

Active Directory and Linux

David "Del" Elson 2002-04-03

Introduction

This article discusses the use of Microsoft's Active Directory as an authentication service for Linux systems. Although Linux has a perfectly good directory based authentication system (OpenLDAP), it may be desirable on some sites to authenticate Linux users against a Microsoft Windows 2000 server.

Although this article discusses Linux (because that is the system I have available in my office), this authentication mechanism works well against other Unix systems that have a PAM/NSS mechanism. Currently that includes Solaris, although discussion has taken place on the possibility of getting this to work on HP-UX. Since most of the work is done at the Windows 2000 end, the instructions for getting this to work on Solaris are not too different from what I have described here.

Before attempting any of this, you should be familiar with the concepts of PAM and NSS, and be familiar with how to set up PAM and NSS on your Linux distribution or operating system.

Also, before I begin, I should state that I really do not believe that Active Directory is the best way to authenticate Linux clients, nor is it the best way to authenticate users in a multi platform environment. It may be the case that in your network, for political and financial reasons, you are constrained to using Active Directory as a directory product, however, and so this document may be of some benefit to you if you are unable to rethink your IT strategy in this area.

Furthermore, I really recommend taking caution in implementing any of the instructions in this article. Microsoft does not support authenticating to Active Directory for any non-Microsoft client, and Microsoft does not support the instructions contained in this article in any way. If you wish to follow the instructions in this article then you do so at your own risk.

Authentication Systems

The traditional way of authenticating users on a Linux system is to store the accounts and passwords in an `/etc/passwd` file or, more commonly, a combination of `/etc/passwd` and `/etc/`

shadow.

This is fine when there is only a single Linux system on a network, as the `/etc/passwd` file is local to the system on which it is stored. Although it is possible to share a single `/etc/passwd` file amongst multiple Linux systems, using such tools as `rsync`, it makes more sense to have a centralized authentication database containing all of the user accounts and passwords.

There are many such centralized authentication systems available, including Kerberos, NIS, and others. LDAP is a protocol that can be used to enable a fairly powerful mechanism to store centralized authentication as well as user information.

Cross-Platform Authentication

The aim of cross-platform authentication is to have a single, centralized password database that can be used to authenticate users on both Unix, Windows, and perhaps even other systems such as Macintosh or NetWare.

Because LDAP-based authentication is supported on the most recent Microsoft systems, including Windows 2000 and XP, and is also supported on Linux and other Unix systems (such as Solaris), it makes an excellent choice for a cross-platform authentication system. Note that there are limitations to this. Firstly, the Microsoft clients for Windows 2000 and XP are specific to authenticating against a Microsoft Active Directory server. Although OpenLDAP uses the same LDAP protocol, there are other features of Active Directory (including a modified version of Kerberos with a Microsoft specific mechanism called a "PAC") which means that Active Directory clients will not necessarily be able to authenticate against OpenLDAP.

The second limitation of Active Directory is that clients for it (in LDAP mode, that is) are only available on Windows 2000 and Windows XP. Although Active Directory will work in a "down-level" mode to support older Microsoft clients, users of systems such as Windows NT and Windows ME/98/95 who wish to have full LDAP/Kerberos based authentication support are forced to upgrade.

Of course, another limitation of Active Directory is that it only runs on Microsoft Windows 2000 servers. There are no releases of Active Directory for other platforms. In contrast, many of the other directory services available (such as those from iPlanet and Novell) are supported as servers on many platforms.

LDAP Alternatives

Other alternatives to Active Directory exist where an LDAP-based authentication system is desired. These include:

- OpenLDAP. As mentioned earlier, this is an excellent authentication system for Linux clients; however, Microsoft clients will not be able to authenticate to it. An article on implementing OpenLDAP in a Linux environment is available here: <http://online.securityfocus.com/infocus/1427>
- iPlanet Directory Service. iPlanet directory service runs on Windows platforms, as well as on Linux and Solaris systems. Although the iPlanet directory server contains a Windows NT to LDAP password synchronisation system, direct authentication to iPlanet directory server is not possible from Windows systems.
- NDS. Novell's directory service is probably the most credible implementation of a cross-platform directory service. The server runs on Solaris, Linux, Windows, and NetWare. Clients are available for many platforms including nearly all Microsoft Windows platforms (including Windows 98 and Windows NT), and also Linux and Solaris. NDS is standards-compliant and I have found it to perform well in many implementations. If a true cross-platform directory service is required, then NDS is probably the best option. NDS's only limitation is the cost, which can be quite high in an environment with a large user base.

MKS AD4Unix

What is AD4Unix?

MKS AD4Unix is a plug-in extension for Microsoft's Active Directory Server, that enables Unix-related authentication and user information to be stored in Active Directory. AD4Unix includes a schema update, and an extension to Microsoft's User & Group manager (part of the Active Directory administration interface, which is in turn part of the Microsoft Management Console).

The primary goal of AD4Unix is to create a unified account database for Windows and Unix servers via Active Directory. This is what specifically enables cross-platform authentication using Active Directory.

AD4Unix was written by Maxim Batourine of the Faculty of Architecture, Landscape, and Design at the University of Toronto. Full credit should be given to Maxim for the assistance that I

received during preparation of this article.

Obtaining MKS AD4Unix

MKS AD4Unix can be obtained via the [AD4Unix download page](#).

AD4Unix is delivered as a single .MSI (Microsoft Installer) file that can be installed directly onto a Windows 2000 server.

Installing MKS AD4Unix

The original installation instructions for AD4Unix, and guidelines for its use were written by JJ Streicher-Bremer for AD4Unix 1.1.1. Things have changed somewhat since then, there is now an installation package (MSI format), however there are still a few hairy parts of the installation because the installation package is not perfect. JJ has also been particularly helpful in preparing this article, both personally and in discussions and posts to the nss_ldap mailing list.

My goal in installing AD4Unix was to take a machine with a blank, formatted, hard disk, and install Windows 2000 and then AD4Unix from scratch. This took a bit of messing about.

These instructions are based on Windows 2000 server, service pack 2, and AD4Unix version 1.5. Here follows a log of what I did to get the product up and running:

- Installed Windows 2000, from the installation CD. Since I had a single empty hard disk, I just told Windows 2000 to install onto the hard disk and use all available disk space. If you have multiple hard disks (to use Microsoft's guidelines of having separate physical disks for directory and log files) then you will need to mess with the installation a little. For me, I just followed the step-by-step installation instructions and chose the defaults.
- Note that although I am in Australia, I chose to install only the US English language support (which was the default during the Windows 2000 installation). I had some trouble installing AD4Unix when I chose an alternative default language, and Maxim Batourine's comment was that "AD4Unix does not currently support the full zoo of languages on Windows". I can understand this limitation, and in any case it only affected the server on which I was installing Windows 2000 and not any other workstations. Maxim promised better language support in future versions of AD4Unix.
- Installed the Windows 2000 High Encryption Pack. I am not sure if this is still required,

since it appears to be installed with service pack 2 which I installed next. It did appear to make the AD4Unix installation go more smoothly when it was installed first, however. Perhaps this is another case of the DLL hell that all Microsoft Windows administrators are faced with? The installation of this pack required a reboot.

- Installed Windows 2000 Service Pack 2. Note that all Windows 2000 service packs are available from the [Microsoft Web site](#) - I won't detail the specifics of whereabouts in the site to obtain the service packs from because the site does tend to change periodically; however, they are usually to be found in the "Downloads" area. This required a reboot.
- Using the "Configure Your Server" wizard, I set up Active Directory. In this case, since it was a new installation in an isolated environment, I created a new domain, new tree, new forest of trees, and created a new DNS zone. You may wish to configure your server differently, or join it to an existing tree or forest. Beware that we are about to install schema updates, which could play hell with any existing directory tree or forest that you have, unless you install them correctly.
- Allowed schema updates on the domain controller. To do this, I followed these instructions of JJ's:
 - Open a command window (Start->Run->CMD)
 - Type the command: **regsvr32 c:\winnt\system32\schmmgmt.dll** This registers schmmgmt.dll as a MMC (Microsoft Management Console) snap in. You can now close the command window by typing exit.
 - Create a Schema Management MSC, as follows:
 - Start -> Run -> MMC
 - From the console menu, select "add/remove snapin" and then click the "Add..." button.
 - Select Active Directory Schema and click "Add"
 - click "Close"
 - click "OK"
 - Choose the domain controller you want to update the schema on:
 - Right click on "Active Directory Schema" and select "Change Domain Controller"
 - Select "Specify name" and type in the DNS name or address of your Domain controller
 - Allow updates on the domain controller
 - Right click on "Active Directory Schema" and select "Operations Master"
 - Click the checkbox labelled "The Schema may be modified on this Domain Controller"
 - Click OK
- Now it is possible to install the ADS4Unix plugin. To do this, find the location where the . MSI installer file was downloaded to, and double click on it in file manager.
- Say Yes to the questions about schema updates.

- On the Start menu under "Programs" there should now be an extra menu titled "AD4Unix". This contains the AD4Unix configuration program (MKSADPluginSettings). Run this configuration program, and set up an NIS name -- I just used babelads, which was the lower case version of my Windows NT (downlevel) domain name. It doesn't matter much what you enter in here, as you will not be using NIS, but something needs to be entered.

Adding User Entries

Now that the AD4Unix plugins are installed, it is possible to use Active Directory Users and Computers to enter new Unix users into the Active Directory system. You could also modify the Unix user and group attributes of your existing Active Directory users to make those users visible on a Unix system.

To add a new user, run the program "Active Directory Users and Computers" from the "Administrative Tools" menu. Note that you need to run this program from the same computer on which the ADS4Unix plugins were installed - if you normally manage your user base from another workstation then you will need to install the plugins there as well, perhaps this time without the schema updates.

After creating a new user, the user editor window (obtained by double clicking a user in the user list) will contain an extra tab, titled "Unix settings". This contains the following extra fields:

- NIS: Set this to the NIS domain you created in the configuration program.
- UID: The numeric UNIX user ID of this user.
- GID: The numeric UNIX group ID of this user.
- Description: This replaces the "comment" field in the /etc/passwd file.
- Home folder: This is the user's Unix home directory.
- Shell: This should be set to something useful, e.g.: /bin/bash for a user where interactive logins are required, or perhaps /bin/false for a user who is not allowed an interactive login session.

Note that these fields replicate the information in the /etc/passwd file normally found on a Unix system.

Group Entries

For Active Directory groups, there will now also be a "Unix settings" tab in the Active Directory Users and Groups tool. This tab contains two fields:

- group: The symbolic UNIX group name for this group.
- GID: The numeric UNIX group ID for this group.

Adding a user to a UNIX group is again done via the Active Directory Users and Groups tool, simply by the same method that you would use to add an Active Directory user account to an Active Directory group. Group membership on Linux or Unix mirrors the membership on Active Directory.

Other Components

Now that the Active Directory (server) end of things is configured, we need to configure a Linux box to act as an Active Directory client. This includes setting up a few pieces of software, including:

- OpenLDAP (specifically the client side of OpenLDAP).
- NSS_LDAP and PAM_LDAP.

OpenLDAP client

The OpenLDAP client can be obtained in a number of ways. This includes:

- Obtaining the source code from [OpenLDAP](#).
- Installing the .RPM or .DEB files for OpenLDAP for your distribution.

Note that although we only want to have the OpenLDAP client, the OpenLDAP project includes a single source tarball containing the server and client components. If you prefer to build from the source code then you will need to obtain the latest `openldap.tar.gz` file from the OpenLDAP download page, and read the installation instructions included.

The client components that we are concerned with include the shared libraries and the `ldapsearch` program, which we will use for testing. If you are going to compile `NSS_LDAP` and `PAM_LDAP` from source as well, then you will need the OpenLDAP header files. All of these files should be installed when you run "make install" from the OpenLDAP source directory.

If you have a Red Hat system then you may just want to install the OpenLDAP RPMs. In Red Hat this is the `openldap` and `openldap-client` RPM files. You should use a recent version of OpenLDAP, for example release 2.0.21 from the latest Red Hat 7.2 updates. Note that you do not need the `openldap-server` RPM.

In any case you will need a version of OpenLDAP that is more recent than 2.0.12 and you should certainly not use any of the release 1.x versions of OpenLDAP.

If you are installing on a Red Hat Linux system, then you will find the OpenLDAP client configuration file installed on your system as `/etc/openldap/ldap.conf`. You should edit this file in a text editor (such as `vi` or `emacs`) and include the following two lines:

```
HOST  
BASE
```

Where `""` is the DNS name or IP address of your Active Directory server (I prefer to use the IP address, in case of DNS failure, at least then you will be able to find your LDAP server), and `""` is your LDAP search base set up by Active Directory. This would have been entered by you when you set up Active Directory using the wizard, and is probably a string like `"dc=yourcompany,dc=com"` or similar.

Once you have configured your `ldap.conf` file, then you should be able to perform a simple LDAP query using the `ldapsearch` program. For example:

```
ldapsearch -x ""
```

Note the empty string in quotes at the end of the command line. On running this search command you should see some interesting if not particularly informative details.

One of the interesting quirks of Active Directory is that it doesn't allow searches below the top level of the directory while anonymously bound (not authenticated) to the directory. So, for example, the following search will fail:

```
ldapsearch -x "sAMAccountName=del"
```

whereas, once bound (for example, as administrator), the command will succeed:

```
ldapsearch -x -D "cn=Administrator,cn=Users,dc=somecompany,dc=com" -  
W/  
"sAMAccountName=del"
```

(the above command will prompt you for a password - you must supply the Administrator password for your Active Directory).

Note that if your base DN for Active Directory is "dc=somecompany,dc=com" then your default Administration DN will be "cn=Administrator,cn=Users,dc=somecompany,dc=com". This may have changed if you have modified your Active Directory installation, so check this before using it in the above LDAP query.

You should get your OpenLDAP client working before progressing any further. If you cannot perform at least simple top level searches on your Active Directory, then you should locate and fix any problems before attempting the next step.

NSS_LDAP

On the Linux side, the most important components are `nss_ldap` and `pam_ldap`. These modules are both available from [PADL Software](#). The modules are also shipped with most modern Linux distributions including Red Hat.

If you want to obtain the modules from source code, then you will need to download them from the "Software" page at the PADL web site, above. You will need both `pam_ldap` (available as `pam_ldap.tgz`) and `nss_ldap` (`nss_ldap.tgz`).

If you recompile from source, get the latest (183 or thereabouts as of the current writing) version of `nss_ldap`, and make sure you use the following two options on the `./configure` line when compiling: .

```
/configure --enable-schema-mapping --enable-rfc2307bi
```

The `--enable-rfc2307bis` flag is required for any `nss_ldap` version after 172 (Red Hat 7.2 uses 172 so this flag is not needed).

If you prefer to install from `.DEB` or `.RPM` files, you will need to make sure that the `nss_ldap` and `pam_ldap` modules are installed on your system. Note that in Red Hat Linux, both

pam_ldap and nss_ldap are stored in the same RPM module -- nss_ldap-172-2.i386.rpm.

You will, however, need to rebuild the nss_ldap RPM, as follows.

Recompiling the NSS_LDAP RPM file

The nss_ldap-172-2.i386.rpm file supplied by Red Hat does not include a particular flag that is required to allow nss_ldap to work against the Active Directory schema (which is different to the OpenLDAP schema normally used on Red Hat Linux systems). The best way to enable this flag is to recompile the nss_ldap RPM file.

Firstly, obtain the nss_ldap-172-2.src.rpm file. This can be obtained either from your Red Hat CD set (if you have the full CD set with the source CD), or from the Red Hat FTP site (<ftp://ftp.redhat.com/>) or one of its nearby mirrors. The file you are looking for will be in the `redhat-7.2/en/os/i386/SRPMS/` directory on the FTP server.

Install the source RPM file using:

```
rpm -ihv nss_ldap-172.i386.src.rpm
```

You will then need to edit the file `/usr/src/redhat/SPECS/nss_ldap.spec`. Around line 67 of the file, you will find the configure command, which reads:

```
%configure --with-ldap=openldap --libdir=/lib
```

You will need to change this to read:

```
%configure --with-ldap=openldap --libdir=/lib --enable-schema-mapping
```

Also, near the top of the file you will find the line:

```
Release: 2
```

Change this to:

```
Release: 3
```

You can then rebuild the RPM using:

```
cd /usr/src/redhat/SPECS
rpm -ba --clean nss_ldap.spec
```

After the build process finishes, you will have a file in /usr/src/redhat/RPMS/i386 called nss_ldap-172-3.i386.rpm. (Note: this is for an Intel / AMD based system -- on a SPARC system you will be looking for a file called nss_ldap-172-3.sparc.rpm). You should install this file using the following command:

```
rpm -Uhv  usr/src/redhat/RPMS/i386/nss_ldap-172-3.i386.rpm
```

Note that the --enable-rfc2307bis flag on the %configure line is also required for any nss_ldap version after 172. The release of NSS_LDAP shipped with Red Hat 7.2 is version 172 so this flag is not needed, but if you obtain a more recent RPM then you will need the following configure line instead of the one shown above:

```
%configure --with-ldap=openldap --libdir=/lib
--enable-schema-mapping --enable-rfc2307bis
```

NSS_LDAP configuration

The configuration file for nss_ldap is the file /etc/ldap.conf. This will normally have been created on your system when you installed the nss_ldap module.

Once you have installed nss_ldap and pam_ldap, you will need to edit the /etc/ldap.conf file, as follows.

Authconfig

Red Hat Linux comes with an authentication configuration program called authconfig, which you can use to perform the basic configuration to allow your system to use LDAP authentication. To run this, log on as root and run:

```
authconfig
```

On the first screen, select "Use LDAP". Enter the IP address of the LDAP server (which is your Windows 2000 Active Directory) and the base DN that you entered when you set up Active Directory (e.g.: dc=yourcompany,dc=com). Also make sure that "Use LDAP Authentication" is checked (next page).

Authconfig is intended to set up a system to authenticate to an OpenLDAP server, and therefore doesn't perform all of the functions needed to set up your system to authenticate against Active Directory. To complete the configuration, you will need to edit your `/etc/ldap.conf` file, which is the primary configuration file for OpenLDAP.

You should also uncomment the following lines, which should be in your basic `LDAP.conf` file, but commented out:

```
nss_base_passwd cn=Users,?sub
nss_base_shadow cn=Users,?sub
nss_map_objectclass posixAccount User
nss_map_objectclass shadowAccount User
nss_map_attribute uid sAMAccountName
# nss_map_attribute userPassword msSFUPassword
nss_map_attribute homeDirectory msSFUHomeDirectory
nss_map_objectclass posixGroup Group
nss_map_attribute uniqueMember member
nss_map_attribute cn sAMAccountName
pam_login_attribute sAMAccountName
pam_filter objectclass=user
pam_password ad
```

Note that in the above configuration file entries, `cn=Users,?sub` should be replaced by the Base DN that was created when you installed Active Directory. You will have used this several times before by now, for example in your `/etc/openldap/ldap.conf` file.

Allowing Anonymous Searches in Active Directory

At this stage, everything should nearly be working. You should be able to log in as a user in your Active Directory domain, using the Active Directory password, and mostly get logged in. You will see some error messages, however, including:

```
id: cannot find name for user ID 1001
id: cannot find name for group ID 1000
```

This is because your Active Directory cannot be searched anonymously. There are two solutions for this problem:

1. Enable anonymous searches on your Active Directory

2. Insert an administrator DN and password into /etc/ldap.conf

To enable anonymous searches on your Active Directory, follow these steps:

- On your Windows 2000 Active Directory server, run the Active Directory Users and Groups administration tool.
- Select the top level of the directory from the tree view in the left hand panel, and right click. A menu will appear. Select the first item, which should be "Delegate Control..."
- Click "Next"
- In the next window, titled "Users or Groups", click "Add ..."
- In the next list, select "ANONYMOUS LOGON" and click "Add". You may also need to select "Everyone" and the "Guests" group, depending on how you have Active Directory configured. Click OK when this is done.
- Click "Next"
- Select "Create a custom task to delegate" and click "Next".
- Click "Next"
- In the next list, select "Read". "Read All Properties" will be selected at the same time. Click "Next" when this is done.
- Click "Finish".

Inserting an Administration DN into /etc/ldap.conf

An alternative to allowing anonymous searches on your Active Directory is to allow the nss_ldap routines to bind as an administrator DN to your directory and perform searches in privileged mode. To do this, insert the following lines in your /etc/ldap.conf file:

```
binddn cn=Administrator,cn=Users,  
bindpw
```

You should be used to the "" thing by now.

WARNING: The above example shows that the administrator user name and password have been coded in clear text in the /etc/ldap.conf file! Unfortunately, this file must always remain world-readable, because otherwise users logged on to the system will not be able to read data from the directory. You should not do this on a system where any user has shell access to your system, or can in any other way read this file.

Of course, there is a third alternative, which is to create a new user within Active Directory and assign this user no rights other than read access to the directory. You could then key this user's logon and password into the `/etc/ldap.conf` file, rather than the administration DN and password. I don't consider this method any safer than allowing anonymous binds; however, as any user that could (anonymously perhaps) read this file could read the logon name and password of the special DN that you have created.

PAM_LDAP

On a Red Hat system, PAM_LDAP is part of the NSS_LDAP RPM. PAM_LDAP by default uses the same configuration file (`/etc/ldap.conf`), so all of the `nss_ldap` changes work also for `pam_ldap`

Note that PAM_LDAP is only used for authentication, whereas NSS_LDAP is used for all user information. A user can still log on to a Linux system without NSS_LDAP working, although they will get frequent messages saying "I have no name!". If this is happening to you, then it is possible that you have PAM_LDAP working correctly but NSS_LDAP broken.

I have had it reported to me that on some recent Debian systems, PAM_LDAP and NSS_LDAP have different configuration files. I am not entirely convinced of the wisdom of this, but it could be useful in certain situations, and for debugging purposes. If you are on such a system, it would be wise to locate all files containing the name "ldap.conf" on your system and work out which of these belong to PAM_LDAP and which belong to NSS_LDAP.

Note that to be able to change passwords on an Active Directory server from Linux, you need to have SSL and TLS enabled, both at the client end and at the server end. Enabling SSL and TLS on an LDAP client is covered in the SecurityFocus article, [Linux Authentication Using OpenLDAP, Part One](#).

Web Resources

There are many resources on the Internet that can assist you in getting all of this mess running on your network.

The Active Directory documentation, supplied by Microsoft (full documentation is supplied with every copy of Windows 2000 Server) contains all of the information you are ever likely to need about Active Directory. It contains a complete installation guide, a full set of details about the

schema management utilities, and extensions to the [RFC 2307](#) schema that have been implemented by Microsoft along with a complete rationale. Guides to extending the Active Directory schema are included, along with a helpful set of utilities allowing you to use all manner of third party clients in an Active Directory environment. Please note that everything that I have said in this paragraph is basically incorrect and largely tongue-in-cheek.

The OpenLDAP consortium page, which is to be found at <http://www.openldap.org/> contains much useful documentation about OpenLDAP and LDAP in general.

Innosoft's LDAP world page, to be found at <http://www.innosoft.com/ldapworld/> is a useful jumping off point for many LDAP-related topics.

The home page for MKS AD4Unix can be found at <http://www.css-solutions.ca/ad4unix/>

PADL, a company based in Melbourne, Australia, makes the `nss_ldap` and `pam_ldap` modules available for free. They also offer a product called `ypldapd` which can make LDAP servers visible in a NIS environment, for Unix systems that do not support LDAP based authentication or name services. This could be an exceptionally useful extension to any cross-platform LDAP service especially in a Unix environment where PAM and NSS are not supported. Their home page can be found at <http://www.padl.com/>

Quartet, is a new project, which attempts to implement much of the functionality of Active Directory on Linux. The Wiki Wiki Web page for Quartet can be found at <http://www.babel.com.au/quartet/>, although it should be noted that no actual code has been written for this project yet and it may be some time before it is ever useful.

Everyone who is working in a multi platform Unix and Windows environment should be familiar with the SAMBA project. SAMBA version 3.0 can join an Active Directory domain. Work is progressing on getting user information for SAMBA stored in LDAP. The SAMBA home page can be found at <http://www.samba.org/>

[Privacy Statement](#)

Copyright 2006, SecurityFocus