SSSD and OpenLDAP

This page will describe how we have to setup **SSSD** and an **OpenLDAP** server to manage users authentication one various machines, when all the user's information are stored in the remote **OpenLDAP** server. **SSSD** is a package build on top of the various services like **PAM**, **NSS**, **SSH, etc**.

Steps

1. Install **SSSD** if it's not already present
2. Configure **SSSD**
3. Configure the **LDAP** server

SSSD

SSSD is a system daemon that "*provides access to identity and authentication remote resource through a common framework that can provide caching and offline support to the system*" (https://fedorahosted.org/sssd/). It offers the following features:

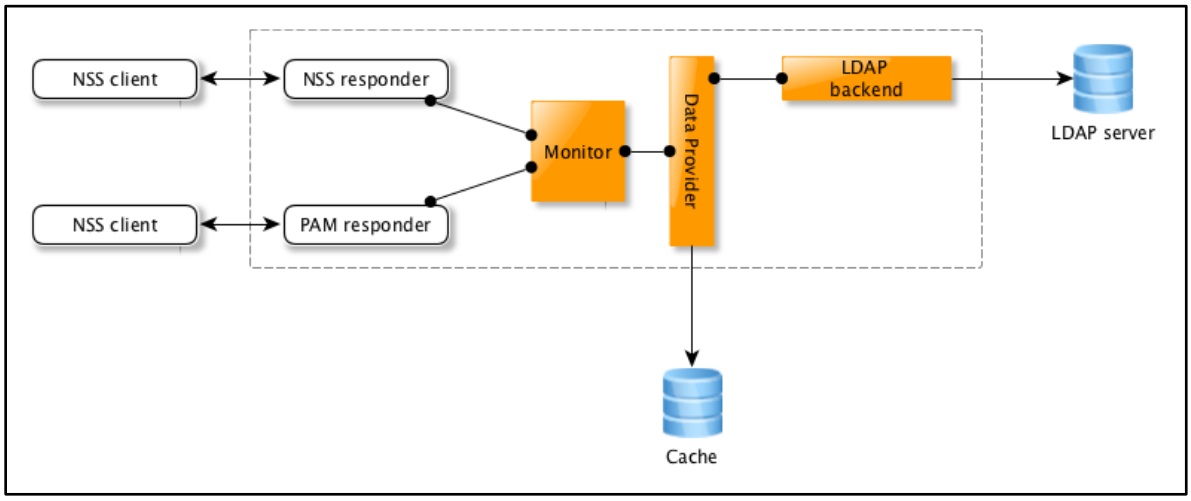
- Offline authentication: if your machine can't connect to the remote **LDAP** server, you still can log on it
- Server load reduction: it opens one single connection to the **LDAP** server
- Multiple domain support: you can have more than one remote source of identity
- And many other minor features.

For us, the main point is that **SSSD** becomes the single point of configuration, when you had many without it.

How it works

SSSD is a service that manage the access to the remote data and cache them locally. The existing services that are used by applications will now send their request to **SSSD** instead of requesting them from the remote database themselves.

Here is the **SSSD** architecture:



The important thing is that **SSSD** plays nice with the existing pieces since it is not replacing anything. It simply intercepts the component's requests and returns the expected result to the requesters. SSSD offers one single point of configuration making it easier for system administrators to handle the authentication system configuration.

Installation

```
sudo yum install sssd
```

```
sudo apt-get install sssd libpam-sss libnss-sss
```

Basic commands

Before exposing the configuration part, let's see how we manage the **SSSD** service. It's a daemon, so it has to be restarted after some modifications have been done in its configuration. Stopping, starting and restarting the **SSSD** daemon is done this way:

```
sudo service sssd
[stop|start|restart|status|reload|force-reload|condrestart|try-restart]
```

Note: that I have listed all the possible commands, but you'll probably want to use only the first four.

```
sudo /etc/init.d/sssd
[stop|start|restart|status|reload|force-reload|condrestart|try-restart]
```

That was for the service itself. Now, starting the service might lock you out, if you have the wrong configuration. There are three things you absolutely need to do when playing with **SSSD**:

1. **ALWAYS** have another session opened beside. You may have to use it to revert the configuration
2. **ALWAYS** test your configuration. You have a script called **authconfig** that can be used for that
3. **ALWAYS** save the previous configuration. Same thing, you have the **authconfig** script for that.

The authconfig tool

There is a python script that is quite useful when it comes to manage the configuration: **autconfig**. It should be installed with **SSSD**.

Testing the configuration

```
sudo authconfig --test
```

That will dump the full configuration for all the services you have set.

Here is an example :

```
[elecharny@livarot ~]$ sudo authconfig --test
caching is disabled
nss_files is always enabled
nss_compat is disabled
nss_db is disabled
nss_hesiod is disabled
hesiod LHS = ""
hesiod RHS = ""
nss_ldap is enabled
SSSD commands
SSSD Command Debian
authconfig test
SSSD config sample
LDAP+TLS is enabled
LDAP server = "ldap://brie.rb.symas.net"
LDAP base DN = "dc=symas,dc=com"
nss_nis is disabled
NIS server = ""
NIS domain = ""
nss_nisplus is disabled
nss_winbind is disabled
SMB workgroup = "MYGROUP"
SMB servers = ""
SMB security = "user"
SMB realm = ""
Winbind template shell = "/bin/false"
SMB idmap range = "16777216-33554431"
nss_sss is enabled by default
nss_wins is disabled
nss_mdns4_minimal is disabled
DNS preference over NSS or WINS is disabled
pam_unix is always enabled
shadow passwords are enabled
password hashing algorithm is sha512
pam_krb5 is disabled
krb5 realm = "EXAMPLE.COM"
krb5 realm via dns is disabled
krb5 kdc = "kerberos.example.com"
krb5 kdc via dns is disabled
krb5 admin server = "kerberos.example.com"
pam_ldap is enabled
LDAP+TLS is enabled
LDAP server = "ldap://brie.rb.symas.net"
LDAP base DN = "dc=symas,dc=com"
LDAP schema = "rfc2307bis"
pam_pkcs11 is disabled
use only smartcard for login is disabled
smartcard module = ""
smartcard removal action = ""
pam_fprintd is disabled
```

```
pam_winbind is disabled
SMB workgroup = "MYGROUP"
SMB servers = ""
SMB security = "user"
SMB realm = ""
pam_sss is enabled by default
credential caching in SSSD is enabled
SSSD use instead of legacy services if possible is enabled
IPAv2 is disabled
IPAv2 domain was not joined
IPAv2 server = ""
IPAv2 realm = ""
IPAv2 domain = ""
pam_cracklib is enabled (try_first_pass retry=3 type=)
pam_passwdqc is disabled ()
pam_access is disabled ()
pam_mkhomedir or pam_oddjob_mkhomedir is enabled (umask=0077)
Always authorize local users is enabled ()
Authenticate system accounts against network services is disabled
[elecharny@livarot ~]$
```

The results are quite verbose, but complete.

Creating a configuration backup

```
sudo authconfig --savebackup /tmp/sssd.save
```

This will dump all the configuration files into a directory (**sssd.save**).

Here is an example:

```
[elecharny@livarot ~]$ ll /tmp/sssd.conf/
total 96
drwxrwxrwt. 5 root root 4096 Dec 3 06:55 ..
-rw-r--r--. 1 root root 585 Dec 3 06:55 yp.conf
-rw-r--r--. 1 root root 410 Dec 3 06:55 openldap.conf
-rw-r--r--. 1 root root 449 Dec 3 06:55 krb5.conf
-rw-r--r--. 1 root root 9778 Dec 3 06:55 smb.conf
-rw-r--r--. 1 root root 1724 Dec 3 06:55 nsswitch.conf
-rw-r--r--. 1 root root 1 Dec 3 06:55 cacheenabled.conf
-rw-r--r--. 1 root root 1177 Dec 3 06:55 system-auth-ac
-rw-r--r--. 1 root root 1177 Dec 3 06:55 password-auth-ac
-rw-r--r--. 1 root root 839 Dec 3 06:55 fingerprint-auth-ac
-rw-r--r--. 1 root root 891 Dec 3 06:55 smartcard-auth-ac
-rw-r--r--. 1 root root 402 Dec 3 06:55 authconfig
-rw-r--r--. 1 root root 49 Dec 3 06:55 network
-rw-r--r--. 1 root root 2293 Dec 3 06:55 libuser.conf
-rw-r--r--. 1 root root 1816 Dec 3 06:55 login.defs
-rw-------. 1 root root 4381 Dec 3 06:55 sssd.conf
----------. 1 root root 1071 Dec 3 06:55 shadow
-rw-r--r--. 1 root root 1487 Dec 3 06:55 passwd
----------. 1 root root 572 Dec 3 06:55 gshadow
-rw-r--r--. 1 root root 701 Dec 3 06:55 group
```

As you can see, many files will be impacted by a change done.

Restoring the backup

```
sudo authconfig --restorebackup /tmp/sssd.save
```

This will restore the various files that have been previously save with the **--savebackup** command.

It's also possible to revert the last change with the **--restorelastbackup** command, as **authconfig** do save the current configuration before applying some change. Although keep in mind that will not come back any farther than the last applied modification.

Applying the configuration change

```
sudo authconfig --update
```

This will write the various files that have been configured with the tool.

Configuring SSSD

ldap.conf

This file has to be configured separately, as we are using the one we have installed. It is located in **/opt/symas/etc/openldap/**.
Here is a typical content, where we are listing two **LDAP** servers (if one is not reachable, the other one will be used).

```
#
# LDAP Defaults
#
# See ldap.conf(5) for details
# This file should be world readable but not world writable.
#SIZELIMIT 12
#TIMELIMIT 15
#DEREF never
# The CA certificate that was used to sign the server certificates
TLS_CACERT /opt/symas/ssl/cacert.pem
# The URI of the two LDAP servers we will connect to. Note that
there is
# no LDAPS URI, startTLS is mandatory.
URI ldap://brie.rb.symas.net/, ldap://cantal.rb.symas.net/
# The base root for the users and groups entries.
BASE dc=symas,dc=com
```

The data we are interested in are stored in the **dc=symas,dc=com** branch (typically, groups are stored in **ou=groups,dc=symas,dc=com** and users in **ou=people,dc=symas,dc=com**).

The remote server **CA** certificate is also stored in the **/opt/symas** directory.

sssd.conf

This is the main configuration file. It's located in **/etc/sssd/**. It covers all the services that needed to be configured separately before **SSSD** was created : **NSS**, **PAM**, **SSH**, **SUDO** in our case.

Its structure is quite simple:

```
[sssd]
<parameters>
[<service>]*
<service parameters>
[<domain type>/<domain name>]*
```

```
<domain parameters>
```

We may have many services and many domains. We will now cover all the sections one by one.

[sssd] section

This contains global parameters for **SSSD**. This is where you list the services and domains that will be used.

Here is an example :

```
#-------------------------------------------
# The SSSD configuration
#-------------------------------------------
[sssd]
config_file_version = 2
#   Trace   Fatal(0x0010),   Critical(0x0020),   Serious(0x0040),
Minor(0x0080), Config(0x0100) and Operation(0x0400) messages
# Can also be a number between 0 and 9
# check /var/log/sssd/sssd.log
debug_level=7
# the supported services
services = nss, pam, sudo,ssh
# The associated domains (only one here)
domains = rb.symas.net
```

Here, we have defined one single domain, which name is **rb.symas.net**, and 4 services. We will then have 5 more sections present in the configuration file.

The debug level can either be a number between 1 and 9, or a combination of flags (each flag represents a single bit in a 2 bytes integer, they are combined by OR-ing them). Here are the different possible values:

0x0001:	Unused
0x0002:	Unused
0x0004:	Unused
0x0008:	Unused
0x0010 / 0:	Fatal failures. Anything that would prevent SSSD from starting up or causes it to cease running.
0x0020 / 1:	Critical failures. An error that doesn't kill the SSSD, but one that indicates that at least one major feature is not going to work properly.
0x0040 / 2:	Serious failures. An error announcing that a particular request or operation has failed.
0x0080 / 3:	Minor failures. These are the errors that would percolate down to cause the operation failure of 2.
0x0100 / 4:	Configuration settings.
0x0200 / 5:	Function data.
0x0400 / 6:	Trace messages for operation functions.
0x0800:	Unused
0x1000 / 7:	Trace messages for internal control functions.
0x2000 / 8:	Contents of function-internal variables that may be interesting.
0x4000 / 9:	Extremely low-level tracing information.
0x8000:	Unused

In this list, the (deprecated) integer number from 0 to 9 is cumulative, 7 for instance is equivalent to 0x0010 + 0x0020 + 0x0040 + 0x0080 + 0x0100 + 0x0200 + 0x0400 + 0x1000, which is 0x17F0.

[nss] section

This is the part where you configure the **Name Service Switch** service. This service basically provides sources for configuration and name resolution mechanisms. It lists how a database should fetch what it needs from a given source or mechanism, like:

```
passwd: files ldap
```

That tells the password system to find the requested data from either a file or **LDAP**. The configuration is stored in the **/etc/nsswitch.conf** file. With **SSSD**, you won't have to modify this file, it will be handled automatically.

Here is the suggested configuration for this section:

```
#----------------------------------------------
# NSS service configuration
#
# Don't lookup for root on the LDAP server
#----------------------------------------------
[nss]
#   Trace   Fatal(0x0010),   Critical(0x0020),   Serious(0x0040),
Minor(0x0080), Config(0x0100) and Operation(0x0400) messages
# check /var/log/sssd/sssd_nss.log
debug_level=7
# Try three times when the remote LDAP server is not reachable
reconnection_retries = 3
# Don't get the root group from LDAP
filter_groups = root
# Don't get the root user from LDAP
filter_users = root
# The default Home dir
homedir_substring = /home
# The home dir to use if there is none configured in LDAP
fallback_homedir = /home/%u
# The default shell to use if none is configured in LDAP or does
not exist
shell_fallback = /bin/bash
```

Here, we have configured a few things, beside the log level. The most important part is to exclude the **root** user from any **LDAP** request (as we should always be able to log as **rot** on the machine even if we can't connect to the remote **LDAP** server). Otherwise, we tell the system to set the home directory of new users to be stored in **/home** with the user ID as the name of the directory (**fallback_homedir = /home/%u**) and defined its default shell (note that those definitions are not strictly necessary, they are just provided for clarity).

There are a few more parameters that are can be used, but the default values already fit most of the use case.

[pam] section

The **Pluggable Authentication Module** configuration is done here. **PAM** is meant to decouple authentication from the application, by delegating it to third party modules. An application will just have to use the **PAM API**, and the **PAM** configuration will route the request to the best authentication service.

Here is what we set:

```
#-------------------------------------------
# PAM service configuration
#-------------------------------------------
[pam]
#   Trace   Fatal(0x0010),   Critical(0x0020),   Serious(0x0040),
Minor(0x0080), Config(0x0100) and Operation(0x0400) messages
# check /var/log/sssd/sssd_pam.log
debug_level=7
# Try three times when the remote LDAP server is not reachable
reconnection_retries = 3
# The verbosity level (the higher, the more messages will be printed
on the user console)
# 0 : none
# (1): important messages
# 2 : informational messages
# 3 : all messages and debug information
pam_verbosity = 1
Nothing special here. It's important to understand that most of
the configuration takes place in the [domain] section.
[sudo] and [ssg] sections
We have grouped those two sections, as their configuration are the
same :
#-------------------------------------------
# SUDO service configuration
#-------------------------------------------
[sudo]
#   Trace   Fatal(0x0010),   Critical(0x0020),   Serious(0x0040),
Minor(0x0080), Config(0x0100) and Operation(0x0400) messages
# check /var/log/sssd/sssd_sudo.log
debug_level=7
# Try three times when the remote LDAP server is not reachable
reconnection_retries = 3
```

Note that the SSH section will be named [ssh], but will contain the same parameters with the same values.

[domain/xxx] section

This is the most important section in the **sssd.conf** file. It's where we tell the **SSSD** daemon how information are retrieved from the remote **LDAP** server. Actually, all the other sections depend on this section, which makes it way simpler to configure, as we don't have to duplicate the information in many configuration files (**nsswitch.conf**, **system-auth-ac**, **password-auth-ac**, ...).

The first important thing to remember is that the domain section has a name, which is the one we have put in the **[sssd]** section. In our example, this is **rb.symas.net**. Here is the proposed configuration:

```
#-------------------------------------------
# The SYMAS domain
#-------------------------------------------
[domain/rb.symas.net]
# Trace Fatal(0x0010), Critical(0x0020), Serious(0x0040), Minor(0x0080),
Config(0x0100) and Operation(0x0400) messages
# check /var/log/sssd/sssd_<domain name>.log
debug_level=7
```

```
# Try three times when the remote LDAP server is not reachable
reconnection_retries = 3
# Allow the system to cache credentials so that a user can always login
on the machine even if not being
# able to contact the remote LDAP server
cache_credentials = True
# Define the range of UID we are going to fetch from the LDAP server
min_id = 500
max_id =100000
# Producer mapped to ldap
id_producer = ldap
access_producer = ldap
auth_producer = ldap
chpass_producer = ldap
autofs_producer = ldap
sudo_producer = ldap
# LDAP information
ldap_uri = ldap://brie.rb.symas.net, ldap://cantal.rb.symas.net
ldap_search_base = dc=symas,dc=com
ldap_user_search_base                                                    =
ou=People,dc=symas,dc=com?subtree?(objectclass=posixaccount)
ldap_group_search_base                                                   =
ou=Groups,dc=symas,dc=com?subtree?(objectclass=posixGroup)
ldap_sudo_search_base = ou=People,dc=symas,dc=com
# Expect the schema used in the LDAP server to be RFC2307Bis
ldap_schema = rfc2307bis
# LDAP attributeType mapping
ldap_user_uid_number = uidNumber
ldap_user_gid_number = gidnumber
ldap_user_gecos = sn
ldap_user_home_directory = homeDirectory
ldap_user_shell = loginShell
ldap_user_ssh_public_key = sshPublicKey
ldap_user_fullname = cn
ldap_user_member_of = memberUid
ldap_group_object_class = posixGroup
ldap_group_name = cn
ldap_group_gid_number = gidNumber
ldap_group_member = member
ldap_access_filter = (objectClass=posixAccount)
ldap_tls_reqcert = demand
ldap_tls_cacert = /opt/symas/ssl/cacert.pem
ldap_id_use_start_tls = True
```

The parameters

cache_credentials

Tells the system to keep a tack of the (hashed) credentials, so that a user can login even if the remote **LDAP** is not reachable. The cache can be configured in the **[pam]** section, if needed (especially the expiration date)

min_id, max_id

Defines the range for UID and GID that are allowed.

id_producer, xxx_producer

Defines the producer in charge of managing identification and other operations. Here, it's clearly **LDAP**

ldap_uri

The URI for all the **LDAP** servers in charge of providing the requested operations. We may have many, to cover the case of an unreachable server.

ldap_xxx_search_base

The position in the **LDAP** server **DIT** where **SSSD** will look for information. Here we look for two places, with a **SUBTREE** search and we filter the returned entries using a specific **ObjectClass** (respectively **posixAccount** and **posixGroup**)

ldap_schema

Defines the schema to use on the **LDAP** schema for **LDAP** attribute we will get back.

ldap_user_xxx/ldap_group_yyy

The definition of attribute mapping. It associates a **SSSD** attribute with the equivalent **LDAP** attribute. Here, we just explicitly listed many of them, assuming most are already defaulting to the same value. This was done for clarity purposes.

ldap_access_filter

This parameter is mandatory as we have set the **access_producer** to use **LDAP**. Basically we set it to only accept entries that have the **posixAccount ObjectClass** (there are other possible options)

ldap_id_use_start_tls

Inform **SSSD** to use StartTLS to communicate in a secured fashion with the **LDAP** server.

ldap_tls_cacer

The file that contains the remote **LDAP** server **CA** certificate

ldap_tls_reqcert

Requires the server certificate to be asked and checked.

This is a working configuration for the servers that we have set, so for any other setup, some of those parameters might have to be modified (this is expected for the remote **LDAP** servers **URIs** for instance)

Configuring the remote LDAP server

Now that we have described the client configuration, we have to configure the remote **LDAP** server and the entries that will be injected in it.

LDAP server configuration

There is not so much things we need to change on the server. As we have made the **SSH** service depending on the **LDAP** content, we need to extend the server so that it allows entries to store the public key. There is no default Attribute or ObjectClass that cover this aspect (the **nisKeyObject ObjectClass** is certainly not the right one to use - it forces you to store both the public **and** private key for each user. A typical security breach!)

We will define an auxiliary **ObjectClass** and an **AttributeType** for this purpose, and store them into the **/opt/symas/etc/openldap/schema/openssh.schema** file.

```
#
# LDAP Public Key Patch schema for use with openssh-ldappubkey
# Author: Eric AUGE <eau@phear.org>
#
# Based on the proposal of : Mark Ruijter
#
# octetString SYNTAX
attributetype ( 1.3.6.1.4.1.24552.500.1.1.1.13 NAME 'sshPublicKey'
DESC 'MANDATORY: OpenSSH Public key'
EQUALITY octetStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.40 )
# printableString SYNTAX yes|no
objectclass ( 1.3.6.1.4.1.24552.500.1.1.2.0 NAME 'ldapPublicKey' SUP top
AUXILIARY
DESC 'MANDATORY: OpenSSH LPK objectclass'
MUST ( sshPublicKey $ uid )
)
```

Now, we must tell the server to use this schema. This is done by adding a reference to this schema in the **slapd.conf** file:

```
#---------------------------------------------------------------------
# SCHEMA INCLUDES
# Use only those you need and leave the rest commented out.
include /opt/symas/etc/openldap/schema/core.schema
include /opt/symas/etc/openldap/schema/ppolicy.schema
include /opt/symas/etc/openldap/schema/cosine.schema
include /opt/symas/etc/openldap/schema/inetorgperson.schema
include /opt/symas/etc/openldap/schema/krb5-kdc.schema
include /opt/symas/etc/openldap/schema/rfc2307bis.schema
include /opt/symas/etc/openldap/schema/openssh.schema
```

Note the last line.
The server need to be restarted after this change.

Modifying the entries

Now we need to inject the user's public key in each user's entry in the server. Each entry has to be modified so that the **ldapPublicKey ObjectClass** is added and the **sshPublicKey** attribute is added. Here is an exemple:

```
dn: uid=elecharny,ou=People,dc=symas,dc=com
objectClass: ldapPublicKey
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
cn: Emmanuel Lecharny
gidNumber: 2023
homeDirectory: /home/elecharny
sn:: TMOpY2hhcm55
```

```
sshPublicKey::
c3NoLWRzcyBBQUFBQjNOemFDMWtjM01BQUFDQkFNVWlRVTBvM1ZsUWRkTDArdkxpS0ZVMFI1
V3ZaVEc3OExyYk9pZytMbGt6RjBHS0ZHTTZsczE1eE51NkZxQzBvNHBERklwWk1QK3VXRjA0
WTMycXZmS2hCMWdFT0hPV3I3VE9yazZGbVJPOEoyb2NqaFQrMDdwcFRKSWVLS3N6NktQS0xw
WlVZMkdnTDF4U2hha2w4ZXdzVlNRZmh1TDUrWG1HM0RuVmc2Rm5BQUFBRlFEWDduN2o1QjJn
dGg3QVh0QTI3a3NvRUs4K05RQUFBSUVBdWpDVS9oREZ2RmpYNW13MnJ2K0owcHJuMDFOS3py
VStrZVYvWFZES1pPL3pyRWUwRFBZSEZmWFN0d21aTFNZcFV5R3h4cVNSd1VlUnAvRWE5STYx
NHdrSjFEUnJ5ZlBmZWg3czkxUlAwMFN3a2JaOS92NE8xLzBGUVpIZ29XY1hGa1V5RDFCOE1x
SENVVEhyYUwybWkyQkhHRTExSStTU2JQc2gwYm9sdTNzQUFBQ0FjRzN1VWdkNVg2bWZqdCtI
d1hPaFF3QjdmSGVRR2p5VWlrM09ZcU1GS1Avc25COVpIWlFrcktEQ2k5WjVUTDFtekRhTjc4
c2Q4U3FvdGFjWlgrWVJzdDE5ZzRkd3hLL3NMSitjMDc1YTA5YmNSeU1RVm52Yk5raXRxaXlS
UGVlOXNOci9XNnFzdDVSREVlZ2o3ZmZFdWpBWjRpdHVYTWJ5Qy81SW4yNjBmeWM9IGVsZWNo
YXJueUBlbGVjaGFybnktbGFwdG9wCg==
uid: elecharny
uidNumber: 2023
givenName: Emmanuel
loginShell: /bin/bash
mail: elecharny@symas.com
ou: Symas
userPassword:: xxxxxxx
```

The **sshPublicKey** contain the text form of the public key. Here it's shown in **Base64** format, but actually, the stored value is:

```
ssh-dss
      AAAAB3NzaC1kc3MAAACBAMUiQU0o3VlQddL0+vLiKFU0R5WvZTG78LrbOig+LlkzF
      0GKFGM6ls15xNu6FqC0o4pDFIpZMP+uWF04Y32qvfKhB1gEOHOWr7TOrk6FmRO8J2
      ocjhT+07ppTJIeKKsz6KPKLpZUY2GgL1xShakl8ewsVSQfhuL5+XmG3DnVg6FnAAA
      AFQDX7n7j5B2gth7AXtA27ksoEK8+NQAAAIEAujCU/hDFvFjX5mw2rv+J0prn01NK
      zrU+keV/XVDKZO/zrEe0DPYHFfXStwmZLSYpUyGxxqSRwUeRp/Ea9I614wkJ1DRry
      fPfeh7s91RP00SwkbZ9/v4O1/0FQZHgoWcXFkUyD1B8MqHCUTHraL2mi2BHGE11I+
      SSbPsh0bolu3sAAACAcG3uUgd5X6mfjt+HwXOhQwB7fHeQGjyUik3OYqMFKP/snB9
      ZHZQkrKDCi9Z5TL1mzDaN78sd8SqotacZX+YRst19g4dwxK/sLJ+c075a09bcRyMQ
      VnvbNkitqiyRPee9sNr/W6qst5RDEegj7ffEujAZ4ituXMbyC/5In260fyc=elech
      arny@elecharny-laptop
```