DAT151
Database and Unix System Management

Spring 2024

Assignment 5 - Unix

Obligatory assignment. Deadline: Sunday 24.03.2024.

The report should include all necessary commands to complete the tasks, printout from the system, explanation of what is done, the result and explanation of the result.

Remember to include the names of the group members on the front page of the report. The report can be in English or Norwegian. The report should be handed in via Canvas.

The assignment should be accomplished in groups of one to three students. Signing up for a group will be closed on Thursday 21.03.

You must select a group "Lab5 N" when delivering to see all the comments that you will get on this assignment.

***Observe**: Task two, three and four will require a minimum of two lab computers, and a minimum group size of two is necessary.*

***Observe**: Before you modify an existing configuration file, save a copy of the original somewhere on the computer, e.g. below /root/origs.*

Lecturers:
Bjarte Kileng <Bjarte.Kileng@hvl.no>, room E418
Haakon André Reme-Ness <Haakon.Andre.Reme-Ness@hvl.no>, Fabrikkgaten 5, R235-02

Last changed, 21.03.2024 (BK)

Task 1: Logging

a) Configure systemd journal at the lab to save the logs persistent in the file system.

b) Protect the journal logs from unnoticed alteration by enabling the Seal feature and create a sealing key.

c) The syslog facility names "**local0**" to "**local7**" are used for local custom messages defined by an administrator. Create your own configuration that saves only the critical level and information level messages from the **local5** facility to a file *local5* in the */var/log* directory. Test your configuration using the **logger** command with specific messages.

Look under the */var/log* directory that the messages you added are saved in the correct location.

Task 2: LDAP

You will need a minimum of two lab computers for this task!

Background

In this task you will set up an LDAP server and client(s). Use one of the computers of the group as the LDAP server, and the others as clients.

An LDAP database can store any kind of organized records in an hierarchical structure. In this task you will set up and LDAP database to store user records that later will be used for login on a Linux computer, the client computer.

Document your configurations clearly, include your LDIF files in your report, and explain how user records were added to your server.

You can organize your work such that each group sets up one dedicated server machine, or you can use one machine acting as both the server and client. The alternatives:

1. One group set up an OpenLDAP server. The other group uses this server to accept logins to their own lab computer.
   - This should be done in both directions, i.e. both groups will set up and OpenLDAP server for the other group.
2. The group set up and OpenLDAP server that is used for login to the same computer.

In this first task you will set up the server. Later in this assignment you will deal with the login.

Installation and configuration

You will need to install the packages *openldap-clients* and *openldap-servers*. Then start and enable the *slapd* service.

The OpenLDAP web page tells you how to configure and set up OpenLDAP.

The server configuration is stored in LDIF files below */etc/openldap/slapd.d*. These files creates a LDAP configuration database for the server. Do not modify theses files by hand! They contain a file checksum, and a mismatch will cause the server to fail[1].

The DN suffix

OpenLDAP will route queries to one of its databases. Each database has a base DN suffix that routes queries to the correct database. The suffix of the query must match that of the database.

The OpenLDAP server package comes with the configuration for a single basic LDAP database. You will use this database as the starting point for the database of user records.

To see the base DN suffix of the preinstalled database, use the command below:

1 Inconsistent configuration values can prevent the server to start. Such a situation can require that you reinstall the server, or modify configuration files by hand. The tool **crc32** from the package *perl-Archive-Zip* will let you recalculate the file checksum after a modification.

```
sudo ldapsearch -LLL -Q -Y EXTERNAL -H ldapi:/// -o ldif-wrap=no \
    -b "olcDatabase={2}mdb,cn=config" olcSuffix
```

The pre installed database probably uses the value *dc=my-domain,dc=com*.

The switch “-Y EXTERNAL” with “ldapi://” corresponds to a access method that we have already met for MariaDB. It allows the Linux root user to login as administrator.

The database will have an administrator that can manage the database. The administrator credentials consists of a DN and a password. To see the DN of the administrator, use the command below:

```
sudo ldapsearch -LLL -Q -Y EXTERNAL -H ldapi:/// -o ldif-wrap=no \
    -b "olcDatabase={2}mdb,cn=config" olcRootDN
```

The preinstalled database probably uses the value *cn=Manager,dc=my-domain,dc=com*.

You should change theses values to something that is unique for you computer. It is very important though the the administrator DN is hierarchical below the base DN, i.e. that the administrator DN ends with the base DN. A mismatch here will make the OpenLDAP server fail! You must update the base DN and the administrator DN in the same transaction!

The base DN should be unique within the group of computers that collaborate. With <*some_number*> the last part of the computer IP, you can e.g. choose the base DN as:

```
dc=h<some_number>,dc=dat151
```

I will use 255 for <*some_number*> in this document, but you should use your own value.

The administrator DN would then be:

```
cn=Manager,dc=h255,dc=dat151
```

To configure the database with the new DNs, create an LDIF file, e.g. *basedn.ldif*, with the following content:

```
dn: olcDatabase={2}mdb,cn=config
changetype: modify
replace: olcSuffix
olcSuffix: dc=h255,dc=dat151
-
replace: olcRootDN
olcRootDN: cn=Manager,dc=h255,dc=dat151
```

The above value of **dn** references the configuration of the preinstalled database. The above **dn** value contains *mdb*, the name of the backend that is used internally by OpenLDAP to store the LDAP records.

You must use LDAP tools to commit the changes, e.g. **ldapmodify**:

```
sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f ~bki/ldap/basedn.ldif
```

A mismatch between *olcSuffix* and *olcRootDN* will cause the server to fail. Therefore, check that you can restart the **slapd** service, and then the status of the restarted service.

OpenLDAP manager password

The Linux root account has access to the configuration database, but there is yet no access for anybody to to create or modify entries of other databases, e.g. user records. This access can be given to Linux root account, but a more common practise is to give the LDAP administrator this authorization.

A password hash for the manager must stored in the configuration database, and the manager must be given authorization to modify the database of user records.

To create the password hash, use the command **slappasswd** command.

To configuration the database with the manager authorization and password, create a file, e.g. *manager.ldif* with the following content:

```
dn: olcDatabase={2}mdb,cn=config
changetype: modify
add: olcRootPW
olcRootPW: <password hash>
-
add: olcAccess
olcAccess: to * by dn="<administrator DN>" write
               by self write
               by * read
```

Replace <password hash> with the output from the **slappasswd** command, and <administrator DN> with its correct value, see above.

The above configuration gives the administrator write access to all LDAP entries. Also the owner of an entry (i.e. *self*) has write access. Everybody else has read access. This authorization is rather wide though and not suitable for a production environment.

Then update the configuration database with the new data:

```
sudo ldapmodify -Y EXTERNAL -H ldapi:/// -f manager.ldif
```

You can check that the result with the below command:

```
sudo ldapsearch -LLL -Q -Y EXTERNAL -H ldapi:/// -o ldif-wrap=no \
    -b "olcDatabase={2}mdb,cn=config"
```

If you would like to give access to the Linux root user, see e.g. the *olcAccess* parameter of “olcDatabase={0}config,cn=config”.

Schemas to store user records

OpenLDAP comes with a collection of LDAP schemas. A schema corresponds to a database layout, or table layout in e.g. MariaDB. A schema specifies what kind of records that can be stored in the LDAP database, data types and constraints. Multiple schemas can be applied on the same database, corresponding to multiple tables and data types in a MariaDB database.

The OpenLDAP server collection of schemas are found in the directory */etc/openldap/schema/*.

For more information on LDAP schemas, see e.g. Understanding LDAP Schema.

The command below will list the schemas of the preintalled LDAP database:

```
sudo ldapsearch -LLL -Y EXTERNAL -H ldapi:/// -b "cn=schema,cn=config" cn
```

The preinstalled database only has the *core* schema. You will find that the returned value is given as "{0}core". The number "{0}" is not part of the name. Such numbers are used by OpenLDAP to enforce a consistent ordering in the configuration database, since LDAP databases are inherently unordered.

A LDAP database of user records will need further schemas:

- NIS: Unix account attributes, required for a database of user accounts.
    - User name, UID, home directory, groups, etc
- COSINE: We need only one attribute from this schema, *manager*. The others we do not need, e.g. *buildingName*, *drink*, *secretary* etc.
- INETORGPERSON: Contains some additional fields that can be useful.

You will need the NIS schema for the user database, but the NIS schema depends on the COSINE schema. You must therefore first install the COSINE schema, then the NIS schema.

```
sudo ldapadd -Y EXTERNAL -H ldapi:// -f /etc/openldap/schema/cosine.ldif
```

Now, install the NIS schema:

```
sudo ldapadd -Y EXTERNAL -H ldapi:// -f /etc/openldap/schema/nis.ldif
```

Check that the schemas was successfully installed, and list the attributes:

```
sudo ldapsearch -LLL -Y EXTERNAL -H ldapi:/// \
    -b "cn={2}nis,cn=schema,cn=config"
```

Create the user record database

You are now in the position to create the database for user records. This must be done as the OpenLDAP manager, unless you authorized the Linux root user this access.

First, create the a LDIF file, e.g. *top.ldif* for the database DN:

```
dn: dc=h255,dc=dat151
dc: h255
objectClass: top
objectClass: domain
```

Now, create the database:

```
sudo ldapadd -D "cn=Manager,dc=h255,dc=dat151" -x -W -f ~/ldap/top.ldif
```

You will be prompted for the manager password that you created above. Check then that the database was added:

```
ldapsearch -LLL -b "dc=h255,dc=dat151" -x dn
```

For the LDAP tools, default values can be put in the file */etc/openldap/ldap.conf*, e.g. add:

```
BASE dc=h255,dc=dat151
URI  ldap://127.0.0.1
```

With the above configuration, the above query can be simplified as:

```
ldapsearch -LLL -x dn
```

Fill the database

With the LDAP database up and running, you can start to fill entries in the database. An LDAP database for user logins on a Linux computer will need DNs for users and groups:

- *dn: ou=Group,dc=h255,dc=dat151*
- *dn: ou=People,dc=h255,dc=dat151*

Verify that you can access the LDAP server from the client(s).

The lab computer with IP 10.0.0.250 is set up to authenticate user logins. You can check how users and groups are stored at that computer. To list the users, run the command below at you lab computer:

```
ldapsearch -LLL -H ldap://10.0.0.250 -x \
    -b "ou=People,dc=dat351,dc=hvl,dc=no"
```

Fill the database with users and groups, at least one user and one group.

LDAP parameters for user entries are described e.g. in the manual page **sssd-ldap-attributes(5)**.

The LDAP attribute for a user password is *userPassword*. As for the manager password, use the command **slappasswd** command to create a password hash, to be used with *userPassword*., see e.g. OpenLDAP Software 2.6 Administrator's Guide: Security Considerations.

Observe, for security reasons, LDAP queries do not return password attributes, unless the query is performed by the manager.

Task 3: SSSD

You will need a minimum of two lab computers for this task!

The SSSD server configuration file is */etc/sssd/sssd.conf*. If the file does not exist you must create it. The file must be owned by root, and must be accessible by root only.

Configure SSSD to use the LDAP from task 2 for user authentication. SSSD require a signed certificate for communication with the LDAP server. Initially, you can skip certificate specifics when testing the setup. To do this, add the line below to the configuration file */etc/sssd/sssd.conf*, in section *[domain/LDAP]*:

```
ldap_auth_disable_tls_never_use_in_production = true
```

The above configuration is meant for debugging purposes only, and strongly discouraged, and not documented in the manual pages. To get this assignment approved though, you must eventually add a signed certificate to authenticate the LDAP server. This is explained at the end of this task.

The configuration parameter *enumerate* can be useful. Check the manual page of **sssd.conf(5)**.

The command **sss_cache** can be useful when working with SSSD. The command let you clear the SSSD cache.

Use **authselect** and select the PAM profile for SSSD.

Add the feature *with-mkhomedir* to the **authselect** profile for SSSD. Then PAM will create the home directory if it is missing. This feature requires that the systemd service *oddjobd.service* is running. The service file is found in the RPM package *oddjob-mkhomedir*.

When everything is set up correctly, try to login using the user that was created in task 2.

The client computer should be a different computer than the LDAP host, and the user should not exist on the client prior to log in. You must must a add screen shot of a working log in on the client:

```
su - <some_user> # To be run on the client, NOT the LDAP host
```

Replace <*some_user*> with a user from the LDAP.

By default on EL9 (see */usr/lib/systemd/system/slapd.service*), LDAP can communicate on the following channels:

- **ldap**: Clear text LDAP, by default on port 389,
- **ldaps**: LDAP over SSL and TLS, by default on port 636.
- **ldapi**: LDAP over IPC, using s Unix domain socket.

With *ldaps* the communication is encrypted and only *ldaps* should be used with SSSD. This communication channel requires a private key and a corresponding certificate signed by a CA, and the CA must be approved by the SSSD server.

With SSSD authentication working against the LDAP server on the clear text *ldap* channel, configure the LDAP server to use the encrypted *ldaps* channel.

For *ldaps*, both the LDAP server and client must have sensible host names known by the DNS, or found in the the */etc/hosts* file. Use the command *hostnamectl* on both the LDAP server and client to give both computers sensible host names, and then make entries in the */etc/hosts* file for both computers on both computers.

The LDAP certificate and corresponding key must exist on the LDAP server only. To make a certificate, you must first create a certificate request and then get the request signed by a certificate authority (CA). The certificate subject must have a CN that matches the host name of the LDAP server, i.e.:

- *CN=LDAP-server-name*

The server name must match the value used with the SSD parameter *ldap_uri*. Use a fully qualified host name.

The certificate and key file must be owned by the **ladp** user and **ldap** group, and the key file should be accessible only by the **ldap** user. Remember also to open the firewall on the LDAP server for port *ldaps*.

The LDAP certificate and key file must be added to the LDAP server configuration, to the top entry of the LDAP configuration tree, i.e. "*dn=config*". The configuration attributes are:

- *olcTLSCertificateFile* : Certificate file path, e.g. */etc/openldap/certs/ldap.cert.pem*
- *olcTLSCertificateKeyFile*: key file path, e.g. */etc/openldap/certs/ldap.key.pem*

You can create your own CA, or you can ask the lecturer to sign the certificate for you. The CA certificate used by the lecturer is found here:

- https://eple.hvl.no/certs/dat151.ldapcacert.pem

The certificate of the CA (not the LDAP certificate) must be known by the SSSD daemon on the LDAP client. First, remove the parameter *ldap_auth_disable_tls_never_use_in_production*, then add the line below to the configuration file /etc/sssd/sssd.conf, in section *[domain/LDAP]*:

```
ldap_tls_cacert = <path to the CA certificate>
```

If you would like to create your own CA (not required), you can use e.g. these guides:

- Simple steps to configure LDAPS with TLS certificates CentOS 7 Linux (also EL9)
- How to setup your own CA with OpenSSL

Task 4: Kerberos

You will need a minimum of two lab computers for this task!

Kerberos is an authentication mechanism. Once authenticated to the Kerberos server, a client is issued a token. This token can be used to authenticate the client to Kerberized Services such as SSH.

In this task you will have to configure a host as a KDC (*Key Distribution Center*) and also use it as a Kerberos client to authenticate SSH logins.

Remember to create the KDC database with a secure password. You have to enable and start both the *kadmin* and *krb5kdc* services so that your Kerberos KDC services will be automatically available after system reboot.

A Kerberos realm is the logical network served by a single Kerberos database, e.g. **DRIFTSLAB.HVL.NO**.

A principal is a unique identity to which Kerberos can assign tickets, i.e. users, computers and services. Principals typically have the form "*primary/instance@REALM*".

The *primary* component of a user principal will be the user name. The *instance* component of a user principal has to do with privileges. For non-administrators, this component is missing. A present *instance* is usually used to extend the privileges beyond that of normal users. Examples for user principals:

- Principal for a regular user: *student@DRIFTSLAB.HVL.NO*
- Principal for an administrator: *student/admin@DRIFTSLAB.HVL.NO*

Observe that the principal "*student@DRIFTSLAB.HVL.NO*" is completely separate from the principal "*student/admin@DRIFTSLAB.HVL.NO*".

The *primary* component of a computer will be the word "**host**" (i.e. not the host name of the computer, but the word "**host**"). The *instance* component will be the host name. Example of a host principal:

- **host/h255.driftslab@DRIFTSLAB.HVL.NO**

The *primary* component of a network service will be the name of the service, and the *instance* component will be the host name. Example of a principal for an LDAP service:

- **ldap/h255.driftslab@DRIFTSLAB.HVL.NO**

Observe that the meaning of the *primary* and *instance* components of the principal is convention. There is nothing in Kerberos that dictates the meaning or use of these components.

The tool **kadmin.local** directly modifies the Kerberos database, whereas the corresponding tool **kadmin** connects with a Kerberos service to modify the database. Since the Kerberos database is only accessible by the Linux *root* user, the tool **kadmin.local** must be run as *root*.

By default, both these tool uses a principal "**$USER/admin**" in the default realm where "**$USER**" is the Linux user that is running the command. Check the manual **kadmin(1)** for how to run these commands using a different principal.

Since only *root* can run **kadmin.local**, and the default principal of root is "**root/admin**", create an administrator principal "**root/admin**".

The **kadmin** tool can be run by normal Linux user, but if not *root* you must modify the file */etc/krb5.conf* and change the logging options.

For Kerberos authentication in SSH, both the Linux user, the SSH server and the KDC server must have valid tickets, i.e. there must be a principal for all these entities

If the SSH server also is the KDC server, Kerberos authentication in SSH requires these two principals:

- A *user principal* for the Linux user that use SSH with Kerberos for login.
- A *host prinicpal* for the SSH/KDC server.

When everything is set up correctly, verify that the server will authenticate SSH logins.

Document your setup and explain each step in your configurations and make sure firewall rules are set correctly.

You should use sensible and fully qualified host names, both for the server and the client. The host name can e.g. be something like "*h<some_number>.driftslab*" where *<some_number>* is the last part of the computer IP number. Add the server and client to the */etc/hosts* file, both on server and client.

For testing, you could use a match block in the configuration of the ssh server to require the client to use Kerberos for access. You could e.g. create a file */etc/ssh/sshd_config.d/60-kerberos.conf* that includes the following code:

```
Match Address IP.number.of.client
      AuthenticationMethods gssapi-with-mic
```

If necessary, set the parameter *GSSAPIDelegateCredentials* on the client if you need the ticket to be forwarded, e.g. through eple.