
4 Zugriffskontrolle mit ACLs

In diesem Kapitel lernen Sie

- ▶ wie man mit ACLs Zugriffsrechte auf einzelne Ressourcen vergibt.

Unter einer *ACL* (*Access Control List*) versteht man eine Liste mit Zugriffsrechten. Anhand dieser Liste entscheidet der OpenLDAP-Server, welchen Zugriff ein Benutzer auf die einzelnen Ressourcen, wie zum Beispiel ein Verzeichnisobjekt, bekommt.

Dadurch erhält man einen feingranularen Zugriffsschutz für Ressourcen. Die ACL listet Operationen, die für eine Person oder Gruppe erlaubt werden, auf. Die ACLs werden in der Konfigurationsdatei `slapd.conf` für den OpenLDAP-Server eingetragen. Hilfe zu LDAP-ACLs befindet sich in der Man-Page: **man slapd.access**.

Der Aufbau einer ACL-Direktive: Syntax:

```
access to <what>
    [by <who> <access> <control>]
```

Der `<what>`-Teil definiert, auf welchen Teil im LDAP-Baum zugegriffen werden kann. Der `<who>`-Teil definiert, auf wen die ACL zutrifft, und der `<access>` Teil definiert, was erlaubt wird. Optional kann eine `<control>`-Anweisung folgen.



Beispiel:

```
access to *
    by * real
```

4.1 Die Teile der ACL-Direktive

4.1.1 Der `<what>`-Teil

Der `<what>`-Teil bestimmt, auf welchen Teil im LDAP-Baum zugegriffen werden kann. Man kann hierzu den „Distinguish Name“ (`dn`), Eigenschaften (`attribute`) und/oder Filter benutzen.

Zusätzlich ist es möglich, mit dem Alias `'*`' alles einzuschließen. Mögliche Syntax für den `<what>`-Teil der ACL:

```
* | [ dn[.<target style>]=<regex>] [filter=<ldapfilter>] [attrs=<attrlist>]
```

Zugriffskontrolle mit ACLs

Distinguish Name (dn) nutzen Bei einem Distinguish Name (dn) kann ein target-style mitgegeben werden, welches die Suchart angibt. Es stehen folgende target-styles zur Verfügung:

regex (Standard) Bedeutet, dass der Wert ein regulärer Ausdruck ist.

base Bezeichnet das Objekt selbst.

one Bezeichnet das Objekt und alle Objekte *eine* Ebene darunter.

subtree Bezeichnet *alle* Objekte unterhalb des Objektes und das Objekt selbst.

children Bezeichnet alle Unterobjekte, aber *nicht* das Objekt selbst.

Beispiel: Das erste Objekt im Baum (base = er selbst):



```
dn.base=""
```

Alle Objekte:

```
dn='.*'  
*
```

Alle Objekte unter ou=people,dc=firma,dc=de:

```
dn='.*,ou=people,dc=firma,dc=de'  
dn.subtree='ou=people,dc=firma,dc=de'
```

Alle Objekte unterhalb von dc=firma,dc=de, aber das Objekt selbst nicht eingeschlossen:

```
dn.children='dc=firma,dc=de'
```

Attribute nutzen Zugriff auf bestimmte Attribute. Nur das Attribut uid:

```
attrs=uid
```

Mehrere Attribute:

```
attrs=uidNumber,gidNumber
```

Filter nutzen Nutzen des LDAP-Filters siehe RFC 2254. *Filter* unter LDAP sind äußerst mächtig in ihrer Funktionalität. Hier werden nur einige einfache Beispiele angegeben.

Beispiel: Alle mit der Objektklasse inetorgperson:



```
filter="objectclass=inetorgperson"
```

Alle mit dem sn (= Nachname) maier:

```
filter='sn=maier'
```

Alle mit dem Nachnamen maier oder mayer:

```
filter='(| (sn=maier) (sn=mayer))'
```

Alle Nachnamen bis mayer:

```
filter='sn<=mayer'
```

4.1.2 Der <who>-Teil

Der <who>-Teil der ACL bestimmt, auf wen sich die Zugriffsregelung beziehen soll. Dazu sollte man sich Gedanken machen, wie der Zugriff auf den LDAP-Baum verläuft. Hier gibt es verschiedene Möglichkeiten sich am Verzeichnisbaum zu authentifizieren und damit zu binden.

```
<who> ::= [* | anonymous | users | self |
          dn[.<subject style>]=<regex>
          [dnattr=<attrname> ]
          [group[/<objectclass>[/<attrname>][.<basic style>]]=<regex> ]
          [peername[.<basic style>]=<regex>]
          [sockname[.<basic style>]=<regex>]
          [domain[.<basic style>]=<regex>]
          [sockurl[.<basic style>]=<regex>]
          [set=<setspec>]
          [aci=<attrname>]
          <subject style> ::= regex | exact | base | one | subtree | children
          <basic style> ::= regex | exact
```

Zugriffskontrolle mit ACLs

Vordefinierte Schlüsselbegriffe Es gibt eine Anzahl von vordefinierten Schlüsselbegriffen:

*****: Alle Benutzer.

anonymous: Unautorisierte Benutzer (diesen gewährt man in der Regel nur eingeschränkten Zugriff, besonders auf Ressourcen wie z.B. Passwörter. (hier darf man sich nur anonym authentifizieren)).

users: Benutzer, die sich über ein Objekt authentifiziert haben.

self: Wie `user`, aber in Bezug auf das eigene Objekt

Des weiteren kann man mit Distinguished Name oder Gruppen-Objekten arbeiten:

dn: Ein Regulärer Ausdruck.

group: Objekt in der Gruppe.

Auch Domains, URLs und Sockets sind möglich; hierbei ist es jedoch wichtig, das Format festzulegen (basic style). Erlaubt sind `exact` oder `regex`.

Arbeiten mit Gruppenobjekten Man kann unter LDAP so genannte Gruppenobjekte definieren. Über diese können dann andere Objekte zugeordnet werden. Dies geschieht, indem die dn-Namen der anderen Objekte aufgelistet werden.

Beispiel: Gruppenobjekt der Objektklasse `groupOfUniqueNames`:



```
cn=ldapadmins,ou=people,dc=firma,dc=de
objectclass: groupOfUniqueNames
objectclass: top
cn: ldapadmins
uniqueMember: joe
uniqueMember: jack
uniqueMember: jane
```

Diese Gruppe kann man dann in einer ACL, für einen `<who>`-Eintrag verwenden:

```
group[/<objectclass>[/<attrname>]]
    [.<style>]=<pattern>
```

Beispiel: Die Mitglieder (über mehrere dn-Einträge im Objekt der Klasse `groupOfUniqueNames` definiert) können nun via 'group=' verwendet werden.



Beispiel:

```
group='cn=ldapadmins,ou=people,dc=firma,dc=de'
```

4.1.3 Der <access>-Teil

Um die Zugriffsberechtigungen zu definieren, hat man im <access>-Teil der ACL die Möglichkeit, mit Access-Leveln oder mit den einzelnen Rechten direkt zu arbeiten.

```
<access> ::= <level>|<priv>
<priv> ::= =|+|-w|r|s|c|x+
<level> ::= none | auth | compare | search | read | write
```

Es gibt folgende Rechte:

```
w = write
r = read
s = search
c = compare
x = auth
```

Diese Rechte können nun absolut (mit =) oder relativ (mit + oder -) gesetzt werden.

Wenn man mit Leveln arbeitet, werden verschiedene Zugriffsebenen definiert. Höhere Ebenen beinhalten automatisch die Zugriffsberechtigung der darunter liegenden Ebenen:

| | |
|---------|--------------|
| none | Keine Rechte |
| auth | x |
| compare | xc |
| search | scx |
| read | rscx |
| write | wrscx |



Hinweis: Bei allen ACLs, die gesetzt werden, ist die Anbindung nicht zu vergessen. So muss ein anonymer Client mindestens die Möglichkeit bekommen, sich zu authentifizieren.

4.1.4 Der <control>-Teil

Hiermit kann das weitere Vorgehen bei einer passenden Regel definiert werden:

Zugriffskontrolle mit ACLs

`<control> ::= [stop | continue | break]`

Default ist `stop`. Wenn diese Regel zutrifft, werden andere Einträge nicht mehr beachtet.

`Continue` heißt in einer Access-Regel, dass die anderen `<who>`-Einträge zu beachten sind.

`Break` geht sogar noch weiter und ermöglicht, dass auch andere `<access>`-Einträge Beachtung finden. Dabei summieren sich alle Rechte inkrementell auf. Das heißt, dass `+r +w = rw`, aber `+r =x` nur noch `x` bedeutet.

Weitere Möglichkeiten Man kann den Zugriff unter anderem auch unter der Bedingung gewähren, dass ein bestimmter *Security Strength Factor* eingehalten wird. Dies geschieht über die Parameter `ssf=<n>`, `transport_ssf=<n>`, `tls_ssf=<n>`, und `sasl_ssf=<n>`. Dabei ist `n` ein Integerwert:

0: Kein Schutz

1: Integritätsschutz

56: Verschlüsselung mit mindestens 56 bit-Schlüssel (DES)

Beispiel: SASL muss mindestens Datenintegrität sicherstellen:



```
sasl_ssf=1
```

Beispiel: SASL muss mindestens mit 56-bit verschlüsseln:



```
sasl_ssf=56
```

Evaluierung der ACLs Die ACLs werden der Reihe nach abgearbeitet, der erste zutreffende `<what>`-Eintrag wird genutzt. Bei diesem Eintrag werden dann der Reihe nach alle `<who>`-Einträge überprüft. Auch hier wird der erste Eintrag verwendet. Werden keine eigenen `<control>`-Definitionen gemacht, ist die Auswertung beendet.

Hinweis: Deshalb sollte man in diesem Fall auch unbedingt zuerst die ACL-Regeln für die Administratoren festlegen. Sonst könnte eine Regel, die in der Reihenfolge vor den Administratoren-Regeln steht, die Rechte der Administratoren einschränken. Schließlich wird diese andere Regel zuerst gefunden, ausgewertet und angewandt. Dass danach noch eine weitere Regel, die explizit auf die Administratoren zutrifft, vorkommt, wird ja nicht mehr überprüft.



4.2 Arbeiten mit Regulären Ausdrücken

(siehe `man 7 regex`)

Im folgenden sollen die wichtigsten *regulären Ausdrücke* erklärt werden:

Suchbegriffe:

. Beliebiges Zeichen.

^ Beginn der Zeile.

\$ Ende der Zeile.

[0-9] ist ein Zeichen innerhalb von 0-9.

| Trennen mehrerer Möglichkeiten.

Mengenangaben eines Suchbegriffs: Man kann ein Suchatom mit Klammern abgrenzen und dann die Anzahl der Treffer durch ein so genannte Mengenangabe festlegen:

* Beliebig oft.

+ Ein- oder mehrmals.

? Null- oder einmal.

oder man gibt den Integerwert an (in geschweiften Klammern).



Beispiel: Alle Einträge für dn finden, die einer ou=People angehören:

```
dn='.*,ou=People,.*'
```

Alle Einträge deren dn mit uid beginnt und die in einer ou=People sind:

```
dn='uid=.*,ou=People,.*'
```


Alle Benutzer herausfinden, deren uid mit user (mindestens einmal) beginnt:

```
dn='uid=(user\)\+.*,ou=People,.*'
```

Alle Benutzer herausfinden, die eine Zahl im uid haben:

```
dn='uid=.*[0-9]\+.*,ou=People,.*'
```

Man kann auch gefundene Atome weiterverwenden. Dies geht beispielsweise mit \$zahl, z.B. \$1 für das erste gefundene Atom:

Beispiel: Unterhalb eines Benutzers liegt ein privates Adressbuch, auf das nur er selbst zugreifen darf: 

```
access to dn="ou=addr,uid=(.*),dc=firma,dc=de"
  by dn="uid=$1,dc=firma,dc=de" write by * none
```

4.3 Komplette Beispiele

Einige Anregungen für den Einsatz von ACLs lassen sich aus den folgenden Praxis-Beispielen gewinnen. Angepasst an Ihre Bedürfnisse, können Sie diese Beispiele auch für eigenen Anwendungen übernehmen.

Beispiel aus dem SuSE E-Mailserver:

```
defaultaccess read

# Private AddressBook
access to dn="ou=addr,uid=(.*),dc=firma,dc=de"
  by dn="uid=$1,dc=firma,dc=de" write by * none

# Hide skyrixGreenConfig
access to attr=skyrixGreenConfig
  by self write
  by * none

# To let PAM authenticate
access to attr=userpassword
  by self write
  by anonymous auth
  by * none

# let addressadmins write below root
access to dn="o=AddressBook,dc=firma,dc=de" attr=children
  by group="cn=AddressAdmins,o=AddressBook,dc=firma,dc=de" write
  by * read

access to dn=".*,o=AddressBook,dc=firma,dc=de"
  filter="objectclass=inetorgperson"
  by group="cn=AddressAdmins,o=AddressBook,dc=firma,dc=de" write
  by * read

# protect root dn of addressbook
```

```
access to dn="o=AddressBook,dc=firma,dc=de"
  by * read

# protect addressadmins group
access to dn="cn=AddressAdmins,o=AddressBook,dc=firma,dc=de"
  by * read

# give write access to the owner of the following attributes
access to attr=c,cn,telephoneNumber,facsimileTelephoneNumber,pager,
title,givenname,sn,l,description,mail,streetAddress,postalCode,st,
homePhone,ou,initials,mobile,labeledURI,preferredLanguage
  by self write
  by * read
```

Beispiel aus der Defaultkonfiguration:

```
access to dn.base="" by * read
access to *
  by self write
  by users read
  by anonymous auth
```

Beispiel für einen einfachen Authentifizierungsserver:

```
access to attr=userPassword
  by self write
  by anonymous auth
  by dn="cn=Manager,dc=muc,dc=suse,dc=de" write
  by * none
access to *
  by anonymous auth
  by self write
  by dn="cn=Manager,dc=muc,dc=suse,dc=de" write
  by * read
```

Beispiel für ein Adressbuch:

```
# Private AddressBook
access to dn="ou=addr,uid=(.)*,dc=firma,dc=de"
  by dn="uid=$1,dc=firma,dc=de" write by * none

# Hide skyrixGreenConfig
```

Zugriffskontrolle mit ACLs

```
access to attr=skyrixGreenConfig
  by self write
  by * none

# To let PAM authenticate
access to attr=userpassword
  by self write
  by anonymous auth
  by * none

# let addressadmins write below root
access to dn="o=AddressBook,dc=firma,dc=de" attr=children
  by group="cn=AddressAdmins,o=AddressBook,dc=firma,dc=de" write
  by * read

access to dn=".*,o=AddressBook,dc=firma,dc=de"
  filter="objectclass=inetorgperson"
  by group="cn=AddressAdmins,o=AddressBook,dc=firma,dc=de" write
  by * read

# protect root dn of addressbook
access to dn="o=AddressBook,dc=firma,dc=de"
  by * read

# protect addressadmins group
access to dn="cn=AddressAdmins,o=AddressBook,dc=firma,dc=de"
  by * read

# give write access to the owner of the following attributes
access to attr=c,cn,telephoneNumber,facsimileTelephoneNumber,pager,
title,givenname,sn,l,description,mail,streetAddress,postalCode,st,
homePhone,ou,initials,mobile,labeledURI,preferredLanguage
  by self write
  by * read
```

Weitere Beispiele:

```
access to attr=userPassword
  by self write
  by anonymous auth
  by dn="cn=Admin,dc=example,dc=com" write
  by * none

access to *
  by self write
```