NAME

`sudoers.ldap` - sudo LDAP configuration

DESCRIPTION

In addition to the standard `sudoers` file, `sudo` may be configured via LDAP. This can be especially useful for synchronizing `sudoers` in a large, distributed environment.

Using LDAP for `sudo` has several benefits:

- **`sudo`** no longer needs to read `sudoers` in its entirety. When LDAP is used, there are only two or three LDAP queries per invocation. This makes it especially fast and particularly usable in LDAP environments.

- **`sudo`** no longer exits if there is a typo in `sudoers`. It is not possible to load LDAP data into the server that does not conform to the `sudoers` schema, so proper syntax is guaranteed. It is still possible to have typos in a user or host name, but this will not prevent `sudo` from running.

- It is possible to specify per-entry options that override the global default options. `/etc/sudoers` only supports default options and limited options associated with user/host/commands/aliases. The syntax is complicated and can be difficult for users to understand. Placing the options directly in the entry is more natural.

- The `visudo` program is no longer needed. `visudo` provides locking and syntax checking of the `/etc/sudoers` file. Since LDAP updates are atomic, locking is no longer necessary. Because syntax is checked when the data is inserted into LDAP, there is no need for a specialized tool to check syntax.

Another major difference between LDAP and file-based `sudoers` is that in LDAP, `sudo`-specific Aliases are not supported.

For the most part, there is really no need for `sudo`-specific Aliases. Unix groups, non-Unix groups (via the `group_plugin`) or user netgroups can be used in place of User_Aliases and Runas_Aliases. Host netgroups can be used in place of Host_Aliases. Since groups and netgroups can also be stored in LDAP there is no real need for `sudo`-specific aliases.

Cmnd_Aliases are not really required either since it is possible to have multiple users listed in a `sudoRole`. Instead of defining a Cmnd_Alias that is referenced by multiple users, one can create a `sudoRole` that contains the commands and assign multiple users to it.

**SUDOers LDAP container**

The `sudoers` configuration is contained in the ou=SUDOers LDAP container.
Sudo first looks for the cn=default entry in the SUDOers container. If found, the multi-valued sudoOption attribute is parsed in the same manner as a global Defaults line in /etc/sudoers. In the following example, the SSH_AUTH_SOCK variable will be preserved in the environment for all users.

```ldif
dn: cn=defaults,ou=SUDOers,dc=example,dc=com
objectClass: top
objectClass: sudoRole
cn: defaults
description: Default sudoOption’s go here
sudoOption: env_keep+=SSH_AUTH_SOCK
```

The equivalent of a sudoer in LDAP is a sudoRole. It consists of the following attributes:

**sudoUser**
A user name, user ID (prefixed with ‘#’), Unix group name or ID (prefixed with ‘%’ or ‘%#’ respectively), user netgroup (prefixed with ‘+’), or non-Unix group name or ID (prefixed with ‘%:’ or ‘%:#’ respectively). Non-Unix group support is only available when an appropriate group_plugin is defined in the global defaults sudoRole object.

**sudoHost**
A host name, IP address, IP network, or host netgroup (prefixed with a ‘+’). The special value ALL will match any host.

**sudoCommand**
A fully-qualified Unix command name with optional command line arguments, potentially including globbing characters (aka wild cards). If a command name is preceded by an exclamation point, ‘!’ , the user will be prohibited from running that command.

The built-in command "sudoedit" is used to permit a user to run sudo with the -e option (or as sudoedit). It may take command line arguments just as a normal command does. Note that "sudoedit" is a command built into sudo itself and must be specified in without a leading path.

The special value ALL will match any command.

If a command name is prefixed with a SHA-2 digest, it will only be allowed if the digest matches. This may be useful in situations where the user invoking sudo has write access to the command or its parent directory. The following digest formats are supported: sha224, sha256, sha384 and sha512. The digest name must be followed by a colon (‘:’) and then the actual digest, in either hex or base64 format. For example, given the following value for sudoCommand:
sha224:0GomF8mNN3wlDt1HD9XldjJ3SNgpFdbjO1+NsQ /bin/ls

The user may only run /bin/ls if its sha224 digest matches the specified value. Command digests are only supported by version 1.8.7 or higher.

**sudoOption**

Identical in function to the global options described above, but specific to the sudoRole in which it resides.

**sudoRunAsUser**

A user name or uid (prefixed with ‘#’) that commands may be run as or a Unix group (prefixed with a ‘%’) or user netgroup (prefixed with a ‘+’) that contains a list of users that commands may be run as. The special value ALL will match any user.

The sudoRunAsUser attribute is only available in sudo versions 1.7.0 and higher. Older versions of sudo use the sudoRunAs attribute instead.

**sudoRunAsGroup**

A Unix group or gid (prefixed with ‘#’) that commands may be run as. The special value ALL will match any group.

The sudoRunAsGroup attribute is only available in sudo versions 1.7.0 and higher.

**sudoNotBefore**

A timestamp in the form yyyyymmdHHMMSSZ that can be used to provide a start date/time for when the sudoRole will be valid. If multiple sudoNotBefore entries are present, the earliest is used. Note that timestamps must be in Coordinated Universal Time (UTC), not the local timezone. The minute and seconds portions are optional, but some LDAP servers require that they be present (contrary to the RFC).

The sudoNotBefore attribute is only available in sudo versions 1.7.5 and higher and must be explicitly enabled via the SUDOERS_TIMED option in /etc/ldap.conf.

**sudoNotAfter**

A timestamp in the form yyyyymmdHHMMSSZ that indicates an expiration date/time, after which the sudoRole will no longer be valid. If multiple sudoNotBefore entries are present, the last one is used. Note that timestamps must be in Coordinated Universal Time (UTC), not the local timezone. The minute and seconds portions are optional, but some LDAP servers require that they be present (contrary to the RFC).
The sudoNotAfter attribute is only available in sudo versions 1.7.5 and higher and must be explicitly enabled via the SUDOERS_TIMED option in /etc/ldap.conf.

**sudoOrder**

The sudoRole entries retrieved from the LDAP directory have no inherent order. The sudoOrder attribute is an integer (or floating point value for LDAP servers that support it) that is used to sort the matching entries. This allows LDAP-based sudoers entries to more closely mimic the behavior of the sudoers file, where the order of the entries influences the result. If multiple entries match, the entry with the highest sudoOrder attribute is chosen. This corresponds to the "last match" behavior of the sudoers file. If the sudoOrder attribute is not present, a value of 0 is assumed.

The sudoOrder attribute is only available in sudo versions 1.7.5 and higher.

Each attribute listed above should contain a single value, but there may be multiple instances of each attribute type. A sudoRole must contain at least one sudoUser, sudoHost and sudoCommand.

The following example allows users in group wheel to run any command on any host via sudo:

```
dn: cn=%wheel,ou=SUDOers,dc=example,dc=com
objectClass: top
objectClass: sudoRole
  cn: %wheel
  sudoUser: %wheel
  sudoHost: ALL
  sudoCommand: ALL
```

**Anatomy of LDAP sudoers lookup**

When looking up a sudoer using LDAP there are only two or three LDAP queries per invocation. The first query is to parse the global options. The second is to match against the user’s name and the groups that the user belongs to. (The special ALL tag is matched in this query too.) If no match is returned for the user’s name and groups, a third query returns all entries containing user netgroups and other non-Unix groups and checks to see if the user belongs to any of them.

If timed entries are enabled with the SUDOERS_TIMED configuration directive, the LDAP queries include a sub-filter that limits retrieval to entries that satisfy the time constraints, if any.

If the NETGROUP_BASE configuration directive is present (see Configuring ldap.conf below), queries are performed to determine the list of netgroups the user belongs to before the sudoers query. This makes it possible to include netgroups in the sudoers query string in the same manner as Unix groups.
The third query mentioned above is not performed unless a group provider plugin is also configured. The actual LDAP queries performed by `sudo` are as follows:

1. Match all nisNetgroup records with a nisNetgroupTriple containing the user, host and NIS domain. The query will match nisNetgroupTriple entries with either the short or long form of the host name or no host name specified in the tuple. If the NIS domain is set, the query will match only match entries that include the domain or for which there is no domain present. If the NIS domain is `not` set, a wildcard is used to match any domain name but be aware that the NIS schema used by some LDAP servers may not support wild cards for nisNetgroupTriple.

2. Repeated queries are performed to find any nested nisNetgroup records with a memberNisNetgroup entry that refers to an already-matched record.

For sites with a large number of netgroups, using `NETGROUP_BASE` can significantly speed up `sudo`’s execution time.

**Differences between LDAP and non-LDAP sudoers**

There are some subtle differences in the way sudoers is handled once in LDAP. Probably the biggest is that according to the RFC, LDAP ordering is arbitrary and you cannot expect that Attributes and Entries are returned in any specific order.

The order in which different entries are applied can be controlled using the `sudoOrder` attribute, but there is no way to guarantee the order of attributes within a specific entry. If there are conflicting command rules in an entry, the negative takes precedence. This is called paranoid behavior (not necessarily the most specific match).

Here is an example:

```bash
# /etc/sudoers:
# Allow all commands except shell
johnny  ALL=(root)  ALL,!/bin/sh
# Always allows all commands because ALL is matched last
puddles ALL=(root) !/bin/sh,ALL

# LDAP equivalent of johnny
# Allows all commands except shell
dn: cn=role1,ou=Sudoers,dc=my-domain,dc=com
objectClass: sudoRole
objectClass: top
cn: role1
```
sudoUser: johnny
sudoHost: ALL
sudoCommand: ALL
sudoCommand: /bin/sh

# LDAP equivalent of puddles
# Notice that even though ALL comes last, it still behaves like
# role1 since the LDAP code assumes the more paranoid configuration
dn: cn=role2,ou=Sudoers,dc=my-domain,dc=com
objectClass: sudoRole
objectClass: top
cn: role2
sudoUser: puddles
sudoHost: ALL
sudoCommand: /bin/sh
sudoCommand: ALL

Another difference is that negations on the Host, User or Runas are currently ignored. For example, the following attributes do not behave the way one might expect.

# does not match all but joe
# rather, does not match anyone
sudoUser: !joe

# does not match all but joe
# rather, matches everyone including Joe
sudoUser: ALL
sudoUser: !joe

# does not match all but web01
# rather, matches all hosts including web01
sudoHost: ALL
sudoHost: !web01

Sudoers schema
In order to use sudo’s LDAP support, the sudo schema must be installed on your LDAP server. In addition, be sure to index the sudoUser attribute.

Three versions of the schema: one for OpenLDAP servers (schema.OpenLDAP), one for Netscape-derived servers (schema.iPlanet), and one for Microsoft Active Directory (schema.ActiveDirectory) may
be found in the **sudo** distribution.

The schema for **sudo** in OpenLDAP form is also included in the *EXAMPLES* section.

**Configuring ldap.conf**

Sudo reads the `/etc/ldap.conf` file for LDAP-specific configuration. Typically, this file is shared between different LDAP-aware clients. As such, most of the settings are not **sudo**-specific. Note that **sudo** parses `/etc/ldap.conf` itself and may support options that differ from those described in the system’s `ldap.conf(8)` manual. The path to `ldap.conf` may be overridden via the `ldap_conf` plugin argument in `sudo.conf(5)`.

Also note that on systems using the OpenLDAP libraries, default values specified in `/etc/openldap/ldap.conf` or the user’s `.ldaprc` files are not used.

Only those options explicitly listed in `/etc/ldap.conf` as being supported by **sudo** are honored. Configuration options are listed below in upper case but are parsed in a case-independent manner.

The pound sign (`#`) is used to indicate a comment. Both the comment character and any text after it, up to the end of the line, are ignored. Long lines can be continued with a backslash (`\`) as the last character on the line. Note that leading white space is removed from the beginning of lines even when the continuation character is used.

**BIND_TIMELIMIT** *seconds*

The **BIND_TIMELIMIT** parameter specifies the amount of time, in seconds, to wait while trying to connect to an LDAP server. If multiple **URIs** or **HOSTs** are specified, this is the amount of time to wait before trying the next one in the list.

**BINDDN** *DN*

The **BINDDN** parameter specifies the identity, in the form of a Distinguished Name (DN), to use when performing LDAP operations. If not specified, LDAP operations are performed with an anonymous identity. By default, most LDAP servers will allow anonymous access.

**BINDPW** *secret*

The **BINDPW** parameter specifies the password to use when performing LDAP operations. This is typically used in conjunction with the **BINDDN** parameter. The *secret* may be a plain text password or a base64-encoded string with a "base64:" prefix. For example:

```
BINDPW base64:dGVzdA==
```

If a plain text password is used, it should be a simple string without quotes. Plain text passwords
may not include the comment character (‘#’) and the escaping of special characters with a backslash (‘\’) is not supported.

**DEREF never/searching/finding/always**
How alias dereferencing is to be performed when searching. See the ldap.conf(8) manual for a full description of this option.

**HOST name[:port] ...**
If no URI is specified (see below), the HOST parameter specifies a white space-delimited list of LDAP servers to connect to. Each host may include an optional port separated by a colon (‘:’). The HOST parameter is deprecated in favor of the URI specification and is included for backwards compatibility only.

**KR5_CCNAME file name**
The path to the Kerberos 5 credential cache to use when authenticating with the remote server. This option is only relevant when using SASL authentication (see below).

**LDAP_VERSION number**
The version of the LDAP protocol to use when connecting to the server. The default value is protocol version 3.

**NETGROUP_BASE base**
The base DN to use when performing LDAP netgroup queries. Typically this is of the form ou=netgroup,dc=example,dc=com for the domain example.com. Multiple NETGROUP_BASE lines may be specified, in which case they are queried in the order specified.

This option can be used to query a user’s netgroups directly via LDAP which is usually faster than fetching every sudoRole object containing a sudoUser that begins with a ‘+’ prefix. The NIS schema used by some LDAP servers need a modificaton to support querying the nisNetgroup object by its nisNetgroupTriple member. OpenLDAP’s slapd requires the following change to the nisNetgroupTriple attribute:

```plaintext
attributetype ( 1.3.6.1.1.1.14 NAME ’nisNetgroupTriple’
    DESC ’Netgroup triple’
    EQUALITY caseIgnoreIA5Match
    SUBSTR caseIgnoreIA5SubstringsMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
```

**NETGROUP_SEARCH_FILTER ldap_filter**
An LDAP filter which is used to restrict the set of records returned when performing an LDAP
netgroup query. Typically, this is of the form attribute=value or
(&attribute=value)(attribute2=value2)). The default search filter is: objectClass=nisNetgroup. If
ldap_filter is omitted, no search filter will be used. This option is only when querying netgroups
directly via LDAP.

**NETWORK_TIMEOUT seconds**

An alias for **BIND_TIMELIMIT** provided for OpenLDAP compatibility.

**PORT port_number**

If no URI is specified, the **PORT** parameter specifies the default port to connect to on the LDAP
server if a **HOST** parameter does not specify the port itself. If no **PORT** parameter is used, the
default is port 389 for LDAP and port 636 for LDAP over TLS (SSL). The **PORT** parameter is
deprecated in favor of the **URI** specification and is included for backwards compatibility only.

**ROOTBINDDN DN**

The **ROOTBINDDN** parameter specifies the identity, in the form of a Distinguished Name (DN),
to use when performing privileged LDAP operations, such as **sudoers** queries. The password
Corresponding to the identity should be stored in the or the path specified by the **ldap_secret**
plugin argument in sudo.conf(5), which defaults to **/etc/ldap.secret**. If no **ROOTBINDDN** is
specified, the **BINDDN** identity is used (if any).

**ROOTUSE_SASL on/true/yes/off/false/no**

Enable **ROOTUSE_SASL** to enable SASL authentication when connecting to an LDAP server
from a privileged process, such as **sudo**.

**SASL_AUTH_ID identity**

The SASL user name to use when connecting to the LDAP server. By default, **sudo** will use an
anonymous connection. This option is only relevant when using SASL authentication.

**SASL_SECPROPS none/properties**

SASL security properties or **none** for no properties. See the SASL programmer’s manual for
details. This option is only relevant when using SASL authentication.

**SSL on/true/yes/off/false/no**

If the **SSL** parameter is set to on, true or yes, TLS (SSL) encryption is always used when
communicating with the LDAP server. Typically, this involves connecting to the server on port
636 (ldaps).

**SSL start_tls**

If the **SSL** parameter is set to start_tls, the LDAP server connection is initiated normally and TLS
encryption is begun before the bind credentials are sent. This has the advantage of not requiring a dedicated port for encrypted communications. This parameter is only supported by LDAP servers that honor the start_tls extension, such as the OpenLDAP and Tivoli Directory servers.

SUDOERS_BASE base
The base DN to use when performing sudo LDAP queries. Typically this is of the form ou=SUDOers,dc=example,dc=com for the domain example.com. Multiple SUDOERS_BASE lines may be specified, in which case they are queried in the order specified.

SUDOERS_DEBUG debug_level
This sets the debug level for sudo LDAP queries. Debugging information is printed to the standard error. A value of 1 results in a moderate amount of debugging information. A value of 2 shows the results of the matches themselves. This parameter should not be set in a production environment as the extra information is likely to confuse users.

The SUDOERS_DEBUG parameter is deprecated and will be removed in a future release. The same information is now logged via the sudo debugging framework using the "ldap" subsystem at priorities diag and info for debug_level values 1 and 2 respectively. See the sudo.conf(5) manual for details on how to configure sudo debugging.

SUDOERS_SEARCH_FILTER ldap_filter
An LDAP filter which is used to restrict the set of records returned when performing a sudo LDAP query. Typically, this is of the form attribute=value or (&(attribute=value)(attribute2=value2)). The default search filter is: objectClass=sudoRole. If ldap_filter is omitted, no search filter will be used.

SUDOERS_TIMED on/true/yes/off/false/no
Whether or not to evaluate the sudoNotBefore and sudoNotAfter attributes that implement time-dependent sudoers entries.

TIMELIMIT seconds
The TIMELIMIT parameter specifies the amount of time, in seconds, to wait for a response to an LDAP query.

TIMEOUT seconds
The TIMEOUT parameter specifies the amount of time, in seconds, to wait for a response from the various LDAP APIs.

TLS_CACERT file name
An alias for TLS_CACERTFILE for OpenLDAP compatibility.
TLS_CACERTFILE file name
The path to a certificate authority bundle which contains the certificates for all the Certificate Authorities the client knows to be valid, e.g. /etc/ssl/ca-bundle.pem. This option is only supported by the OpenLDAP libraries. Netscape-derived LDAP libraries use the same certificate database for CA and client certificates (see TLS_CERT).

TLS_CACERTDIR directory
Similar to TLS_CACERTFILE but instead of a file, it is a directory containing individual Certificate Authority certificates, e.g. /etc/ssl/certs. The directory specified by TLS_CACERTDIR is checked after TLS_CACERTFILE. This option is only supported by the OpenLDAP libraries.

TLS_CERT file name
The path to a file containing the client certificate which can be used to authenticate the client to the LDAP server. The certificate type depends on the LDAP libraries used.

OpenLDAP:

tls_cert /etc/ssl/client_cert.pem

Netscape-derived:

tls_cert /var/ldap/cert7.db

Tivoli Directory Server:

Unused, the key database specified by TLS_KEY contains both keys and certificates.

When using Netscape-derived libraries, this file may also contain Certificate Authority certificates.

TLS_CHECKPEER on/true/yes/off/false/no
If enabled, TLS_CHECKPEER will cause the LDAP server’s TLS certificated to be verified. If the server’s TLS certificate cannot be verified (usually because it is signed by an unknown certificate authority), sudo will be unable to connect to it. If TLS_CHECKPEER is disabled, no check is made. Note that disabling the check creates an opportunity for man-in-the-middle attacks since the server’s identity will not be authenticated. If possible, the CA’s certificate should be installed locally so it can be verified. This option is not supported by the Tivoli Directory Server LDAP libraries.

TLS_KEY file name
The path to a file containing the private key which matches the certificate specified by TLS_CERT. The private key must not be password-protected. The key type depends on the
LDAP libraries used.

OpenLDAP:
   tls_key /etc/ssl/client_key.pem

Netscape-derived:
   tls_key /var/ldap/key3.db

Tivoli Directory Server:
   tls_key /usr/ldap/ldapkey.kdb
When using Tivoli LDAP libraries, this file may also contain Certificate Authority and client certificates and may be encrypted.

**TLS_CIPHERS cipher list**
The **TLS_CIPHERS** parameter allows the administrator to restrict which encryption algorithms may be used for TLS (SSL) connections. See the OpenLDAP or Tivoli Directory Server manual for a list of valid ciphers. This option is not supported by Netscape-derived libraries.

**TLS_KEYPW secret**
The **TLS_KEYPW** contains the password used to decrypt the key database on clients using the Tivoli Directory Server LDAP library. The *secret* may be a plain text password or a base64-encoded string with a "base64:" prefix. For example:

   TLS_KEYPW base64:dGVzdA==

If a plain text password is used, it should be a simple string without quotes. Plain text passwords may not include the comment character ("#") and the escaping of special characters with a backslash (\) is not supported. If this option is used, /etc/ldap.conf must not be world-readable to avoid exposing the password. Alternately, a *stash file* can be used to store the password in encrypted form (see below).

If no **TLS_KEYPW** is specified, a *stash file* will be used if it exists. The *stash file* must have the same path as the file specified by **TLS_KEY**, but use a .sth file extension instead of .kdb, e.g. ldapkey.sth. The default ldapkey.kdb that ships with Tivoli Directory Server is encrypted with the password ssl_password. The gsk8capicmd utility can be used to manage the key database and create a *stash file*. This option is only supported by the Tivoli LDAP libraries.

**TLS_RANDFILE file name**
The **TLS_RANDFILE** parameter specifies the path to an entropy source for systems that lack a random device. It is generally used in conjunction with prngd or egd. This option is only
supported by the OpenLDAP libraries.

URI ldap[s]://[hostname[:port]] ...
Specifies a white space-delimited list of one or more URIs describing the LDAP server(s) to connect to. The protocol may be either ldap ldaps, the latter being for servers that support TLS (SSL) encryption. If no port is specified, the default is port 389 for ldap:// or port 636 for ldaps://. If no hostname is specified, sudo will connect to localhost. Multiple URI lines are treated identically to a URI line containing multiple entries. Only systems using the OpenSSL libraries support the mixing of ldap:// and ldaps:// URIs. Both the Netscape-derived and Tivoli LDAP libraries used on most commercial versions of Unix are only capable of supporting one or the other.

USE_SASL on/true/yes/off/false/no
Enable USE_SASL for LDAP servers that support SASL authentication.

ROOTSASL_AUTH_ID identity
The SASL user name to use when ROOTUSE_SASL is enabled.

See the ldap.conf entry in the EXAMPLES section.

Configuring nsswitch.conf
Unless it is disabled at build time, sudo consults the Name Service Switch file, /etc/nsswitch.conf, to specify the sudoers search order. Sudo looks for a line beginning with sudoers: and uses this to determine the search order. Note that sudo does not stop searching after the first match and later matches take precedence over earlier ones. The following sources are recognized:

files read sudoers from /etc/sudoers
ldap read sudoers from LDAP

In addition, the entry [NOTFOUND=return] will short-circuit the search if the user was not found in the preceding source.

To consult LDAP first followed by the local sudoers file (if it exists), use:

sudoers: ldap files

The local sudoers file can be ignored completely by using:

sudoers: ldap
If the `/etc/nsswitch.conf` file is not present or there is no sudoers line, the following default is assumed:

```
sudoers: files
```

Note that `/etc/nsswitch.conf` is supported even when the underlying operating system does not use an nsswitch.conf file, except on AIX (see below).

**Configuring netsvc.conf**
On AIX systems, the `/etc/netsvc.conf` file is consulted instead of `/etc/nsswitch.conf`. `sudo` simply treats netsvc.conf as a variant of nsswitch.conf; information in the previous section unrelated to the file format itself still applies.

To consult LDAP first followed by the local sudoers file (if it exists), use:

```
sudoers = ldap, files
```

The local *sudoers* file can be ignored completely by using:

```
sudoers = ldap
```

To treat LDAP as authoritative and only use the local sudoers file if the user is not present in LDAP, use:

```
sudoers = ldap = auth, files
```

Note that in the above example, the auth qualifier only affects user lookups; both LDAP and *sudoers* will be queried for Defaults entries.

If the `/etc/netsvc.conf` file is not present or there is no sudoers line, the following default is assumed:

```
sudoers = files
```

**Integration with sssd**
On systems with the *System Security Services Daemon* (SSSD) and where `sudo` has been built with SSSD support, it is possible to use SSSD to cache LDAP sudoers rules. To use SSSD as the sudoers source, you should use sssd instead of ldap for the sudoers entry in `/etc/nsswitch.conf`. Note that the `/etc/ldap.conf` file is not used by the SSSD *sudo* back end. Please see `sssd-sudo(5)` for more information on configuring *sudo* to work with SSSD.

**FILES**

`SUDOERS.LDAP(8)`  
System Manager’s Manual  
`SUDOERS.LDAP(8)`

Sudo 1.8.12  January 30, 2015  Sudo 1.8.12
**EXAMPLES**

**Example ldap.conf**

```
# Either specify one or more URIs or one or more host:port pairs.
# If neither is specified sudo will default to localhost, port 389.
#
#host ldapserver
#host ldapserver1 ldapserver2:390
#
# Default port if host is specified without one, defaults to 389.
#port 389
#
# URI will override the host and port settings.
uri ldap://ldapserver
#uri ldaps://secureldapserver
#uri ldaps://secureldapserver ldap://ldapserver
#
# The amount of time, in seconds, to wait while trying to connect to
# an LDAP server.
bind_timelimit 30
#
# The amount of time, in seconds, to wait while performing an LDAP query.
timelimit 30
#
# Must be set or sudo will ignore LDAP; may be specified multiple times.
sudoers_base ou=SUDOers,dc=example,dc=com
#
# verbose sudoers matching from ldap
#sudoers_debug 2
#
# Enable support for time-based entries in sudoers.
#sudoers_timed yes
#
# optional proxy credentials
#binddn <who to search as>
```
#bindpw  <password>
#rootbinddn  <who to search as, uses /etc/ldap.secret for bindpw>
#
# LDAP protocol version, defaults to 3
#ldap_version 3
#
# Define if you want to use an encrypted LDAP connection.
# Typically, you must also set the port to 636 (ldaps).
#ssl on
#
# Define if you want to use port 389 and switch to
# encryption before the bind credentials are sent.
# Only supported by LDAP servers that support the start_tls
# extension such as OpenLDAP.
#ssl start_tls
#
# Additional TLS options follow that allow tweaking of the
# SSL/TLS connection.
#
#tls_checkpeer yes  # verify server SSL certificate
#tls_checkpeer no   # ignore server SSL certificate
#
# If you enable tls_checkpeer, specify either tls_cacertfile
# or tls_cacertdir. Only supported when using OpenLDAP.
#
#tls_cacertfile /etc/certs/trusted_signers.pem
#tls_cacertdir /etc/certs
#
# For systems that don’t have /dev/random
# use this along with PRNGD or EG D.pl to seed the
# random number pool to generate cryptographic session keys.
# Only supported when using OpenLDAP.
#
#tls_randfile /etc/egd-pool
#
# You may restrict which ciphers are used. Consult your SSL
# documentation for which options go here.
# Only supported when using OpenLDAP.
#
#tls_ciphers <cipher-list>
# Sudo can provide a client certificate when communicating to
# the LDAP server.
# Tips:
# * Enable both lines at the same time.
# * Do not password protect the key file.
# * Ensure the keyfile is only readable by root.
#
# For OpenLDAP:
#tls_cert /etc/certs/client_cert.pem
#tls_key /etc/certs/client_key.pem
#
# For SunONE or iPlanet LDAP, tls_cert and tls_key may specify either
# a directory, in which case the files in the directory must have the
# default names (e.g. cert8.db and key4.db), or the path to the cert
# and key files themselves. However, a bug in version 5.0 of the LDAP
# SDK will prevent specific file names from working. For this reason
# it is suggested that tls_cert and tls_key be set to a directory,
# not a file name.
#
# The certificate database specified by tls_cert may contain CA certs
# and/or the client’s cert. If the client’s cert is included, tls_key
# should be specified as well.
# For backward compatibility, ”sslpath” may be used in place of tls_cert.
#tls_cert /var/ldap
#tls_key /var/ldap
#
# If using SASL authentication for LDAP (OpenSSL)
#use_sasl yes
#sasl_auth_id <SASL user name>
#rootuse_sasl yes
#rootsasl_auth_id <SASL user name for root access>
#sasl_secprops none
#krb5_ccname /etc/ldapcache

**Sudo schema for OpenLDAP**
The following schema, in OpenLDAP format, is included with `sudo` source and binary distributions as `schema.OpenLDAP`. Simply copy it to the schema directory (e.g. `/etc/openldap/schema`), add the proper include line in `slapd.conf` and restart `slapd`. 
attributetype ( 1.3.6.1.4.1.15953.9.1.1
  NAME 'sudoUser'
  DESC 'User(s) who may run sudo'
  EQUALITY caseExactIA5Match
  SUBSTR caseExactIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.15953.9.1.2
  NAME 'sudoHost'
  DESC 'Host(s) who may run sudo'
  EQUALITY caseExactIA5Match
  SUBSTR caseExactIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.15953.9.1.3
  NAME 'sudoCommand'
  DESC 'Command(s) to be executed by sudo'
  EQUALITY caseExactIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.15953.9.1.4
  NAME 'sudoRunAs'
  DESC 'User(s) impersonated by sudo'
  EQUALITY caseExactIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.15953.9.1.5
  NAME 'sudoOption'
  DESC 'Options(s) followed by sudo'
  EQUALITY caseExactIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.15953.9.1.6
  NAME 'sudoRunAsUser'
  DESC 'User(s) impersonated by sudo'
  EQUALITY caseExactIA5Match
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.15953.9.1.7
  NAME 'sudoRunAsGroup'
DESC 'Group(s) impersonated by sudo'
EQUALITY caseExactIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

attributetype ( 1.3.6.1.4.1.15953.9.1.8
  NAME 'sudoNotBefore'
  DESC 'Start of time interval for which the entry is valid'
  EQUALITY generalizedTimeMatch
  ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )

attributetype ( 1.3.6.1.4.1.15953.9.1.9
  NAME 'sudoNotAfter'
  DESC 'End of time interval for which the entry is valid'
  EQUALITY generalizedTimeMatch
  ORDERING generalizedTimeOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 )

attributeTypes ( 1.3.6.1.4.1.15953.9.1.10
  NAME 'sudoOrder'
  DESC 'an integer to order the sudoRole entries’
  EQUALITY integerMatch
  ORDERING integerOrderingMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )

objectclass ( 1.3.6.1.4.1.15953.9.2.1 NAME 'sudoRole' SUP top STRUCTURAL
  DESC 'Sudoer Entries’
  MUST ( cn )
  MAY ( sudoUser $ sudoHost $ sudoCommand $ sudoRunAs $ sudoRunAsUser $ sudoRunAsGroup $ sudoOption $ sudoNotBefore $ sudoNotAfter $ sudoOrder $ description )
)

SEE ALSO
ldap.conf(5), sssd-sudo(5), sudo.conf(5), sudoers(8)

CAVEATS
Note that there are differences in the way that LDAP-based sudoers is parsed compared to file-based sudoers. See the Differences between LDAP and non-LDAP sudoers section for more information.
BUGS
If you feel you have found a bug in sudo, please submit a bug report at https://bugzilla.sudo.ws/

SUPPORT
Limited free support is available via the sudo-users mailing list, see
https://www.sudo.ws/mailman/listinfo/sudo-users to subscribe or search the archives.

DISCLAIMER
sudo is provided "AS IS" and any express or implied warranties, including, but not limited to, the
implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the
LICENSE file distributed with sudo or https://www.sudo.ws/license.html for complete details.