

Authentication Web

ABOU JAMRA Maysa, HOUMANI Zeina



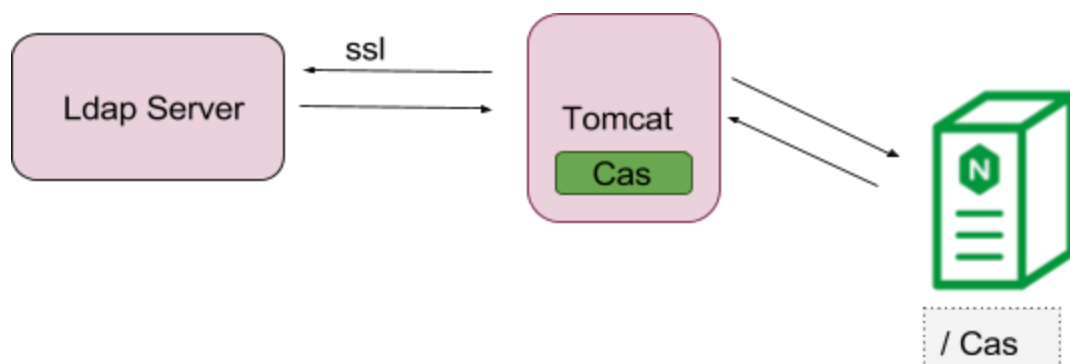
Master SRIV - Architectures de Sécurité

Enseignants : Fabien RICO, Yves CANIOU

Introduction

Dans ce TP on va utiliser 3 conteneurs docker:

1. Serveur Web **NGINX** : ce premier conteneur joue le rôle d'un reverse proxy qui consiste à recevoir les requêtes des clients pour les transmettre d'une façon sécurisée à notre serveur CAS.
2. Serveur **Tomcat + CAS** : Ce conteneur comporte notre serveur CAS qui est déployé sur un serveur tomcat. Le CAS est notre système d'authentification qui va permettre aux clients NGINX de s'authentifier auprès du serveur nginx .
3. Serveur **Ldap** : Nous allons installer un annuaire LDAP dans le 3ème conteneur afin de centraliser les informations utilisateurs pour l'authentification au niveau de notre serveur CAS .



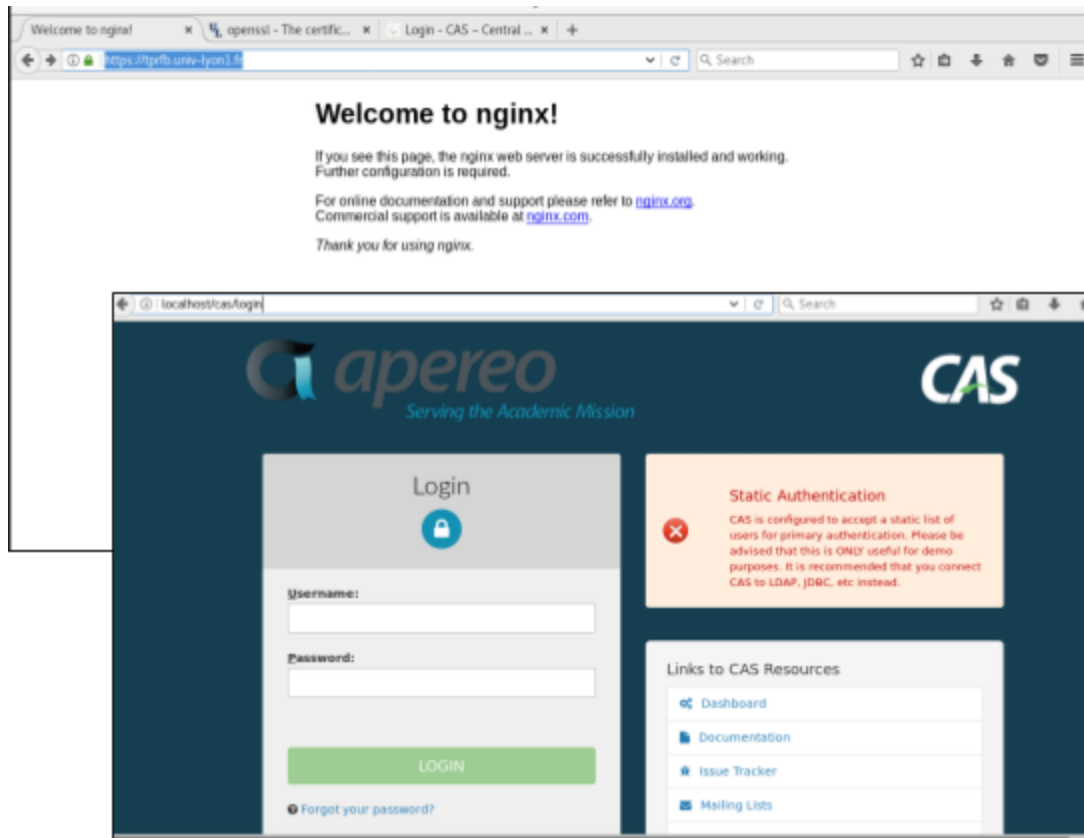
Scénario: Quand le client accède au serveur web, ce dernier va diriger les requêtes, en utilisant une connection sécurisée SSL, vers le serveur CAS. Ensuite, l'authentification au niveau du serveur CAS est déléguée à l'annuaire LDAP.

Pour mettre en place cette architecture on va commencer par la configuration SSL pour la connection sécurisée entre le serveur NGINX et le CAS.

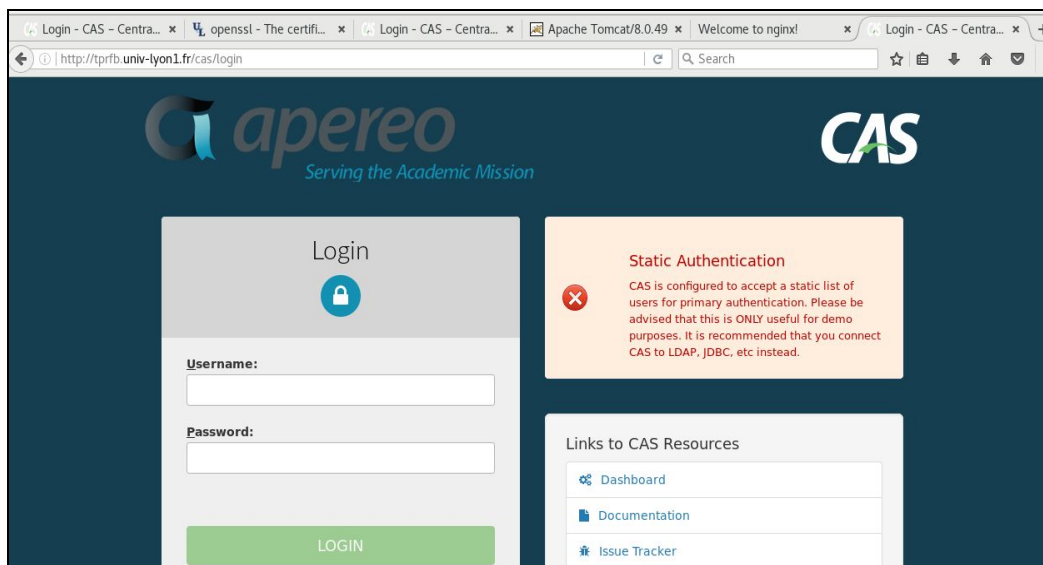
Etape 1 : Vérification des conteneurs

On lance les conteneurs:

- NGINX avec <http://tprfb.univ-lyon1.fr>
- CAS (déployé sur Tomcat) avec <http://localhost:8080/cas>



- CAS (de reverse proxy NGINX) avec <http://tprfb.univ-lyon1.fr/cas>



Etape 2 : La création de certificat

Dans le répertoire de configuration de notre serveur NGINX /etc/nginx on a créé un répertoire appelé ssl. Dans ce répertoire on va placer tous les fichiers nécessaires pour mettre en place notre certificat signé.

1. Génération de la clé privée

En utilisant cette commande openssl , on a créé la clé privée de notre serveur NGINX :

```
root@nginx:/etc/nginx# openssl genrsa -out ssl/tprfb.key 2048
Generating RSA private key, 2048 bit long modulus
.+++
.....+++
```

2. Création de demande de signature de certificat CSR

La commande ci-dessous a pour effet de créer notre requête tprfb.csr à partir de notre clé privée préalablement créée ainsi que des paramètres saisis dans le fichier de configuration openssl.cnf.

```
root@nginx:/etc/nginx# openssl req -new -out ssl/tprfb.csr -key
ssl/tprfb.key -config ssl/openssl.cnf

You are about to be asked to enter information that will be incorporated
into your certificate request.
-----
Country Name (2 letter code) [FR]:
State or Province Name (full name) [Rhone-Alpes-Auvergne]:
Locality Name (eg, city) [Lyon]:
Organization Name (eg, company) [TPR]:
Organizational Unit Name (eg, section) [SRIV]:
Common Name (e.g. server FQDN or YOUR name) [tprfb.univ-lyon1.fr]:
Email Address [maysa.abou-jamra@univ-lyon1.fr]:
```

Ce fichier contient notre clé publique à certifier par l'autorité de certification ainsi que notre signature pour qu'il vérifie notre identité.

```
root@nginx:/etc/nginx/ssl# cat tprfb.csr
```

```
-----BEGIN CERTIFICATE-----
MIIDuzCCAqMCCQCr6IB2GT+p4DANBgkqhkiG9w0BAQsFADCBmDELMAkGA1UE
IWMRwwwGgYDVQQDDBN0cHlyTS51bml2LWx54xLmZyMS0wKwYJKoZIhvcNAQkB
Fh5tYXlzYS5hYm91LWphbXJhQHVVuaXYtbHlv
bjEuZnlgwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDIgVHYj6uEZk
4Y
wtVC3Rd5Ea/9h5iirXclalH0/rv4QdAQFoyt2KCoBec4/GsXZNvo2fQIF1oggb//
ZDSVqJ3mhGL1InGwP4AqgUH/ZAEowCage/XD/tlb5
Pm7ExmttFtYci6xiEDOhTk1UcySKA+3wG30Smlc/Rkzt7Lor4dQxfTI3tLC6n5VI
aU4qgxhJAgMBAAEwDQYJKoZIhvcNAQELBQADggEBAGTZU5y0UGsEI4Yx4x+bH
MogY9IHpcf2bQfucJxi5W44PakzbTIFX1bJTgUA4OAK0oc5
CEWFQzOrV6ud7cX6S4yF5yhotdUkH4L9256cq0sblAht0KpRzTSJqQGtyadm
6YiW05QszN4Lkj2I119uDI8Av77Dzc+tJRkqEjIM8iPpYX9Q8WmOLxXiWh++hhA
hhxiFN3u3G2c3VJ6OBMKZipFEE3LqRGKQsxO6A/LikFPHZ2yY1SzX1vWiiX+2zk=
-----END CERTIFICATE-----
```

3. La signature du certificat par le CA

Avec cette commande, on obtient notre certificat tpr2M.crt signée par la clé privée du CA “cakey.key” qui correspond à son clé public certifié par cacert.crt.

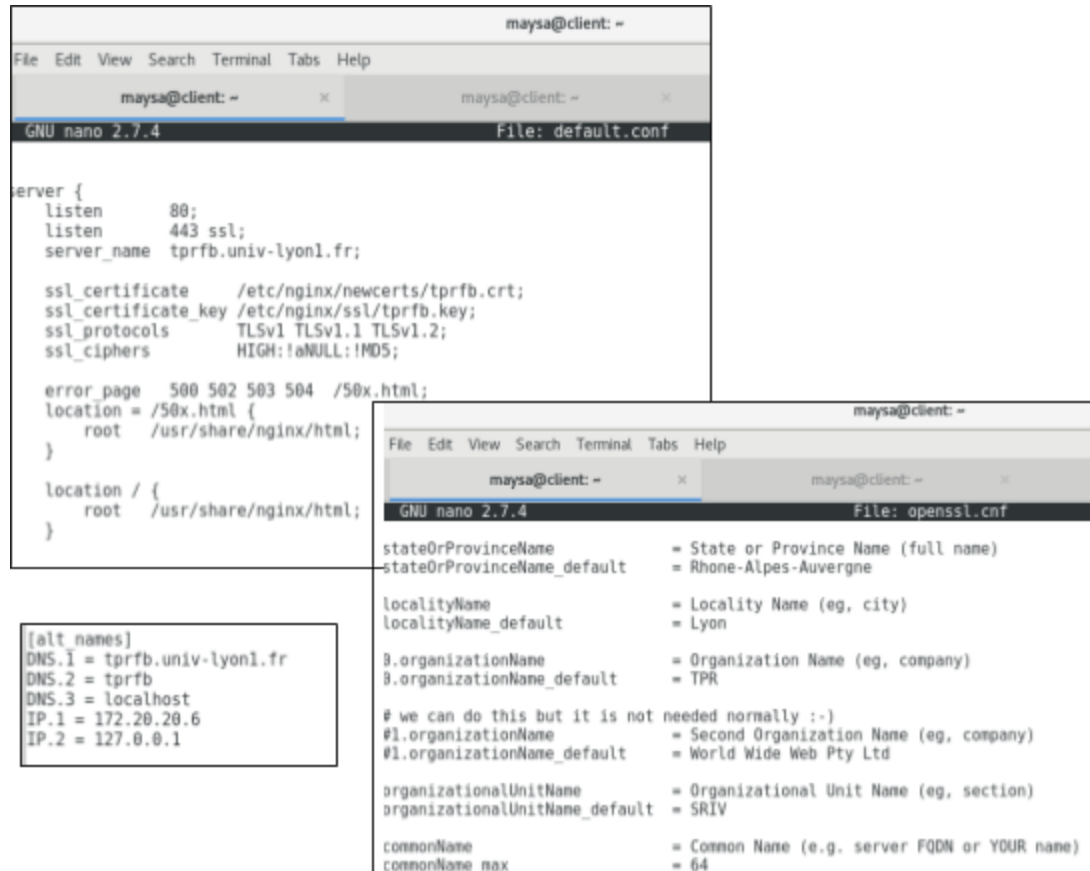
La sortie de la commande est la suivante :

```
root@client:/home/maysa/docker-etu/AutoriteDeCertificat# openssl x509 -req -in
/home/maysa/docker-etu/Nginx/config/ssl/tprfb.csr -CA
/home/maysa/docker-etu/Nginx/config/cacert.pem -CAkey cakey.pem
-CAcreateserial -out /home/maysa/docker-etu/Nginx/config/newcerts/tprfb.crt -days
500 -sha256
```

```
Signature ok
subject=C = FR, ST = Rhone-Alpes-Auvergne, L = Lyon, O = TPR, OU = SRIV, CN =
tpr2M.univ-lyon1.fr, emailAddress = maysa.abou-jamra@univ-lyon1.fr
Getting CA Private Key
Enter pass phrase for cakey.pem:
```

4. Configuration du fichier default.conf

Il faut configurer le fichier `/etc/nginx/config/conf.d/default.conf` dans le serveur NGINX pour prendre en compte le certificat signé.



The image shows two terminal windows. The top window displays the configuration of `default.conf` in the `nano` editor. The bottom window displays the configuration of `openssl.cnf` in the `nano` editor.

```
server {
    listen      80;
    listen      443 ssl;
    server_name tprfb.univ-lyon1.fr;

    ssl_certificate      /etc/nginx/newcerts/tprfb.crt;
    ssl_certificate_key  /etc/nginx/ssl/tprfb.key;
    ssl_protocols        TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers           HIGH:!aNULL:!MD5;

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root /usr/share/nginx/html;
    }

    location / {
        root /usr/share/nginx/html;
    }
}
```

```
[alt names]
DNS.1 = tprfb.univ-lyon1.fr
DNS.2 = tprfb
DNS.3 = localhost
IP.1  = 172.20.20.6
IP.2  = 127.0.0.1
```

```
stateOrProvinceName      = State or Province Name (full name)
stateOrProvinceName_default = Rhone-Alpes-Auvergne

localityName              = Locality Name (eg, city)
localityName_default      = Lyon

organizationName          = Organization Name (eg, company)
organizationName_default  = TPR

# We can do this but it is not needed normally :-}
#1.organizationName       = Second Organization Name (eg, company)
#1.organizationName_default = World Wide Web Pty Ltd

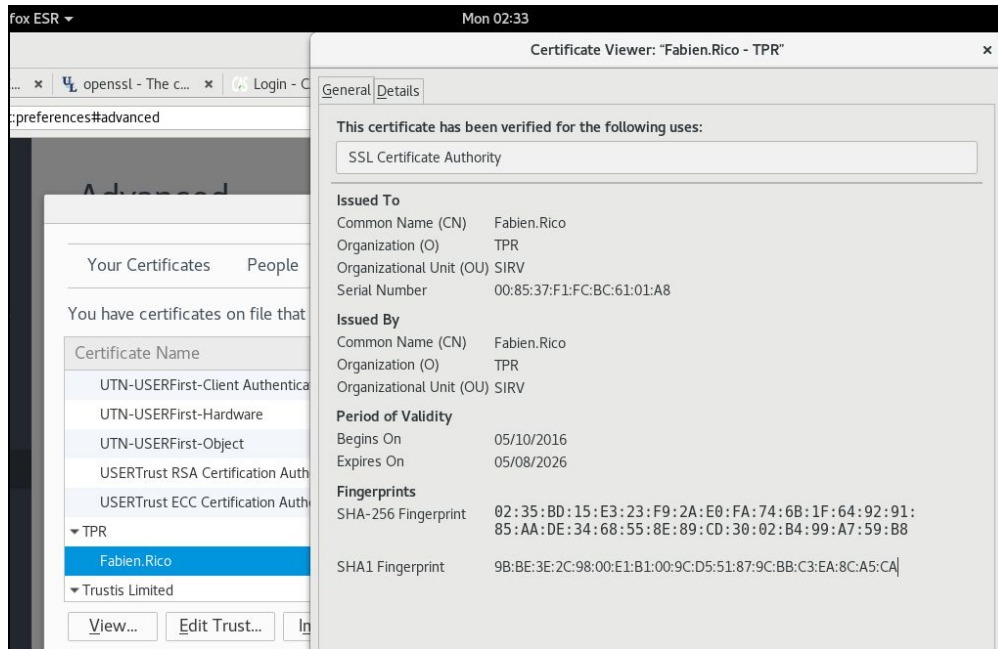
organizationalUnitName    = Organizational Unit Name (eg, section)
organizationalUnitName_default = SRIV

commonName                = Common Name (e.g. server FQDN or YOUR name)
commonName_max            = 64
```

5. installer le certificat de CA dans notre navigateur WEB

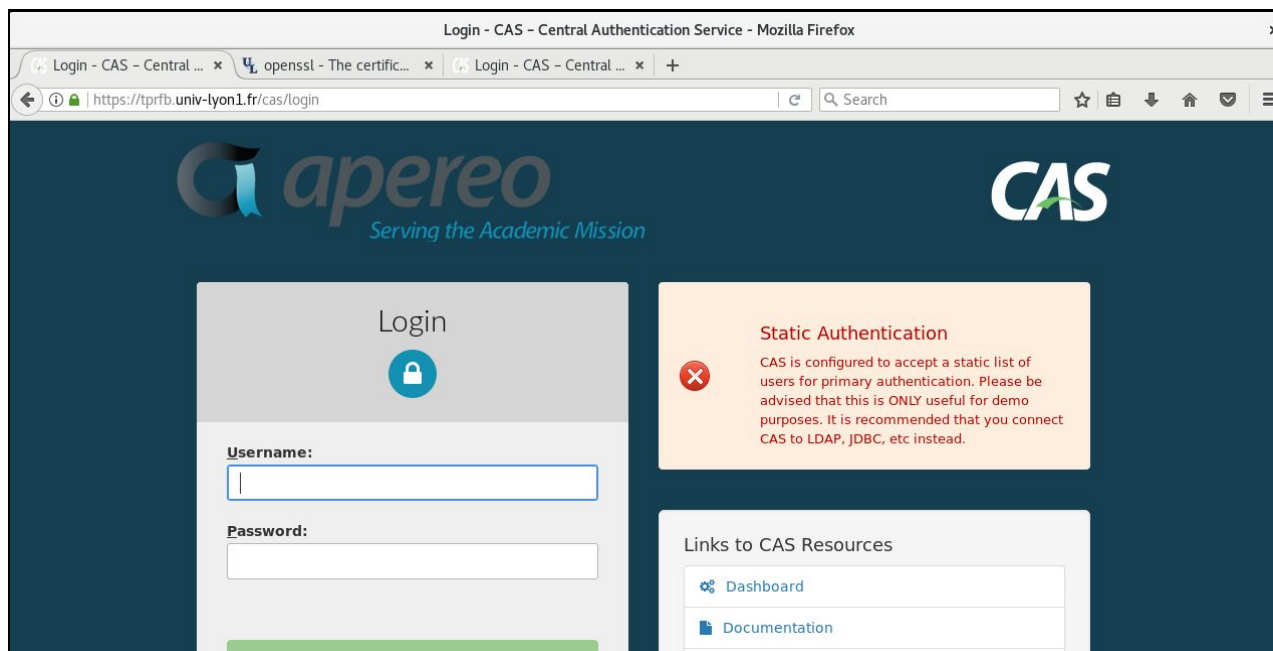
Afin qu'une connexion SSL sera établi, le client web doit être capable de vérifier la validité de certificat signé envoyer par le serveur NGINX. Ce certificat est signé par la clé privée du CA "cakey.pem" et qui peut être vérifié par le certificat cacert.pem . C'est pourquoi il faut importer le certificat d dans le browser des clients pour vérifier la signature.

La figure ci-dessous montre les informations dans le certificat de CA Issued To/By Fabien Rico



6. Tester la connection SSL

Enfin, quand on lance dans le browser l'url de serveur NGINX avec https, on reçoit l'affichage ci-dessous.



SUCCESS !!

Etape 2 : La configuration de serveur LDAP

1. Ajouter des entrées

```
root@openldap:/container/service/slapd/assets/certs# ldapadd -x -c -D  
"cn=admin,dc=tp,dc=univ-lyon1,dc=fr" -wtotoplop -H ldap://localhost:389 -f  
users.ldiff
```

```
Adding new entry "ou=Group,dc=tp,dc=univ-lyon1,dc=fr"  
adding new entry "ou=People,dc=tp,dc=univ-lyon1,dc=fr"  
adding new entry "uid=frico,ou=People,dc=tp,dc=univ-lyon1,dc=fr"  
adding new entry "uid=chaprot,ou=People,dc=tp,dc=univ-lyon1,dc=fr"  
adding new entry "uid=newton,ou=People,dc=tp,dc=univ-lyon1,dc=fr"
```

2. Tester le LDAPS

```
root@openldap:/container/service/slapd/assets/certs#  
LDAPTLS_CACERT=cacert.pem ldapsearch -x -H ldaps://localhost -b  
"dc=tp,dc=univ-lyon1,dc=fr" -D  
"uid=frico,ou=People,dc=tp,dc=univ-lyon1,dc=fr" -w totoplop
```

```
# extended LDIF  
#  
# LDAPv3  
# base <dc=tp,dc=univ-lyon1,dc=fr> with scope subtree  
# filter: (objectclass=*)  
# requesting: ALL  
#
```

```
# search result  
search: 2  
result: 32 No such object
```

```
# numResponses: 1
```


3. Tester la commande ldapsearch

```
root@openldap:/container/service/slapd/assets/certs# ldapsearch -x -D
"uid=newton,ou=People,dc=tp,dc=univ-lyon1,dc=fr" -wtoto

# extended LDIF
#
# LDAPv3
# base <> (default) with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# search result
search: 2
result: 32 No such object

# numResponses: 1
```

4. Autre commande

```
root@openldap:/container/service/slapd/assets/certs#
LDAPTLS_CACERT=./openldap/certif/cacert.pem ldapsearch -x -H
ldaps://127.0.0.2 -b "dc=tp,dc=univ-lyon1,dc=fr" -D
"uid=frico,ou=People,dc=tp,dc=univ-lyon1,dc=fr" -w totoplop -d1

ldap_url_parse_ext(ldaps://127.0.0.2)
ldap_create
ldap_url_parse_ext(ldaps://127.0.0.2:636/??base)
ldap_sasl_bind
ldap_send_initial_request
ldap_new_connection 1 1 0
ldap_int_open_connection
ldap_connect_to_host: TCP 127.0.0.2:636
ldap_new_socket: 3
ldap_prepare_socket: 3
ldap_connect_to_host: Trying 127.0.0.2:636
ldap_pvt_connect: fd: 3 tm: -1 async: 0
attempting to connect:
connect success
ldap_err2string
ldap_sasl_bind(SIMPLE): Can't contact LDAP server (-1)
```