT terms, you'd probably tell me the standard stuff - username, password, home directory, access to this, denied access to that, email account with an email address, etc (funny old view we have of people, right?). Take some time right now, and you'll see that you can soon write a list of stuff defining what a user is on your network. So if we create an object in our directory that represents a user from the IT perspective, the attributes this object must have are the ones that are meaningful in an IT sense - and you've just listed them.

All this stuff is well understood, and well thought through. A central notion in the LDAP world is that of 'schemas'. You've probably noticed by now that I like to explain complex stuff in simple terms. I'm aware that there's a million and one subtleties, and my descriptions are not 'precise' in absolute terms, but we'll leave that argument to the pedants, my descriptions are good enough for where we're going. A schema is a big batch of definitions of well understood attributes, things like telephone numbers, email addresses and passwords. By registering a schema with your LDAP server you can then associate its definitions with your objects. Simple, right?

So we start by adding some schemas that represent people (our somewhat odd IT notion of what people look like that is!) and also that represent an account on a UNIX box. You now have a basic directory service, and you're ready to actually do something with it.

Now we need to move on to the boxes you want to authenticate to your shiny new Directory server. As mentioned, we're going to start with your UNIX, GNU/Linux and BSD machines. By default, these machines have a local account database (/etc/passwd) - a legacy from the early days of UNIX. This database is sometimes exactly what you need, but the truth is, in most cases it's a nightmare to manage. Imagine a network of 500 servers, each with its own user database, each of which needs visiting every time a change happens in your user population. Yuck! Traditionally UNIX solved this with systems like NIS, but this is not truly cross-platform in any trivial-to-implement way, and we want to do waaay better than that, especially when we bring our Macs and Windows boxes online!

The answer is Pluggable Authentication Modules, affectionately known as PAM. The notion is simple... Abstract the authentication layer so that your UNIX system can be authenticated by any method which has a PAM module. PAM makes the method look like traditional UNIX authentication. There are plenty of them, including MySQL, Radius, Kerberos and, for our purposes, LDAP. Simply better software:

Once your LDAP PAM is plugged into your machine, you can authenticate the underlying operating system against your new Directory. One account base across all your UNIX and GNU/Linux servers - how cool is that? Not only that, but FreeBSD (the world's finest web server!) has had PAM capabilities since release 5, and since MacOSX is based on a

BSD core, it can participate in this scheme too. There's only one significant OS missing from your new account management paradise... It's OK. Don't worry, we're going to hit them next month! I'll show you how to extend your new Directory to enable logons to your Windows servers and desktops. These logons will be with exactly the same accounts you've just set up to enable logons to your UNIX, GNU/Linux and BSD machines. Single identity across all of your platforms - now won't that be nice?

We'll do all this without you having to buy one of those ridiculously expensive directory solutions from one of the proprietary vendors. It'll do it better. And it's all because it's simply better software.