PSAwise2324Team10Aufgabe08

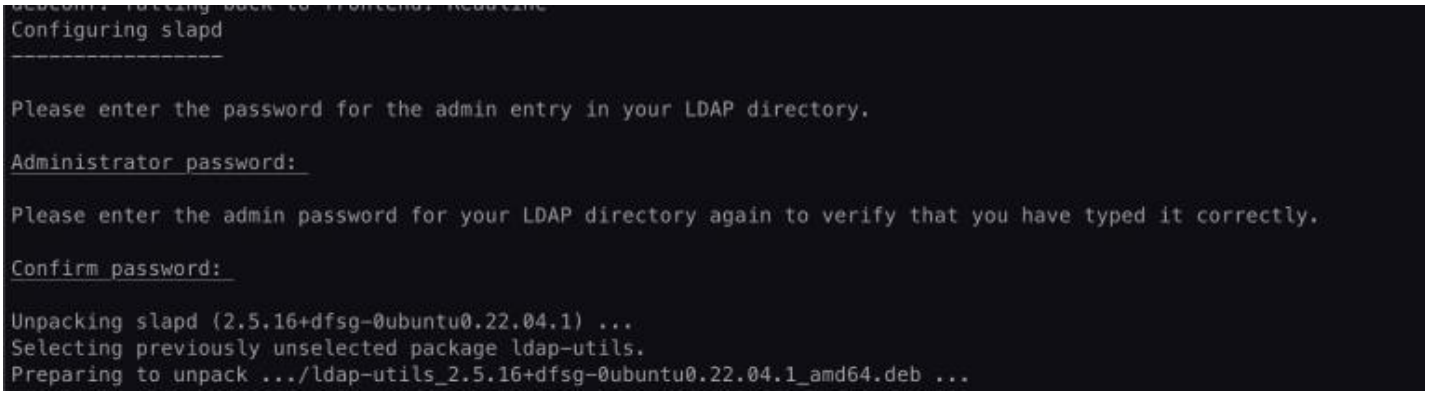
LDAP

Installation

We decided to create a new VM7 and set up the LDAP server there. Start by installing the required packages.

```
sudo apt install slapd ldap-utils
```

While configuring slapd, you need to enter a password for the admin entry.

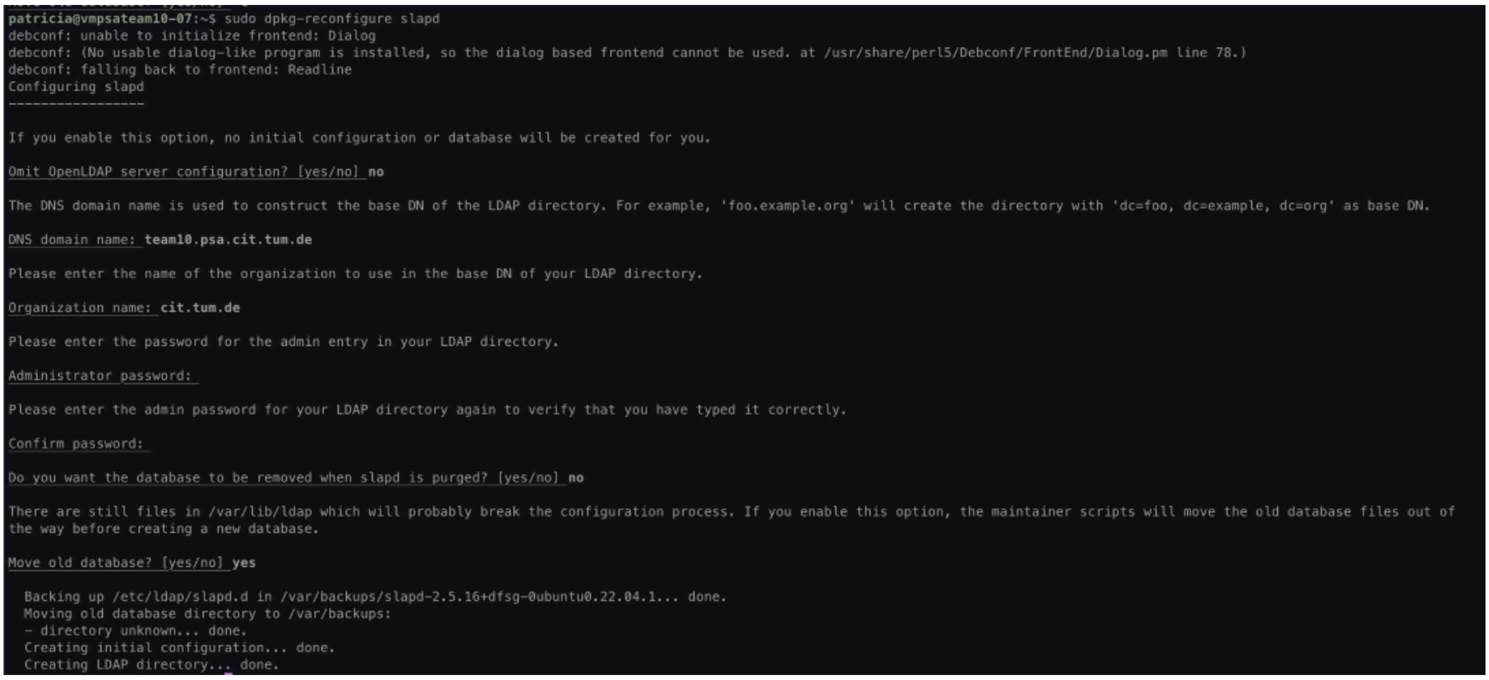


```
Configuring slapd
-----------------

Please enter the password for the admin entry in your LDAP directory.

Administrator password:

Please enter the admin password for your LDAP directory again to verify that you have typed it correctly.

Confirm password:

Unpacking slapd (2.5.16+dfsg-0ubuntu0.22.04.1) ...
Selecting previously unselected package ldap-utils.
Preparing to unpack .../ldap-utils_2.5.16+dfsg-0ubuntu0.22.04.1_amd64.deb ...
```

To continue with the configuration, run a new command afterwards:

```
sudo dpkg-reconfigure slapd
```

- Omit OpenLDAP server configuration: no
- DNS domain name: team10.psa.cit.tum.de
- Organization name: cit.tum.de
- Remove DB when slapd is purged: no
- Move old DB: yes



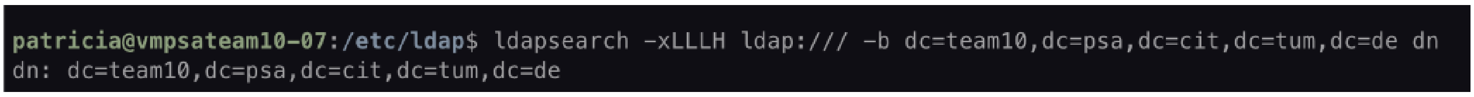
```
patricia@vmpsateam10-07:~$ sudo dpkg-reconfigure slapd
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78.)
debconf: falling back to frontend: Readline
Configuring slapd
-----------------

If you enable this option, no initial configuration or database will be created for you.

Omit OpenLDAP server configuration? [yes/no] no

The DNS domain name is used to construct the base DN of the LDAP directory. For example, 'foo.example.org' will create the directory with 'dc=foo, dc=example, dc=org' as base DN.

DNS domain name: team10.psa.cit.tum.de

Please enter the name of the organization to use in the base DN of your LDAP directory.

Organization name: cit.tum.de

Please enter the password for the admin entry in your LDAP directory.

Administrator password:

Please enter the admin password for your LDAP directory again to verify that you have typed it correctly.

Confirm password:

Do you want the database to be removed when slapd is purged? [yes/no] no

There are still files in /var/lib/ldap which will probably break the configuration process. If you enable this option, the maintainer scripts will move the old database files out of
the way before creating a new database.

Move old database? [yes/no] yes

  Backing up /etc/ldap/slapd.d in /var/backups/slapd-2.5.16+dfsg-0ubuntu0.22.04.1... done.
  Moving old database directory to /var/backups:
  - directory unknown... done.
  Creating initial configuration... done.
  Creating LDAP directory... done.
```

Afterwards, you need to edit the the LDAP config file `/etc/ldap/ldap.conf` to make sure it includes these lines:

```
BASE dc=team10,dc=psa,dc=cit,dc=tum,dc=de
URI ldaps://vmpsateam10-07.psa-team10.cit.tum.de:636
```

After saving the changes, restart slapd.

You can test if our LDAP works with this command: `ldapsearch -xLLLH ldap:/// -b dc=team10,dc=psa,dc=cit,dc=tum,dc=de dn`.
The answer should look like this:



```
patricia@vmpsateam10-07:/etc/ldap$ ldapsearch -xLLLH ldap:/// -b dc=team10,dc=psa,dc=cit,dc=tum,dc=de dn
dn: dc=team10,dc=psa,dc=cit,dc=tum,dc=de
```

To make the server accessible from the outside, add rules for port 636 (LDAPS) to the firewall.

TLS Configuration

We want communication with our LDAP server to be encrypted, so we use TLS. Therefore, the server requires a certificate that is trusted by the client. Since we don't have a CA that will sign certificates for us, we act as our own CA and use self-signed certificates.

First, install these tools:

```
sudo apt install gnutls-bin ssl-cert
```

Then, generate a private key for the CA:

```
sudo certtool --generate-privkey --bits 4096 --outfile
/etc/ssl/private/CA_key.pem
```

Create a file in /etc/ssl called ca.info. The purpose of this file is to provide basic information about our CA, so give it the following contents:

```
cn = ca.team10.psa.cit.tum.de
ca
cert_signing_key
expiration_days = 3650
```

Using the private key and the info file, you can now generate a self signed certificate for the CA:

```
sudo certtool --generate-self-signed \
--load-privkey /etc/ssl/private/CA_key.pem \
--template /etc/ssl/ca.info \
--outfile /usr/local/share/ca-certificates/CA_cert.crt
```

Run `sudo update-ca-certificates` to add this certificate to the list of trusted CAs on this machine. This also creates a symlink to the certificate, called CA_cert.pem, to /ect/ssl/certs. With the CA established, we move onto the server. Same as with the CA, you need to create a private key, info file and certificate.

Generate a private key:

```
sudo certtool --generate-privkey \
--bits 2048 \
--outfile /etc/ldap/server_key.pem
```

And change the permissions and group of the key:

```
sudo chgrp openldap /etc/ldap/server_key.pem
sudo chmod 0640 /etc/ldap/server_key.pem
```

Create a server.info file in /etc/ssl with these contents:

```
organization = Example Company
cn = vmpsateam10-07.psa-team10.cit.tum.de //!! actual resolvable domain name
of this machine
tls_www_server
encryption_key
signing_key
expiration_days = 365
```

And finally, generate the certificate, which is signed by our CA:

```
sudo certtool --generate-certificate \
--load-privkey /etc/ldap/server_key.pem \
--load-ca-certificate /etc/ssl/certs/CA_cert.pem \
--load-ca-privkey /etc/ssl/private/CA_key.pem \
--template /etc/ssl/server.info \
--outfile /etc/ldap/server_cert.pem
```

All that's left on the server side is to inform OpenLDAP of these files and activate TLS. To do this, create a tls.ldif file and specify the changes as follows:

```
dn: cn=config
add: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/ssl/certs/CA_cert.pem
-
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ldap/server_cert.pem
-
add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ldap/server_key.pem
```

Then apply the ldif file with this command:

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

You may also need to uncomment or add the line `SLAPD_SERVICES="ldap:/// ldapi:/// ldaps:///"` to /etc/default/slapd.
Restart slapd with `sudo systemctl restart slapd` and TLS should work on this machine.
On client machines, we will need to add the certificate of our CA to make them trust the server. More on that in the next section.
To test the connection on this machine, run

```
ldapsearch -xLLLH ldaps://192.168.10.7:636 -D
"cn=admin,dc=team10,dc=psa,dc=cit,dc=tum,dc=de" dn -W
```

Client LDAP Configuration

First, add rules for port 636 to the firewall (input tcp sport, output tcp dport), as we will be using it for LDAPS.

Next, run `sudo apt install ldap-utils libnss-ldap` to get all the LDAP functionality that the client needs. During installation, you will be made to select a couple configuration details for ldap-auth-config, which is part of these packages. We picked these options:

```
should debconf manage configuration? yes
enter URI: ldaps://vmpsateam10-07.psa-team10.cit.tum.de:636
Distinguished name of the LDAP search base:
dc=team10,dc=psa,dc=cit,dc=tum,dc=de
LDAP version to use: 3
Make local root Database admin: No
Does the LDAP database require login? No
Local crypt to use when changing passwords: None
```

If you want to change the configuration again, simply run `sudo dpkg-reconfigure ldap-auth-config`.
Our system doesn't trust self signed certificates, like the one we are using, so we need to manually add our server's CA certificate to the trusted certificates of this machine. This certificate will later be used to determine that the server certificate, which was signed by the CA, is trustworthy.

To copy the certificate safely, we ran this scp command from the LDAP server:

```
sudo scp -P <client port> /usr/local/share/ca-certificates/CA_cert.crt
<user>@psa.in.tum.de:/home/<user>
```

We couldn't copy directly into /usr/local/share/ca-certificates, which is where this file is supposed to end up, so we copied it to the home directory of the current user and moved it to its correct location.

Then, you have to run `sudo update-ca-certificates` and the certificate will be added to the trusted list.
In order for OpenLDAP to find this certificate, we also need to specify them in /etc/ldap.conf:

```
tls_cacertfile /etc/ssl/certs/CA_Cert.pem
tls_cacertdir /etc/ssl/certs
```

To test the connection, you can run `ldapsearch -xLLLH ldaps://vmpsateam10-07.psa-team10.cit.tum.de:636 -D "cn=admin,dc=team10,dc=psa,dc=cit,dc=tum,dc=de" -b "dc=team10,dc=psa,dc=cit,dc=tum,dc=de" -W dn` . It should output a list of DNs that currently exist in the searchbase indicated by -b.
Notably, this only works with this URI. ldaps://192.168.10.7:636 does not work, because the hostname in the URI has to match the hostname specified in the server certificate.

Generating Users

PSA Users

We need a new organizational unit to store our users in, so we create the LDIF ((LDAP Data Interchange Format) file user_group.ldif in /etc/ldap and specify ou=users in it

```
dn: ou=users,dc=team10,dc=psa,dc=cit,dc=tum,dc=de
objectClass: organizationalUnit
ou: users
```

Then we apply the LDIF by running `sudo ldapadd -x -H ldaps:/// -f user_group.ldif  -D "cn=admin,dc=team10,dc=psa,dc=cit,dc=tum,dc=de" -W`.
First, we want to add the users that already have a login on our server. For this purpose, we created a CSV file that contains the username, UID, last name, first name and email address of all PSA users.

```
username,uid_number,last_name,first_name,email
dietr,1401,Dietrich,Thomas,ge65wug@psa.in.tum.de
schoe,1101,Schönberger,Josef,ge93kiv@psa.in.tum.de
steph,1102,Stephan,Alexander,ge35fel@psa.in.tum.de
<...>
```

We wrote a a bash script that parses the CSV file and creates a entry for every users in the newly created LDIF (LDAP Data Interchange Format) file users.ldif in `/etc/ldap`. The bash script is located at `/usr/bin/add_users2ldap.sh`.
Of note: The users all belong to the group posixAccount, which allows LDAP to match them to the existing user accounts in our system. This way, the users can keep their home directories etc. All passwords are salted and hashed using slappasswd.

```
#!/bin/bash

ldif_file="/etc/ldap/users.ldif"

# make sure users.ldif exists
if [ -e $ldif_file ]; then
    echo "File exists. We can add the human users to it."
    # clears file; convenient for debugging and fine in terms of efficiency bc
we only ever add all users at once
    # if he had a large userbase and only wanted to add 1 user it would
usually be overkill
    echo -n > $ldif_file
else
    echo "File does not exist. We need to create it first."
    touch $ldif_file
fi

users_file="/home/patricia/user_list.csv"
base_dn="ou=users,dc=team10,dc=psa,dc=cit,dc=tum,dc=de" # => add users to
ou=users
# iterate over each entry in the txt file and create a new entry in users.ldif
tail -n +2 "$users_file" | while IFS=, read -r username uid_number last_name
first_name email; do
        pw=$(slappasswd -s psa -h {SSHA}) # salt and hash password
        cat <<EOF >> "$ldif_file"
dn: uid=$username,$base_dn
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
uid: $username
uidNumber: $uid_number
gidNumber: $uid_number
cn: $first_name $last_name
sn: $last_name
givenName: $first_name
mail: $email
homeDirectory: /home/$username
userPassword: $pw

EOF
done

echo "LDIF entries created and saved to '$ldif_file'."
```

After successfully running this script, we add the users to the LDAP system by running:

```
sudo ldapadd -x -D cn=admin,dc=team10,dc=psa,dc=cit,dc=tum,dc=de  -W -f
users.ldif
```

On our VMs, we use not only the PSA user accounts but also personal accounts, which are named patricia or ida depending on the VM. We also added these users to LDAP the same way as the PSA users, but their user entries, which have the same format as the others, are in a separate LDIF file /etc/ldap/team10_users.ldif , since we are more likely to change their data at some point and this might be useful. They also have a different password than the rest.

Users from Long List

The last user group that needed to be added were the fictional users from the CSV file. For this purpose, we used the same basic process, but had to also make a new object class and write a new, more complex bash script. The UIDs and numerical UIDs also needed to be generated dynamically.

The object class was needed because the users had attributes that none of the inbuilt classes offered. To create it, navigate to the folder /etc/ldap/schema. Create two files, extendeduser.schema and extendeduser.ldif. extendeduser.schema contains a definition of the new object class and its attributes, as seen below:

```
attributetype: ( 1.3.6.1.4.1.995.2.1.1 NAME 'geschlecht'
 EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )

# <do the same thing for every other attribute>

objectclass: ( 2.99.99.6 NAME 'extendeduser'
 SUP inetOrgPerson STRUCTURAL MAY ( telefon $ geschlecht $ geburtstag $
geburtsort $ nationalitaet $ stadt $ matrnr ) )
```

We decided to add the attributes for phone number, gender, birth date, birthplace, nationality, city and matriculation number. Other attributes, like the postal code, are already provided by the objectClass organizationalPerson that we use for our users.

Note that the name as well as the OID of each entry, which appears at the start of each definition, has to be unique. Also of note is that our extendeduser class inherits from inetOrgPerson (`SUP inetOrgPerson`), so it will have every attribute of that class in addition to its own.
extendeduser.ldif contains the same entries as extendeduser.schema, but in a slightly different format and with a changetype preamble:

```
dn: cn=extendeduser,cn=schema,cn=config
objectClass: olcSchemaConfig
cn: extendeduser
olcAttributeTypes: ( 1.3.6.1.4.1.995.1.2.1 NAME 'geschlecht'
 EQUALITY caseIgnoreMatch SUBSTR caseIgnoreSubstringsMatch SYNTAX
1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE )

# <do the same for every other attribute>

olcObjectClasses: ( 2.99.99.6 NAME 'extendeduser'
 SUP inetOrgPerson STRUCTURAL MAY ( telefon $ geschlecht $ geburtstag $
geburtsort $ nationalitaet $ stadt $ matrnr ) )
```

Apply the LDIF file using `sudo ldapadd -x -D "cn=admin,cn=config" -W -H ldaps://vmpsateam10-07.psa-team10.cit.tum.de:636 -f extendeduser.ldif` and a schema configuration for your new object class and attributes should be created. Make sure to check if cn=admin,cn=config has a password (should be listed in /etc/ldap/slapd.d/'olcDatabase={0}config.ldif') before executing this command. If it does not, add one by applying an LDIF file such as the one below:

```
dn: olcDatabase={0}config,cn=config
changetype: modify
add: olcRootPW
olcRootPW: <put password here>
```

After creating the object class, you can move onto generating the entries. First, you have to change the character set used by the CSV file. As `file -i /home/patricia/benutzerdaten.csv` reveals, it is currently ISO-8859-1, not UTF-8, which will cause issues. To change it to UTF-8, run `sudo iconv -f ISO-8859-1 -t UTF-8 benutzerdaten.csv > benutzerdaten_utf8.csv`.
Now you can generate the user entries using the script below. This script can be found in /usr/bin/add_long_csv_users2ldap.sh.

```
#!/bin/bash

# <create or clear dest file depending on whether it already exists (same as
first script)>

# some functions to retrieve IDs that are already in use
get_used_uids () {
    sudo ldapsearch -x -LLL uid=* | grep uid: | cut -d: -f2
}

get_used_uid_numbers () {
        sudo ldapsearch -x -LLL uid=* | grep uidNumber: | cut -d: -f2
}

get_used_gid_numbers () {
        sudo ldapsearch -x -LLL uid=* | grep gidNumber: | cut -d: -f2
}


gen_uid () {
        local first="$1"
        local last="$2"
        local used_uids="$3"

        user_id="${last:0:5}" # bash implementation should make sure this
stays withing buffer, so this is ok

        length_last=${#last}
        if [ $length_last -lt 5 ]; then
                length_remaining=$((5 - $length_last))
                user_id+="${first:0:$length_remaining}"
        fi

        # convert to lowercase
        user_id=$(echo "$user_id" | tr '[:upper:]' '[:lower:]')


        # make sure it doesn't already exist. If it does, add lowest number >
1 available
        # this is done via substring comparison. Every uid has same length and
there's whitespace in between,
        # so there should be no accidental match
        og_id=$user_id
        counter=2
        while [[ $used_uids == *"$user_id"* ]]; do
                user_id="${og_id}${counter}"
                ((counter++))
        done

        echo $user_id
}

gen_uid_number () {
        used_uid_nums="$1"
        used_gid_nums="$2"
        # bc the numbers are separated by newlines and in ascending order
        highest_uid=$(echo -e "$used_uid_nums" | tail -n 1)
        highest_gid=$(echo -e "$used_gid_nums" | tail -n 1)

        if [ $highest_uid -gt $highest_gid ]; then
                ((highest_uid++))
                echo $highest_uid
        else
                ((highest_gid++))
               echo $highest_gid
        fi
}

users_file="/home/patricia/benutzerdaten_utf8.csv"
base_dn="ou=users,dc=team10,dc=psa,dc=cit,dc=tum,dc=de"
used_uids=$(get_used_uids)
used_uid_numbers=$(get_used_uid_numbers)
used_gid_numbers=$(get_used_gid_numbers)
# iterate over each entry in the txt file and create a new entry in users.ldif
tail -n +2 "$users_file" | while IFS=, read -r last_name first_name gender
birthdate birthplace nationality street zip city phone matnr; do
        pw=$(slappasswd -s psa -h {SSHA})
        uid=$(gen_uid "$first_name" "$last_name" "$used_uids")
        used_uids+="\n${uid}"
        uid_number=$(gen_uid_number "$used_uid_numbers" "$used_gid_numbers")
        used_uid_numbers+="\n${uid_number}"
        used_gid_numbers+="\n${uid_number}"
        echo $uid
        echo $uid_number
        echo ""
        cat <<EOF >> "$ldif_file"
dn: uid=$uid,$base_dn
```

```
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: extendeduser
objectClass: posixAccount
uid: $uid
uidNumber: $uid_number
gidNumber: $uid_number
cn: $first_name $last_name
sn: $last_name
givenName: $first_name
userPassword: $pw
geschlecht: $gender
geburtstag: $birthdate
geburtsort: $birthplace
nationalitaet: $nationality
street: $street
postalCode: $zip
stadt: $city
telefon: $phone
matrnr: $matnr
homeDirectory: /home/$uid

EOF
done

echo "LDIF entries created and saved to '$ldif_file'."
```

If everything runs without error, you can now add them the same way as the other users.

User certificates

Each user is supposed to have their own certificate. We stored the certificates and private keys in /etc/ssl/temp_certs, /etc/ssl/temp_priv and the public keys in our LDAP directory. To store them in LDAP, we used the userCertificate attribute of the object class inetOrgPerson. This attribute stores binary data, so we need to adjust the format of the public key before storing it.

The first step is to generate certificate and the set of keys for each user. We created a script that does this and generates an entry in an LDIF file for the public keys at the same time.

```
#!/bin/bash

search_output=$(sudo ldapsearch -xLLLH ldaps://vmpsateam10-07.psa-
team10.cit.tum.de:636 -D \
 "uid=kastl,ou=users,dc=team10,dc=psa,dc=cit,dc=tum,dc=de" -b \
 "ou=users,dc=team10,dc=psa,dc=cit,dc=tum,dc=de" -w psa dn)


users=($(echo "$search_output" | grep -E "^dn:" | awk '{print $2}'))

ldif_file="/etc/ldap/user_certs.ldif"
priv_path="/etc/ssl/temp_priv"
pub_path="/etc/ssl/temp_pub"
```

```
certs_path="/etc/ssl/temp_certs"

#priv_path="/home/patricia"
#pub_path="/home/patricia"
#certs_path="/home/patricia"

# make sure users.ldif exists
if [ -e $ldif_file ]; then
    echo "File exists. We can add the human users to it."
    # delete contents
    echo -n > $ldif_file
else
    echo "File does not exist. We need to create it first."
    touch $ldif_file
fi


keys() {
        prompt_answers="DE\nBY\nMunich\nPSA\nteam10\n$1\n\n\n"
        echo -e $prompt_answers | sudo openssl req -newkey rsa:2048 -nodes -
days 365000 \
        -keyout $priv_path/$1_priv.pem \
        -out $pub_path/$1_req.pem > /dev/null 2>&1
        echo "Generated keys and csr for $1"
}



cert() {
        sudo openssl x509 -req -days 365000 -set_serial 01 \
        -in $pub_path/$1_req.pem \
        -out $certs_path/$1_cert.pem \
        -CA /etc/ssl/certs/v4_CA_cert.pem \
        -CAkey /etc/ssl/private/v4_CA_key.pem > /dev/null 2>&1
        echo "Generated cert for $1"
}


pub_key_ldap() {
        # public key is part of csr (req), we need to extract it
        req_path=$pub_path/$1_req.pem
        pub_key_definitive_path=$pub_path/$1_pub.pem
        openssl req -in $req_path -noout -pubkey -out $pub_key_definitive_path
        # convert to binary

        #remove BEGIN and END section (idk dude)
#       tail -n +2 $pub_key_definitive_path | head -n -1 >
$pub_path/$1_pub_content_only.pem
        openssl pkey -in $pub_key_definitive_path -pubin -outform der -out
$pub_path/$1_pub.der
#       base64 -w 0 $pub_path/$1_pub_content_only.pem >
$pub_path/${dn}_pub.txt
}


for dn in "${users[@]}"; do
```

```
        keys "$dn"
        cert "$dn"
        pub_key_ldap "$dn"
        #pub_key=$(cat $pub_path/${dn}_pub.der)
#       pub_key=$(cat $pub_path/${dn}_pub.txt)
#       pub_key=$(cat  $pub_path/${dn}_pub_content_only.pem)
        cat <<EOF >> "$ldif_file"
dn: $dn
changetype: modify
add: userCertificate
userCertificate;binary:< file://$pub_path/${dn}_pub.der

EOF
done
```

After running this script, you can apply the LDIF file by running `sudo ldapmodify -x -D "cn=admin,dc=team10,dc=psa,dc=cit,dc=tum,dc=de" -W -f user_certs.ldif`. Now every user should have their certificate.

Centralized Authentication

To make a machine use LDAP for authentication, we need to edit `/etc/nsswitch.conf`. Replace "files systemd" with "files ldap" for the following entries:

```
passwd:         ldap files
group:          ldap files
shadow:         ldap files
gshadow:        ldap files
```

This tells the system to try looking at ldap for information on these things first. We left the "files" option in as well, however, as a backup.

Next, we need to remove use_authok from /etc/pam.d/common-password

```
password        [success=1 user_unknown=ignore default=die]     pam_ldap.so
try_first_pass      #used to be in this line
```

Passwd

To change the functionality of passwd, we renamed the original definition (/usr/bin/passwd) to backup_passwd and replaced it with a symlink of the same name, which points to our passwd version, custom_passwd_ldap.

Create the file custom_passwd_ldap in /usr/bin and run `sudo chmod 4755 /usr/bin/custom_passwd_ldap`.

The owner and group of the file should be root:root. Enter these contents:

```
#!/bin/bash

uid=$(whoami) # get username of person who executed this command

ldappasswd  -xH ldaps://vmpsateam10-07.psa-team10.cit.tum.de:636 -D
"uid=$uid,ou=users,dc=team10,dc=psa,dc=cit,dc=tum,dc=de" -W -S
```

-W will prompt the user for their current password, -S will prompt them to enter a new one. We already set the URI and domain to increase convenience for the user. We don't pass additional options along to this function, because we feel that would only create confusion, as it's not clear to the user that they would be passing the options to ldappasswd.

To create the symlink, run

```
sudo ln -sf /usr/bin/custom_passwd_ldap /usr/bin/passwd
```

Now every user should be able to use passwd to change their LDAP password.

Restart the machine after setting up once you are finished setting up the centralized authentication.

Anonymous Bind

Anonymous bind (access by unauthenticated users) can be configured through access rules. We want to configure it for our PSA directory, so the correct place to make these changes is /etc/ldap/slapd.d/'cn=config'/'olcDatabase={1}mdb.ldif' . (The other two database configs are frontend and config, which we don't want to change)

To start with, this file contained the following access rules:

```
# allows authenticated users to change their own password, unauthenticated
(anonymous) users to do
authentication based on the stored passwords and rebuffs every
other access attempt (by * none)
olcAccess: {0}to attrs=userPassword by self write by anonymous auth by * none

# allows authenticated user to change time between January 1, 1970 and last
password change
olcAccess: {1}to attrs=shadowLastChange by self write by * read

# allows everyone to read all attributes
olcAccess: {2}to * by * read
```

The last rule especially might seem odd at first glance, since it means "everyone can read everything" but it is important to note that the access rules are evaluated in order of descending indices. The indices are the numbers in curly brackets (e.g. {0}). If a rule with a lower index forbids access, higher index rules won't even be evaluated. Thus, the current settings actually only allow access for authenticated users, while unauthenticated users can't do anything besides try to get authenticated. As far as we understood, the assignment asks for anonymous users to be able to authenticate themselves (already given by current rules) as well as list the existing user IDs. For the latter part, a new rule has to be added. Create an LDIF file titled ACL_anonymous_bind.ldif and give it these contents:

```
dn: olcDatabase={1}mdb,cn=config
changetype: modify
delete: olcAccess
-
add: olcAccess
olcAccess: {0}to attrs=userPassword by self write by anonymous auth by * none
-
add: olcAccess
olcAccess: {1}to dn.subtree="ou=users,dc=team10,dc=psa,dc=cit,dc=tum,dc=de"
attrs=cn,uid by * read
-
add: olcAccess
olcAccess: {2}to attrs=shadowLastChange by self write by * read
-
add: olcAccess
olcAccess: {3}to * by * read
```

Then apply the changes using `sudo ldapadd -x -D "cn=admin,cn=config" -W -H ldaps://vmpsateam10-07.psa-team10.cit.tum.de:636 -f ACL_anonymous_bind.ldif`. The new rule, {1}, allows everyone (authenticated as well as anonymous users) to read the attributes CN and UID of our users.

Testing

Located on VM7.

```
#!/bin/bash

failed_tests=0

fail() {
    ((failed_tests++))
    echo "FAIL $@"
}

ok() {
    echo "OK $@"
}


# executes ldapsearch with given credentials
# arg 1 = name, arg 2  = descriptor
# password = psa is fixed bc we're only using that one
ldapsearch_as_user() {

    output=$(ldapsearch -xLLLH ldaps://vmpsateam10-07.psa-
team10.cit.tum.de:636 -D "$1" dn -w psa)


    if [ $? -eq 0 ]; then
        ok "LDAP search successfull for $2 ($1)"
    else
        fail "LDAP search failed for $2 ($1)"
    fi

     echo -e "\tsample output: $output" | head -n 1

}


ldapsearch_as_user "uid=kastl,ou=users,dc=team10,dc=psa,dc=cit,dc=tum,dc=de"
"PSA user"
ldapsearch_as_user "uid=schmo,ou=users,dc=team10,dc=psa,dc=cit,dc=tum,dc=de"
"long list user"

entry_count=$(ldapsearch -xLLLH ldaps://vmpsateam10-07.psa-
team10.cit.tum.de:636 -D
"uid=kastl,ou=users,dc=team10,dc=psa,dc=cit,dc=tum,dc=de" \
 -b "ou=users,dc=team10,dc=psa,dc=cit,dc=tum,dc=de" dn -w psa | grep -c "dn:")

if [ $entry_count -eq 107 ]; then
        ok "The amount of entries is correct"
else
        fail "The amount of entries is wrong"
fi


echo "Failed tests: $failed_tests"
```