Konfiguration auf Clientseite zur Client-Authentification mit technischem User

Als Erweiterung zu unserem ersten Test wollen wir nun keine anonymen LDAP Abfragen zulassen, sondern hierzu einen speziellen technischen User verwenden.

In abgesicherten Umgebungen wird der Zugriff auf den LDAP-Server nicht von jedermann ohne Passwort, auch *anonymous bind* genannt unterbunden. Stattdessen muss ich der Klient bei den Anfragen eines technischen Users bedienen, der auch ein Passwort benutzt, welches dem LDAP-Server bekannt ist.

Auch hier wollen wir bei den betreffenden Clients die Authentifizierung der einzelnen User nicht mehr gegen die lokale **/etc/shadow** laufen lassen. Auch bei diesem Anwendungsfall werden wir bei der Realisierung keinen System Security Services Daemon **sssd** einsetzen. Auf diesen werden wir noch in einem separatem Kapitel detailliert eingehen.

technischer User für LDAP-Server-Zugriff

Bei unserem Anwendungsbeispiel gehen wir von einem bereits installiertem und konfiguriertem OpenLDAP-Server aus, wie im Kapitel Grundinstallation des OpenLDAP Servers und Datenerstbefüllung des OpenLDAP Servers aus.

Konfiguration

Im ersten Schritt werden wir uns nun einen eigenen speziellen technischen User anlegen, mit dem später die Anfragen an unseren OpenLDAp-Server gerichtet werden sollen.

Der Einfachheit halber wollen wir hierzu folgenden User in unserem DIT[1)] hinterlegen:

- **cn=Technischeruser,dc=nausch,dc=org**

Unser User benötigt natürlich auch ein entsprechendes Passwort, welches wir nun wir folgt anlegen.

```
# slappasswd -h {SSHA}
```

```
New password:
Re-enter new password:
{SSHA}YpKKoS1lV1AdAX1StGe1lTembvZW4XagnkLdWZ2Y4Xkw
```

Im nächsten Schritt legen wir uns eine Konfigurationsdatei im ***.LDIF**-Format an, die Die Definition unseres speziellen Users beinhaltet. Wir legen also im Verzeichnis ***/etc/openldap/ldif/*** die Datei

technischeruser.ldif an.

```
# vim /etc/openldap/ldif/technischeruser.ldif
```

/etc/openldap/ldif/technischeruser.ldif

```
dn: cn=Technischeruser,dc=nausch,dc=org
cn: Technischeruser
objectClass: organizationalRole
objectClass: simpleSecurityObject
objectClass: top
userPassword: {SSHA}YpKKoS1lV1AdAX1StGe1lTembvZW4XagnkLdWZ2Y4Xkw
```

Bevor wir nun die Daten aus der LDIF-Datei ***/etc/openldap/ldif/technischeruser.ldif*** in den DIT[2] importieren können, stoppen wir kurz den OpenLDAP-Server.

```
 # service slapd stop
```

```
 Stopping slapd:                                            [  OK
]
```

Mit folgendem Befehl importieren wir nun die Daten aus der LDIF-Datei in den DIT.

```
# slapadd -v -l /etc/openldap/ldif/technischeruser.ldif
```

```
added: "cn=Technischeruser,dc=nausch,dc=org" (00000009)
_#################### 100.00% eta   none elapsed            none fast!
Closing DB...
```

Anschließend vergessen wir nicht unseren OpenLDAP-Server wieder zu staren!

```
 # service slapd start
```

```
 Stopping slapd:                                            [  OK
]
```

Abfragetest

Wir können nun unsere Konfiguration überprüfen und eine LDAP-Abfrage mit unserem gerade angelegtem *Technischenuser* und seinem zugehörigen Passwort testen.

```
# ldapsearch -x -LLL -H ldap://ldap.dmz.nausch.org -b "dc=nausch,dc=org"
```

```
"cn=Technischeruser" -W -D "cn=Technischeruser,dc=nausch,dc=org"
```

```
Enter LDAP Password:
dn: cn=Technischeruser,dc=nausch,dc=org
cn: Technischeruser
objectClass: organizationalRole
objectClass: simpleSecurityObject
objectClass: top
userPassword:: YpKKoS1lV1AdAX1StGe1lTembvZW4XagnkLdWZ2Y4Xkwa3c=
```

Deaktivierung des "Anonymous bind" beim LDAP-Server-Zugriff

In unserer abgesicherten Umgebung sollen ausschließlich authentifizierte Abfragen und Zugriffe auf unseren OpenLADP-Server gestattet sein. Hierzu haben wir uns im vorherigem Abschnitt eigens einen technischen Useraccount angelegt.

Konfiguration

Zur Deaktivierung werden wir nun unsere zentrale Konfiguration im OpenLDAP-Server anpassen.

Zum besseren Verständnis fragen wir erst eiunmal die aktuelle Konfiguration ab.

```
 # ldapsearch -W -x -D cn=config -b cn=config "(objectclass=olcGlobal)"
```

```
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <cn=config> with scope subtree
# filter: (objectclass=olcGlobal)
# requesting: ALL
#

# config
dn: cn=config
objectClass: olcGlobal
cn: config
olcConfigFile: /etc/openldap/slapd.conf
olcConfigDir: /etc/openldap/slapd.d
olcAllows: bind_v2
olcArgsFile: /var/run/openldap/slapd.args
olcAttributeOptions: lang-
olcAuthzPolicy: none
olcConcurrency: 0
olcConnMaxPending: 100
olcConnMaxPendingAuth: 1000
```

```
olcGentleHUP: FALSE
olcIdleTimeout: 15
olcIndexSubstrIfMaxLen: 4
olcIndexSubstrIfMinLen: 2
olcIndexSubstrAnyLen: 4
olcIndexSubstrAnyStep: 2
olcIndexIntLen: 4
olcLocalSSF: 71
olcLogLevel: Stats
olcPidFile: /var/run/openldap/slapd.pid
olcReadOnly: FALSE
olcReferral: ldap://ldap.dmz.nausch.org
olcReverseLookup: FALSE
olcSaslSecProps: noplain,noanonymous
olcSockbufMaxIncoming: 262143
olcSockbufMaxIncomingAuth: 16777215
olcThreads: 16
olcTLSCRLCheck: none
olcTLSVerifyClient: never
olcToolThreads: 1
olcWriteTimeout: 0

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Die Änderung an diesewr Konfiguration nehmen wir nun nicht an Hand einer Änderung einer zentralen Konfigurationsdatei vor, wie wir dies unter Umstaänden von früheren Installationen unter CentOS 5 gewohnt waren.

Die Änderungen erfolgen mit Hilfe eines ***.LDIF**-Datei. Wir legen uns also diese im gewohnten Verzeichnis ***/etc/openldap/ldif/*** an und wählen hierzu z.B. als Dateinamen **disallow_anonymous_bind.ldif**.

```
# vim /etc/openldap/ldif/disallow_anonymous_bind.ldif
```

/etc/openldap/ldif/disallow_anonymous_bind.ldif

```
dn: cn=config
changetype: modify
add: olcDisallows
olcDisallows: bind_anon
-
add: olcRequires
olcRequires: authc
```

Zur Übernahme der Änderungen in den laufenden OpenLDAP-Server **slapd**, die wir gerade in der

LDIP-Datei definiert haben, benutzen wir den Befehl **ldapmodify**

```
 # ldapmodify -W -x -D cn=config -f
/etc/openldap/ldif/disallow_anonymous_bind.ldif
```

```
 Enter LDAP Password:
 modifying entry "cn=config"
```

Fragen wir nun erneiut die Konfiguration unseres **slapd** ab, so finden wir am Ende die erfolgte Änderung aus unserer ldif-Datei:

- **olcDisallows: bind_anon**
- **olcRequires: authc**

```
 # ldapsearch -W -x -D cn=config -b cn=config "(objectclass=olcGlobal)"
```

```
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <cn=config> with scope subtree
# filter: (objectclass=olcGlobal)
# requesting: ALL
#

# config
dn: cn=config
objectClass: olcGlobal
cn: config
olcConfigFile: /etc/openldap/slapd.conf
olcConfigDir: /etc/openldap/slapd.d
olcAllows: bind_v2
olcArgsFile: /var/run/openldap/slapd.args
olcAttributeOptions: lang-
olcAuthzPolicy: none
olcConcurrency: 0
olcConnMaxPending: 100
olcConnMaxPendingAuth: 1000
olcGentleHUP: FALSE
olcIdleTimeout: 15
olcIndexSubstrIfMaxLen: 4
olcIndexSubstrIfMinLen: 2
olcIndexSubstrAnyLen: 4
olcIndexSubstrAnyStep: 2
olcIndexIntLen: 4
olcLocalSSF: 71
olcLogLevel: Stats
olcPidFile: /var/run/openldap/slapd.pid
olcReadOnly: FALSE
olcReferral: ldap://ldap.dmz.nausch.org
olcReverseLookup: FALSE
```

```
olcSaslSecProps: noplain,noanonymous
olcSockbufMaxIncoming: 262143
olcSockbufMaxIncomingAuth: 16777215
olcThreads: 16
olcTLSCRLCheck: none
olcTLSVerifyClient: never
olcToolThreads: 1
olcWriteTimeout: 0
olcDisallows: bind_anon
olcRequires: authc

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Abfragetest

Versuchen wir nun eine anonymous-bind Abfrage gegen unseren OpenLDAP-Server hat dies keinen Erfolg und wir bekommen einen entsprechenden Warnhinweis.

```
 # ldapsearch -x -LLL -H ldap://ldap.dmz.nausch.org -b "dc=nausch,dc=org"
"uid=django"
```

```
 ldap_bind: Inappropriate authentication (48)
         additional info: anonymous bind disallowed
```

Die Abfrage unseres Nutzers *Django* erfolgt nun richtiger Wiese mit Hilfe unseres technischen Users *Technischeruser* den wir uns hierzu eigens konfiguration angelegt haben.

```
 # ldapsearch -x -LLL -H ldap://ldap.dmz.nausch.org -b "dc=nausch,dc=org"
"uid=django" -W -D "cn=Technischeruser,dc=nausch,dc=org"
```

```
Enter LDAP Password:
dn: uid=django,ou=People,dc=nausch,dc=org
uid: django
cn: Django
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
shadowLastChange: 15272
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 500
```

```
gidNumber: 500
homeDirectory: /home/django
gecos: Django
```

Zugriffsrechte beschränken

Mit Hilfe des nachfolgenden Befehls kann kontrolliert werden, welche Zugriffsrechte in den aktuell enthaltenen Benutzerstrukturen im DIT[3)] enthalten sind. Somit kann man sich einen Überblick verschaffen, welche Anpassungen ggf. im DIT noch vorzunehmen sind.

```
 # ldapsearch -W -x -D cn=config -b olcDatabase={-1}frontend,cn=config
```

```
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <olcDatabase={-1}frontend,cn=config> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# {-1}frontend, config
dn: olcDatabase={-1}frontend,cn=config
olcPasswordHash: {SSHA}
objectClass: olcDatabaseConfig
objectClass: olcFrontendConfig
olcDatabase: {-1}frontend
olcAccess: {0}to attrs=userPassword  by self write  by
dn.base="cn=manager,dc=nausch,dc=org" write  by anonymous auth  by * none
olcAccess: {1}to *  by self write  by dn.base="cn=manager,dc=nausch,dc=org"
write  by * read
olcAddContentAcl: FALSE
olcLastMod: TRUE
olcMaxDerefDepth: 0
olcReadOnly: FALSE
olcSchemaDN: cn=Subschema
olcMonitoring: FALSE

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Die beiden Zeilen mit dem vorangestelltem **olcAccess** wollen wir uns kurz genauer ansehen.

Die vorangestellten Zeilen haben folgende Bedeutung:

- **userPassword**
  - kann nur vom **„Eigentümer selbst“** *geschrieben*,
  - oder von **„dn.base=„cn=manager,dc=nausch,dc=org“** *geschrieben*,
  - oder von **„anonymous“** *authentifizierend zugegriffen*,
  - und von allen anderen kann *kein Zugriff* durchgeführt werden.
- *** (auf den gesamten Baum)**
  - kann nur vom **„Eigentümer selbst“** *geschrieben*,
  - oder von **„dn.base=„cn=manager,dc=nausch,dc=org“** *geschrieben*,
  - oder von allen anderen *lesender Zugriff* durchgeführt werden.

Zur Änderung dieser bereits vorhandenen Zugriffsberechtigungen muß erst die vorhandene aktuelle ACL-Konfiguration gelöscht und anschließend die künftige ACL-Konfiguration neu angelegt werden.

Konfiguration

Da wir uns bei der Installation und Konfiguration unseres OpenLDAP-Servers entschieden hatten, die komplette Konfiguration via **.LDIF**-Dateien zu erledigen, dürfen wir **keinenfalls** versuchen, Änderungen an den Dateien unterhalb ***/etc/openldap/*** vorzunehmen!

Vielmehr legen wir uns zwei **.ldif**-Dateien an, mit Hilfe derer wir die Anpassung der Benutzerrechte anpassen.

1. **Löschen** der aktuellen ACL-Konfiguration:

```
# vim /etc/openldap/ldif/frontend_acl_delete.ldif
```

/etc/openldap/ldif/frontend_acl_delete.ldif

```
dn: olcDatabase={-1}frontend,cn=config
delete: olcAccess
olcAccess: to attrs=userPassword  by self write  by
dn.base="cn=manager,dc=nausch,dc=org" write  by anonymous auth  by
* none
olcAccess: to *  by self write  by
dn.base="cn=manager,dc=nausch,dc=org" write  by * read
```

2. **Neuanlage** der zukünftigen ACL-Konfiguration:

```
# vim /etc/openldap/ldif/frontend_acl_update.ldif
```

/etc/openldap/ldif/frontend_acl_update.ldif

```
dn: olcDatabase={-1}frontend,cn=config
add: olcAccess
olcAccess: to
attrs=userPassword,shadowLastChange,shadowMax,shadowWarning  by
self write  by dn="cn=Manager,dc=nausch,dc=org" write  by
dn="cn=Technischeruser,dc=nausch,dc=org" read  by anonymous auth
by * none
olcAccess: to dn="cn=Manager,dc=nausch,dc=org"  by self write  by
* none
olcAccess: to dn="cn=Technischeruser,dc=nausch,dc=org"  by self
write  by dn="cn=Manager,dc=nausch,dc=org" write  by * none
olcAccess: to dn.regex="cn=([^,]+),ou=Group,dc=nausch,dc=org"  by
self write  by dn="cn=Manager,dc=nausch,dc=org" write  by
dn="cn=Technischeruser,dc=nausch,dc=org" read  by
dn.exact,expand="uid=$1,ou=People,dc=nausch,dc=org" read  by *
none
olcAccess: to dn.regex="uid=([^,]+),ou=People,dc=nausch,dc=org"
by self write  by dn="cn=Manager,dc=nausch,dc=org" write  by
dn="cn=Technischeruser,dc=nausch,dc=org" read  by
dn.exact,expand="uid=$1,ou=People,dc=nausch,dc=org" read  by *
none
olcAccess: to *  by self write  by
dn.base="cn=manager,dc=nausch,dc=org" write  by * read
```

Anschließend laden wir die beiden **.LDIF**-Dateien in den OpenLDAP-Server und löschen die bestehenden Zugriffsregelungen und tragen unsere neuen ein.

1. **Löschen** der aktuellen ACL-Konfiguration:

```
# ldapmodify -W -x -D cn=config -f
/etc/openldap/ldif/frontend_acl_delete.ldif
```

```
Enter LDAP Password:
modifying entry "olcDatabase={-1}frontend,cn=config"
```

2. **Neuanlage** der neuen ACL-Konfiguration:

```
# ldapmodify -W -x -D cn=config -f
/etc/openldap/ldif/frontend_acl_update.ldif
```

```
Enter LDAP Password:
modifying entry "olcDatabase={-1}frontend,cn=config"
```

Ob nun unsere Änderungen in den laufenden slampd-Prozess übernommen wurden überprüfen wir am einfachsten mit dem nachfolgenden Aufruf, der die gesamten Zugriffsrechte in den aktuell enthaltenen Benutzerstrukturen im DIT[4)] enthalten sind.

```
# ldapsearch -W -x -D cn=config -b olcDatabase={-1}frontend,cn=config
```

```
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <olcDatabase={-1}frontend,cn=config> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# {-1}frontend, config
dn: olcDatabase={-1}frontend,cn=config
olcPasswordHash: {SSHA}
objectClass: olcDatabaseConfig
objectClass: olcFrontendConfig
olcDatabase: {-1}frontend
olcAddContentAcl: FALSE
olcLastMod: TRUE
olcMaxDerefDepth: 0
olcReadOnly: FALSE
olcSchemaDN: cn=Subschema
olcMonitoring: FALSE
olcAccess: {0}to attrs=userPassword,shadowLastChange,shadowMax,shadowWarning
by self write  by dn="cn=Manager,dc=nausch,dc=org" write  by
dn="cn=Technischeruser,dc=nausch,dc=org" read  by anonymous auth  by * none
olcAccess: {1}to dn="cn=Manager,dc=nausch,dc=org"  by self write  by * none
olcAccess: {2}to dn="cn=Technischeruser,dc=nausch,dc=org"  by self write  by
dn="cn=Manager,dc=nausch,dc=org" write  by * none
olcAccess: {3}to dn.regex="cn=([^,]+),ou=Group,dc=nausch,dc=org"  by self
write  by dn="cn=Manager,dc=nausch,dc=org" write  by
dn="cn=Technischeruser,dc=nausch,dc=org" read  by
dn.exact,expand="uid=$1,ou=People,dc=nausch,dc=org" read  by * none
olcAccess: {4}to dn.regex="uid=([^,]+),ou=People,dc=nausch,dc=org"  by self
write  by dn="cn=Manager,dc=nausch,dc=org" write  by
dn="cn=Technischeruser,dc=nausch,dc=org" read  by
dn.exact,expand="uid=$1,ou=People,dc=nausch,dc=org" read  by * none
olcAccess::
ezV9dG8gKiAgYnkgc2VsZiB3cml0ZSAgYnkgZG4uYmFzZT0igVsZiz1uYXVzY2gsZGM9b3JnIiB3
cml0ZSAgKiAgYnkgYnkgKiByZWFkIA==

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Überprüfung

Zur Überprüfung ob unsere neu gesetzten Zugriffsregelung auch greifen versuchen wir im ersten

Schritt, die Daten des Nutzers **bigchief** abzufragen, dies jedoch als Nutzer **django**.

```
 # ldapsearch -x -LLL -H ldap://ldap.dmz.nausch.org -b
"ou=People,dc=nausch,dc=org" "uid=bigchief" -W -D
"uid=django,ou=People,dc=nausch,dc=org"
```

```
 Enter LDAP Password:
```

Erhalten wir ***keine Ausgabe*** ist alles **O.K.**!

Rufen wir die Daten des zugrhörigen Nutzers, ab so klappt dies natürlich wie erhofft. Im folgenden Beispiel frägt also der Nutzer **django** die Daten des Nutzers **django** ab.

```
 # ldapsearch -x -LLL -H ldap://ldap.dmz.nausch.org -b
"ou=People,dc=nausch,dc=org" "uid=django" -W -D
"uid=django,ou=People,dc=nausch,dc=org"
```

```
Enter LDAP Password:
dn: uid=django,ou=People,dc=nausch,dc=org
uid: django
cn: Django
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword::
e2NyeXB0fSQ2JENna3VQVFplJDRiT2wvR2d0815lxQjRfuCKaU4yYVN5VndHUWE
2SVlubW40eGlGdzJkVjRsbWNK0815Yzlxd2tFYWJQdTZUL1BITWNXcWFLbW9KUnd6NlhwVTd3Vm0
x
shadowLastChange: 15272
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 500
gidNumber: 500
homeDirectory: /home/django
gecos: Django
```

Nach wie vor kann aber unser technischer User mit dem gleichlautenden Namen ***Technischeruser*** die Daten der beiden User **django** und auch **bigchief** abfragen.

```
 # ldapsearch -x -LLL -H ldap://ldap.dmz.nausch.org -b "dc=nausch,dc=org"
"uid=django" -W -D "cn=Technischeruser,dc=nausch,dc=org"
```

```
Enter LDAP Password:
dn: uid=django,ou=People,dc=nausch,dc=org
uid: django
```

```
cn: Django
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword::
e2NyeXB0fSQ2s1LV1ad1Eg31L3SaU2wvR2dSMUg4OWlxQjRtaU4yYVN5VndHUWE
2SVlubW40eGlGdzJkVjRsbWNKakRoYzlxd2tFYWJQdTZUL1BITWNXcWFLbW9KUnd6NlhwVTd3Vm0
x
shadowLastChange: 15272
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 500
gidNumber: 500
homeDirectory: /home/django
gecos: Django
```

```
 # ldapsearch -x -LLL -H ldap://ldap.dmz.nausch.org -b "dc=nausch,dc=org"
"uid=bigchief" -W -D "cn=Technischeruser,dc=nausch,dc=org"
```

```
Enter LDAP Password:
dn: uid=bigchief,ou=People,dc=nausch,dc=org
uid: bigchief
cn: BigChief
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword::
e2NyeXB0fSQ2JGo1cmxhSkd1JGlPVkt6T2pwOUdOSWpPNDFMYVlpTzFxTk9rc0h
rbFQ1eGpTc0oxMDhjZ21mVnBuZ0AsCHleKK3nOFBoOGV4bmZlbFJjaFBSOUlYOXRmalVxbm1JZWg
u
shadowLastChange: 15274
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 501
gidNumber: 501
homeDirectory: /home/bigchief
gecos: BigChief
```

Konfiguration des Clients

Die Konfiguration unseres Clients nehmen wir am einfachsten mit Hilfe des Programmes ***authconfig*** aus dem RPM-Paket **authconfig-gtk** vor. Hierzu rufen wir **authconfig** mit den nötigen Optionen für unsere (Test-)Umgebung auf.

- **disablesmartcard** SmartCard-Unterstützung deaktivieren
- **disablefingerprint** Fingerprintleser deaktivierung
- **disablemd5** MD5 Passworter abschalten
- **passalgo** Definition des Passworthash-Algoritmuses
- **enablemkhomedir** Homedirectory beim ersten Login eines neuen Users automatisch anlegen
- **enableldap** LDAP User Informationen aktivieren
- **enableldapauth** LDAP Authentifizierung aktivieren
- **ldapserver** LDAP Servername oder URI Definition
- **ldapbasedn** LDAP Basde DN Definition
- **update** Update der Konfigurationsdateien mit den gesetzten Werten.

Eine ausführliche Beschreibung der optionen erhält man übder die Manpage von authconfig oder beim Aufruf der Option *–help*.

```
 # authconfig --help
```

Wir Konfigurieren nun also unsere LDAP-Client-Authentifizierung wie folgt.

```
 # authconfig --disablesmartcard --disablefingerprint --disablemd5 --
passalgo=sha256 --enablemkhomedir --enableldap --enableldapauth --
ldapserver=ldap.dmz.nausch.org --ldapbasedn="dc=nausch,dc=org" --update
```

```
 nslcd starten:                                             [  OK  ]
 oddjobd starten:                                           [  OK  ]
```

Die einzelnen Konfigurationsdateien, die mit dem vorgenannten Programmaufruf angepasst wurden, werden wir uns im Detail betrachten, ggf. anpassen und mit Bearbeitungsvermerken versehen, damit wir später noch nachvollziehen können, welche Änderungen im Detail notwendig waren um die LDAP Client Authentifizierung aktiviert werden konnte.

Zur Dokumentation und ggf. spätere weitere Dokumentationsschritte versehen wir optional alle Änderungen mit einem Kommntar, ala: **# Django : Datum [optionaler Grund]**.

authconfig

In der Konfigurationsdatei ***/etc/sysconfig/authconfig*** setzen wir die beiden folgenden Werte von **no** auf **yes**:

- **USELDAP=yes** *(LDAP-Authentifizierung aktivieren.)*
- **FORCELEGACY=yes** *(CentOS 6 nutzt standardmäßig TLS für die LDAP-Authentifizierung. Mit diesem Schalter wird diese Voreinstellung deaktiviert und die die unverschlüsselte Kommunikation mit dem LDAP-Server erzwungen.)*

Zur Bearbeitung der Konfigurationsdatei nutzen wir wie so oft immer unseren Editor der Wahl **vim**.

```
 # vim /etc/sysconfig/authconfig
```

/etc/sysconfig/authconfig

```
USEMKHOMEDIR=no
```

```
USEPAMACCESS=no
CACHECREDENTIALS=yes
USESSSDAUTH=no
USESHADOW=yes
USEWINBIND=no
USEDB=no
FORCELEGACY=yes
USEFPRINTD=yes
FORCESMARTCARD=no
PASSWDALGORITHM=sha512
USELDAPAUTH=no
USEPASSWDQC=no
USELOCAUTHORIZE=yes
USECRACKLIB=yes
USEWINBINDAUTH=no
USESMARTCARD=no
USELDAP=yes
USENIS=no
USEKERBEROS=no
USESYSNETAUTH=no
USESMBAUTH=no
USESSSD=no
USEHESIOD=no
```

ldap.conf

In der Konfigurationsdatei ***/etc/openldap/ldap.conf*** tragen wir folgende Daten nach:

- **BASE dc=nausch, dc=org**
- **URI ldap://ldap.dmz.nausch.org**
- **TLS_CACERTDIR /etc/openldap/cacerts**

Zur Bearbeitung der Konfigurationsdatei nutzen wir wie so oft immer unseren Editor der Wahl **vim**.

```
# vim /etc/openldap/ldap.conf
```

/etc/openldap/ldap.conf

```
#
# LDAP Defaults
#

# See ldap.conf(5) for details
# This file should be world readable but not world writable.

#BASE   dc=example, dc=com
#URI    ldap://ldap.example.com ldap://ldap-master.example.com:666
```

```
#SIZELIMIT      12
#TIMELIMIT      15
#DEREF          never

# Django : 2011-10-28 LDAP Client Authentication
BASE    dc=nausch, dc=org
URI     ldap://ldap.dmz.nausch.org
TLS_CACERTDIR /etc/openldap/cacerts
```

pam_ldap.conf

In der Konfigurationsdatei ***/etc/pam_ldap.conf*** tragen wir folgende Daten nach:

- **binddn dc=nausch,dc=org**
- **bindpw Klaus-ist-der-groesste!**
- **uri ldap://ldap.dmz.nausch.org**
- **ssl no**
- **tls_cacertdir /etc/openldap/cacerts**
- **pam_password sha512**

Zur Bearbeitung der Konfigurationsdatei nutzen wir wie so oft immer unseren Editor der Wahl **vim**.

```
# vim /etc/pam_ldap.conf
```

/etc/pam_ldap.conf

```
# @(#)$Id: ldap.conf,v 1.38 2006/05/15 08:13:31 lukeh Exp $
#
# This is the configuration file for the LDAP nameservice
# switch library and the LDAP PAM module.
#
# The man page for this file is pam_ldap(5)
#
# PADL Software
# http://www.padl.com
#

# Your LDAP server. Must be resolvable without using LDAP.
# Multiple hosts may be specified, each separated by a
# space. How long nss_ldap takes to failover depends on
# whether your LDAP client library supports configurable
# network or connect timeouts (see bind_timelimit).

# Django : 2011-10-28 LDAP Client-Authentication
# default : host 127.0.0.1

# The distinguished name of the search base.
# Django : 2011-11-10 LDAP Client-Authentication
```

```
# base dc=example,dc=com
binddn dc=nausch,dc=org

# Another way to specify your LDAP server is to provide an
# uri with the server name. This allows to use
# Unix Domain Sockets to connect to a local LDAP Server.
#uri ldap://127.0.0.1/
#uri ldaps://127.0.0.1/
#uri ldapi://%2fvar%2frun%2fldapi_sock/
# Note: %2f encodes the '/' used as directory separator

# The LDAP version to use (defaults to 3
# if supported by client library)
#ldap_version 3

# The distinguished name to bind to the server with.
# Optional: default is to bind anonymously.
#binddn cn=proxyuser,dc=example,dc=com

# The credentials to bind with.
# Optional: default is no credential.
#bindpw secret
# Django : 2011-11-10 LDAP Client-Authentication
bindpw Klaus-ist-der-groesste!

# The distinguished name to bind to the server with
# if the effective user ID is root. Password is
# stored in /etc/ldap.secret (mode 600)
#rootbinddn cn=manager,dc=example,dc=com

# The port.
# Optional: default is 389.
#port 389

# The search scope.
#scope sub
#scope one
#scope base

# Search timelimit
#timelimit 30

# Bind/connect timelimit
#bind_timelimit 30

# Reconnect policy: hard (default) will retry connecting to
# the software with exponential backoff, soft will fail
# immediately.
#bind_policy hard

# Idle timelimit; client will close connections
```

```
# (nss_ldap only) if the server has not been contacted
# for the number of seconds specified below.
#idle_timelimit 3600

# Filter to AND with uid=%s
#pam_filter objectclass=account

# The user ID attribute (defaults to uid)
#pam_login_attribute uid

# Search the root DSE for the password policy (works
# with Netscape Directory Server)
#pam_lookup_policy yes

# Check the 'host' attribute for access control
# Default is no; if set to yes, and user has no
# value for the host attribute, and pam_ldap is
# configured for account management (authorization)
# then the user will not be allowed to login.
#pam_check_host_attr yes

# Check the 'authorizedService' attribute for access
# control
# Default is no; if set to yes, and the user has no
# value for the authorizedService attribute, and
# pam_ldap is configured for account management
# (authorization) then the user will not be allowed
# to login.
#pam_check_service_attr yes

# Group to enforce membership of
#pam_groupdn cn=PAM,ou=Groups,dc=example,dc=com

# Group member attribute
#pam_member_attribute uniquemember

# Specify a minium or maximum UID number allowed
#pam_min_uid 0
#pam_max_uid 0

# Template login attribute, default template user
# (can be overriden by value of former attribute
# in user's entry)
#pam_login_attribute userPrincipalName
#pam_template_login_attribute uid
#pam_template_login nobody

# HEADS UP: the pam_crypt, pam_nds_passwd,
# and pam_ad_passwd options are no
# longer supported.
#
```

```
# Do not hash the password at all; presume
# the directory server will do it, if
# necessary. This is the default.
#pam_password clear

# Hash password locally; required for University of
# Michigan LDAP server, and works with Netscape
# Directory Server if you're using the UNIX-Crypt
# hash mechanism and not using the NT Synchronization
# service.
#pam_password crypt

# Remove old password first, then update in
# cleartext. Necessary for use with Novell
# Directory Services (NDS)
#pam_password clear_remove_old
#pam_password nds

# RACF is an alias for the above. For use with
# IBM RACF
#pam_password racf

# Update Active Directory password, by
# creating Unicode password and updating
# unicodePwd attribute.
#pam_password ad

# Use the OpenLDAP password change
# extended operation to update the password.
#pam_password exop

# Redirect users to a URL or somesuch on password
# changes.
#pam_password_prohibit_message Please visit http://internal to change
your password.

# RFC2307bis naming contexts
# Syntax:
# nss_base_XXX          base?scope?filter
# where scope is {base,one,sub}
# and filter is a filter to be &'d with the
# default filter.
# You can omit the suffix eg:
# nss_base_passwd       ou=People,
# to append the default base DN but this
# may incur a small performance impact.
#nss_base_passwd        ou=People,dc=example,dc=com?one
#nss_base_shadow        ou=People,dc=example,dc=com?one
#nss_base_group         ou=Group,dc=example,dc=com?one
#nss_base_hosts         ou=Hosts,dc=example,dc=com?one
#nss_base_services      ou=Services,dc=example,dc=com?one
```

```
#nss_base_networks      ou=Networks,dc=example,dc=com?one
#nss_base_protocols     ou=Protocols,dc=example,dc=com?one
#nss_base_rpc           ou=Rpc,dc=example,dc=com?one
#nss_base_ethers        ou=Ethers,dc=example,dc=com?one
#nss_base_netmasks      ou=Networks,dc=example,dc=com?ne
#nss_base_bootparams    ou=Ethers,dc=example,dc=com?one
#nss_base_aliases       ou=Aliases,dc=example,dc=com?one
#nss_base_netgroup      ou=Netgroup,dc=example,dc=com?one

# attribute/objectclass mapping
# Syntax:
#nss_map_attribute      rfc2307attribute        mapped_attribute
#nss_map_objectclass    rfc2307objectclass      mapped_objectclass

# configure --enable-nds is no longer supported.
# NDS mappings
#nss_map_attribute uniqueMember member

# Services for UNIX 3.5 mappings
#nss_map_objectclass posixAccount User
#nss_map_objectclass shadowAccount User
#nss_map_attribute uid msSFU30Name
#nss_map_attribute uniqueMember msSFU30PosixMember
#nss_map_attribute userPassword msSFU30Password
#nss_map_attribute homeDirectory msSFU30HomeDirectory
#nss_map_attribute homeDirectory msSFUHomeDirectory
#nss_map_objectclass posixGroup Group
#pam_login_attribute msSFU30Name
#pam_filter objectclass=User
#pam_password ad

# configure --enable-mssfu-schema is no longer supported.
# Services for UNIX 2.0 mappings
#nss_map_objectclass posixAccount User
#nss_map_objectclass shadowAccount user
#nss_map_attribute uid msSFUName
#nss_map_attribute uniqueMember posixMember
#nss_map_attribute userPassword msSFUPassword
#nss_map_attribute homeDirectory msSFUHomeDirectory
#nss_map_attribute shadowLastChange pwdLastSet
#nss_map_objectclass posixGroup Group
#nss_map_attribute cn msSFUName
#pam_login_attribute msSFUName
#pam_filter objectclass=User
#pam_password ad

# RFC 2307 (AD) mappings
#nss_map_objectclass posixAccount user
#nss_map_objectclass shadowAccount user
#nss_map_attribute uid sAMAccountName
#nss_map_attribute homeDirectory unixHomeDirectory
```

```
#nss_map_attribute shadowLastChange pwdLastSet
#nss_map_objectclass posixGroup group
#nss_map_attribute uniqueMember member
#pam_login_attribute sAMAccountName
#pam_filter objectclass=User
#pam_password ad

# configure --enable-authpassword is no longer supported
# AuthPassword mappings
#nss_map_attribute userPassword authPassword

# AIX SecureWay mappings
#nss_map_objectclass posixAccount aixAccount
#nss_base_passwd ou=aixaccount,?one
#nss_map_attribute uid userName
#nss_map_attribute gidNumber gid
#nss_map_attribute uidNumber uid
#nss_map_attribute userPassword passwordChar
#nss_map_objectclass posixGroup aixAccessGroup
#nss_base_group ou=aixgroup,?one
#nss_map_attribute cn groupName
#nss_map_attribute uniqueMember member
#pam_login_attribute userName
#pam_filter objectclass=aixAccount
#pam_password clear

# Netscape SDK LDAPS
#ssl on

# Netscape SDK SSL options
#sslpath /etc/ssl/certs

# OpenLDAP SSL mechanism
# start_tls mechanism uses the normal LDAP port, LDAPS typically 636
#ssl start_tls
#ssl on

# OpenLDAP SSL options
# Require and verify server certificate (yes/no)
# Default is to use libldap's default behavior, which can be configured
in
# /etc/openldap/ldap.conf using the TLS_REQCERT setting.  The default
for
# OpenLDAP 2.0 and earlier is "no", for 2.1 and later is "yes".
#tls_checkpeer yes

# CA certificates for server certificate verification
# At least one of these are required if tls_checkpeer is "yes"
#tls_cacertfile /etc/ssl/ca.cert
#tls_cacertdir /etc/ssl/certs
```

```
# Seed the PRNG if /dev/urandom is not provided
#tls_randfile /var/run/egd-pool

# SSL cipher suite
# See man ciphers for syntax
#tls_ciphers TLSv1

# Client certificate and key
# Use these, if your server requires client authentication.
#tls_cert
#tls_key

# Disable SASL security layers. This is needed for AD.
#sasl_secprops maxssf=0

# Override the default Kerberos ticket cache location.
#krb5_ccname FILE:/etc/.ldapcache

# SASL mechanism for PAM authentication - use is experimental
# at present and does not support password policy control
#pam_sasl_mech DIGEST-MD5

# Django : 2011-10-28 LDAP Client-Authentication, automatisch
eingetragen mit Hilfe von authconfig
uri ldap://ldap.dmz.nausch.org
ssl no
tls_cacertdir /etc/openldap/cacerts
pam_password sha256
```

nslcd.conf

In der Konfigurationsdatei ***/etc/nslcd.conf*** tragen wir folgende Daten nach:

- **uri ldap://ldap.dmz.nausch.org**
- **binddn dc=nausch,dc=org**
- **bindpw Klaus-ist-der-groesste!**
- **ssl no**

Zur Bearbeitung der Konfigurationsdatei nutzen wir wie so oft immer unseren Editor der Wahl **vim**.

```
# vim /etc/nslcd.conf
```

/etc/nslcd.conf

```
# This is the configuration file for the LDAP nameservice
# switch library's nslcd daemon. It configures the mapping
# between NSS names (see /etc/nsswitch.conf) and LDAP
# information in the directory.
```

```
# See the manual page nslcd.conf(5) for more information.

# The uri pointing to the LDAP server to use for name lookups.
# Multiple entries may be specified. The address that is used
# here should be resolvable without using LDAP (obviously).
#uri ldap://127.0.0.1/
#uri ldaps://127.0.0.1/
#uri ldapi://%2fvar%2frun%2fldapi_sock/
# Note: %2f encodes the '/' used as directory separator
# uri ldap://127.0.0.1/

# The LDAP version to use (defaults to 3
# if supported by client library)
#ldap_version 3

# The distinguished name of the search base.
# base dc=example,dc=com

# The distinguished name to bind to the server with.
# Optional: default is to bind anonymously.
#binddn cn=proxyuser,dc=example,dc=com

# The credentials to bind with.
# Optional: default is no credentials.
# Note that if you set a bindpw you should check the permissions of
this file.
#bindpw secret
# Django : LDAP Client Authentifizierung mit technischem User
bindpw Klaus-ist-der-groesste!

# The distinguished name to perform password modifications by root by.
#rootpwmoddn cn=admin,dc=example,dc=com

# The default search scope.
#scope sub
#scope one
#scope base

# Customize certain database lookups.
#base   group  ou=Groups,dc=example,dc=com
#base   passwd ou=People,dc=example,dc=com
#base   shadow ou=People,dc=example,dc=com
#scope  group  onelevel
#scope  hosts  sub

# Bind/connect timelimit.
#bind_timelimit 30

# Search timelimit.
#timelimit 30
```

```
# Idle timelimit. nslcd will close connections if the
# server has not been contacted for the number of seconds.
#idle_timelimit 3600

# Use StartTLS without verifying the server certificate.
#ssl start_tls
#tls_reqcert never

# CA certificates for server certificate verification
#tls_cacertdir /etc/ssl/certs
#tls_cacertfile /etc/ssl/ca.cert

# Seed the PRNG if /dev/urandom is not provided
#tls_randfile /var/run/egd-pool

# SSL cipher suite
# See man ciphers for syntax
#tls_ciphers TLSv1

# Client certificate and key
# Use these, if your server requires client authentication.
#tls_cert
#tls_key

# NDS mappings
#map group uniqueMember member

# Mappings for Services for UNIX 3.5
#filter passwd (objectClass=User)
#map    passwd uid              msSFU30Name
#map    passwd userPassword     msSFU30Password
#map    passwd homeDirectory    msSFU30HomeDirectory
#map    passwd homeDirectory    msSFUHomeDirectory
#filter shadow (objectClass=User)
#map    shadow uid              msSFU30Name
#map    shadow userPassword     msSFU30Password
#filter group  (objectClass=Group)
#map    group  uniqueMember     msSFU30PosixMember

# Mappings for Services for UNIX 2.0
#filter passwd (objectClass=User)
#map    passwd uid              msSFUName
#map    passwd userPassword     msSFUPassword
#map    passwd homeDirectory    msSFUHomeDirectory
#map    passwd gecos            msSFUName
#filter shadow (objectClass=User)
#map    shadow uid              msSFUName
#map    shadow userPassword     msSFUPassword
#map    shadow shadowLastChange pwdLastSet
#filter group  (objectClass=Group)
#map    group  uniqueMember     posixMember
```

```
# Mappings for Active Directory
#pagesize 1000
#referrals off
#filter passwd
(&(objectClass=user)(!(objectClass=computer))(uidNumber=*)(unixHomeDire
ctory=*))
#map    passwd uid              sAMAccountName
#map    passwd homeDirectory    unixHomeDirectory
#map    passwd gecos            displayName
#filter shadow
(&(objectClass=user)(!(objectClass=computer))(uidNumber=*)(unixHomeDire
ctory=*))
#map    shadow uid              sAMAccountName
#map    shadow shadowLastChange pwdLastSet
#filter group  (objectClass=group)
#map    group  uniqueMember     member

# Mappings for AIX SecureWay
#filter passwd (objectClass=aixAccount)
#map    passwd uid              userName
#map    passwd userPassword     passwordChar
#map    passwd uidNumber        uid
#map    passwd gidNumber        gid
#filter group  (objectClass=aixAccessGroup)
#map    group  cn               groupName
#map    group  uniqueMember     member
#map    group  gidNumber        gid
uid nslcd
gid ldap
# This comment prevents repeated auto-migration of settings.
# Django : 2011-10-28 LDAP Client Authentication, angefügt durch den
Aufruf von authconfig
uri ldap://ldap.dmz.nausch.org/
binddn cn=Technischeruser,dc=nausch,dc=org
ssl no
tls_cacertdir /etc/openldap/cacerts
```

nsswitch.conf

In der Konfigurationsdatei ***/etc/pam.d/system-auth*** tragen wir folgende Daten nach:

- **passwd: files ldap**
- **shadow: files ldap**
- **group: files ldap**
- **netgroup: ldap**
- **automount: files ldap**

Zur Bearbeitung der Konfigurationsdatei nutzen wir wie so oft immer unseren Editor der Wahl **vim**.

```
# vim /etc/nsswitch.conf
```

/etc/nsswitch.conf

```
#
# /etc/nsswitch.conf
#
# An example Name Service Switch config file. This file should be
# sorted with the most-used services at the beginning.
#
# The entry '[NOTFOUND=return]' means that the search for an
# entry should stop if the search in the previous entry turned
# up nothing. Note that if the search failed due to some other reason
# (like no NIS server responding) then the search continues with the
# next entry.
#
# Valid entries include:
#
#       nisplus                 Use NIS+ (NIS version 3)
#       nis                     Use NIS (NIS version 2), also called YP
#       dns                     Use DNS (Domain Name Service)
#       files                   Use the local files
#       db                      Use the local database (.db) files
#       compat                  Use NIS on compat mode
#       hesiod                  Use Hesiod for user lookups
#       [NOTFOUND=return]       Stop searching if not found so far
#

# To use db, put the "db" in front of "files" for entries you want to
be
# looked up first in the databases
#
# Example:
#passwd:    db files nisplus nis
#shadow:    db files nisplus nis
#group:     db files nisplus nis

# Django : 2011-10-28 LDAP Client Authentication
# default
# passwd:     files
# shadow:     files
# group:      files
passwd:     files ldap
shadow:     files ldap
group:      files ldap

#hosts:     db files nisplus nis dns
hosts:      files dns
```

```
# Example - obey only what nisplus tells us...
#services:   nisplus [NOTFOUND=return] files
#networks:   nisplus [NOTFOUND=return] files
#protocols:  nisplus [NOTFOUND=return] files
#rpc:        nisplus [NOTFOUND=return] files
#ethers:     nisplus [NOTFOUND=return] files
#netmasks:   nisplus [NOTFOUND=return] files

bootparams: nisplus [NOTFOUND=return] files

ethers:     files
netmasks:   files
networks:   files
protocols:  files
rpc:        files
services:   files

# Django : 2011-10-28 LDAP Client Authentication
# default
# netgroup:   nisplus
netgroup:   ldap

publickey:  nisplus

# Django : 2011-10-28 LDAP Client Authentication
# default
# automount:  files nisplus
automount:  files ldap
aliases:    files nisplus
```

system-auth

Durch den Aufruf des Programmes authconfig wurden die folgenden **pam.d**-Konfigurationsdateien angepasst:

- **/etc/pam.d/fingerprint-auth**
- **/etc/pam.d/password-auth**
- **/etc/pam.d/smartcard-auth**
- **/etc/pam.d/smtp**
- **/etc/pam.d/system-auth**

Zur Bearbeitung der Konfigurationsdatei nutzen wir wie so oft immer unseren Editor der Wahl **vim**.

```
# vim /etc/pam.d/fingerprint-auth
```

/etc/pam.d/fingerprint-auth

```
#%PAM-1.0
```

```
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        sufficient    pam_fprintd.so
auth        required      pam_deny.so

account     required      pam_unix.so broken_shadow
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 500 quiet
account     [default=bad success=ok user_unknown=ignore] pam_ldap.so
account     required      pam_permit.so

password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
session     optional      pam_oddjob_mkhomedir.so
session     [success=1 default=ignore] pam_succeed_if.so service in
crond quiet use_uid
session     required      pam_unix.so
session     optional      pam_ldap.so
```

```
# vim /etc/pam.d/password-auth
```

/etc/pam.d/password-auth

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 500 quiet
auth        sufficient    pam_ldap.so use_first_pass
auth        required      pam_deny.so

account     required      pam_unix.so broken_shadow
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 500 quiet
account     [default=bad success=ok user_unknown=ignore] pam_ldap.so
account     required      pam_permit.so

password    requisite     pam_cracklib.so try_first_pass retry=3 type=
password    sufficient    pam_unix.so sha256 shadow nullok
try_first_pass use_authtok
password    sufficient    pam_ldap.so use_authtok
password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
```

```
session     optional      pam_oddjob_mkhomedir.so
session     [success=1 default=ignore] pam_succeed_if.so service in
crond quiet use_uid
session     required      pam_unix.so
session     optional      pam_ldap.so
```

```
# vim /etc/pam.d/smartcard-auth
```

/etc/pam.d/smartcard-auth

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        [success=done ignore=ignore default=die] pam_pkcs11.so
wait_for_card card_only
auth        required      pam_deny.so

account     required      pam_unix.so broken_shadow
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 500 quiet
account     [default=bad success=ok user_unknown=ignore] pam_ldap.so
account     required      pam_permit.so

password    required      pam_pkcs11.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
session     optional      pam_oddjob_mkhomedir.so
session     [success=1 default=ignore] pam_succeed_if.so service in
crond quiet use_uid
session     required      pam_unix.so
session     optional      pam_ldap.so
```

```
# vim /etc/pam.d/smtp
```

/etc/pam.d/smtp

```
#%PAM-1.0
auth       include      password-auth
account    include      password-auth
```

```
# vim /etc/pam.d/system-auth
```

/etc/pam.d/system-auth

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        sufficient    pam_fprintd.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 500 quiet
auth        sufficient    pam_ldap.so use_first_pass
auth        required      pam_deny.so

account     required      pam_unix.so broken_shadow
account     sufficient    pam_localuser.so
account     sufficient    pam_succeed_if.so uid < 500 quiet
account     [default=bad success=ok user_unknown=ignore] pam_ldap.so
account     required      pam_permit.so

password    requisite     pam_cracklib.so try_first_pass retry=3 type=
password    sufficient    pam_unix.so sha256 shadow nullok
try_first_pass use_authtok
password    sufficient    pam_ldap.so use_authtok
password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
session     optional      pam_oddjob_mkhomedir.so
session     [success=1 default=ignore] pam_succeed_if.so service in
crond quiet use_uid
session     required      pam_unix.so
session     optional      pam_ldap.so
```

automatischer Systemstart des nslc-Dämon

Damit nun beim nächsten Start des Systems der notwendige **naming services LDAP client daemon** kurz **nslcd** mit gestartet wird, versetzen wir das Startscript in den Modus „*on*“.

```
# chkconfig nslcd on
```

Den Status überprüfen wir bei Bedarf mittels:

```
# chkconfig --list | grep nslcd
```

```
nslcd           0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

Zum Abschluss unserer Konfiguration starten wir nun unseren CentOS 6 Client einmal durch.

```
# reboot
```

Test

LDAP Abfrage

Zur Abfrage eines LDAP-Users können wir folgenden Aufruf verwenden:

```
 $ ldapsearch -x -LLL -H ldap://ldap.dmz.nausch.org -b "dc=nausch,dc=org"
"uid=django" -W -D "cn=Technischeruser,dc=nausch,dc=org"
```

```
Enter LDAP Password:
dn: uid=django,ou=People,dc=nausch,dc=org
uid: django
cn: Django
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword::
e2NyeXB0fSQ2JENna3VQVFplsdfsT2wvR2dSMUg4OWlxQjRtaU4yYVN5VndHUWE
2SVlubW40eGlGdzJkVjRsbWNKakRoYzlxd2tFYWJQdTZUL1BITWNXcW215Sdsads6NlhwVTd3Vm0
x
shadowLastChange: 15272
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 500
gidNumber: 500
homeDirectory: /home/django
gecos: Django
```

LDAP-Abfrage mit dem User *Django* aber mit **falschem** Passwort:

```
 $ ldapsearch -x -LLL -H ldap://ldap.dmz.nausch.org -b "dc=nausch,dc=org"
"uid=django" -W -D "uid=django,ou=People,dc=nausch,dc=org"
```

```
Enter LDAP Password:
ldap_bind: Invalid credentials (49)
```

LDAP-Abfrage mit dem User *Django* aber mit **richtigem** Passwort:

```
 $ ldapsearch -x -LLL -H ldap://ldap.dmz.nausch.org -b "dc=nausch,dc=org"
"uid=django" -W -D "uid=django,ou=People,dc=nausch,dc=org"
```

```
Enter LDAP Password:
dn: uid=django,ou=People,dc=nausch,dc=org
uid: django
cn: Django
objectClass: account
```

```
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword::
e2NyeXB0fSQ2JENna3VQVFplJDss2wvR2dSMUg4OWlxQjRtaU4yYVN5VndHUWE
2SVlum31nScH3fFiSt31nV0lLd3pPzlxd2t564JQdTZUL1BITWNXcWFLbW9KUnd6NlhwVTd3Vm0x
shadowLastChange: 15272
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 500
gidNumber: 500
homeDirectory: /home/django
gecos: Django
```

Clienttest

Die erfolgreiche Konfiguration unseres Rechners überprüfen wir so:

1. Mit **getent** lassen wir uns die Informationen eines Users anzeigen, der sowohl in der /etc/shadow wie auch im zentralen LDAP-Verzeichnisdienst hinterlegt ist. Wenn alles gut gelaufen ist, werden uns zwei Einträge präsentiert.

   ```
   $ getent passwd | grep django
   django:x:500:500::/home/django:/bin/bash
   django:x:500:500:Django:/home/django:/bin/bash
   ```

2. Als nächstes wählen wir einen Nutzer der nur im LDAP-Verzeichnisdienst einen Account hat, nicht aber auf der lokalen Maschine.

   ```
   $ getent passwd | grep bigchief
   bigchief:x:501:501:BigChief:/home/bigchief:/bin/bash
   ```

3. Dann melden wir uns nun an unserem Client als ein Benutzer an, der lokal auf der Maschine nicht existiert, werden wir beim Login nach dem Passwort gefragt, welches gegen den zentralen OpenLDAP-Server verifiziert wird. Ist das Passwort richtig wird auch gleich das zugehörige Nutzer-Homeverzeichnis angelegt.

   ```
   [django@vml010008 ~]$ su - bigchief
   Password:
   Creating directory '/home/bigchief'.
   [bigchief@vml010008 ~]$
   ```

Links

- **Zurück zum Kapitel >>OpenLDAP Server unter CentOS 6.x<<**
- **Zurück zu >>Projekte und Themenkapitel<<**
- **Zurück zur Startseite**

[1)] [2)] [3)] [4)]

Directory **I**nformation **T**ree