SASL Pass-Through Authentication

Note: Not Currently Available for Solaris Operating Systems

Pass-through authentication is a mechanism used by some LDAP directories to delegate authentication operations (BIND) to other backends. It should be distinguished from the external authentication methods that are managed by the LDAP client to authenticate on a trusted source and then connect to the directory. Pass-through authentication is purely transparent for LDAP clients, as they send standard authentication operations to the LDAP directory, which will then handle the delegation and forward the response to the client, as though the authentication was done locally.

A real-world use case is the coexistence between OpenLDAP and Active Directory: the password is stored in AD, and OpenLDAP is configured to pass-through authentication between OpenLDAP and AD. With this setup, authentication is done on the OpenLDAP server using the AD password.

Technical description

OpenLDAP is known to be able to use pass-through authentication. This option should be compiled into it. If not, get the source and use this option in the configure step:

```
./configure --enable-spasswd --with-cyrus-sasl
```

This will allow you to store password with this syntax in userPassword attribute:

```
userPassword: {SASL}user@domain
```

This option is enabled in Symas OpenLDAP packages.

You then need the saslauthd daemon, which is available on most Linux distributions.

Pass-Through Authentication Process

1. A BIND operation is received by OpenLDAP with parameters DN1 and PWD1
2. OpenLDAP gets the DN1 entry and reads the userPassword attribute
3. DN1 password is a SASL password so OpenLDAP does an SASL authentication operation with user@domain and PWD1 credentials
4. SASL authentication daemon uses the credentials to look for the user in the backend (for example Active Directory) and gets the matching DN, DN2
5. SASL does a BIND operation with DN2 and PWD1
6. The backend manages the BIND and returns a response to SASL
7. SASL sends a response to OpenLDAP (yes/no)
8. OpenLDAP returns a response to the LDAP client

Pass-through Authentication on one LDAP directory

This is the standard use case: the password is stored in a directory and other LDAP directories delegate authentication to it.

1. Connect to the backend

You need to get all connection parameters to the authentication backend. An example with Active Directory:

```
Server address: ldap://ad.example.com
Bind DN: ADusername@example.com
```

```
Bind Password: ADpassword
User's branch: CN=DomainUsers,DC=example,DC=com
```

You can check these settings with an ldapsearch:

```
ldapsearch -x -H ldap://ad.example.com -D
ADusername@example.com -w ADpassword -b
CN=DomainUsers,DC=example,DC=com
```

2. Configure saslauthd

First, check that your SASL daemon supports LDAP:

```
saslauthd -v
```

If not, reinstall an LDAP aware saslauthd daemon.


```
sudo yum install cyrus-sasl -y
```


```
sudo apt-get install sasl2-bin -y
```

Follow these instructions:
http://software.opensuse.org/download.html?project=network&package=cyrus-sasl

Feature unavailable on Solaris Releases:
http://docs.oracle.com/cd/E26502_01/html/E29015/egyrr.html

Then, to activate LDAP as SASL mechanism:



```
sudo vi /etc/sysconfig/saslauthd
```


```
sudo vi /etc/default/saslauthd
```

```
...
SOCKETDIR=/var/run/saslauthd
# By default startTLS=yes
...
FLAGS="-O /etc/saslauthd.conf -d"
...
```

```
sudo chkconfig saslauthd on
```

Or

```
sudo systemctl enable saslauthd
```

To finish, enter all connection information found by the ldapsearch in step 1.

```
sudo vi /etc/saslauthd.conf
```

```
ldap_servers: ldap://ad.example.com
ldap_search_base: CN=DomainUsers,DC=example,DC=com
ldap_timeout: 10
ldap_filter: sAMAccountName=%U
ldap_bind_dn: <Bind DN from AD>
ldap_password: ADpassword
ldap_deref: never
ldap_restart: yes
ldap_scope: sub
ldap_use_sasl: no
ldap_start_tls: no
ldap_version: 3
ldap_auth_method: bind
log_level: 1
```

Main parameters are:

ldap_servers: LDAP URI, space separated for redundancy
ldap_bind_dn: DN for connection
ldap_password: Password for connection
ldap_search_base: Search base
ldap_filter: Search filter
ldap_scope: Search scope

In parameters ldap_search_base and ldap_filter, you can use these variables (example for SASL password user@domain):

```
%u: user@domain
%U: user
%d: domain
```

Restart saslauthd:

```
service saslauthd restart
```

3. Communication between OpenLDAP and saslauthd

The communication between the two daemons are done through a mutex, configured like this:

```
sudo vi /usr/lib/sasl2/slapd.conf
```

```
pwcheck_method: saslauthd
saslauthd_path: /var/run/saslauthd/mux
```

Add OpenLDAP user to sasl group (adapt names to your distribution settings):

```
usermod -a -G sasl <ldap user or root>
```

4. OpenLDAP configuration

Edit the global section of the OpenLDAP configuration (slapd.conf) with the SASL parameters:

```
sudo vi /opt/symas/etc/openldap/slapd.conf
        sasl-host       localhost
        sasl-secprops   none
        sasl-realm  <AD domain>
```

Restart OpenLDAP:

```
sudo service solserver restart
```

5. You can test the SASL functionality with this command:

```
testsaslauthd -u <your AD username> -p <your AD
password>
```

Create an account in OpenLDAP by creating an ldif file and adding it using the ldapadd command, for example:

```
sudo vi /opt/symas/etc/openldap/sasl-add.ldif
        dn: uid=<your AD username>,ou=Peons,dc=example,dc=com
        objectClass: inetOrgPerson
        objectClass: organizationalPerson
        objectClass: person
        objectClass: top
        uid: <your AD username>
        cn: <Your First Name + Last Name>
        sn: <your Last Name>
        userPassword: {SASL}<your AD username>@example.com
```

```
ldapadd -x -H ldap://localhost -D dc=example,dc=com -w
secret -f /opt/symas/etc/openldap/sasl-add.ldif
```

Now you can bind to OpenLDAP with AD password:

```
ldapsearch -x -H ldap://ldap.example.com -b
dc=example,dc=com -D uid=<your AD
username>,ou=Peons,dc=example,dc=com -W
```

You should be prompted for your AD password

Pass-through authentication on several LDAP directories - with OpenLDAP Meta backend

This section explains how configure Pass-through authentication on several LDAP backends with OpenLDAP Meta backend. This adds complexity as the SASL daemon can only be configured to connect to a single remote directory, and OpenLDAP cannot use several SASL authentication daemons. The solution described here uses a Meta directory between SASL daemon and remote directories. The choice of the backend to contact will be done in the SASL password value, for example {SASL}user@LDAP1 and {SASL}user@LDAP2.

You need to install all the components of the previous section. This section only describes the changes to be made from that configuration.

1. Create the Meta Directory

Configure a new OpenLDAP instance that will be a Meta directory for the LDAP backends, for example:

```
sudo vi /opt/symas/etc/openldap/slapd.conf
        # Database
        database      meta
        suffix "dc=local"
        rootdn "cn=Manager,dc=local"
        rootpw secret
```

```
# LDAP 1
uri    ldap://ldap1.example.com/ou=LDAP1,dc=local
lastmod      off
suffixmassage  "ou=LDAP1,dc=local" "dc=example1,dc=com"
idassert-bind bindmethod=simple
  binddn="cn=admin,dc=example1,dc=com"
  credentials="secret"
  mode=none
  flags=non-prescriptive
idassert-authzFrom "dn.exact:cn=Manager,dc=local"

# LDAP 2
uri    ldap://ldap2.example.com/ou=LDAP2,dc=local
lastmod      off
suffixmassage "ou=LDAP2,dc=local" "dc=example2,dc=com"
idassert-bind bindmethod=simple
  binddn="cn=admin,dc=example2,dc=com"
  credentials="secret"
  mode=none
  flags=non-prescriptive
idassert-authzFrom "dn.exact:cn=Manager,dc=local"
```

Launch this server on a new port (or another server), that will be accessible from SASL daemon. For example it will be launched on ldap://127.0.0.1:390/

2. Reconfigure saslauthd

Adapt SASL daemon configuration to contact the Meta directory:

```
vi /etc/saslauthd.conf
```

```
ldap_servers: ldap://127.0.0.1:390/
ldap_search_base: ou=%d,dc=local
ldap_timeout: 10
ldap_filter: (|(uid=%U)(SAMACCOUNTNAME=%U))
ldap_bind_dn: cn=Manager,dc=local
ldap_password: secret
ldap_deref: never
ldap_restart: yes
ldap_scope: sub
ldap_use_sasl: no
ldap_start_tls: no
ldap_version: 3
ldap_auth_method: bind
```

The interesting changes are:

ldap_search_base: we use the domain component (%d) to match to destination backend, through the Meta directory DIT

ldap_filter: we mix the filters with an OR filter, so that the user (%U) will be found whatever backend is called

Restart saslauthd:

```
service saslauthd restart
```

3. Testing

Redo the tests found in “Pass-through Authentication on one LDAP directory” step 5, with different users in LDAP1 and LDAP2, and

appropriate users in the main OpenLDAP server. By playing with the SASL password value, you are able to choose the authentication backend for pass-through authentication.

Pass-through authentication on several LDAP directories - with OpenLDAP ldap backend

This chapter explains how configure Pass-through authentication on several LDAP backends with OpenLDAP ldap backend. The advantage over the Meta backend is the possibility to use the rwm overlay with specific configuration for a backend directory, and for those using the cn=config backend, to manage the configuration into it (at the time of publication, backend Meta is not supported in cn=config).

1. Create the Proxy Directory

Configure a new OpenLDAP instance that will be a proxy directory for the LDAP backends, for example:

```
sudo vi /opt/symas/etc/openldap/slapd.conf
```

```
# Database LDAP for local Manager authentication
database ldap
suffix "cn=manager,dc=local"
rootdn "cn=manager,dc=local"
rootpw secret
# Database LDAP for LDAP 1
database        ldap
suffix          "ou=LDAP1,dc=local"
uri ldap://ldap1.example.com
idassert-bind bindmethod=simple
  binddn="cn=admin,dc=example1,dc=com"
  credentials="secret"
  mode=none
  flags=non-prescriptive
idassert-authzFrom "dn.exact:cn=Manager,dc=local"

overlay rwm
rwm-suffixmassage   "ou=LDAP1,dc=local" "dc=example,dc=com"

# Database LDAP for LDAP 2
database        ldap
suffix          "ou=LDAP1,dc=local"
uri  ldap://ldap2.example.com
idassert-bind bindmethod=simple
  binddn="cn=admin,dc=example2,dc=com"
  credentials="secret"
  mode=none
  flags=non-prescriptive
idassert-authzFrom "dn.exact:cn=Manager,dc=local"

overlay rwm
rwm-suffixmassage   "ou=LDAP1,dc=local" "dc=example,dc=com"

# Example of rwm configuration for Active Directory
rwm-map attribute uid sAMAccountName
rwm-map attribute * *
```

2. Reconfigure saslauthd

Adapt SASL daemon configuration to contact the meta directory:

```
vi /etc/saslauthd.conf
```

```
ldap_servers: ldap://127.0.0.1:390/
ldap_search_base: ou=%d,dc=local
ldap_timeout: 10
ldap_filter: uid=%U
ldap_bind_dn: cn=Manager,dc=local
ldap_password: secret
ldap_deref: never
ldap_restart: yes
ldap_scope: sub
ldap_use_sasl: no
ldap_start_tls: no
ldap_version: 3
ldap_auth_method: bind
```

We just changed the ldap_search_base parameter to use the domain component (%d) to match to destination backend, through the Meta directory DIT. You can keep a simple ldap_filter parameter, as we use rwm overlay to match the login attribute in both directories.

Restart saslauthd:

```
service saslauthd restart
```

3. Testing

Redo the tests found in “Pass-through Authentication on one LDAP directory” step 5, with different users in LDAP1 and LDAP2, and appropriate users in the main OpenLDAP server. By playing with the SASL password value, you are able to choose the authentication backend for pass-through authentication.