

Dante Module Documentation

LDAP Module (Dante 1.4.3)

Inferno Nettverk A/S

Date: 2021/04/28 17:36:32

1 Description

The LDAP module extends the functionality of the Dante SOCKS server by integrating LDAP-based *authentication* and *authorization*.

Authentication The *ldapauth* authentication method verifies a given username/password combination, provided to Dante by the SOCKS client, with the appropriate LDAP server.

If the username/password combination does not match that stored at the LDAP server, Dante blocks the client.

Authorization The *ldap.group* authorization functionality supports access control based on a users LDAP group membership. The Privilege Account Certificate (PAC) functionality furthermore adds Kerberos group based user access control.

This can be used to limit the network access of different SOCKS users based on their LDAP and/or Kerberos group membership.

The location of LDAP servers in a network can be specified either explicitly in the Dante server's configuration file, or it can be discovered automatically by the LDAP module. This makes it easier to integrate Dante in existing GSSAPI/LDAP setups, e.g., networks using Active Directory.

2 LDAP server identity

In some configurations, the LDAP module will be able to do most of the work involved in identifying and contacting the LDAP servers in a network without further configuration required in Dante.

If the username provided to Dante by the SOCKS client contains a domain extension, the LDAP module determines the LDAP server automatically using the following method to obtain a list of available servers:

1. Extract the domain name from the username.

E.g., *DOMAIN.COM* from *user@DOMAIN.COM*, either when GSSAPI authentication is used, or if the username includes a domain with username/password authentication.

2. Perform a DNS SRV record lookup of the domain name (typically available in a Windows environment with Active Directory):

Without SSL: from *_ldap._tcp.DOMAIN.COM*.

With SSL: from *_ldaps._tcp.DOMAIN.COM*.

If this entry does not exist, follow the same procedure as *Without SSL*.

3. Perform a DNS *A record* lookup of *DOMAIN.COM*.
4. Use */etc/hosts* file entry for *DOMAIN.COM*.
5. Sort entries by weight and priority and remove duplicates.

If the username does not contain any domain extension, a pre-configured LDAP URL can be set in the Dante configuration file and used to verify the LDAP group membership of users.

The module authenticates to the LDAP server using SASL/GSSAPI with the appropriate entry of the GSSAPI or LDAP specific keytab, or the username/password provided as part of the LDAP URL.

3 Configuration examples

This section starts with some general templates for Dante configurations, without LDAP functionality, and then shows how these templates can be modified and extended to use the LDAP module functionality in various usage scenarios.

3.1 Dante PAM configuration – no LDAP functionality

Inferno Nettverk A/S provides Dante in a package that consists of both a SOCKS client and a SOCKS server. While part of the same package, they work independent of each others and one is not required for the other.

When *PAM* is used to verify usernames/passwords in the server, the method *username* should be used in the client.

3.1.1 Client configuration

```
logoutoutput: socks.log
# debug: 1

route {
    from: 0/0 to: 0/0 via: 10.0.0.1 port = 1080
    proxyprotocol: socks_v5
    method: username
}
```

NOTE: The client username method sends the username/password in plaintext and may not be appropriate for use unless combined with other security mechanisms.

3.1.2 Server configuration

With PAM, the Dante server will usually need *root* privileges to verify the username/-password combination received from the user, though this will depend on the PAM configuration on the given system, which is controlled externally to Dante.

```
logoutput: /var/log/sockd.log
# debug: 1

internal: eth0 port = 1080
external: eth1

socksmethod: pam.username

user.privileged      : root
user.notprivileged   : sockd

client pass {
    from: 0/0 to: 0/0
    log: connect disconnect error
}

socks pass {
    from: 0/0 to: 0/0
    log: connect disconnect error
}
```

3.2 LDAP Authentication

The *LDAP Authentication* mechanism uses the *ldapauth* method for authentication. Like with PAM, the client configuration file needs to specify the *username* method to supply the username/password to the Dante server.

3.2.1 Client configuration

```
logoutput: socks.log
# debug: 1

route {
    from: 0/0 to: 0/0 via: 10.0.0.1 port = 1080
    proxyprotocol: socks_v5
    method: username
}
```

NOTE: The client username method sends the username/password in plaintext and may not be appropriate for use unless combined with other security mechanisms.

3.2.2 Server configuration – Automated server detection

In contrast with PAM, the Dante server does *not* require *root* privileges to verify the username/password combination received from the user.

The configuration given next requires the LDAP server to be findable via DNS. Unless this functionality is disabled, or an LDAP server is explicitly specified in the Dante configuration file, the LDAP module will attempt to automatically determine the name and address of the LDAP server.

```
logoutput: /var/log/sockd.log
# debug: 1

internal: eth0 port = 1080
external: eth1

socksmethod: ldapauth # ldap authentication

user.privileged      : sockd # extra privileges not required
user.notprivileged   : sockd

client pass {
    from: 0/0 to: 0/0
    log: connect disconnect error
}

socks pass {
    from: 0/0 to: 0/0
    log: connect disconnect error
}
```

3.2.3 Server configuration – Hardcoded server configuration

The location of the LDAP server, or servers, can also be specified directly in the Dante configuration file.

This configuration specifies the simplest and most compact way of providing the server information. The *ldap.auth.url* keyword is used to specify the name of the LDAP server, along with the username and password to use for accessing the LDAP server. SSL is disabled in this example.

```
logoutoutput: /var/log/sockd.log
# debug: 1

internal: eth0 port = 1080
external: eth1

socksmethod: ldapauth # ldap authentication

user.privileged      : sockd
user.notprivileged   : sockd

client pass {
    from: 0/0 to: 0/0
    log: connect disconnect error
}

socks pass {
    from: 0/0 to: 0/0
    log: connect disconnect error

    ldap.auth.auto.off: yes # disable automatic ldap server lookup
    ldap.auth.ssl: no      # disable ssl

    # use the below URL, with username and password, for accessing
    # the LDAP server.
    ldap.auth.url: ldap://user:pass@ldap.example.com/basedn
}
```

3.2.4 Server configuration – SSL protected LDAP lookup

This is a variant of the previous configuration, with the LDAP server hardcoded, and SSL enabled for the connection between the Dante server and the LDAP server.

The *ldap.auth.url* keyword is used to specify the name of the LDAP server, along with the username and password to use for accessing the LDAP server. SSL is enabled in this example.

```
logoutoutput: /var/log/sockd.log
# debug: 1

internal: eth0 port = 1080
external: eth1

socksmethod: ldapauth # ldap authentication

user.privileged      : sockd
user.notprivileged   : sockd

client pass {
    from: 0/0 to: 0/0
    log: connect disconnect error
}

socks pass {
    from: 0/0 to: 0/0
    log: connect disconnect error

    ldap.auth.auto.off: yes # disable auto ldap server lookup
    ldap.auth.certcheck: yes # certificate check enabled

    # LDAP server, specified with ldaps url
    ldap.auth.url: ldaps://user:pass@ldap.example.com/basedn
}
```


3.2.5 Server configuration – SASL/GSSAPI LDAP lookup

The communication between the Dante server and the LDAP server can also be done over SASL/GSSAPI:

```
logoutput: /var/log/sockd.log
# debug: 1

internal: eth0 port = 1080
external: eth1

socksmethod: ldapauth none

user.privileged      : sockd
user.notprivileged   : sockd

client pass {
    from: 0/0 to: 0/0
}
socks pass {
    from: 0/0 to: 0/0

    ldap.auth.keytab: /etc/sockd-ldap.keytab
    ldap.auth.domain: EXAMPLE.COM
    ldap.auth.url: ldaps://ldap.example.com
}
```

3.3 Dante GSSAPI configuration – no LDAP functionality

With GSSAPI, users already authenticated to a Windows AD server or similar can automatically authenticate to the Dante SOCKS server, which will result in all communication between the client and the Dante server being encrypted.

3.3.1 Client configuration

```
logoutput: socks.log
# debug: 1

route {
    from: 0/0 to: 0/0 via: 10.0.0.1 port = 1080
    proxyprotocol: socks_v5
    method: gssapi
}
```

3.3.2 Server configuration

The Dante server requires a keytab file, that is specified in the *client pass* rule.

```
logoutput: /var/log/sockd.log
# debug: 1

internal: eth0 port = 1080
external: eth1

socksmethod: gssapi

user.privileged      : root
user.notprivileged   : sockd

client pass {
    from: 0/0 to: 0/0
    log: connect disconnect error

    # keytab
    gssapi.keytab: /etc/sockd.keytab
}

socks pass {
    from: 0/0 to: 0/0
    log: connect disconnect error
}
```

3.4 LDAP Authorization

The LDAP authorization functionality is typically used with GSSAPI authentication, with membership to a LDAP group required for users to have sessions forwarded by the Dante SOCKS server.

For Windows clients, the OpenText (formerly Hummingbird) client can be used (see <https://connectivity.opentext.com/products/socks-client.aspx>).

As with the LDAP authentication configurations, the LDAP module will by default attempt to automatically locate the LDAP server via DNS. The config file below does not specify any LDAP server or disable the automatic lookup, so DNS will be used. The name of the LDAP server, and how communication between the LDAP module and the LDAP server should be handled, can be configured for the LDAP authorization functionality in the same way as for the LDAP authentication functionality, with the difference being that instead of *ldap.auth*, the prefix is *ldap* (e.g., *ldap.url*, etc.).

3.4.1 Client configuration

No changes are needed to the GSSAPI client configuration:

```
logoutput: socks.log
# debug: 1

route {
    from: 0/0 to: 0/0 via: 10.0.0.1 port = 1080
    proxyprotocol: socks_v5
    method: gssapi
}
```

3.4.2 Server configuration – Limiting access to web/http

The rules below shows an example of how one can limit access to web sites from clients on the 10.0.0.0/8 network to members of the *SOCKS_ALLOW* LDAP group.

```
logoutput: /var/log/sockd.log
# debug: 1

internal: eth0 port = 1080
external: eth1

socksmethod: gssapi

user.privileged      : sockd
user.notprivileged   : sockd

client pass {
    from: 0/0 to: 0/0
    log: connect disconnect error

    # keytab for GSSAPI authentication
    gssapi.keytab: /etc/sockd.keytab
}

pass {
    from: 10.0.0.0/8 to: 0/0 port = http

    # only members of LDAP group can access via this rule.
    ldap.group: SOCKS_ALLOW
}
```

For an OpenLDAP server with a *rfc2307bis schema* or an Active Directory server, with the configuration example given in the Appendix, *User1* and *User2* will be allowed, whereas *User3* will be refused access.

3.4.3 Server configuration – Limiting access to SSL VPNs

The next rule, if placed before other general rules, shows how one can limit access for temporary staff on the 10.0.0.0/8 network to only a specific work related site.

```
logoutoutput: /var/log/sockd.log
# debug: 1

internal: eth0 port = 1080
external: eth1

socksmethod: gssapi

user.privileged      : sockd
user.notprivileged   : sockd

client pass {
    from: 10.0.0.0/8 to: 0/0
}

pass {
    from: 10.0.0.0/8 to: sslvpn.example.com port = 443
    command: connect

    ldap.group: Temporary
    ldap.filter: (uid=%s)
    ldap.attribute: employeeType
}

pass {
    from: 10.0.0.0/8 to: 0/0 port = 443
    command: connect

    ldap.group: Permanent
    ldap.filter: (uid=%s)
    ldap.attribute: employeeType
}
```

Assuming the OpenLDAP configuration in the Appendix example is used, the temporary user *User3* is only allowed to connect to *sslvpn.example.com* on port 443 whereas the permanent users *User1* and *User2* can connect to any secure web site via *https*.

3.4.4 Server configuration – Limiting ftp to company employees only

The next rule shows how one can limit access to ftp sites to company employees on the 10.0.0.0/8 network only.

```
logoutoutput: /var/log/sockd.log
# debug: 1

internal: eth0 port = 1080
external: eth1

socksmethod: gssapi

user.privileged      : sockd
user.notprivileged   : sockd

client pass {
    from: 10.0.0.0/8 to: 0/0
}

pass {
    from: 10.0.0.0/8 to: 0/0 port = ftp

    ldap.group: MyCompany
    ldap.keeprealm: yes
    ldap.filter.ad: (userprincipalname=%s)
    ldap.attribute.ad: company
}

block { # other users are not allowed to connect to FTP servers.
    from: 0/0 to: 0/0 port = ftp
}

pass { # access to all-non FTP ports allowed for everyone.
    from: 0/0 to: 0/0
}
```

Assuming the Active Directory example in the Appendix is used, *User3* is only allowed to connect to ftp data whereas the users *User1* and *User2* are not allowed.

3.5 Server configuration – LDAP URL usage

The next rule shows how one can limit access to ftp sites for company employees on the 10.0.0.0/8 network without requiring GSSAPI authentication. An LDAP URL with a directly specified username (here *user*) and password (here *pass*) is used for authentication.

In this example, two LDAP servers are specified, with the second server (*ldap2*) contacted only if *ldap1* is not available.

```
logoutoutput: /var/log/sockd.log
# debug: 1

internal: eth0 port = 1080
external: eth1

socksmethod: gssapi

user.privileged      : sockd
user.notprivileged   : sockd

client pass {
    from: 10.0.0.0/8 to: 0/0
}

pass {
    from: 10.0.0.0/8 to: 0/0 port = ftp
    command: connect
    ldap.group: SOCKS_ALLOW
    ldap.auto.off: yes
    ldap.url: ldap://user:pass@ldap1.example.com:389/OU=SALES,DC=MYCOMPANY,DC=EXAMPLE.COM
    ldap.url: ldap://user:pass@ldap2.example.com:389/OU=SALES,DC=MYCOMPANY,DC=EXAMPLE.COM
}

block { # other users are not allowed to connect to FTP servers.
    from: 0/0 to: 0/0 port = ftp
}

pass { # access to all-non FTP ports allowed for everyone.
    from: 0/0 to: 0/0
}
```

3.6 PAC Authorization

The Privilege Account Certificate (PAC) functionality relies on the SOCKS client using GSSAPI authentication with the Dante server, and uses the Microsoft Kerberos PAC authorisation-data field. This is an extension element of the authorization-data field contained in the client's Kerberos ticket (See https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-pac/).

PAC requires the user to have authenticated with GSSAPI. For Windows clients, the OpenText (formerly Hummingbird) client can be used (see <https://connectivity.opentext.com/products/socks-client.aspx>).

3.6.1 Finding PAC Group SIDs

To get the SID from an Active Directory Server use *ldapsearch*, or a similar command with SASL/GSSAPI authentication to an Active Directory server.

```
ldapsearch -LLL -H ldap://dc1.samba.home:389 -s sub \
  -b "OU=testgroups,dc=samba,dc=home" "(CN=SOCKS_ALLOW)" objectsid
```

The above command should produce output similar to the below:

```
filter: (cn=SOCKS_ALLOW)
requesting: objectsid
dn: CN=SOCKS_ALLOW,OU=TestGroups,DC=samba,DC=home
objectSid:: AQUAAAAAAAAUVAAAA3e5/WdBj9hHz1/+pVgQAAA==
```

The objectSid value can then be converted with the *convert_sid* tool included with the LDAP module:

```
Base64 encoded: AQUAAAAAAAAUVAAAA3e5/WdBj9hHz1/+pVgQAAA==
Hexadecimal: 01 05 00 00 00 00 00 05 15 00 00 00 dd ee 7f 59 d0 63 f6
              11 f3 d7 ff a9 56 04 00 00
SID: S-1-5-21-1501556445-301360080-2852116467-1110
```

The *SID* value can then be used with the *pac.sid* keyword.

Other ways to get the ObjectSid are via a LDAP admin tool or Microsoft's Active Directory Management Tools. These tools can be run from a Windows 10 desktop, see Figure E and Figure E for examples.

3.6.2 Client configuration

No changes are needed to the GSSAPI client configuration:

```
logoutoutput: socks.log
# debug: 1

route {
    from: 0/0 to: 0/0 via: 10.0.0.1 port = 1080
    proxyprotocol: socks_v5
    method: gssapi
}
```

3.6.3 Server configuration – PAC group limiting

The below rule shows how one can limit the access to web sites from the clients on the 10.0.0.0/8 net to members of the SOCKS_ALLOW group, with the PAC id for the group obtained from the LDAP server, as described above.

```
logoutoutput: /var/log/sockd.log
# debug: 1

internal: eth0 port = 1080
external: eth1

socksmethod: gssapi

user.privileged      : sockd
user.notprivileged   : sockd

client pass {
    from: 10.0.0.0/8 to: 0/0
}

pass {
    from: 10.0.0.0/8 to: 0/0 port = http
    pac.sid: S-1-5-21-1501556445-301360080-2852116467-1110
}
```

4 Error and setup debugging

The LDAP module involves a diverse set of protocols and systems; SOCKS, GSSAPI, LDAP, SASL, SSL/TLS, etc. For most of these protocols there are also multiple implementations, meaning that there are many components that need to work together, giving many possible combinations and error situations.

We have tried to make Dante and the LDAP module provide debug information that makes diagnosing configuration problems easier, but there are still other sources of information that might need to be consulted to perform debugging in some cases. This section provides an overview of how to simplify debugging LDAP-related Dante configurations, along with some examples of possible errors.

4.1 Dante and system logging

There are several potential sources of helpful information that can be used when debugging problems related to GSSAPI and LDAP in Dante.

4.1.1 Dante log files

Dante attempts to provide useful information in case of session establishment failures, so the Dante logs are a good place start. The information that is available to Dante can be limited by what is provided by external APIs, but in many situations the Dante logs can provide enough information to determine the source of a problem.

The Dante *sockd.conf* server configuration file specifies how and where normal logging should be done via the *logoutput* keyword.

4.1.2 Dante debug logging

Additional Dante debug information can be enabled by adding the `debug: N` keyword to the Dante *sockd.conf* file, or starting Dante with the `-d N` option, where *N* is the verbosity level. Relevant values for *N* are 1 and 2, with 1 likely being the most useful.

When debugging problems, it can be practical to start Dante manually to make it simpler to adjust command line parameters and environment variables. This involves running Dante without the `-D` (detach) option, as is shown below, assuming Dante is installed as `/usr/sbin/sockd` and that the path to the server configuration file is `/etc/sockd.conf`:

```
/usr/sbin/sockd -dl -f /etc/sockd.conf
```

Logging will still be performed as specified in *sockd.conf*, but Dante can be terminated with `Ctrl-C` and any *stderr* output from linked libraries will be seen.

Note that Dante should be started from the user that normally starts Dante, either *root* or any user specific to Dante.

4.2 LDAP module debug logging

The LDAP module keywords *ldap.debug* and *ldap.auth.debug* enable logging of extra debug information by the Dante server. When linked with OpenLDAP, the OpenLDAP debug level will also be set to the specified value.

The special value *-l* will enable both full OpenLDAP logging and extra log output from the LDAP module, which will provide additional details on the communication between Dante and LDAP/AD servers.

Note that the OpenLDAP libraries log to *stderr*, so Dante should be started manually from the shell, as shown above, to ensure the log output can be seen.

4.2.1 Kerberos log files

For problems related to Kerberos/GSSAPI, the Kerberos log files might provide useful information not available in the Dante log file.

The location of the Kerberos log file can usually be found in the Kerberos configuration file, which will typically contain a lines like the below, that specify the path to the log file:

```
[logging]
kdc = FILE:/var/log/krb5kdc.log
```

4.2.2 Kerberos client tracing

For MIT Kerberos, additional logging can be enabled that show more details about the operations performed by the Kerberos libraries.

This is controlled via the `KRB5_TRACE` environment variable:

```
KRB5_TRACE=krb5client.log
```

If networking programs that communicate with the Dante SOCKS server using GSSAPI authentication are run with this variable set, the `krb5client.log` file should get a trace of operations performed as part of the authentication process.

Ensure that the user running the networking programs has permission to write to the trace file.

4.2.3 Kerberos server tracing

For MIT Kerberos, trace logging can also be enabled that provides additional information about the Kerberos related operations performed in the Dante server.

This requires the `KRB5_TRACE` variable to be set also for the Dante server, which can be done by starting Dante like this:

```
KRB5_TRACE=krb5server.log
/usr/sbin/sockd -dl -f /etc/sockd.conf
```

Ensure that the user running the Dante server can write to the trace file.

4.2.4 LDAP server log files

LDAP server log files might also provide useful information, such as details on what connections are received and reasons for request failures.

4.3 Failure examples

This section provides some examples of commonly observed error conditions and how they can be debugged via log files.

4.3.1 Invalid *ldap.keytab* value

The *ldap.keytab* keyword can be used to set a keytab file to be used when starting an LDAP lookup session. If the keytab file specified does not exist, *ldap.group* lookups can fail and warnings like those below might get logged by the Dante server.

```
warning: krb5_read_keytab(): error starting keytab sequence: No
such file or directory
warning: krb5_create_cache(): reading keytab /nonext into list
failed: No such file or directory
warning: krb5_create_cache(): starting keytab scan failed: No
such file or directory
warning: ldapgroupmatches(): setup of Kerberos credential cache
failed: EXAMPLE.COM, /nonext: Operation not permitted
warning: ldapgroupmatches(): cannot determine which LDAP server
to use
```

The given warnings show that Dante was unable to read the keytab file, named *nonext* in this example, due to the file not existing.

For this type of problem, the warnings logged by Dante provide sufficient information to determine the source of the problem.

4.3.2 Invalid *ldap.url* username/password

An username or password in the *ldap.url* keyword specifying invalid access credentials for the LDAP server, will result in blocked *ldap.group* lookups.

This can result in Dante log warnings like the following:

```
warning: ldapgroupmatches(): binding to LDAP server ldap://ldap
.example.com:389 with username/password failed: Invalid
credentials: no additional error
warning: ldapgroupmatches(): initialization of LDAP connection
failed
```

The first warning indicates that there is a problem with the credentials for binding to the LDAP server. The logs of the LDAP server might have additional information that might be helpful.

4.3.3 Mismatching certificate

If SSL/TLS is used to encrypt communication with the LDAP server and *ldap.certcheck* is set to enable server certificate verification, *ldap.group* lookups should fail if there is a problem with the certificate. This can result in warnings like the following:

```
warning: tool_ldap_open(): start_tls attempt failed for LDAP
ldap.example.com:389: Can't contact LDAP server: no
additional error
warning: ldapgroupmatches(): binding to LDAP server ldaps://
ldap.example.com:389 with username/password failed: Can't
contact LDAP server: error:1416F086:SSL routines:
tls_process_server_certificate:certificate verify failed (
unable to get local issuer certificate)
warning: ldapgroupmatches(): initialization of LDAP connection
failed
```

For this error, the Dante logs provide sufficient information to determine the reason for the error. In this case, the problem appears to be related to the Dante server not having the certificate information required to verify the certificate of the LDAP server available.

A Syntax for LDAP user authentication

The keywords available for *LDAP*-based authentication are listed below. These statements are generally only used as a part of Dante *socks-rules*.

Some keywords can be repeated to specify multiple values, such as multiple LDAP servers, while other keywords should only be specified once per rule. Unless explicitly mentioned, the given keywords should at most be specified once per rule.

A.1 `ldap.auto.off`

Syntax: `ldap.auto.off: <no|yes>`

Disable automatic determination of LDAP server. The default value is **no**.

A.2 `ldap.auth.basedn`

Syntax: `ldap.auth.basedn: <base dn>`

Syntax: `ldap.auth.basedn.hex: <base dn>`

Syntax: `ldap.auth.basedn.hex.all: <base dn@domain.com>`

Specify the base dn to use for searches on LDAP server. The *hex* variant expects only the base DN in hex UTF-8 encoding, while the *hex.all* variant expects both the base DN and domain name in hex UTF-8 encoding.

These statements can be repeated.

A.3 `ldap.auth.certcheck`

Syntax: `ldap.auth.certcheck: <no|yes>`

Require or disable SSL certificate check when connecting to LDAP server. The default value is **no**.

A.4 `ldap.auth.certfile`

Syntax: `ldap.auth.certfile: <filename>`

With compiled with OpenLDAP, specify the path to a CA certificate file.

A.5 `ldap.auth.certpath`

Syntax: `ldap.auth.certpath: <pathname>`

When compiled with OpenLDAP or the Sun/Mozilla LDAP SDK, specify the path to the certificate database.

A.6 ldap.auth.debug

Syntax: `ldap.auth.debug: <debug level>`

Set the debug level for LDAP authentication code. With OpenLDAP, also set the library debug level. The default is *0* (off). Set to *-1* for full debug output. The OpenLDAP binary will send debug output to stderr, so to be visible the server must be started without the `-D` option.

The OpenLDAP debug levels are defined here: <https://openldap.org/doc/admin24/runningslapd.html>.

A.7 ldap.auth.domain

Syntax: `ldap.auth.domain: <domain>`

Set the default Kerberos domain to be used for GSSAPI authentication against the LDAP server. It also determines the LDAP server as `ldap://;domain;` by resolving the domain name via DNS.

A.8 ldap.auth.filter

Syntax: `ldap.auth.filter: <filter>`

The *filter* argument is the search filter for the LDAP server. The default filter is (*samaccountname=%s*) for Active Directory and (*uid=%s*) for other LDAP servers and assumes a *rfc2307bis* schema.

A.9 ldap.auth.keytab

Syntax: `ldap.auth.keytab: <keytab>`

Set the file name of the keytab file containing the Kerberos principals for authentication to the LDAP servers. If this value is not set, the value of *gssapi.keytab* will be used. If *gssapi.keytab* is also not set, the default will be */etc/sockd.keytab*.

A.10 ldap.auth.port

Syntax: `ldap.auth.port: <port>`

Set the port number to be used when contacting the LDAP port (not LDAPS port) on the LDAP server. Used for automatic LDAP server determination if no SRV DNS records exist.

The default value is **389**.

A.11 ldap.auth.port.ssl

Syntax: `ldap.auth.port.ssl: <port>`

Set the port number to be used when contacting the LDAP SSL port on the LDAP server. Used for automatic LDAP server determination if no SRV DNS records exist.

The default value is **636**.

A.12 ldap.auth.server

Syntax: `ldap.auth.server:` `<server@domain.com>`

Set the server name of the LDAP server for domain *domain.com*. This setting avoids the automated server determination via DNS SRV or *A records*.
This statement can be repeated.

A.13 ldap.auth.ssl

Syntax: `ldap.auth.ssl:` `<no|yes>`

Require SSL/TLS for LDAP connection. The default value is *yes*.

A.14 ldap.auth.url

Syntax: `ldap.auth.url:` `<url>`

Specify LDAP server information in URL format:

`ldap(s)://<username>:<password>@<host:port>/<basedn>`

This statement can be repeated.

B Syntax for LDAP group checks

The keywords available for *LDAP*-based authentication are listed below. These statements are generally only used as a part of Dante *socks-rules*.

Some keywords can be repeated to specify multiple values, such as multiple LDAP servers, while other keywords should only be specified once per rule. Unless explicitly mentioned, the given keywords should at most be specified once per rule.

B.1 `ldap.attribute`

Syntax: `ldap.attribute:` `<attribute>`

Syntax: `ldap.attribute.hex:` `<attribute>`

Sets the attribute to use when matching the *ldap.group* value against LDAP users group membership. The module will search recursively through groups. The default value is *cn*.

The *hex* variant sets the attribute using hex UTF-8 encoding.

B.2 `ldap.attribute.ad`

Syntax: `ldap.attribute.ad:` `<attribute>`

Syntax: `ldap.attribute.ad.hex:` `<attribute>`

Sets the attribute to use when matching the *ldap.group* value against LDAP users group membership, when the LDAP server is an Active Directory server. The module will search recursively through groups. The default attribute value is *memberof*.

The *hex* variant sets the attribute using hex UTF-8.

B.3 `ldap.auto.off`

Syntax: `ldap.auto.off:` `<no|yes>`

Disable automatic determination of LDAP server. The default value is *no*, giving automatic lookup.

B.4 `ldap.basedn`

Syntax: `ldap.basedn:` `<base dn|base dn@domain.com>`

Syntax: `ldap.basedn.hex:` `<base dn>`

Syntax: `ldap.basedn.hex.all:` `<base dn@domain.com>`

The parameters are defined as follows:

base dn base DN for LDAP search for any LDAP server.

base dn@domain.com the base DN for LDAP search for LDAP server for domain *domain.com*.

The *hex* variant expects only the base DN in hex UTF-8, while the *hex.all* variant expects both the base DN and domain name in hex UTF-8.

These statements can be repeated.

B.5 ldap.certcheck

Syntax: `ldap.certcheck:` `<no|yes>`

Require or disable SSL certificate check when connecting to LDAP server. The default value is **no**.

B.6 ldap.certfile

Syntax: `ldap.certfile:` `<filename>`

With compiled with OpenLDAP, specify the path to a CA certificate file.

B.7 ldap.certpath

Syntax: `ldap.certpath:` `<pathname>`

When compiled with OpenLDAP or the Sun/Mozilla LDAP SDK, specify the path to the certificate database.

B.8 ldap.debug

Syntax: `ldap.debug:` `<debug level>`

Set the debug level for LDAP authentication code. With OpenLDAP, also set the library debug level. The default is *0* (off). Set to *-1* for full debug output. The OpenLDAP binary will send debug output to stderr, so to be visible the server must be started without the *-D* option.

The OpenLDAP debug levels are defined here: <https://openldap.org/doc/admin24/runningslapd.html>.

B.9 ldap.domain

Syntax: `ldap.domain:` `<domain>`

Set the default Kerberos domain to be used for GSSAPI authentication against the LDAP server. It also determines the LDAP server as `ldap://;domain;` by resolving the domain name via DNS.

B.10 ldap.filter

Syntax: `ldap.filter:` `<filter>`

Syntax: `ldap.filter.hex:` `<filter>`

The *filter* argument is the search filter for the LDAP server. The default filter is (*memberuid=%s*) and assumes a rfc2307bis schema.
The *hex* variant sets the filter using hex UTF-8.

B.11 ldap.filter.ad

Syntax: `ldap.filter.ad: <filter>`
Syntax: `ldap.filter.ad.hex: <filter>`

Set search filter for an Active Directory server. The default filter is (*samaccountname=%s*).
The *hex* variant sets the filter using hex UTF-8.

B.12 ldap.group

Syntax: `ldap.group: <ldap-group|ldap-group@|ldap-group@domain.com>`
Syntax: `ldap.group.hex: <ldap-group>`
Syntax: `ldap.group.hex.all: <ldap-group@domain.com>`

The parameters are defined as follows:

ldap-group name of LDAP group to be used for any user.

ldap-group@ name of LDAP group to be used for users who have a domain extension in their username (e.g., *user@domain1.com*).

ldap-group@domain.com name of LDAP group to be used only for users who have a domain extension of *domain.com* in their username.

The *hex* variant expects only the group in hex UTF-8, while the *hex.all* variant expects both the group and domain name in hex UTF-8.

These statements can be repeated.

B.13 ldap.keeprealm

Syntax: `ldap.keeprealm: <no|yes>`

Keep the realm name when comparing username with LDAP user attribute. The default value is *no*.

B.14 ldap.keytab

Syntax: `ldap.keytab: <keytab>`

Set the file name of the keytab file containing the Kerberos principals for authentication to the LDAP servers. If this value is not set, the value of *gssapi.keytab* will be used. If *gssapi.keytab* is also not set, the default will be *FILE:/etc/sockd.keytab*.

B.15 ldap.mdepth

Syntax: `ldap.mdepth: <maximal search depth>`

Set the maximal search depth of recursive group searches in Active Directory. The default value is *0*.

B.16 ldap.port

Syntax: `ldap.port: <port>`

Set the port number to be used when contacting the LDAP server. Used for automatic LDAP server determination if no SRV DNS records exist.

The default value is **389**.

B.17 ldap.port.ssl

Syntax: `ldap.port.ssl: <port>`

Set the SSL port number to be used when contacting the LDAP server. Used for automatic LDAP server determination if no SRV DNS records exist.

The default value is **636**.

B.18 ldap.server

Syntax: `ldap.server: <server@domain.com>`

Set the server name of the LDAP server for domain *domain.com*. This setting avoids the automated server determination via DNS SRV or *A records*.

This statement can be repeated.

B.19 ldap.ssl

Syntax: `ldap.ssl: <no|yes>`

Require SSL/TLS for LDAP connection. The default value is *no*.

B.20 ldap.url

Syntax: `ldap.url: <url>`

Specify LDAP server information in URL format:

`ldap(s)://<username>:<password>@<host:port>/<basedn>`

This statement can be repeated.

C Syntax for PAC group checks

The keywords available for *PAC*-based group checks are listed below. These statements are generally only used as a part of Dante *socks-rules*.

Some keywords can be repeated to specify multiple values, such as multiple groups, while other keywords should only be specified once per rule. Unless explicitly mentioned, the given keywords should at most be specified once per rule.

C.1 **pac.sid**

Syntax: `pac.sid: <group-sid>`

Syntax: `pac.sid.b64: <group-sid>`

Set the SID of an Active Directory (or Samba) *security* group. The *socks-rule* these keywords are used in will only match for users that are members of the specified groups. The *b64* variant expects the SID in base64 encoded format.

These statements can be repeated.

C.2 **pac.off**

Syntax: `pac.off: <yes|no>`

Enable or disables the caching of the objectSid for authorisation, the default is *on*.

When used with the *ldap.group* functionality, the group objectSid of an Active Directory group will automatically be cached and used for verification if GSSAPI authentication is used. This avoids additional LDAP lookups and speeds up the authorisation process.

D LDAP module related compile-time values

In addition to configuration that can be changed by changing the Dante server configuration file, the Dante server also uses various timeout values as defined at compile-time. Normally there is no need to change these, but if necessary, they can be changed by the operator before recompiling Dante.

The following values are defined in the file *include/sockd.h*, and can be changed at compile-time if so desired. All values are given in seconds:

SOCKD_LDAP_DEADTIME the time to wait before a non-responsive LDAP server should be retried.

SOCKD_LDAP_SEARCHTIME the maximum time an LDAP search can take before Dante will give up waiting for a response.

SOCKD_LDAP_TIMEOUT the maximum time a connection to a LDAP server can take to establish before Dante will give up on waiting for the connection to be established.

The following value is defined in *include/config.h*:

SOCKD_LDAPCACHE_TIMEOUT is the maximal time a LDAP group result is cached.

Should it be necessary to change any of these values, the values will need to be changed and the Dante server recompiled.

E PAC SID Example Screenshots

The screenshot shows the LDAP Admin console for the DC=samba,DC=home [192.168.1.15] server. The left pane shows the directory tree with the following structure:

- DC=samba,DC=home [192.168.1.15]
 - OU=Domain Controllers
 - OU=TestGroups
 - CN=SOCKS_ALLOW (selected)
 - CN=SQUID_ALLOW
 - OU=TestServices
 - OU=TestUsers
 - CN=Builtin
 - CN=Computers
 - CN=ForeignSecurityPrincipals
 - CN=Infrastructure
 - CN=LostAndFound
 - CN=NTDS Quotas
 - CN=Program Data
 - CN=System
 - CN=Users

The right pane displays the properties of the selected CN=SOCKS_ALLOW group:

Attribute	Value	Type	Size
objectClass	top	Text	3
objectClass	group	Text	5
cn	SOCKS_ALLOW	Text	11
instanceType	4	Text	1
whenCreated	20191226201634.0Z	Text	17
uSNCreated	3813	Text	4
name	SOCKS_ALLOW	Text	11
objectGUID	08 7A DD 13 CE 35 8D 44 88 25 BC 57 ED F6 DD DE	Binary	16
objectSid	01 05 00 00 00 00 05 15 00 00 00 DD EE 7F 59 D0 63 F6 11 F3 D7 FF A9 S6 04 00 00	Binary	28
sAMAccountName	SOCKS_ALLOW	Text	11
sAMAccountType	268435456	Text	9
groupType	-2147483646	Text	11
objectCategory	CN=Group,CN=Schema,CN=Configuration,DC=samba,DC=home	Text	52
member	CN=markus,CN=Users,DC=samba,DC=home	Text	35
whenChanged	20191227171535.0Z	Text	17
uSNChanged	3821	Text	4
distinguishedName	CN=SOCKS_ALLOW,OU=TestGroups,DC=samba,DC=home	Text	45

At the bottom, the status bar shows: Samba RW, Server: 192.168.1.15, User: CN=markus,CN=Users,DC=samba,DC=home, CN=SOCKS_ALLOW,OU=TestGroups,DC=samba,DC=home, 0 subentries.

The screenshot shows the ADSI Edit console for the Default naming context [dc=samba,dc=home]. The left pane shows the directory tree with the following structure:

- Default naming context [dc=samba,dc=home]
 - CN=Users
 - CN=Computers
 - CN=Builtin
 - OU=Domain Controllers
 - CN=ForeignSecurityPrincipals
 - CN=LostAndFound
 - CN=NTDS Quotas
 - CN=Program Data
 - CN=System
 - OU=TestServices
 - OU=TestUsers
 - OU=TestGroups
 - CN=SQUID_ALLOW
 - CN=SOCKS_ALLOW (selected)

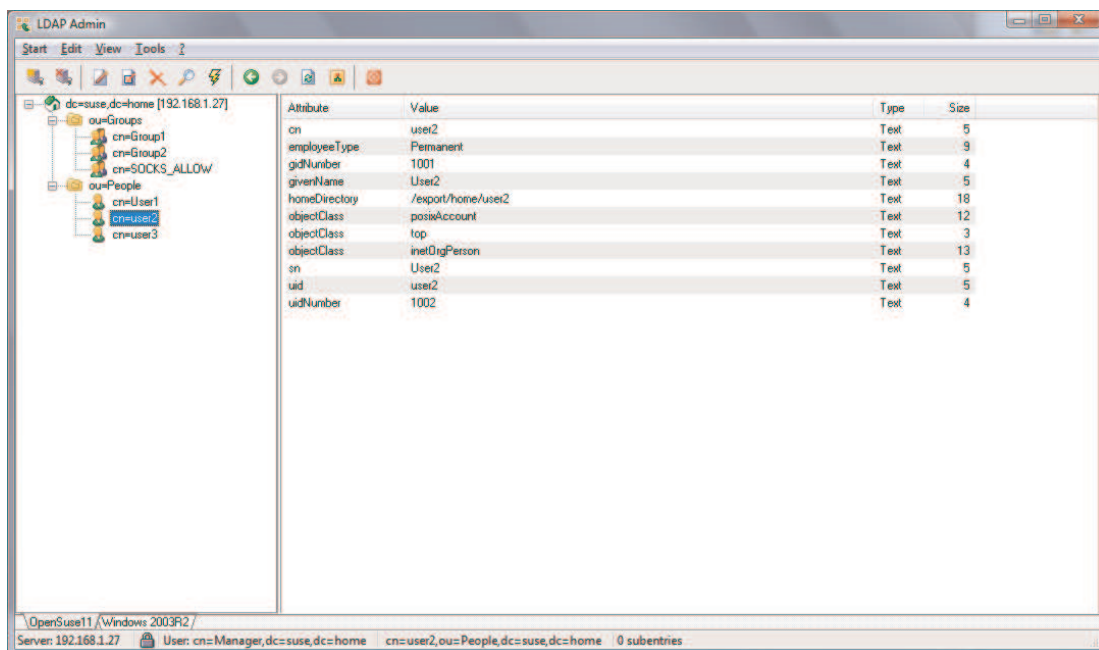
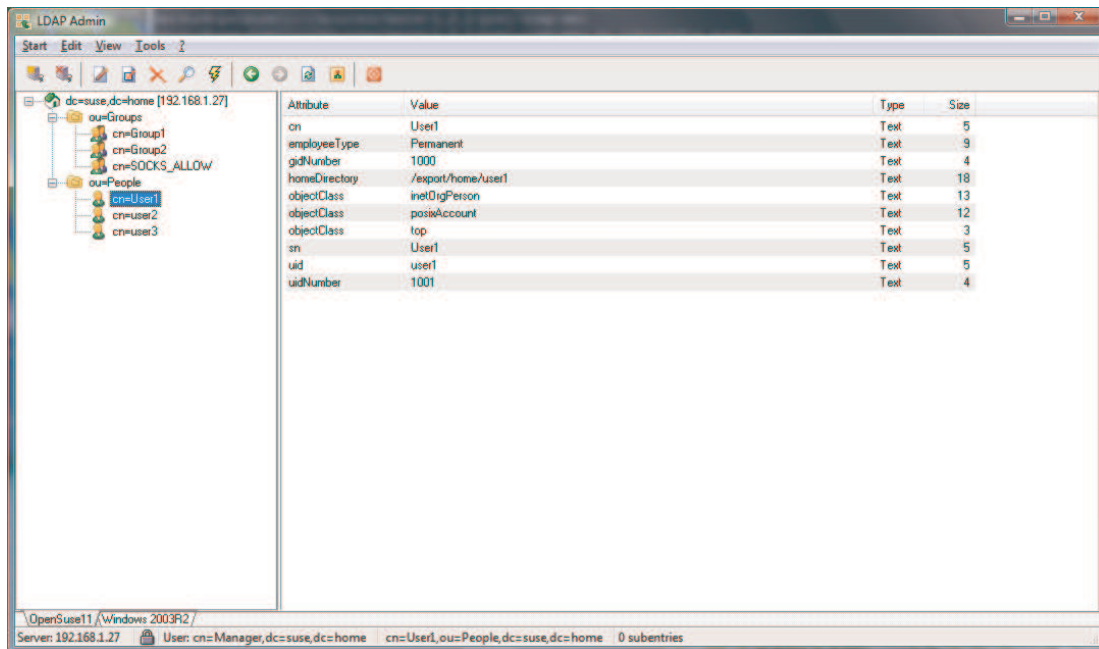
The right pane displays the properties of the selected CN=SOCKS_ALLOW group:

Attribute	Value
cn	SOCKS_ALLOW
distinguishedName	CN=SOCKS_ALLOW,OU=TestGroups,DC=samba,DC=home
groupType	0x80000002 = (ACCOUNT_GROUP SECURITY_GROUP)
instanceType	0x4 = (WRITE)
member	CN=markus,CN=Users,DC=samba,DC=home
name	SOCKS_ALLOW
objectCategory	CN=Group,CN=Schema,CN=Configuration,DC=samba,DC=home
objectClass	top; group
objectGUID	13dd7a0b-35ce-448d-8825-bc57edf6ddde
objectSid	S-1-5-21-1501556445-301360080-28521164
replPropertyMetaData	AttID Ver Loc:USN Org:DSA
sAMAccountName	SOCKS_ALLOW
sAMAccountType	268435456 = (GROUP_OBJECT)
uSNChanged	3821

The bottom of the console shows the Actions pane with the following actions:

- CN=SOCKS_ALLOW
- More Actions

F OpenLDAP Example Screenshots



LDAP Admin

Start Edit View Tools ?

dc=suse,dc=home [192.168.1.27]

- ou=Groups
 - cn=Group1
 - cn=Group2
 - cn=SOCKS_ALLOW
- ou=People
 - cn=User1
 - cn=user2
 - cn=user3

Attribute	Value	Type	Size
cn	user3	Text	5
displayName	User3	Text	5
employeeType	Temporary	Text	9
gidNumber	1002	Text	4
givenName	User3	Text	5
homeDirectory	/export/home/user3	Text	18
mail	user3@SUSE.HOME	Text	15
objectClass	posixAccount	Text	12
objectClass	top	Text	3
objectClass	inetOrgPerson	Text	13
sn	User3	Text	5
uid	user3	Text	5
uidNumber	1003	Text	4

OpenSuse11 / Windows 2003R2 /

Server: 192.168.1.27 User: cn=Manager,dc=suse,dc=home cn=user3,ou=People,dc=suse,dc=home 0 subentries

LDAP Admin

Start Edit View Tools ?

dc=suse,dc=home [192.168.1.27]

- ou=Groups
 - cn=Group1
 - cn=Group2
 - cn=SOCKS_ALLOW
- ou=People
 - cn=User1
 - cn=user2
 - cn=user3

Attribute	Value	Type	Size
cn	Group1	Text	6
gidNumber	1001	Text	4
objectClass	posixGroup	Text	10
objectClass	namedObject	Text	11
objectClass	top	Text	3

OpenSuse11 / Windows 2003R2 /

Server: 192.168.1.27 User: cn=Manager,dc=suse,dc=home cn=Group1,ou=Groups,dc=suse,dc=home 0 subentries

LDAP Admin

Start Edit View Tools ?

dc=suse,dc=home [192.168.1.27]

- ou=Groups
 - cn=Group1
 - cn=Group2
 - cn=SOCKS_ALLOW
- ou=People
 - cn=User1
 - cn=user2
 - cn=user3

Attribute	Value	Type	Size
cn	Group2	Text	6
gidNumber	1002	Text	4
objectClass	posixGroup	Text	10
objectClass	namedObject	Text	11
objectClass	top	Text	3

Server: 192.168.1.27 User: cn=Manager,dc=suse,dc=home cn=Group2,ou=Groups,dc=suse,dc=home 0 subentries

LDAP Admin

Start Edit View Tools ?

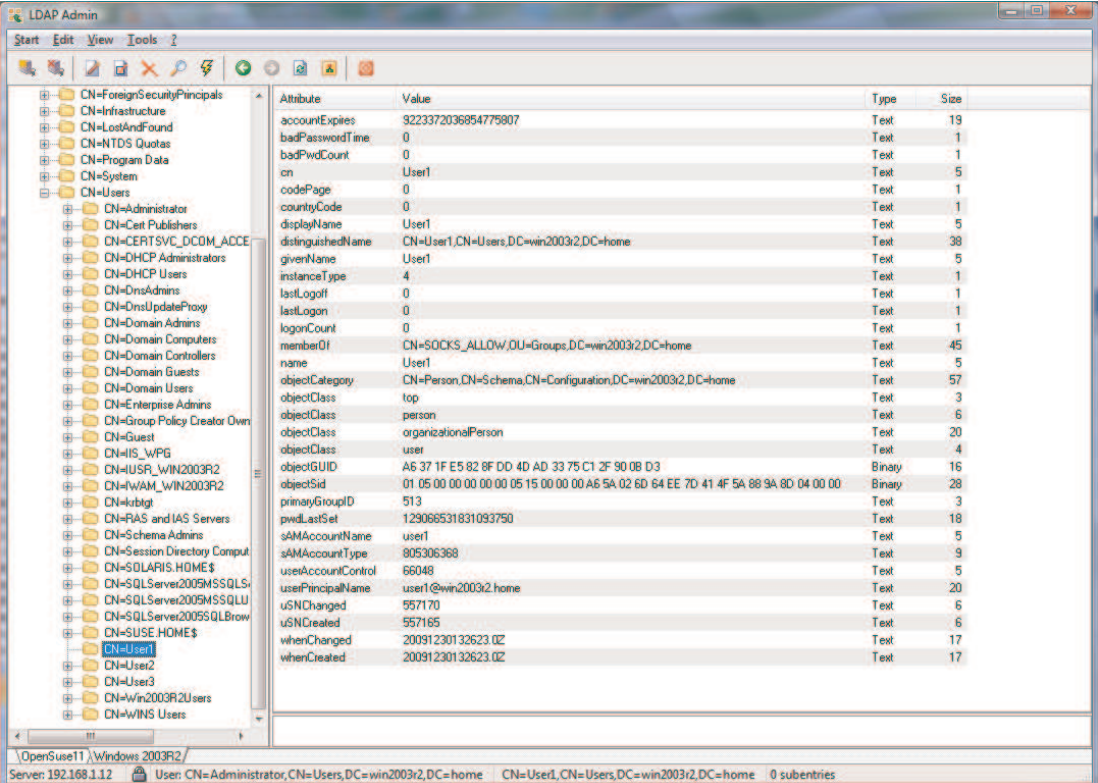
dc=suse,dc=home [192.168.1.27]

- ou=Groups
 - cn=Group1
 - cn=Group2
 - cn=SOCKS_ALLOW
- ou=People
 - cn=User1
 - cn=user2
 - cn=user3

Attribute	Value	Type	Size
cn	SOCKS_ALLOW	Text	11
gidNumber	1000	Text	4
memberUid	user2	Text	5
objectClass	posixGroup	Text	10
objectClass	namedObject	Text	11
objectClass	top	Text	3

Server: 192.168.1.27 User: cn=Manager,dc=suse,dc=home cn=SOCKS_ALLOW,ou=Groups,dc=suse,dc=home 0 subentries

G Active Directory Example Screenshots



The screenshot shows the LDAP Admin console with the following attributes and values for the 'CN=User1' object:

Attribute	Value	Type	Size
accountExpires	9223372036854775807	Text	19
badPasswordTime	0	Text	1
badPwdCount	0	Text	1
cn	User1	Text	5
codePage	0	Text	1
countryCode	0	Text	1
displayName	User1	Text	5
distinguishedName	CN=User1,CN=Users,DC=win2003r2,DC=home	Text	38
givenName	User1	Text	5
instanceType	4	Text	1
lastLogoff	0	Text	1
lastLogon	0	Text	1
logonCount	0	Text	1
memberOf	CN=SOCKS_ALLOW,OU=Groups,DC=win2003r2,DC=home	Text	45
name	User1	Text	5
objectCategory	CN=Person,CN=Schema,CN=Configuration,DC=win2003r2,DC=home	Text	57
objectClass	top	Text	3
objectClass	person	Text	6
objectClass	organizationalPerson	Text	20
objectClass	user	Text	4
objectGUID	A6 37 1F E5 82 8F DD 4D AD 33 75 C1 2F 90 08 D3	Binary	16
objectSid	01 05 00 00 00 00 05 15 00 00 00 A6 5A 02 6D 64 EE 7D 41 4F 5A 88 9A 8D 04 00 00	Binary	28
primaryGroupID	513	Text	3
pwdLastSet	129066531831093750	Text	18
sAMAccountName	user1	Text	5
sAMAccountType	805306368	Text	9
userAccountControl	66048	Text	5
userPrincipalName	user1@win2003r2.home	Text	20
uSNChanged	557170	Text	6
uSNCreated	557165	Text	6
whenChanged	20091230132623.02	Text	17
whenCreated	20091230132623.02	Text	17

The status bar at the bottom indicates the server is '192.168.1.12', the user is 'CN=Administrator,CN=Users,DC=win2003r2,DC=home', and the current object is 'CN=User1,CN=Users,DC=win2003r2,DC=home' with 0 subentries.

LDAP Admin

Start Edit View Tools ?

CN=ForeignSecurityPrincipals
 CN=Infrastructure
 CN=LostAndFound
 CN=NTDS Quotas
 CN=Program Data
 CN=System
 CN=Users

Attribute	Value	Type	Size
accountExpires	3223372036854775807	Text	19
badPasswordTime	0	Text	1
badPwdCount	0	Text	1
cn	User2	Text	5
codePage	0	Text	1
countryCode	0	Text	1
displayName	User2	Text	5
distinguishedName	CN=User2,CN=Users,DC=win2003r2,DC=home	Text	38
givenName	User2	Text	5
instanceType	4	Text	1
lastLogoff	0	Text	1
lastLogon	0	Text	1
logonCount	0	Text	1
memberOf	CN=SOCKS_GROUP1,OU=Groups,DC=win2003r2,DC=home	Text	46
name	User2	Text	5
objectCategory	CN=Person,CN=Schema,CN=Configuration,DC=win2003r2,DC=home	Text	57
objectClass	top	Text	3
objectClass	person	Text	6
objectClass	organizationalPerson	Text	20
objectClass	user	Text	4
objectGUID	2F E7 1D 81 03 A5 8A 40 A3 D8 7A 05 2D 9E CA 9A	Binary	16
objectSid	01 05 00 00 00 00 00 05 15 00 00 00 A6 5A 02 6D 64 EE 7D 41 4F 5A 88 9A 8E 04 00 00	Binary	28
primaryGroupID	513	Text	3
pwdLastSet	129086532149218750	Text	18
sAMAccountName	user2	Text	5
sAMAccountType	805306368	Text	9
userAccountControl	66048	Text	5
userPrincipalName	user2@win2003r2.home	Text	20
uSNChanged	557177	Text	6
uSNCreated	557172	Text	6
whenChanged	20091230132654.0Z	Text	17
whenCreated	20091230132654.0Z	Text	17

\OpenSuse11\Windows 2003R2/
 Server: 192.168.1.12 User: CN=Administrator,CN=Users,DC=win2003r2,DC=home CN=User2,CN=Users,DC=win2003r2,DC=home 0 subentries

LDAP Admin

Start Edit View Tools ?

CN=ForeignSecurityPrincipals
 CN=Infrastructure
 CN=LostAndFound
 CN=NTDS Quotas
 CN=Program Data
 CN=System
 CN=Users

Attribute	Value	Type	Size
accountExpires	9223372036954775807	Text	19
badPasswordTime	0	Text	1
badPwdCount	0	Text	1
cn	User3	Text	5
codePage	0	Text	1
company	MyCompany	Text	9
countryCode	0	Text	1
displayName	User3	Text	5
distinguishedName	CN=User3,CN=Users,DC=win2003r2,DC=home	Text	38
givenName	User3	Text	5
instanceType	4	Text	1
lastLogoff	0	Text	1
lastLogon	0	Text	1
logonCount	0	Text	1
name	User3	Text	5
objectCategory	CN=Person,CN=Schema,CN=Configuration,DC=win2003r2,DC=home	Text	57
objectClass	top	Text	3
objectClass	person	Text	6
objectClass	organizationalPerson	Text	20
objectClass	user	Text	4
objectGUID	FC E1 77 A3 CD A7 D0 4C 9A 35 D0 31 FB ED 18 50	Binary	16
objectSid	01 05 00 00 00 00 05 15 00 00 00 A6 5A 02 6D 64 EE 7D 41 4F 5A 88 9A 8F 04 00 00	Binary	28
primaryGroupID	513	Text	3
pwdLastSet	129066532394375000	Text	18
sAMAccountName	user3	Text	5
sAMAccountType	805306368	Text	9
userAccountControl	86048	Text	5
userPrincipalName	user3@win2003r2.home	Text	20
uSNCreated	557193	Text	6
uSNChanged	557179	Text	6
whenChanged	20091230132931.0Z	Text	17
whenCreated	20091230132719.0Z	Text	17

\OpenSuse11\Windows 2003R2/
 Server: 192.168.1.12 User: CN=Administrator,CN=Users,DC=win2003r2,DC=home CN=User3,CN=Users,DC=win2003r2,DC=home 0 subentries

LDAP Admin

Start Edit View Tools ?

dc=win2003r2,DC=home [192.168.1.12]
 OU=Domain Controllers
 OU=Groups
 CN=Group1
 CN=Group2
 CN=SOCKS_ALLOW
 CN=SOCKS_GROUP1
 OU=Ldapconfig
 OU=NetGroups
 OU=SambaServers
 OU=ServicePrincipals
 CN=Builtin
 CN=Computers
 CN=defaultMigrationContainer30
 CN=ForeignSecurityPrincipals
 CN=Infrastructure
 CN=LostAndFound
 CN=NTDS Quotas
 CN=Program Data
 CN=System
 CN=Users

Attribute	Value	Type	Size
cn	Group1	Text	6
distinguishedName	CN=Group1,OU=Groups,DC=win2003r2,DC=home	Text	40
groupType	-2147483646	Text	11
instanceType	4	Text	1
member	CN=Markus Moeller,CN=Users,DC=win2003r2,DC=home	Text	47
name	Group1	Text	6
objectCategory	CN=Group,CN=Schema,CN=Configuration,DC=win2003r2,DC=home	Text	56
objectClass	top	Text	3
objectClass	group	Text	5
objectGUID	50 26 D1 AF 0A 6E 22 4D A3 A1 39 4B 91 C4 C7 5E	Binary	16
objectSid	01 05 00 00 00 00 05 15 00 00 00 A6 5A 02 6D 64 EE 7D 41 4F 5A 88 9A 75 04 00 00	Binary	28
sAMAccountName	Group1	Text	6
sAMAccountType	268435456	Text	9
uSNCreated	156509	Text	6
uSNChanged	156469	Text	6
whenChanged	20080629114222.0Z	Text	17
whenCreated	20080629113642.0Z	Text	17

\OpenSuse11\Windows 2003R2/
 Server: 192.168.1.12 User: CN=Administrator,CN=Users,DC=win2003r2,DC=home CN=Group1,OU=Groups,DC=win2003r2,DC=home 0 subentries

LDAP Admin

Start Edit View Tools ?

dc=win2003r2,DC=home [192.168.1.12]

OU=Domain Controllers

OU=Groups

 CN=Group1

 CN=Group2

 CN=SOCKS_ALLOW

 CN=SOCKS_GROUP1

OU=Ldapconfig

OU=NetGroups

OU=SambaServers

OU=ServicePrincipals

CN=Builtin

CN=Computers

CN=defaultMigrationContainer30

CN=ForeignSecurityPrincipals

CN=Infrastructure

CN=LostAndFound

CN=NTDS Quotas

CN=Program Data

CN=System

CN=Users

Attribute	Value	Type	Size
cn	Group2	Text	6
distinguishedName	CN=Group2,OU=Groups,DC=win2003r2,DC=home	Text	40
groupType	-2147483646	Text	11
instanceType	4	Text	1
member	CN=Markus Moeller,CN=Users,DC=win2003r2,DC=home	Text	47
memberOf	CN=SOCKS_ALLOW,CN=Users,DC=win2003r2,DC=home	Text	44
name	Group2	Text	6
objectCategory	CN=Group,CN=Schema,CN=Configuration,DC=win2003r2,DC=home	Text	56
objectClass	top	Text	3
objectClass	group	Text	5
objectGUID	C2 DE 96 10 C5 A0 18 4E B1 37 28 A1 11 C9 C1 60	Binary	16
objectSid	01 05 00 00 00 00 05 15 00 00 00 A6 5A 02 6D 64 EE 7D 41 4F 5A 88 9A 76 04 00 00	Binary	28
sAMAccountName	Group2	Text	6
sAMAccountType	268435456	Text	9
uSNChanged	156512	Text	6
uSNCreated	156473	Text	6
whenChanged	20080629114222.0Z	Text	17
whenCreated	20080629113711.0Z	Text	17

Server: 192.168.1.12 User: CN=Administrator,CN=Users,DC=win2003r2,DC=home CN=Group2,OU=Groups,DC=win2003r2,DC=home 0 subentries

LDAP Admin

Start Edit View Tools ?

dc=win2003r2,DC=home [192.168.1.12]

OU=Domain Controllers

OU=Groups

 CN=Group1

 CN=Group2

 CN=SOCKS_ALLOW

 CN=SOCKS_GROUP1

OU=HomeUsers

OU=Ldapconfig

OU=NetGroups

OU=SambaServers

OU=ServicePrincipals

CN=Builtin

CN=Computers

CN=defaultMigrationContainer30

CN=ForeignSecurityPrincipals

CN=Infrastructure

CN=LostAndFound

CN=NTDS Quotas

CN=Program Data

CN=System

CN=Users

Attribute	Value	Type	Size
cn	SOCKS_ALLOW	Text	11
distinguishedName	CN=SOCKS_ALLOW,OU=Groups,DC=win2003r2,DC=home	Text	45
groupType	-2147483646	Text	11
instanceType	4	Text	1
member	CN=User1,CN=Users,DC=win2003r2,DC=home	Text	38
member	CN=SOCKS_GROUP1,OU=Groups,DC=win2003r2,DC=home	Text	46
name	SOCKS_ALLOW	Text	11
objectCategory	CN=Group,CN=Schema,CN=Configuration,DC=win2003r2,DC=home	Text	56
objectClass	top	Text	3
objectClass	group	Text	5
objectGUID	DD 38 F2 25 3D 8C 07 41 90 33 3C 24 89 3D 56 18	Binary	16
objectSid	01 05 00 00 00 00 05 15 00 00 00 A6 5A 02 6D 64 EE 7D 41 4F 5A 88 9A 88 04 00 00	Binary	28
sAMAccountName	SOCKS_ALLOW	Text	11
sAMAccountType	268435456	Text	9
uSNChanged	557266	Text	6
uSNCreated	557154	Text	6
whenChanged	20100101112251.0Z	Text	17
whenCreated	20091230132506.0Z	Text	17

Server: 192.168.1.12 User: CN=Administrator,CN=Users,DC=win2003r2,DC=home CN=SOCKS_ALLOW,OU=Groups,DC=win2003r2,DC=home 0 subentries

LDAP Admin

Start Edit View Tools ?

dc=win2003r2,DC=home [192.168.1.12]

- OU=Domain Controllers
 - OU=Groups
 - CN=Group1
 - CN=Group2
 - CN=SOCKS_ALLOW
 - CN=SOCKS_GROUP1**
 - OU=HomeUsers
 - OU=Ldapconfig
 - OU=NetGroups
 - OU=SambaServers
 - OU=ServicePrincipals
 - CN=Builtin
 - CN=Computers
 - CN=defaultMigrationContainer30
 - CN=ForeignSecurityPrincipals
 - CN=Infrastructure
 - CN=LostAndFound
 - CN=NTDS Quotas
 - CN=Program Data
 - CN=System
 - CN=Users

Attribute	Value	Type	Size
cn	SOCKS_GROUP1	Text	12
distinguishedName	CN=SOCKS_GROUP1,OU=Groups,DC=win2003r2,DC=home	Text	46
groupType	-2147483646	Text	11
instanceType	4	Text	1
member	CN=User2,CN=Users,DC=win2003r2,DC=home	Text	38
memberOf	CN=SOCKS_ALLOW,OU=Groups,DC=win2003r2,DC=home	Text	45
name	SOCKS_GROUP1	Text	12
objectCategory	CN=Group,CN=Schema,CN=Configuration,DC=win2003r2,DC=home	Text	56
objectClass	top	Text	3
objectClass	group	Text	5
objectGUID	D2 52 50 DD 9A 2F ED 4F 9A CE AF AE 1D 25 C9 C8	Binary	16
objectSid	01 05 00 00 00 00 05 15 00 00 00 A6 5A 02 6D 64 EE 7D 41 4F 5A 88 9A 8A 04 00 00	Binary	28
sAMAccountName	SOCKS_GROUP1	Text	12
sAMAccountType	268435456	Text	9
uSNCreated	957263	Text	6
uSNChanged	957150	Text	6
whenChanged	20100101122514.0Z	Text	17
whenCreated	20091230132451.0Z	Text	17

Windows 2003R2

Server: 192.168.1.12 User: CN=Administrator,CN=Users,DC=win2003r2,DC=home CN=SOCKS_GROUP1,OU=Groups,DC=win2003r2,DC=home 0 subentries