Network Working Group   H. Chu
Internet-Draft   Symas Corp.
Intended status: Informational   February 28, 2007
Expires: September 1, 2007

Using LDAP Over IPC Mechanisms
draft-chu-ldap-ldapi-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on September 1, 2007.

Copyright Notice

Abstract

When both the LDAP client and server reside on the same machine, communication efficiency can be greatly improved using host- specific IPC mechanisms instead of a TCP session. Such mechanisms can also implicitly provide the client's identity to the server for extremely lightweight authentication. This document describes the implementation of LDAP over Unix IPC that has been in use in OpenLDAP since January 2000, including the URL format used to specify an IPC session.

Table of Contents

1. Introduction

While LDAP is a distributed access protocol, it is common for clients to be deployed on the same machine that hosts the server. Many applications are built on a tight integration of the client code and a co-resident server. In these tightly integrated deployments, where no actual network traffic is involved in the communication, the use of TCP/IP is overkill. Systems like Unix offer native IPC mechanisms that still provide the stream-oriented semantics of a TCP session, but with much greater efficiency.

Since January 2000, OpenLDAP releases have provided the option to establish LDAP sessions over Unix Domain sockets as well as over TCP/IP. Such sessions are inherently as secure as TCP loopback sessions, but they consume fewer system resources, are much faster to establish and tear down, and they also provide secure identification of the client without requiring any additional passwords or other credentials.

2. Conventions

Imperative keywords defined in [RFC2119] are used in this document, and carry the meanings described there.

3. Motivation

Many LDAP sessions consist of just one or two requests. Connection setup and teardown can become a significant portion of the time needed to process these sessions. Also under heavy load, the constraints of the 2MSL limit in TCP become a bottleneck. For example, a modest single processor dual-core AMD64 server running OpenLDAP can handle over 32,000 authentication requests per second on 100Mbps ethernet, with one connection per request. Connected over a host's loopback interface, the rate is much higher, but connections get completely throttled in under one second, because all of the host's port numbers have been used up and are in TIME_WAIT state. So even when the TCP processing overhead is insignificant, the constraints imposed in [RFC0793] create an artificial limit on the server's performance. No such constraints exist when using IPC mechanisms instead of TCP.

4. User-Visible Specification

The only change clients need to implement to use this feature is to use a special URL scheme instead of an ldap:// URL when specifying the target server. Likewise, the server needs to include this URL in the list of addresses on which it will listen.

4.1. URL Scheme

The "ldapi:" URL scheme is used to denote an LDAP over IPC session. The address portion of the URL is the name of a Unix Domain socket, which is usually a fully qualified Unix filesystem pathname. Slashes in the pathname must be percent-encoded as described in section 2.1 of [RFC3986] since they do not represent URL path delimiters in this usage. E.g., for a socket named "/var/run/ldapi" the server URL would be "ldapi://%26var%26run%26ldapi/". In all other respects, an ldapi URL conforms to [RFC4516].

If no specific address is supplied, a default address MAY be used implicitly. In OpenLDAP the default address is a compile-time constant and its value is chosen by whoever built the software.

5. Implementation Details

The basic transport uses a stream-oriented Unix Domain socket. The semantics of communication over such a socket are essentially identical to using a TCP session. Aside from the actual connection establishment, no special considerations are needed in the client, libraries, or server.

5.1. Client Authentication

Since their introduction in 4.2 BSD Unix, Unix Domain sockets have also allowed passing credentials from one process to another. Modern systems may provide a server with easier means of obtaining the client's identity. The OpenLDAP implementation exploits multiple methods to acquire the client's identity. The discussion that follows is necessarily platform-specific.

The OpenLDAP library provides a getpeereid() function to encapsulate all of the mechanisms used to acquire the identity.

On FreeBSD and MacOSX the native getpeereid() is used.

On modern Solaris systems the getpeerucred() system call is used.

On systems like Linux that support the SO_PEERCRED option to getsockopt(), that option is used.

On Unix systems lacking these explicit methods, descriptor passing is used. In this case, the client must send a message containing the descriptor as its very first action immediately after the socket is connected. The descriptor is attached to an LDAP Abandon Request [RFC4511] with message ID zero, whose parameter is also message ID zero. This request is a pure no-op, and will be harmlessly ignored by any server that doesn't implement the protocol.

For security reasons, the passed descriptor must be tightly controlled. The client creates a pipe and sends the pipe descriptor in the message. The server receives the descriptor and does an fstat() on it to determine the client's identity. The received descriptor MUST be a pipe, and its permission bits MUST only allow access to its owner. The owner uid and gid are then used as the client's identity.

Note that these mechanisms are merely used to make the client's identity available to the server. The server will not actually use the identity information unless the client performs a SASL Bind [RFC4513] using the EXTERNAL mechanism. I.e., as with any normal LDAP session, the session remains in the anonymous state until the

client issues a Bind request.

5.2. Other Platforms

It is possible to implement the corresponding functionality on Microsoft Windows-based systems using Named Pipes, but thus far there has been no demand for it, so the implementation has not been written. These are brief notes on the steps required for an implementation.

The Pipe should be created in byte-read mode, and the client must specify SECURITY_IMPERSONATION access when it opens the pipe. The server can then retrieve the client's identity using the GetNamedPipeHandleState() function.

Since Windows socket handles are not interchangeable with IPC handles, an alternate event handler would have to be provided instead of using Winsock's select() function.

6. Security Considerations

This document describes a mechanism for accessing an LDAP server that is co-resident with the client machine. As such, it is inherently immune to security issues associated with using LDAP across a network. The mechanism also provides a means for a client to authenticate itself to the server without exposing any sensitive passwords. The security of this authentication is equal to the security of the host machine.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2717] Petke, R. and I. King, "Registration Procedures for URL Scheme Names", BCP 35, RFC 2717, November 1999.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4511] Sermersheim, J., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, June 2006.

[RFC4513] Harrison, R., "Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms", RFC 4513, June 2006.

[RFC4516] Smith, M. and T. Howes, "Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator", RFC 4516, June 2006.

7.2. Informative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

Appendix A. IANA Considerations

This document satisfies the requirements of [RFC2717] for registration of a new URL scheme.

Author's Address

Howard Chu
Symas Corp.
18740 Oxnard Street, Suite 313A
Tarzana, California 91356
USA

Phone: +1 818 757-7087
Email: hyc@symas.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).