Setup Back_LDAP Proxy

The ldap backend redirects requests to another server, thus acting as a LDAP proxy. It allows the common configuration directives as suffix, which is used to select it when a request is received by the server, ACLs, which are applied to search results, size and time limits, and so on.

Configure the Database Section

Modify `/opt/symas/etc/openldap/slapd.conf` for any authentication backend with the following connection parameters:

```
database ldap
Server address: ldap://ad.example.com
Bind DN: CN=Administrator,CN=Users,DC=example,DC=com
Bind Password: AD_password
Users branch: CN=DomainUsers,DC=example,DC=com
```

The above Server Address is connected to the desired protocol:

```
uri ldap://host[:port]/
uri ldaps://host[:port]/
uri ldapi:///
```

These entries cannot have multiple values; the last supplied silently overrides the previous ones. The server and the uri directives can contain a whitespace or comma-separated list of values, that are passed to the underlying client library. The library tries to connect to each member of the list until one succeeds.

Bind with the Server

The default is to use simple bind, with empty binddn and credentials, which means that the related operations will be performed anonymously. If not set, and if idassert-bind is defined, this latter identity is used instead.

```
acl-bind bindmethod=simple|sasl
  [binddn=<simple DN>]
  [credentials=<simple password>]
  [saslmech=<SASL mech>]
  [secprops=<properties>]
  [realm=<realm>]
  [authcId=<authentication ID>]
  [authzId=<authorization ID>]
  [starttls=no|yes|critical]
  [tls_cert=<file>]
  [tls_key=<file>]
  [tls_cacert=<file>] | [tls_cacertdir=<path>]
  [tls_reqcert=never|allow|try|demand]
  [tls_cipher_suite=<ciphers>]
  [tls_protocol_min=<major>[.<minor>]]
  [tls_crlcheck=none|peer|all]
```

Use these settings to define the authentication method that is internally used by the proxy to collect info related to access control, and whenever an operation occurs with the identity of the rootdn of the LDAP proxy database. The identity defined by this directive, according to the properties associated to the authentication method, is supposed to have read access on the target server to attributes used on the proxy for ACL checking. There is no risk of giving away such values; they are only used to check permissions. The connection between the proxy database and the remote server associated to this identity is cached regardless of the lifespan of the client-proxy connection that first established it. This identity is not implicitly used by the proxy when the client connects anonymously. The idassert-bind feature, instead, in some cases can be crafted to implement that behavior, which is intrinsically unsafe and should be used with extreme care. This directive obsoletes acl-authcDN, and acl-passwd. The TLS settings default to the same as the main slapd TLS settings, except for tls_reqcert which defaults to "demand".

Options

cancel {ABANDON|ignore|exop[-discover]}

Defines how to handle operation cancellation. By default, abandon is invoked, so the operation is abandoned immediately. If set to ignore, no action is taken and any further response is ignored; this may result in further response messages to be queued for that connection, so it is recommended that long lasting connections are timed out either by idle-timeout or conn-ttl, so that resources eventually get released. If set to exop, a cancel operation (RFC 3909) is issued, resulting in the cancellation of the current operation; the cancel operation waits for remote server response, so its use may not be recommended. If set to exop-discover, support of the cancel extended operation is detected by reading the remote server's root DSE.

chase-referrals {YES|no}

enable/disable automatic referral chasing, which is delegated to the underlying libldap, with rebinding eventually performed if the rebind-as-user directive is used. The default is to chase referrals.

conn-ttl <time>

This directive causes a cached connection to be dropped and recreated after a given ttl, regardless of being idle or not.

idassert-authzFrom <authz-regexp>

If defined, selects what local identities are authorized to exploit the identity assertion feature. The string <authz-regexp> follows the rules defined for the authzFrom attribute. See slapd.conf(5), section related to authz-policy, for details on the syntax of this field.

idassert-bind bindmethod=none|simple|sasl

```
[binddn=<simple DN>]
[credentials=<simple password>]
[saslmech=<SASL mech>]
[secprops=<properties>]
[realm=<realm>]
[authcId=<authentication ID>]
[authzId=<authorization ID>]
[authz={native|proxyauthz}]
[mode=<mode>]
[flags=<flags>]
[starttls=no|yes|critical]
[tls_cert=<file>]
[tls_key=<file>]
[tls_cacert=<file>] | [tls_cacertdir=<path>]
[tls_reqcert=never|allow|try|demand]
[tls_cipher_suite=<ciphers>]
[tls_protocol_min=<version>]
[tls_crlcheck=none|peer|all]
```

Allows one to define the parameters of the authentication method that is internally used by the proxy to authorize connections that are authenticated by other databases. Direct binds are always proxied without any idassert handling. The identity defined by this directive, according to the properties associated to the authentication method, is supposed to have auth access on the target server to attributes used on the proxy for authentication and authorization, and to be allowed to authorize the users. This requires to have proxyAuthz privileges on a wide set of DNs, e.g. authzTo=dn.subtree:, and the remote server to have authz-policy set to to or both. See slapd.conf(5) for details on these statements and for remarks and drawbacks about their usage.

The supported bindmethods are none|simple|sasl where none is the default, i.e. no identity assertion is performed.

The authz parameter is used to instruct the SASL bind to exploit native SASL authorization, if available; since connections are cached, this should only be used when authorizing with a fixed identity (e.g. by means of the authzDN or authzID parameters). Otherwise, the default proxyauthz is used, i.e. the proxyAuthz control (Proxied Authorization, RFC 4370) is added to all operations.

The supported modes are: <mode> := {legacy|anonymous|none|self} If <mode> is not present, and authzId is given, the proxy always authorizes that identity. <authorization ID> can be u:<user>, [dn:]<DN>. The former is supposed to be expanded by the remote server according to the authz rules; see slapd.conf(5) for details. In the latter case, whether or not the dn: prefix is present, the string must pass DN validation and normalization. The default mode is legacy, which implies that the proxy will either perform a simple bind as the authcDN or a SASL bind as the authcID and assert the client's identity when it is not anonymous. The other modes imply that the proxy will always either perform a simple bind as the authcDN or a SASL bind as the authcID, unless restricted by idassert-authzFrom rules (see below), in which case the operation will fail; eventually, it will assert some other identity according to <mode>. Other identity assertion modes are anonymous and self, which respectively mean that the empty or the client's identity will be asserted; none, which means that no proxyAuthz control will be used, so the authcDN or the authcID identity will be asserted. For all modes that require the use of the proxyAuthz control, on the remote server the proxy identity must have appropriate authzTo permissions, or the asserted identities must have appropriate authzFrom permissions. Note, however, that the ID assertion feature is mostly useful when the asserted identities do not exist on the remote server.

Flags can be override,[non-]prescriptive,proxy-authz-[non-]critical. When the override flag is used, identity assertion takes place even when the database is authorizing for the identity of the client, i.e. after binding with the provided identity, and thus authenticating it, the proxy performs the identity assertion using the configured identity and authentication method. When the prescriptive flag is used (the default), operations fail with inappropriateAuthentication for those identities whose assertion is not allowed by the idassert-authzFrom patterns. If the non-prescriptive flag is used, operations are performed anonymously for those identities whose assertion is not allowed by the idassert-authzFrom patterns. When the proxy-authz-non-critical flag is used (the default), the proxyAuthz control is not marked as critical, in violation of RFC 4370. Use of proxy-authz-critical is recommended.

The TLS settings default to the same as the main slapd TLS settings, except for tls_reqcert which defaults to "demand". The identity associated to this directive is also used for privileged operations whenever idassert-bind is defined and acl-bind is not. See acl-bind for details.

idassert-passthru <authz-regexp>

If defined, selects what local identities bypass the identity assertion feature. Those identities need to be known by the remote host. The string <authz-regexp> follows the rules defined for the authzFrom attribute. See slapd.conf(5), section related to authz-policy, for details on the syntax of this field.

idle-timeout <time>

This directive causes a cached connection to be dropped an recreated after it has been idle for the specified time.

keepalive <idle>:<probes>:<interval>

The keepalive parameter sets the values of idle, probes, and interval used to check whether a socket is alive; idle is the number of seconds a connection needs to remain idle before TCP starts sending keepalive probes; probes is the maximum number of keepalive probes TCP should send before dropping the connection; interval is interval in seconds between individual keepalive probes. Only some systems support the customization of these values; the keepalive parameter is ignored otherwise, and system-wide settings are used.

network-timeout <time>

Sets the network timeout value after which poll(2)/select(2) following a connect(2) returns in case of no activity. The value is in seconds, and it can be specified as for idle-timeout.

norefs <NO|yes>

If yes, do not return search reference responses. By default, they are returned unless request is LDAPv2.

omit-unknown-schema <NO|yes>

If yes, do not return objectClasses or attributes that are not known to the local server. The default is to return all schema elements.

noundeffilter <NO|yes>

If yes, return success instead of searching if a filter is undefined or contains undefined portions. By default, the search is propagated after replacing undefined portions with (!(objectClass=*)), which corresponds to the empty result set.

onerr {CONTINUE|stop}

This directive allows one to select the behavior in case an error is returned by the remote server during a search. The default, continue, consists in returning success. If the value is set to stop, the error is returned to the client.

protocol-version {0,2,3}

This directive indicates what protocol version must be used to contact the remote server. If set to 0 (the default), the proxy uses the same protocol version used by the client, otherwise the requested protocol is used. The proxy returns unwillingToPerform if an operation that is incompatible with the requested protocol is attempted.

proxy-whoami {NO|yes}

Turns on proxying of the WhoAmI extended operation. If this option is given, back-ldap will replace slapd's original WhoAmI routine with its own. On slapd sessions that were authenticated by back-ldap, the WhoAmI request will be forwarded to the remote LDAP server. Other sessions will be handled by the local slapd, as before. This option is mainly useful in conjunction with Proxy Authorization.

quarantine <interval>,<num>[;<interval>,<num>[...]]

Turns on quarantine of URIs that returned LDAP_UNAVAILABLE, so that an attempt to reconnect only occurs at given intervals instead of any time a client requests an operation. The pattern is: retry only after at least interval seconds elapsed since last attempt, for exactly num times; then use the next pattern. If num for the last pattern is "+", it retries forever; otherwise, no more retries occur. The process can be restarted by resetting the olcDbQuarantine attribute of the database entry in the configuration backend.

rebind-as-user {NO|yes}

If this option is given, the client's bind credentials are remembered for rebinds, when trying to re-establish a broken connection, or when chasing a referral, if chase-referrals is set to yes.

session-tracking-request {NO|yes}

Adds session tracking control for all requests. The client's IP and hostname, and the identity associated to each request, if known, are sent to the remote server for informational purposes. This directive is incompatible with setting protocol-version to 2.

single-conn {NO|yes}

Discards current cached connection when the client rebinds.

t-f-support {NO|yes|discover}

Enable if the remote server supports absolute filters (see RFC 4526 for details). If set to discover, support is detected by reading the remote server's root DSE.

timeout [<op>=]<val> [...]

This directive allows one to set per-operation timeouts. Operations can be

<op> ::= bind, add, delete, modrdn, modify, compare, search

The overall duration of the search operation is controlled either by the timelimit parameter or by server-side enforced time limits (see timelimit and limits in slapd.conf(5) for details). This timeout parameter controls how long the target can be irresponsive before the operation is aborted. Timeout is meaningless for the remaining operations, unbind and abandon, which do not imply any response, while it is not yet implemented in currently supported extended operations. If no operation is specified, the timeout val affects all supported operations.

Note: if the timelimit is exceeded, the operation is cancelled (according to the cancel directive); the protocol does not provide any means to rollback operations, so the client will not be notified about the result of the operation, which may eventually succeeded or not. In case the timeout is exceeded during a bind operation, the connection is destroyed, according to RFC4511.

Note: in some cases, this backend may issue binds prior to other operations (e.g. to bind anonymously or with some prescribed identity according to the idassert-bind directive). In this case, the timeout of the operation that resulted in the bind is used.

tls {[try-]start|[try-]propagate|ldaps} [tls_cert=<file>] [tls_key=<file>] [tls_cacert=<file>] [tls_cacertdir=<path>] [tls_reqcert=never|allow|try|demand] [tls_cipher_suite=<ciphers>] [tls_crlcheck=none|peer|all]

Specify the use of TLS when a regular connection is initialized. The StartTLS extended operation will be used unless the URI directive protocol scheme is ldaps://. In that case this keyword may only be set to "ldaps" and the StartTLS operation will not be used. propagate issues the StartTLS operation only if the original connection did. The try- prefix instructs the proxy to continue operations if the StartTLS operation failed; its use is not recommended.

The TLS settings default to the same as the main slapd TLS settings, except for tls_reqcert which defaults to "demand".

use-temporary-conn {NO|yes}

When set to yes, create a temporary connection whenever competing with other threads for a shared one; otherwise, wait until the shared connection is available.

Access Control

The ldap backend does not honor all ACL semantics as described in slapd.access(5). In general, access checking is delegated to the remote server(s). Only read (=r) access to the entry pseudo-attribute and to the other attribute values of the entries returned by the search operation is honored, which is performed by the frontend.

Overlays

The LDAP backend provides basic proxying functionalities to many overlays. The chain overlay, described in slapo-chain(5), and the translucent overlay, described in slapo-translucent(5), deserve a special mention.
Conversely, there are many overlays that are best used in conjunction with the LDAP backend. The proxycache overlay allows caching of LDAP search requests (queries) in a local database. See slapo-pcache(5) for details. The rwm overlay provides DN rewrite and attribute/objectClass mapping capabilities to the underlying database. See slapo-rwm(5) for details.

Example 1: OpenLDAP as proxy to Active Directory

https://wiki.samba.org/index.php/OpenLDAP_as_proxy_to_AD
Use: If you don't want to have a DC with all its services and open ports in your DMZ, you can setup a back-ldap proxy with OpenLDAP. You can then limit access to your DC to just this one host and the LDAP port 389. All services on other hosts in your DMZ will access the AD using the proxy.

Use the following slapd.conf example:

```
### Schema includes
###########################################################
include      /opt/symas/etc/openldap/schema/core.schema
include      /opt/symas/etc/openldap/schema/cosine.schema
include      /opt/symas/etc/openldap/schema/inetorgperson.schema
include      /opt/symas/etc/openldap/schema/misc.schema
include      /opt/symas/etc/openldap/schema/rfc2307bis.schema

## Module paths ##############################################################
modulepath   /opt/symas/lib64/openldap/
moduleload   back_ldap
# Optional
moduleload   rwm

# Main settings
###############################################################
pidfile      /var/run/openldap/slapd.pid
argsfile     /var/run/openldap/slapd.args

### Database definition (Proxy to AD)
#########################################
database     ldap
readonly     yes
protocol-version    3
rebind-as-user      yes
uri    "ldap://{AD-Hostname/IP}:389"
```

```
suffix        "{your Domain DN}"

overlay       rwm
rwm-map       attribute       uid      sAMAccountName
rwm-map       attribute       mail     proxyAddresses

### Logging
###################################################################
loglevel     0
```

If you already have an OpenLDAP server with a local database running, you can just add the proxy part, as long as your AD resides in a different branch.

If you don't need to remap attributes (e.g. mapping "sAMAccountName" to "uid" and "proxyAddresses" to "mail" in the example above), you can skip these parameters. If you do remap attributes, then, when using ldap/slap commands, you may get errors similar to (for the above two remappings):

```
/etc/openldap/slapd.conf: line 28: warning, destination attributeType
'sAMAccountName' is not defined in schema
PROXIED attributeDescription "SAMACCOUNTNAME" inserted.
/etc/openldap/slapd.conf: line 29: warning, destination attributeType
'proxyAddresses' is not defined in schema
PROXIED attributeDescription "PROXYADDRESSES" inserted.
```

This happens if you remap attributes that are not defined in your included schemas. Search the web to get the valid schema entries, add them to `/opt/symas/etc/openldap/custom_schemas/mapping.schema` and include it in slapd.conf. For the above two mappings, the following should be in the schema file to stop the two errors occurring:

```
attributetype ( 1.2.840.113556.1.4.221
        NAME 'sAMAccountName'
        SYNTAX '1.3.6.1.4.1.1466.115.121.1.15'
        SINGLE-VALUE )

attributetype ( 1.2.840.113556.1.2.210
        NAME 'proxyAddresses'
        SYNTAX '1.3.6.1.4.1.1466.115.121.1.15' )
```

Restart the OpenLDAP service.

Example 2: NSLCD

Use: Retrieve user/groups from AD through OpenLDAP proxy (for Linux OpenLDAP Servers)

Example of where you need this: You need to resolve user/groups from AD through an OpenLDAP proxy, because you want to see the usernames/groups instead of UIDs/GIDs. Or you need to provide authentication to AD through the OpenLDAP proxy.

Note: This requires that you have successfully configured an OpenLDAP proxy to AD.

Create a new user in ADUC that nslcd will use for connecting to the AD (I used "nslcd-connect" in the example below).

Adapt the following "/etc/nlscd.conf" example to your environment:

```
# Mappings for Active Directory
```

```
pagesize 1000
referrals off

# Passwd
filter passwd (&(objectClass=posixAccount)(!(objectClass=computer))(uidNumber=*))
map    passwd homeDirectory     UnixHomeDirectory
map    passwd gecos             displayName
map    passwd gidNumber         primaryGroupID

# Shadow
filter shadow (&(objectClass=posixAccount)(!(objectClass=computer))(uidNumber=*))
map    shadow shadowLastChange  pwdLastSet

# Groups
filter group (&(objectClass=posixGroup)(gidNumber=*))
map    group uniqueMember       member

# Local account for nsclcd
uid nslcd
gid ldap

# Where is the LDAP
uri ldap://{openLDAP-Proxy-Hostname/IP}:389
base cn=Users,{your Domain DN}

# Connect-Account
binddn cn=nslcd-connect,cn=Users,{your Domain DN}
bindpw {password}
```

This example assumes, that you've mapped the attribute "sAMAccountName" to "uid", like in the example of OpenLDAP proxy to AD above. Otherwise you have to map the attribute here. Also it is required, that the user accounts have a uidNumber and the groups a gidNumber attribute.

Start the nslcd service.

Example 3: Authentication against AD through OpenLDAP proxy

Use: Example of where you need this: You want to authenticate users through an OpenLDAP proxy against AD.

Note: This requires that you have successfully configured Nslcd that uses an OpenLDAP proxy to AD to get the user information to the system.

Edit the "/etc/pam_ldap.conf" the following way:

```
base {your Domain DN}
binddn cn=nslcd-connect,cn=Users,{your Domain DN}
bindpw {password}
bind_policy soft
uri ldap://{openLDAP-Proxy-Hostname/IP}:389/
ssl no
```