POLSKO-JAPOŃSKA WYŻSZA SZKOŁA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Sieci Komputerowych

Specjalizacja: Sieci Komputerowe

Aleksander Adamowski

Nr albumu 1869

KrbLDAP - A new secure authentication concept

KrbLDAP - Koncepcja nowego bezpiecznego mechanizmu autentykacji

Master's thesis

Written under the direction of dr. Wojciech Mąka

Praca magisterska

Napisana pod kierunkiem dr. Wojciecha Mąki

Warszawa, luty 2012

Table of Contents

1 Introduction

This paper presents the proposed extension to the LDAP protocol that provides a mechanism for transporting Kerberos messages between a client and a server.

The first chapter, “Discussion of the currently used secure authentication and authorization protocols“, provides introductory information about the present choice of available secure network authentication protocols. It also lists chosen implementations of Kerberos and LDAP protocols, whose integration mechanism is proposed by this paper.

The second chapter, “Applications of Kerberos and LDAP protocols”, discusses uses of the two named protocols and provides necessary background for the next chapter.

The third chapter, “Problems resulting from the intersection of both protocols' functionality“, discusses the problems stemming from the fact that protocols have related uses, but were evolving separately and are complex to integrate in a single solution.

The fourth chapter,“The proposal to merge both protocols“, proposes a solution to the discussed problem by integrating Kerberos with LDAP by means of embedding Kerberos messages inside LDAP extended operation messages, naming the resultant protocol “KrbLDAP”. It also discusses the proof of concept implementation which has been developed during the course of work on this paper.

The fifth chapter,“Tests of the proof of concept solution“, elaborates on the carrying out of integration testing using the proof of concept implementation and discusses its results.

The final chapter, “Summary and Conclusion”, summarizes the results and presents directions for future work.

The appendices that follow contain the actual source code of the proof of concept KrbLDAP implementation.

2 Wstęp w języku polskim

Niniejsza praca przedstawia propozycję rozszerzenia protokołu LDAP, które dostarcza mechanizmu transportu dla komunikatów Kerberos wymienianych pomiędzy klientem a serwerem.

Pierwszy rozdział, “Discussion of the currently used secure authentication and authorization protocols“, dostarcza informacji wprowadzających na temat obecnego wyboru dostępnych bezpiecznych sieciowych protokołów uwierzytelniania. Zawiera on również opis wybranych protokołów - Kerberos oraz LDAP, dla których w niniejszej pracy proponowany jest nowy mechanizm integracyjny.

Drugi rozdział, “Applications of Kerberos and LDAP protocols”, omawia zastosowania obydwu protokołów, dostarczając niezbędnych podstaw merytorycznych dla następnego rozdziału.

Rozdział trzeci, “Problems resulting from the intersection of both protocols' functionality“, opisuje problemy wynikające z faktu, że protokoły te posiadają powiązane zastosowania, ale ewoluowały oddzielnie i są w związku z tym trudne do zintegrowania w ramach pojedynczego rozwiązania.

Rozdział czwarty, “The proposal to merge both protocols“, przedstawia propozycję rozwiązania opisanego problemu poprzez integrację Kerberos i LDAP poprzez osadzenie komunikatów Kerberos wewnątrz komunikatów rozszerzonych operacji LDAP, nadając wynikowemu protokołowi nazwę „KrbLDAP”. Omawia on też eksperymentalną implementację, która powstała podczas pisania niniejszej pracy jako dowód koncepcji.

Rozdział piąty, “Tests of the proof of concept solution“, rozwija temat przeprowadzonych testów integracyjnych wykorzystujących wspomnianą eksperymentalną implementację, oraz omawia ich wyniki.

Ostatni rozdział, “Summary and Conclusion”, podsumowuje wyniki i przedstawia możliwe kierunki przyszłych prac nad tematem.

Następujące dalej aneksy zawierają właściwy kod źródłowy eksperymentalnej implementacji KrbLDAP, stanowiącej dowód koncepcji.

3 Glossary

For the purposes of this paper, the following definitions of given terms will apply:

ASN.1, Abstract Syntax Notation One	A standard for representing structured data using a flexible notation, widely used in telecommunication and networking. It is used, with the aim of interoperability: • as a basis for network protocols like Kerberos, LDAP or SNMP by providing standardized way to encode protocol data units, • in public key infrastructure technologies for encoding X.509 certificates and other types of cryptographic documents, • as a basis for Fast Infoset – an alternate binary encoding standard for XML documents, It's also employed in many other applications.
Authentication	The act of confirming a user's identity to a computer system.
Authorization	The act of confirming an authorized user's rights to specific resources.
BER, Basic Encoding Rules	One of defined encoding formats used to represent ASN.1 structures as binary data. In LDAP (but not in Kerberos), used for encoding messages' protocol data units (PDUs).
Client	An application or system that accesses a service provided by a server.
DER, Distinguished Encoding Rules	One of defined encoding formats used to represent ASN.1 structures as binary data. A subset of BER (valid DER data is also valid BER, but not necessarily the other way around). Used in LDAP and Kerberos for encoding protocol data units (PDUs).
PDU, Protocol Data Unit	A top-level unit of structured data in a protocol exchange, typically corresponding to a single message sent over the network between participating sides.
Principal	In Kerberos: an entity, whose identity can be proven (e.g. a user, workstation or network service).
Secure protocols	Network protocols which provide protection against known passive and active attacks.
Server	An application or system that provides a service to a client.

4 Discussion of the currently used secure authentication and authorization protocols

4.1 Protocols currently in wide use

4.1.1 EAP (Extensible Authentication Protocol)

EAP is an authentication protocol most widely used in wireless networks..

The latest version is standardized in RFC 5247.

The protocol actually serves the role of a framework that can encapsulate and transport various authentication mechanisms, including Kerberos and LDAP simple bind.

4.1.2 Kerberos

Kerberos is a secure authentication protocol designed in the eighties of XX century at MIT (Massachusets Institute of Technology) during the Athena project.

The name has been derived from the Greek myth of Hercules' twelve labors, where the twelfth labor concerned the capture of the three headed dog, named Kerberos (or Cerberus), who guarded the entrance to Hades.

The chosen name “Kerberos” were meant to symbolize the three basic components of security (also known as AAA - three A's): Authentication, Authorization and Accountability. Although the scope of the project had never been expanded beyond authentication, the name has stayed.

Several of its revisions were standardized in RFC documents. The latest, 5th version (at the moment this document was being written), has been described in [RFC 1510] (superseded by the latest [RFC 4120]).

The process of designing the protocol and the security issues that it solves have been graphically described in the [Charon - dialogue] document in a form of dialogue between two persons.

All technical details are specified in [RFC 4120], section 3.

The protocol's principles in the simplest possible variant (without pre-authentication) can be summarized as this:

On the client side, the user inputs a plain text password. His Kerberos client performs a one way function on this password, which results in a hash value. The hash function must be cryptographically strong, which means it should be computationally unfeasible to reverse it given the hash value.

The client then sends a plain text message to the Authentication Service (from now on, referred to by the acronym “AS”), having a type of `KRB_AS_REQ`, that is unencrypted and consists of (among others) the identity (principal name) of the client and the contacted server and a randomly generated nonce that will be used for detecting replay attacks and for associating replies with requests.

At this stage, depending on the configuration, the server may respond with an “additional pre-authentication required” error message (type `KRB_ERROR`), which will result in client trying to pre-authenticate first.

Regardless of whether pre-authentication occurs or not (due to the server demanding it or not), server will eventually respond to a `KRB_AS_REQ` request with a `KRB_AS_REP` reply.

The reply contains (among others) a Ticket Granting Ticket (TGT) to present to the server in further requests, and an encrypted part.

This part's encryption is using the client's secret key, which is known only to the AS and the client due to the password being input. Note that this key has not been transferred through the network.

The encrypted part of the reply contains (among others) a session key that will be shared between client and server, the nonce that must match the value from the request, and the value of the current time on the server.

The ticket is a composite data structure that contains (among others) plain principal name, and its own encrypted part (using the server's secret key which isn't known to the client – therefore this part is opaque to the client).

This way, AS has passed several parts of sensitive authentication data to the client in a secure manner.

Next, the client may choose to perform a similar exchange with a Ticket-Granting Service (TGS), authenticating itself with the freshly obtained TGT, in order to obtain a ticket for a given target network service that uses Kerberos authentication.

By analogy, relevant Kerberos message types are `KRB_TGS_REQ` and `KRB_TGS_REP`.

The exchange is in general principles similar to the one performed against the AS, with the obtained ticket being encrypted by its intended recipient's key (the target network service).

The third message exchange will, as a rule, finally involve the client and the target network service being accessed by the user.

In this case, `KRB_AP_REQ` and `KRB_AP_REP` message types are involved.

The aim of this exchange is for the client to prove its user's identity to the network service being accessed.

The client's request contains (among others) the ticket that the client has in its possession for the given service and a one-time authenticator, constructed using the current timestamp, client's principal name and possibly other data, and encrypted using the session key.

The service, upon successful verification of the `KRB_AP_REQ`, will grant the user access and it may (depending on whether mutual authentication is used) reply with a `KRB_AP_REP` message which authenticates it to the client.

As can be seen, Kerberos is a protocol tailored for performing secure authentication of multiple parties towards other parties. Its messages provide little capacity for additional information that isn't directly related to authentication of participating parties.

4.1.3 LDAP

The LDAP protocol is a directory access protocol modelled on X.500, designed for the simpler TCP/IP network stack (at the moment LDAP was being designed, X.500 required a full OSI stack – currently this requirement no longer applies).

LDAP has been standardized through a number of RFC documents.

The latest version is specified through RFC documents numbered 4510 – 4521.

The protocol is designed to be general purposed.

All messages have a common envelope, named “LDAPMessage”, which has a message ID (used to match responses to requests) and a specification of performed operation's type.

Basic operations include “bind” (which authenticates an LDAP session and assigns a user's identity to it), various types of searches against the directory, directory entry modification requests (add, modify, delete), comparison requests. For asynchronous operations, there's also a special “abandonRequest” operation type that tells the server to interrupt a formerly launched asynchronous operation.

The protocol may be freely extended using special “extended” operations, and this paper proposes using these to transport Kerberos messages.

4.1.4 NTLM (NT LAN Manager)

NTLM is a suite of challenge-response authentication protocols designed by Microsoft and used in its Active Directory product.

It is currently deprecated by Microsoft, as latest versions of Active Directory by default use Kerberos for authentication, and can fall back to NTLM only under specific conditions.

4.1.5 RADIUS

RADIUS is a network authentication, authorization and accountability (AAA) protocol used mainly on network hardware, for controlling access by network administration staff and for roaming network access by users.

It uses UDP as the transport protocol.

It is standardized through RFC documents numbered 2865 – 2869.

It is often integrated with backend Kerberos or LDAP authentication services.

4.1.6 TACACS+

TACACS+, similar to RADIUS, is a network AAA protocol used mainly in network hardware management.

It uses TCP as the transport protocol.

It is standardized through RFC 927 and RFC 1492.

4.2 Available server-side implementations of LDAP and Kerberos protocols

4.2.1 MIT Kerberos

MIT Kerberos has been the first implementation of the Kerberos protocol, developed as the reference implementation to the protocol specification. Both the specification and implementation were the results of Project Athena on MIT.

It is an open source project with the code released under the MIT license.

As the software has been developed in the United States, it is subject to US export control laws governing export of encryption technology.

Citing the MIT Kerberos distribution page:

- *Export of this software from the United States of America may be subject to the Export Administration Regulations of the United States Department of Commerce, currently (October 2003) codified as Title 15 CFR Parts 730-774.*

- *You are responsible for complying with all applicable export regulations, including obtaining an export license if required.*

- *You may not download this software if you are located in, or are a citizen or national of, any country for which the US government prohibits the export of encryption source code, currently (October 2003) Cuba, Iran, Iraq, Libya, North Korea, Sudan, or Syria. (15 CFR Sections 734(b)(3), 740.13(e)(4))*

4.2.2 Heimdal Kerberos

Heimdal Kerberos is a clean room implementation of Kerberos developed in Sweden to avoid US export restrictions (See [OReilly-Krb-Guide]) under a three clause BSD style open source license.

4.2.3 OpenLDAP

OpenLDAP is an open source implementation of LDAP server and client libraries and utilities, released under its own BSD-style license called the OpenLDAP Public License.

Just like Netscape Directory Server, its initial releases in 1998 have been based on the LDAP reference source code from the University of Michigan slapd project.

4.2.4 Netscape / Sun / Red Hat / Fedora / 389 Directory Server

In 1996, Netscape Communications Corporation has hired the staff of the University of Michigan's “slapd” project, which was a long running project facilitating development and evolution of the LDAP protocol.

The project subsequently became known as Netscape Directory Server.

In 1998, Netscape has been acquired by America Online, who sold the intellectual

property related to NDS to Sun Microsystems, while retaining rights akin to ownership. This was a part of the joint marketing and development alliance between AOL and Sun Microsystems named iPlanet.

AOL's rights to NDS codebase were acquired by Red Hat in September 2004.

In 2005, Red Hat has released all the source code under the terms of GNU General Public License, as a part of “Fedora Directory Server” project.

From that time, Red Hat has been releasing commercially supported versions of the LDAP server under the name “Red Hat Directory Server” while managing “Fedora Directory Server” as an open-source project not commercially supported.

In May 2009, “Fedora Directory Server” was re-branded to “389 Directory Server” to avoid direct association with the Fedora Linux distribution.

4.2.5 OpenDS

OpenDS is a pure Java implementation of a LDAP server and client utilities. It's an open source project started by Sun Microsystems in February 2005, released under Sun Microsystems' Common Development and Distribution (CDDL) License.

4.2.6 Apache Directory Server

Apache Directory Server (Apache DS) is a pure Java open source implementation of LDAP and Kerberos servers and client utilities, released under the Apache License 2.0.

It is an Apache Software Foundation project initiated in September 2003.

4.2.7 Novell eDirectory

Novell eDirectory is a commercial LDAP directory server, formerly known as Novell Directory Services (initially release in 1993).

4.2.8 Microsoft Active Directory

Microsoft Active Directory is a commercial identity management server that supports both Kerberos and LDAP protocols in addition to DNS.

4.2.9 Oracle Internet Directory

Oracle Internet Directory (OID) is a scalable commercial LDAP server which uses Oracle Database RDBMS as its data storage backend.

5 Applications of Kerberos and LDAP protocols

The Kerberos protocol is applicable only in the context of authentication - that is, the act of determining the identity of a party that applies for access to a service.

In contrast, the LDAP protocol can be used both in the context of authentication and authorization, since being a generic directory access protocol, it can provide access to group membership information, role assignments, permission maps etc.

Unfortunately, unlike Kerberos, LDAP by itself doesn't provide single sign-on capabilities and authentication on as secure level as Kerberos. The built in LDAP authentication mechanism, the so called “simple bind”, demands supplying a plain text user's distinguished name and password. Kerberos, on the contrary, even in its most basic variant, is based on symmetric encryption and issuing of cryptographic authentication tickets – no passwords themselves are transferred over the network.

Thanks to LDAP version 3 GSSAPI pluggable authentication extensibility, it's possible to integrate both protocols, although in a somewhat cumbersome way. To authenticate, a user obtains a ticket from Kerberos and uses it to prove one's identity for access to LDAP and other services. Kerberos server can even store its database in the LDAP directory itself, possibly working on the same physical server. However, both protocols still work on different network ports and are usually provided by separate software packages, and often don't even share the user database (which introduces potential for serious inconsistencies).

6 Problems resulting from the intersection of both protocols' functionality

The Kerberos and LDAP protocols are often used in a tandem, but their functionalities and responsibilities overlap in some aspects.

Both Kerberos and LDAP require storing information about user accounts in their databases.

To integrate services of both protocols one can apply one of two possible approaches:

- synchronize the contents of both databases (e.g. OpenLDAP + Heimdal Kerberos), which can lead to introduction of inconsistencies between both databases
- use an integrated implementation, that serves Kerberos and LDAP clients using a single common database (e.g. Active Directory).

Both approaches exhibit one common problem: the Kerberos and LDAP protocols function on different network ports, which complicates management of their instances and security (especially accessibility). In the case of separate databases it also becomes a serious concern to keep their contents in a consistent state.

Having related functions served by two different protocols further complicates problem diagnostics and service monitoring, when applying network traffic analysis techniques. In addition to correlating IP packets of a single protocol to take TCP sessions into account, one also has to correlate separate protocol exchanges. To diagnose a problem with user access control, one has to analyze at least 2 message exchanges - both the Kerberos one, used to obtain a ticket, and the LDAP one that directly follows it, that may be used for establishing a user's group memberships or other significant information.

This correlation may be hard because the protocols have distinct origins and fundamental design elements – for example, LDAP uses Distinguished Names to identify users and entities, while Kerberos has Principal Names.

7 The proposal to merge both protocols

In this paper, a merging of both protocols into one modular and extensible resultant protocol is postulated. More specifically, integrating the Kerberos into LDAP by implementing Kerberos message transport on top of LDAP extended operations instead of plain TCP/UDP.

This is possible thanks to two facts:

- the LDAP protocol is extremely extensible and flexible, thanks to the extension mechanism introduced in protocol version 3,
- both protocols are based on abstract syntax notation (ASN.1 standard). Kerberos requires the DER encoding for representing ASN.1 structures, while LDAP allows both BER and DER.

Implementing new functionality on top of LDAP is therefore largely simplified, since message parsing mechanisms are already provided by standard library routines in existing LDAP implementations and there is a mechanism for implementing additional functionality in LDAP.

An opposite, alternative approach has also been identified – implementing LDAP message transport on top of Kerberos pre-authentication, which is a Kerberos extensibility mechanism.

This approach hasn't been chosen for three reasons:

- it's less flexible than LDAP v3 extended operations, and targeted towards Kerberos's specific authentication data flow,
- it's a much less mature element of the protocol, with the generalized framework based on pre-authentication being standardized in [RFC 6113] only in April 2011 (in contrast, LDAPv3 with extended operations has been standardized in [RFC 2251] in December 1997),
- there exist no fully capable open source server implementations that support both Kerberos and LDAP and are designed with extensibility in mind – the only one, Apache Directory Server, has a limited, non-flexible support for Kerberos pre-authentication.

7.1 The proposed model of the new KrbLDAP protocol

In [RFC 4511] section 4.12, a mechanism for defining extended operations is specified. Additional considerations about extensions to LDAP are documented in [RFC 4521].

In the proposed model of KrbLDAP protocol, Kerberos messages in a form identical to specified in [RFC 4120] (ASN.1 structured encoded using DER) would be encapsulated in LDAP messages as extended requests' values.

Kerberos reply messages, generated by the server (acting as a Key Distribution Center), would by analogy be encapsulated in corresponding LDAP extended responses as response values.

From the standpoint of Kerberos client and server, there would be no difference between

Kerberos messages carried by plain TCP and by KrbLDAP.

Note that, while the whole KrbLDAP message, being an LDAP message, can be seen as a BER-encoded ASN.1 structure. The part which corresponds to the Kerberos message would always be DER encoded, but any since DER is a subset of BER, any DER-encoded structure can also be interpreted as BER-encoded one with identical results.

7.2 Prototype protocol implementation

For the purpose of providing a proof of concept implementation, Apache Directory Server, an existing implementation of LDAP and Kerberos server, has been customized to properly serve KrbLDAP extended operation requests.

This customization has been provided as a Java project in the form of a Maven-based artifact, which follows the conventions of Maven artifacts that comprise official Apache DS modules. The Maven `artifactId` is named **`apacheds-krbldap-test`**, and its `groupId` is `org.apache.directory.server`. Since the whole Apache DS project publishes its source code under the Apache 2.0 license, this artifact is also published under that license. The public source code repository for this artifact is made available on the web under the address https://github.com/aadamowski/apacheds-krbldap-test.

The customization consists mainly of a KrbLDAP Authentication Service handler that takes care of encapsulating/deencapsulating Kerberos messages in LDAP extended operations, and helper classes required for the handler's operation. The code is located in the `src/main/java` subdirectory of the Maven project.

To test the interoperation of the client and the server implementation, an integration test suite has been developed, partly based on Apache Directory Server's integration test framework (to set up a temporary server for each test run), implemented in Java, and partly on pam_krb5 PAM module's test suite which is based on shell scripts.

This integration test suite has been developed first, before beginning work on client and server implementation, so that progress of the implementation can be guided by the test's assertions.

The Java part of the test suite is located in the aforementioned **`apacheds-krbldap-test`** project, in `src/test/java`.

The pam_krb5-based, shell scripted part of the test suite is located in the customized fork of the pam_krb5 project, in the `tests` subdirectory. Since the pam_krb5 project's source code is distributed under a dual LGPL/BSD license, the customized version uses the same dual license. The customized version is named **`pam_krb5-krbldap`** and is published on the web as a fork of the original project under the address https://github.com/aadamowski/pam_krb5-krbldap.

During the work on this paper, initially, most of the client logic implementation has been attempted on the grounds of pam_krb5 module, in the form of C language code. The ASN.1 structures comprising a Kerberos message were being constructed programmatically by hand. It quickly became obvious that, due to the complexity of modern version of Kerberos protocol, this approach is impractical.

Since the same work had already been done previously in the form of the open source

MIT Kerberos 5 library implementation, a different approach has been chosen. The KrbLDAP client implementation, still residing in pam_krb5, has been made to use C functions from MIT libkrb5 public API to perform most of the work involving Kerberos message preparation. It turned out that not all required functions are present in libkrb5 API, so pam_krb5 module's source has been placed inside a libkrb5 source tree to be built with access to its internal symbols. As more and more elements of libkrb5 internal API came into use, it became obvious that this approach is also impractical and not correct. The MIT Kerberos library is the best place to put the KrbLDAP client logic, alongside with the old-style Kerberos client logic.

For these reasons, all of the code for KrbLDAP client from pam_krb5 has been abandoned. In the Github pam_krb5-krbldap repository, it's still present on an unused branch named “krbldap”. The new work is on the branch named “krbldap-simple”.

Then, in the MIT libkrb5 source, the C function responsible for sending a Kerberos message to the server has been located (residing in `src/lib/krb5/os/sendto_kdc.c`). Using C preprocessor “`#ifdef`” statements, a new optional logic has been added that implements KrbLDAP transport for Kerberos message, replacing TCP and UDP transports.

Currently, this logic is switched on using a compile-time define (`KRB5_KRBLDAP`). If the define is present, the whole Kerberos traffic by the resulting compiled krb5 library uses only KrbLDAP. If the future full implementation should be practical to use, a runtime configuration option (for the `krb5.conf` client library configuration file) should be employed for choosing the desired transport protocol on a per-server basis (and possibly globally in a `libdefaults` section of the configuration file).

Source code of this customized fork of the MIT Kerberos 5 library has also been published. Since the original version is published under the MIT open source license, the forked version also employs the same license. It has been published under the name **`krb5-krbldap`** on the web under the address https://github.com/aadamowski/krb5-krbldap.

As can be seen, all of the proof of concept KrbLDAP customizations have been made to open source projects, and published under their respective licenses. Code has been published in Git source control system repositories, published on Github. Github is a free web-based source code hosting service, free to use for open source projects.

The main rationale for publishing the cusomizations using Git on Github is that this allows for the changes to those large code bases to be tracked effectively, thanks to Git's inherent space efficiency for source code changes. Additionally, since the original projects are also on Github, only differential representation of code changes had to be stored.

Git source control system also allows for easy merging of backported upstream changes from respective original projects, making the maintenance of KrbLDAP cusomizations easier.

8 Tests of the proof of concept solution

8.1 Integration testing framework

The main point of entry for the KrbLDAP proof of concept integration test is the test suite contained in the class:
`pl.org.olo.krbldap.apacheds.test.KrbLdapIntegrationTest`.

The class contains two test methods:

- `testShouldPerformSuccessfulKrb5Authentication`, which performs the role of scientific control by performing an ordinary Kerberos AS request using standard, system-wide non customized libkrb5
- `testShouldPerformSuccessfulKrbLdapAuthentication`, which performs actual integration test by performing a KrbLDAP AS request using a customized libkrb5 installed in a separate subdirectory using a compile-time configured installation prefix.

There are the following prerequisites to perform the integration test:

- Computer with a POSIX-like system (e.g. Linux, the solution has been tested on Ubuntu).
- Access to the Internet (so that Maven dependencies can be downloaded during the first build of the integration test).
- The **`apacheds-krbldap-test`** Maven source project, placed anywhere in the filesystem.
- The customized **`pam_krb5-krbldap`** module sources, placed and compiled in appropriate location so that test scripts present in its `tests` subdirectory are reachable through pathnames specified in `KrbLdapIntegrationTest`, in `CLIENT_TEST_SCRIPT_KRBLDAP` and `CLIENT_TEST_SCRIPT_KRB5` constants.
- The customized **`krb5-krbldap`** library sources, configured for build with the "`--prefix=/var/soft/PAM/usr --enable-krbldap`" options and built in a separate build tree from sources (as specified in its build instructions in the `doc/build.texinfo` file), then installed in the subdirectory specified by the aforementioned configure prefix.
  Note that the shell script `tests/run-tests-krbldap-direct.sh` from **`pam_krb5-krbldap`**, which runs the client side portion of KrbLDAP integration test, specifies the full path to the installation location of the customized krb5 library, by setting `LD_LIBRARY_PATH=/var/soft/PAM/usr/lib`.
  Therefore, in case of any modifications to these installation locations, the paths in `tests/run-tests-krbldap-direct.sh` and in arguments to **`krb5-krbldap`**'s configure script must be kept in accord.

8.2 Integration test results

8.2.1 Plain krb5 control test result

This section relates to the test implemented by the following method:

`testShouldPerformSuccessfulKrb5Authentication`

When running the control test that uses plain krb5, the following results are output on the terminal:

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running pl.org.olo.krbldap.apacheds.test.KrbLdapIntegrationTest
kdcServer: KDCServer[DefaultKrbServer], listening on :
    UdpTransport[<localhost:8800>], backlog=0, nbThreads = 0]
    TcpTransport[<localhost:8800>], backlog=50, nbThreads = 3]

Testing started. Check your system's syslog (facility AUTH) for any messages
from the PAM module.
Test script: /var/soft/PAM/krb5-github/src/pam_krb5/tests/run-tests-krb5-
direct.sh
KRB5_CONFIG: /var/soft/PAM/krb5-github/src/pam_krb5/tests/config/krb5.conf
Calling module `/var/soft/PAM/krb5-
github/src/pam_krb5/tests/../src/.libs/pam_krb5.so'.
`Password: ' -> `bar'
[28148] 1327857642.68125: Getting initial credentials for alice@EXAMPLE.COM
[28148] 1327857642.93663: Setting initial creds service to
[28148] 1327857642.93714: Sending request (197 bytes) to EXAMPLE.COM
[28148] 1327857642.94285: Initiating TCP connection to stream ::1:8800
[28148] 1327857642.94346: Terminating TCP connection to stream ::1:8800
[28148] 1327857642.94370: Initiating TCP connection to stream 127.0.0.1:8800
[28148] 1327857642.94473: Sending TCP request to stream 127.0.0.1:8800
[28148] 1327857642.277778: Received answer from stream 127.0.0.1:8800
[28148] 1327857642.278094: Response was from master KDC
[28148] 1327857642.278115: Received error from KDC: -1765328368/KDC has no
support for padata type
AUTH    7       Authentication failure
```

Note that the test fails due to an incompatibility between MIT kerberos 5 client library with settings from pam_krb5 test suite, and the Apache DS's KDC server with its default settings.

The problem is in Apache DS KDC logic – against the recommendations of Kerberos RFC documents, the KDC rejects messages with unknown pre-authentication types (hence the “no support for padata type” error). The problem has been put through a discussion on the MIT Kerberos developers mailing list, and a Kerberos developer, Greg Hudson, has explained that “KDC implementations must ignore unrecognized padata fields. This requirement is a fundamental basis of krb5 protocol extensibility; there

is really no way implementations can work around or accomodate a failure to do so.”

Subsequently, the problem has been reported to Apache DS team on its developers

mailing list. Another minor problem with incompatible default encryption types has also been reported.

As a result, a full review of its Kerberos server code is planned by Apache DS team, so that this and any other issues where Apache DS doesn't follow [RFC 4120] will be addressed.

Meanwhile, the control test ends with a failure and hangs, resulting in the need to interrupt the test suite execution manually and run only one of both tests at a time (the other has to be commented out in the source code).

On the other hand, this demonstrates how far can an analogous KrbLDAP exchange progress if the KrbLDAP implementation is correct and the `testShouldPerformSuccessfulKrbLdapAuthentication` test method is run.

8.2.2 Actual KrbLDAP test result

This section relates to the test implemented by the following method:

`testShouldPerformSuccessfulKrbLdapAuthentication`

When running the actual KrbLDAP test, the following results are output on the terminal:

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running pl.org.olo.krbldap.apacheds.test.KrbLdapIntegrationTest
kdcServer: KDCServer[DefaultKrbServer], listening on :
    UdpTransport[<localhost:8800>], backlog=0, nbThreads = 0]
    TcpTransport[<localhost:8800>], backlog=50, nbThreads = 3]

Testing started. Check your system's syslog (facility AUTH) for any messages
from the PAM module.
Test script: /var/soft/PAM/krb5-github/src/pam_krb5/tests/run-tests-krbldap-
direct.sh
GNU gdb (Ubuntu/Linaro 7.3-0ubuntu2) 7.3-2011.08
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>...
Reading symbols from /var/soft/PAM/krb5-
github/src/pam_krb5/tests/tools/pam_harness...done.
[Thread debugging using libthread_db enabled]
Calling module `/var/soft/PAM/krb5-
github/src/pam_krb5/tests/../src/.libs/pam_krb5.so'.
`Password: ' -> `bar'
[28519] 1327859146.83809: Getting initial credentials for alice@EXAMPLE.COM
[28519] 1327859146.205649: Setting initial creds service to
krbtgt/EXAMPLE.COM@EXAMPLE.COM
krb5_sendto_kdc(197@0x63cca0, "EXAMPLE.COM", use_master=0, tcp_only=0)
```

```
[28519] 1327859146.205763: Sending request (197 bytes) to EXAMPLE.COM
in module_locate_server
ran off end of plugin list
module_locate_server returns -1765328135
looking in krb5.conf for realm EXAMPLE.COM entry kdc; ports 88,750
found 1 entries under 'kdc'
entry 0 is 'hostname:8800'
krb5int_locate_server found 1 addresses
k5_locate_kdc retval: [0]
using krbldap protocol to send kerberos message.
Trying server [hostname] on port [1389].
LDAP URL: [ldap://hostname:1389]
rc: [0], LDAP_SUCCESS: [0]
Before ldap_extended_operation_s:
ldap_extended_operation_s
ldap_extended_operation
ldap_send_initial_request
ldap_new_connection 1 1 0
ldap_int_open_connection
ldap_connect_to_host: TCP hostname:1389
ldap_new_socket: 10
ldap_prepare_socket: 10
ldap_connect_to_host: Trying ::1 1389
ldap_pvt_connect: fd: 10 tm: -1 async: 0
ldap_close_socket: 10
ldap_new_socket: 10
ldap_prepare_socket: 10
ldap_connect_to_host: Trying 127.0.0.1:1389
ldap_pvt_connect: fd: 10 tm: -1 async: 0
ldap_open_defconn: successful
ldap_send_server_request
ldap_result ld 0x63d330 msgid 1
wait4msg ld 0x63d330 msgid 1 (infinite timeout)
wait4msg continue ld 0x63d330 msgid 1 all 1
** ld 0x63d330 Connections:
* host: hostname  port: 1389  (default)
  refcnt: 2  status: Connected
  last used: Sun Jan 29 18:45:46 2012


** ld 0x63d330 Outstanding Requests:
 * msgid 1,  origid 1, status InProgress
   outstanding referrals 0, parent count 0
  ld 0x63d330 request count 1 (abandoned 0)
** ld 0x63d330 Response Queue:
   Empty
  ld 0x63d330 response count 0
ldap_chkResponseList ld 0x63d330 msgid 1 all 1
ldap_chkResponseList returns ld 0x63d330 NULL
ldap_int_select
```

```
read1msg: ld 0x63d330 msgid 1 all 1
read1msg: ld 0x63d330 msgid 1 message type extended-result
read1msg: ld 0x63d330 0 new referrals
read1msg:  mark request completed, ld 0x63d330 msgid 1
request done: ld 0x63d330 msgid 1
res_errno: 0, res_error: <>, res_matched: <>
ldap_free_request (origid 1, msgid 1)
ldap_parse_extended_result
ldap_parse_result
ldap_msgfree
After ldap_extended_operation_s.
exop rc: [0]
Finished processing on server index [0]. Ending loop.
in module_locate_server
ran off end of plugin list
module_locate_server returns -1765328135
looking in krb5.conf for realm EXAMPLE.COM entry master_kdc; ports 88,750
found 1 entries under 'kdc'
entry 0 is 'hostname:8800'
krb5int_locate_server found 1 addresses
[28519] 1327859146.633314: Response was from master KDC
[28519] 1327859146.633352: Received error from KDC: -1765328368/KDC has no
support for padata type
AUTH    7       Authentication failure
[Inferior 1 (process 28519) exited normally]
No stack.
```

As can be seen thanks to configured debug settings, the LDAP client library is invoked to send the Kerberos message embedded in an LDAP extended operation (the `ldap_extended_operation_s` function call).

Also, in the log file created during the test, parts information show consecutive stages of extraction of the Kerberos message from LDAP message and its decoding and processing:

```
...


[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.extras.extended.ads_impl.krbLdap.KrbLdapDecoder] -
KrbLDAP message data to be decoded:
[6a81c23081bfa103020105a20302010aa30e300c300aa10
402020095a2020400a481a230819fa00703050040800000a1123010a003020101a10930071b05616
c696365a20d1b0b4558414d504c452e434f4da320301ea003020102a11730151b066b72627467741
b0b4558414d504c452e434f4da411180f32303132303132393137343534365aa511180f323031323
03133303033343534365aa611180f32303132303133303033343534365aa706020441fb2a3ea80e3
00c020112020111020110020117]
[2012-01-29 18:45:46] DEBUG [org.apache.directory.shared.asn1.ber.Asn1Decoder] -
>>>========================================
[2012-01-29 18:45:46] DEBUG [org.apache.directory.shared.asn1.ber.Asn1Decoder] -
--> Decoding a PDU
[2012-01-29 18:45:46] DEBUG [org.apache.directory.shared.asn1.ber.Asn1Decoder] -
>>>----------------------------------------
```

```
...

[2012-01-29 18:45:46] DEBUG
[org.apache.directory.shared.asn1.ber.grammar.AbstractGrammar] - Transition from
state <START_STATE> to state <AS_REQ_STATE>, tag <0x6A>, action : null


...

[2012-01-29 18:45:46] INFO
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
Handling KrbLdap AS request.
[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
LdapSession: [LdapSession : <,...>]
[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
ExtendedRequest: [MessageType : EXTENDED_REQUEST
Message ID : 1
    Extended request
        Request name : '1.3.6.1.4.1.38261.1.1'
pl.org.olo.krbldap.apacheds.extras.extended.KrbLdapRequestImpl@48b3e197]
[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
KerberosMessage contained in ExtendedRequest: AS-REQ
pvno : 5
msg-type : padata : PreAuthenticationData : {
    padata-type: null(0)
}

kdc-req-body : KDCOptions : FORWARDABLE RENEWABLE
cname : PrincipalName : {
    name-type: KRB_NT_PRINCIPAL
    name-string : <'alice'>
}
realm : EXAMPLE.COM
sname : PrincipalName : {
    name-type: KRB_NT_SRV_INST
    name-string : <'krbtgt', 'EXAMPLE.COM'>
}
from : 20120129174546Z
till : 20120130034546Z
rtime : 20120130034546Z
nonce : 1106979390
etype : aes256-cts-hmac-sha1-96 (18) aes128-cts-hmac-sha1-96 (17) des3-cbc-sha1-
kd (16) rc4-hmac (23)

[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
KerberosMessage class: org.apache.directory.shared.kerberos.messages.AsReq
[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
ldapServer available: LdapServer[DefaultLdapServer], listening on :
    TcpTransport[<localhost:1389>], backlog=50, nbThreads = 3]
```

```
[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
PaData list contents:
[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
PaData type value: 0
[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
PaData value: null
[2012-01-29 18:45:46] WARN
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
Zero-type PaData found: PreAuthenticationData : {
    padata-type: null(0)
}


...


[2012-01-29 18:45:46] WARN
[org.apache.directory.server.kerberos.protocol.KerberosProtocolHandler] - KDC
has no support for padata type (16)
org.apache.directory.shared.kerberos.exceptions.KerberosException: KDC has no
support for padata type
        at
org.apache.directory.server.kerberos.kdc.authentication.AuthenticationService.ve
rifyEncryptedTimestamp(AuthenticationService.java:291)
        at
org.apache.directory.server.kerberos.kdc.authentication.AuthenticationService.ex
ecute(AuthenticationService.java:118)
        at
org.apache.directory.server.kerberos.protocol.KerberosProtocolHandler.messageRec
eived(KerberosProtocolHandler.java:159)
        at
pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler.handleEx
tendedOperation(KrbLdapAuthServiceHandler.java:143)
        at
pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler.handleEx
tendedOperation(KrbLdapAuthServiceHandler.java:55)
        at
org.apache.directory.server.ldap.handlers.ExtendedHandler.handle(ExtendedHandler
.java:57)
        at
org.apache.directory.server.ldap.handlers.ExtendedHandler.handle(ExtendedHandler
.java:37)
        at
org.apache.directory.server.ldap.handlers.LdapRequestHandler.handleMessage(LdapR
equestHandler.java:221)
        at
org.apache.directory.server.ldap.handlers.LdapRequestHandler.handleMessage(LdapR
equestHandler.java:56)
        at
org.apache.mina.handler.demux.DemuxingIoHandler.messageReceived(DemuxingIoHandle
r.java:232)
        at
org.apache.directory.server.ldap.LdapProtocolHandler.messageReceived(LdapProtoco
lHandler.java:226)
        at
```

```
org.apache.mina.core.filterchain.DefaultIoFilterChain$TailFilter.messageReceived
(DefaultIoFilterChain.java:716)
        at
org.apache.mina.core.filterchain.DefaultIoFilterChain.callNextMessageReceived(De
faultIoFilterChain.java:434)
        at
org.apache.mina.core.filterchain.DefaultIoFilterChain.access$1200(DefaultIoFilte
rChain.java:46)
        at
org.apache.mina.core.filterchain.DefaultIoFilterChain$EntryImpl$1.messageReceive
d(DefaultIoFilterChain.java:796)
        at
org.apache.mina.core.filterchain.IoFilterEvent.fire(IoFilterEvent.java:75)
        at org.apache.mina.core.session.IoEvent.run(IoEvent.java:63)
        at
org.apache.mina.filter.executor.UnorderedThreadPoolExecutor$Worker.runTask(Unord
eredThreadPoolExecutor.java:480)
        at
org.apache.mina.filter.executor.UnorderedThreadPoolExecutor$Worker.run(Unordered
ThreadPoolExecutor.java:434)
        at java.lang.Thread.run(Thread.java:722)
[2012-01-29 18:45:46] DEBUG
[org.apache.directory.server.kerberos.protocol.KerberosProtocolHandler] -
Responding to request with error:
        explanatory text:      KDC has no support for padata type
        error code:            KDC has no support for padata type
        clientPrincipal:       null@null
        client time:           null
        serverPrincipal:       PrincipalName : {
    name-type: KRB_NT_PRINCIPAL
    name-string : <'krbtgt', 'EXAMPLE.COM'>
}@EXAMPLE.COM
        server time:           20120129174546Z
[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
message received in session: KRB-ERROR : {
    pvno: 5
    msgType: KRB_ERROR
    sTime: 20120129174546Z
    susec: 0
    errorCode: KDC has no support for padata type
    realm: EXAMPLE.COM
    sName: PrincipalName : {
    name-type: KRB_NT_PRINCIPAL
    name-string : <'krbtgt', 'EXAMPLE.COM'>
}
    eText: KDC has no support for padata type
}

[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
Kerberos message stored in session: KRB-ERROR : {
    pvno: 5
    msgType: KRB_ERROR
```

```
    sTime: 20120129174546Z
    susec: 0
    errorCode: KDC has no support for padata type
    realm: EXAMPLE.COM
    sName: PrincipalName : {
    name-type: KRB_NT_PRINCIPAL
    name-string : <'krbtgt', 'EXAMPLE.COM'>
}
    eText: KDC has no support for padata type
}

[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
Setting Kerberos Reply: KRB-ERROR : {
    pvno: 5
    msgType: KRB_ERROR
    sTime: 20120129174546Z
    susec: 0
    errorCode: KDC has no support for padata type
    realm: EXAMPLE.COM
    sName: PrincipalName : {
    name-type: KRB_NT_PRINCIPAL
    name-string : <'krbtgt', 'EXAMPLE.COM'>
}
    eText: KDC has no support for padata type
}

[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.extras.extended.ads_impl.krbLdap.KrbLdapResponseDec
orator] - Length of Kerberos reply: 131

...

[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.extras.extended.ads_impl.krbLdap.KrbLdapResponseDec
orator] - Encoded Kerberos message as extended response value.

...

[2012-01-29 18:45:46] DEBUG
[pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler] -
Wrote to Ldap IO Session the following response:
KrbLdapResponseImpl{kerberosReply=KRB-ERROR : {
    pvno: 5
    msgType: KRB_ERROR
    sTime: 20120129174546Z
    susec: 0
    errorCode: KDC has no support for padata type
    realm: EXAMPLE.COM
    sName: PrincipalName : {
    name-type: KRB_NT_PRINCIPAL
    name-string : <'krbtgt', 'EXAMPLE.COM'>
```

```
}
    eText: KDC has no support for padata type
}
}

...
```

The end result is the same as in the plain Kerberos control test, thus demonstrating that Kerberos messages have successfully been exchanged by the client and server, embedded in LDAP extended operation messages. The Kerberos reply message has, unfortunately, been an error message, but it has been correctly encoded by Apache DS and sent back to the MIT krb5/pam_krb5 client, which has correctly decoded it from a KrbLDAP response.

8.2.3 Workaround for the “no support for padata type” error

Since, for the needs of the proof of concept KrbLDAP implementation, a custom extended operation handler has been implemented, an idea for working around the pre-authentication problem has surfaced.

The workaround would only apply to KrbLDAP, not plain Kerberos, and would become redundant once the problem gets fixed in upstream version of Apache DS.

The workarount is to detect an unsupported pre-authentication data inside Kerberos requests and remove it before passing on to the integrated Apache DS's KDC.

The following patch achieves this goal:

```
diff --git
a/src/main/java/pl/org/olo/krbldap/apacheds/handlers/extended/KrbLdapAuthService
Handler.java
b/src/main/java/pl/org/olo/krbldap/apacheds/handlers/extended/KrbLdapAuthService
Handler.
index aca9912..629800e 100644
---
a/src/main/java/pl/org/olo/krbldap/apacheds/handlers/extended/KrbLdapAuthService
Handler.java
+++
b/src/main/java/pl/org/olo/krbldap/apacheds/handlers/extended/KrbLdapAuthService
Handler.java
@@ -120,59 +123,67 @@ public class KrbLdapAuthServiceHandler implements
ExtendedOperationHandler<KrbLd
         if (kerberosMessage instanceof AsReq) {
             LOG.debug("PaData list contents:");
             final List<PaData> paDataList = ((AsReq)
kerberosMessage).getPaData();
             final Iterator<PaData> iterator = paDataList.iterator();
             while (iterator.hasNext()) {
                 PaData paData = iterator.next();
                 LOG.debug("  PaData type value: " +
paData.getPaDataType().getValue());
                 LOG.debug("  PaData value: " + paData.getPaDataValue());
                 if (paData.getPaDataType().getValue() == 0) {
```

```
                    LOG.warn("    Zero-type PaData found: " + paData);
+                   // Workaround for
http://mailman.mit.edu/pipermail/krbdev/2012-January/010640.html :
+                   LOG.warn("    removing.");
+                   iterator.remove();
                }
            }
        }
        /** TODO: perform message processing similar in behaviour to
         * {@link
org.apache.directory.server.kerberos.protocol.KerberosProtocolHandler#messageRec
eived}
         */
        final IoSession ldapIoSession = session.getIoSession();
        final KrbLdapAuthServiceHandlerIoSession handlerSession = new
KrbLdapAuthServiceHandlerIoSession();
        handlerSession.setRemoteAddress(ldapIoSession.getRemoteAddress());
```

With this workaround, the KrbLDAP test progresses further:

```
After ldap_extended_operation_s.
exop rc: [0]
Finished processing on server index [0]. Ending loop.
in module_locate_server
ran off end of plugin list
module_locate_server returns -1765328135
looking in krb5.conf for realm EXAMPLE.COM entry master_kdc; ports 88,750
found 1 entries under 'kdc'
entry 0 is 'ania-vaio:8800'
krb5int_locate_server found 1 addresses
[28878] 1327860867.771375: Response was from master KDC
[28878] 1327860867.848657: Received error from KDC: -1765328359/Additional pre-
authentication required
[28878] 1327860867.848736: Processing preauth types: 11
[28878] 1327860867.848749: Selected etype info: etype aes256-cts, salt "(null)",
params ""
[28878] 1327860867.848753: Produced preauth for next request: (empty)
AUTH    7       Authentication failure
```

This time, the KDC responds correctly with an “Additional pre-authentication required” message.

The customized MIT Kerberos library interprets the reply, but for reasons not yet determined, and beyond the scope of this paper, it produces an empty pre-authentication for the next request.

This problem demonstrates that further work on this KrbLDAP integration mechanism requires that incompatibilities between MIT Kerberos library and Apache DS KDC be addressed first.

9 Summary and Conclusion

The goal of this paper was to propose significant simplification of methods of integration for Kerberos and LDAP protocols.

The described customizations to existing Kerberos and LDAP implementations demonstrate a proof of concept integration mechanism, compatible with the attached RFC draft specification.

The proof of concept implementation shows that the goal is fully achievable using reasonable means.

The presented results of integration test show that it's fully possible to transport Kerberos protocol exchange over LDAP messages and achieve a closer and simpler integration of the two protocols than the ones employed in wide use today.

Possible directions for continuation of presented work include:

- Solving incompatibilities between MIT Kerberos v5 client implementation and Apache DS KDC.
- Achieving full ticket obtainment functionality through KrbLDAP between MIT libkrb5-based clients and Apache DS KrbLDAP server.
- Implementing transport protocol configurability in MIT Kerberos v5 library through a krb5.conf file. The defaults should cause the Kerberos client to behave in a manner identical to the pre-KrbLDAP library version – it should use plain TCP or UDP for transporting Kerberos messages. However, if a dedicated transport protocol option is used (either on a per server entry basis, or globally in the `[libdefaults]` section), KrbLDAP transport may be activated.
- Extending the KrbLDAP protocol specification so that LDAP server is allowed to perform binding on the basis of Kerberos message exchange transported over KrbLDAP. For example, during a TGS exchange, the user principal's identity is being proved and this information can be used by the KrbLDAP server to authenticate the user and perform an implied bind on his LDAP connection. It remains to be discussed whether an explicit LDAP bind operation should be required in addition to a KrbLDAP message exchange. A full analysis of any security implications needs to be performed.
- Implementing KrbLDAP support in other server implementations of Kerberos (e.g. MIT, Heimdal) and LDAP (e.g. OpenDS, OpenLDAP, 389 Directory Server).

10 Bibliography

Bibliography

RFC 1510: http://tools.ietf.org/html/rfc1510 1993
RFC 4120: http://tools.ietf.org/html/rfc4120 2005
Charon - dialogue: http://web.mit.edu/kerberos/www/dialogue.html 1988
OReilly-Krb-Guide: http://books.google.pl/books?id=mVIA-CQHDjEC
RFC 6113: http://tools.ietf.org/html/rfc6113 2011
RFC 2251: http://tools.ietf.org/html/rfc2251 1997
RFC 4511: http://tools.ietf.org/html/rfc4511 2006
RFC 4521: http://tools.ietf.org/html/rfc4521 2006

Appendix A: The formal specification of the KrbLDAP protocol as RFC draft document

Presented below is a text of Internet draft document formatted according to RFC rules, describing the KrbLDAP method of transporting Kerberos messages:

```
Network Working Group                                          A. Adamowski
Internet-Draft                                              January 29, 2012
Intended status: Standards Track
Expires: August 1, 2012


                      KrbLDAP - Kerberos over LDAP
                  draft-ietf-ldapext-krb-wg-krbldap-00

Abstract

   This document describes an LDAP extended operation to allow
   transporting Kerberos messages and their corresponding replies to/
   from a Kerberos Key Distribution Center (KDC) instance embedded in an
   LDAP server.  In user organizations, there exist several client and
   server setups, where LDAP is deployed alongside Kerberos in a closely
   integrated manner.  In such setups, both LDAP and Kerberos server
   usually share the same user database.  The proposed LDAP extension
   would provide for radical simplification of such setups, by
   eliminating the necessity to provide Kerberos and LDAP protocols over
   separate network transports, listening on different TCP (or, in the
   case of Kerberos, UDP) ports and embedding the Kerberos Key
   Distribution Center in the LDAP server process.  In addition, such
   integration opens way for further future simplifications, e.g.
   enabling LDAP layer to infer user identity information directly from
   the result of Kerberos message exchange and perform Kerberos-based
   binds without employing the complex GSSAPI mechanism.

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).  Note that other groups may also distribute
   working documents as Internet-Drafts.  The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   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."

This Internet-Draft will expire on August 1, 2012.

Table of Contents

1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Extended operation for transporting Kerberos messages

In RFC 4511 [RFC4511], section 4.12, a mechanism for defining extended operations in the LDAP protocol is described. The mechanism is based on pairs of ExtendedRequest - ExtendedResponse message pairs. In KrbLDAP, these extended operation messages are used to transport Kerberos V5 messages.

The form of Kerberos messages being transported MUST be identical to the one defined by RFC 4120 [RFC4120]. The requestName in the LDAP ExtendedRequest message and the responseName in the ExtendedResponse message MUST be set to OID value '1.3.6.1.4.1.38261.1.1'.

For the purpose of this document, such requests will be named 'KrbLDAP requests' and responses will be named 'KrbLDAP responses'.

2.1. Generation of KrbLDAP requests by the client

When the client wishes to send a KrbLDAP message to server, it constructs a Kerberos V5 PDU using the same rules as described in RFC 4120 [RFC4120].

It then embeds the resulting encoded Kerberos PDU in a KrbLDAP

extended request by placing the unmodified Kerberos V5 PDU in KrbLDAP request's requestValue.

This KrbLDAP request is then sent to the LDAP server.

2.2. Receipt of KrbLDAP requests by the server

Upon receipt of an KrbLDAP request, if the LDAP server doesn't support the extended operation, its behaviour MUST be as specified in RFC 4511 [RFC4511], section 4.12.

Conversely, if LDAP server supports KrbLDAP, it shall extract the requestValue from the KrbLDAP request, and treat it as a Kerberos PDU, passing it over to its integrated Kerberos Key Distribution Center (KDC).

2.3. Generation of KrbLDAP responses by the server

After the KDC generates the Kerberos V5 reply, the encoded PDU of that reply shall be placed by the LDAP server in the ExtendedResponse's responseValue and sent back to the client.

The result code of the LDAP ExtendedResponse MUST be success, if no LDAP-level errors have occured; that is, even if the KDC's reply is a Kerberos error, it should be only presented as a KRB_ERROR in the

Kerberos PDU, while the LDAP Extended Response shall specify a "success" resultCode. Non success LDAP result codes may only be set in case of problems originating at LDAP layer's level.

2.4. Receipt of KrbLDAP responses by the client

Upon receiving a KrbLDAP reply, the client extracts the Kerberos V5 PDU from ExtendedResponse's requestValue and processes it in the same way as described in RFC 4120 [RFC4120].

3. Interoperability Considerations

Currently there are several LDAP and Kerberos client and server implementation available on the market. It is common for those clients and servers to exhibit interoperability problems stemming from a lack of features, misconfiguration or implementation errors. When diagnosing interoperability problems between KrbLDAP clients and servers, the RECOMMENDED course of action is to start with testing Kerberos interoperability first, by falling back to plain TCP-based Kerberos V5 (if possible with the given chois of KrbLDAP client and server). This may expose an incompatibility on Kerberos level, which

is the most probable cause for interoperability problems.

4. Security Considerations

None. Security of Kerberos messages exchange isn't based on assumptions about the underlying transport protocol. Currently, UDP and TCP are possible as transport protocols for Kerberos. Because

the LDAP protocol is itself transported over TCP, embedding Kerberos messages inside it doesn't remove any guarantees that the TCP protocol provides to plain Kerberos (e.g. ordering of packets).

5. References

5.1. Normative References

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

5.2. The Kerberos Network Authentication Service (V5)

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

5.3. Lightweight Directory Access Protocol (LDAP): The Protocol

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

Author's Address

Aleksander Adamowski

Email: aleksander.adamowski@gmail.com
URI: http://olo.org.pl

Full Copyright Statement

Intellectual Property

rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Appendix B – source code for `apacheds-krbldap-test`

Below, a dump of the **`apacheds-krbldap-test`** source project is presented.

The code is covered by Apache 2.0 license. It is also available on the public Github repository at https://github.com/aadamowski/apacheds-krbldap-test.

```
# ./pom.xml
<?xml version="1.0" encoding="UTF-8"?><!--
  Licensed to the Apache Software Foundation (ASF) under one
  or more contributor license agreements.  See the NOTICE file
  distributed with this work for additional information
  regarding copyright ownership.  The ASF licenses this file
  to you under the Apache License, Version 2.0 (the
  "License"); you may not use this file except in compliance
  with the License.  You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing,
  software distributed under the License is distributed on an
  "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
  KIND, either express or implied.  See the License for the
  specific language governing permissions and limitations
  under the License.
-->
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.apache.directory.server</groupId>
    <artifactId>apacheds-parent</artifactId>
    <version>2.0.0-M4</version>
  </parent>

  <groupId>pl.org.olo.krbldap</groupId>
  <artifactId>apacheds-krbldap-test</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <apache-ds.version>${project.parent.version}</apache-ds.version>
  </properties>

  <build>
    <pluginManagement>
      <plugins>
```

```
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-surefire-plugin</artifactId>
          <configuration>
            <forkMode>never</forkMode>
          </configuration>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>

  <dependencies>
<!--
    <dependency>
      <groupId>org.apache.directory.server</groupId>
      <artifactId>apacheds-all</artifactId>
      <version>${apache-ds.version}</version>
    </dependency>
-->
    <dependency>
      <groupId>org.apache.directory.server</groupId>
      <artifactId>apacheds-server-integ</artifactId>
      <version>${apache-ds.version}</version>
    </dependency>
    <dependency>
      <groupId>org.apache.directory.server</groupId>
      <artifactId>apacheds-core-integ</artifactId>
      <version>${apache-ds.version}</version>
    </dependency>
  </dependencies>
</project>
# ./README.md

# ApacheDS KrbLDAP proof of concept implementation and test suite

  ApacheDS-based server side implementation of KrbLDAP protocol (Kerberos v5
using LDAP extended operations as carrier protocol).

  Encloses:

  * ApacheDS extended operation handler implementation that takes care of
extracting Kerberos v5 messages from the LDAP extended operation message and
feeding it to the ApacheDS Kerberos KDC / TGS service
  * Test suite that launches customized pam-krb5 module's tests against a
KrbLDAP-ized test instance of the ApacheDS server.

  The test suite requires a customized build of pam-krb5 PAM module, which acts
as a KrbLDAP client. It's available on Github as [pam_krb5-krbldap]
(http://github.com/aadamowski/pam_krb5-krbldap).

  The pam_krb5-krbldap itself requires a customized version of MIT Kerberos v5
library, which is also present on Github as [krb5-krbldap]
```

```
(http://github.com/aadamowski/krb5-krbldap).

# ./LICENSE.txt

                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.
```

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct
or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate
as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

(a) You must give any other recipients of the Work or
Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or

```
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0
```

```
   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.


#
./src/main/java/pl/org/olo/krbldap/apacheds/handlers/extended/KrbLdapAuthService
Handler.java
/*
 *   Licensed to the Apache Software Foundation (ASF) under one
 *   or more contributor license agreements.  See the NOTICE file
 *   distributed with this work for additional information
 *   regarding copyright ownership.  The ASF licenses this file
 *   to you under the Apache License, Version 2.0 (the
 *   "License"); you may not use this file except in compliance
 *   with the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 *   Unless required by applicable law or agreed to in writing,
 *   software distributed under the License is distributed on an
 *   "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 *   KIND, either express or implied.  See the License for the
 *   specific language governing permissions and limitations
 *   under the License.
 *
 */
package pl.org.olo.krbldap.apacheds.handlers.extended;

import java.net.SocketAddress;
import java.util.Collections;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Set;

import org.apache.directory.server.kerberos.protocol.KerberosProtocolHandler;
import org.apache.directory.server.ldap.ExtendedOperationHandler;
import org.apache.directory.server.ldap.LdapServer;
import org.apache.directory.server.ldap.LdapSession;
import org.apache.directory.shared.kerberos.components.PaData;
import org.apache.directory.shared.kerberos.messages.AsReq;
import org.apache.directory.shared.kerberos.messages.KerberosMessage;
import
org.apache.directory.shared.ldap.model.exception.LdapProtocolErrorException;
import org.apache.directory.shared.ldap.model.message.LdapResult;
import org.apache.directory.shared.ldap.model.message.ResultCodeEnum;
import org.apache.mina.core.future.DefaultWriteFuture;
import org.apache.mina.core.future.WriteFuture;
import org.apache.mina.core.session.DummySession;
```

```
import org.apache.mina.core.session.IoSession;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import pl.org.olo.krbldap.apacheds.extras.extended.KrbLdapRequest;
import pl.org.olo.krbldap.apacheds.extras.extended.KrbLdapResponse;
import sun.security.krb5.KrbException;

/**
 * Handler for the KrbLDAP Authentication Service request (Kerberos' AS-REQ).
 *
 * @author <a href="mailto:aleksander.adamowski@gmail.com">Aleksander
Adamowski</a>
 * @see <a href="http://www.ietf.org/rfc/rfc1510.txt">RFC 1510</a>
 */
public class KrbLdapAuthServiceHandler implements
ExtendedOperationHandler<KrbLdapRequest, KrbLdapResponse> {
    // ------------------------------ FIELDS ------------------------------

    private static final Set<String> EXTENSION_OIDS;
    private static final Logger LOG =
LoggerFactory.getLogger(KrbLdapAuthServiceHandler.class);

    protected LdapServer ldapServer;

    private KerberosProtocolHandler kerberosProtocolHandler;

    // ------------------------- STATIC METHODS -------------------------

    static {
        Set<String> set = new HashSet<String>(3);
        set.add(KrbLdapRequest.EXTENSION_OID);
        set.add(KrbLdapResponse.EXTENSION_OID);
        EXTENSION_OIDS = Collections.unmodifiableSet(set);
    }

    // --------------------------- CONSTRUCTORS ---------------------------

    public KrbLdapAuthServiceHandler() {

    }

    // -------------------- GETTER / SETTER METHODS --------------------

    public KerberosProtocolHandler getKerberosProtocolHandler() {
        return kerberosProtocolHandler;
    }

    public void setKerberosProtocolHandler(KerberosProtocolHandler
kerberosProtocolHandler) {
        this.kerberosProtocolHandler = kerberosProtocolHandler;
    }
```

```
    public void setLdapServer(LdapServer ldapServer) {
        LOG.debug("Setting LDAP Service");
        this.ldapServer = ldapServer;
    }

    // ------------------------ INTERFACE METHODS ------------------------


    // --------------------- Interface ExtendedOperationHandler
---------------------

    public String getOid() {
        return KrbLdapResponse.EXTENSION_OID;
    }

    public Set<String> getExtensionOids() {
        return EXTENSION_OIDS;
    }

    public void handleExtendedOperation(LdapSession session, KrbLdapRequest req)
throws Exception {
        LOG.info("Handling KrbLdap AS request.");
        final KerberosMessage kerberosMessage = req.getKerberosMessage();
        if (LOG.isDebugEnabled()) {
            LOG.debug("LdapSession: [" + session.toString() + "]");
            LOG.debug("ExtendedRequest: [" + req.toString() + "]");
            LOG.debug("KerberosMessage contained in ExtendedRequest: " +
kerberosMessage);
            LOG.debug("KerberosMessage class: " +
kerberosMessage.getClass().getName());
            LOG.debug("ldapServer available: " + this.ldapServer);
        }
        // Clean up zero-type padata entries from the list.
        // TODO: to be removed once Apache DS correctly handles unknown pre-
authentication types.
        if (kerberosMessage instanceof AsReq) {
            LOG.debug("PaData list contents:");
            final List<PaData> paDataList = ((AsReq)
kerberosMessage).getPaData();
            final Iterator<PaData> iterator = paDataList.iterator();
            while (iterator.hasNext()) {
                PaData paData = iterator.next();
                LOG.debug("  PaData type value: " +
paData.getPaDataType().getValue());
                LOG.debug("  PaData value: " + paData.getPaDataValue());
                if (paData.getPaDataType().getValue() == 0) {
                    LOG.warn("    Zero-type PaData found: " + paData);
                    // Workaround for
http://mailman.mit.edu/pipermail/krbdev/2012-January/010640.html :
                    LOG.warn("    removing.");
                    iterator.remove();
```

```
                }
            }
        }
        /** perform message processing similar in behaviour to
         * {@link
org.apache.directory.server.kerberos.protocol.KerberosProtocolHandler#messageRec
eived}
         */
        final IoSession ldapIoSession = session.getIoSession();
        final KrbLdapAuthServiceHandlerIoSession handlerSession = new
KrbLdapAuthServiceHandlerIoSession();
        handlerSession.setRemoteAddress(ldapIoSession.getRemoteAddress());
        KerberosMessage kerberosReply = null;
        kerberosProtocolHandler.messageReceived(handlerSession,
kerberosMessage);
        kerberosReply = handlerSession.getKerberosMessage();
        if (kerberosReply == null) {
            LOG.warn("kerberosReply in " +
KrbLdapAuthServiceHandlerIoSession.class.getName() +
                    " is null, which means that messageReceived didn't set any
reply.");
        }

        final KrbLdapResponse resultResponse = req.getResultResponse();

        if (resultResponse == null) {
            final String message = "Request has no resultResponse!";
            LOG.error(message + " request is: " + req.toString());
            throw new LdapProtocolErrorException(message);
        }

        resultResponse.setKerberosReply(kerberosReply);
        final LdapResult ldapResult = resultResponse.getLdapResult();

        if (ldapResult == null) {
            final String message = "Response has no ldapResult!";
            LOG.error(message + " response is: " + resultResponse.toString());
            throw new LdapProtocolErrorException(message);
        }
        ldapResult.setResultCode(ResultCodeEnum.SUCCESS);
        resultResponse.setMessageId(req.getMessageId());
        LOG.debug("Setting Kerberos Reply: " +
resultResponse.getKerberosReply());
        ldapIoSession.write(resultResponse);
        LOG.debug("Wrote to Ldap IO Session the following response: " +
resultResponse.toString());
    }

    // ------------------------ INNER CLASSES ------------------------

    private class KrbLdapAuthServiceHandlerIoSession extends DummySession {
        // ---------------------------- FIELDS ----------------------------
```

```
        private SocketAddress remoteAddress;

        private KerberosMessage kerberosMessage = null;

        // ---------------------- GETTER / SETTER METHODS ---------------------

        public KerberosMessage getKerberosMessage() {
            return kerberosMessage;
        }

        public void setKerberosMessage(KerberosMessage kerberosMessage) {
            this.kerberosMessage = kerberosMessage;
        }

        @Override
        public SocketAddress getRemoteAddress() {
            return this.remoteAddress;
        }

        public void setRemoteAddress(SocketAddress remoteAddress) {
            this.remoteAddress = remoteAddress;
        }

        // ------------------------ INTERFACE METHODS ------------------------


        // --------------------- Interface IoSession ---------------------

        @Override
        public WriteFuture write(Object message) {
            final DefaultWriteFuture writeFuture;
            LOG.debug("message received in session: " + message);
            if (message instanceof KerberosMessage) {
                writeFuture = (DefaultWriteFuture)
DefaultWriteFuture.newWrittenFuture(this);
                writeFuture.setValue(message);
                setKerberosMessage((KerberosMessage) message);
            } else {
                final KrbException krbException =
                        new KrbException("Message written to " +
KrbLdapAuthServiceHandlerIoSession.class.getName() +
                                " is not of class " +
KerberosMessage.class.getName() + " but instead of class " +
                                message.getClass().getName());
                LOG.error(krbException.getMessage(), krbException);
                writeFuture = (DefaultWriteFuture)
DefaultWriteFuture.newNotWrittenFuture(this, krbException);
            }
            LOG.debug("Kerberos message stored in session: " +
getKerberosMessage());
```

```
            return writeFuture;
        }
    }
}
#
./src/main/java/pl/org/olo/krbldap/apacheds/extras/extended/KrbLdapResponseImpl.
java
package pl.org.olo.krbldap.apacheds.extras.extended;

import javax.naming.ldap.ExtendedResponse;

import org.apache.directory.shared.kerberos.messages.KerberosMessage;
import org.apache.directory.shared.ldap.model.message.ExtendedRequestImpl;
import org.apache.directory.shared.ldap.model.message.ExtendedResponseImpl;
import org.apache.directory.shared.ldap.model.message.LdapResult;
import org.apache.directory.shared.ldap.model.message.ResultCodeEnum;

/**
 *
 */
public class KrbLdapResponseImpl extends ExtendedResponseImpl implements
KrbLdapResponse {
    // ------------------------------ FIELDS ------------------------------

    private KerberosMessage kerberosReply;

    // --------------------------- CONSTRUCTORS ---------------------------

    public KrbLdapResponseImpl() {
        super(EXTENSION_OID);
    }

    public KrbLdapResponseImpl(int messageId) {
        super(messageId, EXTENSION_OID);
    }

    // --------------------- GETTER / SETTER METHODS ---------------------

    public KerberosMessage getKerberosReply() {
        return kerberosReply;
    }

    public void setKerberosReply(KerberosMessage kerberosReply) {
        this.kerberosReply = kerberosReply;
    }

    @Override
    public String toString() {
        return "KrbLdapResponseImpl{" +
                "kerberosReply=" + kerberosReply +
```

```
                '}';
    }
}
#
./src/main/java/pl/org/olo/krbldap/apacheds/extras/extended/KrbLdapRequest.java
package pl.org.olo.krbldap.apacheds.extras.extended;

import org.apache.directory.shared.kerberos.messages.KerberosMessage;
import org.apache.directory.shared.ldap.model.message.ExtendedRequest;

/**
 * The interface for the KerbeLDAP extended operation request.
 *
 * @author <a href="mailto:aleksander.adamowski@gmail.com">Aleksander
Adamowski</a>
 * @see <a href="http://www.ietf.org/rfc/rfc1510.txt">RFC 1510</a>
 */
public interface KrbLdapRequest extends ExtendedRequest<KrbLdapResponse> {
    public static final String EXTENSION_OID = KrbLdapResponse.EXTENSION_OID;

    public KerberosMessage getKerberosMessage();

    public void setKerberosMessage(KerberosMessage kerberosMessage);
}
#
./src/main/java/pl/org/olo/krbldap/apacheds/extras/extended/ads_impl/krbLdap/Krb
LdapDecoder.java
package pl.org.olo.krbldap.apacheds.extras.extended.ads_impl.krbLdap;

import java.nio.ByteBuffer;
import java.nio.charset.Charset;
import java.util.Arrays;

import org.apache.commons.io.HexDump;
import org.apache.directory.shared.asn1.Asn1Object;
import org.apache.directory.shared.asn1.DecoderException;
import org.apache.directory.shared.asn1.ber.Asn1Container;
import org.apache.directory.shared.asn1.ber.Asn1Decoder;
import org.apache.directory.shared.kerberos.codec.KerberosMessageContainer;
import org.bouncycastle.util.encoders.Hex;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 *
 */
public class KrbLdapDecoder extends Asn1Decoder {
    private static final Logger LOG =
LoggerFactory.getLogger(KrbLdapDecoder.class);
    /**
     * The decoder
```

```
     */
    private static final Asn1Decoder decoder = new Asn1Decoder();

    /**
     * Decodes the incoming byte stream to {@link
org.apache.directory.shared.kerberos.messages.KerberosMessage}.
     *
     * @param stream
     * @return
     * @throws DecoderException
     */
    public Asn1Object decode(byte[] stream) throws DecoderException {
        LOG.debug(
                "KrbLDAP message data to be decoded: [" + new
String(Hex.encode(stream), Charset.forName("US-ASCII")) +
                        "]");
        ByteBuffer bb = ByteBuffer.wrap(stream);
        final KerberosMessageContainer kerberosMessageContainer = new
KerberosMessageContainer();
        kerberosMessageContainer.setStream(bb);
        kerberosMessageContainer.setGathering( true );
        decoder.decode(bb, kerberosMessageContainer);
        return kerberosMessageContainer.getMessage();
    }
}
#
./src/main/java/pl/org/olo/krbldap/apacheds/extras/extended/ads_impl/krbLdap/Krb
LdapRequestDecorator.java
package pl.org.olo.krbldap.apacheds.extras.extended.ads_impl.krbLdap;

import org.apache.directory.shared.asn1.DecoderException;
import org.apache.directory.shared.kerberos.messages.KerberosMessage;
import org.apache.directory.shared.ldap.codec.api.ExtendedRequestDecorator;
import org.apache.directory.shared.ldap.codec.api.LdapApiService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import pl.org.olo.krbldap.apacheds.extras.extended.KrbLdapRequest;
import pl.org.olo.krbldap.apacheds.extras.extended.KrbLdapResponse;

/**
 *
 */
public class KrbLdapRequestDecorator extends
ExtendedRequestDecorator<KrbLdapRequest, KrbLdapResponse>
        implements KrbLdapRequest {
    // ------------------------------ FIELDS ------------------------------

    private static final Logger LOG =
LoggerFactory.getLogger(KrbLdapRequestDecorator.class);



    // -------------------------- CONSTRUCTORS --------------------------
```

```
    /**
     * Makes a ExtendedRequest a MessageDecorator.
     *
     * @param decoratedMessage the decorated ExtendedRequest
     */
    public KrbLdapRequestDecorator(LdapApiService codec, KrbLdapRequest
decoratedMessage) {
        super(codec, decoratedMessage);
    }

    // -------------------- GETTER / SETTER METHODS --------------------

    public KerberosMessage getKerberosMessage() {
        return getDecorated().getKerberosMessage();
    }

    public void setKerberosMessage(KerberosMessage kerberosMessage) {
        getDecorated().setKerberosMessage(kerberosMessage);
    }

    // ------------------------- OTHER METHODS -------------------------

    @Override
    public void setRequestValue(byte[] requestValue) {
        final KrbLdapDecoder decoder = new KrbLdapDecoder();
        try {
            final KerberosMessage kerberosMessage = (KerberosMessage)
decoder.decode(requestValue);
            LOG.info("kerberosMessage: [" + kerberosMessage + "]");
            setKerberosMessage(kerberosMessage);
        } catch (DecoderException e) {
            LOG.error("Error decoding KerbeLDAP message: " + e.getMessage(), e);
        }
    }
}
#
./src/main/java/pl/org/olo/krbldap/apacheds/extras/extended/ads_impl/krbLdap/Krb
LdapResponseDecorator.java
package pl.org.olo.krbldap.apacheds.extras.extended.ads_impl.krbLdap;

import java.nio.ByteBuffer;

import org.apache.directory.shared.asn1.EncoderException;
import org.apache.directory.shared.kerberos.messages.KerberosMessage;
import org.apache.directory.shared.ldap.codec.api.ExtendedResponseDecorator;
import org.apache.directory.shared.ldap.codec.api.LdapApiService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import pl.org.olo.krbldap.apacheds.extras.extended.KrbLdapResponse;
```

```
/**
 *
 */
public class KrbLdapResponseDecorator extends
ExtendedResponseDecorator<KrbLdapResponse> implements KrbLdapResponse {
    // ------------------------------ FIELDS ------------------------------

    private static final int BUFFER_CAPACITY_MARGIN = 1024;
    private static final Logger LOG =
LoggerFactory.getLogger(KrbLdapResponseDecorator.class);

    // -------------------------- CONSTRUCTORS --------------------------

    /**
     * Makes a ExtendedResponse encodable.
     *
     * @param decoratedMessage the decorated ExtendedResponse
     */
    public KrbLdapResponseDecorator(LdapApiService codec, KrbLdapResponse
decoratedMessage) {
        super(codec, decoratedMessage);
    }

    // ----------------------- INTERFACE METHODS -----------------------


    // --------------------- Interface KrbLdapResponse ---------------------
    public KerberosMessage getKerberosReply() {
        return getDecorated().getKerberosReply();
    }

    public void setKerberosReply(KerberosMessage kerberosReply) {
        getDecorated().setKerberosReply(kerberosReply);
    }

    // ------------------------- OTHER METHODS -------------------------

    @Override
    public byte[] getResponseValue() {
        final KerberosMessage kerberosReply = getKerberosReply();
        if (kerberosReply == null) {
            LOG.warn("Response value requested while no Kerberos reply has been
set. Returning null.");
            return null;
        }
        final int kerberosReplyLength = kerberosReply.computeLength();
        LOG.debug("Length of Kerberos reply: {}", kerberosReplyLength);
        final ByteBuffer buffer = ByteBuffer.allocate(kerberosReplyLength);// +
BUFFER_CAPACITY_MARGIN);
        try {
            kerberosReply.encode(buffer);
```

```
        } catch (EncoderException e) {
            LOG.error("Returning null instead of encoded KrbLDAP response
because of exception [" +
                    e.getClass().getName() + "] when encoding, message: " +
e.getMessage());
            return null;
        }
        LOG.debug("Encoded Kerberos message as extended response value.");
        return buffer.array();
    }



}
#
./src/main/java/pl/org/olo/krbldap/apacheds/extras/extended/ads_impl/krbLdap/Krb
LdapFactory.java
package pl.org.olo.krbldap.apacheds.extras.extended.ads_impl.krbLdap;

import org.apache.directory.shared.asn1.DecoderException;
import org.apache.directory.shared.ldap.codec.api.ExtendedRequestDecorator;
import org.apache.directory.shared.ldap.codec.api.ExtendedRequestFactory;
import org.apache.directory.shared.ldap.codec.api.ExtendedResponseDecorator;
import org.apache.directory.shared.ldap.codec.api.LdapApiService;
import org.apache.directory.shared.ldap.model.message.ExtendedRequest;
import org.apache.directory.shared.ldap.model.message.ExtendedResponse;
import pl.org.olo.krbldap.apacheds.extras.extended.KrbLdapRequest;
import pl.org.olo.krbldap.apacheds.extras.extended.KrbLdapRequestImpl;
import pl.org.olo.krbldap.apacheds.extras.extended.KrbLdapResponse;
import pl.org.olo.krbldap.apacheds.extras.extended.KrbLdapResponseImpl;

/**
 *
 */
public class KrbLdapFactory implements ExtendedRequestFactory<KrbLdapRequest,
KrbLdapResponse> {
    private LdapApiService ldapApiService;

    public KrbLdapFactory(LdapApiService ldapApiService) {
        this.ldapApiService = ldapApiService;
    }

    public String getOid() {
        return KrbLdapResponse.EXTENSION_OID;
    }

    public KrbLdapRequest newRequest() {
        return new KrbLdapRequestDecorator(ldapApiService, new
KrbLdapRequestImpl());
    }

    public KrbLdapRequest newRequest(byte[] value) {
```

```
        final KrbLdapRequestDecorator req = new
KrbLdapRequestDecorator(ldapApiService, new KrbLdapRequestImpl());
        req.setRequestValue(value);
        return req;
    }

    public ExtendedRequestDecorator<KrbLdapRequest, KrbLdapResponse>
decorate(ExtendedRequest<?> modelRequest) {
        if (modelRequest instanceof KrbLdapRequestDecorator) {
            return (KrbLdapRequestDecorator) modelRequest;
        }

        return new KrbLdapRequestDecorator(ldapApiService, (KrbLdapRequest)
modelRequest);
    }

    public KrbLdapResponse newResponse(byte[] encodedValue) throws
DecoderException {
        KrbLdapResponseDecorator response = new
KrbLdapResponseDecorator(ldapApiService, new KrbLdapResponseImpl());
        response.setResponseValue(encodedValue);
        return response;
    }

    public ExtendedResponseDecorator<KrbLdapResponse> decorate(ExtendedResponse
decoratedMessage) {
        if (decoratedMessage instanceof KrbLdapResponseDecorator) {
            return (KrbLdapResponseDecorator) decoratedMessage;
        }

        return new KrbLdapResponseDecorator(ldapApiService, (KrbLdapResponse)
decoratedMessage);
    }
}
#
./src/main/java/pl/org/olo/krbldap/apacheds/extras/extended/KrbLdapResponse.java
package pl.org.olo.krbldap.apacheds.extras.extended;

import org.apache.directory.shared.kerberos.messages.KerberosMessage;
import org.apache.directory.shared.ldap.model.message.ExtendedResponse;

/**
 * The interface for the KerbeLDAP extended operation response.
 *
 * @author <a href="mailto:aleksander.adamowski@gmail.com">Aleksander
Adamowski</a>
 * @see <a href="http://www.ietf.org/rfc/rfc1510.txt">RFC 1510</a>
 */
public interface KrbLdapResponse extends ExtendedResponse {
    public static final String EXTENSION_OID = "1.3.6.1.4.1.38261.1.1";

    public KerberosMessage getKerberosReply();
```

```
    public void setKerberosReply(KerberosMessage kerberosReply);
}
#
./src/main/java/pl/org/olo/krbldap/apacheds/extras/extended/KrbLdapRequestImpl.j
ava
package pl.org.olo.krbldap.apacheds.extras.extended;

import org.apache.directory.shared.kerberos.messages.KerberosMessage;
import org.apache.directory.shared.ldap.model.message.AbstractExtendedRequest;

/**
 *
 */
public class KrbLdapRequestImpl extends AbstractExtendedRequest<KrbLdapResponse>
implements KrbLdapRequest {

    private KerberosMessage kerberosMessage;

    public KrbLdapRequestImpl() {
        setRequestName(EXTENSION_OID);
    }

    public KerberosMessage getKerberosMessage() {
        return kerberosMessage;
    }

    public void setKerberosMessage(KerberosMessage kerberosMessage) {
        this.kerberosMessage = kerberosMessage;
    }

    @Override
    public KrbLdapResponse getResultResponse() {
        if (response == null) {
            response = new KrbLdapResponseImpl();
        }
        return response;
    }
}
# ./src/test/java/pl/org/olo/krbldap/apacheds/test/KrbLdapIntegrationTest.java
package pl.org.olo.krbldap.apacheds.test;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

import org.apache.directory.server.annotations.CreateKdcServer;
import org.apache.directory.server.annotations.CreateLdapServer;
import org.apache.directory.server.annotations.CreateTransport;
import org.apache.directory.server.annotations.SaslMechanism;
```

```
import org.apache.directory.server.core.annotations.ApplyLdifFiles;
import org.apache.directory.server.core.annotations.ContextEntry;
import org.apache.directory.server.core.annotations.CreateDS;
import org.apache.directory.server.core.annotations.CreateIndex;
import org.apache.directory.server.core.annotations.CreatePartition;
import org.apache.directory.server.core.integ.AbstractLdapTestUnit;
import org.apache.directory.server.core.integ.FrameworkRunner;
import org.apache.directory.server.core.kerberos.KeyDerivationInterceptor;
import org.apache.directory.server.kerberos.kdc.KdcServer;
import org.apache.directory.server.kerberos.protocol.KerberosProtocolHandler;
import
org.apache.directory.server.kerberos.shared.store.DirectoryPrincipalStore;
import org.apache.directory.server.kerberos.shared.store.PrincipalStore;
import org.apache.directory.server.ldap.ExtendedOperationHandler;
import
org.apache.directory.server.ldap.handlers.bind.cramMD5.CramMd5MechanismHandler;
import
org.apache.directory.server.ldap.handlers.bind.digestMD5.DigestMd5MechanismHandl
er;
import
org.apache.directory.server.ldap.handlers.bind.gssapi.GssapiMechanismHandler;
import org.apache.directory.server.ldap.handlers.bind.ntlm.NtlmMechanismHandler;
import
org.apache.directory.server.ldap.handlers.bind.plain.PlainMechanismHandler;
import org.apache.directory.shared.kerberos.codec.types.EncryptionType;
import org.apache.directory.shared.ldap.codec.api.LdapApiService;
import org.apache.directory.shared.ldap.codec.api.LdapApiServiceFactory;
import org.apache.directory.shared.ldap.model.constants.SupportedSaslMechanisms;
import org.apache.directory.shared.ldap.model.exception.LdapInvalidDnException;
import org.apache.directory.shared.ldap.model.message.ExtendedRequest;
import org.apache.directory.shared.ldap.model.message.ExtendedResponse;
import org.apache.directory.shared.ldap.model.name.Dn;
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import pl.org.olo.krbldap.apacheds.extras.extended.KrbLdapRequest;
import
pl.org.olo.krbldap.apacheds.extras.extended.ads_impl.krbLdap.KrbLdapFactory;
import pl.org.olo.krbldap.apacheds.handlers.extended.KrbLdapAuthServiceHandler;

/**
 * This test suite tests PAM-krbldap integration with ApacheDS-krbldap.
 */
@RunWith(FrameworkRunner.class)
@CreateDS(name = "SaslGssapiBindITest-class",
        allowAnonAccess = true,
        partitions = {@CreatePartition(
                name = "example",
                suffix = "dc=example,dc=com",
                contextEntry = @ContextEntry(
                        entryLdif = "dn: dc=example,dc=com\n" + "dc: example\n"
+ "objectClass: top\n" +
```

```
                                "objectClass: domain\n\n"),
                indexes = {@CreateIndex(attribute = "objectClass"),
@CreateIndex(attribute = "dc"),
                        @CreateIndex(attribute = "ou"), @CreateIndex(attribute =
"uid")})},
        additionalInterceptors = {KeyDerivationInterceptor.class})
@CreateLdapServer(
        transports = {@CreateTransport(protocol = "LDAP", port = 1389)},
        allowAnonymousAccess = true,
        extendedOpHandlers = KrbLdapAuthServiceHandler.class,
        saslHost = "localhost",
        saslPrincipal = "ldap/localhost@EXAMPLE.COM",
        saslMechanisms = {@SaslMechanism(name = SupportedSaslMechanisms.PLAIN,
implClass = PlainMechanismHandler.class),
                @SaslMechanism(name = SupportedSaslMechanisms.CRAM_MD5,
implClass = CramMd5MechanismHandler.class),
                @SaslMechanism(name = SupportedSaslMechanisms.DIGEST_MD5,
implClass = DigestMd5MechanismHandler.class),
                @SaslMechanism(name = SupportedSaslMechanisms.GSSAPI, implClass
= GssapiMechanismHandler.class),
                @SaslMechanism(name = SupportedSaslMechanisms.NTLM, implClass =
NtlmMechanismHandler.class),
                @SaslMechanism(name = SupportedSaslMechanisms.GSS_SPNEGO,
implClass = NtlmMechanismHandler.class)})
@CreateKdcServer(
        transports = {@CreateTransport(protocol = "UDP", port = 8800),
@CreateTransport(protocol = "TCP", port = 8800)})
@ApplyLdifFiles("test.ldif")
public class KrbLdapIntegrationTest extends AbstractLdapTestUnit {
    /**
     * Pathname of the client test shell script
     */
    private static final String CLIENT_TEST_SCRIPT_KRBLDAP =
            "/var/soft/PAM/krb5-github/src/pam_krb5/tests/run-tests-krbldap-
direct.sh";

    /**
     * Pathname of the client test shell script
     */
    private static final String CLIENT_TEST_SCRIPT_KRB5 =
            "/var/soft/PAM/krb5-github/src/pam_krb5/tests/run-tests-krb5-
direct.sh";
    /**
     * KRB5 conf file location relative to classpath
     */
    private static final String KRB5_CONF_RESOURCE_LOCATION = "krb5.conf";

    @Before
    public void initialize() throws LdapInvalidDnException {
        registerKrbLdapExtendedRequest();
        configureKrbLdapAuthServiceHandler();
    }
```

```
    /**
     * Creates a new instance of SaslGssapiBindTest and sets JAAS system
properties.
     */
    public KrbLdapIntegrationTest() {
        String krbConfPath =
getClass().getClassLoader().getResource(KRB5_CONF_RESOURCE_LOCATION).getFile();
        System.setProperty("java.security.krb5.conf", krbConfPath);
        System.setProperty("sun.security.krb5.debug", "true");

    }

    //@Test
    public void testShouldPerformSuccessfulKrb5Authentication() throws Exception
{
        final String clientTestScript = CLIENT_TEST_SCRIPT_KRB5;
        runTestScript(clientTestScript);

    }

    @Test
    public void testShouldPerformSuccessfulKrbLdapAuthentication() throws
Exception {
        final String clientTestScript = CLIENT_TEST_SCRIPT_KRBLDAP;
        runTestScript(clientTestScript);
    }

    private void runTestScript(String clientTestScript) throws IOException,
InterruptedException {
        final Process process = Runtime.getRuntime().exec(clientTestScript);
        final InputStream errorStream = process.getErrorStream();
        final InputStream inputStream = process.getInputStream();
        final BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream));
        final BufferedReader errorReader = new BufferedReader(new
InputStreamReader(errorStream));
        System.out.println(
                "Testing started. Check your system's syslog (facility AUTH) for
any messages from the PAM module.");
        System.out.println("Test script: " + clientTestScript);
        while (true) {
            while (reader.ready()) {
                System.out.println(reader.readLine());
            }
            while (errorReader.ready()) {
                System.out.println(errorReader.readLine());
            }
            Thread.sleep(50);
        }
        /*
        final int retValue = process.waitFor();
        System.out.println("Return code: [" + retValue + "]");
```

```
        System.out.println("STDOUT:");
        System.out.println(IOUtils.toString(inputStream));
        System.out.println("STDOUT END.");
        System.out.println("STDERR:");
        System.out.println(IOUtils.toString(errorStream));
        System.out.println("STDERR END.");
        if (retValue != 0) {
            throw new RuntimeException("error code [" + retValue + "] returned
from process.");
        }
        */
    }

    private void configureKrbLdapAuthServiceHandler() throws
LdapInvalidDnException {
        // Configure the KrbLdapAuthServiceHandler by injecting it with a new
KerberosProtocolHandler
        // based on the present KdcServer:
        final ExtendedOperationHandler<ExtendedRequest<ExtendedResponse>,
ExtendedResponse> extendedOperationHandler =
                getLdapServer().getExtendedOperationHandler(KrbLdapRequest.EXTEN
SION_OID);
        // Workaround for JVM generics bug "javac error: inconvertible types" -
see:
        // http://stackoverflow.com/questions/4829576/javac-error-inconvertible-
types-with-generics
        Object tmp = extendedOperationHandler;
        if (!(tmp instanceof KrbLdapAuthServiceHandler)) {
            throw new IllegalStateException(
                    "extendedOperationHandler not instanceof " +
KrbLdapAuthServiceHandler.class.getName());
        }
        final KrbLdapAuthServiceHandler krbLdapAuthServiceHandler =
(KrbLdapAuthServiceHandler) tmp;
        final KdcServer kdcServer = getKdcServer();
        System.out.println("kdcServer: " + kdcServer);
        kdcServer.setEncryptionTypes(
                new EncryptionType[]{EncryptionType.AES256_CTS_HMAC_SHA1_96,
EncryptionType.AES128_CTS_HMAC_SHA1_96,
                        EncryptionType.DES3_CBC_SHA1_KD});
        PrincipalStore principalStore =
                new DirectoryPrincipalStore(kdcServer.getDirectoryService(), new
Dn(kdcServer.getSearchBaseDn()));
        final KerberosProtocolHandler kerberosProtocolHandler = new
KerberosProtocolHandler(kdcServer, principalStore);
        krbLdapAuthServiceHandler.setKerberosProtocolHandler(kerberosProtocolHan
dler);
    }

    private void registerKrbLdapExtendedRequest() {
        final LdapApiService ldapApiService =
LdapApiServiceFactory.getSingleton();
        final KrbLdapFactory krbLdapFactory = new
```

```
KrbLdapFactory(ldapApiService);
        ldapApiService.registerExtendedRequest(krbLdapFactory);
    }

}
# ./src/test/resources/test.ldif
#
#  Licensed to the Apache Software Foundation (ASF) under one
#  or more contributor license agreements.  See the NOTICE file
#  distributed with this work for additional information
#  regarding copyright ownership.  The ASF licenses this file
#  to you under the Apache License, Version 2.0 (the
#  "License"); you may not use this file except in compliance
#  with the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
#  Unless required by applicable law or agreed to in writing,
#  software distributed under the License is distributed on an
#  "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
#  KIND, either express or implied.  See the License for the
#  specific language governing permissions and limitations
#  under the License.
#
#   EXAMPLE.COM is reserved for testing according to this RFC:
#
#   http://www.rfc-editor.org/rfc/rfc2606.txt
#

dn: ou=users,dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: users


dn: uid=alice,ou=users,dc=example,dc=com
objectClass: top
objectClass: account
objectClass: simpleSecurityObject
objectclass: krb5Principal
objectclass: krb5KDCEntry
uid: alice
userPassword: password
krb5PrincipalName: alice@EXAMPLE.COM
krb5KeyVersionNumber: 0# ./src/test/resources/krb5.conf
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
```

```
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied.  See the License for the
# specific language governing permissions and limitations
# under the License.
[libdefaults]
      default_realm = EXAMPLE.COM

[realms]
      EXAMPLE.COM = {
            kdc = localhost:6088
      }

[domain_realm]
      .example.com = EXAMPLE.COM
      example.com = EXAMPLE.COM

[login]
      krb4_convert = true
      krb4_get_tickets = false# ./src/test/resources/log4j.properties
################################################################################
#    Licensed to the Apache Software Foundation (ASF) under one or more
#    contributor license agreements.  See the NOTICE file distributed with
#    this work for additional information regarding copyright ownership.
#    The ASF licenses this file to You under the Apache License, Version 2.0
#    (the "License"); you may not use this file except in compliance with
#    the License.  You may obtain a copy of the License at
#
#       http://www.apache.org/licenses/LICENSE-2.0
#
#    Unless required by applicable law or agreed to in writing, software
#    distributed under the License is distributed on an "AS IS" BASIS,
#    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#    See the License for the specific language governing permissions and
#    limitations under the License.
################################################################################
log4j.rootCategory=INFO, logfile

#log4j.org.apache.directory.shared.ldap.codec.standalone.StandaloneLdapCodecServ
ice=DEBUG
log4j.logger.pl.org.olo=DEBUG
log4j.logger.org.apache.directory=DEBUG
log4j.logger.org.apache.directory.shared.ldap.model.ldif.LdifReader=INFO
log4j.logger.org.apache.directory.shared.ldap.model.schema.registries=INFO
```

```
log4j.appender.logfile=org.apache.log4j.RollingFileAppender
log4j.appender.logfile.File=target/krbldap-test.log
log4j.appender.logfile.MaxFileSize=5GB
log4j.appender.logfile.layout=org.apache.log4j.PatternLayout
log4j.appender.logfile.layout.ConversionPattern=[%d{yyyy-MM-dd HH:mm:ss}] %p
[%c] - %m%n

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=[%d{HH:mm:ss}] %p [%c] - %m%n
```

Appendix C – unified diff representation of `pam_krb5-krbldap` customizations

Presented below, there is a standard unified diff representation of code customizations against `pam_krb5` module that comprise the customized **`pam_krb5-krbldap`**.

The code is covered by the same dual LGPL/BSD license that the original `pam_krb5` module is published under. It is also available on the public Github repository at https://github.com/aadamowski/pam_krb5-krbldap.

```
diff --git a/configure.ac b/configure.ac
index 83b8ff7..c471c8d 100644
--- a/configure.ac
+++ b/configure.ac
@@ -209,6 +209,7 @@ KRB5_CPPFLAGS=`echo $KRB5_CFLAGS | sed 's,-[^I]
[^[:space:]]*,,g'`
 KRB4_CPPFLAGS=`echo $KRB4_CFLAGS | sed 's,-[^I][^[:space:]]*,,g'`

 CPPFLAGS="$CPPFLAGS $KRB5_CPPFLAGS $KRB4_CPPFLAGS"
+CFLAGS="-g"
 CFLAGS="$CFLAGS $KRB5_CFLAGS $KRB4_CFLAGS"

 AC_CHECK_HEADERS(krb5.h krb5/krb5.h)
@@ -645,6 +646,7 @@ tests/config/kdc.conf
 tests/config/krb.conf
 tests/kdc/Makefile
 tests/testenv.sh
+tests/testenv-krbldap.sh
 tests/tools/Makefile
 po/Makefile.in
 ])
diff --git a/tests/Makefile.am b/tests/Makefile.am
index 5170091..b52795f 100644
--- a/tests/Makefile.am
+++ b/tests/Makefile.am
@@ -1,6 +1,9 @@
 SUBDIRS = config tools kdc

-EXTRA_DIST = run-tests.sh testenv.sh.in pwhelp.txt \
+EXTRA_DIST = run-tests.sh testenv.sh.in run-tests-krbldap.sh testenv-
krbldap.sh.in pwhelp.txt \
+     000-pambasic_krbldap/run.sh \
+     000-pambasic_krbldap/stderr.expected \
+     000-pambasic_krbldap/stdout.expected \
      001-pambasic/run.sh \
      001-pambasic/stderr.expected \
      001-pambasic/stdout.expected \
diff --git a/tests/config/krb5.conf.in b/tests/config/krb5.conf.in
index 59a526f..48dd769 100644
--- a/tests/config/krb5.conf.in
```

```
+++ b/tests/config/krb5.conf.in
@@ -11,6 +11,7 @@
 [realms]
  EXAMPLE.COM = {
   kdc = @TESTHOST@:8800
+  master_kdc = @TESTHOST@:8800
   admin_server = @TESTHOST@:8801
   kpasswd_server = @TESTHOST@:8802
  }
@@ -20,7 +21,7 @@

 [appdefaults]
  pam = {
-   debug = false
+   debug = true
    ticket_lifetime = 36000
    renew_lifetime = 36000
    forwardable = true
diff --git a/tests/krbldap-tests/001-pambasic_krbldap/run.sh b/tests/krbldap-
tests/001-pambasic_krbldap/run.sh
new file mode 100755
index 0000000..4110b45
--- /dev/null
+++ b/tests/krbldap-tests/001-pambasic_krbldap/run.sh
@@ -0,0 +1,33 @@
+#!/bin/sh
+
+. $testdir/testenv-krbldap.sh
+
+test_flags="$test_flags ignore_afs"
+
+echo "test_run -auth $test_principal $pam_krbldap $test_flags -- bar"
+echo ""; echo Fail: incorrect password.
+test_run -auth $test_principal $pam_krbldap $test_flags -- bar
+
+#echo ""; echo Fail: incorrect password.
+#test_run -auth $test_principal $pam_krbldap $test_flags -- foolong
+#
+#echo ""; echo Fail: incorrect password.
+#test_run -auth $test_principal $pam_krbldap $test_flags -- foolongerstill
+#
+#echo ""; echo Succeed: correct password.
+#test_run -auth -setcred -session $test_principal $pam_krbldap $test_flags --
foo
+#
+#echo ""; echo Fail: cannot read password.
+#test_run -auth $test_principal $pam_krbldap $test_flags use_first_pass -- foo
+#
+#echo ""; echo Succeed: correct password, incorrect first attempt.
+#test_run -auth -setcred $test_principal $pam_krbldap $test_flags
try_first_pass -- foo
```

```
+#
+#echo ""; echo Succeed: correct password, maybe use incorrect second attempt.
+#test_run -auth -session $test_principal $pam_krbldap $test_flags -authtok foo
-- bar
+#
+#echo ""; echo Succeed: correct password, ignore second attempt.
+#test_run -auth -setcred -session $test_principal $pam_krbldap $test_flags
-authtok foo use_first_pass -- bar
+#
+#echo ""; echo Succeed: correct password, maybe use incorrect second attempt.
+#test_run -auth $test_principal $pam_krbldap $test_flags -authtok foo
try_first_pass -- bar
diff --git a/tests/krbldap-tests/001-pambasic_krbldap/stderr.expected
b/tests/krbldap-tests/001-pambasic_krbldap/stderr.expected
new file mode 100644
index 0000000..e69de29
diff --git a/tests/krbldap-tests/001-pambasic_krbldap/stdout.expected
b/tests/krbldap-tests/001-pambasic_krbldap/stdout.expected
new file mode 100644
index 0000000..e69de29
diff --git a/tests/run-tests-krb5-direct.sh b/tests/run-tests-krb5-direct.sh
new file mode 100755
index 0000000..d9deaae
--- /dev/null
+++ b/tests/run-tests-krb5-direct.sh
@@ -0,0 +1,13 @@
+#!/bin/sh
+
+export KRB5_TRACE=/dev/stdout
+TESTROOTDIR=$(dirname $0)
+
+KRB5_CONFIG=$(readlink -f $TESTROOTDIR/config/krb5.conf) ; export KRB5_CONFIG;
echo "KRB5_CONFIG: $KRB5_CONFIG"
+KRBCONFDIR=$TESTROOTDIR/config ; export KRBCONFDIR
+KRB_CONF=$TESTROOTDIR/config/krb.conf ; export KRB_CONF
+KRB5RCACHEDIR=$TESTROOTDIR/kdc ; export KRB5RCACHEDIR
+KRB5CCNAME=/dev/bogus-missing-file ; export KRB5CCNAME
+KRBTKFILE=/dev/bogus-missing-file ; export KRBTKFILE
+
+$TESTROOTDIR/tools/pam_harness -auth alice
$TESTROOTDIR/../src/.libs/pam_krb5.so  ignore_afs unsecure_for_debugging_only --
bar 2>&1
diff --git a/tests/run-tests-krbldap-direct.sh b/tests/run-tests-krbldap-
direct.sh
new file mode 100755
index 0000000..625f05f
--- /dev/null
+++ b/tests/run-tests-krbldap-direct.sh
@@ -0,0 +1,14 @@
+#!/bin/sh
+
+export LD_LIBRARY_PATH=/var/soft/PAM/usr/lib
```

```
+export KRB5_TRACE=/dev/stdout
+TESTROOTDIR=$(dirname $0)
+
+KRB5_CONFIG=$TESTROOTDIR/config/krb5.conf ; export KRB5_CONFIG
+KRBCONFDIR=$TESTROOTDIR/config ; export KRBCONFDIR
+KRB_CONF=$TESTROOTDIR/config/krb.conf ; export KRB_CONF
+KRB5RCACHEDIR=$TESTROOTDIR/kdc ; export KRB5RCACHEDIR
+KRB5CCNAME=/dev/bogus-missing-file ; export KRB5CCNAME
+KRBTKFILE=/dev/bogus-missing-file ; export KRBTKFILE
+
+LD_PRELOAD=/lib/x86_64-linux-gnu/libpthread.so.0 gdb -x /var/soft/PAM/gdb.cmds
--args $TESTROOTDIR/tools/pam_harness -auth alice
$TESTROOTDIR/../src/.libs/pam_krb5.so  ignore_afs unsecure_for_debugging_only --
bar 2>&1
diff --git a/tests/run-tests-krbldap.sh b/tests/run-tests-krbldap.sh
new file mode 100755
index 0000000..0f18bc6
--- /dev/null
+++ b/tests/run-tests-krbldap.sh
@@ -0,0 +1,36 @@
+#!/bin/sh
+
+testdir=`dirname "$0"`
+testdir=`cd "$testdir" ; pwd`
+export testdir
+
+. $testdir/testenv-krbldap.sh
+echo "Running tests using test principal \"$test_principal\"".
+echo "Running tests using KDC on \"$test_host\"".
+getent hosts "$test_host"
+
+
+# Run each test with clear log files and a fresh copy of the KDC and kadmind,
+# and a fresh 524d if available.
+for test in ${@:-"$testdir"/krbldap-tests/0*} ; do
+     if ! test -s $test/run.sh ; then
+          continue
+     fi
+     echo -n `basename "$test"` ..." "
+     meanwhile "$test/run.sh" > $test/stdout 2> $test/stderr
+     if test -s $test/stdout.expected ; then
+          if ! cmp -s $test/stdout.expected $test/stdout ; then
+               echo ""
+               diff -u $test/stdout.expected $test/stdout | sed "s|
$testdir/||g"
+               echo "Test $test stdout unexpected error!"
+               exit 1
+          fi
+          if ! cmp -s $test/stderr.expected $test/stderr ; then
+               echo ""
+               diff -u $test/stderr.expected $test/stderr | sed "s|
```

```
$testdir/||g"
+                 echo "Test $test stderr unexpected error!"
+                 exit 1
+           fi
+     fi
+     echo OK
+done
diff --git a/tests/testenv-krbldap.sh.in b/tests/testenv-krbldap.sh.in
new file mode 100755
index 0000000..b0ffe82
--- /dev/null
+++ b/tests/testenv-krbldap.sh.in
@@ -0,0 +1,37 @@
+#!/bin/sh
+
+PATH=${testdir}/tools:${PATH}; export PATH
+
+test_principal=`id -nu`
+test_host=@TESTHOST@
+
+if test @USE_ADDRESSES@ -eq 1 ; then
+     test_addresses=true
+else
+     test_addresses=false
+fi
+
+pam_krbldap=@abs_builddir@/../src/pam_krb5.so
+if ! test -x $pam_krbldap ; then
+     pam_krbldap=@abs_builddir@/../src/.libs/pam_krb5.so
+fi
+
+KRB5_CONFIG=@abs_builddir@/config/krbldap.conf ; export KRB5_CONFIG
+KRBCONFDIR=@abs_builddir@/config ; export KRBCONFDIR
+
+test_settle() {
+     sleep 1
+}
+
+test_cleanmsg ()
+{
+     sed -e "s,Warning: Your password will expire in less than one hour.
[^\']*,WARN1HOUR,g" \
+         -e "s,Warning: .*password has expired[^\']*,WARNEXPIRED,g" \
+         -e "s|krb5cc_"`id -u`"_......|"'krb5_cc_$UID_XXXXXX|g'
+}
+
+
+test_run() {
+     # Filter out the module path and clean up messages.
+     @abs_builddir@/tools/pam_harness "$@" 2>&1 | sed s,"\`.*pam",'\`pam',g |
test_cleanmsg
```

```
+}
diff --git a/tests/tools/pam_harness.c b/tests/tools/pam_harness.c
index 6e403b8..8c72d4a 100644
--- a/tests/tools/pam_harness.c
+++ b/tests/tools/pam_harness.c
@@ -115,6 +115,8 @@ converse(int num_msgs,
 int
 main(int argc, char **argv)
 {
+     /* Disable buffering on stdout so that output lines up with error output
*/
+     setbuf(stdout, NULL);
      void *dlhandle;
      int doauth, doaccount, dosession, dosetcred, dochauthtok, doprompt;
      int noreentrancy;
```

Appendix D – unified diff representation of `krb5-krbldap` customizations

Presented below, there is a standard unified diff representation of code customizations against MIT Kerberos 5 library sources that comprise the customized **`krb5-krbldap`**.

The code is covered by the same MIT license that the original MIT Kerberos 5 library is published under. It is also available on the public Github repository at https://github.com/aadamowski/krb5-krbldap.

```
diff --git a/src/aclocal.m4 b/src/aclocal.m4
index 7e635c8..620679d 100644
--- a/src/aclocal.m4
+++ b/src/aclocal.m4
@@ -84,6 +84,7 @@ AC_SUBST_FILE(libnodeps_frag)
 dnl
 KRB5_AC_PRAGMA_WEAK_REF
 WITH_LDAP
+KRB5_AC_ENABLE_KRBLDAP
 KRB5_LIB_PARAMS
 KRB5_AC_INITFINI
 KRB5_AC_ENABLE_THREADS
@@ -1268,6 +1269,18 @@ ns_initparse ns_name_uncompress dn_skipname res_search)
     fi
   fi
 ])
+dnl
+dnl The following was written by aleksander.adamowski@gmail.com
+dnl
+dnl This test ensures propagation of KRBLDAP defines.
+AC_DEFUN(KRB5_AC_ENABLE_KRBLDAP, [
+AC_ARG_ENABLE([krbldap],
+[  --enable-krbldap  enable KrbLDAP transport in addition to TCP and UDP], ,
+[enable_krbldap=no])
+  if test "$enable_krbldap" = yes; then
+    ENABLE_KRBLDAP=yes
+  fi
+])
 AC_DEFUN([_KRB5_AC_CHECK_RES_FUNCS],
 [AC_FOREACH([AC_Func], [$1],
   [AH_TEMPLATE(AS_TR_CPP(HAVE_[]AC_Func),
diff --git a/src/configure.in b/src/configure.in
index 4c6ec8c..9a74929 100644
--- a/src/configure.in
+++ b/src/configure.in
@@ -1116,7 +1116,7 @@ fi

 ldap_plugin_dir=""
 ldap_lib=""
```

```
-if test -n "$OPENLDAP_PLUGIN"; then
+if test -n "$OPENLDAP_PLUGIN" -o -n "$ENABLE_KRBLDAP"; then
   AC_CHECK_HEADERS(ldap.h lber.h, :, [AC_MSG_ERROR($ac_header not found)])
   AC_CHECK_LIB(ldap, ldap_init, :, [AC_MSG_ERROR(libldap not found or missing
ldap_init)])
   old_LIBS="$LIBS"
@@ -1136,16 +1136,24 @@ if test -n "$OPENLDAP_PLUGIN"; then
       AC_ERROR("BER library missing - cannot build LDAP database module")
     fi
   fi
-  AC_DEFINE([ENABLE_LDAP], 1, [Define if LDAP KDB support within the Kerberos
library (mainly ASN.1 code) should be enabled.])
+  if test -n "$ENABLE_KRBLDAP"; then
+    AC_DEFINE(KRB5_KRBLDAP, 1,[Define for support for KrbLDAP protocol])
+    KRBLDAP=yes
+  else
+    KRBLDAP=no
+  fi
+  if test -n "$OPENLDAP_PLUGIN"; then
+    AC_DEFINE([ENABLE_LDAP], 1, [Define if LDAP KDB support within the Kerberos
library (mainly ASN.1 code) should be enabled.])
+
+    K5_GEN_MAKEFILE(plugins/kdb/ldap)
+    K5_GEN_MAKEFILE(plugins/kdb/ldap/ldap_util)
+    K5_GEN_MAKEFILE(plugins/kdb/ldap/libkdb_ldap)
+    ldap_plugin_dir='plugins/kdb/ldap plugins/kdb/ldap/ldap_util'
+    LDAP=yes
+  else
+    LDAP=no
+  fi
   AC_SUBST(LDAP_LIBS)
-
-  K5_GEN_MAKEFILE(plugins/kdb/ldap)
-  K5_GEN_MAKEFILE(plugins/kdb/ldap/ldap_util)
-  K5_GEN_MAKEFILE(plugins/kdb/ldap/libkdb_ldap)
-  ldap_plugin_dir='plugins/kdb/ldap plugins/kdb/ldap/ldap_util'
-  LDAP=yes
-else
-  LDAP=no
 fi
 AC_SUBST(ldap_plugin_dir)
 AC_SUBST(LDAP)
diff --git a/src/include/k5-int.h b/src/include/k5-int.h
index 7a196c6..fcc25c8 100644
--- a/src/include/k5-int.h
+++ b/src/include/k5-int.h
@@ -379,6 +379,12 @@ typedef INT64_TYPE krb5_int64;
 #define KRB_AP_ERR_IAKERB_KDC_NO_RESPONSE       86 /* The KDC did not respond
                                                       to the IAKERB proxy */

+#ifdef KRB5_KRBLDAP
```

```
+#define KRBLDAP_OID_EXOP_BASE "1.3.6.1.4.1.38261.1"
+#define KRBLDAP_OID_EXOP_AS_REQ KRBLDAP_OID_EXOP_BASE ".1"
+#define KRBLDAP_OID_EXOP_TGS_REQ KRBLDAP_OID_EXOP_BASE ".2"
+#endif
+
 /*
  * A null-terminated array of this structure is returned by the KDC as
  * the data part of the ETYPE_INFO preauth type.  It informs the
@@ -1348,7 +1354,11 @@ struct _krb5_context {
 #ifdef KRB5_DNS_LOOKUP
     krb5_boolean    profile_in_memory;
 #endif /* KRB5_DNS_LOOKUP */
-
+
+#ifdef KRB5_KRBLDAP
+    krb5_boolean    use_krbldap;
+#endif /* KRB5_DNS_LOOKUP */
+
     /* locate_kdc module stuff */
     struct plugin_dir_handle libkrb5_plugins;
     struct krb5plugin_service_locate_ftable *vtbl;
diff --git a/src/lib/krb5/Makefile.in b/src/lib/krb5/Makefile.in
index a63270a..b5bd354 100644
--- a/src/lib/krb5/Makefile.in
+++ b/src/lib/krb5/Makefile.in
@@ -56,7 +56,7 @@ RELDIR=krb5
 SHLIB_EXPDEPS = \
      $(TOPLIBD)/libk5crypto$(SHLIBEXT) \
      $(COM_ERR_DEPLIB) $(SUPPORT_DEPLIB)
-SHLIB_EXPLIBS=-lk5crypto -lcom_err $(SUPPORT_LIB) @GEN_LIB@ $(LIBS)
+SHLIB_EXPLIBS=-lk5crypto -lcom_err $(SUPPORT_LIB) @GEN_LIB@ $(LDAP_LIBS) $
(LIBS)
 SHLIB_DIRS=-L$(TOPLIBD)
 SHLIB_RDIRS=$(KRB5_LIBDIR)

diff --git a/src/lib/krb5/os/Makefile.in b/src/lib/krb5/os/Makefile.in
index 19f5c33..a3a690c 100644
--- a/src/lib/krb5/os/Makefile.in
+++ b/src/lib/krb5/os/Makefile.in
@@ -4,7 +4,7 @@ KRB5_RUN_ENV = @KRB5_RUN_ENV@
 PROG_LIBPATH=-L$(TOPLIBD)
 PROG_RPATH=$(KRB5_LIBDIR)
 DEFS=
-DEFINES=-DLIBDIR=\"$(KRB5_LIBDIR)\"
+DEFINES=-DLIBDIR=\"$(KRB5_LIBDIR)\" -DDEBUG
 LOCALINCLUDES=-I$(top_srcdir)/util/profile

 ##DOS##BUILDTOP = ..\..\..
diff --git a/src/lib/krb5/os/locate_kdc.c b/src/lib/krb5/os/locate_kdc.c
index acfce8c..a111098 100644
```

```
--- a/src/lib/krb5/os/locate_kdc.c
+++ b/src/lib/krb5/os/locate_kdc.c
@@ -113,7 +113,7 @@ k5_free_serverlist (struct serverlist *list)
 static inline void
 Tprintf(const char *fmt, ...)
 {
-#ifdef TEST
+#ifdef DEBUG
     va_list ap;
     va_start(ap, fmt);
     vfprintf(stderr, fmt, ap);
diff --git a/src/lib/krb5/os/os-proto.h b/src/lib/krb5/os/os-proto.h
index 665187c..cd3b561 100644
--- a/src/lib/krb5/os/os-proto.h
+++ b/src/lib/krb5/os/os-proto.h
@@ -100,6 +100,18 @@ krb5_error_code k5_sendto(krb5_context context, const
krb5_data *message,
                                         void *),
                           void *msg_handler_data);

+#ifdef KRB5_KRBLDAP
+krb5_error_code krbldap_sendto(krb5_context context, const krb5_data *message,
+                          const struct serverlist *addrs,
+                          int socktype1, int socktype2,
+                          struct sendto_callback_info *callback_info,
+                          krb5_data *reply, struct sockaddr *remoteaddr,
+                          socklen_t *remoteaddrlen, int *server_used,
+                          int (*msg_handler)(krb5_context, const krb5_data *,
+                                         void *),
+                          void *msg_handler_data);
+#endif
+
 krb5_error_code krb5int_get_fq_local_hostname(char *, size_t);

 /* The io vector is *not* const here, unlike writev()!  */
diff --git a/src/lib/krb5/os/sendto_kdc.c b/src/lib/krb5/os/sendto_kdc.c
index 63dbcd8..a72ed16 100644
--- a/src/lib/krb5/os/sendto_kdc.c
+++ b/src/lib/krb5/os/sendto_kdc.c
@@ -51,14 +51,19 @@
 #endif
 #endif

+#ifdef KRB5_KRBLDAP
+#include <lber.h>
+#include <ldap.h>
+#endif
+
 #define MAX_PASS                    3
 #define DEFAULT_UDP_PREF_LIMIT   1465
```

```
 #define HARD_UDP_LIMIT          32700 /* could probably do 64K-epsilon ? */

-#undef DEBUG
+/*#undef DEBUG*/

 #ifdef DEBUG
-int krb5int_debug_sendto_kdc = 0;
+int krb5int_debug_sendto_kdc = 1;
 #define debug krb5int_debug_sendto_kdc

 static void
@@ -333,12 +338,23 @@ krb5_sendto_kdc(krb5_context context, const krb5_data
*message,

     retval = k5_locate_kdc(context, realm, &servers, *use_master,
                            tcp_only ? SOCK_STREAM : 0);
-    if (retval)
+    dprint("k5_locate_kdc retval: [%d]\n", retval);
+
+    if (retval) {
+        dprint("Error [%d], message: [%s]\n", context->err.code, context-
>err.msg);
         return retval;
-
+    }
+#ifdef KRB5_KRBLDAP
+    dprint("using krbldap protocol to send kerberos message.\n");
+    retval = krbldap_sendto(context, message, &servers, socktype1, socktype2,
+                        NULL, reply, NULL, NULL, &server_used,
+                        check_for_svc_unavailable, &err);
+#else
+    dprint("using kerberos v5 protocol to send kerberos message.\n");
     retval = k5_sendto(context, message, &servers, socktype1, socktype2,
                        NULL, reply, NULL, NULL, &server_used,
                        check_for_svc_unavailable, &err);
+#endif
     if (retval == KRB5_KDC_UNREACH) {
         if (err == KDC_ERR_SVC_UNAVAILABLE) {
             retval = KRB5KDC_ERR_SVC_UNAVAILABLE;
@@ -1352,3 +1368,81 @@ cleanup:
     free(sel_state);
     return retval;
 }
+#ifdef KRB5_KRBLDAP
+krb5_error_code
+krbldap_sendto(krb5_context context, const krb5_data *message,
+          const struct serverlist *servers, int socktype1, int socktype2,
+          struct sendto_callback_info* callback_info, krb5_data *reply,
+          struct sockaddr *remoteaddr, socklen_t *remoteaddrlen,
+          int *server_used,
+          /* return 0 -> keep going, 1 -> quit */
```

```
+          int (*msg_handler)(krb5_context, const krb5_data *, void *),
+          void *msg_handler_data)
+{
+    LDAP *ldap;
+    krb5_boolean done = FALSE;
+    struct server_entry *entry;
+    int ldap_version = LDAP_VERSION3;
+    struct berval berval;
+    struct berval *retdata = NULL;
+    char *retoid = NULL;
+    char *hostname = NULL;
+    char ldap_url[max(NI_MAXHOST + NI_MAXSERV + 30, 200)];
+    int s;
+    int port = 0;
+    int debug = 0xffffff;
+    int rc;
+    int retval = 0;
+
+    /* TODO: use *servers */
+    for (s = 0; s < servers->nservers && !done; s++) {
+        entry = &servers->servers[s];
+        hostname = entry->hostname;
+        port = 1389;
+        dprint("Trying server [%s] on port [%d].\n", hostname, port);
+        if (snprintf(ldap_url, sizeof (ldap_url), "ldap://%s:%d", hostname,
port) >= sizeof (ldap_url)) {
+            /* Error, size limit hit */
+            retval = ENOMEM;
+            dprint("LDAP URL size greater than limit\n");
+            goto cleanup;
+        }
+        dprint("LDAP URL: [%s]\n", ldap_url);
+
+        rc = ldap_initialize(&ldap, ldap_url);
+        dprint("rc: [%d], LDAP_SUCCESS: [%d]\n", rc, LDAP_SUCCESS);
+
+        ldap_set_option(ldap, LDAP_OPT_PROTOCOL_VERSION, &ldap_version);
+
+        berval.bv_len = message->length;
+        berval.bv_val = message->data;
+
+        reply->length = 0;
+        reply->data = NULL;
+
+        ldap_set_option(NULL, LDAP_OPT_DEBUG_LEVEL, &debug);
+        dprint("Before ldap_extended_operation_s:\n");
+        rc = ldap_extended_operation_s(ldap, KRBLDAP_OID_EXOP_AS_REQ, &berval,
NULL, NULL, &retoid, &retdata);
+        dprint("After ldap_extended_operation_s.\n");
+        dprint("exop rc: [%d]\n", rc);
+        if (rc == 0) {
```

```
+            *server_used = s;
+            done = TRUE;
+            dprint("Finished processing on server index [%d]. Ending loop.\n",
s);
+
+            reply->length = retdata->bv_len;
+            reply->data = malloc(reply->length);
+            if (reply->data == NULL) {
+                /* allocation failure */
+                retval = ENOMEM;
+                goto cleanup;
+            }
+            memcpy(reply->data, retdata->bv_val, retdata->bv_len);
+        }
+    }
+
+cleanup:
+    ber_memfree(retoid);
+    ber_bvfree(retdata);
+    return retval;
+}
+#endif
```