Integrating HP-UX Account Management and Authentication with LDAP [1]

Hewlett-Packard
May 04, 2000

Why Integrate LDAP with UNIX?

The acceptance of Lightweight Directory Access Protocol (LDAP) technology has progressed at a rapid pace. Many enterprises have already deployed LDAP directories, primarily for messaging and security products. As more applications are directory enabled, important tasks such as administration, authentication and authorization, are being consolidated and centralized. Integrating important operating systems into the directory greatly enhances the value of this consolidation. Most UNIX vendors have some primitive directory enablement, but little real integration with LDAP.

In June 2000, HP is releasing several new products on HP-UX 11.0 that will provide a full range of directory integration. Customers may choose the level of integration that meets their needs, or may migrate their environments one level at a time:

- **YPLDAP** is a protocol gateway that allows UNIX configuration data to be migrated to an LDAP directory, and accessed via existing client software (NIS). Previously, this product was only available on HP-UX 10.20.
- **NSS_LDAP** accesses configuration data via native LDAP.
- **PAM_LDAP** authenticates HP-UX users to an LDAP directory.
- **LDAP Access Profiles** provide the ability to customize NSS_LDAP and PAM_LDAP directory access to enable integration and data sharing with other applications and platforms using LDAP.

Lightweight Directory Access Protocol

Lightweight Directory Access Protocol (LDAP) is an Internet standard produced by the Internet Engineering Task Force (IETF). The original LDAP RFC was written by W. Yeong, T. Howes, and S. Kille in University of Michigan. The protocol was designed to provide access to the Directory while reducing the resource requirements of the Directory Access Protocol (DAP). The key feature of LDAP was that the protocol ran directly over TCP or other transports without requiring the overhead of session/presentation layer overhead. LDAP providers support the most popular methods of authentication, including password based, Secure Socket Layer (SSL), and Kerberos. LDAP support of the Simple Authentication and Security Layer (SASL) allows for additional authentication methods to be negotiated. The following RFCs provide detailed information about LDAPv3 protocol, and other LDAP related standards:

- Lightweight Directory Access Protocol v3 (RFC 2251)
- An Approach for Using LDAP as a Network Information Service (RFC 2307)
- A Summary of the X.500(96) User schema for use with LDAPv3 (RFC 2256)
- LDAPv3 Attribute Syntax Definitions (RFC 2252)
- UTF-8 String Representation of Distinguished Names (RFC 2253)
- The String Representation of LDAP Search Filters (RFC 2254)
- The LDAP URL Format (RFC 2255)
- Simple Authentication and Security Layer (RFC 2222)
- SSL 3.0 specification

UNIX and Directories Today

Originally, UNIX account and configuration information was stored in a series of text files. As the need to share this information across systems increased, the first widely accepted product named Yellow Pages, and later renamed to Network Information Service (NIS) was developed by Sun Microsystems. NIS provides network wide management of many UNIX configuration files (e.g., /etc/passwd, /etc/group, /etc/services). An NIS master server generates maps based on the configuration files and transfers copies to slave servers. On NIS client systems, operations reading the configuration file are redirected to send a request across the network to retrieve the information from an NIS server.

While providing a high degree of backward compatibility with file based configuration, NIS has limitations in scale and security that prevent it from being easily deployed in enterprise environments. NIS does not support delta based updates, causing entire maps to be transferred to all the slave servers. These maps are transferred across the network unencrypted. The underlying database used by NIS servers can support a limited number of entries, requiring administrators to break up the data by creating multiple NIS domains. Despite these shortcomings, NIS is widely used today across a variety of UNIX platforms.

NIS+ was introduced as a successor to NIS to provide greater scalability and security. While succeeding to some extent, NIS+ has not achieved the level of acceptance of NIS. UNIX administrators have reported that the level of complexity in administering NIS+ often outweighs the benefits. With the arrival of more general purpose directories, the potential of a more powerful generic directory has supplanted the NIS model in the imagination of the UNIX community. NIS+ suffers from lack of interoperability and therefore it is missing the much needed flexibility in hybrid environments.

The acceptance of LDAP, and the deployment of LDAP directories in many enterprises has created a need for existing UNIX clients from a variety of vendors to access data stored in an LDAP directory.

HP-UX NIS/LDAP Gateway (YPLDAP)

The HP-UX NIS/LDAP Gateway product (YPLDAP) is a protocol gateway that translates requests and replies between NIS clients and LDAP servers. UNIX information such as user accounts, groups, and services are stored in an LDAP Directory in the format defined by the RFC 2307 schema. UNIX clients configured to use NIS will be able to interact with YPLDAP without modification.

YPLDAP replaces the NIS slave server *(Figure 1)*. NIS slaves are updated by receiving complete maps transferred from an NIS master. YPLDAP requests only the required data from the LDAP server, eliminating the transferring of maps across the network. This saves processing time on the servers and minimizes the network bandwidth required.

NIS master servers are not needed with YPLDAP. This functionality is now being provided by an LDAP directory. This allows a greater number of entries to be stored than in a traditional NIS master. The credentials (e.g., user name and password) that YPLDAP uses to connect to the LDAP directory can be configured. Communications between YPLDAP and the LDAP directory can be protected with a Secure Socket Layer (SSL) connection using X.509 certificates for authentication and encryption.

By taking advantage of the scalability of LDAP, YPLDAP supports much larger domains than NIS. Administrators now can consolidate NIS domains. YPLDAP employs caching to minimize latency when accessing the LDAP server. It supports the commonly used NIS maps, including passwd, group, hosts, networks, aliases, netgroup, and services. YPLDAP will be available on HP-UX 10.20 and 11.0 platforms and will support any NIS version 2 compatible client, including HP-UX 10.20.

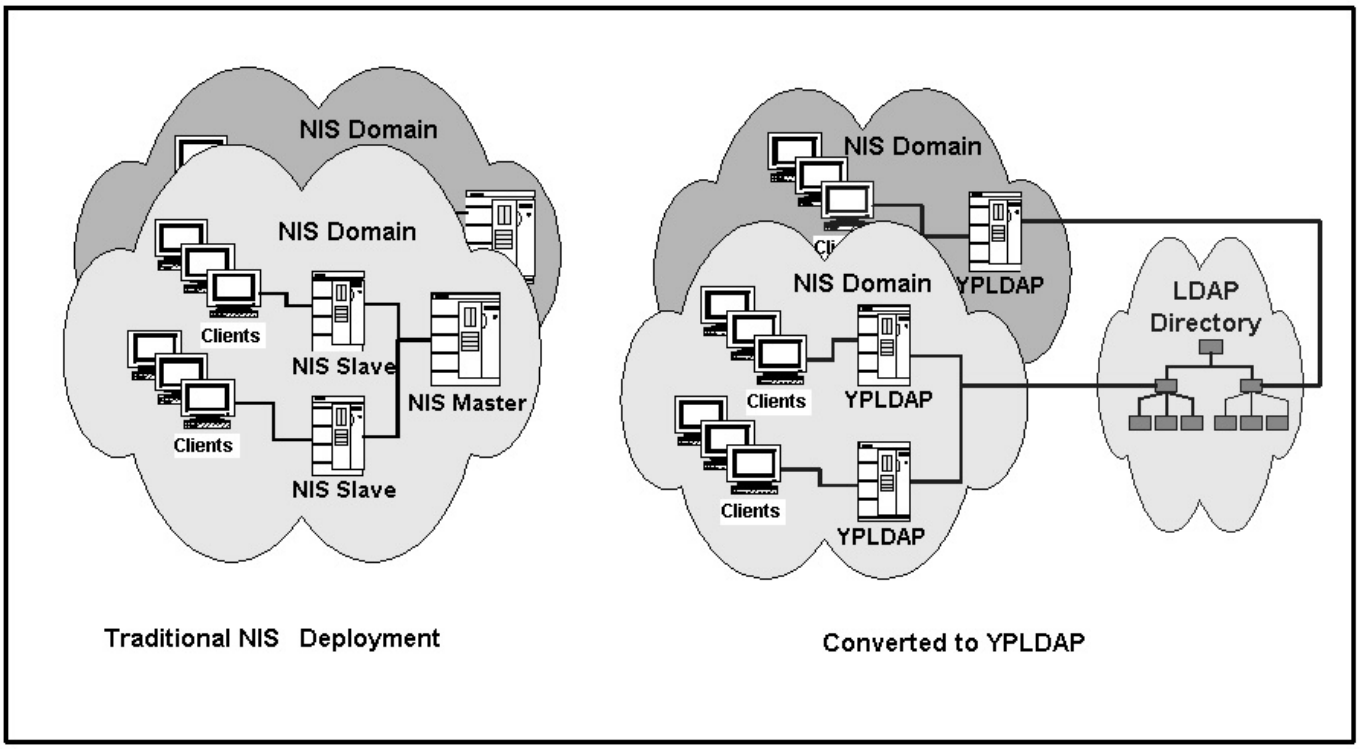

Figure 1. Migrating NIS to LDAP Gateway (YPLDAP). *The NIS slaves are replaced by YPLDAP daemons. The NIS masters are replaced by subtrees in a single LDAP directory.*

HP-UX LDAP Integration Client Services (LDAP-UX)

The HP-UX LDAP client integration product (LDAP-UX Client Services) is a set of modifications allowing HP-UX to more directly interact with LDAP. Designed with the goal of being directory vendor neutral, and flexible regarding tree structure, schema and naming conventions, LDAP-UX often can be introduced into an existing LDAP directory tree. LDAP-UX Client Services allows HP-UX to retrieve account, group, and system LDAP configuration from, and authenticate (i.e. login) to an LDAP directory. LDAP-UX Client Services consists of the following functionality:

- **NSS_LDAP**
- **PAM_LDAP**
- **LDAP Access Profiles**

NSS_LDAP

As of 11.0, HP-UX supports the Name Service Switch (NSS) architecture, allowing commands and applications to retrieve name service information (users, groups, services, etc.) without having knowledge of where or how it is stored. Commands and applications call standard C library functions, which in turn use the Name Service Switch to determine which name service "back-end" routines to call. For example, if the NSS configuration file /etc/nsswitch.conf was configured to first look in files and then in NIS for user entries, and a user entered the ***who*** command the following would happen *(Figure 2)*:

1. The ***who*** command calls the function ***getpwuid()*** in the C library, passing it the user id number for the current session.
2. The ***getpwuid()*** function calls into the Name Service Switch "front end " with the request. The front end calls into the back-end for files, which would search the /etc/passwd file for the user.
3. If the user id was not found, the front end calls the next configured back-end (in this example NIS.) The NIS back-end makes a set of Remote Procedure Calls to an NIS Server to search for the user entry.

4. If found, the front end returns a UNIX password record to ***getpwuid()*** which in turn passes the result back to the ***who*** command.
5. The ***who*** command now displays the user name from the password record.

NSS_LDAP is a new back-end that searches an LDAP directory for name service information. The first release will support user, group and shadow password entries. It has the ability to connect to the directory as an anonymous user, a configured proxy user, or as the user id of the calling process when used in conjunction with PAM_LDAP.

The advantage of using NSS_LDAP over YPLDAP is derived mainly from accessing the LDAP directory directly. There is no longer a need to configure a separate gateway server to translate requests. NIS domains are no longer required. A rich set of configuration options allow each HP-UX system to specify up to three search filters into the directory. Search filters specify where in the directory tree to start the search, how deep to search, and what rules to apply to determine a match. Attribute mappings may also be configured to allow NSS_LDAP to integrate with directories that do not store name service data in the format specified by the RFC 2307 schema.

This flexibility in searching and schema support allows NSS LDAP to coexist with other products inside a single directory, sharing data such as user names and passwords. It should be noted that YPLDAP may still be a better choice for environments that currently support 10.20, or prefer backwards compatibility over closer directory integration.

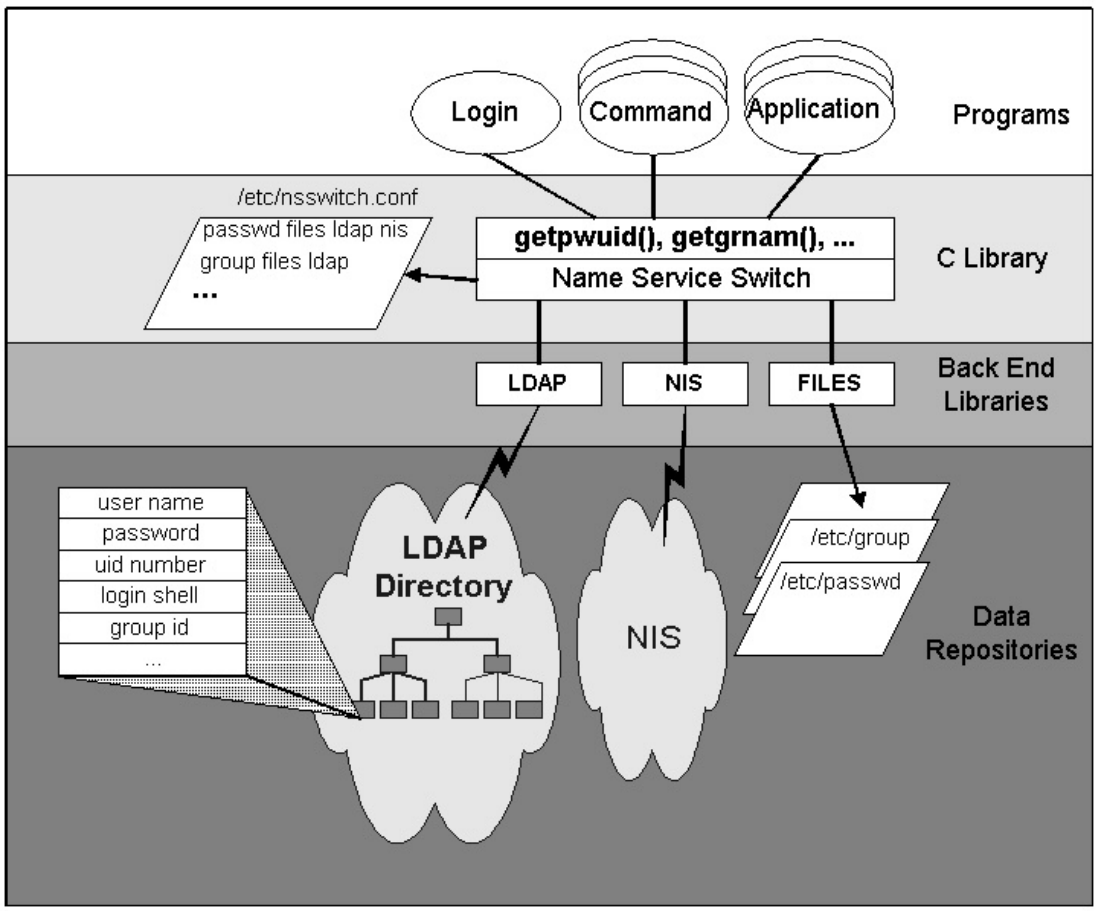

Figure 2. LDAP and the Name Service Switch Architecture. *In this example, a user entry will first be searched for in /etc/passwd, then in an LDAP directory, and finally in NIS.*

PAM_LDAP

Similar to the Name Service Switch, the Pluggable Authentication Module (PAM) architecture allows the UNIX administrator to configure multiple authentication methods. Unlike NSS, PAM allows configuration on a per service basis. The default method is traditional UNIX authentication. In this example the Name Service Switch module is also configured to use LDAP:

1. The login program calls the C library function ***pam_authenticate()***.
2. ***pam_authenticate()*** calls the authentication function for the ***PAM_UNIX*** module.
3. ***PAM_UNIX*** calls ***getpwnam()*** to retrieve the passwd record for the user logging in.
4. ***getpwnam()*** uses the NSS as previously discussed to retrieve the password record.
5. ***PAM_UNIX*** now challenges the user for a password, encrypts it (via UNIX crypt), and compares the password with the one retrieved in the password record.
6. ***PAM_UNIX*** applies any relevant policy management, such as lockout procedures. Policy management for ***PAM_UNIX*** is minimal, unless the system is configured for commercial security.
7. ***PAM_UNIX*** returns the result of the authentication. If the user has been authenticated, login calls getpwnam() to retrieve account information, including the user's home directory group and login shell.

Using PAM_UNIX with LDAP has the benefit of using LDAP access while preserving traditional UNIX behavior (e.g., all NIS maps are supported). However, it does so at the expense of directory integration and security. The password must be stored in the directory in UNIX crypt format, and the directory must allow read access to all passwords for an anonymous or configured proxy user.

PAM_LDAP allows HP-UX users to authenticate directly to an LDAP directory. The following example shows a password based authentication to

an LDAP directory (*Figure 3*):

1. The login program calls the C library function ***pam_authenticate()***.
2. ***pam_authenticate()*** calls the authentication function for the ***PAM_UNIX*** module.
3. ***PAM_UNIX*** attempts to find and authenticate the user based on the entry returned by ***getpwnam()***. Note that LDAP-UX has the capability to disable PAM_UNIX authentication for user entries found in the LDAP directory.
4. If ***PAM_UNIX*** authentication was unsuccessful, ***pam_authenticate()*** calls the authentication function for the ***PAM_LDAP*** module.
5. ***PAM_LDAP*** searches the directory for an entry with a user name that matches the name of the user attempting to log in.
6. ***PAM_LDAP*** attempts to bind to the directory as the user with the matching entry.
7. The directory receives the bind request, and applies its own password check and policy management routines to determine whether to accept or reject the bind request.
8. If the directory accepts the bind request, ***PAM_LDAP*** stores the user's login credentials. These credentials will be used later when the user makes any system or library call that requires retrieving data from LDAP.
9. ***PAM_LDAP*** returns the result of the authentication. If the user has been authenticated, login calls ***getpwnam()*** to retrieve account information.

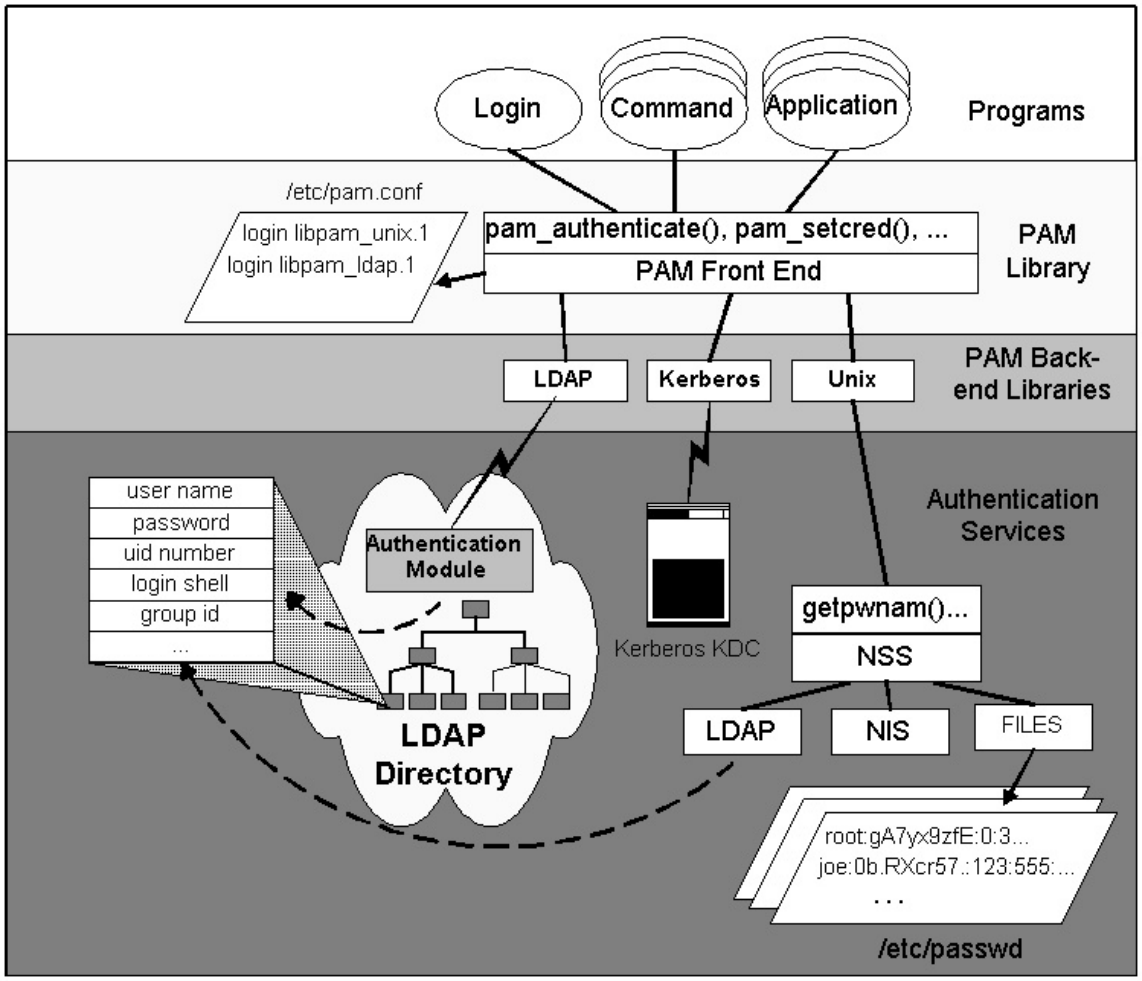

Figure 3. Pluggable Authentication Modules and LDAP. *This system has been configured to authenticate root and user joe through PAM_UNIX, while other users will authenticate via PAM_LDAP. If /etc/nsswtch.conf has been configured as shown in Figure 2, then root and joe can be managed locally by the system administrator.*

In combination with NSS_LDAP, PAM_LDAP provides closer integration with an LDAP directory. The primary benefit is that of common authentication. HP-UX is using the same authentication mechanisms that other directory enabled applications use. PAM_LDAP is neutral to the password storage format. All authentication is subject to the directory's security policy management which provides two benefits:

- Policy is enforced across systems. This is important when the authentication information such as password is shared across multiple systems. For example, a directory administrator configures a lockout policy that disables accounts after five failed attempts. A cracker program fails to log in after five attempts on one HP-UX system. Switching to a second system will not yield any further attempts.
- Policy is enforced across applications. HP-UX login, directory-enabled applications running on a variety of platforms, and web-based applications share the same policy management and administration tools. So not only is policy enforced across HP-UX systems, the same policy is enforced across a variety of applications.

LDAP Access Profile

When applications are carefully designed, using standard access methods, allowing the customer to choose tree structure and naming conventions, the full power of the directory is harnessed. A single directory can be deployed to support multiple applications and operating systems. For example, an organization may design its tree such that a single user entry contains a combination of the information required for cooperating operating systems and applications. Common attributes (e.g., password, common name) is shared instead of duplicated.

Deploying NSS_LDAP and PAM_LDAP into an LDAP directory requires the ability to configure where, how, and what to search for in the directory. Since this configuration will typically be the same for a number of HP-UX systems, LDAP-UX stores this sharable configuration in the

LDAP directory in an entry called a profile. Each individual HP-UX system now only needs to configure an LDAP server and the name of the profile to be used.

The HP-UX LDAP profile includes the following:

- **Server name(s) and port(s)**. This is a set of server names and port ids where the actual search for data will be performed. If a server does not respond to a bind request, HP-UX will attempt to bind to the next server listed.
- **Authentication level**. This value specifies whether the directory should be accessed anonymously, or via a configured proxy user.
- **Proxy user**. The administrator can specify an identity to use when connecting to the directory in cases when the server should not connect as a specific user. This can be anonymous access, or a locally configured directory user credential.
- **Search filters**. Up to three search filters may be configured for each name service (e.g. users, groups). A search filter specifies a directory tree location to start the search, search depth, and an LDAP filter to control which entries to retrieve. Search filters are a key component of the flexibility in LDAP-UX, allowing HP-UX to be integrated into a wide variety of directory trees.
- **Attribute mappings.** These mappings match UNIX data structures to one or more directory attributes defined by the directory schema. While typically defaulted to follow RFC 2307, attribute mapping can be used as a customization tool. For example, the UNIX gecos field in the passwd file is mapped by RFC 2307 to the gecos attribute in the directory. However the directory administrator can choose to build the gecos field by selecting existing attributes such as cn (common name), officePhone, and location. Attribute mappings are also useful to integrate with directories that do not fully support RFC 2307.

LDAP-UX downloads the profile to each HP-UX system for performance reasons. A set of tools and customizable scripts are provided to create, modify and periodically download profiles.

Putting It all Together

By using a combination of of NSS_LDAP, PAM_LDAP and HP-UX LDAP Profiles, a high level of integration with existing LDAP directories and directory enabled applications can be achieved. In the first example *(Figure 4)* HP-UX is integrated into an existing Netscape LDAP directory that is primarily used for email. The directory schema already contains RFC 2307, and the directory supports dynamic addition of auxiliary object classes. Existing email entries in the directory are extended to include UNIX account information. Attributes in common between email and UNIX users, including user name and password are shared (i.e. each record would contain a single user name and password). A UNIX user logs in and changes his/her password using an HP-UX command. Now the same user logs in with the new password to an email application on a PC. Third party directory based administration tools can be used to manage the account information.

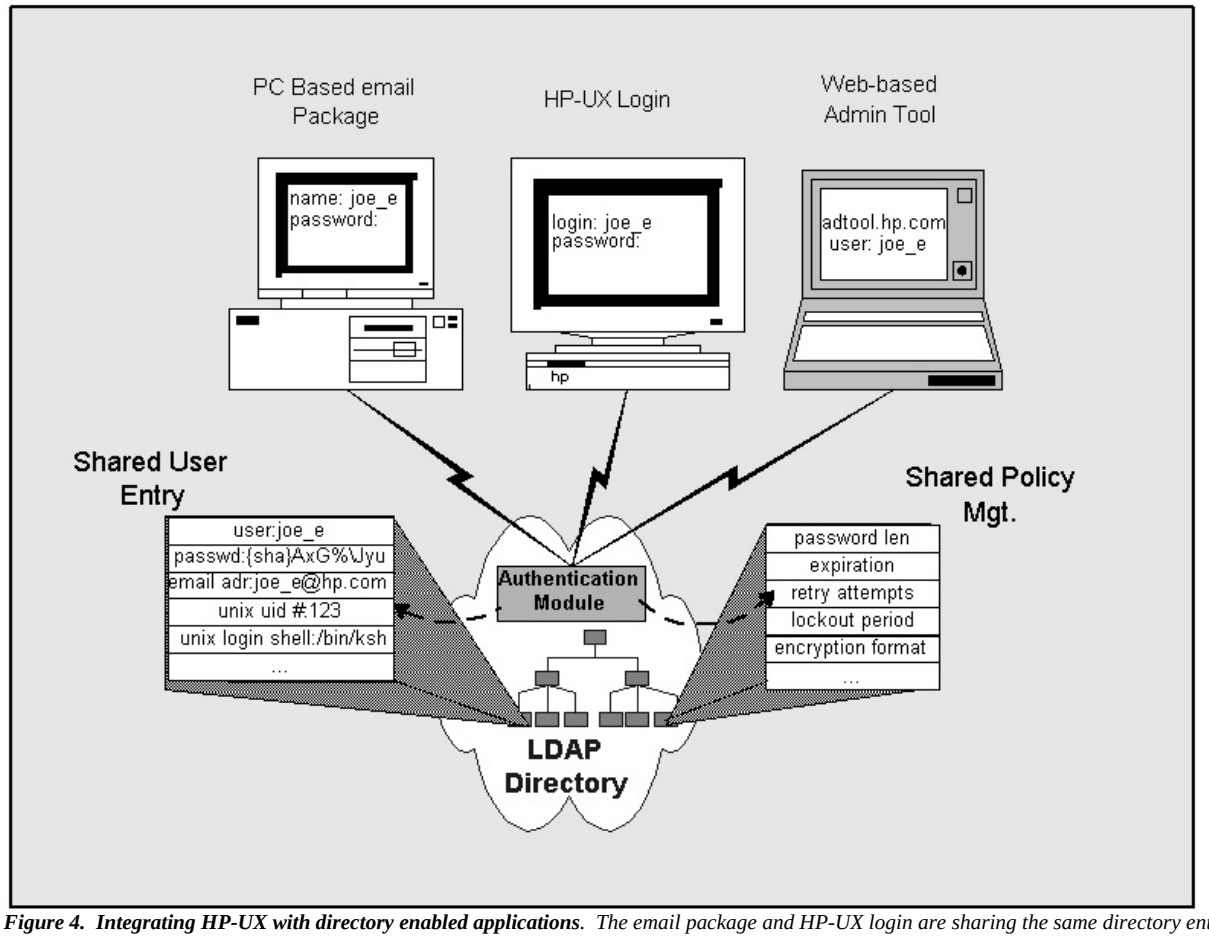

Figure 4. Integrating HP-UX with directory enabled applications*. The email package and HP-UX login are sharing the same directory entry, authentication module, and policy management. Various administration tools (including web based) may also be used to manage the entry.*

In the second example (Figure 5), we have a more ambitious implementation. This directory is divided into several subtrees based on geographic organization. Each subtree has a separate profile. The LDAP-UX search filters are configured to first look for users and groups in the local organization and then to search the full directory.

This organization also has some systems that restrict access to a subset of users. These key user entries are extended to include a new object class with an attribute (created by this organization) called *accessType*. These systems with restricted access will use a separate profile called *finance profile*, where the search filter will specify that the user entry must also have the appropriate value for *accessType*. When a user attempts to log in, PAM_LDAP will only find the user entries that have a match in the *accessType* attribute.

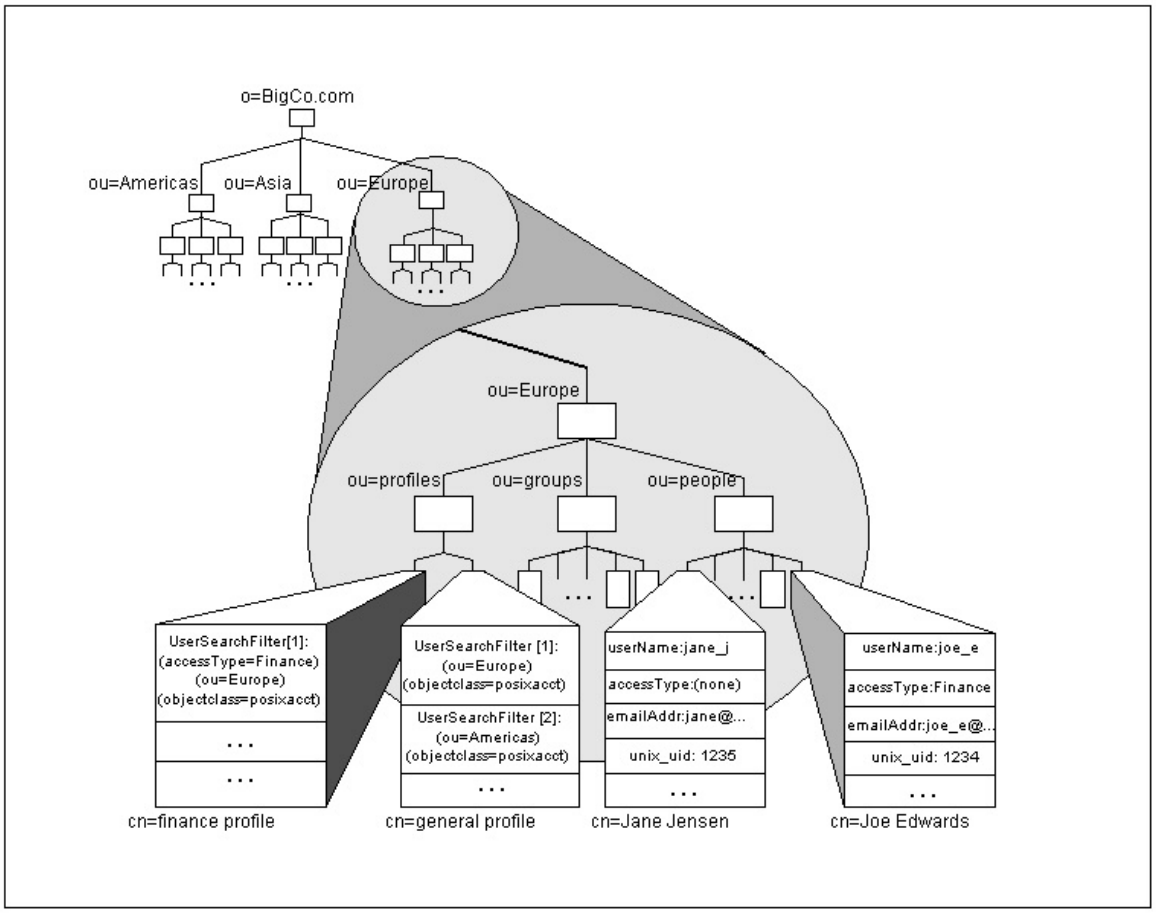

Figure 5. Controlling UNIX login with profiles*. On the systems using general profile, either Jane or Joe may log in. On systems using finance profile, only Joe is accepted.*

Some directories contain rich access control list (ACL) functionality that can be used as well. Features such as limiting access to an entry, profile or subtree based on the caller's encryption level, IP address, or DNS domain, can be utilized in a variety of situations. Currently, LDAP ACL functionality is vendor specific. An LDAP ACL draft is under review at the IETF.

What's Next? The future of HP-UX/LDAP Integration

This paper describes the first phase in a continuing effort by HP-UX to further integrate with LDAP and to interoperate with directory enabled applications. HP is currently developing enhancements to LDAP-UX to support greater password security via Digest-MD5 and/or SSL, and interoperability with the existing PAM_Kerberos product. Certification of Microsoft ADS and integration with Windows 2000 accounts are also under development. Additional Name Service databases will also be supported by NSS_LDAP.

Other HP-UX products are also developing or investigating directory enablement. These products include DNS, Web QoS, IPSec, VPN, and system configuration management.

[1] This paper describes products currently under development. Hewlett-Packard reserves the right to modify development plans for any of the features discussed in this document.