LDAP Anbindung

LDAP ist eines der möglichen und häufig verwendeten, externen Authentifikationsverfahren.

Allgemeine Infos

FabAccess unterstützt von Haus aus derzeitig noch keine LDAP-Integration. Das liegt unter anderem daran, dass die interne Benutzerdatenbank das besondere Passwort-Hashing-Verfahren Argon2 verwendet. Argon2 wird nur von OpenLDAP Servern unterstützt. Alle anderen LDAP Server wie z.B. Synology, FreeIPA, Active Directory oder ähnlich geben hierfür keine Unterstützung.

Es besteht jedoch die Möglichkeit über ein Python-Script Nutzer in eine `users.toml` Benutzerdatendatei (siehe hier) zu exportieren und diese dann wiederrum in FabAccess zu importieren. Dieser Umweg bedeutet, dass der FabAccess Server bei etwaigen Benutzeränderungen regelmäßig neugestartet werden muss - zum Beispiel 1x täglich nachts per Cronjob oder ähnlichem. Siehe Bekannte Probleme.

FabAccess users.toml LDAP Import

Quelle: https://github.com/vmario89/fabaccess-users-toml-ldap

Zweck

Dieses Script verbindet sich bei Ausführung mit den angegebenen Credentials zu einem LDAP(S)-Server und sucht nach passenden Nutzern, um eine FabAccess-kompatible `users.toml` Datei zu erzeugen.

Das Script dient außerdem auch als Beispielvorlage für andere Entwickler, die ggf. andere Anwendungen bzw. Benutzerquellen an FabAccess anbinden wollen und nach geeigneten Code-Quellen suchen.

> **Wichtig**: Dieses Script ersetzt **keine** native LDAP-Integration in FabAccess!

Installation

```
sudo apt install build-essential python3-dev libldap2-dev libsasl2-dev ldap-utils
```

```
cd /opt/fabinfra/scripts/
git clone https://github.com/vmario89/fabaccess-users-toml-ldap.git

cd /opt/fabinfra/scripts/fabaccess-users-toml-ldap/
chmod +x /opt/fabinfra/scripts/fabaccess-users-toml-ldap/main.py

python3 -m venv env
. env/bin/activate #activate venv
pip install -r requirements.txt

chown -R bffh:bffh /opt/fabinfra/scripts/fabaccess-users-toml-ldap/
```

Benutzung

Hilfe / Parameter anzeigen

```
cd /opt/fabinfra/scripts/fabaccess-users-toml-ldap/
python3 main.py --help
```

```
usage: main.py [-h] [-s SERVER] [-u USER] [-p PASSWORD] [-b BASEDN] [--filter_user
FILTER_USER] [--regex_groups REGEX_GROUPS] [--attrib_user ATTRIB_USER] [--attrib_groups
ATTRIB_GROUPS]
               [--attrib_password ATTRIB_PASSWORD] [--output OUTPUT]

options:
  -h, --help            show this help message and exit
  -s SERVER, --server SERVER
                        LDAP Server (Syntax: <protocol>://host:port, e.g.
ldap://192.168.1.1:389 or ldaps://192.168.1.1.:636)
  -u USER, --user USER  User, e.g. 'uid=root,cn=users,dc=yourserver,dc=com'
  -p PASSWORD, --password PASSWORD
                        Password
  -b BASEDN, --basedn BASEDN
                        BaseDN, for example 'cn=users,dc=yourserver,dc=com'
  --filter_user FILTER_USER
                        LDAP user filter, e.g. '(&(uid=*)(objectClass=posixAccount))'
  --regex_groups REGEX_GROUPS
                        LDAP group regex, e.g. 'cn=(.*),cn=groups,dc=yourserver,dc=com'. If
```

```
your group result is 'cn=administrator,cn=groups,dc=yourserver,dc=com', then the word
                        'administrator' gets properly exctracted. You can use
https://regex101.com for testing.
  --attrib_user ATTRIB_USER
                        Attribute name for FabAccess user name, e.g. 'uid'
  --attrib_groups ATTRIB_GROUPS
                        Attribute name for FabAccess user roles, e.g. 'memberOf'
  --attrib_password ATTRIB_PASSWORD
                        Attribute name for FabAccess user password hash, e.g.
'sambaNTPassword'. For OpenLDAP there is Argon2 hash support!
  --output OUTPUT       Target directory + file where to write the toml file. Please provide
the full name. If not given, users.toml will be written
```

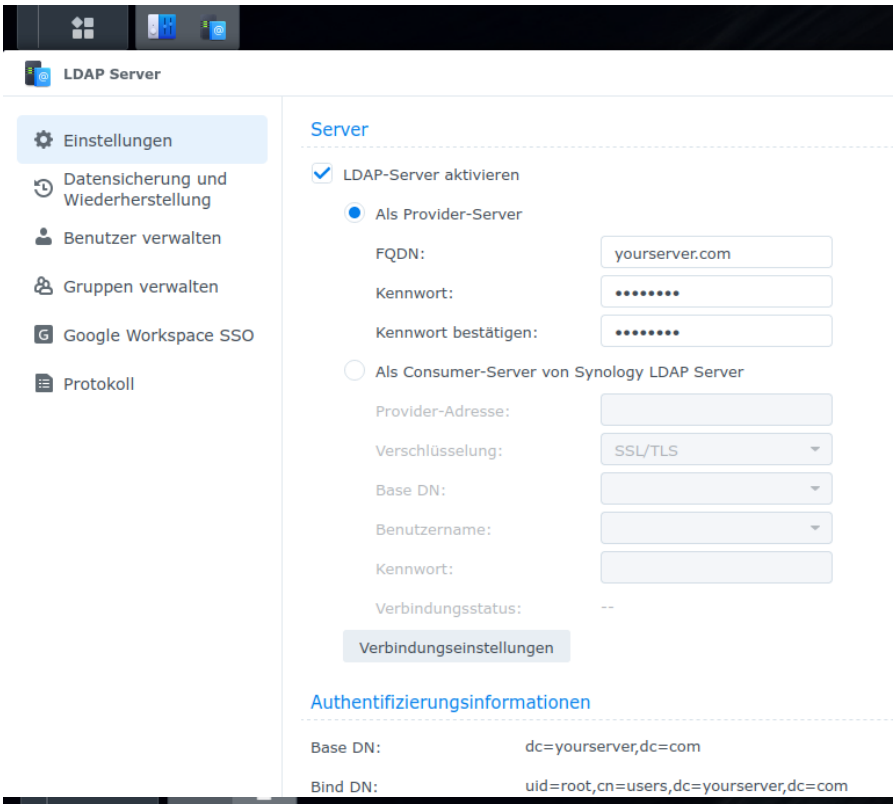
`users.toml` Datei schreiben

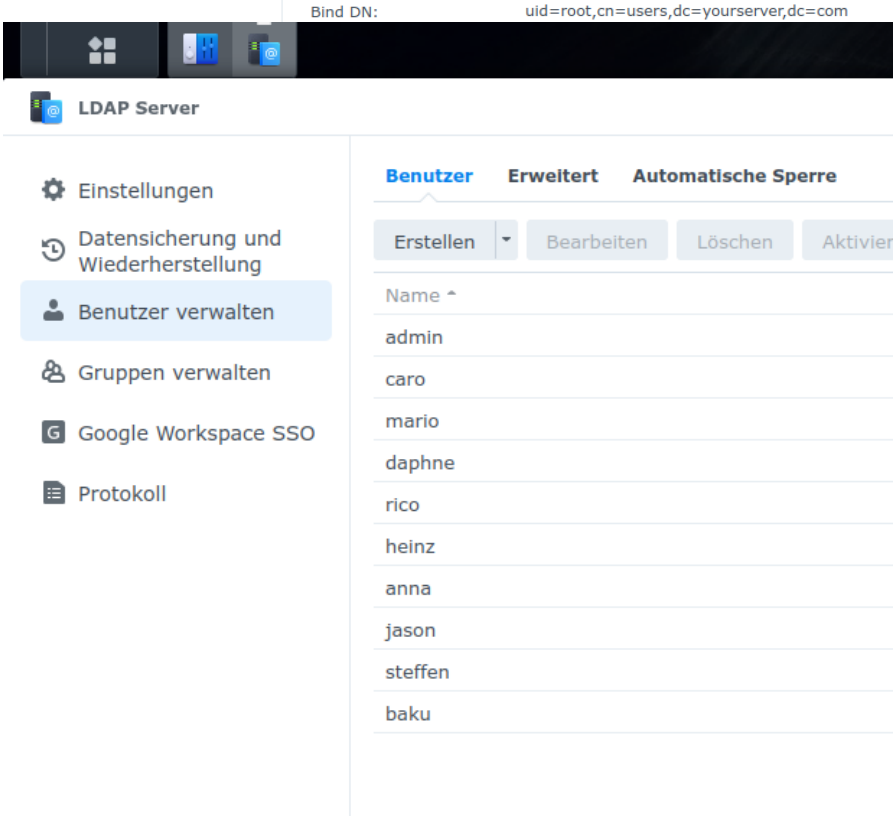
```
cd /opt/fabinfra/scripts/fabaccess-users-toml-ldap/
env/bin/python3 main.py \
  --server ldap://192.168.188.1:389 \
  --user="uid=admin,cn=users,dc=yourserver,dc=com" \
  --password pw \
  --basedn "cn=users,dc=yourserver,dc=com" \
  --filter_user "uid=*" \
  --regex_groups "cn=(.*),cn=groups,dc=yourserver,dc=com" \
  --attrib_user uid \
  --attrib_groups memberOf \
  --attrib_password sambaNTPassword \
  --output /opt/fabinfra/bffh-data/config/users.toml
```

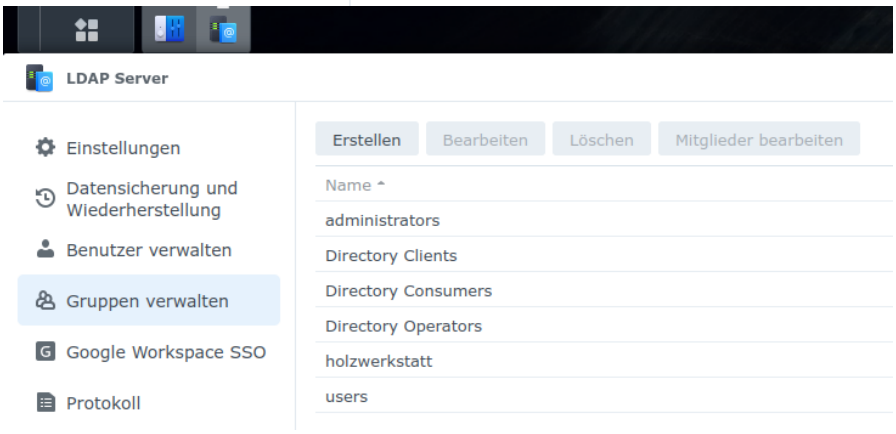
Nach dem Erstellen der Datei sollte diese überprüft und im Anschluss per `bffhd --load users.toml` geladen werden, um die Änderungen entsprechend zu reflektieren. Dafür gibt es auch geeignete Scripts.

Beispiel-Anbindung eines Synology LDAP Servers

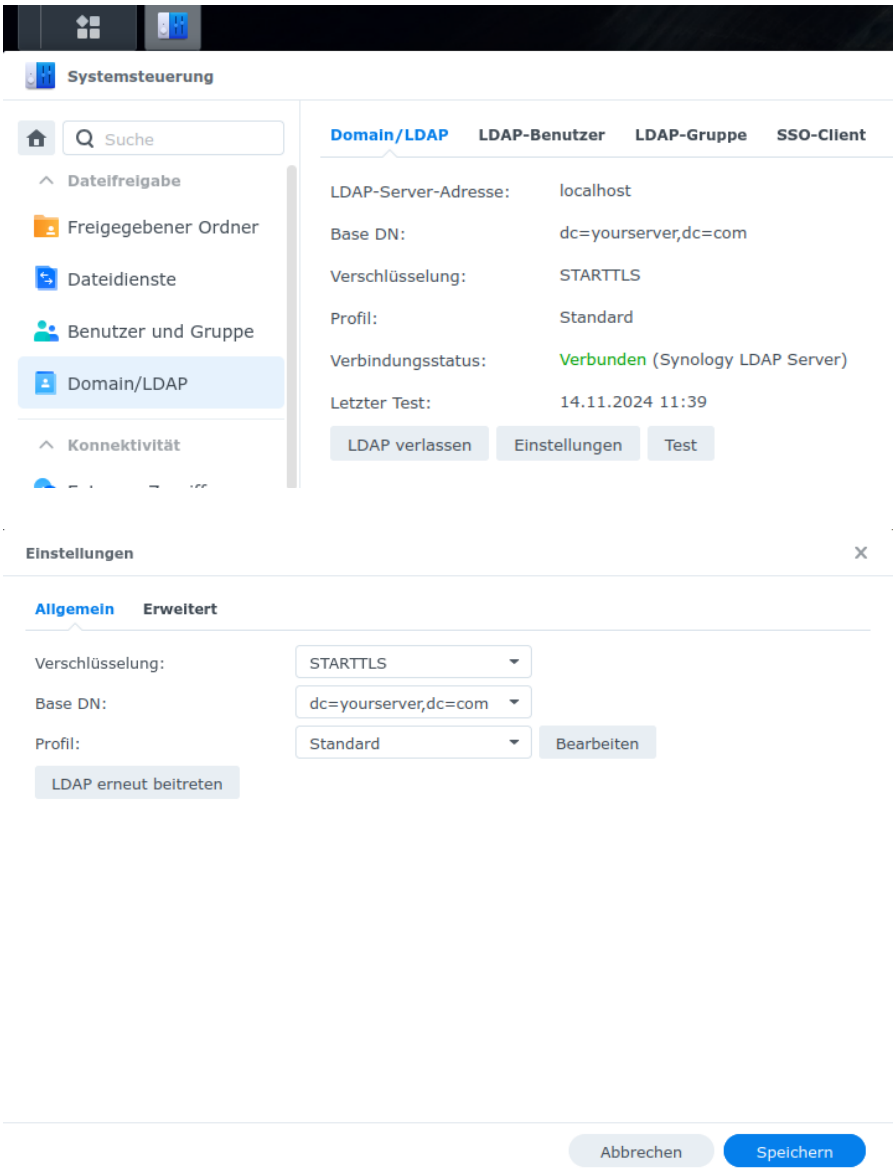
Vorraussetzung ist ein installierter und laufenden Synology LDAP Server. Hierzu auch siehe https://kb.synology.com/de-de/DSM/help/DirectoryServer/ldap_server?version=7. Dieser enthält das festgelegte Gruppen- und Benutzerschema und das entsprechende Berechtigungskonzept:


LDAP Server
Einstellungen
Datensicherung und Wiederherstellung
Benutzer verwalten
Gruppen verwalten
Google Workspace SSO
Protokoll
Server
LDAP-Server aktivieren
Als Provider-Server
FQDN:
yourserver.com
Kennwort:
Kennwort bestätigen:
Als Consumer-Server von Synology LDAP Server
Provider-Adresse:
Verschlüsselung:
SSL/TLS
Base DN:
Benutzername:
Kennwort:
Verbindungsstatus:
--
Verbindungseinstellungen
Authentifizierungsinformationen
Base DN:
dc=yourserver,dc=com
Bind DN:
uid=root,cn=users,dc=yourserver,dc=com


LDAP Server
Einstellungen
Datensicherung und Wiederherstellung
Benutzer verwalten
Gruppen verwalten
Google Workspace SSO
Protokoll
Benutzer
Erweitert
Automatische Sperre
Erstellen
Bearbeiten
Löschen
Aktivier
Name
admin
caro
mario
daphne
rico
heinz
anna
jason
steffen
baku


LDAP Server
Einstellungen
Datensicherung und Wiederherstellung
Benutzer verwalten
Gruppen verwalten
Google Workspace SSO
Protokoll
Erstellen
Bearbeiten
Löschen
Mitglieder bearbeiten
Name
administrators
Directory Clients
Directory Consumers
Directory Operators
holzwerkstatt
users

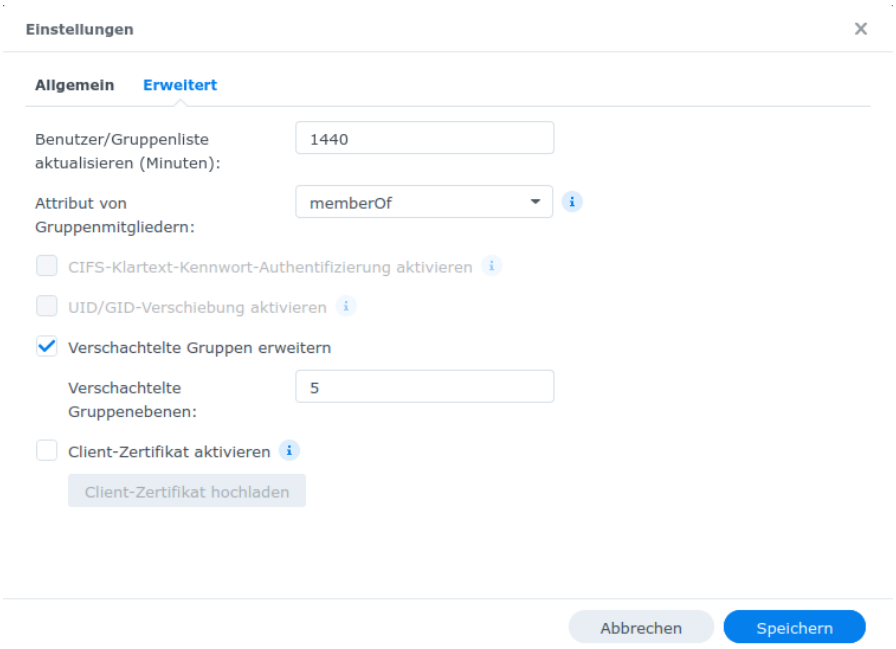
Ist dieser einmal eingerichtet, sollte die Synology NAS mit ihrem eigenen LDAP Server verbunden sein, also der Domäne beitreten:



Der Server sollte dann von allen Clients entsprechend erreichbar sein. Hierzu sind ggf. Interfaces, Port-Weiterleitungen oder die Firewall anzupassen. Wer eine sicherere Übertragung nutzen will, der benutzt das `ldaps://` Protokoll. Im Screenshot fnden wir auch auch das Attribut `--basedn` mit dem Beispielwert `dc=yourserver,dc=com`.

Das Attribut für die Benutzer (`--attrib_user`) lautet bei den meisten LDAP-Servern standardmäßig `uid` - so auch bei Synology. Unser `--user-filter` ist ein klassischer LDAP-Filter, der nach "uid" per Wildcard sucht und außerdem die Objektklasse "posixAccount" verlangt: `(&(uid=*)(objectClass=posixAccount))`. Hier muss je nach individuellem Setup herumprobiert werden, was das beste und stabilste Ergebnis wiederspiegelt.

Das gesuchte Attribut `--attrib_groups` finden wir in der Auswahlbox "Attribut von Gruppenmitgliedern:". In unserem Beispiel ist das `memberOf`.



Damit die ausgelesenen Gruppennamen nicht ellenlang werden, nutzen wir einen Regex-Filter --`regex_groups`, um nur gekürzte Rollennamen verwenden zu können. So wird aus der Zeichenkette `cn=administrator,cn=groups,dc=yourserver,dc=com` im Zusammenspiel mit dem Regex-String `cn=(.*),cn=groups,dc=yourserver,dc=com` letztlich nur das noch Gruppenwort `administrator` herausgefiltert. Für das Erstellen von `--regex_groups` kann https://regex101.com genutzt werden.

Für das Attribut --`attrib_password` können wir zum Beispiel das vom LDAP-Server bereitgestellte Passwortattribut `sambaNTPassword` verwenden. Hierzu sei gesagt, dass dies nicht hilfreich ist, da sich die Benutzer mit diesem Passworthash in FabAccess nicht anmelden können. Der Administrator muss das Passwort des Nutzers zurücksetzen. Das liegt daran, dass hier eine grundsätzliche Implementierung in bffh fehlt, die den LDAP-Server anspricht und nachfragt, ob der Nutzer in LDAP gerade existiert, die passende Berechtigung hat und die entspreche Aktion (z.B. den Login) ausführen darf bzw. das korrekte Passwort eingegeben hat. Von außen können wir den Passworthash zwar validieren, aber nicht in ein plaintext-Format umwandeln und damit also auch nicht in Argon2 umwandeln.

Bekannte Probleme

Dieses Script stellt **keine** saubere Lösung für die Nutzung von LDAP mit FabAccess bffh dar (zumindest nicht mit Version 0.4.2):

- etwaige Passwortänderungen am zentralen LDAP-Server müssen erneut in die `users.toml` Datei übertragen werden
- ändert der Nutzer oder der Administrator für den Nutzer das Passwort über die Client App, dann würde der Nutzer beim nächstens `users.toml` Import wieder

überschrieben. Diese Änderungen werden also auch nicht an den LDAP-Server gesendet
- die Password Hashes aus dem LDAP Server können in der Regel nicht mit FabAccess verwendet werden, da sie kein Argon2-Format aufweisen. Einzig OpenLDAP unterstützt Argon2 überhaupt. Aus adminstrativen Gründen macht eine Umstellung aller Nutzerpassworthashes auf Argon2 jedoch keinen Sinn. Eine native Integration von LDAP direkt in FabAccess ist also unumgänglich.

Empfehlung

Die LDAP-Gruppen sollten auf Rollen aufgebaut sein, die geeignet für die Werkstattabläufe sind. Deshalb empfehlen wir die Verwendung des FabAccess Config Generator. In diesem lassen sich Rollen geeignet definieren und die entsprechenden Maschinen in die Hauptkonfiguration schreiben. Davon abgeleitet bedarf es dann einer kompatibel gestaltetenen `users.toml`.

Automatisierung

Dieses Python-Script kann zum Beispiel als Cronjob automatisiert werden, um im Zeitintervall die LDAP-Benutzer zu aktualisieren und dann den bffhd Daemon neu zu starten. Falls zum Beispiel systemd verwendet wird, könnte das wie folgt aussehen. Wir packen dabei den obigen Script-Aufruf in ein eigenständiges Bash-Script, um den Cron Job übersichtlicher zu halten:

```
vim /opt/fabinfra/scripts/fabaccess-users-toml-ldap/cron.sh
```

```
#!/bin/bash

# create a recent users.toml by connecting to LDAP Server, grabbing data
/opt/fabinfra/scripts/fabaccess-users-toml-ldap/env/bin/python3
/opt/fabinfra/scripts/fabaccess-users-toml-ldap/main.py \
  --server ldap://192.168.188.1:389 \
  --user="uid=admin,cn=users,dc=yourserver,dc=com" \
  --password pw \
  --basedn "cn=users,dc=yourserver,dc=com" \
  --filter_user "uid=*" \
  --regex_groups "cn=(.*),cn=groups,dc=yourserver,dc=com" \
  --attrib_user uid \
  --attrib_groups memberOf \
  --attrib_password sambaNTPassword \
```

```
  --output /opt/fabinfra/bffh-data/config/users.toml

# restart bffhd
systemctl restart bffh.service
```

```
chmod +x /opt/fabinfra/scripts/fabaccess-users-toml-ldap/cron.sh
```

```
sudo vim /etc/cron.d/fabaccess-users-toml-ldap
```

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# At minute 0 past every hour.
0 */1 * * * bffh /opt/fabinfra/scripts/fabaccess-users-toml-ldap/cron.sh
```

Hinweise

Das Script basiert auf der Idee von https://gitlab.bht-berlin.de/innovisionlab/ldapbodge

Python Module

Das Script wurde getestet mit:

```
python -V
3.12.3

pip list installed
pip 24.0
pyasn1 0.6.1
pyasn1_modules 0.4.1
python-ldap 3.4.4
toml 0.10.2
```

Version #22
Erstellt: 2024-12-09 22:26:04 CET von Mario Voigt (Stadtfabrikanten e.V.)
Zuletzt aktualisiert: 2025-01-01 13:32:24 CET von Mario Voigt (Stadtfabrikanten e.V.)