Unified Authentication, Authorization and User Administration – An Open Source Approach

Ted C. Cheng, Howard Chu, Matthew Hardin

{tedcheng,hyc,mhardin}@symas.com

Introduction

Authentication, authorization, and user administration are critical components in secure enterprise computing. This paper presents a unified architecture for authentication, authorization, and user administration for Linux and Unix-variant environments. The architecture aims at providing robustness, performance, scalability, high availability, and low total cost of ownership (TCO) for enterprise deployments, while preserving the compatibility with existing IT infrastructure. The design scales vertically and horizontally by distributing workload among multiple OpenLDAP [1] servers. The flexible cache configuration minimizes network access traffic, delivering the high performance demanded by enterprise business needs. The design offers high availability via the LDAP Sync Replication (*syncrepl)* [2] services.

Enterprises can deploy services in a modular and scalable manner based on corporate geographical regions and organizational units. One or more servers can be deployed for each geographical region to minimize cross-region access latency. Clients can further be configured to access services for their respective organizational units from the regional servers. The cache configuration on the client systems stores information locally, eliminating redundant accesses to the regional but remote servers. This is particularly critical for heavily loaded systems in which repeated service requests for the same information can be handled via local cache.

The design of the unified authentication, authorization, and user administration architecture is the result of the cumulative efforts of the open source community in the past many years, involving technologies such as Name Service Switch (NSS), Network Information Services (NIS/NIS+), Domain Name Services (DNS), Pluggable Authentication Modules (PAM) [3], Lightweight Directory Access Protocol (LDAP) [4], LDAP Content Synchronization, Secure Socket Layer/Transport Layer Security [5] (SSL/TLS), OpenLDAP SLAPD, and so on.

This paper reviews the evolution of related technologies and provides an overview of the architecture. The presentation then focuses on two distinct enhancements, i.e., LDAP client-side caching and disconnected operations, using the OpenLDAP Name-Service Overlay (*nssov*) and the Proxy Cache [6] engine. The nssov overlay improves the robustness of the design by addressing many issues in the PADL approach, while the Proxy Cache engine offers persistent caching capability and takes advantage of the connection pooling of the ldap backend.

Evolution of Related Technologies

Traditionally Unix-like systems require various name services to function properly. The system local storage using flat files is constrained in terms of performance and scalability. Furthermore, flat files cannot be readily shared by multiple systems, even when those systems share the same configuration. Additional name service modules, such as Network Information Services (NIS/NIS+), were developed to support alternative database stores and facilitate data sharing.

Pluggable Authentication Modules (PAM) is a unified authentication framework in which multiple modules can be stacked together to provide authentication services based on the needs of each individual application. The framework further divides functionality into authentication, account management, session management, and password modules which can be implemented and plugged in separately.

As directories become popular in an enterprise IT infrastructure in which enterprise information can be stored centrally and hierarchically in a distributed database, the PADL [7] approach introduced two libraries, nss_ldap and pam_ldap, that integrate LDAP directories into the aforementioned name services and pluggable authentication framework.

The PADL approach was a big step forward and has become popular in many large deployments. As the approach gained popularity, a number of opportunities for improvement also surfaced. For example, the libraries that nss_ldap depends on may conflict with those loaded by applications, when nss_ldap is built as a shared library. On the other hand, applications may become bloated if nss_ldap is built as a static library. In addition, applications loaded with nss_ldap may not be thread-safe due to the fact that nss_ldap makes use of libraries that are not themselves thread-safe.

Furthermore, nss_ldap and pam_ldap provide very limited caching capability, which poses performance issues, particularly over high latency or low-bandwidth networks. Also there is no resource management for connections to LDAP servers, and both nss_ldap and pam_ldap become non-functional when the LDAP servers are disconnected.

The nss-pam-ldapd daemon [8] developed by Arthur de Jong and Howard Chu also provides LDAP support for system name resolution, authentication, and authorization services. The design includes nss_ldap and pam_ldap similar to the PADL's approach, except that those libraries do not issue LDAP client calls directly to LDAP servers. Instead, the libraries communicate with the nss-pam-ldapd daemon via the NSLCD protocol through a Unix domain socket. The approach modularized LDAP-related functionalities into a separate process, which addressed many issues in the original PADL design.

Unified Authentication, Authorization, and User Administration using OpenLDAP SLAPD and its Extensions

The unified authentication, authorization, and user administration design replaces the nss-pam-ldapd daemon with the SLAPD server, further making available full-fledged SLAPD features such as performance, scalability, and high availability.

The architecture employs OpenLDAP Name-Service Overlay (*nssov*) which is configured as part of the SLAPD server. The *nssov* overlay listens to a Unix domain socket for service requests which can be serviced either by database backend(s) or by remote, and possibly foreign, LDAP servers, taking advantage of the connection pooling of the ldap backend as well as the persistent caching capability of the OpenLDAP Proxy Cache engine. Furthermore, a replicated database provides the failover capability during network outage, Figure 1.

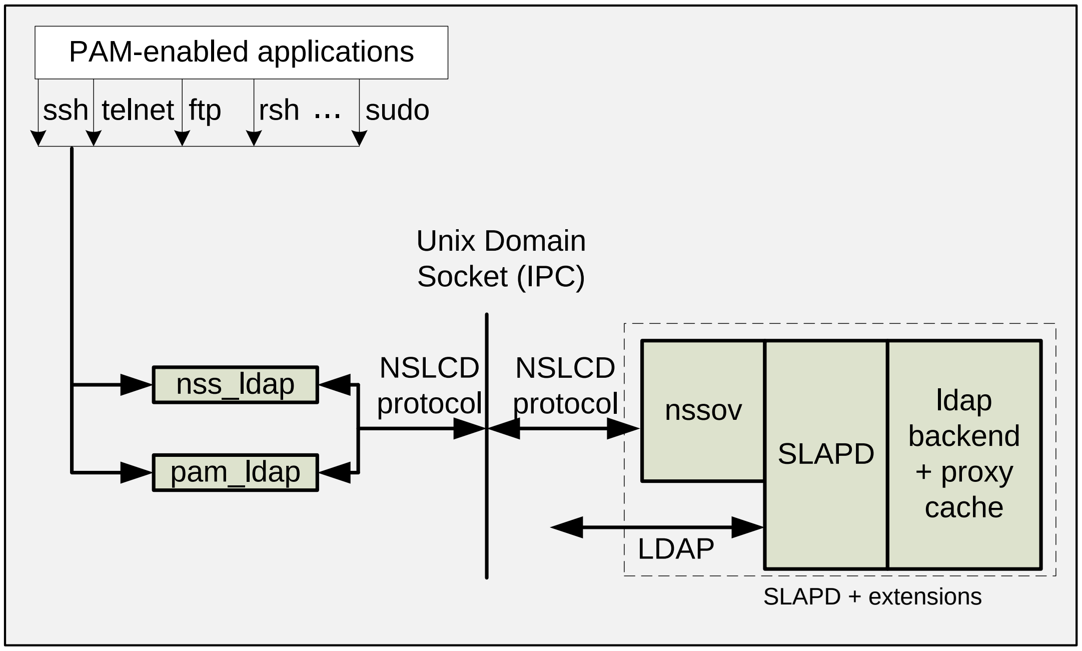

Figure 1 The unified authentication, authorization, and user administration architecture using OpenLDAP SLAPD with Name-Service Overlay (nssov), ldap backend, and proxy cache

The design inherits the architecture enhancements from nss_ldap, pam_ldap, and nss-pam-ldapd, with the added features of client-side caching, support for disconnected operations, and LDAP connection sharing:

- The authentication, authorization, and user administration information can be distributed in a hierarchical and scalable manner, contrasting the flat-file approach.
- The system works alongside with existing solutions, such as NIS, NIS+, DNS, and flat files.
- No application re-compilation or re-linking is needed.
- The LDAP library is not loaded with every process that accesses the services.
- There is no deadlock in host name lookup of LDAP server during boot process.
- Resource management is available for connections to LDAP servers.
- Cache extension can be configured to hide access latency.
- Replication is possible for high availability and disaster recovery.
- The design offers support for disconnected operations.
- Account management can be configured for policy compliance.

Sample architecture for enterprise deployments is illustrated in Figure 2. The architecture employs two SLAPD servers as the directory backbone, each serving its own regional information demands. Both SLAPD servers are configured with the same suffix and with *syncrepl* for replication. Each regional directory server supports multiple clients (organizational units); each client system is configured with a unified authentication, authorization, and user administration module as depicted in Figure 1. The architecture supports the following goals:

- Regional clients are serviced by their respective regional SLAPD servers. Cross-region accesses by clients are reduced to a minimum, if not completely eliminated.
- Cross-region replication can involve less-critical business information for local regions, and therefore can be scheduled.

- Each client accesses its own sub-tree in the Directory Information Tree (DIT). The modular approach offers scalability for enterprise business growth, as naming space can be provisioned as new clients come on board.
- Each client is equipped with cache configuration. Repeated client accesses will be resolved locally, which eliminates latencies for remote server accesses.

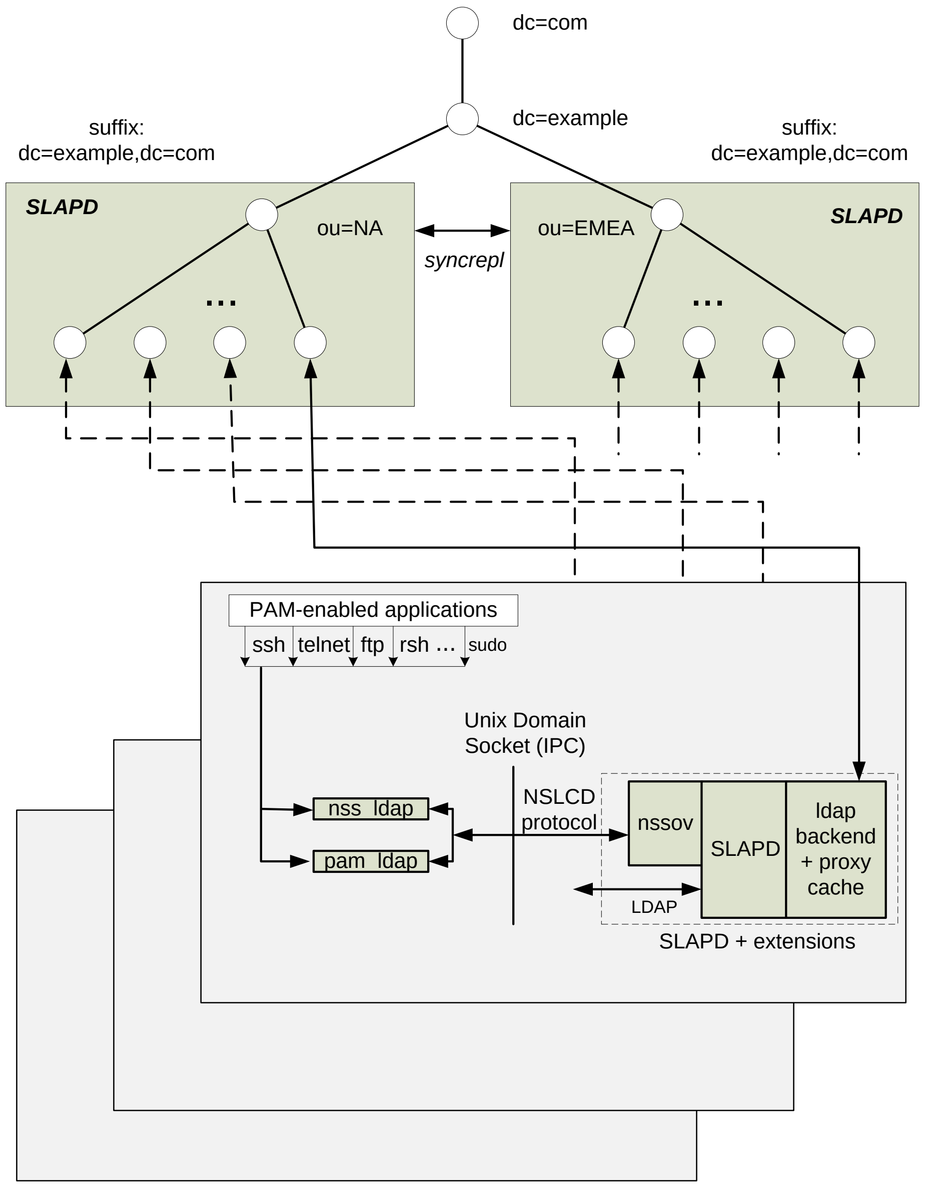

Figure 2 Sample unified authentication, authorization, user administration architecture, using the ldap backend and proxy cache, for enterprise deployments

The Name-Service (nssov) Overlay

The OpenLDAP overlays are software components that can be stacked together to customize SLAPD behavior. The nssov overlay offers the additional NSLCD client communication protocol to the SLAPD server.

The nssov overlay may be configured with Service Search Descriptors (SSDs) for each NSS service:

```
nssov-ssd <service> <url>
```

where <service> may be one of the following: aliases, ethers, group, hosts, netgroup, networks, passwd, protocols, rpc, services, shadow. The <url> must be of the form:

```
ldap:///[<basedn>][??[<scope>][?<filter>]]
```

The default <basedn> and <scope> are respectively first suffix of the current database and *subtree*. The default <filter> depends on which service is being used.

For example, the *nssov* overlay may be configured with the OpenLDAP ldap backend by adding the following to slapd.conf:

```
include <path to>nis.schema
moduleload <path to>nssov.la
database    ldap
overlay     nssov

nssov-ssd passwd    ldap:///ou=users,dc=example,dc=com??one
nssov-ssd shadow    ldap:///ou=users,dc=example,dc=com??one
nssov-ssd group     ldap:///ou=group,dc=example,dc=com??one
nssov-ssd hosts     ldap:///ou=hosts,dc=example,dc=com??one
nssov-ssd services  ldap:///ou=services,dc=example,dc=com??one
nssov-ssd networks  ldap:///ou=networks,dc=example,dc=com??one
nssov-ssd protocols ldap:///ou=protocols,dc=example,dc=com??one
nssov-ssd rpc       ldap:///ou=rpc,dc=example,dc=com??one
nssov-ssd ethers    ldap:///ou=hosts,dc=example,dc=com??one
nssov-ssd netgroup  ldap:///ou=netgroup,dc=example,dc=com??one
nssov-ssd aliases   ldap:///ou=aliases,dc=example,dc=com??one
```

If the local database is actually a proxy to a foreign LDAP server, mapping of schema may be needed. Simple attribute substitutions may be performed using the following:

```
nssov-map <service> <original attribute> <new attribute>
```

The overlay also supports dynamic configuration under “cn=config”. The layout of an example configuration entry is as follows:

```
dn: olcOverlay={0}nssov,olcDatabase={1}hdb,cn=config
objectClass: olcOverlayConfig
objectClass: olcNssOvConfig
olcOverlay: {0}nssov
olcNssSsd: passwd ldap:///ou=users,dc=example,dc=com??one
olcNssMap: passwd uid accountName
```

which enables the passwd service and uses the accountName attribute to fetch what is usually retrieved from the uid attribute.

PAM authentication, account management, session management, and password management are supported.

The Proxy Cache Engine

The OpenLDAP Proxy Cache engine was designed to improve the responsiveness of the ldap and meta backends. Instead of caching only individual entries, the proxy cache stores data and semantic information corresponding to recently answered queries. The cache manager implements the following three algorithms:

- Query containment algorithm decides whether an incoming search request is semantically contained in any of the recently answered queries, e.g., (shoesize >= 9) is contained in (shoesize >=8). A contained query is answerable from the cache.
- Cache replacement algorithm determines when a query and entries should be removed from the cache.
- Consistency control algorithm maintains the consistency between cached data and the corresponding information stored in persistent data store.

The proxy cache handles a search query by first determining whether it is contained in any cached search filter. Contained requests are answered from the proxy cache's local database. Other requests are passed on to the underlying ldap or meta backend and processed as usual. The LDAP matching rules and syntaxes are used while comparing assertions for query containment.

To simplify the query containment implementation, a list of cacheable *templates* (defined below) is specified at configuration time. A query is cached or answered only if it belongs to one of these templates. The entries corresponding to cached queries are stored in the proxy cache local database, Berkeley DB or Memory-Mapped Database [9], while their associated meta-information (filter, scope, base, attributes) is stored in main memory. The design of Memory-Mapped Database improves over Berkeley DB on cache management and lock management.

The string representation of prototype filters is similar to LDAP filters [10], except that the assertion values are missing. A template is a prototype filter for generating LDAP search requests. Search filters are templates associated with their respective list of attribute values. Examples of prototype filters are (sn=) and (&(sn=)(givenname=)) which are instantiated by search filters (sn=Doe) and (&(sn=Doe)(givenname=John)) respectively.

The cache replacement policy removes the least recently used (LRU) query and entries belonging to only that query. Queries are allowed a maximum time to live (TTL) in the cache thus providing weak consistency. A background task periodically checks the cache for expired queries and removes them.

The following directive enables proxy caching and defines the configuration parameters:

```
proxycache  <db> <maxentries> <nattrsets> <entrylimit> <period>
```

The <db> parameter specifies the underlying database type which is used to hold the cache entries. The <maxentries> parameter specifies the total number of entries that may be held in the cache. The <nattrsets> parameter specifies the total number of attribute sets that may be defined. The <entrylimit> parameter specifies the maximum number of entries in a cacheable query. The <period> parameter specifies the consistency checking period (in seconds). In each check period, queries with expired TTL are removed.

The proxyAttrSet directive is used to associate a set of attributes to an index:

```
proxyAttrSet <index> <attrs …>
```

The proxyTemplate directive further associates a cacheable prototype filter and the time-to-live (TTL) parameter with an index of an attribute set:

```
proxyTemplate  <prototype filter> <attrset_index> <TTL>
```

The following sample SLAPD configuration defines the proxy attribute sets and proxy template for user passwd and group services.

```
overlay proxycache
proxycache  bdb 100000 11 1000 100
# posixAccount
proxyAttrset 0 cn uid uidNumber gidNumber homeDirectory userPassword loginShell gecos
description objectClass
# shadowAccount
proxyAttrset 1 uid userPassword shadowLastChange shadowMin shadowMax shadowWarning
shadowInactive shadowExpire shadowFlag description objectClass
# posixGroup
proxyAttrset 2 cn gidNumber userPassword memberUid uniqueMember description objectClass

# proxy templates
proxyTemplate (&(objectClass=)(uid=))             0 3600
proxyTemplate (&(objectClass=)(uidNumber=))   0 3600
proxyTemplate (objectClass=)                           0 3600
proxyTemplate (&(objectClass=)(uid=))             1 3600
proxyTemplate (&(objectClass=)(cn=))               2 3600
proxyTemplate (objectClass=)                           2 3600
proxyTemplate (&(objectClass=)(gidNumber=))   2 3600
proxyTemplate (&(objectClass=)(|(memberUid=)(uniqueMember=)) 2 3600
```

Summary

The unified authentication, authorization, and user administration design offers performance, scalability, and high availability for enterprise deployments, while preserving the compatibility with existing IT infrastructure. The flexible design allows authentication, authorization, and user administration services to be provisioned modularly as business needs grow. The OpenLDAP nssov overlay and Proxy Cache engine offer LDAP client-side caching and disconnected operations, as well as address many robustness issues. The OpenLDAP Memory-Mapped Database improves Proxy Cache in cache and lock management over the traditional Berkeley DB implementations.The architecture has been based on open standards and evolved with the collective efforts of the open source community for the past many years. The implementation is contributed back to the open source community and freely available.

References

[1] The OpenLDAP Project, "OpenLDAP Software 2.4 Administrator's Guide," http://www.openldap.org/, March 28, 2011

[2] K. Zeilenga, J.H. Choi, "The Lightweight Directory Access Protocol (LDAP) Content Synchronization Operation," RFC 4533, June 2006

[3] V. Samar, "Unified login with pluggable authentication modules (PAM)," Proceedings of the 3rd ACM conference on computer and communications security, 1996, pp. 1-10

[4] J. Sermersheim, Ed., "Lightweight Directory Access Protocol (LDAP): The Protocol," RFC 4511, June 2006

[5] T. Dierks, C. Allen, "The TLS Protocol Version 1.0," RFC 2246, January 1999

[6] A. Kumar, "The OpenLDAP Proxy Cache," IBM, India Research Lab, http://www.openldap.org/pub/kapurva/proxycaching.pdf

[7] L. Howard, PADL nss_ldap and pam_ldap, www.padl.com

[8] A. de Jong, nss-ldapd and nss-pam-ldapd, http://arthurdejong.org

[9] H. Chu, "MDB: A Memory-Mapped Database and Backend for OpenLDAP," LDAPConf 2011, October 10-11, Heidelberg, Germany

[10] M. Smith, T. Howes, "Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters," RFC 4515, June 2006

Biographies

Ted C. Cheng is a Principal Software Engineer at Symas Corporation. His R&D interests include on-demand computing, provisioning internet services in cloud and enterprise environments, performance, scalability, and high availability. He received his Ph.D. degree in EE-Systems from the University of Southern California in 1993 and has been in enterprise computing and Open Source development ever since. He is a member of IEEE.

Howard Chu is the Chief Architect of OpenLDAP and CTO of Symas Corporation. Prior to founding Symas Corporation, Howard worked at the U. Michigan, JPL, Locus Computing, and Platinum Technology in software development roles. Howard is a prolific contributor to the Open Source software community.

Matthew Hardin is a co-founder of Symas Corporation and currently serves as its VP of Product Engineering. Throughout his 30-plus-year career he has acquired in-depth experience with embedded OS development, communications, server, and client software development in a wide variety of environments.