# Utilizing Secure LDAP to Centrally Manage ASE Authentication

**By Cory Sane**

The need for tighter security and better authentication methods are always in demand. Government oversight and corporate risk initiatives are forever evolving to make data security better. Yet better computer authentication methods are useless if password notes are posted on PC monitors or tucked neatly under keyboards or spoken out loud.

A shared password repository is an excellent way to eliminate multiple system passwords where each password has a different length, validation rule, and expiration date. Enterprises can deploy a Lightweight Directory Access Protocol (LDAP) server to manage a single password for each person in the organization. ASE is able to bypass the internally stored **syslogins** password and authenticate a user via a LDAP directory since version 12.5.1.

With recent improvements and customer demands to eliminate clear-text network transmissions, Sybase ASE 15.0.2 ESD #2 LDAP user authentication (LDAPUA) now supports the "LDAP secured" (LDAPS) and STARTTLS standards. These standards are the ability to encrypt network packets of the authentication requests between ASE and the LDAP server.

Sybase engineers have created a highly configurable LDAPUA interface. There are two types of binding methods to the LDAP server, along with primary and secondary LDAP server support, and two types of encryption available in 15.0.2 ESD #2.

*Cory Sane has worked with Sybase ASE for 14 years. He is a Database Technical Specialist in the financial services industry. He is a member of TeamSybase and can be reached at Cory@TheSanes.com*

## Configuration

I am using an ASE 15.0.2 ESD#2, OpenLDAP 2.2.27 and OpenSSL for this demonstration. OpenLDAP is one of several LDAP server products that Sybase ASE can interface with today. This article will show the LDAP servers residing on the same host as the ASE server. In a production environment, these entities should not co-exist on the same host.

The LDAP server entries were created with the following information:

```
dn: dc=thesanes,dc=com
dc: thesanes
objectClass: dcObject
objectClass: organizationalUnit
ou: The Sanes Dot Com

# people, thesanes.com
dn: ou=people,dc=thesanes,dc=com
ou: people
objectClass: organizationalUnit

# ASEvalid, people, thesanes.com
dn:  cn=ASEvalid,ou=people,dc=thesanes,dc=com
cn:  ASEvalid
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
sn: ASEvalid
userPassword:: <some value>

# cory, people, thesanes.com
dn: cn=cory,ou=people,dc=thesanes,dc=com
cn: cory
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
sn: sane
uid: cms4321
description: ASEuser
userPassword:: <some value>
```

The book *LDAP System Administration* by Gerald Carter from O'Reilly Media Inc. is a good source of information on defining the LDAP server. The Java based GUI LDAP browsers JXplorer (http://www.jxplorer.org) is a good tool for adding user entries to the LDAP server.

## Some basics

The ASE authentication process is regulated by the "enable ldap user auth" configuration parameter. The parameter has three values:

- 0=ldap not enabled
- 1=Authenticate in this order:
  - LDAP
  - ASE syslogins
  - Failure - 4002
- 2=Authenticate only via LDAP or failure 4002.

If option "2" is invoked, only the "sa" account can gain access to the ASE system via **syslogins** authentication. I have tried logins with both "sa_role" and "sso_role" – these also authenticate via the LDAP protocol. If the "sa" login is not available with option "2", the ASE dataserver should be stopped, the parameter edited to "0" or "1" in the server-name.cfg file, and the ASE dataserver restarted before another login can use **syslogins** to authenticate.

When an ASE login uses LDAP authentication, the **syslogins** password expiration date is not checked to force a new password. Complexity rules of the password need to be managed by the LDAP server because the ASE server can not govern the requirements of a password stored outside of its control.

ASE offers two choices for building the initial bind to the LDAP server. The "Composed DN" algorithm uses the ASE login as part of the distinguished name (DN) when binding to the LDAP server. If the ASE login cannot be used as part of the string, the "Searched DN" algorithm must be used because it supplies an "access-account" in the initial DN string that binds to the LDAP server. I will use the "Searched DN" algorithm in this demonstration because I never expect that the ASE login will match an LDAP DN. The access account is an account in the LDAP Server that can authenticate itself so that it can search the LDAP Server for valid ASE logins. In this demonstration, the access account will be defined as the following in the LDAP Server:

```
# ASEvalid, people, thesanes.com
dn: cn=ASEvalid,ou=people,dc=thesanes,dc=com
cn: ASEvalid
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
sn: ASEvalid
userPassword:: <some value>
```

ASE logins that are authenticating via LDAPUA must also be defined in the LDAP server. Here is the definition for one person in the LDAP server:

```
# cory, people, thesanes.com
dn: cn=cory,ou=people,dc=thesanes,dc=com
cn: cory
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
sn: sane
uid: cms4321
description: ASEuser
userPassword:: <some value>
```

There are four important attributes to this person.

- **dn** field is the unique distinguished name for this record
- **userPassword** will be compared to the supplied password from the ASE login attempt
- **uid** will be used in the search to match the ASE login name
- **description** is an additional filter that will be defined to insure that records can be used to authenticate ASE logins

## Turning on LDAPUA in ASE

The **sp_ldapadmin** stored procedure defines the LDAP server parameters to the ASE server. The full list of options for the **sp_ldapadmin** stored procedure can be found in the ASE Reference Manual. Minor improvements to **sp_ldapadmin** may be documented in the ASE New Features Guide. The stored procedure also has a "help" parameter.

Five steps are needed to build a minimum "Searched DN" LDAP definition: For this demonstration, I will set the value of "**enable ldap user auth**" to "**1**" because I still want some of my ASE logins to authenticate using **syslogins**.

```
exec sp_configure 'enable ldap user auth', 1

exec sp_ldapadmin set_primary_url, 'ldap://localhost:389/dc=the-
sanes,dc=com'

exec sp_ldapadmin set_dn_lookup_url,
'ldap://localhost:389/ou=people,dc=thesanes,dc=com??sub?(&(uid=*)(de
scription=ASEuser))'

exec sp_ldapadmin set_access_acct, "cn=ASEvalid,ou=people,dc=the-
sanes,dc=com", "mypassword"

exec sp_ldapadmin activate, primary
```

- The **set_primary_url** parameter defines the LDAP host:port /{base distinguished name of the search}
- The **set_dn_lookup_url** parameter defines the LDAP URL to be used for the search. The "*" is replaced with the userid attempting to be authenticated. This is a compound filtered search because a second attribute "**description=ASEuser**" must also be matched.
- The **set_access_acct** parameter defines the login used to bind to the LDAP service. If anonymous bind is not allowed on the LDAP server and the set_access_acct parameter is not correct in the ASE server, the **set_primary_url** and **set_dn_lookup_url** parameters can not be set correctly until the **set_access_acct** parameter is fixed or set to **null**. I always populate this parameter last, just before activating the definition.
- The **activate** parameter defines when the primary LDAP servers is usable. A **suspend** parameter is also defined. These parameters can be used to toggle LDAP traffic off and on for maintenance of the LDAP server.

Warning messages may appear and can be ignored when using the **sp_ldapadmin** tool to set the primary URL and the dn lookup URL until the access account is set.

A second LDAP directory can be defined for redundancy purposes. If the primary fails, the secondary will be queried. However, the primary LDAP definition must be manually reactivated if it becomes inactive because there is no mechanism for ASE to know when the primary LDAP definition is available.

I have seen enterprises create redundant LDAP services using domain name services tools. This allows the secondary LDAP entry stored in the ASE server to be used as a disaster recovery definition and never activated unless needed. Below is my definition of my secondary LDAP Server for failover:

```
exec sp_ldapadmin
'set_secondary_url','ldap://host2:389/dc=thesanes,dc=com'

exec sp_ldapadmin 'set_secondary_dn_lookup_url',
'ldap://host2:389/ou=people,dc=thesanes,dc=com??sub?(&(uid=*)
(description=ASEuser))'

exec sp_ldapadmin 'set_secondary_access_acct', "cn=ASEvalid,
ou=people,dc=thesanes,dc=com", ASEvalid

exec sp_ldapadmin activate, secondary
```

After the LDAP environment is defined in an ASE server, all logins by default will attempt to authenticate through the LDAP server unless a login has an explicit authentication method defined. ASE also allows for Kerberos and PAM authenticators. ASE uses the order of Kerberos, LDAP and then PAM if multiple authenticators are defined. Only one external authentication method will be attempted.

If the external authentication fails and "enable ldap user auth" has a value of "**1**", then ASE **syslogins** will be tried as a secondary authentication method. If there are ASE logins that should always use syslogins, then the **sp_addlogin** has the "**@auth_mech=**" parameter and **sp_modifylogin** has the "**authenticate with**" parameter for explicitly defining the authentication method. Values for these parameters are:

- **KERBEROS**
- **LDAP**
- **PAM**
- **ASE**
- **ANY (the highest ranked authentication mechanism active)**

Here are examples of creating and modifying ASE logins with explicit authentication directives:

```
-- Creating a new login with ASE authentication
sp_addlogin BatchCycle, "1502good", mydb, @auth_mech="ASE"

-- Modify a login to use ASE authentication
sp_modifylogin myacct, "authenticate with", "ASE"
```

This concludes the minimal steps needed to define unencrypted LDAP user authentication services. ASE login accounts can now authenticate via LDAP services. This is a good place to verify the work done so far. The ASE traceflag combination "3635, 3605" will print LDAP diagnostic information to the ASE server log. These traceflags have helped me find many mistakes in my work.

## Adding encryption

The LDAP protocol definition offers two methods of encryption.  The older LDAPS encryption method found in OpenLDAP (one of many LDAP implementations) has a “timeout” bug in many versions of the software libraries. LDAPS must use a different network port than unencrypted LDAP. The newer STARTTLS method can use the same port as unencrypted LDAP.

STARTTLS starts using clear-text network packets then switches to encrypted packets during the authentication process.  STARTTLS will be used for this demonstration. The first step to adding encryption is obtaining an encryption key and certificate. The command following creates a self-generated key and certificate from OpenSSL for our needs:

```
openssl req -new -x509 -nodes -out slapdcert.pem \
-keyout slapdkey.pem -days 365
```

The OpenLDAP slapd.conf file needs additional directives to enable STARTTLS encryption. These additional directives to OpenLDAP define the level of encryption allowed and location of the encryption key and certificate:

```
# SSL Certificate registration
TLSCipherSuite  HIGH:MEDIUM:+SSLv2
TLSCertificateFile  /etc/openldap/slapdcert.pem
TLSCertificateKeyFile /etc/openldap/slapdkey.pem
```

The public key (**slapdkey.pem**) created for the LDAP server also needs to be copied to the ASE host and renamed to [**aseservername**].txt so that the ASE server can find it. The file must reside in one of two places:

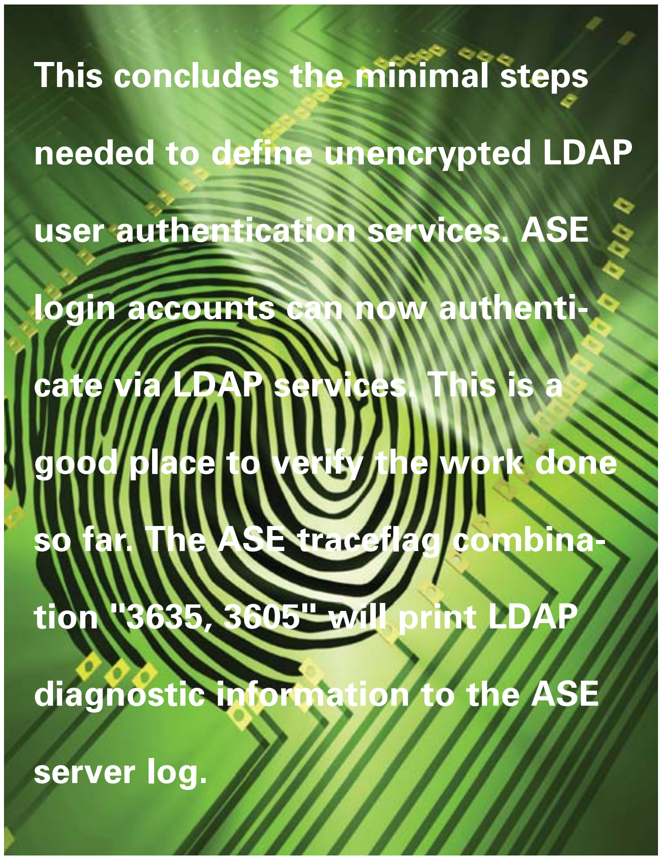
```
$SYBASE_CERTDIR/aseservername.txt
(If this environment variable is used, it must be defined  for the
environment starting the ASE dataserver programs.)

or

$SYBASE/$SYBASE_ASE/certificates/aseservername.txt
```

Then STARTTLS ability needs to be added to ASE with the following commands:

```
sp_ldapadmin 'starttls_on_primary', true
sp_ldapadmin 'starttls_on_secondary', true
sp_ldapadmin activate, primary
sp_ldapadmin activate, secondary
```



**This concludes the minimal steps needed to define unencrypted LDAP user authentication services. ASE login accounts can now authenticate via LDAP services. This is a good place to verify the work done so far. The ASE traceflag combination "3635, 3605" will print LDAP diagnostic information to the ASE server log.**

At this point, connections can be authenticated via LDAP with STARTTLS encryption.  The best way to insure STARTTLS functionallity is to enable to the (3635, 3605) traceflags and look for the following entry in the ASE server log during a login:

```
klbindalt: StartTLS is set to 'true'
```

The additional parameters of the ASE **sp_ldapadmin** can be used to tune the software.  Many of the parameters are already set to handle a high volume of concurrent logins. The **set_max_ldapua_desc** parameter has a default value of 20. This defines the number of logins an engine can handle at one time. If **set_abandon_ldapua_when_full** parameter is set to false, then the login is blocked until the engine has a login descriptor available.

## Shredding and recycling password notes

LDAP with STARTTLS encryption is an excellent tool for enterprises to build a shared password repository. This eliminates password notes, user fustrations and calls to the IT helpdesk. Sybase ASE offers an easy method to add LDAP user authentication to the DBMS. ■