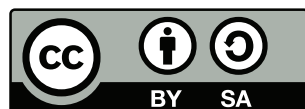
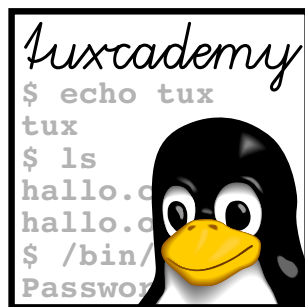


LDAP und OpenLDAP

Installation und Betrieb unter Linux



Das tuxcademy-Projekt bietet hochwertige frei verfügbare Schulungsunterlagen zu Linux- und Open-Source-Themen – zum Selbststudium, für Schule, Hochschule, Weiterbildung und Beruf.
Besuchen Sie <https://www.tuxcademy.org/>! Für Fragen und Anregungen stehen wir Ihnen gerne zur Verfügung.

LDAP und OpenLDAP Installation und Betrieb unter Linux

Revision: ldap:8b571a6627d574f9:2014-04-03

ldap:de5f1ca787ab6b36:2013-02-13 1–8

nadm:34ccb7a5ca5eb94a:2014-04-03 B

ldap:FZ3ovI7i1s4TGYrC9zjqga

© 2015 Linup Front GmbH Darmstadt, Germany

© 2016 tuxcademy (Anselm Lingnau) Darmstadt, Germany

<http://www.tuxcademy.org> · info@tuxcademy.org

Linux-Pinguin »Tux« © Larry Ewing (CC-BY-Lizenz)

Alle in dieser Dokumentation enthaltenen Darstellungen und Informationen wurden nach bestem Wissen erstellt und mit Sorgfalt getestet. Trotzdem sind Fehler nicht völlig auszuschließen. Das tuxcademy-Projekt haftet nach den gesetzlichen Bestimmungen bei Schadensersatzansprüchen, die auf Vorsatz oder grober Fahrlässigkeit beruhen, und, außer bei Vorsatz, nur begrenzt auf den vorhersehbaren, typischerweise eintretenden Schaden. Die Haftung wegen schuldhafter Verletzung des Lebens, des Körpers oder der Gesundheit sowie die zwingende Haftung nach dem Produkthaftungsgesetz bleiben unberührt. Eine Haftung über das Vorgenannte hinaus ist ausgeschlossen.

Die Wiedergabe von Warenbezeichnungen, Gebrauchsnamen, Handelsnamen und Ähnlichem in dieser Dokumentation berechtigt auch ohne deren besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne des Warenzeichen- und Markenschutzrechts frei seien und daher beliebig verwendet werden dürften. Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen Dritter.



Diese Dokumentation steht unter der »Creative Commons-BY-SA 4.0 International«-Lizenz. Sie dürfen sie vervielfältigen, verbreiten und öffentlich zugänglich machen, solange die folgenden Bedingungen erfüllt sind:

Namensnennung Sie müssen darauf hinweisen, dass es sich bei dieser Dokumentation um ein Produkt des tuxcademy-Projekts handelt.

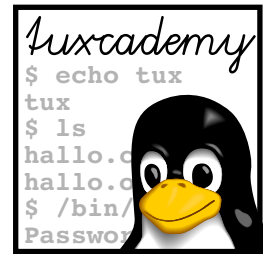
Weitergabe unter gleichen Bedingungen Sie dürfen die Dokumentation bearbeiten, abwandeln, erweitern, übersetzen oder in sonstiger Weise verändern oder darauf aufbauen, solange Sie Ihre Beiträge unter derselben Lizenz zur Verfügung stellen wie das Original.

Mehr Informationen und den rechtsverbindlichen Lizenzvertrag finden Sie unter <http://creativecommons.org/licenses/by-sa/4.0/>

Autor: Anselm Lingnau

Technische Redaktion: Anselm Lingnau (anselm@tuxcademy.org)

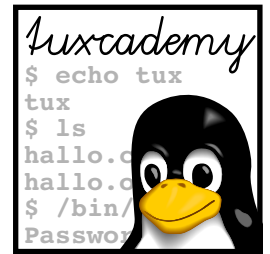
Gesetzt in Palatino, Optima und DejaVu Sans Mono



Inhalt

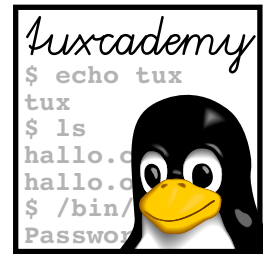
1	LDAP-Einführung	1
1.1	Was ist LDAP?	2
1.2	Das LDAP-Datenmodell	3
1.3	Das LDAP-Namensmodell	7
1.4	Das funktionale Modell	10
1.5	Das Sicherheitsmodell	14
1.6	LDAP-URLs	15
1.7	LDIF	16
2	LDAP-Verzeichnisse verwalten	23
2.1	LDAP-Werkzeuge für die Kommandozeile	24
2.2	Daten suchen mit ldapsearch	24
2.3	Daten hinzufügen, ändern und löschen	27
3	Der OpenLDAP-Server	31
3.1	Installation von OpenLDAP	32
3.2	Konfiguration von OpenLDAP	32
3.2.1	Allgemeines.	32
3.2.2	»Traditionelle« Konfiguration mit slapd.conf	34
3.2.3	»Moderne« Konfiguration über LDAP	38
3.3	Ein Konfigurationsbeispiel	41
3.4	OpenLDAP-Betrieb	42
3.5	Weiterleitung	43
4	OpenLDAP-Sicherheit	47
4.1	Zugriffsregeln für Verzeichnisdaten	48
4.1.1	Zugriffsrechte	48
4.1.2	Direktiven	49
4.1.3	Auswertung von Zugriffsregeln	50
4.1.4	Beispiele für Zugriffsregeln	50
4.1.5	Tipps und Tricks	52
4.2	OpenLDAP und SASL	53
4.2.1	SASL-Grundlagen	53
4.2.2	OpenLDAP mit SASL-Authentisierung	55
4.2.3	Testen der Konfiguration	58
4.2.4	Proxy-Authentisierung.	58
4.3	OpenLDAP und TLS.	60
4.3.1	Grundlagen.	60
4.3.2	Server-Zertifikate.	61
4.3.3	Client-Zertifikate	62
5	Replikation	65
5.1	Grundlagen	66
5.2	Einfache Replikation.	67
5.3	Delta-Replikation	70
5.4	Multi-Master-Replikation	72

6 LDAP für Benutzerdaten	75
6.1 Linux-Benutzerdaten im LDAP-Verzeichnis	76
6.2 LDAP und der Name-Service-Switch	79
6.3 LDAP und PAM	80
6.4 Kennwortverwaltung für LDAP-basierte Benutzer	83
7 Apache und LDAP	85
7.1 Überblick	86
7.2 Konfiguration von mod_authnz_ldap	86
7.2.1 Phasen der Zugriffskontrolle.	86
7.2.2 Die Authentisierungsphase	87
7.2.3 Die Autorisierungsphase	88
7.3 Beispiele	90
7.4 Das Modul mod_ldap	92
8 LDAP, Postfix und Dovecot	95
8.1 Überblick	96
8.2 Die Datenbankstruktur	97
8.3 Postfix	100
8.3.1 Allgemeines.	100
8.3.2 Mail-Zustellung	101
8.4 Dovecot	102
8.4.1 Allgemeines.	102
8.4.2 Authentisierung und Benutzerdaten	103
8.5 Extras	106
8.5.1 SMTP-AUTH	106
8.5.2 Mail-Weiterleitung	106
A Musterlösungen	109
B X.509-Crashkurs	113
B.1 Einleitung: Kryptografie, Zertifikate und X.509	113
B.2 Eine Zertifizierungsstelle generieren	115
B.3 Server-Zertifikate generieren.	118
Index	121



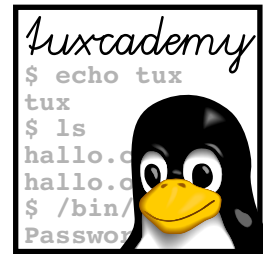
Tabellenverzeichnis

1.1	Syntaxdefinitionen für Attribute (Auswahl)	4
1.2	Häufig verwendete Vergleichsregeln	5
1.3	Typische Schlüsselwörter in Distinguished Names	8
2.1	Ausgabeformate für ldapsearch	25
3.1	loglevel-Werte und dazugehörige Inhalte	36
4.1	Zugriffsrechte in LDAP	48



Abbildungsverzeichnis

1.1	Syntax einer Attributtypdefinition gemäß [RFC4512]	3
1.2	Einige Attributtypinformationen aus [RFC4519]	4
1.3	Syntax einer Objektklassendefinition gemäß [RFC4512]	6
1.4	Definition der Objektklassen person und organizationalPerson gemäß [RFC4519]	6
1.5	Die LDAP-Klassenhierarchie (Ausschnitt, nur strukturelle Klassen)	7
1.6	<i>Directory Information Tree</i> (DIT)	8
1.7	Syntax für Suchfilter gemäß [RFC4515]	12
1.8	Eine beispielhafte LDIF-Datei	16
3.1	DIT für die »moderne« Konfigurationsmethode	38
6.1	Objektklassen für Linux-Benutzer gemäß RFC 2307	76
6.2	LDIF-Datei mit Informationen über den Benutzer hugo	77
6.3	LDIF-Datei für einen Benutzer mit Zugangs- und persönlichen Informationen	78
6.4	Beispiel für eine PAM-Konfiguration für login	81
8.1	Attribute und Objektklassen für Mail	98
B.1	Konfigurationsdatei für eine OpenSSL-basierte CA	117



Vorwort

Dieser Kurs richtet sich an Systemadministratoren, die LDAP und den OpenLDAP-Server zum Aufbau eines leistungsfähigen, interoperablen Verzeichnisdienstes auf der Basis offener Standards verwenden wollen. Nach einer Einführung in die Grundlagen von LDAP wird der OpenLDAP-Server vorgestellt und seine Installation und Konfiguration erläutert. Dazu gehört auch die Replikation von Verzeichnissen und das Einbinden eines OpenLDAP-Servers in ein übergreifendes, „globales“ Verzeichnis. Es folgt eine Einführung in die Sicherheitskonzepte von LDAP und OpenLDAP, die Definition von Zugriffsregeln auf Verzeichnisdaten sowie die Konfiguration von starken Authentisierungsmechanismen und dem Zugang zu OpenLDAP-Servern über TLS. Ferner wird erklärt, wie man Benutzer von Unix- und Windows-Rechnern mit Hilfe von LDAP-Verzeichnissen authentisieren kann, und es werden einige andere beispielhafte Anwendungen von LDAP und OpenLDAP gezeigt.

Voraussetzung zum Verständnis des Kurses sind solide Linux- und Netzwerkkenntnisse; ein grundlegender Einblick in Themen wie Datenbanken und System-sicherheit ist von Vorteil. Programmierkenntnisse sind nicht erforderlich.

Diese Schulungsunterlage soll den Kurs möglichst effektiv unterstützen, indem das Kursmaterial in geschlossener, ausführlicher Form zum Mitlesen, Nach- oder Vorarbeiten präsentiert wird. Das Material ist in Kapitel eingeteilt, die jeweils für sich genommen einen Teilaspekt umfassend beschreiben; am Anfang jedes Kapitels sind dessen Lernziele und Voraussetzungen kurz zusammengefasst, am Ende finden sich eine Zusammenfassung und (wo sinnvoll) Angaben zu weiterführender Literatur oder WWW-Seiten mit mehr Informationen.

Kapitel

Lernziele

Voraussetzungen



Zusätzliches Material oder weitere Hintergrundinformationen sind durch das »Glühbirnen«-Sinnbild am Absatzanfang gekennzeichnet. Zuweilen benutzen diese Absätze Aspekte, die eigentlich erst später in der Schulungsunterlage erklärt werden, und bringen das eigentlich gerade Vorgestellte so in einen breiteren Kontext; solche »Glühbirnen«-Absätze sind möglicherweise erst beim zweiten Durcharbeiten der Schulungsunterlage auf dem Wege der Kursnachbereitung voll verständlich.



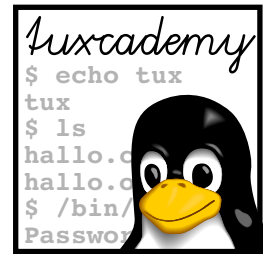
Absätze mit dem »Warnschild« weisen auf mögliche Probleme oder »gefährliche Stellen« hin, bei denen besondere Vorsicht angebracht ist. Achten Sie auf die scharfen Kurven!



Die meisten Kapitel enthalten auch Übungsaufgaben, die mit dem »Bleistift«-Sinnbild am Absatzanfang gekennzeichnet sind. Die Aufgaben sind nummeriert und Musterlösungen für die wichtigsten befinden sich hinten in dieser Schulungsunterlage. Bei jeder Aufgabe ist in eckigen Klammern der Schwierigkeitsgrad angegeben. Aufgaben, die mit einem Ausrufungszeichen (»!«) gekennzeichnet sind, sind besonders empfehlenswert.

Übungsaufgaben

Auszüge aus Konfigurationsdateien, Kommandobeispiele und Beispiele für die Ausgabe des Rechners erscheinen in Schreibmaschinenschrift. Bei mehrzeiligen Dialogen zwischen Benutzer und Rechner werden die Benutzereingaben in **fetter Schreibmaschinenschrift** angegeben, um Missverständnisse zu vermeiden. Wenn



1

LDAP-Einführung

Inhalt

1.1	Was ist LDAP?	2
1.2	Das LDAP-Datenmodell	3
1.3	Das LDAP-Namensmodell	7
1.4	Das funktionale Modell	10
1.5	Das Sicherheitsmodell	14
1.6	LDAP-URLs	15
1.7	LDIF	16

Lernziele

- Ein grundsätzliches Verständnis von LDAP haben
- Das Datenaustauschformat LDIF kennen und anwenden können

Vorkenntnisse

- Grundbegriffe von Verzeichnisdiensten und DNS
- Grundbegriffe von Datenbanken (wünschenswert)

1.1 Was ist LDAP?

»LDAP« ist die Abkürzung für *Lightweight Directory Access Protocol*, also »leichtgewichtiges Protokoll zum Zugriff auf Verzeichnisdienste«. Es ist ein TCP/IP-basiertes Protokoll, das vor allem dafür gedacht ist, X.500-basierte Verzeichnisdienste anzusprechen. X.500 ist Teil der ISO/OSI-Netzwerkstruktur (einem »Konkurrenzprodukt« zu TCP/IP), das sich wegen seiner Komplexität nicht flächendeckend durchsetzen konnte. Insbesondere ist das **Directory Access Protocol** (DAP) von seinem Datenmodell und seinen Operationen her zu »schwergewichtig«, als dass man einen kompletten DAP-Client auf einem kleineren Computer implementieren könnte. Daher LDAP.



Zumindest in den 1980er Jahren, als X.500 neu war, war das der Fall. Heutzutage haben gängige Mobiltelefone technische Daten, die man seinerzeit höchstens in der mittleren Datentechnik gefunden hätte, so dass dieses konkrete Problem wahrscheinlich keine unüberwindliche Hürde mehr wäre.

LDAP verwendet im wesentlichen dasselbe Daten- und Namensmodell wie X.500, aber setzt direkt auf TCP/IP auf und vermeidet einige der ausgefalleneren Aspekte des DAP. Die fehlenden Teile sind nicht so wichtig und können durch in LDAP existierende Funktionalität ersetzt werden. Aktuell (Januar 2010) ist LDAP Version 3, kurz »LDAPv3«, gemäß [RFC4511]. LDAP erlaubt den Zugriff auf, das Durchsuchen und Ändern von Verzeichnissen.

Ein **Verzeichnisdienst** enthält (typischerweise) Informationen über Benutzer von Rechnersystemen, Dienste, Programme, Drucker und andere Ressourcen, die in einem Netz zugänglich sind. Viele Systeme bieten spezialisierte Verzeichnisdienste an, zum Beispiel Novells **NDS** oder Microsofts **Active Directory**. Oft sind diese Verzeichnisdienste »Insellösungen«, die an bestimmte Systeme gebunden sind. LDAP ist ein Ansatz dafür, eine gemeinsame *lingua franca* für solche Verzeichnisdienste zu finden. Natürlich sind Verzeichnisdienste nicht auf Computeranwendungen beschränkt, sondern Sie können auch ein Telefonbuch oder einen Bibliothekskatalog als Verzeichnisdienst ansehen. Verzeichnisdienste erlauben ihren Benutzern, Ressourcen aufzufinden, die sie zur Erledigung ihrer Aufgaben benötigen. In einem Benutzerverzeichnis könnten Sie zum Beispiel die E-Mail-Adresse oder die Faxnummer einer Person finden, in einem Bibliothekskatalog die ISBN oder den Standort eines Buchs; in einem Rechnernetz könnte das Verzeichnis den nächsten Farb-Laserdrucker auffinden. Verzeichnisse können nach verschiedenen Kriterien sortiert sein, zum Beispiel nach dem *Namen* der enthaltenen Einträge (wie das Telefonbuch) oder nach gewissen *Eigenschaften*, die die Einträge haben (wie das Branchentelefonbuch, die »gelben Seiten«).

Verzeichnisdienste sind normalerweise für Leseoperationen optimiert. Daten werden zum Beispiel weithin repliziert, um Zuverlässigkeit, Verfügbarkeit und Antwortzeiten zu verbessern. Dafür toleriert man auch die Tatsache, dass nach Änderungen manchmal für eine gewisse Zeit nicht alle replizierten Kopien der Datenbank konsistent sind (dieselben Daten enthalten). Ein einfacher, nicht replizierter Verzeichnisdienst für Benutzerdaten ist **finger** [RFC1288]; LDAP und X.500 sind »globale« Verzeichnisdienste, die in einem wesentlich größeren Kontext arbeiten und Daten replizieren können. In diesem Sinne ähneln sie dem World-Wide Web – jeder, der einen LDAP-Client hat, kann auf das globale X.500-Verzeichnis so zugreifen wie der Inhaber eines WWW-Browsers auf das World-Wide Web.

LDAP benutzt einen Client-Server-Ansatz. Ein LDAP-**Verzeichnisbaum** (engl. *directory information tree*, DIT) wird in einer **Backend-Datenbank** abgelegt, die auf einem oder mehreren LDAP-Servern enthalten sein kann. Ein LDAP-Client verbindet sich mit einem Server und fragt ihn etwas; der Server schickt die Antwort oder einen Verweis auf eine Stelle, wo der Client mehr Informationen bekommen kann (typischerweise einen anderen LDAP-Server). Der Client sieht dasselbe Verzeichnis, egal mit welchem Server er Kontakt aufnimmt; in einem globalen Verzeichnisdienst wie LDAP gibt jeder Server bei einer Anfrage nach demselben Eintrag dieselben Daten zurück (wenn überhaupt).

```

<AttributeTypeDescription> ← ( ( <wsp>
    <numericoid>
    [ <sp> NAME <sp> <qdescrs> ]           Name des Attributs
    [ <sp> DESC <sp> <qdstring> ]         Beschreibung
    [ <sp> OBSOLETE ]
    [ <sp> SUP <sp> <oid> ]               Abgeleitet von diesem Attribut
    [ <sp> EQUALITY <sp> <oid> ]         Regel für »Gleichheit«
    [ <sp> ORDERING <sp> <oid> ]        Regel zum Sortieren
    [ <sp> SUBSTR <sp> <oid> ]          Regel für »Teilkettenvergleich«
    [ <sp> SYNTAX <sp> <noidlen> ]
    [ <sp> SINGLE-VALUE ]               Sonst: mehrere Werte möglich
    [ <sp> COLLECTIVE ]                 Sonst: nicht kollektiv
    [ <sp> NO-USER-MODIFICATION ]       Sonst: modifizierbar
    [ <sp> USAGE <sp> <usage> ]
    <extensions>
  <wsp> )
<usage> ← userApplications
         |directoryOperation
         |distributedOperation
         |dSAOperation
<wsp> ← ein Leerzeichen, mehrere oder auch gar keins
<sp> ← ein Leerzeichen oder mehrere
<numericoid> ← ein numerischer Objektbezeichner (OID)
<noidlen> ← <numericoid> { <länge> }
<qdescrs> ← ein oder mehrere Namen
<qdstring> ← eine Zeichenkette in Anführungszeichen

```

Bild 1.1: Syntax einer Attributtypdefinition gemäß [RFC4512]

Strenggenommen ist LDAP nicht selbst ein Verzeichnisdienst, sondern lediglich ein Protokoll zum Zugriff auf Verzeichnisdienste. Es gibt zum Beispiel die Möglichkeit, über LDAP auf die `/etc/passwd`-Datei eines Unix-Rechners zuzugreifen (auch wenn Sie das natürlich als »mit Kanonen auf Spatzen schießen« ansehen könnten). Der eigentliche Verzeichnisdienst ist die Backend-Datenbank, die selber nicht zwingend etwas mit LDAP zu tun haben muss; LDAP ist möglicherweise nur einer von mehreren unterstützten Zugangswegen. Trotzdem gibt es LDAP-Server wie das im folgenden diskutierte OpenLDAP-Paket, die direkt selbst eine Backend-Datenbank enthalten können; in diesem Sinne kann man auch von LDAP als »Verzeichnisdienst« sprechen.

1.2 Das LDAP-Datenmodell

Welche Daten sind über LDAP zugänglich? Das ist nicht genau festgelegt. Das Datenmodell von X.500 basiert auf **Einträgen**, die Informationen über bestimmte Objekte (etwa Personen – sorry!) enthalten. Einträge bestehen aus **Attributen**, die einen **Typ** und einen oder mehrere **Werte** haben. Jedes Attribut hat eine **Syntax**, die festschreibt, welche Arten von Werten im Attribut erlaubt sind und wie diese Werte sich bei Verzeichnisoperationen benehmen. Solche Syntaxen gibt es unter anderem für IA5- bzw. ASCII-Zeichenketten, JPEG-Bilder, URLs oder PGP-Schlüssel. Die Syntax für eine Telefonnummer könnte zum Beispiel vorschreiben, dass eine Telefonnummer eine Folge von Ziffern, Leerzeichen und Strichen (»-«) ist, die lexikographisch sortiert werden kann, wobei bei Vergleichen alle Leerzeichen und Striche ignoriert werden.

Wie man Attribute definiert, steht in [RFC4512]. Bild 1.1 zeigt die Syntax einer »offiziellen« Attributtypdefinition, so wie der RFC sie vorgibt. Die Begriffe in spit-

Eintrag
Attribute
Typ
Werte
Syntax

```
( 2.5.4.41 NAME 'name'
  EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

( 2.5.4.3 NAME ( 'cn' 'commonName' ) SUP name )
```

Bild 1.2: Einige Attributtypinformationen aus [RFC4519]

Tabelle 1.1: Syntaxdefinitionen für Attribute (Auswahl)

Syntax	OID	Beschreibung
boolean	1.3.6.1.4.1.1466.115.121.1.7	Wahrheitswert
distinguishedName	1.3.6.1.4.1.1466.115.121.1.12	DN (siehe Abschnitt 1.3)
directoryString	1.3.6.1.4.1.1466.115.121.1.15	UTF-8-Zeichenkette
IA5String	1.3.6.1.4.1.1466.115.121.1.26	ASCII-Zeichenkette
Integer	1.3.6.1.4.1.1466.115.121.1.27	Ganzzahl
Name and Optional UID	1.3.6.1.4.1.1466.115.121.1.34	DN mit UID
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	Numerische Zeichenkette
OID	1.3.6.1.4.1.1466.115.121.1.38	
Octet String	1.3.6.1.4.1.1466.115.121.1.40	Beliebige Bytes
Printable String	1.3.6.1.4.1.1466.115.121.1.44	Druckbare Zeichenkette

zen Klammern sind »Platzhalter« für tatsächlich einzusetzende Werte, die zum Teil noch genauer definiert sind; Teile der Definition, die in eckigen Klammern stehen, können auch wegfallen und ihre Werte werden dann durch Standardvorgaben ersetzt.

numerischer Objektbezeichner

Einige Begriffe in Bild 1.1 verdienen eine nähere Betrachtung: Jedem Attributtypnamen wird ein **numerischer Objektbezeichner**, kurz OID, zugeordnet, der ihn (hoffentlich) weltweit eindeutig identifiziert. Tatsächlich sind die OIDs ausschlaggebend; die textuellen Namen aus dem NAME-Feld der Attributtypdefinition dienen nur der Vereinfachung. OIDs sind durch Punkte getrennte Folgen von Dezimalzahlen, zum Beispiel steht 2.5.4.3 für den Attributtyp `commonName`.

Internet Assigned
Numbers Authority



Prinzipiell ist es nicht erlaubt, numerische OIDs einfach frei zu erfinden; die anerkannte Methode besteht darin, bei der **Internet Assigned Numbers Authority** (IANA) um die Zuteilung eines OID-»Präfixes« zu bitten. Hierzu füllen Sie das Web-Formular auf <http://www.iana.org/cgi-bin/enterprise.pl> aus und warten ungefähr eine Woche auf das Ergebnis. (Im Formular ist die Rede von »MIB/SNMP private enterprise numbers«, davon sollten Sie sich nicht verwirren lassen: Die LDAP-OIDs und die MIB-OIDs teilen sich denselben Namensraum. Das »private enterprise« bedeutet, dass ein Teilbereich der möglichen OIDs für die Verwendung von Unternehmen der Privatwirtschaft vorgesehen ist, und aus diesem Teilbereich werden dann OID-Präfixe vergeben.) Das Resultat ist gemeinhin etwas wie 1.3.6.1.4.1.x, und Sie dürfen dann beliebige weitere Zahlen und Punkte anhängen. Wenn Sie es nicht so lange aushalten, können Sie für private Experimente OIDs benutzen, die mit 1.1 anfangen; der Teilbereich des Namensraums unter 1.1 gilt als »tot«. In Umlauf bringen sollten Sie solche OIDs allerdings nicht. Alternativ dazu können Sie auch OIDs unter 1.3.6.1.4.1.4203.666 benutzen, welcher Bereich von den OpenLDAP-Entwicklern für Experimente reserviert wurde. Auch solche OIDs gehören natürlich nicht an die Öffentlichkeit. – Mehr über OIDs können Sie zum Beispiel bei [AOID] nachlesen.

Die SYNTAX-Angabe in der Attributtypdefinition bestimmt die tatsächliche Syntax des Attributs. Leider muss diese durch einen numerischen OID angegeben

Tabelle 1.2: Häufig verwendete Vergleichsregeln

Name	Typ	Beschreibung
booleanMatch	EQUALITY	Boolescher Vergleich
octetStringMatch	EQUALITY	Zeichenkettenvergleich
objectIdentifizierMatch	EQUALITY	OID-Vergleich
distinguishedNameMatch	EQUALITY	DN-Vergleich
numericStringMatch	EQUALITY	numerischer Vergleich
numericStringOrderingMatch	ORDERING	
numericStringSubstringsMatch	SUBSTR	
caseIgnoreMatch	EQUALITY	Groß/Klein und Freiplatz ignorieren
caseIgnoreOrderingMatch	ORDERING	
caseIgnoreSubstringsMatch	SUBSTR	
caseExactMatch	EQUALITY	Groß/Klein ignorieren, Freiplatz nicht
caseExactOrderingMatch	ORDERING	
caseExactSubstringsMatch	SUBSTR	

werden, der mehr oder weniger vom Himmel fällt (siehe Tabelle 1.1). Die SYNTAX-Angabe enthält auch eine Länge (in geschweiften Klammern), die allerdings eher eine Zierfunktion hat.

Die Vergleichsregeln für die EQUALITY-, ORDERING- und SUBSTR-Angaben dürfen Sie glücklicherweise mit Namen statt numerischen OIDs bestimmen. Hier zeigt Tabelle 1.2 eine kleine Auswahl der Möglichkeiten.

Bild 1.2 zeigt zwei Beispiele für Attributtypdefinitionen. Der Typ name ist für Personen- oder Objektnamen gedacht, wobei Groß- und Kleinschreibung nicht beachtet werden. commonName oder cn sind zwei gleichberechtigte Namen für den Typ 2.5.4.3, der von name, genauer gesagt 2.5.4.41, abgeleitet ist. In beiden Fällen ist der Wert (genauer gesagt möglicherweise die Werte – SINGLE-VALUE ist nicht spezifiziert) ein directoryString, also eine gemäß UTF-8 codierte Zeichenkette, der von Benutzern verändert werden kann (NO-USER-MODIFICATION ist nicht angegeben). Das Attribut ist für die Verwendung in Benutzerprogrammen gedacht (USAGE ist nicht angegeben, also wird der Standardwert userApplications angenommen).

Die Attribute, die ein bestimmter Eintrag im LDAP-Verzeichnisbaum haben kann, werden durch seine **Objektklasse** definiert. Die LDAP-Standards definieren eine Reihe von Objektklassen: Objektklasse

- Gruppen im Verzeichnis, unter anderem unsortierte Listen individueller Objekte oder Objektgruppen
- Ortsangaben, etwa Ländernamen und -beschreibungen,
- Organisationen,
- Personen

Ein Eintrag kann mehr als einer Objektklasse angehören. Zum Beispiel wird der Eintrag für eine Person durch die Objektklasse person beschrieben, aber kann auch Attribute der Objektklassen inetOrgPerson, groupOfNames und organization enthalten. Die Gesamtheit der für einen Eintrag zulässigen Attribute wird durch das **Schema**, die Objektklassenstruktur des Servers festgelegt. Strukturierungsvorschläge für X.500-Verzeichnisbäume werden im [RFC1617] diskutiert. Schema

Jede Objektklasse kann manche Attribute als *notwendig* und andere als *optional* kennzeichnen. Als Beispiel zeigt Bild 1.4 die Definition der Objektklassen person und organizationalPerson aus [RFC4519]. Hier muss jeder Eintrag vom Typ person die Attribute sn («surname») und cn («common name») haben, da sie in einer MUST-Definition auftauchen; die in der MAY-Definition genannten Attribute userPassword usw. müssen nicht notwendigerweise im Eintrag enthalten sein. Die Definition von organizationalPerson definiert eine von person abgeleitete Objektklasse (dies

```

(ObjectClassDescription) ← ( <wsp>
    <numericoid>
    [ <sp> NAME <sp> <qdescrs> ]           Name des Attributs
    [ <sp> DESC <sp> <qdstring> ]         Beschreibung
    [ <sp> OBSOLETE <whsp> ]
    [ <sp> SUP <sp> <oids> ]             Abgeleitet von diesen Klassen
    [ <sp> {ABSTRACT | STRUCTURAL | AUXILIARY} ]   Sonst: STRUCTURAL
    [ <sp> MUST <sp> <oids> ]
    [ <sp> MAY <sp> <oids> ]
    <extensions> <wsp> )
<oids> ← ein oder mehrere Namen oder OIDs, getrennt durch $ (!)

```

Bild 1.3: Syntax einer Objektklassendefinition gemäß [RFC4512]

```

( 2.5.6.6 NAME 'person' SUP top STRUCTURAL
  MUST ( sn $ cn )
  MAY ( userPassword $ telephoneNumber $ seeAlso $ description ) )

( 2.5.6.7 NAME 'organizationalPerson' SUP person STRUCTURAL
  MAY ( title $ x121Address $ registeredAddress $ destinationIndicator $
    preferredDeliveryMethod $ telexNumber $ teletexTerminalIdentifier $
    telephoneNumber $ internationalISDNNumber $
    facsimileTelephoneNumber $ street $ postOfficeBox $ postalCode $
    postalAddress $ physicalDeliveryOfficeName $ ou $ st $ l ) )

```

Bild 1.4: Definition der Objektklassen person und organizationalPerson gemäß [RFC4519]

wird durch das »SUP person« ausgedrückt) und fügt einen Posten weitere optionale Attribute hinzu.

Jeder Verzeichniseintrag hat ein notwendiges Attribut `objectClass`, das seine Objektklasse(n) angibt. Normalerweise sind das zumindest `top` – die »Wurzel« der Objektklassenhierarchie – und die tatsächliche Objektklasse, etwa `person`. Gehört ein Objekt der Objektklasse *A* an, die von *B* abgeleitet ist, wobei *B* wiederum von `top` abgeleitet ist, sollte sein `objectClass`-Attribut die Werte *A*, *B* und `top` haben. Alternativ zu `top` ist auch `alias` möglich, wenn ein Verzeichniseintrag ein »Spitzname« für einen anderen ist (siehe Abschnitt 1.3).

Objektklassen sind entweder `STRUCTURAL`, `AUXILIARY` oder `ABSTRACT`. Klassen, die als `STRUCTURAL` gelten, können prinzipiell die einzige Objektklasse eines Eintrags sein, während `AUXILIARY`-Klassen nur als zusätzliche Objektklassen in Frage kommen. Sie dienen meistens dazu, einen Eintrag mit weiteren Attributen zu versehen (Anhänger der »objektorientierten Programmierung« kennen das unter dem Stichwort *Mixin-Klasse*). `ABSTRACT`-Klassen können ebenfalls nicht die einzige Objektklasse eines Eintrags sein, sondern sie haben nur einen Platz in der Vererbungshierarchie und dienen dazu, dass andere Objektklassen von ihnen abgeleitet werden (auch dieses Konzept ist Anhängern der objektorientierten Programmierung wohlvertraut).

Jeder LDAP-Server muss laut Norm die (`ABSTRACT`-)Objektklassen `top` und `subschema` kennen. Ferner ist es aus Kompatibilitätsgründen *sehr* empfehlenswert, die in [RFC4519] genannten Klassen, Attribute und Definitionen zu unterstützen. Ansonsten ist es aber auch ohne weiteres möglich, eigene Objektklassen und Attribute zu definieren, die die mitgelieferten Klassen erweitern oder von ihnen völlig unabhängig sind. Ein LDAPv3-Server muss in der Lage sein, Informationen über das von ihm verwendete Schema zu liefern.

extensibleObject



Die Objektklasse `extensibleObject` erlaubt die Speicherung *beliebiger* Attribu-

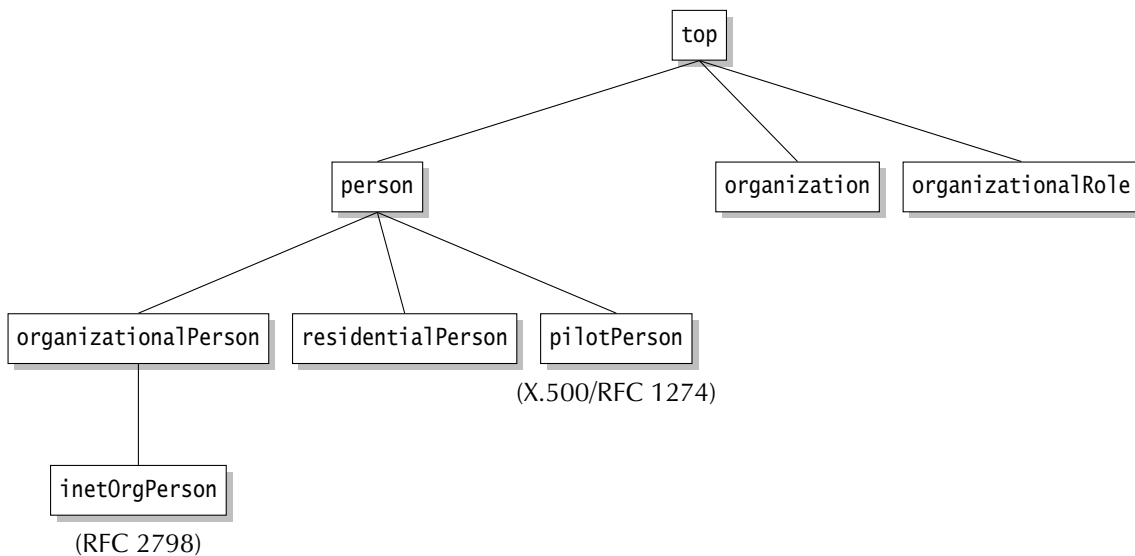


Bild 1.5: Die LDAP-Klassenhierarchie (Ausschnitt, nur strukturelle Klassen)

te. Dies ist möglicherweise einfacher als die Definition einer neuen Objektklasse, aber unterläuft natürlich die Fähigkeit von LDAP-Software, die Konsistenz der Datenbank zu prüfen. Aus diesem Grund sollten Sie dies mit Vorsicht genießen.

Übungen



1.1 [!1] Welche Objektklassen in [RFC4519] sind direkt von person abgeleitet? Gibt es in [RFC4519] Objektklassen, die von einer solchen Objektklasse abgeleitet sind?



1.2 [2] Schlagen Sie notwendige und optionale Attribute für die Objektklasse buch in einem Bibliothekskatalog vor.



1.3 [3] Was unterscheidet einen LDAP-basierten Verzeichnisdienst von einer relationalen Datenbank? Ist eine relationale Datenbank als Backend für einen LDAP-basierten Verzeichnisdienst geeignet? Wenn ja, warum? Wenn nein, warum nicht?

1.3 Das LDAP-Namensmodell

Einträge in einem LDAP-basierten Verzeichnis werden in einer Hierarchie angeordnet, dem *Directory Information Tree* oder DIT (Bild 1.6). Maßgeblich für diese Einordnung sind die **Distinguished Names** der Einträge. Der Distinguished Name (DN) eines Eintrags ist ein eindeutiger Name für den Eintrag.

Distinguished Name

Distinguished Names sind Folgen von »relativen« Distinguished Names (RDN), wobei jeder RDN einem »Ast« zwischen zwei Einträgen im DIT entspricht. Ein einfacher RDN hat die Form »<Attribut>=<Wert>«; ein RDN kann aber auch eine Kombination mehrerer Attribute sein, die dann mit »+« aneinandergereiht werden. Wichtig ist, dass der RDN *eindeutig* unter seinen »Geschwistern« im DIT ist; zusätzliche Attribute können zum Beispiel verwendet werden, um zwischen zwei Personen gleichen Namens in derselben Organisation zu unterscheiden: Die RDNs

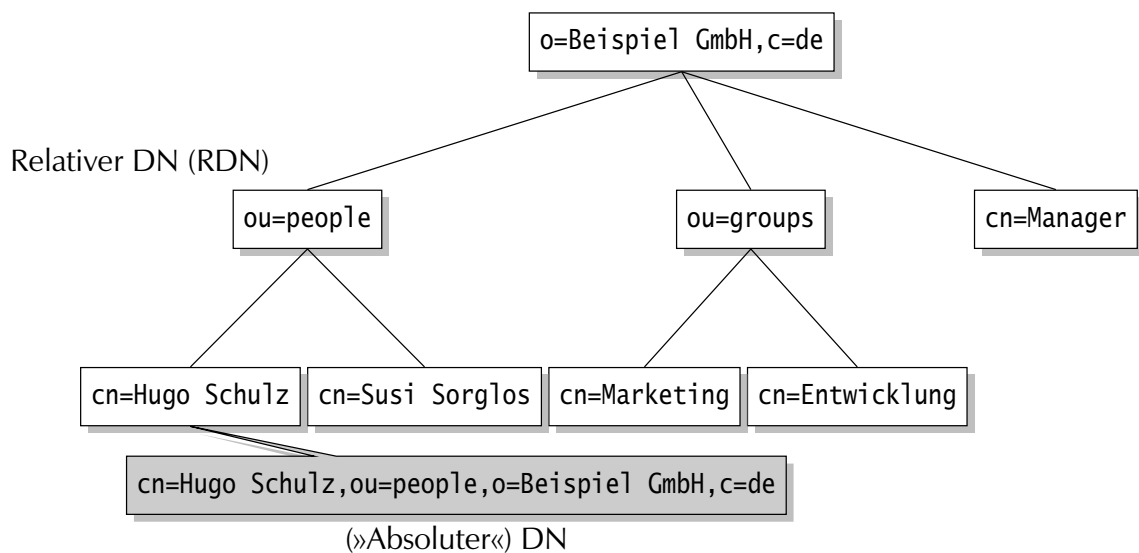


Bild 1.6: Directory Information Tree (DIT)

Tabelle 1.3: Typische Schlüsselwörter in Distinguished Names

Schlüssel	Bedeutung
cn	Name
l	Ortsangabe (typisch Stadt)
st	(Bundes-)Staat, Provinz, Bundesland
o	Organisation
ou	Organisations-Untereinheit
c	Staat/Land (2-Buchstaben-Code nach ISO 3166)
street	Straßenadresse
uid	Benutzername
dc	Domainnamen-Komponente

```
cn=Hugo Schulz+ou=Marketing
cn=Hugo Schulz+ou=Entwicklung
```

beschreiben zwei verschiedene Einträge. DNs im engeren Sinne sind durch Komma getrennte Folgen von RDNs, vom Eintrag bis zur Wurzel des DIT:

```
cn=Hugo Schulz+ou=Marketing,o=Beispiel GmbH,c=DE
```

(Dieser DN ist 3 Ebenen »tief« und nicht zu verwechseln mit dem DN

```
cn=Hugo Schulz,ou=Marketing,o=Beispiel GmbH,c=DE
```

der Tiefe 4.)

Ein DIT kann auf verschiedene Arten organisiert werden, etwa geographisch oder der internen Struktur einer Organisation folgend. In einem »geographischen« DIT wären Staaten die obersten Objekte, darunter kämen dann Bundesländer, Bundesstaaten, Provinzen oder Organisationen auf staatlicher Ebene (in Deutschland etwa der Bundestag). Auf der nächsten Ebene könnten dann Städte folgen, genau wie Organisationen auf der Ebene der Bundesländer. Die »Blätter« des DIT wären Personen oder Ressourcen innerhalb allfälliger Unter- oder Unter-Unter-Organisationen. Die Tiefe oder Breite eines DIT unterliegt keinen Einschränkungen.

Distinguished Names werden in X.500 normalerweise in ASN.1 codiert (einer Schreibweise für strukturierte Daten aus der ISO/OSI-Netzwerkwelt). Für die Zwecke von LDAP ist ASN.1 allerdings zu aufwendig zu unterstützen. Aus diesem Grund gibt es eine Darstellung von DN's als UTF-8-Zeichenketten [RFC4514], die die ASN.1-Definitionen abbildet. Diese Darstellung entspricht der früher in diesem Kapitel gezeigten Form, wobei es noch einige Spezialfälle gibt: Treten in DN's gewisse Sonderzeichen auf wie Leerzeichen oder # am Anfang eines Attributwerts, Leerzeichen am Ende eines Attributwerts oder die Zeichen »,+"\<>«, dann muss eine Ersatzdarstellung gewählt werden, indem ein »\« davorgestellt wird. Beliebige Zeichen können dargestellt werden, indem sie byteweise durch die Zeichenkombination »\xy« ersetzt werden; xy ist dabei die hexadezimale Darstellung des betreffenden Bytes.

Einige Beispiele für Distinguished Names (nach [RFC4514]):

```
cn=Hugo Schulz,o=Beispiel GmbH,c=DE
ou=Verkauf+cn=Peter Schmidt,o=Muster AG,c=AT
cn=R. Echtsverdrehler,o=Klag\, Abzock & Gierig,c=DE
cn=Vorher\@DNachher,o=Test,c=DE
1.3.6.1.4.1.1466.0=#04024869,o=Test,c=DE
```

Der CN im vorletzten Beispiel enthält einen Zeilenumbruch; das letzte Beispiel enthält ein Attribut, das keinen bekannten Namen hat, sondern nur durch einen OID beschrieben wird. Der Wert dieses Attributs ist die Codierung der zwei Byte langen Oktettfolge (72, 105) gemäß der *Basic Encoding Rules* (BER).



[RFC4514] ist der letzte in einer ganzen Reihe von RFCs – beginnend mit dem »Original«, [RFC1779] –, die sich mit dem Thema »Distinguished Names« befassen. Von Version zu Version wurden verschiedene Dinge präzisiert, aber seit [RFC2253] hat sich nichts wirklich Wesentliches geändert.



[RFC2253] war im Wesentlichen ein Neuaufguss von [RFC1779] für LDAPv3. [RFC1779] war in einigen Punkten wesentlich freizügiger als [RFC2253] – zum Beispiel waren Leerzeichen vor und nach »Trennkommas« in DN's erlaubt und wurden ignoriert, und das »Trennkomma« durfte auch ein Semikolon sein –, allerdings ist diese Syntax seit [RFC2253] offiziell verpönt.

Ein alternativer Ansatz besteht darin, den DN für eine Organisation (und die darin enthaltenen Einträge) vom DNS-Domainnamen der Organisation abzuleiten. Dazu werden die einzelnen Bestandteile des Domainnamens als »Domain-Komponenten« (kurz dc) in einem DN aneinandergehängt. Aus example.com wird so dc=example,dc=com, und einer Adresse wie hugo@example.com entspricht ein DN wie uid=hugo,dc=example,dc=com.



Eine nette Konsequenz dieser Methode ist, dass es damit einfach wird, den LDAP-Server für eine Organisation mit bekanntem DN im DNS »findbar« zu machen. [RFC2052] beschreibt, wie das geht: Für dc=example,dc=com hinterlegen Sie im DNS ein SRV-Record der Form

```
_ldap._tcp.example.com. IN SRV 10 1 389 ldap.example.com.
```

Die genauen Details gehören eher zum Thema »DNS«, aber das SRV-Record sagt etwas aus wie »Für den TCP-Dienst LDAP steht für die Domain example.com ein Server mit der Priorität 10 und dem Gewicht 1 auf ldap.example.com, Port 389 zur Verfügung.« Mit der Priorität können Sie zwischen »primären« und »sekundären« Servern unterscheiden (denken Sie an MX-Records), während Sie mit dem Gewicht bestimmte Rechner derselben Priorität bevorzugen können: Angenommen, es gibt drei Server mit derselben Priorität, die respektive das Gewicht 2, 3 und 5 haben. Die Summe der Prioritäten ergibt 10 – der erste Server sollte also theoretisch rund 20%, der zweite

rund 30% und der dritte rund 50% der Last abbekommen. Implementieren kann man das in einem Client, indem man den Client »würfeln« lässt (ein zehneitiger Würfel wäre hier richtig) und je nach dem Ergebnis (1–2, 3–5 oder 6–10) den entsprechenden Server verwendet.



Strenggenommen ist ein DIT eine »baumartige« Datenstruktur, nicht notwendigerweise wirklich ein Baum. Das liegt an der Existenz von Aliasnamen, die auf andere Einträge im Baum verweisen können und so die strikte Baumstruktur aufweichen.

Übungen



1.4 [!1] Erfinden Sie einen »geographischen« DN für Sie selbst, der Ihre Organisationszugehörigkeit, den Ort der Organisation und so weiter beschreibt. Müssen Sie sich besonders bemühen, um Ihren DN eindeutig zu machen?



1.5 [1] Was ist der Unterschied zwischen den DNs `cn=Emil Huber+ou=Buchhaltung,o=Beispiel GmbH,l=Musterstadt,c=DE` und `cn=Emil Huber,ou=Buchhaltung,o=Beispiel GmbH,l=Musterstadt,c=DE`?



1.6 [*3] Diskutieren Sie die Vor- und Nachteile »geographischer« DNs gegenüber solchen auf der Basis von DNS-Domainnamen gemäß [RFC2052].

1.4 Das funktionale Modell

LDAP definiert nicht nur die Struktur und Namensgebung von Daten in einem Verzeichnis, sondern auch Operationen zum Zugriff. Diese lassen sich grob in drei Gruppen einteilen:

Authentisierung Mit den Authentisierungsoperationen kann ein Clientprogramm sich an einen LDAP-Server binden oder die Bindung lösen. Dabei werden die Identität des Clients ermittelt und die passenden Zugriffsrechte auf die Informationen im Verzeichnis etabliert.

Suche Die Suchoperationen erlauben das Durchsuchen eines Verzeichnisses nach diversen Kriterien sowie den Vergleich von Daten mit beliebigen Einträgen.

Aktualisierung Informationen können hinzugefügt, verändert und gelöscht werden. Außerdem ist es möglich, RDNs zu modifizieren.

Authentisierung Am Anfang einer LDAP-Sitzung muss der Client sich an einen LDAP-Server *binden*. Hierbei hat er Gelegenheit, sich zu identifizieren, um zum Beispiel erweiterte Lese- und Schreibrechte zu bekommen; es sind auch **anonyme Sitzungen** möglich, bei denen der Client sich nicht authentisieren muss. Vor LDAPv3 konnte das Protokoll keine (sinnvolle) starke Authentisierung; in LDAPv3 wurde SASL [RFC2222] integriert, um authentifizierte und verschlüsselte Sitzungen zu ermöglichen. LDAP unterstützt die folgenden **Authentisierungsoperationen**:

Authentisierungsoperationen

bind Eröffnet eine LDAP-Sitzung zwischen einem Client und einem Server. Der Client kann sich identifizieren.

unbind Beendet eine LDAP-Sitzung.

abandon Bricht eine laufende Operation ab.

Suche Die **Suchoperationen** in LDAP erlauben es, ein Verzeichnis zu lesen und Einträge daraus aufzulisten. In LDAP sind dies beides Aspekte der Funktion `search`. Einige Beispiele für LDAP-Suchoperationen:

- Finde die E-Mail-Adresse von `cn=Hugo Schulz,o=Beispiel GmbH,c=DE`
- Finde alle Einträge unterhalb von `ou=Verkauf,o=Beispiel GmbH,c=DE`
- Finde die Telefonnummern aller Angestellten der `Beispiel GmbH`, die »Hugo« mit Vornamen heißen

Die `search`-Operation ist sehr allgemein und erlaubt es, diverse Parameter anzugeben:

Ausgangspunkt der Suche Die Suche beginnt immer bei einem **Basisobjekt**, einem bestimmten Knoten im DIT. Das Basisobjekt wird über einen DN identifiziert. Normalerweise ist es möglich, ein Basisobjekt vorherzubestimmen, so dass es nicht für jede Suchoperation angegeben werden muss.

Suchbereich Der Suchbereich bestimmt, welche Objekte ausgehend vom Basisobjekt betrachtet werden. Es gibt drei Möglichkeiten:

baseObject (`base`) Nur das Basisobjekt wird angeschaut.

singleLevel (`one`) Nur die unmittelbaren »Kinder« des Basisobjekts werden angeschaut, das Basisobjekt nicht.

wholeSubtree (`sub`) Das Basisobjekt und der ganze Teilbaum unterhalb werden angeschaut.

Suchfilter Der Suchfilter beschreibt die Kriterien, die ein Objekt erfüllen muss, um als Ergebnis der Suche in Frage zu kommen. Dies kann eine Kombination verschiedener Attributwertbedingungen sein, wobei eine **Attributwertbedingung** den Wert eines Attributs zum Beispiel auf Gleichheit mit einer Konstanten oder Übereinstimmung mit einem regulären Ausdruck prüft.

Gewünschte Attribute Es ist möglich, anzugeben, welche Attribute der auf den Suchfilter passenden Einträge zurückgeliefert werden sollen. Außer den Werten der Attribute kann auch (nur) der Typ von Attributen abgefragt werden, so dass Sie bei Attributen mit potenziell umfangreichen Werten die Werte später gezielt holen können.

Auswertung von Aliasnamen Bei der Suche können Sie angeben, ob Aliasnamen als solche zurückgeliefert oder »verfolgt« werden sollen. Dies erlaubt es, Aliasnamen direkt als Hilfsmittel zum Finden von Einträgen im Verzeichnis zu benutzen, oder die Aliasnamen selbst anzuschauen.

Begrenzungen Hiermit ist es möglich, Suchoperationen auf eine bestimmte zu verbrauchende Rechenzeit oder einen gewissen Ergebnisumfang zu beschränken, um zu vermeiden, dass einzelne Benutzer mit riesigen Suchoperationen einen LDAP-Server monopolisieren oder lahmlegen.

Wenn das Basisobjekt nicht auf dem gefragten Server selbst liegt, kann dieser einen **Verweis** auf den richtigen Server liefern (in LDAPv3). Die Suchoperation selbst kann neben Einträgen auch Verweise auf andere Einträge liefern; diese Verweise heißen **Fortsetzungsverweise**, weil sie angeben, wo die Suche fortgesetzt werden könnte. Ein Fortsetzungsverweis garantiert nicht, dass am Ziel des Verweises tatsächlich weitere passende Einträge zu finden sind, sondern macht nur einen Vorschlag.

Die Vergleichsoperation (`compare`) prüft ein Attribut eines bestimmten Eintrags auf Gleichheit zu einem vorgegebenen Wert. Sind Attributwert und Vorgabe gleich, ist das Resultat »Wahr«, sonst »Falsch«. Dies ist eine gegenüber der allgemeinen Suche deutlich eingeschränkte Operation, was sich später bei der

```

<filter> ← ( <filtercomp> )
<filtercomp> ← <and> | <or> | <not> | <item>
<and> ← & <filterlist>
<or> ← | <filterlist>
<not> ← ! <filter>
<filterlist> ← <filter>+
<item> ← <simple> | <present> | <substring> | <extensible>
<simple> ← <attr> <filtertype> <value>
<filtertype> ← = | ~= | >= | <=
<extensible> ← <attr> [:dn] [:<matchingrule>] := <value>
    | [:dn] :<matchingrule> := <value>
<present> ← <attr> =*
<substring> ← <attr> = [ <initial> ] <any> [ <final> ]
<initial> ← <value>
<any> ← * ( <value> )*
<final> ← <value>
<attr> ← ein Attributname, siehe [RFC4512, 2.5]
<matchingrule> ← der Name einer Vergleichsregel, siehe [RFC2251, 4.1.9]
<value> ← ein Attributwert, siehe [RFC4511, 4.1.5]

```

Bild 1.7: Syntax für Suchfilter gemäß [RFC4515]

Diskussion von Zugriffsrechten (Abschnitt 4.1) äußern wird. Existiert das Attribut überhaupt nicht, ist das Ergebnis »Falsch«; im Fall einer (ansonsten sehr ähnlichen) Suchoperation mit Suchbereich `baseObject` würde »Nicht gefunden« zurückgegeben werden, was nicht von dem Fall zu unterscheiden ist, dass das durchsuchte Objekt selbst überhaupt nicht existiert.

Suchfilter **Suchfilter** machen es möglich, bestimmte Einträge unter den prinzipiell in Frage kommenden (also denen mit den gegebenenfalls gewünschten Attributen) auszuwählen. [RFC4515] definiert eine Standardnotation für Suchfilter, die in Bild 1.7 kurz dargestellt ist. Ein einfacher Suchfilter wäre etwas wie

```
(cn=Hugo Schulz)
```

(alle Einträge, deren `cn`-Attribut den Wert »Hugo Schulz« hat), und Sie können mehrere solche Suchfilter logisch und- oder oder-verknüpfen, wobei eine Präfixnotation verwendet wird:

```
(&(cn=Hugo Schulz)(title=Generaldirektor))
(|(cn=Hugo Schulz)(cn=Emil Huber)(cn=Gabi Becker))
```

Auch die logische Negation einer solchen Bedingung ist erlaubt:

```
(!(cn=Hugo Schulz))
```

trifft auf alle Einträge zu, deren `cn`-Attribut *nicht* den Wert »Hugo Schulz« hat. Es ist möglich, nach Teilzeichenketten in den Attributwerten zu suchen, wobei die nicht näher bestimmten Abschnitte durch den Stern (*) vertreten werden:

```
(cn=Hugo *)
```

Der *erweiterbare* (*extensible*) Vergleich erlaubt die Verwendung einer beliebigen (vordefinierten) Vergleichsregel sowie das Einbeziehen von Komponenten des DN in den Eintrag:

```
(cn:1.2.3.4.5:=Hugo Schulz)
(sn:dn:2.4.6.8:=Emil Huber)
```

Hierbei vergleicht das erste Beispiel den Wert des Attributs `cn` gemäß der durch den OID 1.2.3.4.5 identifizierten Vergleichsregel mit dem Text »Hugo Schulz«. Im zweiten Beispiel wird nicht nur das `sn`-Attribut des Eintrags, sondern auch eine gegebenenfalls vorhandene `sn`-Komponente des DN, der den Eintrag im DIT identifiziert, in den Vergleich mit einbezogen. Übrigens können Sie den Attributnamen auch weglassen:

```
(:dn:3.6.9.12:=Hugo Schulz)
```

vergleicht alle Attribute, die die Vergleichsregel 3.6.9.12 unterstützen, mit dem vorgegebenen Wert »Hugo Schulz«.

Die Zeichen `*` `()` `\` in Werten müssen durch einen Rückstrich gefolgt von ihrem hexadezimalen Zeichencode, also (respektive) `\2a`, `\28`, `\29` und `\5c` ersetzt werden, das Nullbyte durch `\00`; alle anderen Zeichen können wahlweise auch so ersetzt werden. Dies ermöglicht die Darstellung aller Suchfilter als nullterminierte UTF-8-Zeichenketten.

Aktualisierung Die folgenden Aktualisierungsoperationen sind standardisiert:

add Fügt dem Verzeichnis neue Einträge hinzu.

delete Löscht existierende Einträge aus dem Verzeichnis. Nur »Blätter« im DIT, also Knoten ohne untergeordnete Knoten, dürfen entfernt werden.

modify Erlaubt das Verändern, Hinzufügen und Löschen von Attributen und ihren Werten in einem Eintrag.

modify DN Erlaubt das Verändern von einzelnen Komponenten eines DN. Dies ist nicht nur für »Blätter« im DIT erlaubt, sondern für beliebige Objekte, so dass es möglich ist, damit einen kompletten Teilbaum des DIT zu »verschieben«. Allerdings ist es nicht unbedingt möglich, damit Einträge von einem Server auf einen anderen zu verlagern.

LDAP-Operationen können über **controls** innerhalb des Protokolls verändert oder erweitert werden. Außerdem ist es möglich, ganz neue Operationen (**extended operations**) einzuführen.

Übungen



1.7 [!2] Geben Sie Suchfilter gemäß [RFC4515] an, mit denen Sie Einträge suchen würden, die jeweils die folgenden Anforderungen erfüllen:

1. die Landesangabe ist »Deutschland« (ISO 3166: `de`) oder »Österreich« (`at`);
2. wie zuvor, nur soll auch der DN mitbetrachtet werden;
3. der »bürgerliche« Name ist »François André de la Meunière«;
4. das Attribut `fasel` enthält einen Stern (*).



1.8 [2] Der Manager Hugo Schulz mit dem DN `cn=Hugo Schulz,ou=Entwicklung,o=Beispiel GmbH,l=Musterstadt,c=DE` wechselt innerhalb der Beispiel GmbH von der Entwicklungs- in die Marketingabteilung. Mit welcher Operation würden Sie den LDAP-Verzeichnisbaum für die Beispiel GmbH diesem Umstand anpassen?

1.5 Das Sicherheitsmodell

bind-Operation Die Sicherheit von LDAP-Zugriffen beruht zunächst auf der **bind-Operation**, über die ein Client eine Verbindung zum LDAP-Server herstellt. Für LDAP sind verschiedene Arten von **bind-Operationen** standardisiert: Authentisiertes und anonymes Binden mit Klartextkennwörtern sowie Zugriff mit Kerberos-Authentisierung (in LDAPv2, verpönt in LDAPv3) und mit Authentisierung über das Simple Authentication and Security Layer (SASL, in LDAPv3; eine der Möglichkeiten von SASL umfasst Kerberos).

Simple Authentication and Security Layer

Anonymes Binden

Im einfachsten Fall präsentiert der LDAP-Client dem Server einen DN und ein Klartextkennwort (die Methode ähnelt der *Basic*-Authentisierung von HTTP). Passt das Klartextkennwort zum angegebenen DN, ist der Client authentisiert. Ohne DN und Kennwort ist nur anonymes Binden möglich. (Eine Folge hiervon ist, dass z. B. Unix-Benutzerauthentisierung gegen eine LDAP-Datenbank nur möglich ist, wenn die LDAP-Datenbank anonymen Authentisierungszugriff gestattet, da der Benutzer »vor Ort« eben noch nicht authentisiert ist.) Das ist natürlich nicht der Gipfel der Sicherheit und sollte darum im Allgemeinen nur angewendet werden, wenn ein anderes Verfahren die Authentizität und Vertraulichkeit der LDAP-Sitzung sicherstellt oder wenn es nicht auf Authentizität und Vertraulichkeit ankommt, etwa bei Zugriffen auf ein öffentliches Telefonverzeichnis. Bei einer Anmeldung ohne DN und Kennwort nimmt der LDAP-Server an, dass anonymes Binden gewünscht ist, und behandelt die Sitzung entsprechend.

Profile (SASL) SASL [RFC2222] dient dazu, verbindungsorientierte Protokolle um zusätzliche Authentisierungsmechanismen zu erweitern. Für solche Protokolle werden **Profile** definiert, die regeln, wie das Protokoll mit SASL interagiert; solche Profile gibt es für gängige Protokolle wie IMAP, SMTP und eben auch LDAP. Normalerweise geht dies mit einer Erweiterung des Protokolls einher, das SASL verwenden soll. SASL erlaubt den Zugriff auf diverse **Mechanismen** zur Authentisierung, etwa S/Key, CRAM-MD5 oder Kerberos; weitere Mechanismen können als *external* integriert werden. Ferner gibt es einen »ANONYMOUS«-Mechanismus, über den Sie sich als »anonymer« Benutzer authentisieren können. Nach einer erfolgreichen Authentisierung kann über SASL auch ein Verschlüsselungsverfahren ausgehandelt werden. Neben einem DN muss zur SASL-basierten Authentisierung der gewünschte Authentisierungsmechanismus nebst etwaigen dafür notwendigen Parametern (*credentials*) angegeben werden. Der zu verwendende Authentisierungsmechanismus muss vom angesprochenen LDAP-Server unterstützt werden; welche Mechanismen ein LDAP-Server unterstützt, läßt sich durch einen LDAP-Zugriff auf

Mechanismen (SASL)

```
ldap://<Server>/?supportedsaslm mechanisms
```

herausfinden (LDAP-URLs werden in Abschnitt 1.6 genauer erklärt).

Transport Layer Security Ein anderes mögliches Verfahren zur Authentisierung und Verschlüsselung von LDAP-Sitzungen ist **Transport Layer Security (TLS)**, der Nachfolgestandard des **Secure Socket Layer (SSL)**. Für LDAPv3 ist eine Erweiterung definiert, die es gestattet, LDAP-Sitzungen über TLS abzuwickeln. Hiermit können nicht nur die Kommunikationsinhalte verschlüsselt, sondern auch der Server gegenüber dem Client sowie der Client gegenüber dem Server authentisiert werden, damit beide wissen, dass sie mit dem richtigen Partner und nicht einem »Hochstapler« kommunizieren. Für LDAPv2 haben einige Hersteller (etwa IBM oder Netscape) proprietäre SSL-Erweiterungen realisiert, die mit der Ankunft der »offiziellen« TLS-Erweiterung obsolet geworden sind. SSL bzw. TLS wird für LDAP außerdem als *external*-Mechanismus mit SASL verwendet und stellt das gängigste Verfahren hierzu dar.

Secure Socket Layer

Übungen



1.9 [2] Warum ist es sinnvoll, Authentisierung beim Zugriff auf Verzeichnisse einzusetzen? Überlegen Sie sich Argumente für Authentisierung von

Clients beim Server und auch Authentisierung von Servern beim Client.



1.10 [2] (Falls Sie schon Zugriff auf einen LDAP-Server haben:) Finden Sie heraus, welche SASL-Mechanismen Ihr LDAP-Server unterstützt (wenn überhaupt).

1.6 LDAP-URLs

Uniform Resource Locators (URLs) dienen zur Identifizierung von Ressourcen auf dem Internet. Bekannt sind sie vor allem durch das World Wide Web, das URLs der Form `http://...` verwendet, um auf Ressourcen zu verweisen, die über HTTP zugänglich sind, aber auch für zahlreiche andere Protokolle sind URL-Schemata definiert, unter anderem auch für LDAP (siehe [RFC4516]). LDAP-URLs beginnen mit `ldap://` oder `ldaps://` (wenn über SSL/TLS auf einen Server zugegriffen wird). Sie dienen zur Bezeichnung einzelner LDAP-Server, aber auch zu vielen anderen Zwecken; so können sie zum Beispiel komplexe Suchoperationen ausdrücken. (Leider können die meisten Web-Browser mit `ldap://`-URLs nichts anfangen. Der KDE-Browser »Konqueror« bildet eine rühmliche Ausnahme.)

Die abstrakte Syntax eines LDAP-URL ist

```
ldap[s]://[<Rechner>[:<Port>]][/<DN>][?<Attribute>\
[?<Suchbereich>][?<Filter>][?<Erweiterungen>]]]]
```

Dabei sind die einzelnen Bestandteile die folgenden:

<Rechner> Name oder IP-Adresse des LDAP-Servers

<Port> Portnummer des LDAP-Servers, ersatzweise 389 (für `ldap://`) oder 636 (für `ldaps://`)

<DN> Der DN, von dem die Suche ausgehen soll (das Basisobjekt, siehe S. 11)

<Attribute> Die Liste der gewünschten Attribute, durch Kommas getrennt. Ist dieses Feld leer, werden alle Attribute zurückgegeben

<Suchbereich> Der gewünschte Suchbereich (base, one oder sub, ersatzweise base)

<Filter> Ein Suchfilter, ersatzweise `objectClass=*` (lässt alle Objekte durch)

<Erweiterungen> Erlaubt die Definition von Erweiterungen für LDAP-URLs

Hier sind einige Beispiele für LDAP-URLs:

```
ldap://ldap.beispiel.de:389/
```

Dieser URL verweist auf den LDAP-Server `ldap.beispiel.de`. Die Portnummer 389 könnte auch weggelassen werden, da diese Portnummer standardmäßig eingesetzt wird, wenn der URL keine Portnummer enthält.

```
ldap://ldap.beispiel.de/o=Beispiel%20GmbH,c=de
```

Dieser URL liefert alle Attribute des Eintrags `o=Beispiel GmbH,c=de` vom Server `ldap.beispiel.de`. Das `%20` ist eine Schreibweise für das Leerzeichen innerhalb von URLs; Leerzeichen und diverse andere Sonderzeichen, die in DNs eigentlich erlaubt sind und nicht besonders codiert werden müssen, müssen zur Verwendung in URLs umschrieben werden, um Clientprogramme nicht zu verwirren.

```
ldap://ldap.beispiel.de/o=Beispiel%20GmbH,c=de??sub?cn=Hugo*
```

```

dn: dc=example,dc=com
objectclass: organization
objectclass: dcObject
o: Beispiel-Domain
dc: example

dn: ou=users,dc=example,dc=com
objectclass: organizationalUnit
ou: users

dn: uid=hugo,ou=users,dc=example,dc=com
objectclass: inetOrgPerson
uid: hugo
cn: Hugo Schulz
sn: Schulz

```

Bild 1.8: Eine beispielhafte LDIF-Datei

Mit diesem URL bekommt man alle Attribute von Einträgen im DIT-Teilbaum unter `o=Beispiel GmbH,c=de`, deren Namen mit »Hugo« anfangen.

```

ldap://ldap.beispiel.de/o=Beispiel%20GmbH,c=de?ou?sub
?objectClass=organizationalUnit

```

(in einer Zeile) Dieser URL liefert die Namen aller organisatorischen Untereinheiten der Beispiel GmbH.

LDAP-URLs können an vielen Orten verwendet werden. Beispielsweise ist es möglich, Teilbäume des DIT auf andere Server zu verteilen und über LDAP-URLs auf diese zu verweisen. Eine andere interessante Anwendung, die ein LDAP-basiertes E-Mail-Adressbuch voraussetzt, ist die Formulierung von Mail-Verteilerlisten als LDAP-URLs: Ein URL wie

```

ldap://ldap.beispiel.de/o=Beispiel%20GmbH,c=de?mail?sub?ou=Marketing

```

könnte die E-Mail-Adressen aller Mitglieder der Marketingabteilung liefern. Das Schöne an dieser Art von Verteilerliste ist, dass die Adressaten erst zum Zeitpunkt des Versands einer Rundmail bestimmt werden und damit keine Liste manuell aktualisiert werden muss, wenn Personen neu in die Abteilung kommen oder sie verlassen. – Außerdem verwenden viele Dienste LDAP-URLs in ihrer Konfiguration und legen so Quellen für Authentisierungsdaten fest. Auf diese Weise lässt sich zum Beispiel der Kreis der autorisierten Benutzer eines bestimmten Dienstes auf solche einschränken, deren LDAP-Einträge bestimmte Objektklassen, Attribute oder Attributwerte haben. Eine Anmeldung an einem Samba-Server sollte zum Beispiel nur dann möglich sein, wenn der Benutzer in seinem Verzeichniseintrag auch die dafür nötigen erweiterten Informationen hat.

Übungen



1.11 [!1] Formulieren Sie eine LDAP-URL, mit der Sie im DIT-Teilbaum der Beispiel GmbH die Telefonnummer von Hugo Schulz suchen können.

1.7 LDIF

Zum Im- und Export von Verzeichnisinformationen zwischen LDAP-basierten LDAP Data Interchange Format Servern oder zum Eintragen von Daten ins Verzeichnis wird meist das **LDAP Da-**

ta **Interchange Format** (LDIF) verwendet [RFC2849]. LDIF speichert Information in objektorientierten Hierarchien von Einträgen. Gängige LDAP-Pakete wie OpenLDAP enthalten Werkzeuge, um LDIF-Einträge ins jeweilige Backend-Format umzuwandeln. Bild 1.8 enthält eine beispielhafte LDIF-Datei.



Wir reden von »LDIF-Dateien«, aber meinen damit auch entsprechend formatierte Datenströme, die zum Beispiel über eine Shell-Pipeline von einem Programm zu einem anderen geschickt werden können, ohne im Dateisystem zu landen.

LDIF-Dateien gibt es in zwei Geschmacksrichtungen: Zum einen können sie eine statische Menge von Objekten beschreiben, zum anderen können sie aber auch eine Folge von Operationen (Attribut- oder Objekthinzufügungen, Attributänderungen, Attribut- oder Objektlöschungen, ...) enthalten, die auf einen DIT angewendet werden sollen. Eine LDIF-Datei gehört immer entweder zur einen oder zur anderen Sorte, nie zu beiden.



Zeilen in LDIF-Dateien, die mit »#« anfangen, sind Kommentarzeilen und werden komplett ignoriert.

Statische Menge Ein Eintrag für ein Objekt in einer LDIF-Datei, die eine statische Menge von Objekten beschreibt, beginnt immer mit dem DN des Objekts in der Form

```
dn: uid=hugo,ou=users,dc=example,dc=com
```

Dann folgen ein oder mehrere Attribut-Wert-Angaben. Diese bestehen aus einem Attributnamen gefolgt von einem Doppelpunkt, dann kommen (optional) Leerzeichen und dann der Attributwert:

```
objectclass: inetOrgPerson
uid: hugo
cn: Hugo Schulz
sn: Schulz
```



»Lange« Zeilen dürfen über mehrere Zeilen in der LDIF-Datei verteilt werden, indem man an der gewünschten Stelle einen Zeilentrenner unmittelbar gefolgt von einem Leerzeichen einbaut. (Dabei sollte man keine UTF-8-Zeichen in Stücke hacken, die mehr als ein Byte für ihre Darstellung brauchen.)

```
cn: Hugo
   Schulz
```

Beim Wiederzusammensetzen solcher Zeilen wird genau ein Leerzeichen am Anfang jeder Fortsetzungszeile entfernt, Sie sollten also keine zusätzlichen Leerzeichen einbauen, die Sie nicht hinterher in Ihren Daten haben möchten.



»Leere« Attributwerte (also solche der Länge 0) sind erlaubt; in diesem Fall dürfen nach dem Doppelpunkt nur Leerzeichen stehen.



Folgt dem Attributnamen die Zeichensequenz »:<«, dann ist der Rest der Zeile ein URL mit dem tatsächlichen Wert des Attributs:

```
jpegPhoto:< file://path/to/file.jpg
```



Dass :< die Angabe eines URL erlaubt, heißt nicht, dass man hier ungeniert mit beliebigen URL-Schemata operieren dürfte. Die Unterstützung von file://-URLs wird zumindest dringend empfohlen; ob http://- oder ftp://-URLs funktionieren, ist überhaupt nicht garantiert, und andere Schemata gibt es vermutlich erst recht nicht.

Jeder Eintrag im DIT braucht außer einem DN auch noch die Angabe seiner Objektklasse:

```
dn: uid=hugo,ou=users,dc=example,dc=com
objectclass: inetOrgPerson
```



Die Meinungen gehen auseinander, ob nur die speziellste Objektklasse angegeben werden muss oder auch noch die komplette Klassenhierarchie bis hin zu top, also zum Beispiel

```
dn: uid=hugo,ou=users,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
```

Es kann sein, dass ein LDAP-Server die komplette Klassenhierarchie braucht, um überprüfen zu können, ob die Attribute im Eintrag zu dem Schema passen, das sich aus der Klassenhierarchie für den Eintrag ergibt. OpenLDAP ist clever genug, die komplette Klassenhierarchie aus der Angabe der »untersten« Klasse (also etwa `inetOrgPerson`) zu rekonstruieren, aber das ist nicht universell garantiert. Wenn Sie also LDIF verwenden, um Daten aus OpenLDAP zu lesen und in einen anderen LDAP-Server zu importieren, dann könnte es sein, dass es Verwirrung gibt. Es ist aber nie ein Fehler (auch nicht bei OpenLDAP), die komplette Klassenhierarchie anzugeben.

Neben dem DN und der Objektklasse muss der Eintrag alle Attribute enthalten, die für sein Schema (zusammengesetzt aus der STRUCTURAL- und allen AUXILIARY-Objektklassen) vorgeschrieben sind – OpenLDAP überprüft das normalerweise auch. Daneben sind optional die Attribute erlaubt, die das Schema als wahlfrei vorsieht – auch das wird geprüft. Treten Attribute auf, die nicht im Schema zu finden sind, ist das ein Fehler.

In LDIF-Dateien sind nur Zeichen aus dem (7-Bit-)ASCII erlaubt. Wenn Sie beliebige Binärdaten darstellen wollen, müssen Sie sie Base64-codieren (Base64 ist eine Schreibweise für Daten, bei der drei beliebige Bytes durch vier Zeichen aus dem (ASCII-)Zeichenvorrat »A...Za...z0...9+/=« dargestellt werden, der auf allen wesentlichen Rechnerplattformen identisch dekodierbar ist und auch auf Strecken übertragen werden kann, die nur den 7-Bit-ASCII durchlassen). Dies wird im LDIF dadurch signalisiert, dass dem Attributnamen zwei Doppelpunkte statt nur einem folgen:

```
cn:: SHVnbyBTY2h1bHoK
```

»Hugo Schulz« in Base64



Base64-Codierung ist auch zwingend vorgeschrieben für Attributwerte, die mit einem »unsicheren« Zeichen anfangen. Als »unsicher« gelten das Nullbyte (ASCII `0x00`), der Zeilenvorschub (*line feed*, `0x0a`), der Wagenrücklauf (*carriage return*, `0x0d`), das Leerzeichen, der Doppelpunkt und das Kleiner-Zeichen (»<<«).



Attributwerte, die mit einem Leerzeichen *aufhören*, sollten ebenfalls Base64-codiert werden.



Base64-codierte DNs werden gemäß UTF-8 interpretiert, was Sonderzeichen und ähnliches angeht. Für Attributwerte im Allgemeinen gilt diese Annahme laut [RFC2849] *nicht*.

Folge von Operationen Statt einer statischen Menge von Einträgen kann eine LDIF-Datei auch eine Folge von Operationen beschreiben, die auf ein Verzeichnis angewendet werden soll. Im einfachsten Fall geht es um komplette Einträge:

```
dn: uid=susi,ou=users,dc=example,dc=com
changetype: add
objectclass: person
uid: susi
cn: Susi Sorglos
sn: Sorglos
```

beschreibt die Hinzufügung des Eintrags mit dem DN `uid=susi,ou=users,dc=example,dc=com` – beachten Sie das »changetype: add« unmittelbar hinter dem DN! Mit etwas wie

```
dn: uid=fritz,ou=users,dc=example,dc=com
changetype: delete
```

können Sie einen Eintrag löschen lassen.

Sie können auch einzelne Attribute in einem Eintrag modifizieren. In

```
dn: uid=hugo,ou=users,dc=example,dc=com
changetype: modify
add: mail
mail: hugo@example.com
-
replace: description
description: Unser Hugo
-
delete: telephonenumber
```

werden der Wert `hugo@example.com` zum Attribut `mail` hinzugefügt, der Wert des Attributs `description` durch »Unser Hugo« ersetzt und das Attribut `telephonenumber` komplett entfernt.



Erinnern Sie sich daran, dass Attribute mehr als einen Wert haben dürfen. Im Fall des Attributs `mail` ist es also durchaus denkbar, dass schon ein Wert existiert; in dem Fall hätte `mail` danach zwei Werte. Die `replace`-Operation dagegen ersetzt einen bestehenden Wert komplett.



Wenn Sie nur einen bestimmten Wert in einem Attribut mit mehreren Werten löschen möchten, können Sie das so beschreiben:

```
changetype: modify
delete: telephonenumber
telephonenumber: +49(0)1234-5678
```

Andere Werte desselben Attributs sind davon dann nicht betroffen.

Zu guter Letzt können Sie auch den DN eines Eintrags ändern. Im einfachsten Fall den »vordersten« RDN:

```
dn: cn=Susi Sorglos,ou=users,dc=example,dc=com
changetype: modrdn
newrdn: cn=Susi Sorglos-Zittermann
deleteoldrdn: 1
```

Treulich geführt ...



Das »deleteoldrdn: 1« verdient eine gesonderte Erklärung: Normalerweise enthält der Eintrag `cn=Susi Sorglos,ou=users,dc=example,dc=com` ein Attribut

```
cn: Susi Sorglos
```

Nach der Umbenennung würde dieses Attribut aber nicht mehr zum RDN des Eintrags passen. Aus diesem Grund sollte es entfernt werden. Die `modrdn`-Operation sorgt von selbst dafür, dass das `cn`-Attribut den neuen Wert

```
cn: Susi Sorglos-Zittermann
```

bekommt. (Wenn Sie »`deleteoldrdn: 0`« angeben, dann bleibt der alte Wert des Attributs zusätzlich enthalten – was im vorliegenden Fall vielleicht gar nicht mal eine *so* dumme Idee wäre ...)

LDAPv3-Server müssen auch unterstützen, dass ein RDN in der Mitte des DN geändert wird. Allerdings ist diese Operation etwas aufwendiger hinzuschreiben:

```
dn: cn=Lisa Lustig,ou=Marketing,dc=example,dc=com
changetype: modrdn
newrdn: cn=Lisa Lustig
deleteoldrdn: 0
newsuperior: ou=Vertrieb,dc=example,dc=com
```

Damit wird der Eintrag für Lisa Lustig aus dem Teilbaum des DIT unter `ou=Marketing,dc=example,dc=com` entfernt und direkt unter dem Eintrag `ou=Vertrieb,dc=example,dc=com` platziert.



X.500 verlangt nicht, dass das funktionieren muss, wenn `ou=Vertrieb,dc=example,dc=com` auf einem anderen Server liegt als `ou=Marketing,dc=example,dc=com`. Es ist aber auch nicht verboten, dass es klappt. Sie sollten sich halt nicht darauf verlassen.

Übungen



1.12 [!2] Erstellen Sie eine beispielhafte LDIF-Datei, die Einträge für die Organisation »Beispiel GmbH«, deren Unterorganisation »Marketing« und die Person Hugo Schulz enthält. (Tipp: Verwenden Sie für Herrn Schulz die Objektklasse `organizationalPerson`, siehe Bild 1.4 und [RFC2256]).



1.13 [2] Erstellen Sie eine beispielhafte LDIF-Datei, die zum Verzeichnis aus der vorigen Übung eine Unterorganisation »Entwicklung« hinzufügt und im Eintrag von Hugo Schulz ein Attribut hinzufügt, eines entfernt und eines ändert.

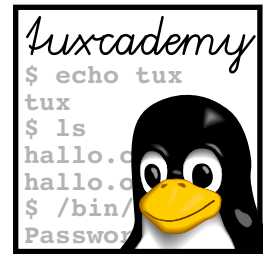
Zusammenfassung

- LDAP ist ein Protokoll für Client-Server-basierte, an X.500 angelehnte Verzeichnisdienste auf der Basis von TCP/IP.
- Ein LDAP-Verzeichnis enthält Einträge und diese wiederum Attribute mit Werten. Einträge gehören zu bestimmten Objektklassen.
- Daten in einem LDAP-Verzeichnis sind Bestandteil des *Directory Information Tree* (DIT)
- Einträge im DIT werden durch *Distinguished Names* (DNs) identifiziert. Die Baumstruktur ergibt sich aus der Struktur der DNs.
- LDAP bietet diverse Operationen aus den drei Funktionsbereichen Authentisierung, Suche und Datenaktualisierung.
- Sicherer Zugriff auf LDAP-Verzeichnisse kann über SASL oder SSL/TLS erfolgen.
- LDAP-URLs können Verweise auf LDAP-Server, LDAP-Anfragen und vieles mehr enthalten.
- Das LDIF-Format dient zum Im- und Export von LDAP-Verzeichnisinformationen.

Literaturverzeichnis

- AOID** Harald Alvestrand. »Object Identifiers«. Ein Überblick über OIDs mitsamt einer Möglichkeit, nachzuschlagen, welcher OID wem gehört.
<http://www.alvestrand.no/objectid/>
- RFC1288** D. Zimmerman. »The Finger User Information Protocol«, Dezember 1991.
<http://www.ietf.org/rfc/rfc1288.txt>
- RFC1617** P. Barker, S. Kille, T. Lenggenhager. »Naming and Structuring Guidelines for X.500 Directory Pilots«, Mai 1994.
<http://www.ietf.org/rfc/rfc1617.txt>
- RFC1779** S. Kille. »A String Representation of Distinguished Names«, März 1995.
<http://www.ietf.org/rfc/rfc1779.txt>
- RFC2052** A. Gulbrandsen, P. Vixie. »A DNS RR for specifying the location of services (DNS SRV)«, Oktober 1996.
<http://www.ietf.org/rfc/rfc2052.txt>
- RFC2222** J. Myers. »Simple Authentication and Security Layer«, Oktober 1997.
<http://www.ietf.org/rfc/rfc2222.txt>
- RFC2251** M. Wahl, T. Howes, S. Kille. »Lightweight Directory Access Protocol (v3)«, Dezember 1997.
<http://www.ietf.org/rfc/rfc2251.txt>
- RFC2253** M. Wahl, S. Kille, T. Howes. »Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names«, Dezember 1997.
<http://www.ietf.org/rfc/rfc2253.txt>
- RFC2256** M. Wahl. »A Summary of the X.500(96) User Schema for use with LDAPv3«, Dezember 1997.
<http://www.ietf.org/rfc/rfc2256.txt>
- RFC2849** G. Good. »The LDAP Data Interchange Format (LDIF) – Technical Specification«, Juni 2000.
<http://www.ietf.org/rfc/rfc2849.txt>
- RFC4511** J. Sermersheim. »Lightweight Directory Access Protocol (LDAP): The Protocol«, Juni 2006.
<http://www.ietf.org/rfc/rfc4511.txt>
- RFC4512** J. Sermersheim, K. Zeilenga. »Lightweight Directory Access Protocol (LDAP): Directory Information Models«, Juni 2006.
<http://www.ietf.org/rfc/rfc4512.txt>

- RFC4514** J. Sermersheim, K. Zeilenga, K. Zeilenga. »Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names«, Juni 2006. <http://www.ietf.org/rfc/rfc4514.txt>
- RFC4515** J. Sermersheim, K. Zeilenga, K. Zeilenga, et al. »Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters«, Juni 2006. <http://www.ietf.org/rfc/rfc4515.txt>
- RFC4516** M. Smith, T. Howes. »Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator«, Juni 2006. <http://www.ietf.org/rfc/rfc4516.txt>
- RFC4519** A. Sciberras. »Lightweight Directory Access Protocol (LDAP): Schema for User Applications«, Juni 2006. <http://www.ietf.org/rfc/rfc4519.txt>



2

LDAP-Verzeichnisse verwalten

Inhalt

2.1	LDAP-Werkzeuge für die Kommandozeile	24
2.2	Daten suchen mit <code>ldapsearch</code>	24
2.3	Daten hinzufügen, ändern und löschen	27

Lernziele

- Mit den OpenLDAP-Werkzeugen Daten in einem LDAP-Verzeichnis suchen, neue Einträge importieren, bestehende ändern und entfernen können.

Vorkenntnisse

- Umgang mit der Kommandozeile und mit einem Texteditor

Dieses Kapitel setzt voraus, dass Sie Zugriff auf einen installierten OpenLDAP-Server haben. In einem Kurs wird Ihr Trainer dafür sorgen; als Selbstlerner sollten Sie zuerst einen Blick in Kapitel 3 werfen, um die nötigen Rudimente aufzuspüren.

2.1 LDAP-Werkzeuge für die Kommandozeile

Die Kommandozeile ist – allen Fortschritten seitens grafischer Oberflächen zum Trotz – für die Benutzung von Linux nach wie vor sehr wichtig. Aus diesem Grund stehen auch für die Interaktion mit LDAP-Servern wie OpenLDAP Programme für die Kommandozeile zur Verfügung, mit denen alle wichtigen LDAP-Operationen ausgelöst werden können. Auf den ersten Blick erscheint das schwerfällig und altmodisch, aber die Vorteile liegen auf der Hand:

- Die Programme ermöglichen eine sehr detaillierte und wiederholbare Interaktion mit dem Verzeichnis.
- Die Programme erlauben das Erstellen von Shellskripten, die mehrfach benötigte Aufgaben automatisieren helfen.
- Die Programme können als Basis für bequemere Werkzeuge dienen, indem sie die Details des LDAP abstrahieren. Ein Programm mit grafischer Oberfläche muss also nicht notwendigerweise auch noch das Zugriffsprotokoll implementieren, sondern kann geeignete shellbasierte Kommandos zusammensetzen, diese ausführen und ihre Ausgabe verarbeiten. Das legt die Latenz für komfortable Werkzeuge deutlich niedriger.
- Die Programme funktionieren mit jeder konformen LDAP-Implementierung (nicht nur OpenLDAP).



Im Kontext des Lernens über LDAP erlauben die Kommandozeilen-Werkzeuge uns, uns auf die wesentlichen Zugriffsmöglichkeiten zu konzentrieren und diese mit leicht reproduzierbaren Beispielen zu illustrieren. Auch das ist ein wesentlicher Vorteil.

2.2 Daten suchen mit `ldapsearch`

Die einfachste Operation ist das Suchen nach Einträgen in einem LDAP-Verzeichnis. Hierzu dient das Kommando `ldapsearch`. Ein Aufruf der Form

```
$ ldapsearch -x
```

zum Beispiel gibt alle Einträge aus, die im Verzeichnisbaum unter dem DN »`dc=example,dc=com`« enthalten sind:

```
$ ldapsearch -x
# extended LDIF
#
# LDAPv3
# base <dc=example,dc=com> (default) with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# example.com
dn: dc=example,dc=com
objectClass: organization
```

Tabelle 2.1: Ausgabeformate für ldapsearch

Option	Ausgabe
keine	Ausgabe in einem erweiterten LDIF-Format
-L	Ausgabe im LDIFv1-Format
-LL	Kommentare werden unterdrückt
-LLL	Kommentare und LDIF-Versionsnummer werden unterdrückt

```
objectClass: dcObject
dc: example
o: Beispiel
description: Beispiel

# postmaster,example.com
<<<<<<
```



Wenn Ihr Verzeichnisbaum nicht bei »dc=example,dc=com« anfängt, können Sie mit der Option `-b` einen alternativen Basis-DN angeben:

```
$ ldapsearch -x -b dc=linupfront,dc=de
```

Das funktioniert natürlich auch, wenn Sie nur in einem Teilbaum des DIT suchen wollen.



Die Voreinstellung für den Basis-DN steht (normalerweise) in einer Datei namens `ldap.conf`, typischerweise in einem Verzeichnis wie `/etc/ldap` oder `/etc/openldap` (distributionsabhängig). Passen Sie auf – es könnte in Ihrem System auch andere `ldap.conf`-Dateien geben, die zum Beispiel von den PAM- und NSS-Modulen für LDAP verwendet werden; auch hier hängen die Details von Ihrer Distribution ab.



Versuchen Sie das Beispiel besser nicht mit dem Verzeichnisbaum Ihres multinationalen Unternehmens mit drei Millionen Einträgen.

Die Option `-x` wird hier gebraucht, um »einfache« (PLAIN) Authentisierung zu aktivieren. Wir sind in kindlicher Naivität davon ausgegangen, dass Sie sich nicht wirklich an Ihrem Verzeichnis authentisieren müssen, was für eine »Spielzeuginstallation«, wie sie etwa im Rahmen von Kapitel 3 vorgestellt wird, zutrifft (in Kapitel 4 zeigen wir Ihnen, wie Sie die Schotten etwas dichter machen können). Sollte Ihr LDAP-Server Sie aber kalt lächelnd abblitzen lassen, versuchen Sie etwas wie

```
$ ldapsearch -x -D "uid=hugo,ou=users,dc=example,dc=com" -W
```

Hierbei gibt die Option `-D` einen DN an, mit dem Sie sich beim Verzeichnis authentisieren wollen (das moralische Äquivalent zu einem Linux-Benutzernamen), und die Option `-W` sorgt dafür, dass `ldapsearch` Sie nach dem dazu passenden Kennwort fragt.



Wenn Sie die Option `-x` weglassen, nimmt `ldapsearch` an, dass Sie eine SASL-basierte Authentisierung vornehmen möchten. Das geht bei einem nicht dafür konfigurierten OpenLDAP-Server schief (wir verweisen wieder auf Kapitel 4), aber wenn Ihr LDAP-Server Sie mit PLAIN-Authentisierung nicht zulässt, dann könnten Sie es durchaus mal ohne das `-x` probieren und schauen, was passiert.

Wenn Sie genau aufgepasst haben, wird Ihnen aufgefallen sein, dass das Ausgabeformat von `ldapsearch` eine verdächtige Ähnlichkeit zu dem in Abschnitt 1.7

besprochenen LDIF aufweist. Das ist durchaus Absicht, da Sie die Ausgabe von `ldapsearch` so ohne weiteres in eine Datei schreiben, ändern und anschließend wieder ans Verzeichnis verfüttern können. Die Option `-L` gibt Ihnen eine gewisse Kontrolle über die Details des Ausgabeformats; Tabelle 2.1 enthält einen Überblick.

Filter Das erste `ldapsearch`-Argument ist ein Filter, der das Resultat auf bestimmte Einträge einschränken kann:

```
$ ldapsearch -LLL -x 'cn=Hugo Schulz'
dn: uid=hugo,ou=users,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Hugo Schulz
sn: Schulz
uid: hugo
```

gibt den Eintrag für den Benutzer Hugo Schulz aus. Sie können Filter in der Standardsyntax aus [RFC4515] spezifizieren (siehe auch Seite 11). Ist kein Filter angegeben, tritt der Standardfilter, »(objectclass=*)«, in Kraft.



Sie sollten darauf achten, den Suchfilter für die Shell immer in Anführungszeichen (am besten einfache) zu setzen. Besonders bei komplexeren Filtern wie

```
(&(objectClass=person)(cn=Hugo *))
```

oder

```
(|(ou=Marketing)(ou=Entwicklung))
```

werden oft Zeichen verwendet, die auch für die Shell eine Sonderbedeutung haben und darum »versteckt« werden müssen.



Die Option `-s` bestimmt den Suchbereich (base, one oder sub, Standardwert ist sub).

Weitere Argumente benennen die Attribute, die ausgegeben werden sollen:

```
$ ldapsearch -LLL -x 'cn=Hugo Schulz' sn
dn: uid=hugo,ou=users,dc=example,dc=com
sn: Schulz
```

Normalerweise versucht `ldapsearch`, einen LDAP-Server auf dem aktuellen Rechner anzusprechen. Sie können das Programm aber auf einen entfernten LDAP-Server verweisen, indem Sie mit der Option `-H` dessen URL angeben:

```
$ ldapsearch -H ldap://ldap.example.com/ -x
```

(Siehe Abschnitt 1.6.)



Sie dürfen auch mehrere URLs angeben, durch Freiplatz oder Kommas getrennt. In diesem Fall werden die Server nacheinander durchprobiert.



Früher war es üblich, mit der Option `-h` den Rechnernamen und (falls nötig) mit `-p` die Portnummer des LDAP-Servers festzulegen. Heute ist das zugunsten von `-H` verpönt.



Alternativ können Sie in `ldap.conf` mit dem URI-Schlüssel eine Liste von LDAP-URLs angeben:

```
URI ldap://ldap.example.com ldap://ldap-backup.example.com
```

ldapsearch unterstützt diverse weitere Optionen, von denen Sie einige später kennenlernen werden. Die meisten ldapsearch-Optionen gelten auch für die anderen LDAP-Kommandozeilenwerkzeuge.

2.3 Daten hinzufügen, ändern und löschen

Suchen können im Verzeichnis ist eine nette Sache, aber hin und wieder ist es auch nötig, ins Verzeichnis zu schreiben. Auch dafür gibt es Werkzeuge, von denen wir im Folgenden ein paar vorstellen.

Daten hinzufügen Mit ldapadd können Sie neue Daten in ein LDAP-Verzeichnis aufnehmen. Diese Daten müssen im LDIF vorliegen, typischerweise in einer Datei (Bild 1.8 enthält ein Beispiel) und können dem Verzeichnis zum Beispiel wie folgt hinzugefügt werden:

```
$ ldapadd -x -D "cn=Manager,dc=example,dc=com" -W -f beispiel.ldif
```

Hier binden wir uns mit dem DN des Administrators, cn=Manager,dc=example,dc=com, an das Verzeichnis (das -W erwartet wie bei ldapsearch, dass Sie auf Anfrage das Kennwort eingeben, und -x verlangt PLAIN-Authentisierung), was für unser einfaches Beispiel aus Kapitel 3 stimmt, aber in der freien Wildbahn natürlich nicht notwendigerweise zutreffen muss.



Mit der Option -w können Sie das Kennwort auch direkt auf der Kommandozeile angeben (etwa »-w geheim«). Das ist aber nicht empfehlenswert.

Die Einträge in der Datei beispiel.ldif dürfen im Verzeichnis noch nicht existieren, sonst gibt es eine Fehlermeldung. Sie sollten auch darauf achten, den DIT »von oben nach unten« aufzubauen; wenn Sie zum Beispiel den Eintrag cn=Hugo Schulz,ou=People,dc=example,dc=com hinzufügen wollen, dann muss in dem betreffenden Moment der Eintrag ou=People,dc=example,dc=com schon existieren, sonst gibt es ebenfalls eine Fehlermeldung.

Daten ändern Daten im Verzeichnis ändern können Sie mit dem Kommando ldapmodify. Ähnlich wie ldapadd übernimmt es Verzeichnisdaten gemäß LDIF, diesmal allerdings im »Folge von Operationen«-Format (siehe S. 19). Auch bei ldapmodify entsprechen die Optionen, die sich zum Beispiel mit Authentisierung befassen, denen von ldapsearch.



Genau genommen ist ldapadd nichts anderes als »ldapmodify -a«. Die beiden Kommandos werden tatsächlich vom selben Programm implementiert, das sich anhand seines Aufrufnamens für eine der beiden Geschmacksrichtungen entscheidet.



LDAP garantiert, dass eine Folge von Attributänderungen in der angegebenen Reihenfolge und »atomar« ausgeführt wird [RFC4511, 4.6]. Das heißt, entweder werden *alle* Änderungen ausgeführt oder überhaupt keine. Ferner muss der zu ändernde Eintrag nur unmittelbar vor und unmittelbar nach der Änderungsoperation dem Schema entsprechen; während der Operation ist das nicht erforderlich. Sie können also in derselben Änderungsoperation zum Beispiel alle Werte eines MUST-Attributs entfernen und neu hinzufügen, ohne dass der LDAP-Server sich über eine Schemaverletzung beschwert.

Um mit den shellbasierten Werkzeugen Einträge im Verzeichnis zu ändern, ist es oft sinnvoll, den zu ändernden Eintrag erst mit `ldapsearch` zu suchen und das Ergebnis in eine LDIF-Datei zu schreiben, die Sie dann mit einem Texteditor ändern können. Die geänderte LDIF-Datei schreiben Sie danach mit `ldapmodify` ins Verzeichnis zurück.

```
$ ldapsearch -LLL -x -b "dc=example,dc=com" \
> "cn=Hugo Schulz" >tmp.ldif
... fröhliches Editieren von tmp.ldif ...
$ ldapmodify -x -D "cn=Manager,dc=example,dc=com" -W \ -f tmp.ldif
```

Beim Löschen und Ändern von Einträgen muss normalerweise natürlich das Kennwort des Verzeichnisadministrators (oder eines anderen Verzeichnisbenutzers mit passendem Schreibrecht) angegeben werden!



Ein schöneres Programm zur Verzeichnisverwaltung ist `ldapvi`, das zum Beispiel in Debian GNU/Linux enthalten ist. Es automatisiert den im vorigen Absatz gezeigten Zyklus in sehr bequemer Weise.

Daten löschen Zum Löschen dient das Kommando `ldapdelete` – zwar können Sie das auch mit `ldapmodify`, aber `ldapdelete` übernimmt die DNs der zu entfernenden Einträge direkt auf der Kommandozeile (oder der Standardeingabe, wenn keine auf der Kommandozeile stehen) und enthebt Sie daher der Pflicht, LDIF-Dateien mit den entsprechenden Löschanforderungen zu konstruieren:

```
$ ldapdelete -x "uid=goner,ou=users,dc=example,dc=com"
```

RDNs ändern Ähnlich wie `ldapdelete` zum Löschen von Einträgen bietet `ldapmodrdn` eine bequeme Möglichkeit zum Ändern von RDNs, ohne dass Sie das über LDIF und `ldapmodify` erledigen müssen:

```
$ ldapmodrdn -x "cn=Susi Sorglos,ou=users,dc=example,dc=com" \
> "cn=Susi Zittermann"
```

LDAP verlangt bekanntlich, dass der »linkeste« RDN des Eintrags auch als Attributwert im Eintrag selbst vorhanden ist (in unserem Beispiel war das vor der Umbenennung »cn: Susi Sorglos«). `ldapmodrdn` kümmert sich automatisch darum, einen Attributwert für den *neuen* RDN einzutragen – in unserem Beispiel »cn: Susi Zittermann«, aber was mit dem »alten« Attributwert von vor der Umbenennung passieren soll, müssen Sie entscheiden: Die Option `-r` entfernt alte RDN-Werte im Eintrag; Standard ist, sie beizubehalten.



Was hier die richtige Vorgehensweise ist, ist nicht ganz klar und hängt von der Art der Daten ab. In unserem Beispiel könnte man vermuten, dass die frischgebackene Frau Zittermann sinnvollerweise auch noch unter ihrem alten Namen zu finden sein sollte und dass es darum vernünftig scheint, das »cn: Susi Sorglos« nicht zu löschen. In anderen Fällen kann das anders aussehen.



Denken Sie daran, dass ein Eintrag der Klasse `person` (oder einer, der von `person` abgeleitet ist) auch ein `sn`-Attribut hat. Wenn Sie einen auf `cn` basierenden RDN ändern, müssen Sie wahrscheinlich auch das `sn`-Attribut anpassen, und das nimmt `ldapmodrdn` Ihnen leider nicht ab.

Mit `-s` können Sie einen neuen übergeordneten Eintrag angeben und damit Einträge im DIT verschieben. Garantiert funktioniert das aber nur auf demselben Verzeichnisserver.

Kommandos in diesem Kapitel

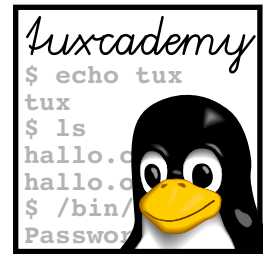
ldapadd	Fügt Einträge aus einer LDIF-Datei in ein LDAP-Verzeichnis ein	ldapadd(1)	27
ldapdelete	Löscht Einträge aus einem LDAP-Verzeichnis	ldapdelete(1)	28
ldapmodify	Ändert Einträge in einem LDAP-Verzeichnis anhand einer LDIF-Datei	ldapmodify(1)	27
ldapmodrdn	Ändert RDNs in einem LDAP-Verzeichnis	ldapmodrdn(1)	28
ldapvi	LDAP-Client auf vi-Basis	ldapvi(1)	28

Zusammenfassung

- OpenLDAP enthält leistungsfähige kommandozeilenorientierte Werkzeuge zur Abfrage und Manipulation von LDAP-Verzeichnissen.
- Mit `ldapsearch` können Sie Suchoperationen im Verzeichnis durchführen.
- Die Kommandos `ldapadd` und `ldapmodify` dienen zum Hinzufügen und Modifizieren von Verzeichniseinträgen.
- Kommandos wie `ldapdelete` und `ldapmodrdn` erleichtern spezielle Aufgaben.

Literaturverzeichnis

- RFC4511** J. Sermersheim. »Lightweight Directory Access Protocol (LDAP): The Protocol«, Juni 2006. <http://www.ietf.org/rfc/rfc4511.txt>
- RFC4515** J. Sermersheim, K. Zeilenga, K. Zeilenga, et al. »Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters«, Juni 2006. <http://www.ietf.org/rfc/rfc4515.txt>



3

Der OpenLDAP-Server

Inhalt

3.1	Installation von OpenLDAP	32
3.2	Konfiguration von OpenLDAP	32
3.2.1	Allgemeines.	32
3.2.2	»Traditionelle« Konfiguration mit slapd.conf	34
3.2.3	»Moderne« Konfiguration über LDAP	38
3.3	Ein Konfigurationsbeispiel	41
3.4	OpenLDAP-Betrieb	42
3.5	Weiterleitung	43

Lernziele

- Den OpenLDAP-Server auf einem Linux-System installieren und konfigurieren können
- Mit den OpenLDAP-Werkzeugen Daten in ein LDAP-Verzeichnis importieren, in einem LDAP-Verzeichnis suchen, Daten ändern und löschen können
- Weiterleitungen konfigurieren können

Vorkenntnisse

- TCP/IP-Netzwerkverwaltung unter Linux
- Zusätzliche Software für eine Distribution oder vom Quellcode aus installieren können

3.1 Installation von OpenLDAP

OpenLDAP ist eine Implementierung von LDAPv3, die das OpenLDAP-Projekt frei zur Verfügung stellt. Neben dem eigentlichen LDAP-Server (`slapd`) enthält das Paket auch die in Kapitel 2 diskutierten Werkzeuge zur Verzeichnisverwaltung. Zur Installation werden außerdem benötigt:

- Die TLS-Bibliotheken von OpenSSL;
- Die *Simple Authentication and Security Layer* (SASL)-Bibliotheken von Cyrus;
- Eine Datenbanksoftware, vorzugsweise Berkeley DB 4 von Sleepycat Software;
- Eine Thread-Implementierung (ist heutzutage unter Linux Standard).

Nicht notwendig sind, aber unterstützt werden:

- Kerberos-Authentisierungsdienste (Heimdal-Kerberos oder MIT Kerberos V)
- TCP-Wrapper

Distributionen Der OpenLDAP-Server kann von <http://www.openldap.org/> heruntergeladen werden. OpenLDAP und die dafür nötige weitere Software ist auch in vielen namhaften Linux-Distributionen, etwa denen von Novell/SUSE oder Debian GNU/Linux, enthalten. Wenn Sie eine solche Distribution verwenden, können Sie sich Mühe sparen, indem Sie die entsprechenden Pakete einspielen; ansonsten lässt der OpenLDAP-Server sich auf die inzwischen weithin übliche Weise mit `configure`, `make` usw. übersetzen, binden und installieren.



Die OpenLDAP-Entwickler empfehlen dringend, OpenLDAP selbst vom Quellcode aus zu installieren, um in den Genuss von Weiterentwicklungen und vor allem Fehlerkorrekturen zu kommen, die möglicherweise in den Distributions-Versionen nicht enthalten sind. Ob Sie sich dieser Empfehlung anschließen wollen, müssen Sie selbst wissen. Es ist ja auch gut möglich, dass zum Beispiel Zertifizierungsfragen Sie dazu zwingen, offizielle Software aus einer (»Enterprise«-)Distribution zu verwenden.

Init-Skript Wie der OpenLDAP-Server nach der Installation tatsächlich in Gang gesetzt wird, hängt davon ab, ob Sie ihn selbst installiert haben oder eine Distribution verwenden und, wenn ja, welche. Üblicherweise wird der Server beim Booten über ein **Init-Skript** gestartet, das sich zum Beispiel in `/etc/init.d` befindet (mit symbolischen Links, die dafür sorgen, dass der Server in den richtigen Runlevels läuft oder nicht läuft).

3.2 Konfiguration von OpenLDAP

3.2.1 Allgemeines

Nachdem OpenLDAP installiert wurde, muss es konfiguriert werden. Für den `slapd` gibt es dafür fundamental zwei Möglichkeiten:

- Die althergebrachte Methode platziert die Konfigurationseinstellungen in der Datei `slapd.conf`. Diese Datei findet sich bei Distributionen meist in einem Verzeichnis wie `/etc/openldap`; ein selbstinstalliertes OpenLDAP tut sie nach `/usr/local/etc/openldap`, wenn bei der Installation nichts anderes festgelegt wurde. (Dasselbe gilt für die anderen Konfigurationsdateien.)
- Die modernere Methode verwendet eine Konfigurationsdatenbank, typischerweise in einem Verzeichnis namens `slapd.d`, das dort steht, wo sonst die Datei `slapd.conf` stehen würde.

Die Idee hinter der modernen Methode ist, dass die Einträge der Konfigurationsdatenbank LDIF-Dateien sind und durch LDAP-Zugriffe auf den Server geändert werden können. Während Sie bei der traditionellen Methode nach Änderungen der Datei `slapd.conf` den `slapd` benachrichtigen müssen, damit er die Änderungen tatsächlich einliest, ist das bei Änderungen per LDAP gemäß der neuen Methode nicht mehr nötig.



Standardmäßig verwendet die Debian-Implementierung von OpenLDAP die moderne Konfigurationsmethode. Sie können die traditionelle Methode verwenden, indem Sie in der Datei `/etc/default/slapd` in der Variablen `SLAPD_CONF` die Konfigurationsdatei benennen, also zum Beispiel

```
SLAPD_CONF=/etc/ldap/slapd.conf
```

(Ein Beispiel finden Sie in `/usr/share/doc/slapd/examples/slapd.conf`.) Das Init-Skript kümmert sich dann um den Rest.



Die openSUSE-Distribution verwendet »ab Werk« die traditionelle Konfigurationsmethode, aber lässt sich auf die modernere Methode umstellen, indem Sie in der Datei `/etc/sysconfig/openldap` den Eintrag

```
OPENLDAP_CONFIG_BACKEND=ldap
```

setzen. Das Init-Skript nimmt dann an, dass die Konfiguration im Verzeichnis `/etc/openldap/slapd.d` steht.

Ein OpenLDAP-Server kann gleichzeitig mehrere **Datenbanken** (engl. *database*) zur Verfügung stellen, die verschiedene Bereiche des DIT abdecken. Für jede dieser Datenbanken müssen Sie ein **Backend** wählen, das `slapd` verwendet, um sich die Daten tatsächlich zu besorgen. Im einfachsten Fall kümmert der Server sich selbst um das Speichern und Wiederfinden der Daten, aber es ist auch möglich, dass er wiederum auf andere Server (oder allgemeiner gesagt Informationsquellen) zurückgreift, um Anfragen zu beantworten. Mehrere Datenbanken können gleichzeitig dasselbe Backend verwenden. Hier ist eine Liste der aktuell mitgelieferten Backends:

bdb (und **hdb**) Hochleistungsfähige Backends auf der Basis von Sleepycat Berkeley DB 4, die Transaktionen unterstützen und mit einer großen Zahl von gleichzeitigen Lese- und Schreiboperationen zurechtkommen. Außerdem sind sie robust gegenüber Systemabstürzen und Hardwarefehlern und verlieren dadurch keine bestätigten Transaktionen.



Der Unterschied zwischen **bdb** und **hdb** ist im Wesentlichen, dass **hdb** das Umbenennen von Unterbäumen erlaubt. Irgendwann in der Zukunft wird **bdb** vermutlich abgeschafft. Es gibt aktuell keinen zwingenden Grund, nicht **hdb** zu verwenden, außer möglicherweise akuter Speicherknappheit – **hdb** braucht größere Caches als **tdb**, um effizient zu arbeiten. Siehe hierzu `slapd-bdb(5)`.

Diese Backends bieten sich am ehesten für die »ernsthafte« Verwendung von OpenLDAP als Verzeichnisdienst an.

ldif Dieses Backend verwendet das Dateisystem, um Verzeichniseinträge in LDIF-Dateien zu speichern. Diese simple (wenn nicht übermäßig effiziente) Methode wird zum Beispiel von der modernen Konfigurationsmethode verwendet.

ldap Ein Backend, mit dem ein LDAP-Server als »Proxy« auftreten kann: Er reicht eine Anfrage an einen anderen LDAP-Server weiter und leitet dessen Antwort zurück an den ursprünglichen Client. Das `ldap`-Backend dient als Grundlage für viele andere Backends und Overlays.

meta Dieses Backend ähnelt ldap, aber erlaubt das Weiterreichen von Anfragen an mehrere entfernte LDAP-Server sowie das Umschreiben von DN's. Die Idee ist, »Metaverzeichnisse« aufzubauen, die verschiedene Verzeichnisse in einer einheitlichen Struktur erscheinen lassen.


monitor Dieses Backend speichert keine Daten, sondern dient der Beobachtung des slapd.

null Dieses Backend tut gar nichts bzw. das minimal Mögliche: Suchvorgänge liefern keine Ergebnisse, Aktualisierungen liefern Erfolg, aber die Daten werden verworfen usw. (denken Sie an /dev/null).

passwd Ein Backend für die /etc/passwd-Datei (nur zu Demonstrationszwecken).

shell (und perl) Diese Backends leiten Anfragen an Shell- bzw. Perl-Code weiter, den Sie als Administrator angeben können. Das ist extrem flexibel, aber kein Geschwindigkeitswunder; die Backends dienen vor allem dazu, Prototypen für die Anbindung existierender Datenbanken an LDAP zu implementieren und zu testen.

sql Dieses Backend verwendet ODBC zum (lesenden) Zugriff auf SQL-Datenbanken.

Overlays  Ferner unterstützt slapd **Overlays**, die sozusagen auf Backends gestapelt werden können, um deren Benehmen zu verändern. Mit dem accesslog-Overlay können Sie zum Beispiel alle Zugriffe auf eine Datenbank in einer anderen Datenbank protokollieren (und später über LDAP auf dieses Protokoll zugreifen).

Entsprechend enthält die OpenLDAP-Konfiguration drei Sorten von Informationen: *globale*, *backend-spezifische* und *datenbankspezifische* Einträge. Über die backend-spezifischen Konfigurationseinträge können Sie Voreinstellungen für alle Datenbanken treffen, die das betreffende Backend verwenden. Die datenbankspezifischen Einträge schließlich beschreiben individuelle Datenbanken.

3.2.2 »Traditionelle« Konfiguration mit slapd.conf

Bei der »traditionellen« Methode steht die Konfiguration für den slapd in einer Datei, die nach Konvention slapd.conf heißt (siehe auch Abschnitt 3.2.1). Um diese Methode zu verwenden, rufen Sie slapd mit der Option -f auf:

```
# slapd -f /etc/ldap/slapd.conf Dateiname kann variieren
```

Syntax Die Datei slapd.conf beginnt mit den globalen Konfigurations-Einstellungen, dann folgen die backend-spezifischen und dann die datenbankspezifischen Einstellungen. backend-spezifische, die sich auf ein bestimmtes Datenbank-Backend bezieht, und dann die datenbankspezifische, die eine konkrete Datenbank beschreibt. In der Datei werden Leerzeilen und Zeilen, die mit »#« anfangen (Kommentarzeilen) ignoriert. Zeilen, die mit Freiplatz anfangen, gelten als Fortsetzung der vorigen Zeile. Konfigurationsdirektiven können Argumente haben, die durch Freiplatz voneinander getrennt werden; Argumente, die Freiplatz enthalten, müssen in doppelte Hochkommas eingefaßt werden ("so wie hier"). Doppelte Hochkommas oder Rückstriche (\) in Attributwerten sollten durch einen Rückstrich versteckt werden.

Globale Direktiven Hier ist eine auszugsweise Liste der wichtigsten globalen Direktiven. Die hier beschriebenen Direktiven beziehen sich auf alle Backends und Datenbanken, wenn nichts anderes angegeben wird.

include *<datei>* Liest an dieser Stelle *<datei>* als Konfigurationsdatei ein, bevor mit der gerade aktuell betrachteten Datei fortgefahren wird. Verschachtelte Aufrufe – include innerhalb einer Datei, die gerade mit include gelesen wird – sind erlaubt, aber man muss selber darauf achten, keine Schleife zu produzieren, indem man aus Datei A Datei B einliest und darin wieder auf A zurückgreift. Eine wichtige Anwendung für diese Direktive besteht im Einlesen von Schemadefinitionen.

objectclass *<RFC4512-Typdefinition>* Definiert eine Objektklasse (siehe [RFC4512])

attributetype *<RFC4512-Typdefinition>* Definiert einen Attributtyp (siehe [RFC4512])

objectidentifizier *<name>* *{<OID>|<name'>[:<Suffix>]}* Definiert *<name>* als Abkürzung für *<OID>*. Sie können *<name>* auch mit einem Suffix der Form »:1.2« verwenden: Nach

```
objectidentifizier lf 1.3.6.1.4.1.15534
```

steht »lf:1.2.3« für den OID 1.3.6.1.4.1.15534.1.2.3. (So etwas darf dann auch als zweiter Parameter in weiteren objectidentifizier-Direktiven auftauchen.)

sizeLimit *<anzahl>*, **timeLimit** *<zeit>* Gibt Grenzen für die Anzahl von Einträgen (sizeLimit) und die maximal zu verwendende Zeit (timeLimit in Sekunden – Echtzeit, nicht CPU-Zeit) für eine Anfrage an. Die Standardwerte sind respektive 500 und 3600.

referral *<url>* Der Verweis, den slapd zurückgeben soll, wenn er für die Anfrage keine lokale Datenbank finden kann. Die Direktive

```
referral ldap://root.openldap.org
```

zum Beispiel verweist nichtlokale Anfragen an den globalen Root-LDAP-Server des OpenLDAP-Projekts. Clients können die Anfrage dann dort noch einmal stellen. (Siehe Abschnitt 1.6 für LDAP-URLs.)

LogLevel *<zahl>* Die *<zahl>* gibt an, welche Art von Ablaufverfolgungs- und Fehler-suchinformation an den syslogd weitergegeben wird. Hierzu muss OpenLDAP mit der Option --enable-debug übersetzt worden sein. Die möglichen Werte finden Sie in Tabelle 3.1, wobei Sie die dort angegebenen Kategorien wie folgt kombinieren können:

- Sie können die numerischen Werte addieren. Der Wert 11 zum Beispiel protokolliert Funktionsaufrufe, Paketbearbeitung und Verbindungsverwaltung. (Das ist auch hexadezimal möglich, statt »11« können Sie also auch »0xb« schreiben.)
- Sie können eine Liste von (dezimalen oder hexadezimalen) Zahlen angeben. Statt »11« können Sie also auch »1 2 8« oder »0x1 0x2 0x8« schreiben.
- Sie können eine Liste von Namen angeben. Statt »11« können Sie also auch »trace packets conns« schreiben.



Diese Werte repräsentieren unabhängige Subsysteme, keine ansteigende Hierarchie der Ausführlichkeit wie bei manchen anderen Programmen.

Der Wert -1 (oder any) protokolliert »alles«. Der Wert 32768 (oder none) läßt nur solche Nachrichten durch, die so wichtig sind, dass sie immer – unabhängig vom eingestellten Wert – ausgegeben werden. Der Wert 0 schaltet die Protokollierung komplett aus.

Tabelle 3.1: loglevel-Werte und dazugehörige Inhalte

Wert	Name	Ausgabeinhalt
1	trace	Funktionsaufrufe
2	packets	Paketbearbeitung
4	args	Umfangreiche Ablaufverfolgung
8	conns	Verbindungsverwaltung
16	BER	Eingegangene und verschickte Pakete ausgeben
32	filter	Suchfilter-Verarbeitung
64	config	Konfigurationsdatei-Verarbeitung
128	ACL	Zugriffskontroll-Listen-Verarbeitung
256	stats	Statistik für Verbindungen, Anfragen und Ergebnisse
512	stats2	Statistik für verschickte Einträge
1024	shell	Kommunikation mit Shell-Backends
2048	parse	Analyse von Einträgen
16384	sync	LDAPSync-Replikation
32768	none	nur Nachrichten, die bei beliebiger Priorität geschrieben werden

Der Standardwert – auch ohne `--enable-debug` – ist 256, womit Verbindungen und andere Tätigkeitsberichte protokolliert werden. OpenLDAP verwendet normalerweise die Syslog-*facility* LOCAL4.

idletimeout *<zeit>* Definiert die Zeit in Sekunden, nach der eine inaktiver Client vom Server abgemeldet wird. Der standardmäßig vorgegebene Wert 0 deaktiviert das zwangsweise Abmelden.

pidfile *<datei>* **und** **argsfile** *<datei>* Diese Direktiven benennen die Dateien, in denen der `slapd` respektive seine Prozess-ID und seine Kommandoargumente ablegt. Erstere wird gebraucht, um den Server kontrolliert stoppen zu können, während letztere möglicherweise bei der Fehlersuche helfen können.

Neben diesen allgemeinen Direktiven gibt es globale Direktiven, die sich mit Zugriffskontrolle beschäftigen; diese werden im Kapitel 4 vorgestellt.

Backend-Direktiven Die Direktiven in diesem Abschnitt beziehen sich nur auf das Backend, für das sie definiert wurden (obwohl jedes Backend sie unterstützt). Die Backend-Direktiven gelten für jede Datenbank, die auf dem betreffenden Backend beruht; Datenbank-Direktiven können sie überschreiben.

backend *<typ>* Wählt ein Backend für die backendspezifischen Direktiven aus. Diverse Werte sind möglich, etwa `bdb` für das BDB-Backend und `ldif` für das LDIF-Backend.

Datenbankspezifische Direktiven Die Direktiven in diesem Abschnitt beziehen sich nur auf die Datenbank, für die sie definiert wurden (und überschreiben gegebenenfalls globale oder backendspezifische Direktiven). Sie werden, falls nicht anders angegeben, von jedem Backend unterstützt.

database *<typ>* Wählt ein Backend für die Datenbank aus. Der *<typ>* ist derselbe wie für die backend-Direktiven.

suffix *<dn-suffix>* Gibt das Suffix von Anfragen an, die an die betreffende Datenbank weitergereicht werden. Es können mehrere `suffix`-Direktiven angegeben werden; jede Datenbankdefinition muss mindestens eine haben.



Wenn das Suffix einer Datenbank im DIT »innerhalb« dessen einer anderen Datenbank liegt (denken Sie an etwas wie `ou=People,dc=example,`

dc=com vs. dc=example,dc=com), dann muss die Datenbank mit dem »enthaltenen« Suffix weiter vorne in der Konfigurationsdatei stehen als die mit dem »enthaltenden«.

readonly {*on* | *off*} Schaltet den Schreibschutz für die Datenbank ein bzw. aus. Schreibzugriffe auf eine schreibgeschützte Datenbank werden mit einer Fehlermeldung abgewiesen.

rootdn <*typ*> Gibt einen Distinguished Name an, für den die Zugriffsrechte oder Antwort-Grenzwerte nicht gelten. Der DN muss nicht (aber kann) auf einen Eintrag im Verzeichnis verweisen. Wenn kein **rootdn** angegeben ist, gibt es keinen entsprechenden DN (es gibt also keinen vordefinierten Standardwert).

rootpw <*kennwort*> Sofern der **rootdn** innerhalb des Namenskontextes der Datenbank liegt (mit anderen Worten, ein **suffix** der Datenbank als Suffix hat), können Sie mit **rootpw** ein Kennwort für den **rootdn** angeben, das immer funktioniert, egal ob tatsächlich ein Eintrag für den **rootdn** in der Datenbank existiert oder dieser ein Kennwort hat. Vom Einsatz dieser Direktive ist abzuraten; man sollte lieber SASL benutzen oder zumindest mit **slappasswd** ein verschlüsseltes Kennwort erzeugen, das man dann hier einträgt.

directory <*dir*> Datenbankdateien für die Datenbank werden in diesem Verzeichnis abgelegt. (Gilt nur für bestimmte Backends, etwa **bdb** oder **ldif**.)

index {<*attrlist*> | **default**} [**pres** | **eq** | **approx** | **sub** | **none**] Gibt an, ob und welche Indizes für welche Attribute in der Datenbank angelegt werden sollen. Jedes Attribut kann für die folgenden Filteroperationen indiziert werden (aber manche Attribute unterstützen nur einen Teil der möglichen Indizes):

pres Test auf Anwesenheit des Attributs in einem Eintrag

eq Vergleich auf Gleichheit

approx Vergleich auf ungefähre Gleichheit

sub Vergleich von Teilzeichenketten

none Keinen Index für dieses Attribut anlegen



Mit dem »Attribut« **default** kann ein Standardsatz von Indizes angelegt werden, der in Kraft tritt, wenn eine **index**-Direktive keine Indexangaben enthält.



Sie sollten immer einen **eq**-Index für das Attribut **objectClass** vorsehen, da sonst die Leistung des Servers sehr leidet.



Den Indextyp **sub** können Sie auch noch weiter unterteilen: **subinitial** indiziert Zeichenketten, die am Anfang eines Attributwerts stehen, **subfinal** solche, die am Ende eines Attributwerts stehen, und **subany** solche, die irgendwo im Attributwert stehen. Für alle drei Geschmacksrichtungen können Sie über globale Direktiven eine Minimal- und eine Maximallänge (in Zeichen) festlegen, die bestimmen, wie groß die Stücke von Zeichenketten sind, die in die betreffenden Indizes eingehen.



Immer wenn Sie die Indexeinstellungen in der Datei **slapd.conf** ändern, müssen Sie die Indizes neu generieren (siehe **slapindex(8)**). (Wenn Sie bei der »modernen« Konfigurationsmethode die Indexkonfiguration dynamisch über LDAP ändern, ist das nicht nötig, da der Server sich dann darum kümmert.)

(**index** gibt es nur für bestimmte Backends, etwa **bdb** oder **hdb**.)

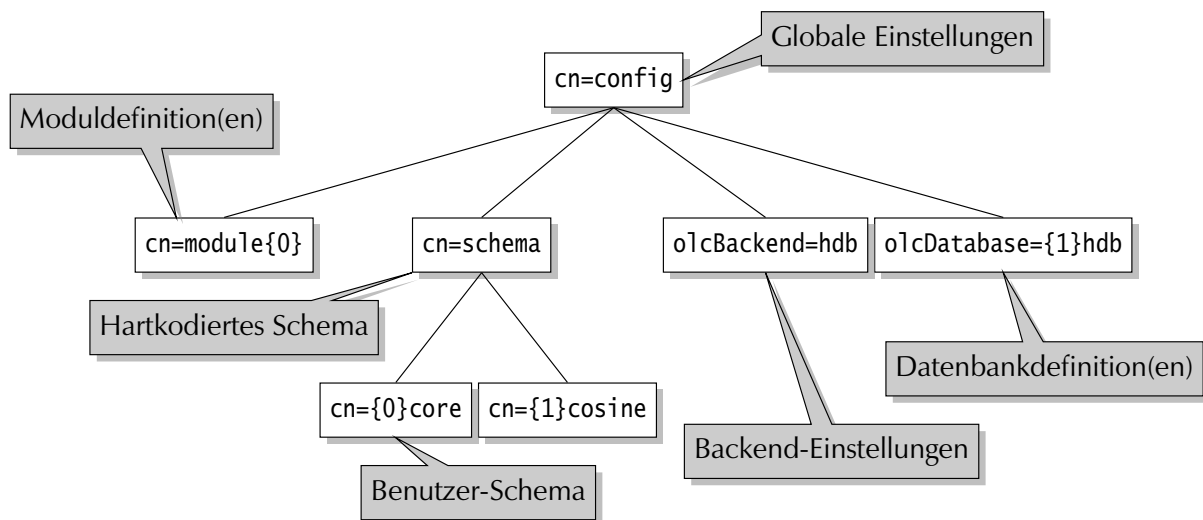


Bild 3.1: DIT für die »moderne« Konfigurationsmethode

3.2.3 »Moderne« Konfiguration über LDAP

In der »modernen« Konfigurationsmethode, die seit OpenLDAP 2.4 unterstützt wird, werden die Konfigurationseinstellungen nicht in einer Konfigurationsdatei abgelegt, sondern im Verzeichnis selbst. Dafür wird das `ldif`-Backend verwendet; die Dateien liegen normalerweise in einem Unterverzeichnis `slapd.d` des OpenLDAP-Konfigurationsverzeichnis (je nach Distribution zum Beispiel `/etc/ldap` oder `/etc/openldap`).

💡 Wenn Sie von Ihrer Distribution mit der modernen Konfigurationsmethode »konfrontiert« werden, ist das erste Problem meistens, herauszufinden, wie Sie auf das Konfigurations-Verzeichnis zugreifen können. Hier ist mitunter das Talent des sagenhaften Freiherrn von Münchhausen gefordert, sich am eigenen Haarschopf aus dem Sumpf zu ziehen. Wir sagen zu diesem Thema am Ende dieses Abschnitts mehr.


Struktur Bild 3.1 zeigt die Struktur des DIT für die Konfiguration. An seiner Wurzel ist der Eintrag `cn=config`, dessen Attribute den Direktiven im »globalen« Bereich der traditionellen Konfigurationsdatei entsprechen. (Die dazugehörige Objektklasse heißt `o1cGlobal`). Die Namen der Attribute ergeben sich dabei aus den Namen der Direktiven mit einem vorgesetzten `o1c` (wie »OpenLDAP Konfiguration«). Aus referral wird also zum Beispiel `o1cReferral`.

💡 Genau nachlesen können Sie das in `slapd-config(5)` oder im *OpenLDAP Administrator's Guide* (Kapitel 5).


Die Einträge unterhalb von `cn=config` entsprechen mehr oder weniger den Abschnitten der Konfigurationsdatei `slapd.conf`. Die etwas merkwürdigen DN's in Bild 3.1 – `module{0}`, `{1}hdb` und so – ergeben sich aus dem Umstand, dass es bei manchen Definitionen auf die Reihenfolge ankommt, ein LDAP-Verzeichnis als solches aber keine definierte »Reihenfolge« für die Kinder eines Eintrags im Baum kennt. Die Zahlen in geschweiften Klammern dienen also dazu, eine solche Reihenfolge zu erzwingen.


💡 Sie können das Problem in den meisten Fällen ignorieren, da OpenLDAP die Nummerierung bei Bedarf selbst hinzufügt – die betreffenden Einträge werden in der Reihenfolge des Anlegens nummeriert.

- `cn=schema,cn=config` (Objektklasse `olcSchemaConfig`) enthält die Schemadefinitionen (die Sie sonst im Unterverzeichnis `schema` finden würden). Die Attribute dieses Eintrags selbst entsprechen den in OpenLDAP hart kodierten Definitionen und sind darum in der LDIF-Datei mit der Konfiguration nicht zu finden; OpenLDAP gibt sie bei Bedarf selbst aus.
- Kinder von `cn=schema,cn=config` entsprechen den Schemadateien im Unterverzeichnis `schema`. Statt den Direktiven `attributetype` und `objectclass` verwenden Sie hier die Attribute `olcAttributeTypes` und `olcObjectClasses` (beachten Sie das »s« am Schluss).


 Hier zum Beispiel kommt die oben angesprochene Nummerierungsmethode für die Einträge zum Tragen.

- `olcBackend=...`-Einträge (Objektklasse `olcBackendConfig`) dienen theoretisch als Ersatz für die »backendspezifischen« Abschnitte von `slapd.conf`. Wir sagen hier mit Absicht »theoretisch«, weil die Implementierung dieser Einträge in OpenLDAP noch auf sich warten lässt. Aus diesem Grund werden Sie sie in der Praxis auch nicht finden.
- `olcDatabase=...`-Einträge (Objektklasse `olcDatabaseConfig`) definieren einzelne Datenbanken, in Analogie zu den `database/suffix`-Abschnitten in der Datei `slapd.conf`. Der DN ergibt sich dabei aus dem Datenbanktyp (etwa `hdb` oder `config`) und einer vorgesetzten Nummer zur Vermeidung von Mehrdeutigkeiten und für die richtige Reihenfolge.

 Die Datenbank `olcDatabase=config,cn=config` müssen Sie nicht explizit definieren, da OpenLDAP sie automatisch als erste anlegt. Allerdings hält Sie niemand davon ab, in einem entsprechenden Konfigurationseintrag Voreinstellungen dafür zu machen.


 Für `olcDatabase=config,cn=config` gilt das »Highlander-Prinzip« (»Es kann nur eine geben«), und darum wird keine Nummer gebraucht.

Wenn Sie Voreinstellungen für *alle* Datenbanken treffen wollen – die `olcBackend`-Einträge haben wir ja weiter oben als zahnlose Papiertiger entlarvt –, können Sie dafür die spezielle Datenbank `olcDatabase=frontend,cn=config` benutzen. Attribute, die Sie in diesem Eintrag definieren, gelten für alle anderen Datenbanken, sofern diese das betreffende Attribut nicht selber definieren.

 Die frontend-Datenbank muss neben der allgemeinen Objektklasse `olcDatabaseConfig` die Objektklasse `olcFrontendConfig` haben. Darum müssen Sie sich aber nicht selber kümmern, weil OpenLDAP diese Datenbank genau wie die `config`-Datenbank implizit anlegt.

Die Datenbank-Backends BDB und HDB haben ihre eigenen Objektklassen `olcBdbConfig` und `olcHdbConfig`. Wenn Sie zum Beispiel eine HDB-Datenbank konfigurieren, können Sie mit den Attributen von `olcHdbConfig` Einstellungen vornehmen, die spezifisch für das HDB-Backend sind und darum in `olcDatabaseConfig` nicht zur Verfügung stehen. Das sieht (in LDIF) dann etwa so aus:

<pre>dn: olcBackend={1}hdb,cn=config objectClass: olcDatabaseConfig objectClass: olcHdbConfig olcSuffix: dc=example,dc=com olcDbIndex: default pres,eq</pre>	<p><i>Allgemeine Einstellung</i> <i>HDB-spezifische Einstellung</i></p>
--	---

 A propos `olcDbIndex`: Ein Vorteil der »modernen« verzeichnisbasierten Konfiguration ist, dass Änderungen an der Indexkonfiguration

einer Datenbank von OpenLDAP direkt ausgeführt werden, sofern sie im laufenden Betrieb stattfinden. OpenLDAP erledigt das im Hintergrund. Nur wenn Sie OpenLDAP anhalten, bevor die Reindexierung abgeschlossen ist, müssen Sie die Indizes mit `slapindex` neu aufbauen.

Arbeiten mit der Konfiguration Die Frage, die sich aufdrängt, ist natürlich die nach einer bequemen Möglichkeit, diese im LDAP-Verzeichnis vergrabenen Konfigurationseinstellungen zu manipulieren. Hierzu können wir leider auch nichts sagen außer »`ldapsearch`, `ldapadd` und `ldapmodify` sind Ihre Freunde.«



Eventuell werden Sie eine gewisse Versuchung verspüren, die LDIF-Dateien in `slapd.d` direkt mit einem Texteditor zu ändern. Der *OpenLDAP Administrator's Guide* rät davon mit großem Nachdruck ab (das operative Wort ist "never"). Unter Umständen können Sie es sich unserer Erfahrung nach in gewissen Situationen leisten, wenn der `slapd` gerade nicht läuft (womit natürlich einer der wesentlichen Vorteile der verzeichnisbasierten Konfiguration wegfällt). Sollten Sie Probleme kriegen, werden wir allerdings vehement abstreiten, das je behauptet zu haben.

Mitunter besteht das erste Problem darin, überhaupt Zugriff auf die Konfiguration zu bekommen. Wie damit umgegangen wird, hängt von der Distribution ab, die Sie verwenden.



Die Red-Hat-artigen Distributionen lassen Sie hier eiskalt im Regen stehen: Sie brauchen für den Zugriff auf das Konfigurations-Verzeichnis ein Kennwort (Attribut `olcRootPW` im Eintrag `olcDatabase={0}config,cn=config`, aber das ist dort nicht »ab Werk« definiert – Henne-Ei-Problem! Hier hilft nur das beherzte Eingreifen mit einem Texteditor (auch wenn wir gerade gesagt haben, dass Sie das eigentlich nicht sollen). Besorgen Sie sich ein Kennwort in verschlüsselter Form mit `slappasswd`:

```
$ slappasswd
New password: geheim
Re-enter password: geheim
{SSHA}5nPwbpkzyk/0LCcJmpd7Mj4NH0aJJU5k
```

Die letzte Zeile der Ausgabe von `slappasswd` hängen Sie dann als

```
olcRootPW: {SSHA}5nPwbpkzyk/0LCcJmpd7Mj4NH0aJJU5k
```

an die Datei `/etc/ldap/slapd.d/cn=config/olcDatabase={0}config.ldif` an.



Bei Debian GNU/Linux & Co. haben Sie es dagegen gut: Die Voreinstellung dort erlaubt den Zugriff als `root` über ein Unix-Domain-Socket auf demselben Rechner. Verwenden Sie einfach Kommandos wie



```
# ldapsearch -Y EXTERNAL -H ldapi:/// -b "cn=config"
```

zum Durchsuchen der Konfiguration oder

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f <LDIF-Datei>
```

zum Ändern von Einträgen im Verzeichnis. (Es hält Sie natürlich niemand davon ab, auf diesem Weg ein `olcRootPW` zu etablieren.)

Konvertieren Das OpenLDAP-Team hat sehr eindeutig gesagt, dass die »moderne« Konfiguration auf Verzeichnisbasis der Zug der Zukunft ist und die »traditionelle« Methode mit `slapd.conf` verschwinden wird¹. Sie machen also keinen Fehler, wenn Sie sich frühzeitig an die neue Methode gewöhnen. Sollten Sie schon eine `slapd.conf`-Datei für Ihren OpenLDAP-Server haben, die das tut, was sie soll, dann können Sie sie bequem ins neue Format konvertieren. Vorher sollten Sie jedoch sicherstellen, dass der Server Sie auf das Konfiguration in `cn=config` zugreifen läßt. Dazu sollten Sie an Ihre `slapd.conf`-Datei etwas anhängen wie

```
database config
rootpw geheim
```

Später können Sie sich beim Konfigurations-Verzeichnis dann mit dem Root-DN `cn=config` und dem Kennwort `geheim` authentisieren.



Das ist natürlich nichts für »auf Dauer«. Bei der nächsten Gelegenheit sollten Sie entweder SASL-basierte Authentisierung für `cn=config` konfigurieren (Abschnitt 4.2 erklärt, wie) oder das Verzeichnis nur für den lokalen Rechner zugänglich machen (oder beides).

Die eigentliche Konvertierung übernimmt Kommando `slaptest`:

```
# slaptest -f /etc/ldap/slapd.conf -F /etc/ldap/slapd.d
```

generiert eine mit `/etc/ldap/slapd.conf` identische verzeichnisbasierte Konfiguration in `/etc/ldap/slapd.d`. Sie können `slapd.conf` dann entfernen (wir würden zunächst »umbenennen« vorschlagen, man weiß ja nie) und den `slapd` mit der neuen Konfiguration starten.



Wie Sie Ihren Server dazu bringen, tatsächlich die verzeichnisbasierte Konfiguration zu benutzen, hängt von Ihrer Distribution ab. Manche Distributionen prüfen nach, welche Methode Sie verwenden, bei anderen müssen Sie einen expliziten Schalter umlegen.



Bevor Sie jetzt voller Elan Ihre Konfiguration ändern, sollten Sie prüfen, ob alle Ihre Overlays und Datenbank-Backends mit der »modernen« Konfigurationsmethode zurechtkommen. Inzwischen sind die allermeisten portiert, aber kleine »weiße Flecken« auf der Landkarte der verzeichnisbasierten Konfiguration existieren möglicherweise noch.

3.3 Ein Konfigurationsbeispiel

Hier ist eine mehr oder weniger komplette (wenn auch nicht besonders komplexe) Konfiguration eines LDAP-Servers.

```
# Beispiel-Konfiguration -- Globale Direktiven
include /etc/ldap/schema/core.schema
referral ldap://root.openldap.org
access to * by * read
```

In `/etc/ldap/schema/core.schema` sind diverse Attributtypen und Objektklassen aus den LDAPv3-RFCs und anderen ausgewählten Quellen hinterlegt, die man so nicht selber definieren muss. Die `referral`-Direktive leitet nichtlokale Anfragen an den Root-Server weiter (Sie sollten das erst einschalten, *nachdem* Sie den lokalen Verzeichnisdienst zum Laufen gebracht haben); die `access`-Direktive (siehe Kapitel 4) läßt beliebige Lesezugriffe auf das Verzeichnis zu.

¹Man muss das durchaus ernst nehmen. Die kennen da nichts.

```
# Datenbank für example.com
database hdb oder bdb
suffix "dc=example,dc=com"
directory /var/lib/ldap
rootdn "cn=Manager,dc=example,dc=com"
rootpw {SHA}kGByAB793z4R5tK1eC9Hd/4Dhzk=
index uid pres,eq
index cn,sn pres,eq,sub
index objectClass eq
```

Die Datenbank ist zuständig für Anfragen, die sich auf DN's mit dem Suffix `dc=example,dc=com` beziehen. Die Daten liegen in `/var/lib/ldap`. Der Benutzer `Manager@example.com` hat freien Zugriff mit dem Kennwort `geheim`, das hier in SHA-Form vorliegt. Die `index`-Direktiven indizieren die Attribute `uid`, `cn`, `sn` und `objectClass`.

Übungen



3.1 [!3] Konfigurieren Sie den `slapd` so, dass er in Anlehnung an das gezeigte Beispiel eine Datenbank verwaltet (prüfen Sie, welche Backends Ihre `slapd`-Implementierung unterstützt). Importieren Sie einen kleinen DIT auf der Basis einer LDIF-Datei.



3.2 [!2] Benutzen Sie `ldapsearch`, um sich davon zu überzeugen, dass Ihr LDAP-Server das Verzeichnis tatsächlich anbietet. Lassen Sie sich zum Beispiel den kompletten Inhalt des Verzeichnisses auflisten oder suchen Sie nach einzelnen Einträgen oder Einträgen, die alle ein gemeinsames Merkmal haben.



3.3 [2] Was ist der Unterschied zwischen den `ldapsearch`-Optionen `-L`, `-LL`, `-LLL` sowie einer Abfrage ganz ohne diese Optionen? Probieren Sie die verschiedenen Möglichkeiten aus, bevor Sie in der Dokumentation nachschlagen.

3.4 OpenLDAP-Betrieb

Ein einmal installiertes Verzeichnis auf der Basis von OpenLDAP kann mit den in Kapitel 2 gezeigten Kommandos mit Daten beschickt, modifiziert usw. werden.

Für Spezialanwendungen gibt es auch noch ein paar OpenLDAP-spezifische Kommandos, die direkt auf die Datenbank zugreifen:

`slapcat`² dient dazu, eine komplette `slapd`-Datenbank auf die Standardausgabe oder in eine Datei zu schreiben. Dieses Kommando ist nicht mit `ldapsearch` zu vergleichen – es liefert nicht nur die »normalen« Attribute der Einträge, sondern auch diverse zusätzliche Attribute, die `slapd` zur internen Verwaltung der Daten benötigt. Ferner erscheinen die Einträge nicht in der für `ldapadd` nötigen Reihenfolge, bei der die »Wurzel« des DIT zuerst kommt, sondern in der Reihenfolge, in der sie in der Datenbank stehen (was nicht notwendigerweise dasselbe ist). `slapcat` dient demnach eher zum Anlegen von Sicherheitskopien oder zum Übertragen einer kompletten Datenbank auf einen anderen Rechner als zum Lesen und Modifizieren einzelner Einträge (wofür `ldapsearch` sicherlich die bessere Option darstellt).



Während `ldapsearch` über das Netz mit entfernten LDAP-Servern reden kann, müssen Sie `slapcat` auf dem Rechner aufrufen, wo auch der `slapd` läuft, dessen Datenbank Sie lesen möchten.



Wenn Sie sichergehen wollen, konsistente Daten zu erhalten, dann halten Sie Ihren `slapd` vorher an. Von `bdb`- und `hdb`-Datenbanken können Sie aber auch im laufenden Betrieb konsistente Abzüge machen.

²An dieser Stelle möchten wir darauf hinweisen, dass wir jede Form von Tierquälerei ablehnen.

Einen mit `slapcat` erzeugten Abzug einer Datenbank können Sie mit `slapadd` wieder einspielen – entweder auf demselben Server oder einem anderen (der dann passend vorkonfiguriert sein muss). Hier sind die Kommandos, die nötig sind, um eine `slapd`-Datenbank auf einen anderen Rechner zu verschieben:

```
# slapcat | ssh openldap@ldap.example.com slapadd
```



Beachten Sie, dass nach einem `slapadd`-Lauf die erzeugten Dateien dem Benutzer gehören, mit dessen Identität `slapadd` gestartet wurde. Das ist nicht notwendigerweise derselbe Benutzer, dessen Rechte der `slapd` verwendet. Sie müssen also entweder vor dem `slapadd`-Aufruf die Identität des `slapd`-Benutzers annehmen oder nach dem `slapadd`-Lauf die neu angelegten Dateien diesem Benutzer überschreiben.



Sie sollten `slapadd` nicht aufrufen, während der `slapd` läuft – jedenfalls wenn Ihnen die Integrität Ihrer Datenbanken etwas wert ist.

3.5 Weiterleitung

Ein LDAP-Server kann theoretisch den kompletten DIT für eine Organisation vorhalten. Oft ist es aber sinnvoll, einerseits gewisse Teile (Teilbäume) des DIT auf andere Server zu verlagern – etwa um die Administration zu erleichtern –, und andererseits einen LDAP-Server so in eine »globale« Hierarchie einzubinden, dass er Anfragen nach Daten außerhalb seines eigenen DIT an einen anderen Server weiterreicht. OpenLDAP unterstützt beide Arten der Weiterleitung.

Die einfachere von beiden ist die Weiterleitung von Anfragen außerhalb des eigenen DIT. Beispielsweise betreut ein Server gemäß seinem `suffix`-Eintrag in der `slapd.conf`-Datei den Teilbaum unterhalb des Objekts `dc=example,dc=com`. Wird er nun nach einem Eintrag in (beispielsweise) `dc=server,dc=org` gefragt, kann er einen anderen LDAP-Server vorschlagen, der möglicherweise Informationen über jenen Teilbaum vorhält. Dies geschieht, wie schon weiter vorne angedeutet, über eine Direktive der Form

Weiterleitung »nach oben«

```
olcReferral: ldap://other.server.org
```

im `cn=config`-Eintrag der Konfiguration. LDAP-Clients erhalten dann statt der üblichen Antworten sogenannte *referrals*, die sie anweisen, ihre Anfragen an den benannten Server zu schicken. Dieser hilft ihnen dann (hoffentlich) weiter.

Das Verfolgen von Weiterleitungen im LDAP ist tatsächlich Sache der Clients. `ldapsearch` zum Beispiel verfolgt standardmäßig keine Weiterleitungen; dies muss durch die Option `-C` explizit eingeschaltet werden. Da `ldapsearch` gerne zur Diagnose von Problemen mit LDAP-Verzeichnissen eingesetzt wird, ist das eine vernünftige Vorgehensweise; andere LDAP benutzende Applikationen gehen statt dessen möglicherweise davon aus, Weiterleitungen immer automatisch zu verfolgen.



Merken Sie sich die Option `-C` (wie *chase referrals*) gut oder haben Sie beim Arbeiten mit LDAP diese Unterlage zum Nachschlagen parat. In aktuellen Distributionen werden Sie nach dieser Option nämlich in den Handbuchsseiten von `ldapsearch & Co.` vergeblich suchen.

Wenn Sie Teile des Verzeichnisbaums auf andere LDAP-Server auslagern möchten, müssen Sie die Übergänge explizit bekannt machen. Auf dem übergeordneten Server schreiben Sie dafür ein spezielles Objekt der Objektklasse `referral` mit dem Basis-DN des ausgelagerten Teilbaums ins Verzeichnis. Meist fügt man auch noch die Objektklasse `extensibleObject` hinzu, damit relative DNs funktionieren. Sollen zum Beispiel vom Server `ldap.example.com` Anfragen nach Einträgen im Teilbaum unter `ou=sub,dc=example,dc=com` an den Server `subldap.example.com` delegiert werden, könnte der entsprechende Eintrag auf `ldap.example.com` so aussehen:

```
dn: ou=sub,dc=example,dc=com
objectClass: referral
objectClass: extensibleObject
ou: sub
ref: ldap://subldap.example.com/ou=sub,dc=example,dc=com
```

Sinnvollerweise sollte subldap.example.com dann eine Weiterleitung »nach oben« der Form

```
# Eintrag: cn=config
olcReferral: ldap://ldap.example.com
```

enthalten, damit Anfragen außerhalb des ou=sub-Teilbaums an den übergeordneten Server weitergereicht werden.



Dieses Beispiel illustriert einen der seltenen Fälle, in denen die Objektklasse `extensibleObject` verwendet werden muss. Da der relative DN des anzulegenden Objekts `ou=sub` ist, muss in der Definition des Objekts ebenfalls ein Attribut namens `ou` (oder, für Gernschreiber, `organizationalUnitName`) enthalten sein, dem der Wert `sub` zugewiesen wird (frühere OpenLDAP-Versionen hätten sich da nicht so kleinlich angestellt). Ohne die Verwendung von `extensibleObject` wäre das nur dann möglich, wenn das Objekt selbst auch die Klasse `organizationalUnit` besäße. Da das Objekt aber nur eine Weiterleitung darstellt und selbst keine Informationen enthalten soll, ist das sinnlos. `extensibleObject` erlaubt Ihnen hier also, den Namensanforderungen Genüge zu tun und trotzdem keine überflüssigen Informationen im Objekt zu hinterlegen.

Einrichten, ändern und löschen können Sie solche Einträge mit den LDAP-Kommandozeilenwerkzeugen. Hierbei sollten Sie die Option `-M` angeben, damit der OpenLDAP-Server weiß, dass tatsächlich der Weiterleitungseintrag gemeint ist, und keine Weiterleitung an den untergeordneten Server erzeugt:

```
$ ldapmodify -M -x -D "cn=Manager,dc=example,dc=com" -W -f referral.ldif
```

Übungen



3.4 [3] Konfigurieren Sie Ihren LDAP-Server so, dass ein Teilbaum seines DIT auf einen anderen LDAP-Server ausgelagert wird. (Dieser andere LDAP-Server kann der eines anderen Schulungsteilnehmers sein, ein Server in einer anderen virtuellen Maschine auf Ihrem Rechner, oder Sie starten auf Ihrem Rechner eine zweite Ausgabe von `slapd`, die auf einem anderen Port auf Anfragen wartet. Wenn Sie `slapd` zum Beispiel mit

```
# /usr/sbin/slapd -h ldap://127.0.0.1:10389/ -F /etc/ldap/slapd-0.d
```

aufrufen, wird ein zweiter LDAP-Server gestartet, der auf Port 10389 lauscht und seine Konfiguration aus `/etc/ldap/slapd-0.d` liest. Dieser Server ist völlig unabhängig vom ersten.) Installieren Sie auf Ihrem Server ein entsprechendes `referral`-Objekt und überzeugen Sie sich mit `ldapsearch` bzw. `ldapsearch -C`, dass die Weiterleitung tatsächlich erfolgt.



3.5 [3] Sorgen Sie auf dem »untergeordneten« Server dafür, dass Anfragen, die dieser nicht beantworten kann, an Ihren ursprünglichen Server weitergeleitet werden, und testen Sie auch das.

Kommandos in diesem Kapitel

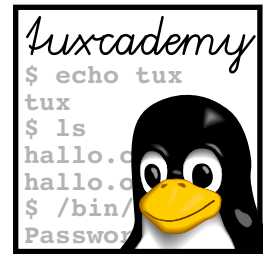
slapadd	Importiert eine OpenLDAP-Datenbank in einen Server	slapadd(8)	42
slapcat	Gibt die Datenbank eines OpenLDAP-Servers aus	slapcat(8)	42
slapd	Der Teil des OpenLDAP-Servers, der LDAP-Anfragen beantwortet	slapd(8)	32
slappasswd	Bestimmt Kennwörter für den Gebrauch mit OpenLDAP	slappasswd(1)	37

Zusammenfassung

- OpenLDAP-Server können über Weiterleitungen in eine globale Verzeichnishierarchie eingebunden werden.

Literaturverzeichnis

- RFC4512** J. Sermersheim, K. Zeilenga. »Lightweight Directory Access Protocol (LDAP): Directory Information Models«, Juni 2006.
<http://www.ietf.org/rfc/rfc4512.txt>



4

OpenLDAP-Sicherheit

Inhalt

4.1	Zugriffsregeln für Verzeichnisdaten	48
4.1.1	Zugriffsrechte	48
4.1.2	Direktiven	49
4.1.3	Auswertung von Zugriffsregeln	50
4.1.4	Beispiele für Zugriffsregeln	50
4.1.5	Tipps und Tricks	52
4.2	OpenLDAP und SASL	53
4.2.1	SASL-Grundlagen	53
4.2.2	OpenLDAP mit SASL-Authentisierung	55
4.2.3	Testen der Konfiguration	58
4.2.4	Proxy-Authentisierung	58
4.3	OpenLDAP und TLS.	60
4.3.1	Grundlagen.	60
4.3.2	Server-Zertifikate.	61
4.3.3	Client-Zertifikate	62

Lernziele

- Zugriffskontrolle auf Daten in einem LDAP-Verzeichnis verstehen und einsetzen können
- Authentisierung von LDAP-Benutzern über SASL konfigurieren können
- Zugang zu einem LDAP-Server über TLS mit Server- und ggf. Client-Authentisierung konfigurieren können

Vorkenntnisse

- Installation und Konfiguration von OpenLDAP
- Grundkenntnisse über Kryptographie, Authentisierung und Verschlüsselung

Tabelle 4.1: Zugriffsrechte in LDAP

Recht	Code	Kombination	Bedeutung
none	0	=0	Kein Zugriff
disclose	d	=d	Ausgabe von Daten in Fehlermeldungen
auth	x	=xd	Verbinden mit dem Verzeichnis erlaubt
compare	c	=cxd	Vergleiche sind erlaubt
search	s	=scxd	Anwendung von Suchfiltern ist erlaubt
read	r	=rscxd	Suchergebnisse dürfen gelesen werden
add	a		Einträge oder Attribute dürfen hinzugefügt werden
delete	z		Einträge oder Attribute dürfen gelöscht werden
write	w	=wrscxd	Einträge dürfen geändert/umbenannt werden
manage	m	=mwrscxd	Einträge dürfen verwaltet werden

4.1 Zugriffsregeln für Verzeichnisdaten

4.1.1 Zugriffsrechte

Zugriffsrechte Es ist möglich, den Zugriff auf ein LDAP-Verzeichnis ziemlich detailliert zu regeln. Auf Einträge oder (eher) Gruppen von Einträgen können Sie verschiedene Zugriffsrechte vergeben. Diese Rechte sind wie folgt definiert:

none schließt alle Zugriffe aus.

disclose schließt alle Zugriffe aus, bis darauf, dass Fehlermeldungen die betroffenen Daten enthalten dürfen.

auth erlaubt den Zugriff auf ein Attribut zu Authentisierungs- und Autorisierungszwecken, ohne dass andere Zugriffe erlaubt sind. So können Sie anonymen Clients das geringstmögliche Zugriffsrecht auf empfindliche Daten wie etwa Kennwörter geben können, etwa um die `bind`-Operation zu erlauben.

compare erlaubt Vergleichsoperationen, indem das anfragende Programm dem Verzeichnis einen Wert vorlegt und nur das Ergebnis eines Vergleichs dieses Werts mit dem korrespondierenden Attribut im Eintrag als Antwort bekommt. Auf diese Weise können zum Beispiel Kennwörter geprüft werden, ohne dass das anfragende Programm Zugriff auf die tatsächlichen Kennwörter im Verzeichnis bekommt (nicht mal in verschlüsselter Form).

search gestattet die Anwendung von Suchfiltern.

read erlaubt das Lesen von Attributwerten.

write ist eigentlich die Kombination der Zugriffsrechte `add` und `delete`. `add` gestattet das Hinzufügen und `delete` das Löschen von Attributen und Einträgen. (Ersetzen ist Löschen gefolgt von Hinzufügen.)

manage erlaubt alles (auch administrativen Zugriff).

Jedes dieser Rechte impliziert die weiter oben genannten. Ein Überblick steht in Tabelle 4.1.

Tabelle 4.1 enthält außerdem auch noch die »Rechtecodes«, die Sie verwenden können, um gezielt bestimmte Rechte zu setzen oder zu löschen. Dies funktioniert ähnlich wie beim Linux-Kommando `chmod`: Jedes Recht hat einen Codebuchstaben, und Sie können beliebige Kombinationen dieser Buchstaben mit einem vorgesetzten »+« verwenden, um *genau* diese Rechte zu aktivieren. Ein vorgesetztes »+« fügt die betreffenden Rechte zu denen hinzu, die vielleicht schon aufgrund anderer Regeln existieren, während ein vorgesetztes »-« die betreffenden Rechte von den geltenden entfernen.



Die dritte Spalte in der Tabelle gibt an, welche Rechte aktiviert werden, wenn das Recht über seinen Namen (und nicht den Codebuchstaben) angesprochen wird.



Die gezielte Vergabe einzelner Rechte kann zum Beispiel nützlich sein, wenn die Kommunikation mit dem LDAP-Server unverschlüsselt erfolgt. Selbst wenn die Kennwörter selbst (etwa mit SASL) nicht übertragen werden, gehen die Nutzdaten im Klartext über das Netz. Hat ein Benutzer Schreibrecht auf sein Kennwort, dann darf er es üblicherweise auch lesen; das Kennwort wird also bei einer entsprechenden Anfrage im Klartext übermittelt. Die Rechtekombination »=wscx« verhindert das und verringert so die Gefahr, dass Unbefugte im Netz das Kennwort erhaschen können.

4.1.2 Direktiven

Zur Vergabe von Zugriffsrechten werden in der Datei `slapd.conf` die `access-` und `defaultaccess-`Direktiven benutzt. Diese Direktiven können im globalen Teil der Konfiguration stehen, aber auch durch neue Einstellungen im backend- oder datenbankspezifischen Teil selektiv überschrieben werden:

access to *<was>* [**by** *<wer>* *<Recht>* *<control>*]+ Hiermit wird der Zugriff auf eine Menge von Einträgen oder Attributen (angegeben durch *<was>*) für gewisse Fragesteller (*<wer>*) gemäß *<Recht>* eingeschränkt oder freigegeben

defaultaccess *<Recht>* Gibt die standardmäßig vergebenen Zugriffsrechte für den Fall an, dass keine `access-`Direktiven angegeben sind. Standardwert ist `read`.

Die `access-`Direktive ist kompliziert genug, dass es sich lohnt, sie genauer anzuschauen. Zunächst wird festgelegt, für was der Zugriff geregelt wird (der *<was>*-Ausdruck). *<was>* ist im einfachsten Fall ein DN, der für einen Eintrag, dessen unmittelbare Kinder oder dessen Unterbaum steht, etwa so:

Zugriff auf was?

<code>access to dn.base="dc=example,dc=com" ...</code>	<i>Nur das Objekt</i>
<code>access to dn.one="dc=example,dc=com" ...</code>	<i>Unmittelbare Kinder</i>
<code>access to dn.subtree="dc=example,dc=com" ...</code>	<i>Unterbaum</i>
<code>access to dn.children="dc=example,dc=com" ...</code>	<i>... ohne Basisobjekt</i>



Der Vergleich von DN's erfolgt gemäß `distinguishedNameMatch` aus [RFC4517]. Frühere OpenLDAP-Versionen verlangten hier »normalisierte« DN's (siehe z. B. [RFC2253]), aber das ist heute nicht mehr relevant.

Sie können auch einen regulären Ausdruck für die betreffenden DN's angeben wie in

```
access to dn="^.*,dc=example,dc=com$" <<<<<<
```

oder einen Suchfilter nach [RFC4515] verwenden:

```
access to filter=(objectClass=sambaSamAccount) ...
```

Weiter strukturieren läßt sich der Zugriff auf einzelne Attribute mit einer komma-separierten Liste von Attributnamen:

```
access to dn=<<<<<<
  attrs=sn,telephoneNumber <<<<<<
```

Mit dem speziellen Selektor `*` können Sie alle Einträge auswählen.

Selbstverständlich lassen sich diese Möglichkeiten auch miteinander kombinieren. Das folgende Beispiel regelt den Zugriff auf die Kennwort-Attribute aller Samba-Benutzerkonten im Teilbaum der Marketing-Abteilung:

```
access to dn.children="ou=Marketing,dc=example,dc=com"
  attrs=userPassword,ntPassword,lmPassword
  filter=(objectClass=sambaSamAccount) <<<<<
```

Zugriff durch wen? Die *by- $\langle wer \rangle$* -Klausel regelt, wer auf den Eintrag zugreifen darf. Die folgenden Angaben sind unter anderem möglich:

*	Alle Benutzer
anonymous	Anonyme (nicht authentifizierte) Benutzer
users	Authentifizierte Benutzer
self	Der Benutzer, dem der Eintrag gehört
dn= $\langle regex \rangle$	Benutzer, die auf den regulären Ausdruck passen
domain= $\langle regex \rangle$	Benutzer von einer Domain, die auf $\langle regex \rangle$ passt
dnattr= $\langle attr \rangle$	Benutzer, deren DN in Attribut $\langle attr \rangle$ vorkommt

Neben diesen Beispielen ist auch die Selektion aller Benutzer eines Teilbaums oder einer bestimmten Ebene im Baum möglich. Die Syntax dafür ist dieselbe, die bereits oben bei der Objektauswahl vorgestellt wurde.

4.1.3 Auswertung von Zugriffsregeln

Vergabe von Zugriffsrechten Bei der Vergabe von Zugriffsrechten geht `slapd` wie folgt vor: Zuerst werden Einträge und ggf. Attribute mit der *was- $\langle was \rangle$* -Klausel verglichen. Für jeden Eintrag gelten zunächst die durch datenbankspezifische Direktiven vorgegebenen Rechte und dann die globalen Zugriffsrechte (Einträge, die in keiner Datenbank stehen, werden gemäß der Zugriffsrechte in der ersten definierten Datenbank behandelt). Innerhalb dieser Rangfolge werden die *access*-Direktiven eine nach der anderen angeschaut, in der Reihenfolge, wie sie in der Konfigurationsdatei stehen. Die erste Direktive, die auf den Eintrag bzw. das Attribut passt, gilt.

Als nächstes wird der zugreifende Benutzer (bzw. das zugreifende Programm) mit den *by- $\langle wer \rangle$* -Klauseln der *access*-Direktive verglichen, in der Reihenfolge des Auftretens. Dadurch ergibt sich die Art des Zugriffsrechts auf den Eintrag bzw. das Attribut. Zum Schluss wird die gewünschte Art des Zugriffs mit dem Zugriffsrecht verglichen; schließt das Zugriffsrecht die Art des Zugriffs ein, wird der Zugriff erlaubt, sonst verboten.

Reihenfolge Aus dieser Beschreibung folgt, dass spezifischere Zugriffsregeln in der Konfigurationsdatei *vor* allgemeineren stehen sollten. Ansonsten würden die allgemeineren zuerst gefunden und die spezifischeren gar nicht verwendet. Dasselbe gilt für *by- $\langle wer \rangle$* -Klauseln in Zugriffsregeln.

4.1.4 Beispiele für Zugriffsregeln

Hier sind einige Beispiele für Zugriffsregeln zur Illustration:

```
access to * by * read
```

gibt allen Benutzern Leserecht.

```
access to *
  by self write
  by anonymous auth
  by * read
```

Diese Regel erlaubt Benutzern, ihre eigenen Einträge zu ändern, gestattet die Authentisierung (die immer »anonym« stattfindet) und erlaubt allen anderen Benutzern das Lesen. Nur der erste passende Zugriff gilt; die letzte Klausel hätte also auch »by users read« heißen können.

```
access to dn.subtree="dc=example,dc=com"
  by * search
access to dn.subtree="dc=com"
  by * read
```

Einträge im Teilbaum unter `dc=com` dürfen gelesen werden, bis auf die unter `dc=example,dc=com`, die nur durchsucht werden dürfen. Auf `dc=com` selbst darf nicht zugegriffen werden, da keine der beiden Regeln auf diesen DN passt. Wäre die Reihenfolge der Direktiven umgekehrt, dann würde die `dc=example,dc=com`-Regel nicht beachtet, da alle solchen Einträge schon von der `dc=com`-Regel abgedeckt werden.

Es ist wichtig, anzumerken, dass jede Regel implizit mit »by * none« und jede Liste von Regeln implizit mit »access to * by * none« aufhört. Das heißt, dass alle Zugriffe *abgewiesen* werden, auf die keine Regel passt.



Wie wir schon früher angedeutet haben, wird der Zugriff auf Kennwort-Attribute gerne beschränkt:

```
access to dn.children="dc=example,dc=com"
  attrs=userPassword,ntPassword,lmPassword
  by self write
  by anonymous auth
  by * none
```

Wenn das Ihre einzige Zugriffsregel ist, werden Sie schnell merken, dass sich der LDAP-Server sehr eigenartig verhält, gewünschte Daten nicht mehr ausgibt und auch sonst kaum zu gebrauchen ist. Schuld daran ist der implizite Abschluss der Regelliste durch

```
access to * by * none
```

der für alle *nicht* in der Zugriffsregel erwähnten Attribute gilt.

Der defaultaccess tritt nur in Kraft, wenn keine access-Regeln angegeben wurden.

Manchmal soll ein DN sich selbst einem Attribut hinzufügen oder aus ihm entfernen können. Wenn es zum Beispiel erlaubt sein soll, dass Benutzer sich in eine Gruppe aufnehmen oder aus ihr entfernen, läßt sich das mit einer Direktive wie

```
access to attrs=member,entry
  by dnattr=member selfwrite
```

erreichen. Die `dnattr`-Klausel sagt, dass die Zugriffsregel auf Benutzer gilt, deren DN im `member`-Attribut vorkommt. Der `selfwrite`-Zugriff bedeutet, dass solche Benutzer nur ihren eigenen DN in das Attribut aufnehmen oder ihn daraus entfernen dürfen, keine anderen. `entry` ist ein besonderes Attribut, das Zugriff auf den *Eintrag* einräumt.

Übungen



4.1 [10] Definieren Sie eine Zugriffsregel, mit der Benutzer in ihrem eigenen LDAP-Eintrag (ausschließlich) den Wert des Attributs `favouriteDrink` ändern dürfen. Alle anderen Attribute sollen schreibgeschützt sein. (Das `favouriteDrink`-Attribut wird in der Datei `cosine.schema` definiert und kommt zum Beispiel in der – ebenfalls dort definierten – `pilotPerson`-Objektklasse vor.)



4.2 [5] Erweitern Sie die Regel aus der vorigen Ausgabe so, dass der Wert des `favouriteDrink`-Attributs nur für Mitglieder der Organisation `example.com` lesbar ist.

4.1.5 Tipps und Tricks

Reguläre Ausdrücke in Zugriffsregeln Reguläre Ausdrücke sind grundsätzlich eine feine Sache. Ein paar ihrer Anwendungen sind allerdings mit Vorsicht zu genießen. Betrachten Sie zum Beispiel die Auswahlregel

```
dn.regex=".*dc=example,dc=com"
```

mit der alle DNs unterhalb von dc=example,dc=com beschrieben werden sollen. Aus verschiedenen Gründen ist das keine gute Idee:

- Der reguläre Ausdruck passt auf alle DNs, die dc=example,dc=com *irgendwo* enthalten, also auch auf Merkwürdigkeiten wie dc=example,dc=committee,dc=org oder yxdc=example,dc=com. Das ist mit einiger Sicherheit unerwünscht. Ein korrekter regulärer Ausdruck wäre etwas wie

```
dn.regex="^(.+)?dc=example,dc=com$"
```

Hier wird sichergestellt, dass der *komplette* DN auf den Ausdruck passen muss und dass der Teil links vom ersten dc – so er denn vorhanden ist – mit einem Komma aufhört.

- Die Vorgehensweise ist ineffizient, weil die Auswertung regulärer Ausdrücke relativ »teuer« ist. Es ist oft günstiger, eine andere Methode zu verwenden. Zum Beispiel würden auch

```
dn.subtree="dc=example,dc=com"
dn.children="dc=example,dc=com"
dn.onelevel="dc=example,dc=com"
```

*Alles, auch der Eintrag selbst
Teilbaum ohne Eintrag
Nur die direkten Kinder*

funktionieren.

Gewöhnen Sie sich an, »([^,]+)« zu schreiben, wenn Sie »genau einen RDN« meinen. »(.+)« passt auf eine beliebige lange Folge von RDNs.

Rund um den rootdn Vermeiden Sie es, den rootdn in Zugriffsregeln zu verwenden. Der rootdn darf sowieso an alles dran, er muss also nicht gesondert aufgeführt werden.

In Kapitel 3 haben wir Ihnen gezeigt, wie Sie ein Kennwort für den rootdn in der slapd-Konfiguration angeben können (mit rootpw). Das ist natürlich nicht gerade der Gipfel der Eleganz (von Sicherheit mal ganz zu schweigen). Wenn Sie die Notwendigkeit eines rootpw umgehen wollen, müssen Sie einen Eintrag für den rootdn ins Verzeichnis tun, ungefähr wie

```
dn: cn=Manager,dc=example,dc=com
objectclass: organizationalRole
objectclass: simpleSecurityObject
cn: Manager
userPassword: geheim
```

Im wirklichen Leben: verschlüsselt!

(organizationalRole liefert die Attribute cn und description, während simpleSecurityObject das userPassword beisteuert.)

Wenn Sie sich dann mit dem rootdn am Verzeichnis anmelden wollen, ist eine normale bind-Operation nötig – mit auth-Zugriff auf den DN und das userPassword des Eintrags. Dafür können Sie natürlich eine Zugriffsregel aufstellen, etwa um Zugriffe (mit dem Kennwort im Verzeichnis) nur von dem Rechner aus zu erlauben, auf dem der OpenLDAP-Server läuft:

```
access to dn.base="cn=Manager,dc=example,dc=com"
  by peername.ip=127.0.0.1 auth
  by users none
  by * none
```

Die `peername.ip=`-Klausel gestattet Vergleiche mit der IP-Adresse des anfragenden Rechners, siehe `slapd.access(5)`.

4.2 OpenLDAP und SASL

4.2.1 SASL-Grundlagen

Wie wir im Abschnitt 1.5 erklärt haben, sieht LDAP verschiedene Arten von Bindeoperationen vor, mit denen ein Client eine Verbindung zu einem LDAP-Server herstellen kann. Bisher haben wir die »einfache« Authentisierung auf der Basis von Klartextkennwörtern verwendet; LDAP erlaubt aber auch eine sichere (wenn auch aufwendigere) Vorgehensweise, bei der keine Klartextkennwörter verwendet werden müssen. Grundlage dieses Ansatzes ist (wie ebenfalls in Abschnitt 1.5 beschrieben) das SASL-Protokoll. Im Falle von OpenLDAP wird die Cyrus-SASL-Bibliothek herangezogen.



Wo Sie Cyrus SASL herbekommen, überlassen wir Ihrer Kreativität. Der einfachste Weg dürfte darin bestehen, die entsprechenden Pakete Ihrer Linux-Distribution zu installieren.

SASL ist vor allem eine Architektur zum Einbinden diverser Authentisierungsmechanismen, unter anderem Kerberos, CRAM-MD5 oder Digest-MD5. Anwendungsprogramme wie OpenLDAP sagen der SASL-Bibliothek, welche Sorte Authentisierungsverfahren gewünscht ist, müssen aber die Details der betreffenden Verfahren nicht kennen. Wenn man m Programme an n Authentisierungsverfahren anpassen möchte, muss man so nur $m + n$ und nicht $m \cdot n$ Schnittstellen implementieren – eine große Vereinfachung. SASL hat auch noch weitere Vorteile, zum Beispiel unterscheidet es zwischen Authentisierungs- und Autorisierungs-Identitäten, wobei die Authentisierungs-Identität (`<authcid>`) angibt, auf welcher Basis das System einen Benutzer *erkennt*, während die Autorisierungs-Identität (`<authzid>`) bestimmt, welche *Rechte* ein Benutzer zugesprochen bekommt.

SASL

Authentisierungs-ID
Autorisierungs-ID

Beispielsweise gibt es auf dem System etwa einen Benutzer `hugo`, dem das Betriebssystem erlaubt, im Verzeichnis `/home/hugo` Dateien zu schreiben, im Verzeichnis `/etc` aber nicht. `hugo` ist in diesem Zusammenhang die Autorisierungs-Identität. Die Rechte des Benutzers `hugo` kann aber jeder bekommen, der den Benutzernamen (`hugo`) und das zugehörige Kennwort (vielleicht `x7.za4r`) kennt. In diesem Sinne ist der Benutzername `hugo` auch die Authentisierungs-Identität. Oft ist es sinnvoll, dass ein Benutzer sich mit einer Identität anmeldet (Authentisierung), aber anschließend mit den Rechten einer anderen Identität agiert (Autorisierung). Zum Beispiel könnte Hugos Sekretärin in Hugos Abwesenheit Hugos E-Mail bearbeiten, indem sie sich unter ihrem eigenen Benutzernamen anmeldet, aber für `hugo` autorisiert wird (Hugo muss das natürlich erlauben).

Ein weiteres SASL-Konzept sind **Authentisierungsbereiche** (engl. *realms*). In einem Authentisierungsbereich werden Benutzer zusammengefasst, die auf dieselbe Weise authentisiert werden (zum Beispiel gegen dieselbe Benutzerdatenbank). Im einfachsten Fall könnten Sie zur Authentisierung auf einem einzelnen Rechner dessen vollqualifizierten Rechnernamen als Namen des Authentisierungsbereichs verwenden. Wollen Sie innerhalb einer Organisation Benutzer auf verschiedenen Rechnern mit denselben Kennwörtern authentisieren, so könnten Sie den Namen der Domain (`EXAMPLE.COM`) als Namen des Authentisierungsbereichs benutzen. Aus Sicherheitsgründen könnten Sie aber die Server (oder jeden Server) der Organisation einem eigenen Authentisierungsbereich zuordnen.

Authentisierungsbereiche

Mechanismen Cyrus SASL unterstützt eine Reihe verschiedener **Mechanismen** zur Authentisierung, von denen im folgenden einige kurz vorgestellt werden.

PLAIN Der PLAIN-Authentisierungsmechanismus verwendet Klartextkennwörter (so wie die »einfache« Authentisierungsmethode von OpenLDAP). Er ist selbst also nicht sicher, sondern ist gedacht für die Verwendung über Verbindungen, die von sich aus sicher sind (etwa TLS). PLAIN überträgt ein Tripel ($\langle\text{authcid}\rangle, \langle\text{authzid}\rangle, \langle\text{Kennwort}\rangle$), und der Server entscheidet, ob dieses Tripel zulässig ist. Es gibt verschiedene Möglichkeiten, die Gültigkeit des Kennworts tatsächlich zu überprüfen, darunter der Rückgriff auf `/etc/passwd`, `/etc/shadow` (hierfür muss das SASL benutzende Programm geeignete Leserechte haben) oder – unter Linux – auf PAM. Mit PAM können Sie natürlich wieder auf `/etc/shadow`, NIS, LDAP, ... zugreifen; Cyrus SASL unter Linux verwendet standardmäßig PAM-Authentisierung mit dem SASL-seitigen Dienstenamen (`ldap`) als »Programmname« für PAM. Eine wichtige Beobachtung in diesem Zusammenhang ist, dass Cyrus SASL PAM *nur* für die Kennwortprüfung benutzt – alle anderen Restriktionen, die Sie mit PAM umsetzen können (»sicheres« Terminal, ...), sollten für Dienste, die SASL zur Authentisierung nutzen, nicht in Anspruch genommen werden.

SASL unterstützt außerdem noch eine eigene Datenbank für Kennwörter (`sasldb`), die den Vorteil hat, mit den sichereren SASL-Mechanismen zu interoperieren (siehe unten). Ferner ist sie die einzige Möglichkeit, mit Klartextkennwörtern mehrere Authentisierungsbereiche auf einem Rechner zu unterstützen. Genauere Informationen über die verschiedenen Überprüfungsmöglichkeiten für Klartextkennwörter enthält die Dokumentation zu Cyrus SASL.

Herausforderung **CRAM-MD5 und DIGEST-MD5** Diese Mechanismen beruhen ebenfalls auf Kennwörtern, haben gegenüber dem PLAIN-Mechanismus aber den Vorteil, dass die Kennwörter bei der Authentisierung nicht über das Netz verschickt werden und so auch nicht ausgespäht werden können. Statt dessen schickt der Server eine **Herausforderung** (engl. *challenge*), aus der der Client auf der Basis seines Kennworts eine Antwort (engl. *response*) bestimmt und zurückschickt. Der Server vergleicht diese Antwort mit der, die sich aus seiner eigenen Kopie des Kennworts ergibt, und authentisiert den Client, wenn die beiden übereinstimmen. (Dieses Prinzip entspricht in etwa dem, was in Windows-Netzen bei der Anmeldung eines Benutzers am Domänencontroller passiert.)

Der Nachteil dieses Ansatzes ist, dass die serverseitige Kennwortdatenbank unbedingt geheimgehalten werden muss, da sie die Kennwörter im Klartext enthält. Eine mögliche Lösung im Fall von SASL ist die schon erwähnte `sasldb`; es gibt ein Kommando `saslpasswd`, mit dem Sie Identitäten und Kennwörter in die `sasldb` eintragen können. Die `sasldb` wird normalerweise in einer Datenbankdatei gespeichert; je nach dem verwendeten Mechanismus – Berkeley DB, `gdbm` oder ähnlichem – heißt sie `/etc/sasldb` mit verschiedenen Suffixen oder ist auch auf zwei Dateien verteilt. Sinnvoller ist es in den meisten Fällen jedoch, das LDAP-Verzeichnis selbst zu nutzen. Seit Cyrus-SASL 2.1 ist dies möglich, es müssen lediglich entsprechende Vorkehrungen getroffen werden, um die Klartext-Kennwörter vor neugierigen Blicken zu schützen (Stichwort: Zugriffsregeln).



DIGEST-MD5 ist übrigens der Nachfolger von CRAM-MD5 und für LDAPv3 als »unbedingt zu implementieren« vorgeschrieben; im Gegensatz zu CRAM-MD5 ist es nicht verwundbar gegen eine Klasse von Angriffen, bei denen der Angreifer den zu verschlüsselnden Klartext kontrolliert (*chosen-plaintext attack*).

Kerberos SASL kann mit verschiedenen Geschmacksrichtungen von Kerberos interoperieren. Es verwendet dafür die Mechanismen `KERBEROS_V4` (der mit jeder Kerberos-4-Implementierung zurecht kommen sollte) und `GSSAPI` (für MIT- und Heimdal-Kerberos). Diese Mechanismen haben keine eigene Kennwortdatenbank, sondern verlassen sich auf Kerberos. Die Installation und Konfiguration

von Kerberos würde den Umfang dieses Kurses sprengen und sie sowie die SASL-Mechanismen dafür werden darum hier nicht weiter besprochen.

Andere Ansätze Der letzte interessante Mechanismus von SASL ist EXTERNAL, ein Mechanismus, der vorhandene Authentisierungsmechanismen tieferliegender Schichten des Systems verwendet, etwa TLS. Hierauf kommen wir in Abschnitt 4.3 zurück. Es gibt auch noch andere starke Authentisierungsverfahren, etwa über Einmalkennwörter, die in dieser Schulungsunterlage aber nicht weiter besprochen werden.

4.2.2 OpenLDAP mit SASL-Authentisierung

Um OpenLDAP mit SASL-Authentisierung zu verwenden, müssen Sie zuerst einen Platz für die Benutzerkennwörter finden. Es wäre möglich, die `/etc/sasl2` zu verwenden, aber es ist mehr als lästig, alle Benutzer dort eintragen zu müssen (von allfälligen Kennwortänderungen mal ganz zu schweigen). Das LDAP-Verzeichnis selbst ist ein viel besserer Platz.



Für Kennwörter haben einige der gängigen Objektklassen ein Attribut namens `userPassword` – `organization`, `organizationalUnit` und `person` sind Beispiele. Alle Einträge, die keine dieser Klassen in ihrer Hierarchie haben, können die AUXILIARY-Objektklasse `simpleSecurityObject` hinzunehmen, die genau dieses Attribut als `MUST` mitbringt. Das ist zum Beispiel für `organizationalRole` nützlich.

Die einfachste Methode, OpenLDAP mit SASL-Authentisierung zu verwenden, besteht darin, das eingebaute SASL-`auxprop`-Plugin zu verwenden, das den Server auf sein eigenes Verzeichnis zugreifen läßt. Sie können das sicherstellen, indem Sie eine Datei namens `slapd.conf` mit dem Inhalt

```
pwcheck_method: auxprop
auxprop_plugin: slapd
```

im Verzeichnis `/usr/lib/sasl2` ablegen. Eine weitere Konfiguration auf der SASL-Seite ist nicht erforderlich.



Verwechseln Sie diese Datei nicht mit der Konfigurationsdatei, die der `slapd` selbst gemäß der »traditionellen« Konfigurationsmethode verwendet. Das ist `/etc/ldap/slapd.conf`, `/etc/openldap/slapd.conf` oder so ähnlich (je nach Ihrer Distribution).



In Anbetracht der Tatsache, dass `/usr/lib` kein guter Platz für Konfigurationsdateien ist, erlauben einige Distributionen auch die Verwendung von Verzeichnissen unter `/etc`: Debian GNU/Linux benutzt `/etc/ldap/sasl2`, die SUSE- und Red-Hat-Distributionen `/etc/sasl2`.

Wir betrachten im folgenden die Konfiguration von OpenLDAP mit SASL auf der Basis des DIGEST-MD5-Mechanismus. Hierbei verfügt der Server über eine Kopie des Kennworts im Klartext (das Kennwort ist das »gemeinsame Geheimnis« von Server und Client bzw. Benutzer, auf dem die Authentisierung aufbaut). Wenn Sie, was sehr empfehlenswert ist, das `ldappasswd`-Kommando zur Kennwortverwaltung verwenden möchten, müssen Sie also dafür sorgen, dass dieses die Kennwörter nicht (wie sonst üblich) verschlüsselt.



Natürlich können Sie Kennwörter auch »mit der Hand«, soll heißen mit `ldapadd` oder `ldapmodify`, ins Verzeichnis tun – aber `ldappasswd` ist die bequemere Möglichkeit.

Um für die richtige Verschlüsselung (oder in unserem Fall *Nicht*-Verschlüsselung) zu sorgen, müssen Sie in der Datei `slapd.conf` (der richtigen, nicht der für SASL) die Zeile

```
password-hash {CLEARTEXT}
```

zu den globalen Konfigurationsparametern hinzufügen. Nach einem Neustart von slapd können Sie dann zum Beispiel versuchen, für den Benutzer `cn=Hugo Schulz,dc=example,dc=com` ein Kennwort zu setzen. Um Schwierigkeiten zu vermeiden, machen Sie das am besten unter dem `rootdn`:

```
$ ldappasswd -W -x -D "cn=Manager,dc=example,dc=com" \  
> -S "uid=hugo,ou=users,dc=example,dc=com"  
New password: abc123  
Re-enter new password: abc123  
Enter LDAP Password: geheim
```

Für den rootdn

(Die Option `-S` fragt das Kennwort interaktiv ab – wie üblich zweimal. Schlagen Sie in `ldappasswd(1)` nach, um die anderen Möglichkeiten zum Setzen von Kennwörtern zu finden.) Vergewissern Sie sich, dass das Kennwort angekommen ist:

```
$ ldapsearch -W -x -D "cn=Manager,dc=example,dc=com" \  
> -LL "uid=hugo" userPassword  
Enter LDAP Password: geheim  
version: 1  
  
dn: uid=hugo,ou=users,dc=example,dc=com  
userPassword:: YWJjMTIz
```

»Halt!« werden Sie jetzt schreien. »Wir hatten doch keine Verschlüsselung ausgemacht!!« Aber schauen Sie genau hin: Die beiden Doppelpunkte hinter `userPassword` deuten auf ein Base64-kodiertes Attribut hin. Und in der Tat:

```
$ echo YWJjMTIz | mimencode -ub  
abc123$
```

Wenn ein Client sich über SASL beim OpenLDAP-Server authentisieren möchte, erscheint seine (Authentisierungs-)Identität als »Authentisierungs-Anfrage-DN« (engl. *authentication request DN*) in der Form

```
uid=<Name>,cn=<Mechanismus>,cn=auth
```

Wenn Sie zum Beispiel `ldapsearch` & Co. verwenden, entspricht `<Name>` Ihrem Linux-Benutzernamen (aber Sie können das mit der Option `-U` überschreiben). Bei anderen Clients ist `<Name>` möglicherweise etwas Anderes. `<Mechanismus>` ist der gewählte SASL-Authentisierungsmechanismus, also etwas wie `digest-md5`.



Wenn der Client einen Authentisierungsbereich `<Bereich>` verwendet, der vom Standard (dem Rechnernamen) abweicht, hat der Authentisierungs-Anfrage-DN die Form

```
uid=<Name>,cn=<Bereich>,cn=<Mechanismus>,cn=auth
```

Da DITs aber normalerweise nicht von `cn=auth` abhängen, werden diese Authentisierungs-Anfrage-DNs abgebildet auf DNs, die tatsächliche Einträge im DIT bezeichnen. Hierzu dient die `authz-regexp`-Direktive in der `slapd.conf`-Datei. Sie beschreibt Authentisierungs-Anfrage-DNs mit Hilfe von regulären Ausdrücken und gibt an, wie daraus »echte« DNs gemacht werden. Zum Beispiel könnte es sein, dass der Authentisierungs-Anfrage-DN eines Benutzers aussieht wie

```
uid=hugo,cn=example.com,cn=digest-md5,cn=auth
```

und sein Eintrag in der LDAP-Datenbank unter dem DN

```
uid=hugo,ou=users,dc=example,dc=com
```

zu finden ist. Ersterer könnte dann über eine `authz-regexp`-Direktive der Form

```
authz-regexp
uid=(.*) ,cn=example.com,cn=digest-md5,cn=auth
dn:uid=$1,ou=users,dc=example,dc=com
```

in letzteren überführt werden. Hier gibt die zweite Zeile (nach dem `authz-regexp`) einen regulären Ausdruck an, der auf den Authentisierungs-Anfrage-DN passen soll, wobei alles zwischen `uid=` und dem nächsten Komma in der »Variablen« `$1` gespeichert wird. Die dritte Zeile setzt aus den davor gemäß der zweiten Zeile gefundenen Elementen dann den »Authentisierungs-DN« zusammen, unter dem der Eintrag im Verzeichnis gesucht wird.



In der `slapd.conf`-Datei dürfen auch mehrere `authz-regexp`-Direktiven auftauchen; dies ist oft nötig, wenn Sie mit mehreren Mechanismen oder Authentisierungsbereichen operieren oder im Client gar kein expliziter Authentisierungsbereich angegeben wird.

Ebenfalls komplizierter zu behandeln ist der Fall, wo der Authentisierungs-DN nicht durch einfache Textersetzung aus dem Authentisierungs-Anfrage-DN hervorgeht. Beispielsweise könnte es sein, dass Einträge im LDAP-Verzeichnis durch DN's der Form

```
cn=Hugo Schulz,ou=users,dc=example,dc=com
```

identifiziert werden und der Benutzername (`uid`) nur ein Attribut eines solchen Eintrags ist. In diesem Fall kann in der `authz-regexp`-Direktive als »Ziel« auch ein LDAP-URL (siehe Abschnitt 1.6) angegeben werden:

```
authz-regexp
uid=(.*) ,cn=example.com,cn=digest-md5,cn=auth
ldap:///ou=users,dc=example,dc=com??one?(uid=$1)
```

In diesem Fall wird der LDAP-Server nach Einträgen mit dem passenden `uid`-Attribut durchsucht. Liefert diese Suche genau einen Eintrag, dann wird dessen DN als Benutzername für die Authentisierung akzeptiert; wenn kein Eintrag gefunden wird oder aber mehrere, gilt die Authentisierung als fehlgeschlagen. Sinnvollerweise sollten Sie dafür sorgen, dass das Verzeichnis für das Attribut, nach dem gesucht wird, einen Index führt, da die Suche sonst ziemlich lange dauern kann.



Auch wenn `ldappasswd` Kennwörter nicht verschlüsselt, werden Benutzer weiterhin ihr Kennwort mit dem bekannten `passwd`-Kommando ändern wollen. Das ist möglich, muss ihm aber mitgeteilt werden, etwa durch einen Eintrag wie

```
pam_password clear
```

in der für PAM, mithilfe `passwd`, maßgeblichen `ldap.conf`-Datei. Für DIGEST-MD5 kommen nur Kennwörter in Frage, die *nach* dieser Änderung ins Verzeichnis eingetragen wurden.

4.2.3 Testen der Konfiguration

Zum Testen von OpenLDAP mit SASL können Sie sich zuerst davon überzeugen, dass die richtigen SASL-Mechanismen unterstützt werden. Dies geschieht, indem Sie das LDAP-Objekt an der Wurzel des Baums konsultieren:

```
$ ldapsearch -x -b "" -s base -LLL supportedSASLMechanisms
dn:
supportedSASLMechanisms: CRAM-MD5
supportedSASLMechanisms: NTLM
supportedSASLMechanisms: DIGEST-MD5
```

(das Ergebnis kann von Server zu Server variieren).

Mit dem Kommando `ldapwhoami` können Sie herausfinden, ob die Authentisierung funktioniert (also das korrekte Kennwort im Verzeichnis gefunden wird) und was der Authentisierungs-DN ist:


```
$ ldapwhoami -U hugo
SASL/DIGEST-MD5 authentication started
Please enter your password: abc123
SASL username: hugo
SASL SSF: 128
SASL data security layer installed.
dn:uid=hugo,ou=users,dc=example,dc=com
```


Bingo!

Im nächsten Schritt sollten Sie dann ausprobieren, was passiert, wenn Sie tatsächlich eine Authentisierung über SASL versuchen. Geben Sie den gewünschten Benutzernamen mit `-U` an, wenn Sie nicht unter dem betreffenden Namen bei Linux angemeldet sind:

```
$ ldapsearch -LLL -U hugo 'cn=Hugo*'
SASL/DIGEST-MD5 authentication started
Please enter your password: blafasel
SASL username: hugo
SASL SSF: 128
SASL data security layer installed.
dn: uid=hugo,ou=users,dc=example,dc=com
objectClass: inetOrgPerson
cn: Hugo Schulz
<<<<<<
```

Übungen

 **4.3 [3]** Konfigurieren Sie Ihren LDAP-Server so, dass er eine Authentisierung über SASL ermöglicht. (Gegebenenfalls müssen Sie aus Ihrer Linux-Distribution Pakete mit den gewünschten SASL-Mechanismen nachinstallieren, etwa bei Debian GNU/Linux.) Welche Authentisierungsmechanismen unterstützt Ihr Server?

 **4.4 [3]** Versehen Sie einen (oder ein paar) Einträge in Ihrem Verzeichnis mit einem `userPassword`-Attribut (notfalls über die Objektklasse `simpleSecurityObject`) und prüfen Sie z. B., ob mit den DNs dieser Einträgen eine Anmeldung mit dem DIGEST-MD5-Mechanismus möglich ist.

4.2.4 Proxy-Authentisierung

Proxy-Authentisierung erlaubt es Ihnen, sich unter einer bestimmten Authentisierungs-ID ans Verzeichnis zu binden, aber anschließend die Rechte einer anderen Autorisierungs-ID in Anspruch zu nehmen. Unter dem Strich kann Benutzer *A* sich mit seinem eigenen Kennwort als Benutzer *B* anmelden.



Dafür gibt es tatsächlich sinnvolle Anwendungen. [The08, 15.3.1] postuliert zum Beispiel ein Web-Interface, mit dem LDAP-Benutzer die persönlichen Informationen in ihren Einträgen ändern dürfen sollen. Ein Benutzer muss sich natürlich gegenüber dem Web-Server authentisieren, aber der Web-Server kann sich nicht direkt als dieser Benutzer gegenüber dem LDAP-Server authentisieren, um die tatsächlichen Änderungen vorzunehmen. Statt dessen authentisiert der Web-Server sich zunächst mit Hilfe einer geeigneten Rolle (`cn=webupdates,dc=example,dc=com` oder so etwas) und kann dann über Proxy-Authentisierung die Identität des Benutzers annehmen, um die tatsächlichen Änderungen zu machen. Damit funktionieren Zugriffsregeln und ähnliches wie gehabt.



Eine andere Anwendung, ebenfalls nach [The08, 15.3.1], besteht darin, direkte Authentisierung mit dem `rootdn` zu unterbinden und lieber eine Liste von DN's vorzusehen, die über Proxy-Authentisierung die Identität des `rootdn` annehmen dürfen. Dies macht es einfacher, nachzuvollziehen, wann etwas am Verzeichnis geändert hat.

Natürlich wollen Sie Proxy-Authentisierung nicht ohne jegliche Kontrolle erlauben (ansonsten könnten Sie sich den Umstand der Authentisierung auch gleich sparen). In der Datei `slapd.conf` (der von OpenLDAP, nicht der von SASL) können Sie mit dem Parameter `authz-policy` eine »Politik« für die Proxy-Authentisierung angeben. Die folgenden Werte sind möglich:

none Proxy-Authentisierung ist komplett verboten. Dies ist der Standardwert, wenn die Konfiguration keine `authz-policy`-Direktive enthält.

to Der Eintrag im Verzeichnis, der durch den Authentisierungs-DN adressiert wird, enthält ein `authzTo`-Attribut, das angibt, welche Autorisierungs-DNs von diesem Authentisierungs-DN erreichbar sind.

from Der Eintrag im Verzeichnis, der durch den Autorisierungs-DN adressiert wird, enthält ein `authzFrom`-Attribut, das angibt, welche Authentisierungs-DNs verwendet werden dürfen, um die Rechte dieses Autorisierungs-DN zu bekommen.

any (früher `both`) prüft erst die `to`-Politik, dann die `from`-Politik; bei Erfolg wird nicht mehr weiter geschaut.

all Sowohl die `to`- als auch die `from`-Politik müssen die Proxy-Authentisierung zulassen.

Ein `authzFrom`-Attribut in einem Eintrag beschreibt, welche anderen DN's über Proxy-Authentisierung an die Rechte dieses Eintrags kommen können. Umgekehrt beschreibt ein `authzTo`-Attribut in einem Eintrag, welche anderen Einträge Sie über Proxy-Authentisierung erreichen können, wenn Sie über diesen Eintrag angemeldet sind.



Sie müssen verhindern, dass beliebige Benutzer beliebige Dinge in die `authzTo`-Attribute ihrer Einträge schreiben können, wenn die `to`- oder `any`-Politik aktiv ist. Das erreichen Sie zum Beispiel über geeignete Zugriffsregeln, die nur privilegierten Benutzern erlauben, diese Attribute zu ändern.



`authzTo` und `authzFrom` sind operationale Attribute, das heißt, sie müssen nicht in einer Objektklasse stehen.

Die Werte von `authzFrom` und `authzTo` können fünf mögliche Formen annehmen, nämlich

```
ldap:///(<Basis-DN>?)[(<Suchbereich>)]?(<Filter>
dn[(<DN-Stil>):(<Muster>
group[/(<Objektklasse>)[/(<Attribut>)]]:(<Muster>
u[(<Mechanismus>)[(<Authentisierungsbereich>)]]:(<Muster>
(<Muster>
```

Dabei ist *<DN-Stil>* ein Wert aus {exact, onelevel, children, subtree, regex}.

Die erste Form entspricht einem LDAP-URL (siehe Abschnitt 1.6), wobei kein Rechnername, keine Portnummer und keine Attribute angegeben werden dürfen. Die zweite Form erlaubt die Angabe von DN's mit den üblichen Suchbereichen sowie regex, das das *<Muster>* als regulären Ausdruck interpretiert. Als besondere Ausnahme beschreibt »*« alle nicht »anonymen« DN's.

Die dritte Form beschreibt eine Gruppe mit einer optionalen Objektklasse und einem optionalen Attribut. Das *<Muster>* gibt einen DN an, dessen Attribut *<Attribut>* betrachtet wird; enthalten die Werte von *<Attribut>* den Authentisierungs- oder Autorisierungs-DN (je nachdem), dann wird Erfolg gemeldet. Die vierte Form beschreibt einen SASL-Benutzer, und die fünfte Form, *<Muster>*, steht aus Kompatibilitätsgründen für einen exakten DN – den Sie aber deutlicher als »dn.exact:<Muster>« schreiben sollten.

Um das Ganze noch kurz zu illustrieren: Betrachten Sie den Verzeichniseintrag `cn=auth,dc=example,dc=com`, der aussieht wie folgt:

```
dn: cn=auth,dc=example,dc=com
objectClass: organizationalRole
objectClass: simpleSecurityObject
objectClass: uidObject
cn: auth
uid: auth
userPassword: auth123
authzTo: ldap:///dc=example,dc=com??sub?(uid=*)
```

Steht die Direktive »authz-policy to« in der Datei `slapd.conf`, so können Sie sich als `cn=auth,dc=example,dc=com` an das Verzeichnis binden und als Autorisierungs-ID jeden DN annehmen, der mit einem Objekt mit einem `uid`-Attribut korrespondiert.



Wir werden auf diesen Eintrag zurückkommen, wenn wir andere Dienste über SASL an das LDAP-Verzeichnis ankoppeln.

Testen können Sie Proxy-Authentisierung übrigens auch mit `ldappasswd`: Mit dem eben gezeigten `cn=auth`-Eintrag funktioniert etwas wie

```
$ ldapwhoami -U auth -X u:hugo
SASL/DIGEST-MD5 authentication started
Please enter your password: auth123
SASL username: u:hugo
SASL SSF: 128
SASL data security layer installed.
dn:uid=hugo,ou=users,dc=example,dc=com Als hugo!
```

Bei der `-X`-Option (die übrigens auch bei den anderen `ldap*`-Kommandos funktioniert) können Sie mit `u:hugo` angeben, dass Sie die Identität des SASL-Benutzers `hugo` annehmen wollen. OpenLDAP rechnet das in den Authentisierungs-DN `uid=hugo,ou=users,dc=example,dc=com` um.

4.3 OpenLDAP und TLS

4.3.1 Grundlagen

SASL ermöglicht zwar die Authentisierung von LDAP-Clients bei LDAP-Servern (und umgekehrt), aber unternimmt nichts zur Verschlüsselung des Datenverkehrs – es werden bestenfalls Anstrengungen unternommen, Authentisierungsinformationen wie die Kennwörter der Benutzer geheimzuhalten. Wenn Sie auch den Datenverkehr verschlüsseln und außerdem Authentisierung bequem erledigen möchten, sollten Sie OpenLDAP mit TLS benutzen. Cyrus SASL kann TLS über den EXTERNAL-Mechanismus ansteuern, um Clients am Server zu authentisieren.

Die Authentisierung von TLS beruht auf **X.509-Zertifikaten**, die Sie zum Beispiel mit den Werkzeugen aus OpenSSL anlegen können. X.509-Zertifikate werden für eine Person oder (typischerweise) einen Server ausgestellt; sie enthalten einen DN des »Inhabers« nebst dem dazugehörigen öffentlichen Schlüssel für ein geeignetes asymmetrisches Kryptoverfahren (etwa RSA) und werden von einer **Zertifizierungsstelle** (CA) digital signiert. Der Umgang mit X.509-Zertifikaten ist ein bisschen umständlich, aber nicht unüberwindbar schwierig. Wir setzen im Folgenden voraus, dass Sie sich mit der Erzeugung von X.509-Zertifikaten auskennen; wenn Sie eine Auffrischung brauchen, bietet Anhang B einen kleinen »Crashkurs« für die Grundlagen von X.509 und OpenSSL.

4.3.2 Server-Zertifikate

Alle Server brauchen Zertifikate, wenn sie über TLS zugänglich sein sollen, und es ist sehr wichtig, dass das `cn`-Attribut im DN des Zertifikats exakt den voll qualifizierten Namen des LDAP-Servers (gemäß DNS oder `/etc/hosts`) enthält. Haben Sie erst einmal ein Serverzertifikat – etwa in der Datei `/etc/ldap/ldap-crt.pem` – und den zugehörigen privaten Schlüssel – etwa in `/etc/ldap/ldap-key.pem` –, können Sie diese dem `slapd` durch Einträge der Form

```
TLSCertificateFile /etc/ldap/ldap-crt.pem
TLSCertificateKeyFile /etc/ldap/ldap-key.pem
```

in der `slapd.conf`-Datei bekanntmachen.



Gelegentlich wird man versuchen, Ihnen weiszumachen, dass Sie auf Ihrem Server auch das Zertifikat der Zertifizierungsstelle installieren müssen, die Ihr Zertifikat beglaubigt hat – die Einstellung dafür ist etwas wie

```
TLSCACertificateFile /etc/ldap/ca-crt.pem
```

Wirklich nötig brauchen Sie das aber nur, wenn der Server *Clients* über X.509-Zertifikate authentisieren soll – und dann benötigt er auch nicht das Zertifikat, mit dem sein eigenes, sondern das (oder die), mit dem die Zertifikate der Clients signiert sind¹.

Auf der Clientseite muss das Zertifikat der Zertifizierungsstelle vorhanden sein, damit der Client sich von der Authentizität des Zertifikats für den Server überzeugen kann. Letzteres schickt ihm der Server beim Verbindungsaufbau, aber es liegt auf der Hand, dass der Server ihm nicht auch das Zertifikat seiner Zertifizierungsstelle schicken kann². Als Betreiber des Clients müssen Sie sich dieses Zertifikat also möglichst direkt von der Zertifizierungsstelle holen und sicherstellen, dass es wirklich echt ist.



Spätestens wenn Sie Client-Zertifikate verwenden wollen, sollten Sie Ihre eigene Zertifizierungsstelle werden (siehe Anhang B). Alles andere ist herausgeschmissenes Geld.

Das Zertifikat der Zertifizierungsstelle machen Sie dem Client in der `ldap.conf`-Datei bekannt, etwa so:

```
TLS_CACERT /etc/ldap/ca-crt.pem
```

¹Die beiden können natürlich identisch sein, aber es ist gute Praxis, die Funktionen »Zertifizierungsstelle für Server« und »Zertifizierungsstelle für Clients« zu trennen.

²Würde der Client einem Zertifikat für die Zertifizierungsstelle des Servers vertrauen, das der Server ihm geschickt hat, könnte ein Angreifer, der so tut, als sei er der Server, einfach ein beliebiges Serverzertifikat und ein dazu passendes Zertifikat einer Zertifizierungsstelle schicken. Das ist natürlich nicht akzeptabel.



Ein LDAP-Client kann es mit diversen Servern zu tun haben und braucht grundsätzlich Zertifikate von allen Zertifizierungsstellen, die Zertifikate für diese Server beglaubigt haben. In diesem Fall ist es am einfachsten, alle diese Zertifikate zu sammeln und an die in der `TLS_CACERT`-Direktive benannte Datei anzuhängen. Damit das funktioniert, müssen die Zertifikate im (text-basierten) PEM-Format vorliegen; ein Zertifikat lässt sich wie folgt ins PEM-Format bringen:

```
$ openssl -in my-ca.crt -out mycacrt.pem -outform PEM
```

Als letztes müssen Sie unbedingt sicherstellen, dass der Client den LDAP-Server unter genau dem Namen anspricht, der im DN des Zertifikats steht. Dies lässt sich durch einen entsprechenden URI-Eintrag in der `ldap.conf`-Datei oder die Kommandozeilenoption `-H` von Programmen wie `ldapsearch` erreichen. TLS-Zugriffe werden über die Kommandooption `-ZZ` ausgelöst:

```
$ ldapsearch -LLL -ZZ 'cn=Hugo*'
```

»-ZZ« bedeutet, dass *nur* TLS-Zugriffe erlaubt sind. Die Option `-Z` würde versuchen, eine TLS-Verbindung aufzubauen, aber die Anfrage auch über normales LDAP stellen, wenn keine TLS-Verbindung zustandekommt.

Übungen



4.5 [!3] Erzeugen Sie gemäß der Anleitung in Anhang B ein Zertifikat und einen privaten Schlüssel für eine private Zertifizierungsstelle und verwenden Sie diese, um einen ebenfalls selbst erzeugten Schlüssel für Ihren LDAP-Server zu zertifizieren. Installieren Sie das Serverzertifikat in Ihrem LDAP-Server und stellen Sie sicher, dass Anfragen mit `ldapsearch -ZZ` erfolgreich behandelt werden. (Achten Sie dazu auf eine korrekte Konfiguration der Client-Seite.)

4.3.3 Client-Zertifikate

Client-Zertifikate werden im wesentlichen genauso erstellt wie Server-Zertifikate. Der DN des Client-Zertifikats sollte dem DN der betreffenden Person im LDAP-Verzeichnis entsprechen; ist das nicht der Fall, so kann auch hier auf `sasl-regexp` zurückgegriffen werden. Der Server muss über das Zertifikat der Zertifizierungsstelle verfügen, die die Client-Zertifikate beglaubigt.

Mit der Direktive `TLSVerifyClient` in der `slapd.conf`-Datei können Sie bestimmen, ob und mit welchem Nachdruck der LDAP-Server auf einer Authentisierung seiner Clients besteht. Die folgenden Werte sind möglich:

never Der LDAP-Server fragt den Client nicht nach einem Zertifikat.

allow Der LDAP-Server fragt nach einem Zertifikat, aber wenn er keins bekommt, geht die Sitzung so weiter, als wäre nichts passiert.

try Der LDAP-Server fragt nach einem Zertifikat, aber wenn er keins bekommt, geht die Sitzung so weiter, als wäre nichts passiert. Bekommt er eins und lässt es sich nicht verifizieren, wird die Sitzung sofort abgebrochen.

demand Der Client muss ein gültiges Zertifikat liefern, ansonsten wird die Sitzung sofort abgebrochen.



Auch Clients können vom Server verlangen, sich vernünftig zu authentisieren. Dazu dient die `TLS_REQCERT`-Direktive in der `ldap.conf`-Datei, deren Werte den eben besprochenen entsprechen.

Für den Client benennen Sie Zertifikat und privater Schlüssel in den Direktiven `TLS_CERT` und `TLS_KEY`. Diese Direktiven könnten grundsätzlich in der `ldap.conf`-Datei stehen, allerdings gelten sie als »benutzerspezifisch« und sind darum nur in einer `.ldaprc`-Datei im Heimatverzeichnis eines Benutzers erlaubt:

```
$ cat ~/.ldaprc
TLS_CERT /home/hugo/.ldap/hugo-crt.pem
TLS_KEY /home/hugo/.ldap/hugo-key.pem
```

Kommandos in diesem Kapitel

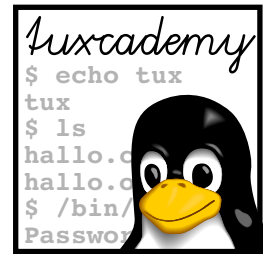
<code>ldapwhoami</code>	Testprogramm für Authentisierung an einem LDAP-Verzeichnis		
		<code>ldapwhoami(1)</code>	58
<code>saslpasswd</code>	Setzt ein Kennwort in einer SASL-Datenbank	<code>saslpasswd(8)</code>	54

Zusammenfassung

- Zugriffsrechte auf das Verzeichnis können genau geregelt werden.
- SASL erlaubt eine Authentisierung von Zugriffen auf das LDAP-Verzeichnis über verschiedene Mechanismen.
- Mit TLS können Zugriffe auf das Verzeichnis verschlüsselt und Client und Server gegenseitig authentisiert werden.

Literaturverzeichnis

- RFC2253** M. Wahl, S. Kille, T. Howes. »Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names«, Dezember 1997. <http://www.ietf.org/rfc/rfc2253.txt>
- RFC4515** J. Sermersheim, K. Zeilenga, K. Zeilenga, et al. »Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters«, Juni 2006. <http://www.ietf.org/rfc/rfc4515.txt>
- RFC4517** S. Legg. »Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules«, Juni 2006. <http://www.ietf.org/rfc/rfc4517.txt>
- The08** The OpenLDAP Foundation. »OpenLDAP Software 2.4 Administrator's Guide«, 2008. <http://www.openldap.org/>



5

Replikation

Inhalt

5.1	Grundlagen	66
5.2	Einfache Replikation.	67
5.3	Delta-Replikation	70
5.4	Multi-Master-Replikation	72

Lernziele

- Die Vor- und Nachteile von LDAP-Replikation und den verschiedenen Replikationsansätzen verstehen und bewerten können
- OpenLDAP-Replikation konfigurieren können

Vorkenntnisse

- LDAP-Grundlagenkenntnisse (Kapitel 1)
- Kenntnisse über OpenLDAP-Kommandozeilenwerkzeuge (Kapitel 2)
- Kenntnisse über die Konfiguration von OpenLDAP (Kapitel 3)

5.1 Grundlagen

Wie schon erwähnt ist es oft sinnvoll, ein LDAP-Verzeichnis an verschiedenen Stellen zu **replizieren**. Das heißt, dass es neben dem ursprünglichen LDAP-Server – dem **Produzenten** – sogenannte **Konsumenten** gibt, die über denselben Datenbestand verfügen und in der Lage sind, anstelle des ursprünglichen Servers Anfragen zu beantworten. Änderungen werden auf alle Server propagiert.

Replikation
Produzent
Konsument



Frühere Versionen von OpenLDAP unterstützten nur eine eingeschränkte Form von Replikation, bei der ein oder mehrere »Slave-Server« den Inhalt eines »Master-Servers« redundant vorhielten. Die Slave-Server akzeptierten Verzeichnisänderungen ausschließlich vom Master-Server, der wiederum Änderungsoperationen von Clients entgegennahm. Die Rollen waren auch starr fixiert: Jede Datenbank auf einem Server konnte entweder als Master oder Slave fungieren. In neuerer Zeit wurde hier viel verallgemeinert. So können zum Beispiel Konsumenten eines Verzeichnisses gegenüber anderen Servern als Produzenten auftreten. Aus diesem Grund wurde die Terminologie hier umgestellt.

Lastverteilung Eine gängige Anwendung der Replikation besteht darin, die Last zahlreicher LDAP-Anfragen auf mehrere Server zu verteilen. Man konfiguriert auf verschiedenen Servern einen Produzenten und mehrere Konsumenten und richtet einen DNS-Namen, etwa `ldap.example.com`, so ein, dass er die Adressen aller Server zurückerliefert. BIND als DNS-Server rotiert die Adressen innerhalb einer Antwort so, dass nacheinander jede der möglichen Adressen als primäre Adresse angenommen wird; damit trägt jeder der n LDAP-Server (Produzent und Konsumenten) im wesentlichen $1/n$ der Last.

Früher wurde Replikation in OpenLDAP über ein spezielles Programm namens `slurpd` implementiert. Es lief auf demselben Rechner wie der Master-`slapd` und reichte Änderungen an die Slave-Server weiter. Man spricht hierbei auch von *push-based replication*, weil der Master-Server (oder Produzent) die Änderungen bei Bedarf den Slave-Servern (oder Konsumenten) »herüberschiebt«. (Das Gegenteil wäre *pull-based replication*, wo der Konsument den Produzenten in periodischen Abständen nach Änderungen fragt.)

slurpd

In aktuellen OpenLDAP-Versionen ist der `slurpd` nicht mehr enthalten. Statt dessen unterstützt den OpenLDAP-Server die »LDAP Sync Replication Engine«, kurz »`syncrepl`«. `Syncrepl` wird nicht vom Produzenten ausgeführt, sondern vom Konsumenten, und kümmert sich um alle Replikationsaufgaben vom ursprünglichen Aufbau einer replizierten Datenbank bis zum Aktualisieren im Betrieb, entweder durch periodische Aktualisierungsanfragen oder durch das Akzeptieren von Änderungen in Echtzeit. Hierfür wird das LDAP-Inhaltssynchronisierungsprotokoll (LDAP Sync) nach [RFC4533] eingesetzt. Dies hat die folgenden Vorteile:

- Konsumenten brauchen kein explizites Protokoll der Änderungen, die sie schon empfangen haben. Statt dessen merken Produzenten und Konsumenten sich individuell ihren Status, und Konsumenten können bei Produzenten genau diejenigen Verzeichniseinträge anfordern, die sie benötigen, um sich bezogen auf die jeweiligen Produzenten aktuell zu machen.
- Neue Konsumenten können auf der Basis von Verzeichnisabzügen entweder eines Produzenten oder eines anderen Konsumenten eingerichtet werden und aktualisieren sich selbst inkrementell, sobald sie Zugriff auf einen Produzenten bekommen.
- `Syncrepl` unterstützt sowohl push-basierte als auch pull-basierte Replikation.
- Sie können den zu replizierenden Teil des DIT über die üblichen LDAP-Suchmechanismen konfigurieren: Sie können einen Basis-DN, einen Suchbereich, Suchfilter und zu replizierende Attribute festlegen. Außerdem wer-

den auf dem Produzent die Zugriffsrechte beachtet, die für die Identität definiert sind, mit der der Konsument sich an den Produzenten bindet.

- Syncrepl muss nur auf dem Konsumenten konfiguriert werden; eine Rekonfigurierung oder ein Neustart des Produzenten ist nicht notwendig, solange der Konsument über die nötigen Zugriffsrechte auf das Verzeichnis auf dem Produzenten verfügt. Ebenso kann der Konsument mit der Replikation aufhören, ohne dass Änderungen am Produzenten oder dessen Neustart erforderlich sind.



Details über LDAP Sync finden Sie in [The08, 18.1.1] oder (ausführlicher, aber komplizierter) in [RFC4533].

5.2 Einfache Replikation

Um die Replikation eines OpenLDAP-Servers (als Produzent) zu ermöglichen, müssen Sie als einzige Vorarbeit einige Direktiven zur Datei `slapd.conf` hinzufügen. Das ist ein einmaliger Vorgang; die Konsumenten können anschließend unabhängig vom Produzenten konfiguriert werden.

Die Syncrepl-Provider-Funktionalität von OpenLDAP ist als Overlay realisiert, also als »Zusatz« zu einer existierenden Backend-Datenbank. Entsprechend könnte eine Konfiguration so aussehen: Overlay


```
dn: cn=module{0},cn=config
objectclass: olcModuleList
cn: module{0}
olcModulePath: /usr/lib/ldap
olcModuleLoad: {0}syncprov.la
Distributionsabhängig

dn: olcDatabase={1}hdb,cn=config
objectclass: olcHdbConfig
olcDatabase={1}hdb
olcSuffix: dc=example,dc=com
olcDbDirectory: /var/lib/ldap
olcDbIndex: objectClass pres,eq
olcDbIndex: entryCSN pres,eq
olcDbIndex: entryUUID pres,eq
Siehe unten


dn: olcOverlay={0}syncprov,olcDatabase={1}bdb,cn=config
objectclass: olcSyncProvConfig
olcOverlay: {0}syncprov
olcSpCheckpoint: 10 10
olcSpSessionlog: 100
Siehe unten
Siehe unten
```


Um dieses Beispiel besser zu verstehen, müssen Sie noch etwas mehr über die Interna von Syncrepl wissen. Das LDAP-Sync-Protokoll (und damit Syncrepl) verwendet zur Identifikation von Einträgen ein Attribut namens `entryUUID`, das für jeden Eintrag einen anderen Wert hat, der sich nie ändert (die DNs der Einträge dagegen können sich ja im Laufe der Zeit ändern und taugen darum nicht als eindeutige Bezeichner). entryUUID

Nach jeder Schreiboperation auf das Verzeichnis prüft `slapd`, ob ein neuer »Checkpoint« angelegt werden soll. Die Details werden über die letzten beiden Zeilen in unserem Beispiel gesteuert: Die `olcSpCheckpoint`-Direktive gibt an, dass seit dem letzten Checkpoint mindestens 10 Schreiboperationen oder mindestens 10 Minuten vergangen sein müssen; ist das der Fall, wird ein neuer Checkpoint angelegt und die »Fristen« beginnen von vorne. Die `olcSpSessionlog`-Direktive gibt an, wie groß das »Sitzungsprotokoll« sein darf, in dem die `entryUUIDs` von gelöschten Objekten vermerkt werden. Der Parameter (hier 100) ist die Anzahl von Objekten, die maximal im Sitzungsprotokoll stehen können.

 Wir hatten weiter oben behauptet, dass LDAP Sync kein Protokoll benötigt. Das heißt aber nicht, dass es nicht aus Effizienzgründen trotzdem eine gute Idee sein kann, eins zu haben – daher das Syncrepl-Sitzungsprotokoll. Wohlgermerkt findet es sich nur auf dem Produzenten, und alle Konsumenten können dasselbe Sitzungsprotokoll des Produzenten verwenden.

 Wenn Sie das Sitzungsprotokoll verwenden, wird in der Datenbank heftigst nach entryUUIDs gesucht. Sie sollten also – wie in unserem Beispiel gezeigt – einen eq-Index für das entryUUID-Attribut einrichten.

 Zugriffe für Replikationszwecke unterliegen der üblichen Zugriffskontrolle des OpenLDAP-Servers (siehe Abschnitt 4.1). Sie sollten also dafür sorgen, dass die Identität(en), die allfällige Konsumenten zum Syncrepl-Zugriff auf den Produzenten verwenden, die zu replizierenden Daten auch lesen dürfen.

 Ebenso sollten Sie dafür sorgen, dass für diese Identität(en) keine Begrenzungen gelten, was die Dauer von Suchvorgängen oder die Anzahl der zurückgegebenen Resultate angeht, da es sonst passieren kann, dass Replikationsvorgänge vorzeitig abgebrochen werden. Wenn der Konsument den DN `cn=sync,dc=example,dc=com` verwendet, dann können Sie eine Regel vorsehen wie

```
# In olcDatabase={1}bdb,cn=config:
<<<<<<
olcLimits: dn.exact="cn=sync,dc=example,dc=com"
           size=unlimited time=unlimited
```

Auf der Konsumentenseite könnte eine Konfiguration zum Beispiel so aussehen:


```
dn: olcDatabase={1}hdb,cn=config
objectclass: olcHdbConfig
olcDatabase: {1}hdb
olcSuffix: dc=example,dc=com
olcDbDirectory: /var/run/ldap
olcRootDN: cn=Manager,dc=example,dc=com
olcDbIndex: objectClass pres,eq
olcDbIndex: entryCSN pres,eq
olcDbIndex: entryUUID pres,eq
olcSyncrepl: rid=1
               provider=ldap://ldap.example.com:389
               bindmethod=simple
               binddn="cn=sync,dc=example,dc=com"
               credentials=sync123
               type=refreshOnly
               interval=01:00:00:00
               searchbase="dc=example,dc=com"
               schemachecking=off
```

Muss zum Produzenten passen


Siehe unten

Syncrepl funktioniert mit beliebigen Backend-Datenbanken, aber `bdb` und `hdb` sind die praktischen Werte. Hier konfigurieren wir eine pull-basierte Replikation vom Server `ldap.example.com`, Port 389, die einmal am Tag aktualisiert wird. Weitere bemerkenswerte Punkte sind:

- `rid=1` gibt diesem Konsumenten eine eindeutige Nummer. Andere Konsumenten müssen einen anderen Wert verwenden, sonst gibt es Durcheinander. Die Nummer darf eine maximal dreistellige Dezimalzahl sein.

 Die Anzahl von Konsumenten für einen Produzenten ist damit auf 999 begrenzt, aber das sollte Ihren Stil nicht einschränken – zum einen ist das eine sehr große Zahl, und zum anderen können Sie immer einzelne Konsumenten zu Produzenten für andere machen und so praktisch unbegrenzt wachsen.

- Mit `schemachecking=off` weisen wir den Konsumenten an, eingehende Einträge nicht auf Konformität zu ihrem Schema zu prüfen. Die Annahme ist, dass das der Produzent schon gemacht hat und dass der Konsument sich diese Arbeit deswegen sparen kann.
- Wir verwenden eine `simple`-Bindeoperation mit dem DN `cn=sync,dc=example,dc=com` und dem Kennwort `sync123`. Es versteht sich von selbst, dass Sie im wirklichen Leben entweder TLS oder aber eine SASL-basierte Authentisierung einsetzen sollten. Zuallermindestens sollten Sie dafür sorgen, dass der Inhalt des Konsumenten-`slapd.d` neugierigen Augen verborgen bleibt, aber das ist im Allgemeinen ja sowieso eine gute Idee.


 Die Identität `cn=sync,dc=example,dc=com` können Sie auf dem Produzenten zum Beispiel als


```
dn: cn=sync,dc=example,dc=com
objectclass: simpleSecurityObject
objectclass: organizationalRole
cn: sync
description: Account for replication
userPassword: sync123 oder besser verschlüsselt
```

anlegen, damit Sie ein Kennwort dafür im Verzeichnis hinterlegen können. (Siehe auch S. 52.)

- Zum Schreiben verwendet `Syncrepl` den `rootdn` der lokalen Datenbank, so dass keine besondere Authentisierung erforderlich ist.

Um die Replikation zu starten, müssen Sie auf dem Produzenten die neue Konfiguration in Kraft setzen, damit er Daten an den oder die Konsumenten ausliefern kann. Wenn Sie den Konsumenten starten, holt er sich automatisch eine Kopie der zu replizierenden Daten vom Produzenten.

 Wenn Sie ein sehr großes Verzeichnis haben – und insbesondere wenn die Bandbreite zwischen Produzent und Konsument begrenzt ist –, ist es in der Regel sinnvoll, den Konsumenten mit einer Kopie des Verzeichnisses zu initialisieren, das Sie mit `slapcat` auf dem Produzenten oder einem anderen Konsumenten angefertigt haben. Diese Kopie muss nicht absolut aktuell sein; der Konsument holt sich das, was ihm noch fehlt, vom Produzenten.

 Sollten Sie in der Protokolldatei auf dem Konsumenten die Fehlermeldung *got search entry without Sync State control* bekommen, dann liegt das mit großer Sicherheit daran, dass auf dem Produzenten die Replikation nicht oder nicht korrekt konfiguriert wurde. Prüfen Sie die Einstellungen dort.

Der Konsument weist Änderungsversuche für die replizierte Datenbank ab. Sie können sie aber an den Produzenten weiterleiten, indem Sie etwas wie

```
# in olcDatabase={1}bdb,cn=config
olcUpdateref: ldap://ldap.example.com:389
```

in den Eintrag für die Datenbank-Konfiguration des Konsumenten aufnehmen.

Wenn Sie statt der `pull`-basierten Replikation eine `push`-basierte Replikation verwenden wollen, bei der der Konsument (nahezu) in Echtzeit vom Produzenten über Änderungen benachrichtigt wird, können Sie das über die folgenden Modifikationen der `olcSyncrepl`-Direktive erreichen: Ersetzen Sie die Zeilen

```
type=refreshOnly
interval=01:00:00:00
```

durch etwas wie

```
type=refreshAndPersist
retry="60 +"
```

Dabei gibt `retry` an, wie der Konsument vorgehen soll, wenn der Produzent nicht erreichbar ist – in unserem Beispiel probiert er es alle 60 Sekunden, bis es wieder funktioniert.



Der Parameter von `retry` ist eigentlich eine Folge von Paaren ($\langle \text{Intervall} \rangle, \langle \text{Anzahl} \rangle$), die sagt, dass der Konsument $\langle \text{Anzahl} \rangle$ mal im Abstand von $\langle \text{Intervall} \rangle$ Sekunden versuchen soll, Kontakt zum Produzenten zu bekommen; anschließend geht er zum nächsten Paar über, falls es noch eins gibt, sonst stellt er seine Versuche ein. Die Anzahl $\gg+\ll$ steht für »unendlich oft«. Also:

<code>retry="60 10"</code>	<i>Zehnmal alle Minute, dann aufhören</i>
<code>retry="60 10 300 +"</code>	<i>Zehnmal alle Minute, dann alle fünf Minuten, bis es wieder funktioniert (oder die Welt untergeht)</i>



Realisiert wird die push-basierte Replikation in etwa so, dass der Konsument eine Suchanfrage an den Produzenten stellt, deren Antwort von diesem aber nie für abgeschlossen erklärt wird. Statt dessen reicht der Produzent immer wieder neue Datensätze nach, solange die Verbindung stehen bleibt.

Übungen



5.1 [13] Konfigurieren Sie – gegebenenfalls in Zusammenarbeit mit einem Schulungskollegen – zwei LDAP-Server gemäß der Anleitung so, dass der eine den Datenbestand des anderen repliziert. Vergewissern Sie sich, dass beide Server dieselben Daten anbieten und dass Änderungen auf dem Master-Server an den Slave-Server weitergeleitet werden.



5.2 [2] Sorgen Sie dafür, dass Änderungswünsche auf dem Slave-Server an den Master-Server weitergereicht werden, und stellen Sie sicher, dass die tatsächlichen Datenbankänderungen dann auch auf dem Slave-Server sichtbar werden.



5.3 [*3] Erstellen Sie einen redundanten »globalen« Verzeichnisdienst, bei dem zwei Server sich gegenseitig replizieren (Server *B* ist Slave-Server für den DIT von Server *A* und umgekehrt). Verwenden Sie für die Client-Programme einen »URI«-Eintrag in der `ldap.conf`-Datei, der beide Server benennt, und vergewissern Sie sich, dass Sie Zugriff auf beide DITs haben, auch wenn einer der beiden Server nicht zugänglich ist.

5.3 Delta-Replikation

Die einfache Replikation mit LDAP Sync basiert auf der Übertragung kompletter Objekte. Dies vereinfacht den Umgang mit »Ketten« von Änderungen der Attribute eines einzelnen Objekts, da nur der Endzustand repliziert werden muss und nicht alle Zwischenzustände. Auf der anderen Seite ist dieses Verfahren ungünstig, wenn Sie viele große Objekte haben, an denen Sie jeweils einzelne kleine Änderungen machen, und das oft.



Betrachten Sie eine Datenbank mit 100.000 Objekten, die je 1 KiB groß sind, und wo (etwa über ein Skript) ein zwei Byte großes Attribut in jedem Objekt geändert wird. Das heißt, es werden netto 200.000 Bytes geändert, aber zur Replikation müssen Daten im Wert von einem Gigabyte übertragen werden (und der Protokollüberhang von LDAP und TCP/IP ist da noch nicht einmal einkalkuliert). Das heißt, 99,98% der Datenübertragung war eigentlich überflüssig, da die betreffenden Daten beim Konsumenten nicht geändert werden mussten. Das belastet nicht nur die Verbindung und verursacht Kosten, sondern kann auch dazu führen, dass die zu »aktualisierenden« Daten sich beim Konsumenten aufstauen und dort eine Verstopfung provozieren. (Beispiel aus [The08, 18.2.1].)

Um solche pathologischen Situationen zu entschärfen, unterstützt Syncrepl eine spezielle Art der Replikation, die **Delta-Replikation**. Sie verwendet ein produktenseitiges Änderungsprotokoll, das von den Konsumenten abgefragt wird, um kurzfristig Änderungen nachzuziehen. Wenn die replizierte Kopie der Datenbank bei einem Konsumenten zu alt ist (oder ganz leer), wird ein normaler Replikationsvorgang angestoßen.

Delta-Replikation

Hier ist ein Konfigurationsbeispiel für den Produzenten:

```
dn: cn=module{0},cn=config
objectclass: olcModuleList
cn: module{0}
olcModulePath: /usr/lib/ldap
olcModuleLoad: {0}accesslog.la
olcModuleLoad: {1}syncprov.la

# Datenbank für das Änderungsprotokoll
dn: olcDatabase={3}hdb,cn=config
objectclass: olcHdbConfig
olcDatabase: {3}hdb
olcSuffix: cn=accesslog
olcDbDirectory: /var/lib/ldap-accesslog
olcRootDN: cn=accesslog
olcDbIndex: entryCSN pres,eq
olcDbIndex: objectClass pres,eq
olcDbIndex: reqEnd pres,eq
olcDbIndex: reqResult pres,eq
olcDbIndex: reqStart pres,eq

dn: olcOverlay={0}syncprov,olcDatabase={3}hdb,cn=config
objectclass: olcSyncProvConfig
olcOverlay: {0}syncprov
olcSpNoPresent: TRUE
olcSpReloadHint: TRUE

# »Echte« Datenbank
dn: olcDatabase={1}hdb,cn=config
objectclass: olcHdbConfig
olcDatabase: {1}hdb
olcSuffix: dc=example,dc=com
olcRootDN: cn=Manager,dc=example,dc=com
# Erlaube dem Konsumenten Zugriff auf alles
# Diese Zeile sollte vor allen anderen Zugriffsregeln stehen
olcAccess: {0}to *
  by dn.base="cn=sync,dc=example,dc=com" read
  by * break
<<<<<
olcDbIndex: entryCSN pres,eq
```

*Distributionsabhängig
Änderungsprotokoll
Replikations-Produzent*

Was auch immer sonst nötig ist

```

olcDbIndex: entryUUID pres,eq
olcLimits: dn.exact="cn=sync,dc=example,dc=com"
           size=unlimited time=unlimited

dn: olcOverlay={0}syncprov,olcDatabase={1}hdb,cn=config
objectclass: olcSyncProvConfig
olcOverlay: {0}syncprov
olcSpCheckpoint: 1000 60

dn: olcOverlay={1}accesslog,olcDatabase={1}hdb,cn=config
objectclass: olcAccessLogConfig
olcAccessLogDB: cn=accesslog
olcAccessLogOps: writes
olcAccessLogSuccess: TRUE
olcAccessLogPurge: 7+00:00 1+00:00

```

*Speichere Protokoll hier
Operationen add, delete, modify, modrdn
Nur erfolgreiche Operationen speichern
Siehe unten*

Die `olcAccessLogPurge`-Zeile sorgt dafür, dass jeden Tag diejenigen Einträge, die älter als eine Woche sind, aus dem Änderungsprotokoll entfernt werden (siehe `slapo-accesslog(5)`).

Auf dem Konsumenten entspricht die Konfiguration weitgehend der für einfache Replikation. In der `syncrepl`-Direktive sollten Sie die letzten Zeilen durch etwas wie

```

type=refreshAndPersist
retry="60 +"
searchbase="dc=example,dc=com"
schemachecking=on
logbase="cn=accesslog"
logfilter="(&(objectClass=auditWriteObject)(reqResult=0))"
syncdata="accesslog"

```

ersetzen.

5.4 Multi-Master-Replikation

Bei Multi-Master-Replikation sind Aktualisierungen auf mehreren Servern möglich, nicht nur auf einem. Hersteller kommerzieller Verzeichnisdienste stellen Multi-Master-Replikation als eine sehr wünschenswerte Eigenschaft dar, die ähnliche Vorteile (Wundheilung, ewige Jugend, ...) verspricht wie der heilige Gral – jedenfalls diejenigen, deren Verzeichnisdienste sie tatsächlich unterstützen. Bei Multi-Master-Replikation gibt es keinen einzelnen Produzenten, der als Zuverlässigkeits-Engpass betrachtet werden muss; Verzeichnisdienste mit Multi-Master-Replikation bieten sich also für den Aufbau von hochverfügbaren Verzeichnisstrukturen an. Allerdings werden oftmals andere Argumente für Multi-Master-Replikation ins Feld geführt, die sich bei näherer Betrachtung als wenig stichhaltig erweisen [The08, 18.2.2]:

- Multi-Master-Replikation ist weder notwendig noch nützlich für die Lastverteilung von Anfragen über mehrere Server. Einfache Replikation reicht dafür völlig aus.
- Auch bei Multi-Master-Replikation müssen Änderungen an alle anderen Server (»Ko-Produzenten« und Konsumenten) weitergereicht werden. Es wird also weder Datenverkehr im Netz noch Last auf den anderen Servern eingespart.
- Die Last auf Produzenten ist bei einfacher und Multi-Master-Replikation im besten Fall identisch. Allerdings ist es bei einfacher Replikation möglich,

die Indizierung auf dem Produzenten speziell zu optimieren, um den unterschiedlichen Zugriffsmuster für Produzent und Konsumenten Rechnung zu tragen. Dann führt einfache Replikation zu höherer Leistung als Multi-Master-Replikation.

Und es gibt wichtige Gründe, die gegen Multi-Master-Replikation sprechen:

- Die Konsistenz des Verzeichnisses kann nicht mehr gesichert werden. Wenn derselbe Eintrag im selben Moment auf zwei verschiedenen »Ko-Produzenten« aktualisiert wird, kann das Verzeichnis nicht entscheiden, wie diese beiden Aktualisierungen unter einen Hut gebracht werden sollen.
- Es kann auch passieren, dass ein Eintrag, den Sie eigentlich für gelöscht gehalten haben, plötzlich wieder von den Toten auferweckt wird – etwa weil er auf einem anderen Produzenten noch nicht entfernt werden konnte und dort jemand einen anderen Eintrag unterhalb des bei Ihnen gelöschten Eintrag im DIT einhängt. (Das Verzeichnis wird sich dann mit großer Sicherheit dafür entscheiden, Ihre Löschoperation zu ignorieren bzw. rückgängig zu machen.) Dies verletzt das Datenbankprinzip der »Dauerhaftigkeit« (engl. *durability*) von Transaktionen.
- Für einen Produzenten ist es schwierig, zu entscheiden, ob ein anderer Produzent nicht zu erreichen ist, weil er abgestürzt ist, oder weil das Netz nicht funktioniert. Wenn es nur am Netz liegt und verschiedene Clients »ihren« Produzenten weiter Daten schicken, kann es sehr schwierig, die jeweiligen Aktualisierungen später wieder unter einen Hut zu bekommen. Es ist oft eine bessere Strategie, einfache Replikation zu verwenden und Clients, die den Produzenten nicht erreichen können, das Schreiben einfach zu verbieten.

Seit Version 2.4 unterstützt OpenLDAP eine Multi-Master-Replikation auf der Basis von Syncrepl (falls Sie es nicht lassen können). Ein Konfigurationsbeispiel dafür finden Sie in [The08, 18.3.3].

Zusammenfassung

- Verzeichnisbäume können auf mehrere Server repliziert werden, um Zugriffe zu beschleunigen und das System weniger fehleranfällig zu machen.

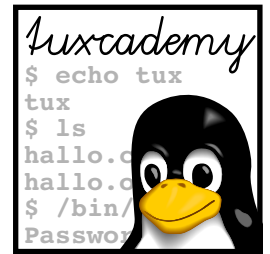
Literaturverzeichnis

RFC4533 S. Legg, K. Zeilenga, J.H. Choi. »The Lightweight Directory Access Protocol (LDAP) Content Synchronization Operation«, Juni 2006.

<http://www.ietf.org/rfc/rfc4533.txt>

The08 The OpenLDAP Foundation. »OpenLDAP Software 2.4 Administrator's Guide«, 2008.

<http://www.openldap.org/>



6

LDAP für Benutzerdaten

Inhalt

6.1	Linux-Benutzerdaten im LDAP-Verzeichnis	76
6.2	LDAP und der Name-Service-Switch.	79
6.3	LDAP und PAM	80
6.4	Kennwortverwaltung für LDAP-basierte Benutzer	83

Lernziele

- Verstehen, wie Benutzerdaten in einem LDAP-Verzeichnis abgelegt werden
- Einen Linux-Rechner so konfigurieren können, dass Benutzerdaten aus einem LDAP-Verzeichnis entnommen werden
- Einen Linux-Rechner so konfigurieren können, dass die Benutzerauthentifizierung mit Daten aus einem LDAP-Verzeichnis erfolgt

Vorkenntnisse

- Konfiguration von OpenLDAP, Umgang mit OpenLDAP-Werkzeugen
- Struktur und Konfiguration der *Pluggable Authentication Modules* (PAM)

6.1 Linux-Benutzerdaten im LDAP-Verzeichnis

LDAP vs. NIS Ein mögliches Einsatzgebiet für ein LDAP-Verzeichnis ist die Speicherung von Benutzerdaten (die sonst in `/etc/passwd` und `/etc/shadow` stehen). LDAP übernimmt hier die Rolle, die traditionell das *Network Information System* (NIS) spielt – Benutzerdaten für ein ganzes Netz können zentral gespeichert und gewartet werden – und erlaubt außerdem die Verwaltung von Linux-Benutzerdaten gemeinsam mit anderen relevanten Informationen über Personen in einer Organisation, etwa Telefonnummern, Adressen und Ähnlichem. LDAP hat gegenüber NIS auch noch zahlreiche weitere Vorteile, unter anderem:

- LDAP verwendet einen wohldefinierten Port, so dass Sie es leichter in Firewall- und Paketfilter-Konfigurationen berücksichtigen können;
- Server können verlangen, dass Clients sich authentisieren, und es ist eine detaillierte Zugriffskontrolle möglich;
- Clients können Einträge aktualisieren (im Rahmen der Zugriffskontrolle);
- LDAP erlaubt Suchoperationen über alle Felder;
- LDAP hat eine hierarchische Struktur und skaliert viel besser als NIS.

Der von OpenLDAP unterstützte RFC 2307 (*An Approach for Using LDAP as a Network Information Service*, [RFC2307]) sieht bereits eine Objektklasse vor, die zur Speicherung von Benutzerinformationen eines Unix-artigen Systems (wie Linux) geeignet ist, nämlich `posixAccount` (Bild 6.1). Sie enthält Attribute für die typischen Felder der Datei `/etc/passwd` wie Benutzername, Kennwort, numerische Benutzer-ID, Heimatverzeichnis usw. Eine zweite Objektklasse, `shadowAccount`, kann die zusätzlichen Verwaltungsinformationen aus der `/etc/shadow`-Datei speichern.

Informationen über einen Benutzer können Sie leicht in ein LDAP-Verzeichnis eintragen, indem Sie eine LDIF-Datei, etwa `hugo.ldif`, mit den Attributen und ihren Werten anlegen (Bild 6.2). Vorbedingung dafür ist, dass ein Eintrag für `dc=example`, `dc=com` existiert, etwa wie in Bild 1.8, der als »Wurzel« für die Benutzereinträge fungieren kann. Die LDIF-Datei in Bild 6.2 enthält bewusst keinen `userPassword`-Eintrag; auf diesen kommen wir gleich zurück.

Die Datei `hugo.ldif` lässt sich, wie im Abschnitt 2.3 gezeigt, mit dem Kommando `ldapadd` ins Verzeichnis aufnehmen (wir nehmen an, dass `BASE` in `ldap.conf` auf `dc=example,dc=com` gesetzt ist):

```
$ ldapadd -x -D "cn=Manager,dc=example,dc=com" -W -f hugo.ldif
```

Anschließend müssen Sie noch mit `ldappasswd` ein Kennwort setzen:

```
objectclass ( 1.3.6.1.1.1.2.0 NAME 'posixAccount' SUP top AUXILIARY
  DESC 'Abstraction of an account with POSIX attributes'
  MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
  MAY ( userPassword $ loginShell $ gecos $ description ) )

objectclass ( 1.3.6.1.1.1.2.1 NAME 'shadowAccount' SUP top AUXILIARY
  DESC 'Additional attributes for shadow passwords'
  MUST uid
  MAY ( userPassword $ shadowLastChange $ shadowMin $
    shadowMax $ shadowWarning $ shadowInactive $
    shadowExpire $ shadowFlag $ description ) )
```

Bild 6.1: Objektklassen für Linux-Benutzer gemäß RFC 2307

```
dn: cn=hugo,dc=example,dc=com
objectclass: posixAccount
cn: hugo
uid: hugo
uidNumber: 999
gidNumber: 100
homeDirectory: /home/hugo
loginShell: /bin/bash
gecos: Hugo Schulz
```

Bild 6.2: LDIF-Datei mit Informationen über den Benutzer hugo

```
$ ldappasswd -x -D "cn=Manager,dc=example,dc=com" -W \
> -S "cn=hugo,dc=example,dc=com"
New password: blafasel
Re-enter new password: blafasel
Enter bind password: geheim
```

Zunächst wird zweimal das neue Kennwort für den Benutzer hugo angegeben, dann einmal das LDAP-Kennwort für `cn=Manager,dc=example,dc=com`.

Ein bekanntes Problem mit NIS ist, dass der Ansatz sich nicht mit Shadow- »Shadow«-Kennwörter Kennwörtern verträgt – in einer NIS-passwd-Map müssen die verschlüsselten Kennwörter stehen. OpenLDAP löst dieses Dilemma sehr elegant. Eine Zugriffsrechtsdefinition der Form

```
access to attrs=userPassword
    by anonymous auth
    by self write
    by * none
```

regelt den Zugriff auf das `userPassword`-Attribut wie folgt: Nicht authentifizierte (anonyme) Benutzer dürfen nur zu Authentifizierungszwecken auf das Kennwort zugreifen (wobei sie den Wert nicht herausbekommen); der Inhaber des betreffenden Eintrags darf das Kennwort ebenfalls schreiben und der Rest der Welt hat gar keinen Zugriff.



Denken Sie an den impliziten Abschluss mit »access to * by * none«, wenn das Ihre einzige Zugriffsregel ist.



Den `rootdn` müssen wir hier nicht erwähnen, da er sowieso vollen Zugriff auf alle Einträge hat.

In der Praxis hat das die folgenden Konsequenzen: Mit `ldapsearch` können Sie sich den Datensatz für hugo anschauen:

```
$ ldapsearch -LLL -x uid=hugo
dn: cn=hugo,dc=example,dc=com
objectClass: posixAccount
cn: hugo
uid: hugo
uidNumber: 999
gidNumber: 100
homeDirectory: /home/hugo
loginShell: /bin/bash
gecos: Hugo Schulz
```

```
dn: cn=Emil Huber,dc=example,dc=com
objectClass: posixAccount
objectClass: organizationalPerson
cn: Emil Huber
uid: emil
uidNumber: 998
gidNumber: 100
homeDirectory: /home/emil
loginShell: /bin/bash
gecos: Emil Huber
sn: Huber
title: Ober-Emil
street: Musterweg 33
l: Beispielshausen
postalCode: 12345
telephoneNumber: +49-9999-99999
```

Bild 6.3: LDIF-Datei für einen Benutzer mit Zugangs- und persönlichen Informationen

Wie Sie sehen, ist das Kennwort aber in der Ausgabe nicht enthalten! Erst wenn der Administrator den Datensatz abrufen, wird das verschlüsselte Kennwort mitgeliefert:

```
$ ldapsearch -LLL -x -D cn=Manager,dc=example,dc=com \
> -W cn=hugo
Enter LDAP Password: geheim
dn: cn=hugo,dc=example,dc=com
objectClass: posixAccount
<<<<<<
gecos: Hugo Schulz
userPassword:: e1NTSEF9bTY0MUR3bFNBU1JFakFwY1VNd3NZRDY3cFFNMzVkr3o=
```

(Übrigens darf auch hugo das verschlüsselte Kennwort abrufen, wenn er sich identifiziert – write-Zugriff impliziert read-Zugriff. Dass das nicht völlig unkritisch ist, haben wir in Abschnitt 4.1 erklärt.)

Ändern und löschen können Sie Benutzerdaten natürlich wie in Kapitel 2 gezeigt über die Kommandos `ldapmodify` und `ldapdelete`. Das muss hier nicht noch einmal wiederholt werden.

Wenn zu den Informationen aus `posixAccount` auch noch weitere Informationen über Benutzer gespeichert werden sollen, dann können Sie diese einfach hinzufügen, indem Sie die passende Objektklasse zusätzlich angeben. Bild 6.3 zeigt eine LDIF-Datei für einen Benutzer mit Attributen der Objektklassen `posixAccount` und `organizationalPerson` (siehe auch Bild 1.4).



Sollten Sie jemals in die Verlegenheit kommen, eine größere¹ Installation von `/etc/passwd` nach LDAP umstellen zu müssen können Sie auf die »MigrationTools« von Luke Howard zurückgreifen (siehe <http://www.padl.com/OSS/MigrationTools.html>). Hierbei handelt es sich um eine Sammlung von Perl-Skripten, die aus den typischen NIS-Dateien unter `/etc` LDIF-Dateien generieren, die Sie dann in Ihr LDAP-Verzeichnis importieren können.



Zumindest bei Debian GNU/Linux stehen die MigrationTools schon als fertiges Paket zur Verfügung.

¹»Größer« ist hier natürlich Ansichtssache – aber seien Sie doch mal ehrlich zu sich selbst: Was machen Sie lieber, eine Viertelstunde lang öde LDIF-Dateien mit der Hand schreiben oder drei Stunden lang ein Skript bauen oder anpassen, das diese LDIF-Dateien automatisch generiert?

6.2 LDAP und der Name-Service-Switch

Natürlich reicht es nicht aus, Benutzerinformationen in einem LDAP-Verzeichnis zu speichern – Linux muss außerdem noch so konfiguriert werden, dass es das LDAP-Verzeichnis nach den Benutzerinformationen durchsucht, etwa wenn die Eigentümer von Verzeichnissen angezeigt werden sollen oder ein Benutzer sich anmelden möchte. Glücklicherweise ist es nicht erforderlich, dafür tiefe Eingriffe ins System vorzunehmen. Es genügt, die Datei `/etc/nsswitch.conf` anzupassen, in der geregelt ist, wie und wo das System, genaugenommen die C-Laufzeitbibliothek »GNU libc«, Informationen zum Beispiel über Benutzer findet. Alle Programme, die Funktionen aus der GNU libc verwenden, um zum Beispiel zu einem gegebenen Benutzernamen oder einer gegebenen numerischen Benutzer-ID die kompletten Benutzerinformationen zu finden – und das sind die allermeisten –, profitieren dann von dem Verzeichnis.

Die GNU libc unterstützt von sich aus die gängigsten Methoden zum Finden von Benutzerinformationen, etwa das Nachschlagen in `/etc/passwd` oder das Konsultieren von NIS. Weitere Methoden können über ladbare Module hinzugefügt werden. Luke Howard von PADL Software hat ein entsprechendes Modul geschrieben und als freie Software veröffentlicht, das der GNU libc den Zugriff auf LDAP-Verzeichnisse erlaubt. Dieses Modul können Sie <ftp://ftp.padl.com/pub> herunterladen; Sie finden es auch in den gängigen Distributionen.



Bei Debian GNU/Linux und Ubuntu müssen Sie das Paket `libnss-ldap` installieren. Konfigurationseinstellungen dafür stehen dann in der Datei `/etc/libnss-ldap.conf`. Netterweise fragt das Paket beim Installieren alle notwendigen Informationen ab, so dass Sie sich für den Rest dieses Abschnitts erst mal zurücklehnen können. (Nur die GNU libc müssen Sie noch selbst konfigurieren.)



Bei den Novell/SUSE-Distributionen steht die Unterstützung im Paket `nss_ldap`. Das Modul dort erwartet seine LDAP-Client-Konfiguration in `/etc/ldap.conf` (nicht zu verwechseln mit `/etc/openldap/ldap.conf`) zurück. – Beachten Sie, dass die SUSE-Version des Moduls standardmäßig TLS in der Konfiguration aktiviert – grundsätzlich ist das natürlich eine gute Idee, aber es setzt voraus, dass Ihr LDAP-Server darauf eingerichtet ist. Suchen Sie gegebenenfalls in `/etc/ldap.conf` nach und setzen Sie ein »#« davor.

Wenn die LDAP-Unterstützung für die GNU libc installiert ist, genügt es, in `/etc/nsswitch.conf` die Zeilen

```
passwd: compat
group:  compat
shadow: compat
```

durch die Zeilen

```
passwd: files ldap
group:  files ldap
shadow: files ldap
```

zu ersetzen. (Theoretisch wäre es möglich, auf `files` ganz zu verzichten – aber wenn dann aus irgendwelchen Gründen der LDAP-Server nicht zur Verfügung steht, kann sich niemand mehr anmelden, nicht einmal `root`. EPIC FAIL!)

Von der verwendeten Linux-Distribution hängt es ab, ob die GNU-libc-LDAP-Unterstützung die allgemeine LDAP-Clientkonfiguration (`ldap.conf` im LDAP-Konfigurationsverzeichnis) mitbenutzt oder separat konfiguriert werden kann. (Details dazu standen weiter oben.) Gegebenenfalls müssen Sie also auch hier die Client-Konfiguration anpassen, etwa wie

```
base dc=example,dc=com
uri ldap://ldap.example.com ldap://ldap-backup.example.com
```

LDAP-Test Es ist leicht möglich, zu prüfen, ob die LDAP-Daten verwendet werden. Das Kommando

```
$ getent passwd
```

listet alle Benutzer auf, die dem System momentan bekannt sind (aus welcher Quelle auch immer sie kommen). Mit

```
$ getent passwd emil
```

können Sie nach dem Benutzereintrag von `emil` suchen; ist `emil` im LDAP-Verzeichnis definiert und die Ausgabe leer, dann können die LDAP-Benutzerdaten aus irgendwelchen Gründen nicht erreicht werden. Beispielsweise könnten die Zugriffsrechte nicht reichen, oder die Client-Konfiguration könnte Fehler aufweisen.



Eine andere Methode: Legen Sie eine Datei oder ein Verzeichnis an, die einer UID gehört, die nur im LDAP-Verzeichnis eingetragen ist. Das Heimatverzeichnis des betreffenden Benutzers ist eine gute Möglichkeit:

```
# mkdir /home/emil
# chown 998:100 /home/emil
# ls -ld /home/emil
drwxr-xr-x  2 emil users 4096 Oct 10 18:38 /home/emil
```

UID/GID siehe oben

Wenn in der `ls`-Ausgabe der Benutzername erscheint, dann wird das LDAP-Verzeichnis benutzt; erscheint nur die numerische Benutzer-ID (hier 998), dann ist irgendetwas faul.

6.3 LDAP und PAM

PAM Für die tatsächliche Authentisierung verwenden heutige Linux-Systeme die *Pluggable Authentication Modules* (PAM). Das PAM-System abstrahiert den tatsächlichen Authentisierungsvorgang und macht es unnötig, bestimmte Authentisierungsverfahren fest in Programme einzubauen. Ein Programm wie `login` oder `sshd` kann die komplette Überprüfung, ob ein Benutzer den betreffenden Dienst verwenden darf, an PAM delegieren. Im Gegenzug bietet PAM die Möglichkeit, alle Aspekte der Authentisierung zentral zu konfigurieren. Auf diese Weise können unterschiedliche Verfahren unterstützt werden, mit denen Benutzer ihre Identität nachweisen – vom einfachen Unix-Kennwort über Einmalkennwörter oder Smartcards bis hin zur Biometrie –, aber auch die verschiedensten Quellen für Benutzerdaten verwendet werden, von den üblichen `/etc/passwd`- und `/etc/shadow`-Dateien über diverse Datenbanken wie DBM oder MySQL bis zu Windows-Domain-Controllern und eben zum Beispiel auch LDAP. (PAM bietet noch einige andere Möglichkeiten, die hier aber nicht im Detail erklärt werden können.)

Um ein LDAP-Verzeichnis als Benutzerdatenquelle für PAM verwenden zu können, muss die entsprechende Schnittstellenfunktionalität installiert sein (ebenfalls von Luke Howard und am selben Platz zu finden.)



Bei Debian GNU/Linux und Ubuntu finden Sie das PAM-LDAP-Modul im Paket `libpam-ldap`. Auch dieses Modul hat seine eigene LDAP-Client-Konfiguration, in der Datei `/etc/pam_ldap.conf`.



Bei den Novell/SUSE-Distributionen müssen Sie das Paket `pam_ldap` installieren, um alles Nötige auf Ihr System zu bekommen. Die PAM-Anpassung hat eine eigene Konfigurationsdatei für LDAP-Zugriffe in `/etc/ldap.conf`.

```

auth    requisite pam_securetty.so
auth    requisite pam_nologin.so
auth    required pam_env.so
auth    required pam_unix.so nullok
account required pam_unix.so
session required pam_unix.so
session optional pam_lastlog.so
session optional pam_motd.so
session optional pam_mail.so standard noenv
password required pam_unix.so nullok obscure min=4 max=8 md5

```

Bild 6.4: Beispiel für eine PAM-Konfiguration für login



An dieser Stelle ist auch der Hinweis angebracht, dass LDAP-Server zu Authentisierungszwecken, falls möglich, über TLS angesprochen werden sollten:

```
URI ldaps://ldap.example.com
```

Sonst werden Kennwörter im Klartext übertragen.

Danach kann die Konfiguration der gewünschten Dienste so geändert werden, dass LDAP-Daten verwendet werden. Für den gewöhnlichen (textkonsolenbasierten) Anmeldevorgang steht die Konfiguration zum Beispiel in `/etc/pam.d/login` (Bild 6.4).

PAM-Dienste konfigurieren

PAM unterscheidet vier Funktionsgruppen, in denen eine Reihe von Modulen – die rechte Spalte in der Konfigurationsdatei – nacheinander abgearbeitet werden: `auth` prüft, ob der Benutzer der ist, der er zu sein behauptet; `account` kümmert sich um Zugangsverwaltung, die nichts mit Authentisierung zu tun hat; `session` erledigt Sachen, die vor oder nach der Sitzung getan werden müssen (Protokoll, Zugänglichmachen von Ressourcen und ähnliches) und `password` befasst sich mit der Kennwortverwaltung. Jedes Modul in jeder Funktionsgruppe hat eine Wichtigkeitsangabe – die mittlere Spalte –: »required«-Module müssen erfolgreich ausgeführt werden, wobei erst nach der Ausführung des letzten Moduls der Folge ein Resultat an das PAM aufrufende Programm geliefert wird; »requisite«-Module beenden die Modulfolge sofort (abschlägig), falls ihre Ausführung fehlschlägt; »sufficient«-Module beenden die Modulfolge sofort (akzeptierend), wenn ihre Ausführung erfolgreich war, und das Ergebnis von »optional«-Modulen ist ziemlich egal. Das Modul `pam_unix.so` im besonderen kümmert sich um die »traditionelle« Unix-Authentisierung auf der Basis von `/etc/passwd` & Co.

Zur Integration von LDAP als Datenquelle könnten Sie die Beispielkonfiguration aus Bild 6.4 zum Beispiel wie folgt ändern: Die Zeile

```
auth    required pam_unix.so nullok
```

wird ersetzt durch die beiden Zeilen

```

auth    sufficient pam_ldap.so
auth    required pam_unix.so nullok use_first_pass

```

Das heißt, eine erfolgreiche Authentisierung gegen das LDAP-Verzeichnis beendet die Abarbeitung der `auth`-Modulfolge; der Benutzer wird akzeptiert. Schlägt die Authentisierung gegen LDAP fehl, so wird die traditionelle Methode verwendet, wobei die `use_first_pass`-Option dafür sorgt, dass nicht noch einmal nach einem Kennwort gefragt, sondern das vorher beim LDAP-Versuch eingegebene Kennwort weiterbenutzt wird.

Wenn die Authentisierung durch die auth-Modulfolge erfolgreich war, wird mit der account-Modulfolge geprüft, ob tatsächlich eine entsprechende Benutzerberechtigung existiert. Deshalb muss auch dort das LDAP-Modul aufgerufen werden:

```
account sufficient pam_unix.so
account required pam_ldap.so
```

Neben der PAM-Konfigurationsdatei für login sollten Sie auch die Konfigurationsdateien für andere, ähnliche Dienste anpassen. Dazu gehören zum Beispiel xdm und kdm (Anmelden auf dem Graphikbildschirm), ssh (Anmelden übers Netz), ppp, su oder sudo.



Moderne Distributionen machen es Ihnen da einfach: Typischerweise wird der »interessante« Teil der Konfiguration, namentlich die Beschreibung des tatsächlichen Anmeldeverfahrens, heutzutage in eine Datei ausgelagert, die die allermeisten Dienste gemeinsam verwenden. Wenn Sie in einer Datei wie /etc/pam.d/login eine Zeile der Form

```
auth include common-auth
```

finden, dann ist das ein Indiz dafür, dass Sie sich die Datei /etc/pam.d/common-auth anschauen sollten. Die Chancen stehen gut, dass Sie diese Datei ändern sollten und nicht die Datei, in der das include steht.



Aus nicht ganz erfindlichen Gründen verwenden Debian GNU/Linux und Ubuntu in ihren PAM-Konfigurationsdateien eine abweichende Syntax, um gemeinsame Konfiguration einzulesen: Seien Sie auf der Hut und achten Sie auf Zeilen der Form

```
@include common-auth
```

Ansonsten gilt das im vorigen Abschnitt Gesagte.



Bei den Novell/SUSE-Distributionen heißt das Modul, das in Bild 6.4 als pam_unix.so auftaucht, pam_unix2.so und macht die LDAP-Einbindung merkbar einfacher: Statt bei allen gewünschten Diensten pam_ldap.so einzutragen, genügt es, die Konfigurationsdatei /etc/security/pam_unix2.conf anzupassen, um LDAP überall da einzubeziehen, wo pam_unix2.so benutzt wird:

```
auth: use_ldap nullok
account: use_ldap
password: use_ldap nullok
session: none
```

(Wenn Sie zur Konfiguration von LDAP YaST verwenden, erledigt er das für Sie.)



Wo wir beim Thema »PAM« sind: Mit dem PAM-Modul pam_mkhome.so können Sie dafür sorgen, dass ein Benutzer, der zwar in der Benutzerdatenbank steht, aber kein Heimatverzeichnis hat, beim ersten Anmelden eines angelegt bekommt. Mit etwas wie

```
session required pam_mkhome.so umask=027 skel=/etc/skel
```

wird bei Bedarf das Heimatverzeichnis angelegt. Außerdem werden die Dateien aus /etc/skel in das neue Verzeichnis kopiert, dabei gilt die Umask 022, so dass diese »Grundausrüstung« für Gruppenmitglieder nicht schreibbar und den »Rest der Welt« nicht lesbar ist (das nur als Beispiel). – Damit genügt es, neue Benutzer im LDAP-Verzeichnis anzulegen.

6.4 Kennwortverwaltung für LDAP-basierte Benutzer

Nicht vergessen werden sollte auch das `passwd`-Programm, mit dem Benutzer ihre Kennwörter ändern können. Benutzer, deren Daten im LDAP-Verzeichnis stehen, sollten natürlich auch dort ein neues Kennwort eintragen können. Dazu können Sie die `password`-Modulfolge in `/etc/pam.d/password` erweitern:

```
password sufficient pam_ldap.so
password required pam_unix.so nullok obscure min=4 max=8 md5
```



Denken Sie auch hier wieder an eine etwa vorhandene »ausgelagerte« Konfiguration, die mehrere Dienste sich teilen. Schließlich ist `passwd` nicht die einzige Methode, sein Kennwort zu ändern – es könnte zum Beispiel sein, dass `login` einen Benutzer zur Eingabe eines neuen Kennworts zwingt, wenn sein altes abgelaufen ist.

Außerdem sollten Sie die Option

```
pam_password exop
```

zur PAM-LDAP-Konfiguration hinzufügen; diese bewirkt, dass das Kennwort über die LDAP-Funktion für Kennwortänderungen installiert wird, anstelle ein von PAM-LDAP lokal verschlüsseltes Kennwort direkt ins Verzeichnis zu schreiben. Der Sinn dahinter ist, dass zum Beispiel die in der `slapd`-Konfiguration angegebene Politik zur Verschlüsselung von Kennwörtern befolgt wird.



Denken Sie etwa daran, dass bei der Verwendung von SASL mit dem DIGEST-MD5-Mechanismus die Kennwörter im Klartext im Verzeichnis stehen müssen.

Die Änderung eines Kennworts auf einem Textterminal könnte dann ungefähr so aussehen:

```
$ whoami
emil
$ passwd
Enter login(LDAP) password: xrQ4j.a
New password: 6;Aq-B/
Re-enter new password: 6;Aq-B/
LDAP password information changed for emil
passwd: password updated successfully
$ _
```



Bei manchen Distributionen, etwa denen von SUSE/Novell, bringt das Paket mit dem `passwd`-Programm seine eigene LDAP-Konfigurationsdatei mit (bei SUSE in `/etc/ldap.conf`).

Kommandos in diesem Kapitel

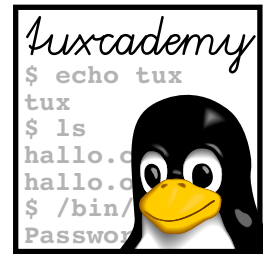
<code>getent</code>	Ruft Einträge aus administrativen Datenbanken ab	<code>getent(1)</code>	79
<code>ldappasswd</code>	Ändert das Kennwort eines LDAP-Eintrags	<code>ldappasswd(1)</code>	76

Zusammenfassung

- OpenLDAP sieht Objektklassen zur Speicherung von Linux-Benutzerinformationen vor.
- Mit einer geeigneten PAM-Konfiguration können Kennwörter im LDAP-Verzeichnis mit dem normalen Linux-passwd-Programm geändert werden.

Literaturverzeichnis

RFC2307 L. Howard. »An Approach for Using LDAP as a Network Information Service«, März 1998. <http://www.ietf.org/rfc/rfc2307.txt>



7

Apache und LDAP

Inhalt

7.1	Überblick.	86
7.2	Konfiguration von <code>mod_authnz_ldap</code>	86
7.2.1	Phasen der Zugriffskontrolle.	86
7.2.2	Die Authentisierungsphase	87
7.2.3	Die Autorisierungsphase	88
7.3	Beispiele	90
7.4	Das Modul <code>mod_ldap</code>	92

Lernziele

- Apache-Benutzer gegen ein LDAP-Verzeichnis authentisieren können
- Das Apache-Module `mod_authnz_ldap` konfigurieren können

Vorkenntnisse

- LDAP-Kenntnisse (aus den vorigen Kapiteln)
- Grundkenntnisse der Konfiguration und des Betriebs von Apache (etwa aus dem Linup-Front-Kurs *Der Web-Server Apache*)

7.1 Überblick

Der Web-Server Apache unterstützt diverse Methoden zur Speicherung von Benutzerdaten (Benutzernamen, Kennwörter und Gruppeninformationen) für die HTTP-Basic-Authentisierung [RFC2617]. Dazu zählen einfache Textdateien – die naheliegendste Methode – genau wie Berkeley-DB-Dateien, MySQL- oder PostgreSQL-Datenbanken, aber auch LDAP-Verzeichnisse. Hierfür stehen verschiedene Anbindungsmodule zur Verfügung. In dieser Schulungsunterlage konzentrieren wir uns auf das Modul `mod_authnz_ldap`, das Bestandteil der Apache-2.2-Standarddistribution ist und darum auch zu den meisten Linux-Distributionen gehört.

Das Modul `mod_authnz_ldap` erlaubt es, die im einfachsten Fall verwendeten Textdateien für Benutzer und Gruppen durch Zugriffe auf einen LDAP-Server zu ersetzen. Dabei können beliebige LDAP-Filter zum Lokalisieren von Einträgen verwendet und beliebige Attribute von LDAP-Einträgen abgefragt werden.

Man sollte gleich sagen, dass HTTP-Basic-Authentisierung für sich genommen nicht den Gipfel der Sicherheit darstellt, im Gegenteil. Immerhin werden Benutzername und Kennwort im Klartext (wenn auch Base-64-codiert) über das Netz geschickt und sind für einen Mithörer leicht zu finden. Es ist allerdings möglich und auch sinnvoll, HTTP-Basic-Authentisierung über eine SSL-verschlüsselte Verbindung (also mit HTTPS) zu verwenden. Dabei wird der Web-Server über sein Zertifikat authentisiert und der Client über Benutzername und Kennwort; eine HTTPS-Authentisierung über Client-Zertifikate ist im Vergleich ein deutlich höherer administrativer Aufwand, wenn auch noch sicherer, da die Zertifikate generiert, verteilt und in die Browser der Benutzer integriert werden müssen.

7.2 Konfiguration von `mod_authnz_ldap`

7.2.1 Phasen der Zugriffskontrolle

Die Zugriffskontrolle in Apache verläuft in zwei Phasen. Zunächst wird die Identität des Benutzers geprüft (**Authentisierungsphase**, im Kontext von `mod_authnz_ldap` auch »Suchphase« oder »Bindephase« genannt), und anschließend werden dem Benutzer aufgrund seiner gerade etablierten Identität Rechte zugeordnet (**Autorisierungsphase**, bei `mod_authnz_ldap` auch »Vergleichsphase«).

In der Authentisierungsphase nimmt `mod_authnz_ldap` Verbindung mit einem LDAP-Server auf und versucht, einen Eintrag zu finden, der zum angegebenen Benutzernamen »passt« (wobei Sie hierfür geeignete Suchfilter angeben können). Wird genau ein passender Eintrag gefunden (keiner oder mehrere sind nicht akzeptabel), dann versucht `mod_authnz_ldap`, sich mit dem DN dieses Eintrags und dem vom Benutzer über HTTP(S) angegebenen Kennwort an den LDAP-Server zu binden. Schlägt dies fehl, so wird der Zugriff abgelehnt. Gelingt der Zugriff, können in der anschließenden Autorisierungsphase noch Bedingungen für den gefundenen Eintrag ausgewertet werden. Im einfachsten Fall können Sie prüfen, ob der Benutzername des DN in einer Positivliste steht, aber Sie können auch die Mitgliedschaft in einer (LDAP-)Gruppe oder den Wert eines beliebigen LDAP-Attributs als Kriterium heranziehen.

Um `mod_authnz_ldap` für die Authentisierung heranzuziehen, müssen Sie es mit der Direktive

```
AuthBasicProvider ldap
```

einschalten.



Sie können bei `AuthBasicProvider` auch mehrere Quellen für Authentisierungsdaten angeben, etwa um mit etwas wie

```
AuthBasicProvider ldap file
```


sowohl LDAP als auch Textdateien heranzuziehen. Die »Lieferanten« der Authentisierungsdaten werden dann in der Reihenfolge ihrer Aufzählung konsultiert.



Der Wert von AuthBasicProvider hat nichts mit der Autorisierungsphase zu tun; wie dort verschiedene Autorisierungsmodule zusammenarbeiten, wird über separate Direktiven wie (im Falle von mod_authnz_ldap) AuthLDAPAuthoritative geregelt.

7.2.2 Die Authentisierungsphase

Für die Authentisierungsphase sind die folgenden Apache-Direktiven gültig (sie müssen entweder in einem <Directory>-Block in der httpd.conf-Datei angegeben werden oder können auch in einer .htaccess-Datei im betreffenden Verzeichnis oder einem übergeordneten Verzeichnis stehen):

AuthLDAPURL <URL> Hiermit wird ein LDAP-URL [RFC2255] angegeben, der den zu durchsuchenden Server, das Basisobjekt der Suche sowie das Attribut, das als Benutzername angenommen werden soll, enthält. Die allgemeine Form ist

```
ldap://<Rechner>:<Port>/<DN>?<Attribut>?<Suchbereich>?<Filter>
```

Dabei ist <Attribut> der Name des Attributs, dessen Wert mit dem Benutzernamen aus der HTTP-Anfrage verglichen wird; es ist möglich, gemäß der Syntax aus RFC 2255 eine Liste von Attributen anzugeben, aber nur das erste davon wird beachtet. Der Standardwert, wenn kein Attribut angegeben wurde, ist uid; sinnvollerweise wählt man ein Attribut, das im ganzen DIT für jeden Eintrag einen eindeutigen Wert hat. Der <Suchbereich> ist one oder sub; die laut RFC 2255 prinzipiell auch erlaubte Spezifikation base wird nicht unterstützt und implizit durch sub ersetzt, was auch der Standardwert ist. <Filter> ist ein gewöhnlicher LDAP-Suchfilter gemäß [RFC2254], wobei vom Apache-Server eine Maximallänge von etwa 8000 Zeichen festgelegt wird (dies sollte keine praktische Einschränkung darstellen). Hier ist der Standardwert wie üblich (objectClass=*), das auf alle Einträge passt. Der <Filter> wird implizit mit dem <Attribut> UND-verknüpft; bei einer Anfrage wie

```
ldap://ldap.example.com/dc=example,dc=com?cn?sub?(userid=*)
```

wird also, wenn der Benutzer einen Benutzernamen wie »Hugo Schulz« angibt, unter dem Strich mit einem Filter wie

```
(&(userid=*)(cn=Hugo Schulz))
```

gesucht. – <DN> bestimmt wie sonst den Basis-DN der Suchoperation.



Statt ldap:// können Sie auch ldaps:// schreiben, wenn Sie über SSL auf den LDAP-Server zugreifen wollen und Apache gegen eine SSL-fähige LDAP-Bibliothek gelinkt ist (unter Linux normalerweise der Fall).



Alternativ können Sie, statt ldaps:// zu verwenden, auch das zweite (optionale) Argument der Direktive angeben. Dessen mögliche Werte sind:

NONE Es wird eine unverschlüsselte Verbindung verwendet (entspricht ldap:// am Anfang des URL).

SSL Es wird eine direkt über LDAP verschlüsselte Verbindung verwendet (entspricht ldaps:// am Anfang des URL).

TLS (oder STARTTLS). Es wird eine unverschlüsselte Verbindung aufgebaut und diese dann in eine verschlüsselte Verbindung umgewandelt.



Den *<Port>* können Sie weglassen, sofern es sich um den TCP-Port 389 (für `ldap://`) bzw. den TCP-Port 636 (für `ldaps://`) handelt. Außerdem können Sie mehrere LDAP-Server angeben, durch Leerzeichen getrennt – mit etwas wie

```
ldap://ldap1.example.com ldap2.example.com/...
```

versucht Apache zunächst den ersten genannten Server zu erreichen, dann den zweiten. Klappt eine Verbindung, so behält Apache diese bei, bis entweder der betreffende Apache-Prozess endet oder die Verbindung zum LDAP-Server verloren geht; im letzteren Fall versucht Apache beginnend beim ersten Server, eine neue Verbindung aufzubauen.

AuthLDAPBindDN und AuthLDAPBindPassword Ein (optionaler) Benutzername und ein Kennwort, unter denen in der Autorisierungsphase versucht wird, mit dem LDAP-Server Verbindung aufzunehmen (um nach passenden Einträgen zu suchen; später versucht `mod_authnz_ldap` noch einmal einen Bindevorgang mit dem resultierenden DN und dem Kennwort aus der HTTP-Anfrage). Wenn diese Direktiven nicht angegeben werden, wird ein anonymer Bindevorgang versucht. Sie sollten diese Direktiven nur verwenden, wenn es nicht anders geht, und gegebenenfalls dafür sorgen, dass das `AuthLDAPBindPassword` nicht für alle Benutzer lesbar ist. Insbesondere sollten Sie es tunlichst vermeiden, hierfür den `rootdn` bzw. das `rootpw` aus der OpenLDAP-Konfiguration zu benutzen!

AuthLDAPAuthoritative {on|off} Bestimmt, ob noch andere Quellen für Benutzerdaten durchsucht werden sollen, wenn `mod_authnz_ldap` einen Benutzer abgewiesen hat. Der Standardwert ist `on`, es werden also keine anderen Quellen betrachtet.

7.2.3 Die Autorisierungsphase

Die wichtigste Direktive für die Autorisierungsphase ist – wie üblich bei Apache-Authentisierungsmodulen – **Require**. Diese Direktive kann bei `mod_authnz_ldap` die folgenden Formen annehmen:

Require valid-user Jeder Benutzer, der die Authentisierungsphase erfolgreich durchlaufen hat, bekommt Zugriff.

Require ldap-user <Benutzername> [<Benutzername> ...] Die in der Direktive aufgezählten Benutzer bekommen Zugriff. Implementiert wird das so: Wenn es `auth_ldap` gelingt, einen eindeutigen Eintrag im Verzeichnis zu finden, dann wird verglichen, ob der Eintrag einen der Benutzernamen aus der `Require`-Direktive enthält. Maßgeblich für den Vergleich ist dabei das Attribut aus dem `AuthLDAPURL`. Wird mit etwas wie

```
AuthLDAPURL http://ldap.example.com/ou=users,dc=example,dc=com?cn
```

zum Beispiel das `cn`-Attribut gesucht, dann können mehrere Benutzer etwa wie folgt zugelassen werden:

```
Require ldap-user "Hugo Schulz" "Susi Sorglos"
Require ldap-user "Gabriele Kerner"
```

(wenn Benutzernamen Leerzeichen enthalten, müssen sie in Anführungszeichen gestellt werden). Bei einem LDAP-Eintrag wie

```
dn: uid=nenal,dc=example,dc=com
objectClass: top
objectClass: person
```

```

cn: Gabriele Susanne Kerner
cn: Gabriele Kerner
cn: Nena
sn: Kerner

```

könnte Gabriele Susanne Kerner sich als »Gabriele Susanne Kerner«, »Gabriele Kerner« oder »Nena« authentisieren, obwohl nur eine »Require ldap-user«-Direktive in der Konfigurationsdatei steht. Wird im AuthLDAPURL statt cn das Attribut uid benutzt, so genügt die einzelne Require-Zeile

```
Require ldap-user hugo nena
```



Das Modul mod_authnz_ldap verwendet zur Überprüfung eine LDAP-Vergleichsoperation, keine Leseoperation. Das heißt, dass für das betrachtete Attribut nur Vergleichszugriff nötig ist, kein Lesezugriff.

Require ldap-group <DN> Gibt den DN einer LDAP-Gruppe an, deren Mitglieder Zugriff bekommen. Eine LDAP-Gruppe können Sie etwa wie folgt definieren:

```

dn: cn=Admins,dc=example,dc=com
objectClass: groupOfUniqueNames
uniqueMember: uid=hugo,dc=example,dc=com
uniqueMember: uid=nenana,dc=example,dc=com

```

Anschließend erlaubt eine Direktive wie

```
Require ldap-group cn=Admins,dc=example,dc=com
```

beiden Gruppenmitgliedern den Zugriff. Diese Sorte Authentisierung läßt sich über die Direktiven AuthLDAPGroupAttribute und AuthLDAPGroupAttributeIsDN steuern. Dabei können Sie die Direktive AuthLDAPGroupAttribute mehrmals angeben, um Attributnamen für die Überprüfung von Gruppenmitgliedschaft zu bezeichnen. Standardwert ist eine Liste mit den Attributen member und uniqueMember. Die Direktive AuthLDAPGroupAttributeIsDN kann die Werte on oder off annehmen; ist der Wert on, dann wird der DN eines Eintrags für die Überprüfung der Gruppenmitgliedschaft herangezogen, sonst der Benutzername aus der HTTP-Anfrage. Schickt der Client zum Beispiel den Benutzernamen nena, der mit dem Eintrag mit dem DN uid=nenana,dc=example,dc=com korrespondiert, dann wird bei der Einstellung on geprüft, ob das uniqueMember-Attribut der Gruppe (unter anderem) den Wert uid=nenana,dc=example,dc=com hat; ist die Einstellung off, so wird statt dessen nach dem Wert nena gesucht.

Require ldap-dn <DN> Wenn der für den Eintrag gefundene DN mit dem DN in der Require-Direktive übereinstimmt, wird der Zugriff gestattet, sonst abgewiesen. Der DN muss dabei *ohne* Anführungszeichen angegeben werden:

```
Require ldap-dn cn=Gabriele Susanne Kerner,dc=example,dc=com
```

(Wenn Sie mehrere unabhängige DN's auf diese Weise autorisieren wollen, brauchen Sie mehrere Require-Direktiven.) Die Direktive AuthLDAPCompareDNOnServer regelt, wie der Vergleich der DN's stattfindet; hat sie den Wert on (auch der Standardwert), dann wird der LDAP-Server zum Vergleich der DN's herangezogen. Dies ist die sicherere, aber auch langsamere Methode; bei off macht auth_ldap selbst einen reinen Zeichenkettenvergleich, was schneller geht, aber »falsche Negative« verursachen kann, bei denen eigentlich berechnigte Benutzer irrtümlich abgewiesen werden. Dank mod_ldap kann auch die serverbasierte Prüfung beschleunigt werden.

Require ldap-attribute *<Attribut>=<Wert>* ... Prüft, ob der LDAP-Eintrag des authentisierten Benutzers ein Attribut mit dem angegebenen Wert enthält. Ist dies der Fall, wird der Zugriff erlaubt, sonst abgewiesen:

```
Require ldap-attribute paidUp=True
```

Sie können auch mehrere Attribut-Wert-Paare angeben. Diese werden dann mit logischem ODER verknüpft, das heißt, irgendeine der Bedingungen muss passen:

```
Require ldap-attribute l="Frankfurt am Main" l=Darmstadt l=Mainz
```

Require ldap-filter *<Filterausdruck>* Hiermit können Sie einen beliebigen LDAP-Filterausdruck zur Autorisierung heranziehen. Der Filter wird auf dem LDAP-Server ausgewertet, und wenn einer der dabei resultierenden DNS auf den DN des authentisierten Benutzers passt, wird der Zugriff gestattet.



Der Unterschied zwischen `ldap-filter` und `ldap-attribute` besteht vor allem darin, dass `ldap-filter` eine Suchoperation auf dem LDAP-Filter auslöst, während `ldap-attribute` sich auf einen einfachen Vergleich beschränkt. Solange Sie nur einen einfachen Vergleich vornehmen wollen, sollten Sie, wenn möglich, `ldap-attribute` verwenden, da `ldap-filter` – vor allem mit einem großen Verzeichnis – ziemlich ineffizient sein kann.

7.3 Beispiele

Hier noch einige Beispiele für die Konfiguration von `mod_authnz_ldap`, adaptiert aus der Dokumentation des Moduls:

```
AuthLDAPURL ldap://ldap.example.com/ou=people,dc=example,dc=com
Require valid-user
```

Erlaubt jedem Benutzer aus dem LDAP-Personenverzeichnis den Zugriff. Einige Standardwerte wurden weggelassen, etwa `sub` für den Suchbereich und `(objectClass=*)` für den Suchfilter.

```
AuthLDAPURL ldap://ldap.example.com/ou=people,dc=example,dc=com?cn
Require valid-user
```

Dies entspricht im wesentlichen dem vorigen Beispiel, aber erlaubt den Zugriff auf der Basis des `cn`. Dies ist möglicherweise ein Problem, wenn mehrere Personen im Verzeichnis denselben `cn` haben (etwa in verschiedenen `ous`), da die Suche nur einen Eintrag als Ergebnis haben darf. Es ist alles in allem besser, nach einem garantiert eindeutigen Attribut zu suchen.

```
AuthLDAPURL ldap://ldap.example.com/dc=example,dc=com?uid
Require ldap-group cn=Admins,dc=example,dc=com
```

Erlaubt allen Mitgliedern der Gruppe `Admins` den Zugriff. Sie müssen sich über ihre `uid` anmelden.

```
AuthLDAPURL ldap://ldap.example.com/dc=example,dc=com?uid??(pagerID=*)
Require valid-user
```

Hier dürfen alle Leute zugreifen, die einen Piepser haben (wobei angenommen wird, dass das Attribut `pagerID` die Rufnummer des Piepsers enthält).

```
AuthLDAPURL ldap://ldap.example.com/dc=example,dc=com?uid?|(pagerID=*)(uid=hugo)
Require valid-user
```

Und hier dürfen sowohl die Piepser-Inhaber als auch der Benutzer hugo (der keinen Piepser hat) zugreifen. Dieser Ansatz ist viel bequemer als der ansonsten (etwa bei der »normalen« dateibasierten Authentisierung) nötige, wo man eine neue Gruppe definieren müsste, die dann irgendwie mit der der Piepser-Inhaber synchron gehalten werden müsste.

Das letzte Beispiel könnten Sie auch als

```
AuthLDAPURL ldap://ldap.example.com/dc=example,dc=com?uid
Require ldap-filter |(pagerID=*)(uid=hugo))
```

hinschreiben. Die Unterschiede zwischen den beiden Möglichkeiten sind subtil: In der Fassung mit `ldap-filter` kommen für die Authentisierung (also das ursprüngliche Binden an das Verzeichnis) grundsätzlich alle Verzeichniseinträge in Frage, und bei der Autorisierung wird dann entschieden, ob der DN des gefundenen Eintrags auch Bestandteil des `ldap-filter`-Suchergebnisses ist. In der Fassung mit `valid-user` dagegen werden schon beim Binden ans Verzeichnis nur die Einträge betrachtet, die alle Kriterien erfüllen. Das ist natürlich schneller, aber die Fassung mit `ldap-filter` ist flexibler, wenn beispielsweise alternativ noch eine Gruppenmitgliedschaft geprüft werden soll.



Beachten Sie die Interaktion von `mod_authnz_ldap` mit anderen Autorisierungsmodulen. In älteren Versionen der Apache-LDAP-Unterstützung wurden statt `Require ldap-user` und `Require ldap-group` die Direktiven `Require user` und `Require group` verwendet. Das ist heute nicht mehr so. Der Sinn der Sache besteht darin, die gleichzeitige Benutzung von `mod_authnz_ldap` mit anderen Autorisierungsmodulen zu erlauben. Beispielsweise könnten Sie etwas verwenden wie

```
AuthzLDAPAuthoritative off
AuthUserFile /var/www/htdocs/.htpasswd
<<<<<<
Require ldap-user "Hugo Schulz"
Require user admin
```

Die `Require`-Direktiven werden in der Autorisierungsphase betrachtet. Der Benutzer wurde bereits authentisiert, das heißt, entweder im LDAP-Verzeichnis oder in der Datei `/var/www/htdocs/.htpasswd` wurde ein Eintrag gefunden, der zu der von ihm im Browser angegebenen Kombination aus Benutzernamen und Kennwort passt. Jetzt wird zunächst geprüft, ob der LDAP-Eintrag des Benutzers (falls vorhanden) ein `cn`-Attribut mit dem Wert »Hugo Schulz« hat. Falls ja, wird der Zugriff erlaubt. Falls nein, wird überprüft, ob der authentisierte Benutzername tatsächlich `admin` ist. Hierum kümmert sich aber nicht `mod_authnz_ldap`, sondern das Apache-Modul `mod_authz_user`. Aus diesem Grund müssen Sie mit `AuthzLDAPAuthoritative off` erlauben, dass auch andere Module außer `mod_authnz_ldap` verwendet werden.



Die Reihenfolge, in der Autorisierungsmodule wie `mod_authnz_ldap` und `mod_authz_user` konsultiert werden, wird von der Reihenfolge bestimmt, in der diese Module beim Start von Apache geladen wurden, also von der Reihenfolge der `LoadModule`-Direktiven in der Apache-Konfiguration.

Übungen



7.1 [!3] Bauen Sie mit Apache eine einfache Webpräsenz auf (eine Seite, die eine »Hallo Welt«-Meldung ausgibt, genügt). Vergewissern Sie sich, dass

der Zugriff darauf ohne Authentisierung funktioniert. Konfigurieren Sie anschließend das Modul `mod_authnz_ldap` so, dass der Zugriff nur für authentifizierte Benutzer gestattet ist, die im LDAP-Verzeichnis stehen (unauthentifizierte Benutzer sollen abgewiesen werden). Testen Sie das mit einem Benutzer, der einen LDAP-Eintrag wie

```
dn: uid=emil,ou=Users,dc=example,dc=com
objectClass: inetOrgPerson
uid: emil
cn: Emil Huber
sn: Huber
userPassword: geheim
```

hat, indem Sie im Browser den Benutzernamen `emil` und das Kennwort `geheim` eingeben. Vergewissern Sie sich, dass Sie mit Kombinationen aus Benutzernamen und Kennwörtern, die nicht mit Verzeichniseinträgen korrespondieren, keinen Zugriff bekommen. Überzeugen Sie sich ferner, dass das auch gilt, wenn Sie den Benutzernamen `emil` mit einem anderen (falschen) Kennwort eingeben.



7.2 [2] Ändern Sie die Konfiguration aus der vorigen Aufgabe so, dass Benutzer zur Authentisierung im Browser nicht ihren Benutzernamen, sondern ihren vollen Namen angeben müssen. Halten Sie das für eine gute Idee?



7.3 [1] Ändern Sie die Konfiguration aus Übung 7.1 so, dass *nur* der Benutzer `emil` Zugriff auf die Seite bekommt. Vergewissern Sie sich, dass andere Benutzer im Verzeichnis keinen Zugriff bekommen, auch wenn deren Benutzernamen und Kennwörter im Browser korrekt eingegeben werden.

7.4 Das Modul `mod_ldap`

Das eben beschriebene Modul `mod_authnz_ldap` verwendet, um tatsächlich mit dem LDAP-Server zu reden, das Modul `mod_ldap`, das einen Cache und einen Verbindungspool zur Verfügung stellt.

Verbindungspool Der Verbindungspool macht es möglich, eine Verbindung zum LDAP-Server über mehrere HTTP-Anfragen hinweg offen zu halten und so Zeit zu sparen. Außerdem öffnet `mod_ldap` bei Bedarf weitere Verbindungen zum Server, damit Anfragen sich nicht um die existierenden Verbindungen »streiten«.

Cache Der Cache dient dazu, Anfragen an den LDAP-Server überhaupt zu vermeiden. Dazu unterstützt `mod_ldap` zwei Arten von Zwischenspeicher:

Such-/Binde-Cache In diesem Cache werden alle Suchoperationen zwischengespeichert, die zu einer erfolgreichen Bindung an den Server führten. (Negative Resultate, also solche, bei denen die Suchoperation kein Ergebnis lieferte oder das Suchergebnis nicht zu einer erfolgreichen Bindung führte, werden nicht zwischengespeichert.) `mod_ldap` speichert den im Browser eingegebenen Benutzernamen, den dazugehörigen DN, das im Browser eingegebene Kennwort und den Zeitpunkt des Bindens. Wird eine neue Verbindung mit demselben Benutzernamen versucht, dann vergleicht `mod_ldap` das dann eingegebene Kennwort mit dem gespeicherten Kennwort. Stimmen diese überein und liegt der erfolgreiche Bindevorgang nicht zu weit in der Vergangenheit, wird die Suchphase übersprungen.



Die Direktive `LDAPCacheTTL` gibt an, wie alt Einträge im Cache maximal sein dürfen, um benutzt werden zu können. Das Argument ist ein Wert in Sekunden, Vorgabe ist

```
LDAPCacheTTL 600
```



Mit `LDAPCacheEntries` können Sie die Größe des Such-/Binde-Caches vorgeben (in Einträgen). Standardwert ist 1024. Der Wert 0 schaltet den Such-/Binde-Cache aus.

Operations-Caches `mod_ldap` führt zwei weitere Caches, um Vergleichsoperationen zu beschleunigen. Der erste speichert die Ergebnisse von DN-Vergleichen, der zweite die Ergebnisse von Tests auf LDAP-Gruppenmitgliedschaft.



Die Direktive `LDAPOpCacheTTL` entspricht sinngemäß der Direktive `LDAPCacheTTL`. Auch hier ist der Standardwert 600 Sekunden.



Die Direktive `LDAPOpCacheEntries` verhält sich zur Direktive `LDAPCacheEntries` wie `LDAPOpCacheTTL` zu `LDAPCacheTTL`.

Bei einem Apache, der Seitenzugriffe über `mod_authnz_ldap` authentisiert, kann Caching den Durchsatz durchaus verdoppeln oder verdreifachen.



Wenn Sie ein Auge auf die Leistung des Cachespeichers haben wollen, können Sie das mit einer Definition wie

```
<Location /cache-info>
  Order deny,allow
  Deny from all
  Allow from 127.0.0.1
  SetHandler ldap-status
</Location>
```

`tun` (aus Sicherheitsgründen schränken wir den Zugriff auf den Rechner mit dem Apache-Server selber ein). Den URL im `location`-Block dürfen Sie sich natürlich frei aussuchen; hier funktionieren dann Zugriffe auf `http://localhost/cache-info`.

Zusammenfassung

- Eine mögliche Quelle für Benutzerinformationen für den Apache-Server ist ein LDAP-Verzeichnis.
- `mod_authnz_ldap` arbeitet in zwei Phasen: der Authentisierungs- und der Autorisierungsphase. In der Authentisierungsphase wird die Identität eines Benutzers bestimmt, in der Autorisierungsphase bekommt ein identifizierter Benutzer Rechte zugesprochen.
- Einträge im Verzeichnis werden anhand des HTTP-Benutzernamens über vordefinierte URLs gefunden, die die Angabe weiterer Suchfilter zulassen.
- Verschiedene Formen der `Require`-Direktive erlauben die Vergabe von Zugriffsrechten an alle registrierten Benutzer, einzelne Benutzer oder Mitglieder bestimmter LDAP-Gruppen.

Literaturverzeichnis

RFC2254 T. Howes. »Lightweight Directory Access Protocol (v3): The String Representation of LDAP Search Filters«, Dezember 1997.

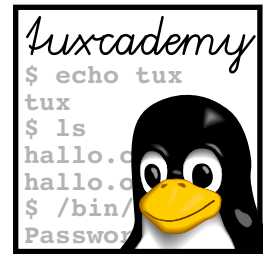
<http://www.ietf.org/rfc/rfc2254.txt>

RFC2255 T. Howes, M. Smith. »The LDAP URL Format«, Dezember 1997.

<http://www.ietf.org/rfc/rfc2255.txt>

RFC2617 J. Franks, P. Hallam-Baker, J. Hostetler, et al. »HTTP Authentication: Basic and Digest Access Authentication«, Juni 1999.

<http://www.ietf.org/rfc/rfc2617.txt>



8

LDAP, Postfix und Dovecot

Inhalt

8.1	Überblick.	96
8.2	Die Datenbankstruktur.	97
8.3	Postfix	100
8.3.1	Allgemeines.	100
8.3.2	Mail-Zustellung	101
8.4	Dovecot	102
8.4.1	Allgemeines.	102
8.4.2	Authentisierung und Benutzerdaten	103
8.5	Extras	106
8.5.1	SMTP-AUTH	106
8.5.2	Mail-Weiterleitung	106

Lernziele

- LDAP für die Konfiguration des Postfix-Mailserver einsetzen können
- Authentisierungs- und Benutzerdaten für den IMAP/POP3-Server Dovecot in einem LDAP-Verzeichnis verwalten können
- Einen Mailserver für virtuelle Benutzer und Domänen aufbauen können

Vorkenntnisse

- LDAP-Kenntnisse (aus den vorigen Kapiteln)
- Grundkenntnisse über die Konfiguration und den Betrieb von Postfix und Dovecot (etwa aus *Linux als Mailserver*)

8.1 Überblick

Dieses Kapitel beschreibt, wie Sie LDAP zur Speicherung von Konfigurationsdaten für Postfix einsetzen können. Wir erklären die Konfiguration von Postfix für einen Mailserver, der Nachrichten für mehrere Domains empfangen und bearbeiten können soll. Die Adressaten der Mail müssen keine Systembenutzer sein; sie bekommen ihre Nachrichten entweder an andere Adressen weitergeleitet oder in lokale Postfächer zugestellt, die sie über IMAP oder POP3 abrufen können. Dafür setzen wir den POP3/IMAP-Server Dovecot ein.



Die offizielle Web-Seite von Postfix ist <http://www.postfix.org/>. Dort finden Sie im Abschnitt »Documentation« unter »Lookup tables (databases)« auch einen Verweis auf das »Postfix LDAP Howto«.



Mehr über Dovecot erfahren Sie von <http://www.dovecot.org/>. Zum Thema »Dovecot und LDAP« empfehlen wir Ihnen <http://wiki.dovecot.org/AuthDatabase/LDAP> und die Seiten, auf die diese Seite verweist.



Eine frühere Version dieser Schulungsunterlage diskutierte Courier-IMAP statt Dovecot. Allerdings ist Dovecot, wenn es um POP3- und IMAP-Server geht, inzwischen eindeutig die bessere Wahl. Da wir mit der Zeit gehen und Ihnen natürlich den aktuellen Stand der Technik präsentieren wollen, ist es nur konsequent, hier umzuschwenken. (Am Rande bemerkt ist das LPI für die Prüfung LPI-202 genau derselben Meinung.)

LDAP bietet sich für diese Anwendung an – vor allem, wenn Sie Daten über Ihre Benutzer sowieso schon in einem LDAP-Verzeichnis vorhalten. Einige der wichtigsten Vorteile sind im einzelnen:

- Der Wartungsaufwand im Betrieb sinkt, da Postfix und Dovecot auf denselben Datenbestand zugreifen können. Sie müssen Benutzer also nur an einer Stelle eintragen.
- Sie können neue Domains und Benutzer mit denselben Werkzeugen verwalten, die Sie ohnehin schon zur Verwaltung Ihres LDAP-Verzeichnisses benutzen.
- Sie müssen nicht nach jeder Datenbankänderung `postmap` aufrufen, damit Postfix die neuen Daten finden kann.
- Der Zugriff auf die Daten ist effizienter (wichtig bei vielen Domains mit vielen Benutzern). Insbesondere können Sie zur Lastverteilung mehrere Mailserver betreiben, die auf dasselbe Verzeichnis zugreifen.
- Sie können die Daten über mehrere LDAP-Server hinweg replizieren und damit hochverfügbare Infrastrukturen erstellen, die den Ausfall einzelner LDAP-Server tolerieren.

Als wesentlicher Nachteil steht diesen Vorteilen ein höherer anfänglicher Konfigurationsaufwand gegenüber.

Postfix und Dovecot unterstützen diverse Möglichkeiten zur Speicherung der Benutzerdaten. Neben LDAP kämen auch relationale Datenbanken wie PostgreSQL oder MySQL in Frage, die ebenfalls gut geeignet, wenn auch weniger flexibel sind.

Das Dovecot-Paket enthält nicht nur den eigentlichen POP3- und IMAP-Server, sondern auch einen lokalen Auslieferungs-Agenten (LDA), den Postfix aufrufen kann, um Nachrichten in die Dovecot-basierten Postfächer von Benutzern (auch »virtuellen« Benutzern, die nicht in der System-Benutzerdatenbank stehen) zuzustellen. Dies vereinfacht die Konfiguration von Postfix ungemein, und wir benutzen diesen Ansatz in diesem Kapitel.

Wir konzentrieren uns hier auf die Speicherung von Konfigurationsdaten und den Zugriff darauf und lassen andere wichtige Aspekte der Konfiguration eines

Mailserver, etwa Spam- und Virenfilerung, absichtlich aus dem Spiel. Mehr hierüber erfahren Sie zum Beispiel aus der Linup-Front-Schulungsunterlage *Linux als Mailserver*.

Übungen



8.1 [2] Welche Vorteile könnte es haben, eine SQL-Datenbank statt eines LDAP-Verzeichnisses zur Datenhaltung für den Mailserver einzusetzen?

8.2 Die Datenbankstruktur

Postfix erlaubt es Ihnen, jeden Parameter aus der zentralen Konfigurationsdatei `main.cf`, der eine Liste von Werten enthält, in einem LDAP-Verzeichnis abzulegen. Parameter, die auf Tabellen verweisen, können auf das LDAP-Verzeichnis als Datenquelle zurückgreifen.



»Tabellen« in Postfix dienen dazu, »Schlüssel« (etwa E-Mail-Adressen) auf »Werte« (etwa andere E-Mail-Adressen zur Weiterleitung) abzubilden. Im einfachsten Fall werden sie über Textdateien mit zwei durch Leerzeichen oder Tabs getrennte Spalten realisiert, wo in jeder Zeile die erste Spalte einen »Schlüssel« und die zweite den dazugehörigen Wert darstellt.



Ein Parameter, der eine Liste von Werten enthält, kann über eine Tabelle implementiert werden, in der nur der »Schlüssel« gefunden werden muss; der dazugehörige Wert in der Tabelle ist unerheblich.

In unserem Fall brauchen wir im Wesentlichen drei verschiedene Datenstrukturen, die für eine Darstellung im LDAP-Verzeichnis in Frage kommen: Datenstrukturen

- Postfix braucht eine Liste von Domains, für die er sich »zuständig« fühlen soll. In der Postfix-Konfiguration entspricht diese Liste dem Parameter `virtual_mailbox_domains`.
- Er braucht eine Liste von E-Mail-Adressen, die in den betreffenden Domains gültig sind. Dies dient zur »Eingangsprüfung« von neu ankommenden Nachrichten, damit Mail an ungültige Adressen so früh wie möglich abgelehnt werden kann – auf jeden Fall, bevor die Nachricht angenommen und in die lokale Warteschlange eingereiht wurde, da sie sonst »gebouncet«, also als Fehlermeldung an den angegebenen Absender zurückgeschickt werden muss.



In der Praxis ist das ein Problem, da die Absenderadressen typischer unerwünschter Nachrichten (»Spam«) in der Regel gefälscht oder frei erfunden sind. Damit werden Unbeteiligte durch die Fehlermeldungen, sogenannten *backscatter*, belästigt. Die Abhilfe besteht darin, Nachrichten an nicht vorhandene Adressaten schon während des SMTP-Dialogs zurückzuweisen, damit Sie erst gar keine Verantwortung für ihre weitere Zustellung übernehmen.

In Postfix können Sie Quellen für diese Tabelle über den Parameter `virtual_mailbox_maps` konfigurieren. In der Praxis kann Dovecot dieselbe Liste verwenden, um lokale Postfächer zu identifizieren und ihnen Kennwörter zuzuordnen, die die Benutzer dann beim Abfragen angeben müssen.

- Für die Weiterleitung von Nachrichten, die an bestimmte Adressen geschickt wurden, an andere Adressaten benötigen wir eine Weiterleitungstabelle. Postfix gestattet die Konfiguration solcher Tabellen über den Parameter `virtual_alias_maps`.

```
attributetype ( 1.3.6.1.4.1.15534.3.1.1 NAME 'mailDomainName'
  DESC 'Domain name for mail delivery'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

objectclass ( 1.3.6.1.4.1.15534.3.2.1 NAME 'mailDomain'
  SUP top AUXILIARY
  DESC 'Domain that can receive mail'
  MUST ( mailDomainName ) )

attributetype ( 1.3.6.1.4.1.15534.3.1.2 NAME 'mailAddress'
  DESC 'Recipient mail address'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

objectclass ( 1.3.6.1.4.1.15534.3.2.2 NAME 'mailRecipient'
  SUP top AUXILIARY
  DESC 'Entity that can receive mail'
  MUST ( mailAddress ) )

attributetype ( 1.3.6.1.4.1.15534.3.1.3 NAME 'mailForwardAddress'
  DESC 'Address to forward mail to'
  EQUALITY caseIgnoreIA5Match
  SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

objectclass ( 1.3.6.1.4.1.15534.3.2.3 NAME 'mailForwarder'
  SUP top AUXILIARY
  DESC 'Entity that can forward mail'
  MUST ( mailAddress ) MAY ( mailForwardAddress ) )
```

Bild 8.1: Attribute und Objektklassen für Mail

Auf der LDAP-Seite könnten wir die Liste der gültigen E-Mail-Adressen über die üblichen Objektklassen für Benutzer (etwa `inetOrgPerson`) realisieren, indem wir die Attribute `mail` und `userPassword` heranziehen. Für die Domainliste und die Weiterleitung gibt es bisher keine Übereinkunft. Um nicht existierende halb fertige Ansätze zweckzuentfremden, ist es am vernünftigsten, sich selbst einige Attribute und Objektklassen zu definieren, die das Nötige tun (Bild 8.1):



Es hat in der Vergangenheit verschiedene Bestrebungen gegeben, standardisierte Attribute und Objektklassen für die Behandlung von Mail zu definieren. Diese sind allerdings zum größten Teil im Sande verlaufen.

- Eine `mailDomain` ist ein Eintrag, das unter dem in `mailDomainName` abgelegten Namen Mail zur Zustellung an lokale Adressen annimmt. Sie können diese AUXILIARY-Objektklasse zum Beispiel an der Wurzel Ihres DIT unterbringen:

```
dn: dc=example,dc=com
objectClass: organization
objectClass: dcObject
objectClass: mailDomain
dc: example
o: Beispiel
mailDomainName: example.com
mailDomainName: example.net
```

(Wie Sie sehen, können Sie durchaus mehr als einen Domainnamen angeben.)

- Entsprechend ist ein `mailRecipient` ein Eintrag, dessen konkrete(n) Adresse(n) im `mailAddress`-Attribut angegeben ist. Wir verwenden mit Absicht nicht das `mail`-Attribut, weil das in allen möglichen Einträgen vorkommen kann, die Sie vielleicht nicht zur Zustellung heranziehen wollen (stellen Sie sich vor, dass Sie ein LDAP-basiertes »Telefonbuch« Ihrer Kunden führen, wo auch deren E-Mail-Adressen abgelegt werden können), sondern werden später die in Frage kommenden Einträge durch einen Filter wie

```
(&(objectClass=mailRecipient)(mailAddress=<Was auch immer>))
```

lokalisieren. Diese Objektklasse können Sie zum Beispiel `person` und seinen Kindern hinzufügen, aber grundsätzlich auch Einträgen vom Typ `organizationalRole` (wobei Sie den nächsten Punkt betrachten sollten).



Auch hier gilt, dass eine Person durchaus mehr als eine Adresse haben kann. Es spricht nichts gegen

```
dn: uid=hugo,ou=Users,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: mailRecipient
uid: hugo
cn: Hugo Schulz
sn: Schulz
mail: hugo.schulz@example.com
mailAddress: hugo@example.com
mailAddress: hugo.schulz@example.com
```

um an `hugo@example.com` oder auch `hugo.schulz@example.com` adressierte Mail an Hugo Schulz zuzustellen. Das `mail`-Attribut könnte hier dazu dienen, die bevorzugte Adresse für *ausgehende* Mail zu bestimmen (hier `hugo.schulz@example.com`).

- mailForwarder erlaubt es, Einträge zu definieren, die zwar eine Adresse haben, aber an sie adressierte Nachrichten anderswohin weiterleiten. Diese Objektklasse empfiehlt sich zum Beispiel zur Kombination mit organizationalRole:

```
dn: cn=hostmaster,dc=example,dc=com
objectClass: organizationalRole
objectClass: mailForwarder
description: Der DNS-Betreuer
cn: hostmaster
mailAddress: hostmaster@example.com
mailForwardAddress: hugo@example.com
```

Die Definitionen aus Bild 8.1 könnten Sie in einer Datei namens `/etc/ldap/schema/ldap.schema` hinterlegen und über

```
include /etc/ldap/schema/mail.schema
```

verfügbar machen. Der Ordnung halber beginnen die Namen aller darin enthaltenen Attributtypen und Objektclassen mit `mail`.



Wir haben in diesem Beispiel der Einfachheit halber den OID-Präfix der Linup Front GmbH benutzt, um die OIDs für die Attributtypen und Objektclassen zu vergeben. Bitte verwenden Sie spätestens in dem Moment, wo Sie auf der Basis dieses Kapitels Ihren unternehmensweiten Mail-Dienst konfigurieren, einen OID-Präfix, den Sie sich selber registriert haben. Vielen Dank.

Übungen



8.2 [!2] Installieren Sie das `mail.schema` in Ihrem OpenLDAP-Server und legen Sie ein paar Testeinträge an bzw. bereichern Sie existierende Einträge um die entsprechenden Attribute. Denken Sie dabei auch an `mailDomain`.

8.3 Postfix

8.3.1 Allgemeines

LDAP-Unterstützung

Bevor Sie Postfix mit LDAP verwenden, sollten Sie sicherstellen, dass Ihr Postfix überhaupt LDAP-Unterstützung enthält. Das erledigen Sie mit dem Kommando `postconf`:

```
$ postconf -m | grep ldap
ldap
```

Liefert das Kommando keine Ausgabe, dann müssen Sie die LDAP-Unterstützung nachinstallieren. (Im schlimmsten Fall bedeutet das eine Neuübersetzung von Postfix aus dem Quellcode.)



Bei Debian GNU/Linux (und Ubuntu) finden Sie die LDAP-Unterstützung für Postfix im Paket `postfix-ldap`.

Postfix unterstützt, wie erwähnt, den Zugriff auf LDAP für Tabellen und »Listenparameter«. Letztere gelten dabei als Sonderfall des Tabellenzugriffs, bei dem es nur auf die Anwesenheit des gesuchten »Schlüssels« im Verzeichnis ankommt, nicht auf den dazugehörigen Wert.

Konfigurationsdatei

Um auf eine Tabelle im LDAP zuzugreifen, müssen Sie eine Konfigurationsdatei anlegen, die den LDAP-Zugriff im Detail beschreibt. Auf diese Datei verweisen Sie dann in der Definition des dazugehörigen Parameters in der `main.cf`-Datei, etwa wie

```
virtual_mailbox_domains = ldap:/etc/postfix/ldap-vmbd.cf
```

In der Datei /etc/postfix/ldap-vmbd.cf steht dann etwas wie

```
server_host = ldap://ldap.example.com/           LDAP-Server
search_base = dc=example,dc=com                 Ausgangspunkt der Suche
scope = base                                     Suchbereich: Nur Ausgangsobjekt
bind = no                                       Nicht an den Server binden
query_filter = (&(objectClass=mailDomain)(mailDomainName=%s))
result_attribute = mailDomainName              Eigentlich egal
```



Eine komplette Liste der möglichen Einstellungen finden Sie in `ldap_table(5)`.

Testen können Sie diese Konfiguration mit dem `postmap`. Wenn Sie nach einem Domainnamen fragen, der als Wert eines `mailDomainName`-Attributs des Eintrags `dc=example,dc=com` existiert, wird er als Ergebnis ausgegeben. Fragen Sie nach einem nicht existierenden Domainnamen, bleibt das Ergebnis leer:

```
$ postmap -q example.com ldap:/etc/postfix/ldap-vmbd.cf
example.com
$ postmap -q example.org ldap:/etc/postfix/ldap-vmbd.cf
$ _
Leere Ausgabe
```



Wenn das aus irgendwelchen Gründen schiefgeht, können Sie zur Fehlersuche die Option `-v` angeben, um ein ausführlicheres Ablaufprotokoll zu erhalten.

Übungen



8.3 [!2] Erstellen Sie die Konfigurationsdateien für den Postfix-Zugriff auf Ihren LDAP-Server und testen Sie sie mit `postmap`.



8.4 [2] In unserem Beispiel betrachten wir nur den Eintrag an der Wurzel des DIT für die Bestimmung der anhand des Verzeichnisses zustellbaren Domains. Ist das vernünftig? Welche Vor- und Nachteile hat dieser Ansatz?

8.3.2 Mail-Zustellung

Wenn Postfix eine Nachricht empfängt, überprüft er zunächst, ob er für die endgültige Zustellung der Nachricht zuständig ist. Das ist der Fall, wenn die Domain der Empfängeradresse aus dem SMTP-Dialog (genaugenommen dem `RCPT-TO`-Kommando) in den (Listen-)Werten von mehreren möglichen Postfix-Parametern steht. Der Parameter `mydestination` zum Beispiel listet die Domains auf, für die Postfix eine »lokale« Zustellung vornehmen soll. Uns interessiert allerdings die »virtuelle« Zustellung, die wir über Dovecot laufen lassen werden. Dieses »virtuelle« Zustellung

Zustellverfahren wählt Postfix für Mail an Adressen in den Domains, die im Parameter `virtual_mailbox_domains` stehen. Wie Sie diese Abfrage ans LDAP-Verzeichnis delegieren, haben Sie ja im vorigen Abschnitt gesehen. Im nächsten Schritt prüft Postfix, ob die Empfängeradresse tatsächlich gültig ist. Für Domains in `virtual_mailbox_domains` bedeutet das, dass die Adresse in einer Tabelle auftauchen muss, auf die der Parameter `virtual_mailbox_maps` verweist. Auch hier kommt es auf den Rückgabewert nicht an; die reine Anwesenheit der Adresse in der Tabelle genügt. Definiert wird das ganz analog zu der Domainprüfung im vorigen Abschnitt:

```
virtual_mailbox_maps = ldap:/etc/postfix/ldap-vmm.cf
```

mit der Konfigurationsdatei

server_host = ldap://ldap.example.com/	<i>LDAP-Server</i>
search_base = ou=people,dc=example,dc=com	<i>Ausgangspunkt</i>
scope = one	<i>Suchbereich: Unter-Ebene</i>
bind = no	<i>Nicht an den Server binden</i>
query_filter = (&(objectClass=mailRecipient)(mailAddress=%s))	
result_attribute = mailAddress	<i>Eigentlich egal</i>

Zustellung über Dovecot Den Rest der Arbeit übernimmt Dovecot, wenn Sie die Zustellung von Mail an »virtual«-Domains an den LDA von Dovecot delegieren. Am bequemsten erreichen Sie das über eine Transportdefinition wie die folgende in `master.cf`:


```
dovecot unix - n n - - pipe
flags=DRhu user=vmail:vmail argv=/usr/lib/dovecot/deliver
-f ${sender} -d ${user}@${nexthop} -n -m ${extension}
```


zusammen mit der Einstellung (in `main.cf`):

```
virtual_transport = dovecot
dovecot_destination_recipient_limit = 1
```

Damit das tatsächlich funktioniert, sind noch einige Vorbereitungen auf der Dovecot-Seite nötig, die wir im nächsten Abschnitt besprechen.

Übungen

 **8.5** [!2] Nehmen Sie die in diesem Abschnitt besprochenen Einstellungen an Postfix vor. Was passiert, wenn Sie jetzt Mail an eine im Verzeichnis definierte Adresse schicken?

 **8.6** [2] Wofür ist die Einstellung

```
dovecot_destination_recipient_limit = 1
```

nötig?

8.4 Dovecot

8.4.1 Allgemeines

Aufgabe von Dovecot Die Aufgabe von Dovecot ist es einerseits, Nachrichten von Postfix zu übernehmen und in die Postfächer von Benutzern zu schreiben, und andererseits, Benutzern über POP3 oder IMAP Zugriff auf diese Postfächer zu gewähren. Der große Vorteil dieses Arrangements besteht darin, dass Postfix sich nur noch um die Gültigkeitsprüfung der Empfängeradressen kümmern muss und nicht mehr um die Details der Zustellung. Ort und Eigentümer der Postfach-Dateien werden ausschließlich in Dovecot konfiguriert. Ein weiterer Vorteil besteht darin, dass Dovecot im Gegensatz zum `virtual`-Mailer von Postfix in der Lage ist, neue Postfächer selbsttätig mit den korrekten Rechten anzulegen. Neue Benutzer müssen Sie also nur im LDAP-Verzeichnis eintragen; auf dem Mailserver sind keine weiteren Aktionen notwendig.

Postfächer In unserem Beispiel platzieren wir die Postfächer im Verzeichnis `/srv/mail`, das wir dem Benutzer `vmail` mit der primären Gruppe `vmail` zusprechen. (Alle Postfächer gehören aus der Sicht von Linux diesem Benutzer.) Anlegen können Sie den Benutzer mit einem Kommando wie

```
# adduser --system --group --home /srv/mail --uid 60000 vmail
```


(für Debian GNU/Linux; das `--system` sorgt dafür, dass `/srv/mail` nicht mit Kram aus `/etc/skel` zugemüllt wird, während die explizite `--uid` jenseits der Mindestgrenze ist, die Dovecot standardmäßig für die UIDs von Benutzern annimmt – sie sollte natürlich anderweitig unbenutzt sein).

Hier sind die für Dovecot notwendigen Konfigurationseinstellungen, die Sie in der Datei `/etc/dovecot/dovecot.conf` vornehmen müssen:

`/etc/dovecot/dovecot.conf`

```
mail_location = maildir:/srv/mail/%d/%n
mail_uid = vmail
mail_gid = vmail
```

Diese drei Parameter definieren die Ordnerstruktur für Postfächer (`%d` und `%n` entsprechen jeweils der Domain und dem lokalen Teil der Adresse des Postfach-Eigentümers) und den Benutzer- und den Gruppennamen, unter dem Dovecot auf die Postfächer zugreift.

Der LDA besteht auf der Definition der Postmaster-Adresse für allfällige Fehlermeldungen:

```
protocol lda {
    postmaster_address = postmaster@example.com
}
```

Außerdem braucht der LDA über das »Master«-Socket Zugriff auf den eigentlichen Dovecot-Server:

```
auth default {
    # Hier definieren wir gleich noch den LDAP-Zugriff
    socket listen {
        master {
            path = /var/run/dovecot/auth-master
            mode = 0660
            group = vmail
        }
    }
}
```

8.4.2 Authentisierung und Benutzerdaten

Dovecot verwendet in der Theorie zwei verschiedene Datenbanken für die Authentisierung (Benutzernamen und Kennwörter) und die benutzerspezifischen Daten für die Postfächer (den Ort, die Benutzer- und Gruppenidentitäten, und weitere benutzerspezifische Informationen wie etwa eine maximale Postfachgröße). Grundsätzlich können diese benutzerspezifischen Daten für jedes Postfach anders sein, aber wir nutzen hier den Umstand aus, dass wir den Postfachort schon weiter vorne über `mail_location` festgelegt haben. Eine weitere Definition in der Benutzer-Datenbank können wir uns also sparen. Auch die Benutzer- und Gruppenidentitäten sind konstant und müssen darum nicht im Verzeichnis nachgeschlagen werden.

Datenbanken

Was wir im Verzeichnis nachschlagen müssen, sind der Benutzername und das dazugehörige Kennwort eines Postfach-Inhabers. Bei Dovecot steht die LDAP-Konfiguration dafür in einer separaten Konfigurationsdatei `/etc/dovecot/dovecot-ldap.conf`, die wir gleich betrachten. In der Datei `/etc/dovecot/dovecot.conf` müssen Sie die Zeile

```
# Hier definieren wir gleich noch den LDAP-Zugriff
```

aus unserem Beispiel durch

```
passdb ldap {
  args = /etc/dovecot/dovecot-ldap.conf
}
userdb static {
  args = uid=vmail gid=vmail
}
```

ersetzen.

dovecot-ldap.conf Hier sind die relevanten Einstellungen in der Datei dovecot-ldap.conf:

```
uris = ldap://ldap.example.com/           LDAP-Server
dn = cn=dovecot,dc=example,dc=com        DN zum Binden
dnpass = dovecot123                      Kennwort dafür
base = ou=people,dc=example,dc=com      Ausgangspunkt der Suche
scope = onelevel                         Suchbereich: Nur die direkten Kinder
pass_filter = (&(objectClass=mailRecipient)(mailAddress=%u))
pass_attrs = mailAddress=user,userPassword=password
```

Der Parameter `pass_filter` enthält einen LDAP-Filter, mit dem Sie den passenden Eintrag für den Benutzer im Verzeichnis suchen können. Wird er gefunden (wovon Sie im Prinzip ausgehen können, weil Postfix ihn ja auch schon einmal gefunden hat), dann bestimmt `pass_attrs`, dass Dovecot den Wert des LDAP-Attributs `mailRecipient` als Benutzername und den von `userPassword` als Kennwort ansieht.

Kennwörter Damit das funktioniert, muss Dovecot die Kennwörter der Benutzer aus dem LDAP-Verzeichnis auslesen können. Im Beispiel regeln wir das, indem wir für Dovecot einen Eintrag im Verzeichnis anlegen, der ihm das Anmelden erlaubt – etwas wie

```
dn: cn=dovecot,dc=example,dc=com
objectClass: organizationalRole
objectClass: simpleSecurityObject
cn: dovecot
userPassword: dovecot123                Besser verschlüsselt
```

Dazu brauchen Sie natürlich auch eine modifizierte Zugriffsregel für die Kennwörter, à la

```
access to attrs=userPassword,shadowLastChange
  by self write
  by anonymous auth
  by dn="cn=dovecot,dc=example,dc=com" read
  by * none
```

Standardmäßig geht Dovecot davon aus, dass LDAP-Kennwörter mit dem Schema `CRYPT` verschlüsselt sind, also dem traditionellen DES-basierten Verschlüsselungsverfahren. Das ist nicht der Gipfel der Sicherheit, aber das einzige Verfahren, das in Dovecot und OpenLDAP genauso heißt. Auf der anderen Seite ist es in der Praxis nicht so schlimm, wie es sich anhört, da Dovecot hierfür die `crypt(3)`-Funktion der C-Bibliothek verwendet, die auch mit einigen anderen Verschlüsselungsverfahren umgehen kann, wenn man ihr die verschlüsselten Kennwörter im richtigen Format vorlegt.

Authentisierung testen Die Authentisierung testen Sie am bequemsten über den POP3-Server mit einem geeigneten Client, etwa `netcat`:

```
$ nc localhost 110
+OK Dovecot ready.
USER susi@example.com
+OK
```

```
PASS susi123
+OK Logged in.
QUIT
+OK Logging out.
```



Sollte es Probleme geben, dann stellen Sie in `dovecot.conf` den Parameter `auth_debug` auf `yes` und probieren Sie es nochmal. Anschließend sollten im Systemprotokoll ausführlichere Fehlermeldungen zu finden sein.

Wenn dieser Schritt geklappt hat, können Sie versuchen, Mail an eine Adresse auf dem Server (in unserem Beispiel etwa `susi@example.com` zu schicken, zum Beispiel mit dem universellen Mailserver-Testprogramm `swaks`: Mail testen

```
$ swaks -s localhost -f hugo@example.com -t susi@example.com
```

Diese Nachricht sollte von Postfix angenommen werden (das erkennen Sie an der Ausgabe von `swaks`) und an den `deliver`-Prozess von Dovecot weitergereicht werden (das erkennen Sie am Inhalt der Datei `/var/log/mail.log`. Bei Schwierigkeiten mit dem `deliver`-Prozess (etwa wenn es mit dem LDAP-Zugriff nicht klappt), landet die Nachricht wieder in der Warteschlange von Postfix und kann dort mit einem

```
# sendmail -q
```

erneut in die Zustellung gegeben werden. Wenn alles funktioniert hat, sollte die neue Nachricht im Postfach von `susi@example.com` auftauchen (schauen Sie mit `ls` unter `/srv/mail` und dort über POP3 oder IMAP zugänglich sein.



Wenn Sie für einen Benutzer mehrere äquivalente Adressen konfigurieren – das `mailAddress`-Attribut darf ja mehr als einen Wert haben –, dann ist es mitunter wichtig, dass dieser Benutzer nicht plötzlich mehr als ein Postfach hat. Um dafür zu sorgen, dass Dovecot alle Adressen eines Benutzers als gleichwertig ansieht, sollten Sie in der Datei `dovecot-ldap.conf` statt

```
pass_attrs = mailAddress=user,userPassword=password
```

lieber

```
pass_attrs = mail=user,userPassword=password
```

angeben und im Benutzer-Eintrag außer den `mailAddress`-Attributen noch ein Attribut `mail` mit der »bevorzugten« Adresse hinterlegen. Diese Adresse dient dann zur Identifizierung des Postfachs.

Übungen



8.7 [!2] Konfigurieren Sie Dovecot und Ihren LDAP-Server wie in diesem Abschnitt beschrieben und vergewissern Sie sich, dass alles korrekt funktioniert.



8.8 [3] Experimentieren Sie mit verschiedenen Formen der Verschlüsselung für Kennwörter im Verzeichnis (Klartext, `CRYPT`, ...) und prüfen Sie, ob Dovecot damit zurecht kommt. Was ist die stärkste Verschlüsselung, die Sie zum Laufen bringen können?



8.9 [3] Ist es überhaupt sinnvoll, dass Benutzer für den Zugriff auf ihr Postfach und für den Zugriff auf das Verzeichnis dasselbe Kennwort verwenden? Wie müßte eine Konfiguration (von Dovecot und OpenLDAP) aussehen, wenn Postempfänger ein separates `mailPassword` haben? Welche Konsequenzen hat das für die Kennwortsicherheit mit Dovecot?



8.10 [4] (Für Tüftler.) Konfigurieren Sie Postfix, Dovecot und OpenLDAP so, dass Benutzer sich statt über Kennwörter über Kerberos authentisieren.

8.5 Extras

8.5.1 SMTP-AUTH

Eine wichtige Erweiterung des Mail-Servers ist SMTP-AUTH, also die Möglichkeit für lokale Benutzer, sich für den Versand von Nachrichten mit Benutzername und Kennwort zu authentisieren. Damit können Sie Mail zur Weiterleitung an beliebige Adressen nicht nur von Benutzern aus Ihrem lokalen Netz akzeptieren, sondern von irgendwo auf dem Internet (etwa von Außendienstmitarbeitern oder Home-Office-Nutzern).

Mit Postfix und Dovecot ist das zum Glück extrem einfach zu konfigurieren, da Postfix die nötigen Interaktionen und LDAP-Abfragen an Dovecot delegieren kann. Dazu müssen Sie in der Datei `dovecot.conf` das Client-Authentisierungs-Socket für Postfix freischalten:

```
auth default {
  <<<<<<
  socket listen {
    <<<<<<
    client {
      path = /var/spool/postfix/private/auth
      mode = 0660
      user = postfix
      group = postfix
    }
  }
}
```

In der Datei `main.cf` von Postfix fehlen dann bloß noch die Einstellungen

```
smtpd_sasl_auth_enable = yes
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
```

zur Weiterleitung von Authentisierungsanfragen an Dovecot sowie

```
smtpd_client_restrictions =
  permit_mynetworks, permit_sasl_authenticated,
  reject_unauth_destination
```

um das Einreichen authentisierter Mail zu ermöglichen.



In Ihrer Definition von `smtpd_client_restrictions` stehen vielleicht noch andere Sachen.



Je nachdem, welche SASL-Authentisierungsmechanismen Sie für SMTP-AUTH aktivieren, könnte es sein, dass Kennwörter im Klartext übertragen werden (etwa bei PLAIN oder LOGIN). In diesem Fall sollten Sie dafür sorgen, dass SMTP-AUTH nur über mit TLS verschlüsselte Verbindungen angeboten wird. Die Details hierzu würden hier zu weit führen; genauere Informationen über die Konfiguration von Postfix und Dovecot für TLS finden Sie zum Beispiel in der Linup-Front-Schulungsunterlage *Linux als Mailserver*.

8.5.2 Mail-Weiterleitung

Von unserer Wunschliste vom Kapitelanfang ist noch die Weiterleitung von Mail an andere Adressen übrig. Diese lässt sich in Postfix einfach über eine Tabelle in `virtual_alias_maps` realisieren. Eine Definition hierfür wäre – in Analogie zu der Gültigkeitsabfrage für Postfächer – zum Beispiel

```
# /etc/postfix/ldap-vam.cf
server_host = ldap://ldap.example.com/
search_base = dc=example,dc=com
scope = subtree
bind = no
query_filter = (&(objectClass=mailForwarder)(mailAddress=%s))
result_attribute = mailForwardAddress
```

*LDAP-Server**Ausgangspunkt der Suche**Suchbereich: Baum**Nicht an den Server binden*

zusammen mit

```
virtual_alias_maps = ldap:/etc/postfix/ldap-vam.cf
```

Beachten Sie, dass es sich hierbei tatsächlich um eine Tabelle handelt, die bedeutsame Rückgabewerte liefert (bei der Gültigkeitsabfrage für Postfächer kam es ja nur darauf an, ob überhaupt ein Eintrag existiert). Aus diesem Grund suchen wir im Verzeichnis nach einem Eintrag, der die Objektklasse `mailForwarder` mit der richtigen Eingangsadresse hat, und liefern den Wert von dessen Attribut `mailForwardAddress` zurück.



Wenn Sie Mail, die an eine `mailForwarder`-Adresse eingeht, an mehrere verschiedene Empfängeradressen weiterleiten wollen, dann können Sie das nicht machen, indem Sie im LDAP-Verzeichnis mehrere Werte für `mailForwardAddress` eintragen, da Postfix damit nicht zurecht kommt (auch wenn das LDAP-technisch natürlich die sauberste und naheliegendste Lösung wäre). Statt dessen müssen Sie, ganz wie in der `virtual`-Tabelle, die Adressen durch Komma getrennt in *einem* `mailForwardAddress`-Wert benennen:

```
mailForwardAddress: hugo@example.com,susi@example.net
```

Übungen



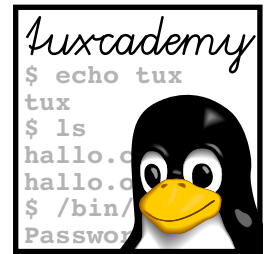
8.11 [!2] Implementieren und testen Sie Mailweiterleitung wie im vorstehenden Abschnitt beschrieben.

Kommandos in diesem Kapitel

<code>postconf</code>	Ändert Einstellungen in der Postfix-Konfigurationsdatei <code>main.cf</code>	<code>postconf(8)</code>	100
<code>postmap</code>	Hilfsprogramm für Postfix-Datenbankdateien	<code>postmap(1)</code>	101

Zusammenfassung

- Die Speicherung von Zustellinformationen in einem LDAP-Verzeichnis ermöglicht den Aufbau effizienter, skalierbarer und fehlertoleranter Mail-Infrastrukturen.
- Für die optimale Unterstützung von Mail-Funktionalität ist normalerweise eine spezielle Schemaerweiterung nötig, da es keinen Standard gibt.
- Postfix kann die Werte nahezu beliebiger Konfigurationsparameter einem LDAP-Verzeichnis entnehmen.
- Postfix muss sich nur um die Zuordnung von Domains zu Zustellverfahren und die Gültigkeitsprüfung von Empfängeradressen kümmern. Die eigentliche Zustellung von Nachrichten in die Postfächer der Empfänger übernimmt der LDA von Dovecot. Damit muss nur Dovecot wissen, wo die Postfächer überhaupt liegen, und Zugriffsrechte darauf haben.
- Postfix erlaubt die Delegation von SMTP-AUTH an die SASL-Implementierung von Dovecot.



A

Musterlösungen

Dieser Anhang enthält Musterlösungen für ausgewählte Aufgaben.

1.1 Direkt von `person` abgeleitet sind die Objektklassen `organizationalPerson` und `residentialPerson`. [RFC4519] enthält keine von `organizationalPerson` oder `residentialPerson` abgeleitete Objektklassen. Allerdings definiert [?] eine Objektklasse `inetOrgPerson`, die von `organizationalPerson` abgeleitet ist.

1.2 Notwendige Attribute für `buch` könnten zum Beispiel der Titel, der Verlag und das Erscheinungsjahr sein, vielleicht auch die ISBN; in Bibliotheken auch die Signatur. Als optionale Attribute kämen zum Beispiel Autor oder Herausgeber in Frage (manche Bücher haben nur das eine oder das andere), Standort oder Schlagwörter.

1.3 (Diese Antwort basiert auf der OpenLDAP-FAQ.) Man könnte mit einiger Berechtigung die Vermutung haben, dass eine gute relationale Datenbank einen LDAP-basierten Verzeichnisdienst schneller, besser, verlässlicher und die Daten auch von anderen Programmen nutzbar machen würde als eine »heimgekochte« Lösung. Immerhin sind diese Datenbanken hochoptimiert darauf, Daten schnell zu liefern und komplexe Suchoperationen abzuarbeiten. In Wirklichkeit ist das nicht so: Leider passen relationale Datenbanken und Verzeichnisdienste à la LDAP prinzipiell gar nicht gut zusammen, weil die Datenmodelle einfach zu verschieden sind.

Betrachten Sie zum Beispiel die Objektklasse `person`: Diese benötigt die Attributtypen `objectClass`, `sn`, `cn` und erlaubt ferner die Attributtypen `userPassword`, `telephoneNumber`, `seeAlso` und `description`. Da alle diese Attribute mehrere Werte haben können, verlangt der Grundsatz der Normalisierung, dass jeder dieser Attributtypen in eine eigene Tabelle getan wird. Als nächstes stellt sich die Frage nach Schlüsseln. Der Primärschlüssel könnte sich irgendwie aus dem DN ergeben, aber das ist bei den meisten Datenbankimplementierungen ziemlich ineffizient. Auf jeden Fall muss für einen Eintrag an diversen Plätzen auf der Platte nach Daten gesucht werden, was für viele Anwendungen die Performance arg verschlechtert. Sie können zwar vorgeschriebene Argumente, die nur einen Wert haben dürfen, in den Haupt-Tabelleneintrag schreiben (und optionale Argumente mit höchstens einem Wert, wenn Sie diesen Wert auf `NULL` setzen, falls das Argument nicht besetzt ist), aber alles andere muss in separate Tabellen.

Leider kommt es aber noch schlimmer: Einträge können *mehrere* Objektklassen haben, die in einer Vererbungshierarchie stehen. Ein Eintrag der Objektklasse `organizationalPerson` hat alle Attribute der Objektklasse `person` und noch ein paar. Heißt das, dass die verschiedenen Objektklassen in unterschiedlichen Tabellen

stehen sollen (eine Person hätte dann Daten teils in der person-Tabelle, teils in der organizationalPerson-Tabelle und so weiter), oder sollten Sie die person-Tabelle nicht ganz abschaffen und alles in die zweite Tabelle tun? Und was passiert mit einem »Filter« wie (cn=*) (der alle Einträge mit einem *common name* durchläßt)? Das cn-Attribut tritt in sehr vielen Objektklassen auf, deren Tabellen Sie dann alle einzeln durchsuchen müssten.

An diesem Punkt haben Sie im Grunde drei Möglichkeiten. Die erste besteht darin, *alles* zu normalisieren, so dass jedes Attribut in seiner eigenen Tabelle steht. Dies ist extrem verschwenderisch, wenn der DN der Primärschlüssel für einen Eintrag ist, so dass Sie zu einem Schema kommen, wo der DN in einen numerischen Index umgesetzt wird, der dann in den Attribut-Tabellen vorkommt. Das lässt sich in SQL noch verwalten, ist aber äußerst umständlich. Die zweite Möglichkeit besteht darin, jeden Eintrag als BLOB (*binary large object*) in eine Tabelle zu tun, die für alle Objektklassen gemeinsam verwaltet wird, und zusätzliche Tabellen zu haben, die vom LDAP-Server (nicht der relationalen Datenbank) als »Indextabellen« verwaltet werden. So eine Datenbank können Sie mit SQL nicht mehr durchsuchen, und rein zufällig ist das genau der Ansatz, der zum Beispiel vom BDB-Backend des OpenLDAP-Servers verfolgt wird – ganz ohne den überflüssigen Overhead einer relationalen Datenbank, wo etwas Einfaches und Schnelles wie Berkeley DB genau dieselbe Aufgabe erfüllt (und obendrein nichts kostet). Als dritte Möglichkeit können Sie das Verzeichnis-Datenmodell auch komplett aufgeben und LDAP als Zugangsprotokoll für Daten ansehen, die halt in einer relationalen Datenbank stehen und mit Verzeichnisdaten sonst nichts weiter zu tun haben. Dies geht meist mit Einschränkungen einher wie dass gewisse Attributtypen, die eigentlich mehrere Werte haben könnten, nur einen haben dürfen oder dass Sie einem existierenden Eintrag keine weitere Objektklasse zuordnen können. Hier haben Sie es strenggenommen nicht mehr mit einem »Verzeichnis« zu tun. Kommerzielle LDAP-Implementierungen auf Basis einer relationalen Datenbank verwenden entweder den ersten oder den dritten Ansatz. OpenLDAP unterstützt übrigens ein SQL-Backend, das unter bestimmten Umständen LDAP-Anfragen auf SQL abbilden kann; das ist nützlich, um einen LDAP-Zugang zu existierenden Daten zu schaffen, aber bildet keine sinnvolle Basis für einen ernstgemeinten Verzeichnisdienst.

1.5 Der erste DN hat die »Tiefe« 3 und der »Personenname« ist ein direkter Nachfolger des Organisationsnamens (er wird durch die Unterorganisationsbezeichnung nur eindeutig gemacht), während der zweite DN die »Tiefe« 4 hat und der »Personenname« Nachfolger des Unterorganisationsnamens ist.

1.7 Hier sind ein paar Vorschläge:

1. (|(c=de)(c=at))
2. (|(c:dn:=de)(c:dn:=at))
3. (cn=Fran\c3\a7ois Andr\c3\9 de la Meuni\c3\8re) (mit UTF-8-Codierung der französischen Sonderzeichen)
4. (fasel=*\2a*)

1.8 Dies ist ein Fall für die »modify RDN«-Operation. In früheren LDAP-Versionen konnten Sie damit nur den »linksten« RDN ändern, so dass der Eintrag an der alten Stelle entfernt und unter ou=Marketing,o=Beispiel GmbH,l=Musterstadt,c=DE neu angelegt werden musste – inzwischen ist das aber nicht mehr so.

1.9 Authentisierung von Clients beim Server ist eine notwendige Vorbedingung für Autorisierung, also das Festlegen von Zugriffsrechten für (identifizierte) Benutzer. Authentisierung von Servern beim Client hilft sicherstellen, dass die Verzeichnisdaten aus einer vertrauenswürdigen Quelle stammen und nicht von ei-

nem fremden Server, der so tut, als wäre er der, von dem Sie eigentlich die Daten haben möchten.

1.11 Beispielsweise: `ldap://ldap.beispiel.de/o=Beispiel%20GmbH,c=de?telephoneNumber?sub?cn=Hugo%20Schulz`

1.12 Eine »Beinahe«-Lösung ist schon in Bild 1.8 enthalten; dazu ist eigentlich gar nicht mehr viel hinzuzufügen.

3.3 Ganz ohne `-L` werden die Daten in einem (leicht) erweiterten LDIF-Format ausgegeben, zusammen mit erklärenden Kommentarzeilen. `-L` erzwingt die Ausgabe im LDIF-Format der Version 1, `-LL` lässt die Kommentare weg, und `-LLL` unterdrückt auch noch die Ausgabe der LDAP-Versionsnummer.

4.1 Etwas wie

```
access to attrs=favouriteDrink
  by self write
access to * by self read
```

4.2 Als Erweiterung zur vorigen Lösung:

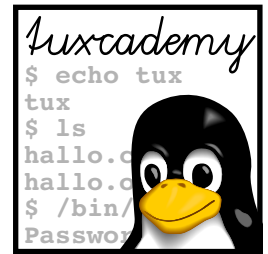
```
access to attrs=favouriteDrink
  by self write
  by dn=".*,dc=example,dc=com" read
```

8.1 Hier ein paar Denkanstöße:

- Wenn Sie eine webbasierte Oberfläche für die Administration von Mail-Benutzerkonten und ähnlichem anbieten wollen, dann können Sie davon profitieren, dass es jede Menge Frameworks für die Entwicklung von Web-Anwendungen gibt, die davon ausgehen, mit einer SQL-Datenbank im Hintergrund zu arbeiten, und dafür gute Unterstützung anbieten (Wir empfehlen Django, <http://www.djangoproject.com/>.) Der Zugriff auf ein LDAP-Verzeichnis ist im Vergleich dazu mühseliger.
- Je nachdem, wie Ihre Institution (Unternehmen, Hochschule, ...) ihre Personaldaten verwaltet, kann es einfacher sein, Postfix und Dovecot direkt an einen SQL-Server anzubinden, der zur Personalabteilung gehört oder seine Daten von dort bekommt, als Daten aus der SQL-Datenbank ins LDAP-Verzeichnis zu exportieren, jedenfalls wenn Sie das nicht sowieso schon machen müssen.

8.6 Sie müssen dafür sorgen, dass für Nachrichten mit mehreren lokalen Empfängern der Dovecot-LDA einmal pro Empfänger aufgerufen wird, nicht einmal pro Nachricht, da der dovecot-Transport eine Nachricht immer nur an einen Empfänger zustellen kann.

8.9 Ein Problem mit der Authentisierung von Dovecot gegen das `userPassword` besteht darin, dass die verschlüsselte Form des Kennworts sowohl für OpenLDAP (zur Authentisierung beim Verzeichnis im Allgemeinen) als auch für Dovecot (zum Öffnen des Postfachs) verständlich sein muss. Ein separates Kennwort nur für das Postfach würde OpenLDAP nur als Zeichenkette speichern, ohne ihm eine besondere Bedeutung beizumessen, so dass Sie in Dovecot ein geeignetes starkes Verschlüsselungsverfahren verwenden können, auch wenn OpenLDAP das nicht direkt versteht. Der Nachteil ist natürlich, dass dieses Kennwort separat von dem OpenLDAP-Kennwort administriert werden muss.



B

X.509-Crashkurs

Dieser Anhang gibt einen schnellen Überblick über X.509 und die Generierung von Zertifikaten. Wenn Sie genau wissen wollen, wie das alles funktioniert, lesen Sie die Linup-Front-Schulungsunterlagen *Apache und SSL (APW2)* oder *Linux als Web- und FTP-Server (WEBF)*.

B.1 Einleitung: Kryptografie, Zertifikate und X.509

Die verschlüsselte Übertragung von Daten ist grundsätzlich ein gelöstes Problem: Symmetrische Kryptoverfahren wie AES machen es möglich, große Datenmengen effizient und für alle praktischen Zwecke unknackbar zu übertragen. Das Problem bei diesen Verfahren ist lediglich der Schlüsselaustausch: Da beide Kommunikationspartner über denselben geheimen Schlüssel verfügen müssen, müssen sie sich irgendwie vertraulich auf einen Schlüssel einigen können.

symmetrische Kryptografie

Dieses Dilemma löst man über asymmetrische Kryptografie. Bei asymmetrischen Kryptoverfahren wie RSA hat jeder Teilnehmer zwei Schlüssel, einen privaten (geheimen) und einen öffentlichen. Der öffentliche Schlüssel kann allgemein bekannt gemacht werden, solange der private Schlüssel vertraulich bleibt. Dadurch wird es möglich, zwei verschiedene Probleme zu adressieren:

asymmetrische Kryptografie

Verschlüsselung Alice¹ möchte eine vertrauliche Nachricht an Bob schicken. Sie verschafft sich Bobs öffentlichen Schlüssel und verschlüsselt damit die Nachricht. Bob verwendet seinen privaten Schlüssel, um die Nachricht wieder lesbar zu machen.

Digitale Signatur Alice möchte kundtun, dass sie ein bestimmtes Dokument verfasst hat. Sie »signiert« (verschlüsselt) das Dokument mit ihrem privaten Schlüssel. Bob (oder wer auch immer sonst über Alices öffentlichen Schlüssel verfügt) kann mit Alices öffentlichem Schlüssel verifizieren, dass Alice (und nicht sonst jemand) das Dokument signiert hat und dass der Inhalt authentisch ist, also dem entspricht, was Alice geschrieben hat.

Ein praktisches Problem ist, dass man mit asymmetrischen Kryptoverfahren nur relativ kleine Datenmengen verschlüsseln kann und dass sie relativ ineffizient sind – symmetrische Kryptoverfahren beruhen im Wesentlichen auf Bitoperationen, die sich sogar in Hardware realisieren lassen, während asymmetrischen Kryptoverfahren aufwendige mathematische Operationen zugrunde liegen. (RSA zum Beispiel rechnet mit sehr langen Zahlen herum.) In der Praxis verwendet man

Probleme mit asymmetrischer Kryptografie

¹In der kryptografischen Literatur sind die beiden Kommunikationspartner traditionell immer »Alice« und »Bob«.

asymmetrische Kryptografie deshalb zusammen mit anderen kryptografischen Verfahren:

- Sitzungsschlüssel **Verschlüsselung** Alice wählt einen zufälligen Schlüssel (den »Sitzungsschlüssel« oder *session key*) für ein geeignetes symmetrisches Kryptoverfahren (etwa AES). Sie verschlüsselt diesen Schlüssel mit Bobs öffentlichem Schlüssel und schickt das Resultat an Bob. Bob entschlüsselt den AES-Schlüssel mit seinem privaten Schlüssel. Anschließend können Alice und Bob vertraulich kommunizieren, indem sie ihre Nachrichten mit AES verschlüsseln.
- Prüfsumme **Digitale Signatur** Alice bildet eine »kryptografische Prüfsumme« über das Dokument, also eine unumkehrbare Funktion, die von jedem Bit der Eingabe abhängt (Stichwort: MD5, SHA-1 und Ähnliches). Anschließend signiert sie diese Prüfsumme mit ihrem privaten Schlüssel. Zur Prüfung der Signatur bildet Bob die kryptografische Prüfsumme über das Dokument. Wenn die Entschlüsselung der Signatur mit Alices öffentlichem Schlüssel (den Bob ja hat) dasselbe Ergebnis liefert, ist die Signatur gültig, und das Dokument ist authentisch.

Auf diese Weise wird das Problem der Schlüsselverteilung gelöst, aber an seine Stelle tritt ein neues Problem: Angenommen, Bob findet irgendwo im Netz Alices öffentlichen Schlüssel (oder einen öffentlichen Schlüssel, der behauptet, zu Alice zu gehören). Wie kann Bob sicher sein, dass dieser Schlüssel *wirklich* Alices Schlüssel ist?

- Zertifizierungsstelle Eine mögliche Lösung für dieses Problem besteht darin, den Zusammenhang zwischen Alice und Alices Schlüssel von einer vertrauenswürdigen² Instanz bestätigen zu lassen. Diese Instanz heißt »Zertifizierungsstelle« (*certificate authority* oder kurz »CA«), und die Bestätigung des Zusammenhangs nennt man »Zertifikat«. X.509 ist ein Standard dafür, wie solche Zertifikate aussehen. Man spricht in diesen Zusammenhang auch von einer *public-key infrastructure* oder PKI.
- Zertifikat Ein Zertifikat besteht im Wesentlichen aus vier Komponenten:
- PKI

- Einem Namen für die Person (oder den Server), von dem die Rede ist. X.509 verwendet dafür sogenannte *distinguished names* der Form

cn=Hugo Schulz,o=Beispiel GmbH,l=Musterdorf,c=DE	Person
cn=www.example.com,o=Beispiel GmbH,l=Musterdorf,c=DE	Server

- Einem öffentlichen Schlüssel, der zu der benannten Person (oder dem benannten Server) gehört. Genau diesen Zusammenhang soll das Zertifikat bestätigen.
- Eine digitale Signatur für das Zertifikat. Diese sichert dessen Authentizität.
- Einen Namen (DN) für die Zertifizierungsstelle.

Sie können die Authentizität eines solchen Zertifikats prüfen, indem Sie sich den öffentlichen Schlüssel der Zertifizierungsstelle besorgen (der dazugehörige Name steht ja im Zertifikat) und damit die Signatur nachrechnen. Diesen öffentlichen Schlüssel bekommen Sie natürlich auch in Form eines X.509-Zertifikats.



Wenn Sie clever sind, dann fragen Sie sich an dieser Stelle, wer wohl das Zertifikat der Zertifizierungsstelle signiert haben mag, um zu dokumentieren, dass *dieses* echt ist. Die Antwort darauf lautet: Das macht die Zertifizierungsstelle selber. Bevor Ihnen das komisch vorkommt (was im ersten Moment absolut entschuldbar wäre), sollten Sie sich überlegen, dass Sie in diesem Geschäft *irgendwem* vertrauen müssen. Da Sie der Zertifizierungsstelle glauben, dass sie zum Beispiel Alices Zertifikat zuverlässig signieren

²»Vertrauenswürdig« heißt in einem Kontext wie diesem gemäß Peter Gutmann dasselbe wie »man verläßt sich drauf, weil man ohnehin keine andere Wahl hat.«

kann, können Sie ihr auch glauben, dass sie ihr eigenes Zertifikat zuverlässig signieren kann – das ist kein großer Schritt. Man spricht bei einem solchen von der Zertifizierungsstelle selbst signierten Zertifikat auch von einem »Wurzelzertifikat« (*root certificate*).

Wurzelzertifikat



Wobei natürlich die Frage bleibt, woher Sie wissen, dass das Wurzelzertifikat *wirklich* echt ist und Ihnen nicht – komplett mit gültiger Signatur – von einem geschickten Angreifer untergeschoben wurde. Sie können dieser Sache natürlich nachgehen, indem Sie sich bei der Zertifizierungsstelle vergewissern – oder (was wahrscheinlicher ist) Sie vertrauen blind dem Hersteller Ihres Browsers, der Ihnen hilfreicherweise (?) ein paar Dutzend Wurzelzertifikate von verschiedenen Zertifizierungsstellen fest eingebaut und (hoffentlich ...) vorher seine Hausaufgaben ordentlich gemacht hat.



Grundsätzlich gibt es auch die Möglichkeit, eine Hierarchie von Zertifizierungsstellen zu bilden. Das heißt, ein Zertifikat für eine Person oder einen Server kann mit einem Zertifikat signiert sein, das nicht direkt das Wurzelzertifikat einer Zertifizierungsstelle ist, sondern wiederum mit einem anderen Zertifikat signiert wurde. Diese »Kette« von Zertifikaten läßt sich weiterverfolgen, bis irgendwann ein selbstsigniertes Zertifikat erreicht ist.

Hierarchie

Um einen Server über X.509 authentisieren zu können, brauchen Sie zumindest ein Zertifikat für diesen Server. Wenn es sich dabei um einen Web-Server handelt, werden Sie kaum anders können, als sich dieses Zertifikat von einer kommerziellen Zertifizierungsstelle wie VeriSign ausstellen zu lassen, damit die gängigen Browser es ohne weitere Mühe verifizieren können.

Server-Zertifikat



CAcert (<http://www.cacert.org/>) ist eine Organisation, die auf nichtkommerzieller Basis (unter anderem) Zertifikate für Server ausstellt. CAcert arbeitet im Moment an einer Akkreditierung durch die wesentlichen Browser-Hersteller; dieser Prozess ist aktuell (Mai 2011) aber noch nicht abgeschlossen.

CAcert

Wenn Sie Ihre Zertifikate nur für »interne« Zwecke brauchen – etwa für eine VPN-Infrastruktur oder einen Web-Server im Intranet –, gibt es keinen vernünftigen Grund, VeriSign und Konsorten Geld in den Rachen zu werfen. Sie können ohne weiteres selbst als Zertifizierungsstelle für Ihre eigenen Zertifikate agieren und so nicht nur jede Menge Kosten sparen, sondern auch eine wesentlich sicherere Infrastruktur aufbauen (da Sie sich selbst deutlich mehr vertrauen dürften als den Windhunden vom kommerziellen Zertifizierungsmarkt).

Eigene Zertifizierungsstelle



Im Rest dieses Kapitels erklären wir, wie Sie X.509-Zertifikate auf der Basis von OpenSSL (<http://www.openssl.org/>) verwalten können. Es gibt alternative frei verfügbare Implementierungen von X.509 sowie bequemere Oberflächen zur Verwaltung von Zertifizierungsstellen, aber es handelt sich auch nicht um Hexenwerk.

B.2 Eine Zertifizierungsstelle generieren

Um zur Zertifizierungsstelle zu werden, müssen Sie ein Schlüsselpaar (privater und öffentlicher Schlüssel) für die Zertifizierungsstelle erzeugen und damit ein selbstsigniertes Wurzelzertifikat generieren. Im Idealfall tun Sie das aus Sicherheitsgründen auf einem Rechner, den Sie für nichts Anderes brauchen – ein Laptop-Computer, den Sie sicher wegschließen, wenn Sie nicht gerade ein Zertifikat ausstellen müssen, ist am besten.

Schlüsselpaar

Sicherheit



Da alles Nötige auf der Kommandozeile stattfinden kann, reicht als Rechner für die Zertifizierungsstelle irgendeine alte Möhre, die sonst keiner mehr benutzen möchte, dicke aus. Machen Sie für den Fall des Falles Sicherheitskopien, die Sie mindestens genauso sicher aufbewahren wie den Rechner.



Manche Linux-Distributionen – etwa der »SUSE Linux Enterprise Server« von Novell – bieten hilfreicherweise an, im Rahmen der Installation Ihres Web-Servers, LDAP-Servers, ... auch gleich eine Zertifizierungsstelle mitzugenerieren. Sowas lehnen Sie natürlich dankend ab.

Verwaltungsinformationen Außerdem müssen Sie als Zertifizierungsstelle einige Verwaltungsinformationen speichern, etwa eine Liste der Zertifikate, die Sie ausgestellt haben. Im einfachsten Fall legen Sie für die Zertifizierungsstelle einen eigenen Benutzer an, in dessen Heimatverzeichnis Sie die benötigte Dateistruktur etablieren:

```
# useradd -m ca
# /bin/su - ca
$ mkdir private certs
$ chmod 700 private certs
$ echo 01 >serial
$ touch index.txt
```

Die Datei `serial` enthält die »Seriennummer« des nächsten zu erzeugenden Zertifikats und in `index.txt` steht die Zertifikatsliste. Im Verzeichnis `private` legen wir den privaten Schlüssel der Zertifizierungsstelle ab, und in `certs` landen die von der Zertifizierungsstelle ausgestellten Zertifikate.

Für die spätere Arbeit ist es am günstigsten, eine Konfigurationsdatei anzulegen, in der die wichtigsten Parameter für die verschiedenen OpenSSL-Werkzeuge stehen. Dies vermeidet Verwirrung und Fehler durch Vergesslichkeit und macht die Vorgänge nachvollziehbar. Eine mögliche Konfigurationsdatei sehen Sie in Bild B.1.

Wurzelzertifikat erzeugen Wenn Sie die Konfigurationsdatei aus Bild B.1 in `/home/ca/openssl-ca.cnf` abgelegt haben, können Sie das Wurzelzertifikat für die Zertifizierungsstelle wie folgt erzeugen:

```
$ export OPENSSL_CONF=~/.openssl-ca.cnf
$ openssl req -x509 -days 1825 -newkey rsa:2048 -out ca-cert.pem
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to '/home/ca/private/ca-key.pem'
Enter PEM pass phrase:geheim
Verifying - Enter PEM pass phrase:geheim
-----
$ _
```

Die Passphrase, die Sie hier festlegen, müssen Sie jedesmal wieder eingeben, wenn Sie mit dieser Zertifizierungsstelle ein neues Zertifikat für einen Benutzer oder Server ausstellen wollen.



Es versteht sich von selbst, dass Sie im wirklichen Leben eine deutlich komplexere Passphrase verwenden wollen als in unserem Beispiel. Von der Sicherheit des privaten Schlüssels Ihres Wurzelzertifikats hängt die Sicherheit Ihrer kompletten Zertifikatsinfrastruktur ab!



Mit der `-days`-Option geben Sie an, wie lang das Wurzelzertifikat gilt. Da mit dem Ablauf dieses Zertifikats die von Ihrer Zertifizierungsstelle ausgestellten Zertifikate nicht mehr verifiziert werden können, sollten Sie diese Dauer mit Bedacht wählen. Eine zu kurze Gültigkeitsdauer zwingt Sie dazu, schon bald alle ausgestellten Zertifikate zu erneuern, während eine zu lange Gültigkeitsdauer im Extremfall bedeuten kann, dass Fortschritte in der Computertechnik es möglich machen, den öffentlichen Schlüssel der Zertifizierungsstelle zu brechen, und die Zertifizierungsstelle dadurch kompromittiert wird. Mit einem 2048-Bit-Schlüssel sollten Sie allerdings für die vorhersehbare Zukunft Ruhe haben.

```

[ca]
default_ca = CA

[CA]
dir                = /home/ca
certificate         = $dir/ca-cert.pem           CA-Zertifikat
private_key        = $dir/private/ca-key.pem     Privater Schlüssel
database          = $dir/index.txt             Zertifikatsliste
new_certs_dir     = $dir/certs                 Neue Zertifikate
serial            = $dir/serial                Seriennummer

default_crl_days  = 7                         Rhythmus für CRL-Veröffentlichung
default_days      = 730                       Gültigkeitsdauer für Zertifikate
default_md        = sha1                      Prüfsumme für Zertifikate

policy            = CA_policy
x509_extensions  = certificate_extensions

[CA_policy]
commonName        = supplied                  Namensregeln für die Zertifikate
emailAddress      = supplied                  Muss angegeben sein
organizationalUnitName = optional            Darf fehlen
organizationName  = match                    Muss mit CA-DN übereinstimmen
localityName      = match
countryName       = match

[certificate_extensions]
basicConstraints  = CA:false

[req]
default_bits     = 2048                       Regeln für CA-Zertifikat
default_keyfile  = /home/ca/private/ca-key.pem
default_md       = sha1
prompt           = no
distinguished_name = CA_dn
x509_extensions  = CA_extensions

[CA_dn]
commonName        = CA                       DN für die Zertifizierungsstelle
emailAddress      = ca@example.com
organizationName  = Beispiel GmbH
localityName      = Musterhausen
countryName       = DE

[CA_extensions]
basicConstraints  = CA:true

```

Bild B.1: Konfigurationsdatei für eine OpenSSL-basierte CA



Sie können das gerade erstellte Zertifikat mit dem Kommando

```
$ openssl x509 -in ca-cert.pem -text -noout
```

anschauen.

B.3 Server-Zertifikate generieren

Der Prozess für die Erzeugung von Server-Zertifikaten ist derselbe, egal ob Sie ein Zertifikat von einer kommerziellen Zertifizierungsstelle wünschen oder ob Sie das Zertifikat selbst ausstellen:

1. Sie (als Inhaber des Servers) erzeugen ein Schlüsselpaar und auf dieser Basis einen *Certificate Signing Request* (CSR) – eine Datei, die Ihren öffentlichen Schlüssel und einen DN für den Server enthält. (Den privaten Schlüssel aus dem Schlüsselpaar behalten Sie natürlich für sich.)
2. Die Zertifizierungsstelle überzeugt sich davon, dass Ihr CSR vernünftig aussieht, stellt das dazugehörige (signierte) Zertifikat aus und schickt es Ihnen zurück.
3. Sie installieren das Zertifikat auf Ihrem Server.

Wenn Sie Ihre eigene Zertifizierungsstelle sind, dann übernehmen Sie den zweiten Schritt natürlich selbst.

Schlüsselpaar und CSR erzeugen Wir müssen Ihnen also als erstes erklären, wie Sie ein Schlüsselpaar und einen CSR generieren. Das funktioniert wieder mit OpenSSL:

```
$ unset OPENSSL_CONF
$ cd /tmp
$ openssl req -newkey rsa:2048 -keyout server-key.pem -out server-csr.pem
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'server-key.pem'
Enter PEM pass phrase:abc123
Verifying - Enter PEM pass phrase:abc123
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:Musterhausen
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Beispiel GmbH
Organizational Unit Name (eg, section) []:↵
Common Name (eg, YOUR name) []:www.example.com
Email Address []:webmaster@example.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:↵
```



```
An optional company name []: 
$ _
```

Was Sie als Bestandteile des DN eingeben, ist wahlfrei, solange es zur »Policy« der Zertifizierungsstelle passt (siehe Bild B.1).



In unserem Beispiel kann der »state or province name« leer bleiben, weil die Policy darüber keine Aussagen macht. Ebenso darf der »organizational unit name« leer bleiben (oder einen beliebigen Wert haben). Der »organization name« muss mit dem im Wurzelzertifikat der Zertifizierungsstelle übereinstimmen. Der »common name« ist grundsätzlich beliebig, aber *muss* für ein Server-Zertifikat dem FQDN des betreffenden Servers entsprechen.



Beim Erzeugen des Schlüsselpaars besteht OpenSSL darauf, dass Sie eine Passphrase zur Verschlüsselung des privaten Schlüssels eingeben. Grundsätzlich ist das lobenswert, aber es gibt Situationen – etwa wenn der Schlüssel für einen Server gebraucht wird, der auch starten soll, ohne dass jemand an der Konsole steht und die Passphrase eingibt –, wo Sie vielleicht lieber einen privaten Schlüssel *ohne* Passphrase hätten. Für diesen Fall können Sie wie folgt eine unverschlüsselte Kopie Ihres privaten Schlüssels erzeugen:

```
$ openssl rsa -in server-key.pem -out server-key-np.pem
Enter pass phrase for server-key.pem: abc123
writing RSA key
```

Anschließend steht der unverschlüsselte private Schlüssel in server-key-np.pem. Behandeln Sie ihn mit Sorgfalt.



Wenn Sie direkt einen nicht verschlüsselten privaten Schlüssel erzeugen wollen, können Sie »openssl req« mit der Option `-nodes` aufrufen.

Zertifikat besorgen Den CSR schicken Sie entweder an eine Zertifizierungsstelle Ihres Vertrauens oder Sie stellen sich selbst ein Zertifikat aus. Für Letzteres müssen Sie im Heimatverzeichnis des Benutzers `ca` stehen, und die Umgebungsvariable `OPENSSL_CONF` muss auf die Konfigurationsdatei für die Zertifizierungsstelle zeigen:


```
$ cd Zurück ins Heimatverzeichnis
$ export OPENSSL_CONF=$HOME/openssl-ca.cnf
$ openssl ca -in /tmp/server-csr.pem
Using configuration from /home/ca/openssl-ca.cnf
Enter pass phrase for /home/ca/private/ca-key.pem: geheim
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName      :PRINTABLE:'DE'
localityName     :PRINTABLE:'Musterhausen'
organizationName :PRINTABLE:'Beispiel GmbH'
commonName      :PRINTABLE:'www.example.com'
emailAddress     :IA5STRING:'webmaster@example.com'
Certificate is to be certified until May  3 10:00:56 2013 GMT (730 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Certificate:
<<<<<<
Data Base Updated
$ _
```

Das neue Zertifikat landet dann im Verzeichnis certs. Die Zertifikate dort sind nach ihrer Seriennummer benannt (das gerade angelegte Zertifikat heißt certs/01.pem), so dass Sie gegebenenfalls in der Datei serial nachschauen müssen, was die letzte vergebene Nummer war. (Denken Sie daran, dass serial immer die Nummer des *nächsten* Zertifikats enthält. Wenn Sie schlau sind, schauen Sie statt dessen in die Datei serial.old.) (Denken Sie auch daran, dass die Seriennummern hexadezimal sind.)

Das neue Zertifikat (und möglicherweise den dazugehörigen privaten Schlüssel) sollten Sie anschließend auf den gewünschten Rechner kopieren. Wenn Sie – wie empfohlen – einen dedizierten Rechner für die Zertifizierungsstelle verwenden, dann involviert dieser Prozess sinnvollerweise einen USB-Stick, denn es ist besser, wenn der Rechner mit der Zertifizierungsstelle nicht ans Netz angeschlossen ist.


Client-Zertifikate **Client-Zertifikate** Zertifikate für Benutzer – sogenannte Client-Zertifikate – können Sie übrigens ganz analog erstellen. Dafür verwenden Sie einfach einen CSR, bei dem das »common name«-Feld keinen FQDN enthält, sondern den Namen der betreffenden Person.

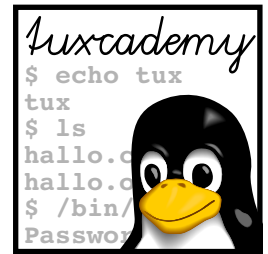
 Im Idealfall erzeugen nicht Sie diesen CSR (und das dazugehörige Schlüsselpaar), sondern der spätere Inhaber des Zertifikats. Nur auf diese Weise kann sicher gestellt werden, dass der private Schlüssel des Benutzers wirklich privat bleibt.

PKCS#12  Web-Browser erwarten Client-Zertifikate gerne im »PKCS#12«-Format, das das Zertifikat und den dazugehörigen privaten Schlüssel zusammenfasst. Wenn Sie sowohl das Zertifikat als auch den privaten Schlüssel haben, können Sie ein PKCS#12-Zertifikat mit dem Kommando

```
$ openssl pkcs12 -export -in hugo-cert.pem -inkey hugo-key.pem -out hugo.p12
Enter pass phrase for hugo-key.pem: x8o,AqS!k
Enter Export Password: foo-bar
Verifying - Enter Export Password: foo-bar
```

erzeugen.

 Das »Export Password« müssen Sie wieder angeben, wenn Sie das PKCS#12-Zertifikat mit dem Browser einlesen. Es muss (bzw. sollte) nichts mit dem Kennwort für den privaten Schlüssel zu tun haben.



Index

Dieser Index verweist auf die wichtigsten Stichwörter in der Schulungsunterlage. Besonders wichtige Stellen für die einzelnen Stichwörter sind durch **fette** Seitenzahlen gekennzeichnet. Sortiert wird nur nach den Buchstaben im Indexeintrag; „~/ .bashrc“ wird also unter „B“ eingeordnet.

- Active Directory, 2
- adduser
 - system (Option), 102
 - uid (Option), 102
- anonyme Sitzungen, 10
- Attribute, 3
- Attributwertbedingung, 11
- Authentisierungsbereiche, 53
- Authentisierungsoperationen, 10
- Authentisierungsphase, 86
- Autorisierungsphase, 86

- Backend, 33
- Backend-Datenbank, 2
- Basisobjekt, 11

- chmod, 48
- configure, 32
- controls, 13
- cosine.schema, 51

- Datenbanken, 33
- Definitionen, x
- Delta-Replikation, 71
- /dev/null, 34
- Directory Access Protocol, 2
- Distinguished Name, 7
- dovecot.conf, 106

- Eintrag, 3
 - /etc/default/slapd, 33
 - /etc/dovecot/dovecot-ldap.conf, 103
 - /etc/dovecot/dovecot.conf, 103
 - /etc/hosts, 61
 - /etc/ldap, 37
 - /etc/ldap.conf, 79, 83
 - /etc/ldap/schema/core.schema, 41
 - /etc/ldap/slapd.d/cn=config/olcDatabase={0}config.ldif, 40
 - /etc/libnss-ldap.conf, 79
 - /etc/nsswitch.conf, 78–79
 - /etc/openldap, 32, 37
 - /etc/openldap/ldap.conf, 79
 - /etc/openldap/slapd.d, 33
 - /etc/pam.d/login, 81
 - /etc/pam.d/password, 82
 - /etc/pam_ldap.conf, 80
 - /etc/passwd, 2, 34, 54, 76, 78–80
 - /etc/sasl2, 54
 - /etc/sasl2, 55
 - /etc/security/pam_unix2.conf, 82
 - /etc/shadow, 54, 76, 80
 - /etc/skel, 82
 - /etc/sysconfig/openldap, 33
- extended operations, 13
- extensibleObject, 6

- finger, 2
- Fortsetzungsverweis, 11

- Herausforderung, 54
- Howard, Luke, 78–80
- .htaccess, 87
- httpd.conf, 87
- hugo.ldif, 76

- Init-Skript, 32
- Internet Assigned Numbers Authority, 4

- Kerberos, 13
- Konsument, 66

- LDAP Data Interchange Format, 16
 - ldap.conf, 25–26, 57, 61–62, 70, 76, 79
 - ldapadd, 27, 29, 40, 42, 55, 76
 - W (Option), 27
 - w (Option), 27
 - x (Option), 27
 - ldapdelete, 28–29, 78
 - ldapmodify, 27–29, 40, 55, 78
 - a (Option), 27
 - M (Option), 44
 - ldapmodrdn, 28–29

- r (Option), 28
- s (Option), 28
- ldappasswd, 55, 57, 60, 76
 - S (Option), 56
 - X (Option), 60
- .ldaprc, 62
- ldapsearch, 24–27, 29, 40, 42–43, 56, 62, 77
 - b (Option), 25
 - C (Option), 43
 - D (Option), 25
 - H (Option), 26, 62
 - h (Option), 26
 - L (Option), 25, 42, 111
 - LL (Option), 42, 111
 - LLL (Option), 42, 111
 - p (Option), 26
 - s (Option), 26
 - U (Option), 56, 58
 - W (Option), 25
 - x (Option), 25
 - Z (Option), 62
 - ZZ (Option), 62
- ldapvi, 28
- ldapwhoami, 58
- login, 80, 83
- ls, 105

- main.cf, 97, 100, 102, 106
- make, 32
- master.cf, 102
- Mechanismen, 53
- Mechanismen (SASL), 14
- mod_authnz_ldap, 91

- NDS, 2
- netcat, 104
- numerischer Objektbezeichner, 4

- Objektklasse, 5
- ODBC, 34
- OID, 4
- OpenSSL, 60
- openssl, 119
 - days (Option), 116
- openssl req
 - nodes (Option), 119
- OPENSSL_CONF (Umgebungsvariable), 119
- Overlays, 34

- pam_ldap.so, 82
- pam_mkhomedir.so, 82
- pam_unix.so, 82
- pam_unix2.so, 82
- passwd, 57, 82–83
- postconf, 100
- postmap, 96, 101
 - v (Option), 101
- Produzent, 66

- Profile (SASL), 14

- Replikation, 66
- Require, 88

- saslpasswd, 54
- Schema, 5
- Secure Socket Layer, 14
- slapadd, 42–43
- slapcat, 42, 69
- slapd, 32–36, 40–44, 50, 55, 61, 66–67, 83
 - f (Option), 34
- slapd.conf, 32, 34, 37, 40–41, 43, 49, 55–57, 59–62, 67
- slapd.d, 32, 69
- slapindex, 39
- slappasswd, 37, 40
- slaptest, 41
- slurpd, 66
- /srv/mail, 105
- sshd, 80
- Suchfilter, 12
- Suchoperationen, 10
- swaks, 105
- Syntax, 3
- syslogd, 35

- Transport Layer Security, 14
- Typ, 3

- Umgebungsvariable
 - OPENSSL_CONF, 119
- Uniform Resource Locators, 15
 - /usr/lib, 55
 - /usr/lib/sasl2, 55
 - /usr/local/etc/openldap, 32

 - /var/lib/ldap, 42
 - /var/log/mail.log, 105
- Verweis, 11
- Verzeichnisbaum, 2
- Verzeichnisdienst, 2

- Werte, 3

- X.509-Zertifikate, 60

- Zertifizierungsstelle, 60