A Planning Guide for Moving to LDAP as Naming Service in the Solaris™ OS

Wajih Ahmed and Abdi Mohammadi

April 2009

Sun Microsystems, Inc.

Table of Contents

Introduction

There are several important decisions for organizations preparing to migrate from Network Information Service (NIS) or NIS+ to Lightweight Directory Access Protocol (LDAP). Similar yet simpler challenges are experienced by organizations that are embracing LDAP as a naming service for the first time.

The goal of this document is to identify the majority of these issues and help readers make informed decisions with respect to their specific environments. This paper specifically deals with migration design issues, but there is no reason why lessons cannot be gleaned from it for green-field implementations.

One aspect of a migration is the heterogeneous nature of various UNIX® LDAP clients. *UNIX LDAP client* is a term used to describe the pieces of the operating system that make up the libraries and binaries required to connect to a LDAP server for authentication and naming services.

A typical organization has a mix of UNIX LDAP clients for major operating systems, such as the Solaris™ Operating System, Red Hat Linux, AIX, and HP-UX. A prerequisite for these clients is RFC 2307 and LDAP v3 compliance.

> **Note**: This document applies to Solaris 8 and later releases.

On the server side, the Solaris client is designed to work optimally with Sun Java™ System Directory Server Enterprise Edition (hereafter call Directory Server). However, it can also work with other LDAP v3 compliant directory servers, such as Microsoft Active Directory, eDirectory, and OpenLDAP.

The Solaris client also requires RFC 2307bis schema. In addition, Sun Java System Identity Synchronization for Windows software can be used to synchronize user names and passwords (and for that matter, other attributes) with Microsoft Active Directory (AD), thus providing the flexibility to run parallel yet synchronized directory services.

The Challenges of Moving to LDAP

One of the biggest impact in migrations is the "folding" of the NIS/NIS+ domains into the LDAP namespace. A typical organization can have several NIS/NIS+ domains, which could span geographic boundaries.

And it is quite common to have `uid` numbers that are nonunique across these domains. This nonunique pattern also can be true for other databases, such as groups, netgroups, and so on. LDAP, on the other hand, has a tradition of having a flat namespace where the distinguished name (DN) is unique. So mapping to this new namespace can be a formidable challenge for organizations that have a large and complex NIS/NIS+ environment.

In addition to the namespace, the delegated administration model of NIS/NIS+ domains needs to be kept in mind when migrating to LDAP. There are several approaches to these challenges, and the design of the Directory Information Tree (DIT) plays a vital role in choosing the correct solution. Hence, there are several sections in this document that discuss the DIT design.

Preparing a DIT

The Solaris 8 OS and later releases provide a script called `idsconfig`(1M) to prepare a fresh Directory Server instance for using LDAP as a naming service. Figure 1 shows how a fresh DIT looks after `idsconfig` has been run on a root suffix.

Basically `idsconfig` creates the containers for the naming services databases, and adds additional schema (not covered by RFC 2307bis), a default profile, a proxy user, associated Access Control Information (ACI), and the required indexes. Note that the latest versions of `idsconfig` might add additional LDAP containers for subsequent Solaris 10 updates.

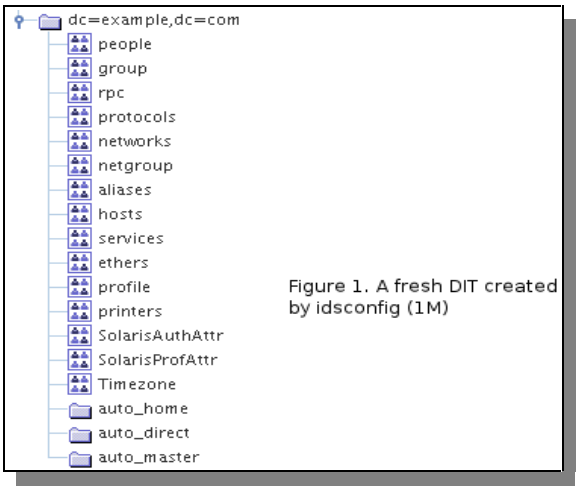

Figure 1: Example of a Fresh Directory Information Tree

DIT Design

As a first decision point, there are typically two paths that can be taken in a migration. One path is to perform a one-to-one mapping of the NIS/NIS+ domains directly into a container of their own (typically `ou`) under the root suffix of the directory, as shown in Figure 2. This is the *least disruptive* approach because it does not require any changes to the user IDs (uids), and hence implementation time is also *shorter*.

Each container is thus self-contained and, in particular, it has its own people branch under which the users reside. The search base for the LDAP client refers to each of these branch points, which would correspond to the (former) NIS domains. Additional ACI might need to be added to give the former domain owners rights to write to these branches, thus allowing them to manage their own branches. Because the uids can be nonunique, it is important to configure the Directory Server uid uniqueness plug-in so that it does not provide uniqueness for uids.

Note that if you want different home directories for your users in each of the domains, this one-to-one DIT design is the only option.

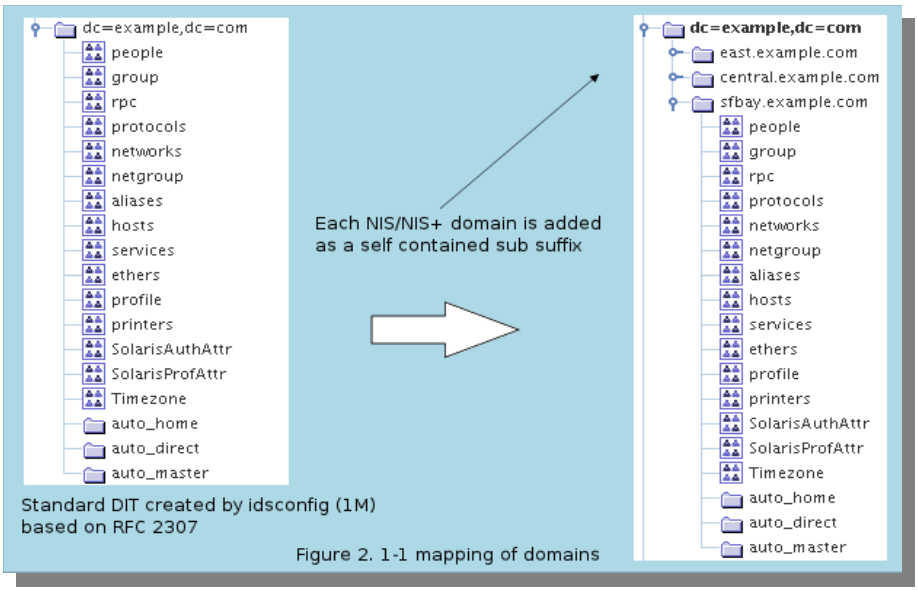

Figure 2: One-to-One Mapping of Domains

Another approach is to make all LDAP uids unique. (Many organizations these days are creating uids that are a combination of the employees' initials and their employee IDs.) This method is actually easier said than done, because it requires uniquely consolidating existing uids and, consequently, the uid number.

Similarly, groups should also be consolidated and, consequently, effect the group ID (gid) number too. The side effect of this consolidation is the daunting task of changing the uid number and possibly gid number on all file systems, although if planned, this can be automated through carefully written scripts.

It is worth noting here that virtual directory, which is a feature of the Java™ System Directory Proxy Server (hereafter "Directory Proxy Server"), also can be used to create a virtual uid attribute derived out of existing attributes. See the "Using Directory Proxy as a Virtual Directory Solution" section for more details.

In Figure 3, there is still a one-to-one mapping between the domain names and the LDAP containers. What is different here is that a single unified `ou=people` branch is used underneath the root suffix. Note that `ou=group` is not shown in Figure 3 at the same level as `ou=people`. Ideally, if `ou=people` is centralized, `ou=group` also would be centralized.

However, standardizing on the system group names and number and placing them under each "domain" allows the flexibility of delegated administration. In the end, the decision to centralize groups has to be carefully taken during the design phase of the migration. This "hybrid" approach keeps the benefit of domain delegation, but on the other hand, it facilitates user account centralization and enforces unique uids and, optionally, gids.

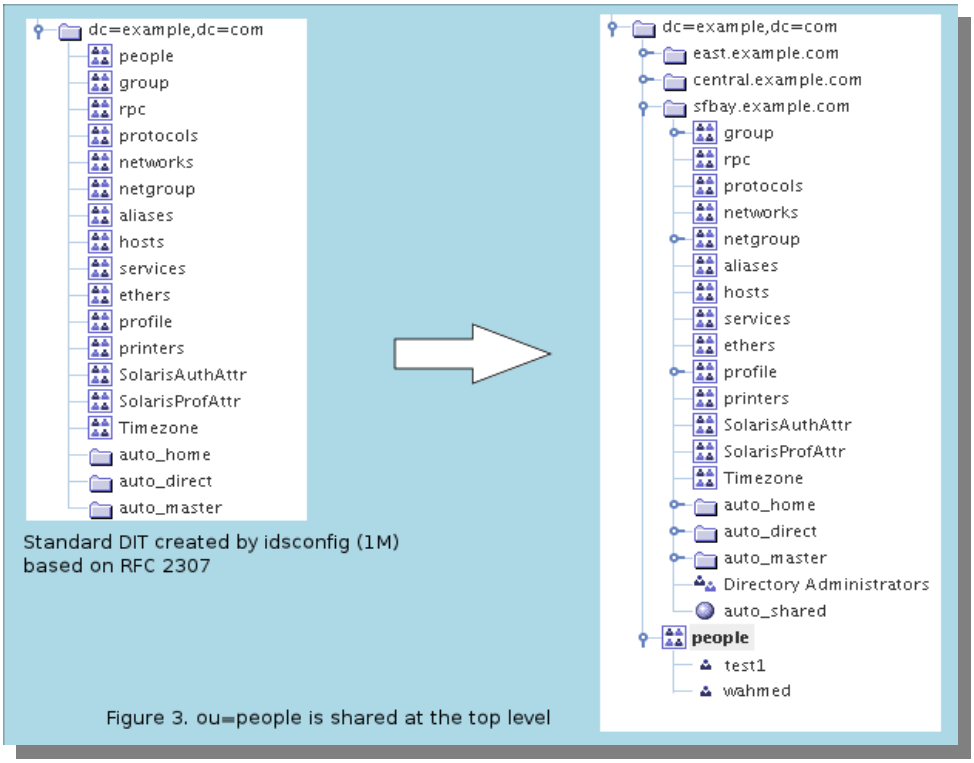

Figure 3: Hybrid DIT Approach

As with any delegated administration approach, proper ACI is needed to assign rights to users or groups that can write to their assigned branch points on the DIT. Note that each "domain container," such as `east.example.com`, requires the `nisDomainObject` object class and the `nisDomain` attribute set to the appropriate value.

There are some use cases that could exhibit side effects from this "hybrid" approach. For example, the user's shell and path (`loginShell` attribute) need to be the same on all workstations. This might require standardization on the location of the shells on all UNIX workstations. If that is not easily done, then an alternate approach can be devised, such as creating profiles that are friendly to each flavor of UNIX, and then defining attribute maps in these profiles to point to an alternate attribute for the shell using attribute mapping. Attribute and object class mappings are discussed further in the "Existing DIT" section.

The question becomes: Why fuss so much over unique uids? Well, the biggest benefits are *compliance* and *management*. In this approach, disabling the user, deleting the user, or both is required only in one place. Having a unique uid also offers the benefit of mobility. When users travel from one location to another, their uids can remain the same. A uid should be given to a user forever regardless of status (married, changed name, and so on). Some companies, such as Sun, provide a uid (Sun ID) that is unique across all former and current employees at Sun. So if employees leave and come back later, they have the same Sun ID as before.

Another advantage of individual domains is sub-suffix replication. However, having too many databases is discouraged, because that can result in a large number of replication agreements, which could be cumbersome to manage. Therefore, create separate databases for those sub-suffixes that need to be replicated to a remote location where a local directory is desired for performance as well as bandwidth reasons.

Implementing a Hybrid DIT

This hybrid DIT design requires the use of Service Search Descriptors (SSDs) for the `passwd` and `shadow` databases, because the centralized `ou=people` container is at least one level above the default search base in the profile. See Figure 4, which shows one such example in its simplest form.

Ideally, the SSDs should be restricted to only one level and can have LDAP filters. Similarly, if deemed appropriate, you can place any other database, such as `ou=group`, outside the default search base and add an SSD for it.

The SSD for each database can also point to multiple containers. This feature might be useful during the transition phase. Other LDAP clients that are based on `nss_ldap` have similar features to SSDs and mappings, so this approach is also viable in a heterogeneous UNIX environment.

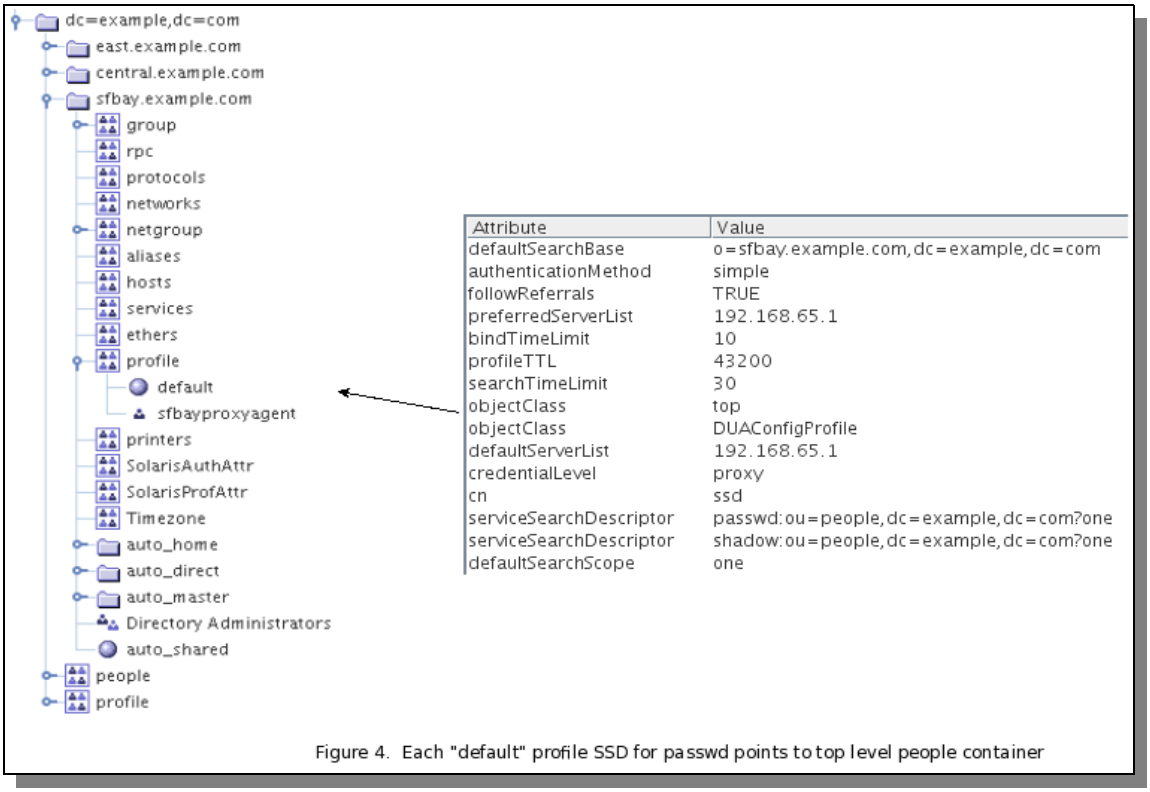

Figure 4: Simple Hybrid Implementation

Additional ACI might also be needed (the default ACI added by `idsconfig` for `proxyagent` can be used as examples). In particular, if an SSD for `passwd` is used, then the `proxyagent` under each domain needs to read the `userPassword` attribute (in case `pam_unix` is being used) and, in general, be allowed to proxy.

Obviously, the existing `proxyagent` (at the root level) can also be used; hence a "profile" container is shown in Figure 4. As a matter of fact, all the containers at the root level can be left "intact." They have been deleted in the figures just for the sake of clarity. Note that in both cases (profiles at the top level or within each domain container), the contents of `/var/ldap/ldap_client_file` is exactly the same.

Another alternative to using SSDs could be LDAP referrals. However, because referrals cause an extra hop and are sometimes cumbersome to manage, this method is not recommended.

Note that our testing revealed that using SSDs invalidates some of the Virtual List View (VLV) indexes defined by `idsconfig`, because they are defined with a `vlvbase` of the root suffix. For example, if an SSD for `passwd` is used, then a `getent passwd` would give an unindexed search. The requirement is to define similar VLV indexes at the correct search base of the SSD.

Other DIT Designs

There are several other combinations of the DIT design that might be more appropriate for an organization. For example, the DIT can be made totally flat so that there is only one container for each database. Thus, all the domains can be folded into a single flat namespace.

There are, however, side effects of this approach when trying to implement the same pre-migration delegated administration model as with NIS/NIS+. Additional ACI has to be added to these containers to control write access.

The benefit here is that there is a flat namespace, but the disadvantage is that additional ACI has to be added and maintained and each entry has to be "tagged" with an attribute (Class of Service can also be used here) that identifies its owner so that ACI can be properly enforced. For example, this attribute could be the `locality` attribute.

The point is that the implementation using ACI is possible and there are several approaches to it. However, the added administrative burden of mandating ACI could be significant and ACI also impacts directory performance. Note that maps, such as automount maps, might no longer be able to use wildcards.

Yet another approach is to use a virtual directory as in Sun Java System Directory Proxy Server 6.x (hereafter "Directory Proxy"). The virtual directory presents a virtual view and makes the actual directory a black box. See the "Virtual Directory (Directory Proxy Server)" section for more details.

It is worth noting here that in a green-field implementation, the DIT created (by `idsconfig`) is actually flat. So a new implementation automatically inherits a flat DIT. This is actually quite suitable for small to medium implementations. Large enterprises spanning geographic boundaries usually start off with this flat DIT, but as LDAP naming services expand throughout the organization, some end up breaking up the DIT for reasons discussed earlier.

Using an Existing DIT

In previous sections, we discussed using a new DIT for LDAP as naming services, one that is created by `idsconfig` on an empty suffix. There are, however, many organizations that already have a well established LDAP infrastructure and, hence, a DIT.

If you want to use an existing DIT for naming services, then it is advisable to first install (by running `idsconfig`) on a separate fresh directory instance, test the instance, and then manually merge it into the existing DIT along with the schema and indexes. It is not implied here that `idsconfig` cannot be run on an existing DIT; however, you have more control when manual modifications are done to an existing DIT, and thus, this is a safer option.

Alternatively, a new sub-suffix or branchpoint can be created and the naming services' DIT can be moved into it. This approach allows you to keep a cleaner DIT and segregates corporate users from UNIX users. Obviously, the base suffix for each of the clients now has to be the sub-suffix, but that is easily dealt with using client profiles.

Depending on the size of your organization, using an existing directory can also have the undesired effect of inundating your directory servers with significant naming services (LDAP) traffic. So sizing also becomes a very important factor to consider when using existing directories. Your directory infrastructure might be more than enough for your corporate load, but it can be pushed really hard or even tipped over with naming services queries.

The Solaris 10 05/08 OS and later releases now use `nscd` to create and manage a connection pool to the directory server. This critical enhancement helps in reducing the load to directory servers significantly. Obviously, for this enhancement to work, `nscd` must be started during system boot.

In addition, the RFC 2307bis schema has to be added to the existing directory instance, and the required object classes have to be added to individual user entries. If you do not want to modify the schema or user entries, then object class and attribute mapping can also be used. Keep in mind that the existing attributes are mapped to hold the correct information, for example:

```
attributeMap=passwd:gecos=cn
```

A word of caution about using mapping in general: Although initially it might seem simpler, there are certain things that could break later, because they might be specifically looking for RFC 2307bis-related object classes. In addition, when using object class mapping, it is important to understand that the object class being mapped to is essentially similar (has the equivalent attributes) to the existing one, and similarly, the mapped attribute has the same schema properties.

Using Directory Proxy as a Virtual Directory Solution

So far, we have been trying to overcome our challenges using pure directory design. Another approach that can immensely change the nature of the challenge is to use the Directory Proxy Server 6.x, which not only is an intelligent proxy (it can perform LDAP operation-based routing), but also has virtualization capabilities.

A singular (joined) virtual view can be created for LDAP clients. This virtual view could be populated from two or more DITs (sources). These sources could actually be on different directory instances or even on separate physical servers. One source could provide only the user information, such as a corporate-wide directory, while the other could be used purely for naming services.

This clear demarcation of duties has the benefit of adding the relatively smaller load of authentication to the corporate directory, as opposed to the higher number of LDAP queries generated by naming services.

Both versions of the Directory Proxy Server (5.2x and 6.x) also can be used for attribute mapping, and hence mapping at the client can be eliminated. Note that the attribute mapping in version 5.x is only for outgoing packets (searches only), while version 6.x can map bidirectionally (modifies too).

A word of caution about performance with Virtual Directory is necessary here. The Virtual Directory pulls data on every query, and hence it does not provide the same performance as a pure directory. In addition, a virtual DIT's performance is the lowest common denominator of the databases of which its view is composed. Testing with accurate use cases is obviously the best way to validate whether a virtual DIT can comply to your performance requirements.

Kerberos and LDAP Combinations

Organizations using Kerberos for authentication before the migration can continue to do so. The design of the DIT is not affected.

There are couple of approaches to using Kerberos and LDAP combinations. In the first approach, the Directory Server is used purely for naming service lookups. Users authenticate to Kerberos using their usual mechanism (`pam_krb5.so`), and the Key Distribution Center (KDC) can also be used for account management. Thus authentication and account management are done through Kerberos, whereas naming services are the Directory's responsibility.

In this case, the client profile in LDAP specifies `anonymous` as the value for the `credentialLevel` and `none` for the `authenticationMethod`, since that is what is required for pure naming service lookups.

Obviously, ACI needs to be in place to allow anonymous access. This particular marriage of Kerberos and LDAP is quite elegant, because it eliminates the need to use Secure Sockets Layer (SSL) from the LDAP clients to the Directory Server.

In the second approach, the Directory Server is configured to verify BINDs using Generic Security Service Application Program Interface (GSSAPI) over Simple Authentication and Security Layer (SASL) through Kerberos. This might be the case when, for example, account management is the responsibility of the Directory Server and `pam_ldap` is being used.

As soon as a user is authenticated to the KDC (through `pam_krb5.so`) for account management, the user still needs to BIND to the directory. Hence, the Directory Server uses the Kerberos ticket to validate the user against the KDC and allow or disallow the BIND.

The user password stored in the Directory Server for this particular case is useless and can be set to anything. As a matter of fact, as a best practice, it is advised that you do not create the `userPassword` attribute at all, so that BINDs are not possible, unless, of course, the Directory Server is being used for other purposes by LDAP clients that are not using Kerberos.

It is worth noting here that there is another unsupported approach, that is, to make LDAP BINDs authenticate directly to Kerberos through a custom pre-op Directory Server plug-in. In this case, the Directory Server acts as a proxy and the use of `pam_krb5.so` is not required for authentication. One such plug-in has been around for quite a while and is now maintained by staff of Duke University. This approach might seem simpler, but it does so at the cost of not being a pure Kerberos solution, and hence, it raises security issues.

Using Netgroups

Netgroups are primarily used to restrict logins for a given system. With netgroups, you have to use `passwd compat` mode and a lookup is done every time the system needs to map the numeric uid of a user to the login name and vice-versa.

When you move to LDAP for naming services, you can also move your existing netgroups to the DIT. However, you need to continue using `compat` mode, so your `nsswtich.conf` would minimally look like this:

```
passwd: compat
passwd_compat: ldap
```

One side effect of using netgroups is that they cause quite a bit of additional traffic in `compat` mode.

A common example will be when running the `ls` command: A lookup is done for every uid in that particular directory to map the uid number to the userid. This can be further compounded when netgroups are nested. So careful testing needs to be done to assess the impact of netgroups on your directory servers, and when possible, old netgroups should be cleaned up before migrating to LDAP.

Starting with the Solaris 10 05/08 OS, a new module called `pam_list.so.1` provides functionality similar to `pam_access` or the unsupported `pam_netgroup`. This allows you to define access at the application level (such as login or `sshd`), and hence eliminates extraneous traffic caused by `compat` mode. In particular, unlike with netgroups in `compat` mode, when you use `pam_list`(5M), a netgroup lookup is done only at login time.

Integration With Microsoft Active Directory

We discussed the subject of Kerberos, so it is worth mentioning that Microsoft Active Directory can also be used as a KDC for the Solaris LDAP client. There are two approaches to this integration.

Prior to the Solaris 10 08/07 OS, you have to use a proxy account to access Active Directory. This approach is described well in a blog (see the link for Solaris 10 and Active Directory integration in the "For More Information" section), but it is not really good for auditing purposes, because a proxy account is used for all users.

The other approach, which is possible with the Solaris 10 08/07 and later releases, may be more elegant and uses the SASL/GSSAPI mechanism for the LDAP client to authenticate individual users to Active Directory (called self-credentials). For more information on this method, see the link for "Using Kerberos to Authenticate a Solaris 10 OS LDAP Client With Microsoft Active Directory" in the "For More Information" section.

Improvements Added by OpenSolaris™ Projects

The OpenSolaris Sparks, Winchester, and Duckwater projects have added significant design improvements to the naming services, new functionality, and management tools. However, only Sparks has been back ported to Solaris 10 08/07 and later releases.

Sparks introduced several improvements in the Solaris name service framework. For example, `nscd` now pools LDAP connections, which are shared by multiple name service requests instead of using the expensive approach of creating and dropping a connection per request. `nscd` also caches more databases than before.

The Solaris LDAP client now supports Kerberos and can use self-credentials to authenticate to the LDAP server instead of using a proxy account. This self-credential functionality has already been back ported to the Solaris 10 08/07 OS and later releases, as evident by the Active Directory integration described in an earlier section.

Winchester provides a service that can be used by Solaris components, for example, the Solaris Common Internet File System (CIFS) server, to map Microsoft Windows identities to Solaris identities and vice-versa.

Duckwater is planned to simplify management of Solaris name services by providing unified tools to configure and administer name services, support multiple name service configurations on a system with the ability to switch easily between the configurations, and provide an improved LDAP client configuration experience.

Note that currently there are no plans to port Winchester and Duckwater to the Solaris 10 OS.

Conclusion

A couple of the important reasons for moving to LDAP as a naming service are to centralize user management and to move away from NIS/NIS+.

In this paper, we presented few approaches for such a move and described their pros and cons. Using unified `people` and `group` containers has long-term compliance and administrative advantages. But these advantages can come at the cost of a potentially significant consolidation effort, depending on the size and complexity of your environment.

Similarly, figuring out the ideal DIT for your environment largely depends on your requirements and topology. Factors such as delegated administration, logistics, time tables, resources, geographic displacement, and technology awareness also play an important role in this transition.

Our experience has shown that making user names and group names (and the uid and gid number) unique should be a step that happens in the existing environment, if possible, before the transition to LDAP. In addition, a phased approach should be used for the transition to make the goals realistic and achievable.

For More Information

Here are additional resources:

- *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*: http://docs.sun.com/app/docs/doc/816-4556
- Sparks/Reno/Duckwater/Winchester Overview (OpenSolaris projects): http://www.opensolaris.org/os/project/sparks/overview/;jsessionid=2C00F42CD9816BCB94B9FFBEB78DBF8E
- OpenSolaris Sparks Project: name service switch/nscd enhancements: http://www.opensolaris.org/os/project/sparks/
- `pam_list`(5): http://docs.sun.com/app/docs/doc/819-2252/pam-list-5?a=view
- `pam_netgroup.c`: http://www.opensolaris.org/os/community/security/files/
- *Windows Security and Directory Services for UNIX Guide v1.0*: http://technet.microsoft.com/en-us/library/bb463150.aspx
- Duke University Directory Plug-in: http://www.oit.duke.edu/~rob/krbdirp/
- RFC 2307: http://www.ietf.org/rfc/rfc2307.txt
- RFC 2307bis: http://www.padl.com/~lukeh/rfc2307bis.txt
- Blogs:
    - Active Directory domain CIFS member server: http://blogs.sun.com/jurasek/entry/ads_domain_member_server1
    - Solaris 10 and Active Directory Integration: http://blog.scottlowe.org/2006/08/15/solaris-10-and-active-directory-integration/

- Related articles on BigAdmin:
    - Using Kerberos to Authenticate a Solaris 10 OS LDAP Client With Microsoft Active Directory: http://www.sun.com/bigadmin/features/articles/kerberos_s10.jsp
    - Sun Java System Directory Server 6.0 as an LDAP Naming Service: http://www.sun.com/bigadmin/features/articles/nis_ldap_part1.jsp
    - Benefits of Using the Solaris 10 OS With Sun Java System Directory Server Enterprise Edition 6.x: http://www.sun.com/bigadmin/features/articles/s10_dsee_benefits.jsp
    - Benefits of an Exclusively Multimaster Deployment of Sun Java System Directory Server Enterprise Edition 6: http://www.sun.com/bigadmin/features/articles/dsee6_multimaster.jsp
    - Maximizing Performance of the Sun Java System Directory Server: http://www.sun.com/bigadmin/features/articles/slapd_sjsds.jsp
    - Resetting the 'admin' Password in Sun Java System Directory Server: http://www.sun.com/bigadmin/features/techtips/admin_pwd.jsp
- Sun Developer Network, including Identity Management section: http://developers.sun.com/identity/
- Download page for Sun Java System Directory Server Enterprise Edition: http://www.sun.com/software/products/directory_srvr_ee/get.jsp
- Sun Java System Directory Server Enterprise Edition web site: http://www.sun.com/software/products/directory_srvr_ee/index.jsp
- Directory Proxy Server web site: http://www.sun.com/software/products/directory_srvr_ee/dir_proxy/index.xml
- Sun training courses at http://www.sun.com/training/, for example:
    - Sun Java System Directory Server Enterprise Edition 6.x: Analysis and Planning (DIR-2217)
    - Sun Java System Directory Server Enterprise Edition 6: Maintenance and Operations (DIR-2340)
    - Sun Java System Directory Server Enterprise Edition: LDAP Concepts (WMT-DIR-1344)
    - LDAP Design and Deployment (WI-3501)
- Sun forums, such as:
    - Web and Directory Servers forum: http://forums.sun.com/category.jspa?categoryID=51
    - BigAdmin discussions: http://www.sun.com/bigadmin/discussions/
- Product documentation at http://docs.sun.com/, such as:
    - Sun Java System Directory Server Enterprise Edition documentation: http://docs.sun.com/app/docs/prod/dirsvr.ee#hic
    - Solaris 10 OS documentation: http://docs.sun.com/app/docs/prod/solaris.10

- OpenSolaris Documentation Community: http://opensolaris.org/os/community/documentation/
- Sun wikis, such as:
  - Sun BluePrints™ wiki: http://wikis.sun.com/display/BluePrints/Main
  - BigAdmin wiki: http://wikis.sun.com/display/BigAdmin/Home
- Support:
  - Sun resources:
    - Register your Sun gear: https://inventory.sun.com/inventory/
    - Services: http://www.sun.com/service/
    - SunSolve℠ Online: http://sunsolve.sun.com/
  - Community system administration experts: http://www.sun.com/bigadmin/content/communityexperts/
- Events of interest to users of Sun products:
  - Worldwide Developer Events and Sun Tech Days: http://developers.sun.com/events/
  - Current Events: http://www.sun.com/events/index.jsp

Licensing Information