Seminar Betriebssysteme

(Institut für Informationsverarbeitung und Mikroprozessortechnik
SS 2004, LVA-Nr.: 353.062)

„Firmenweite konsolidierte Account-Datenhaltung mittels LDAP“

Autoren:

Klaus Pesendorfer * 0156150

Florian Wörter * 0156583

LVA-Leiter: Prof. Jörg R. Mühlbacher, DI Andreas Putzinger
Ort: Johannes Kepler Universität Linz
Datum: 4.6.2004

Kurzfassung:

Zusammenarbeit von Firmen und Personen über Netzwerke wird in der heutigen Zeit immer wichtiger. Verteilte Applikationen interagieren mit Computern im selben Netzwerk (LAN), in einem größeren Intranet oder über das weltweite Internet. Dabei müssen sehr viele Daten und Netzwerkressourcen den verschiedensten Applikationen sowohl zugänglich gemacht, als auch vor nicht autorisierten Programmen oder Benutzern geschützt werden. Und andererseits müssen in einem großen Netzwerk auch eine Menge von Diensten angeboten und konfiguriert (einheitlich oder speziell) werden. Das ist der Punkt, wo LDAP (Lightweight Directory Access Protocol) ins Spiel kommt. Im großen und ganzen dient LDAP der Netzwerkpflege und Administration.
Wir werden im folgenden Dokument (Teil I.) erklären was LDAP ist und verschiedene Varianten von LDAP vorstellen. Dieser Abschnitt wird erklären warum man LDAP überhaupt benötigt und was seine Vorteile in einem großen Unternehmen sind. Im 2.Abschnitt beschreiben wir, wie man ein LDAP-System (d.h. Server + Client) von Grund auf in einer Linux-Umgebung aufsetzt und konfiguriert. Dabei werden wir ein Hauptaugenmerk auf die Einrichtung Internet-basierter Dienste (HTTP, FTP, SSH, etc.), die Benutzerauthentifizierung und eines Adressverzeichnisses werfen.

Abstract:

Networks are getting more and more important in today's business. But such big and interconnected networks also need to be managed and configured. Therefore a central point of administration is essential. LDAP (Lightweight Directory Access Protocol) is one way to provide and use (manual up to full-automated) information either for configuring network-services or otherwise store all other kinds of information as a central information storage.
The first part of our document describes the theoretical basics of directory services, especially of LDAP. Also the most important implementations of the LDAP-Standard are mentioned. In the second part we will show you, how to set up a LDAP-System (based on the open source implementation OpenLDAP). Our server provides internet-based services (HTTP, FTP, SSH, etc.) and the login-process authorized through the LDAP-database and besides a central addressbook is stored on the server, which can be accessed by a client.

Inhalt

Teil I – Theoretische Grundlagen

Warum braucht man überhaupt eine zentrale „Datenhaltung“?
Computernetzwerke spielen heutzutage eine so wichtige Rolle im täglichen Leben, dass sie nicht mehr wegzudenken sind, und auch vieles nicht mehr funktionieren würde, wenn sie einmal zusammenbrechen. Aber diese komplexe Netzwerkstruktur muss auch einmal oder besser gesagt ständig konfiguriert und gewartet werden. Und um diese Konfigurations- und Wartungsarbeiten zu vereinfachen und auch gewisse Informationen konsistent und zentral bereitzustellen, benötigt man eine zentrale „Datenhalten“ – im späteren als *Verzeichnisdienst* bezeichnet.

1 Was ist ein Verzeichnisdienst?

Ein **Verzeichnis** kann man ganz allgemein formulieren: „*A directory is a listing of information about objects arranged in some order that gives details about each object.*“ [Johner98].
Also kurz gesagt, eine Liste von Objekten, mit Beschreibungen zu jedem Objekt.
Beispiele hierfür wären u.a. ein Telefonbuch mit den Namen der Personen als Objekten und den Telefonnummern und Adressen als Detailinformationen dazu, oder ein Buchkatalog mit Autor, Titel und ISBN-Nummer.

Ein **Verzeichnisdienst** wird nun schon in einem speziellen Kontext gesehen. Er besteht aus einer im Netzwerk verteilten Datenbank, die auf dem Client/Server Prinzip basiert. Diese Datenbank beinhaltet nun nahezu beliebige Daten nach dem oben bereits erwähnten Konzept eines Verzeichnisses. Die gespeicherten Informationen lassen sich dann für viele verschiedene Dinge nutzen. Häufig zur Konfiguration und Administration von Anwendungen und Diensten, aber oft auch nur als Informationsquelle oder Nachschlagwerk.

Die Daten sollen folgenden Kriterien entsprechen:

Flexibilität – keine fixe Datenstruktur, diese ist durch den Anwender definierbar

Dynamik – Bearbeitung der Daten soll im laufenden Betrieb möglich sein

Sicherheit – durch Definition von Sicherheitsregeln und Zugriffsbeschränkungen

Ein Verzeichnisdienst ist nicht gleichzusetzen mit einer herkömmlichen Datenbank, denn es ist mehr. Ein Hauptkriterium eines Verzeichnisdienstes ist die Tatsache, dass darauf wesentlich öfter lesend/suchend zugegriffen wird als schreibend, deshalb ist es speziell auf Lese- und Suchoperationen optimiert. Eine herkömmliche Datenbank ist transaktionsorientert und wird meistens mit der Structured Query Language (SQL) bearbeitet. Für Verzeichnisdienste ist dieser größere Verwaltungs- und Bearbeitungsaufwand nicht sinnvoll, aber auch nicht notwendig, denn es ist nicht so schlimm, wenn für kurze Zeit (während Änderungen) einige Daten inkonsistent sind. Zum Beispiel bedeutet es nicht den Weltuntergang, wenn ein Drucker an einen anderen Standort gestellt wird und im Verzeichnisdienst kurzzeitig beide Standorte oder ein falscher steht. LDAP verwendet ein einfaches und optimiertes Zugriffsprotokoll anstatt SQL, um einfacher in Applikationen integriert werden zu können.
[vgl. Johner98 und Banning01]

1.1 Aufbau eines Verzeichnisdienstes:

Verzeichnisdienste (*Directory Services* = DS) basieren generell auf einer Datenbank und sind meist nach dem *X.500-Standard* aufgebaut. Die gewünschten Daten werden zur sinnvollen und effizienten Verwaltung und Nutzung in einer baumartigen Struktur abgelegt. Dieser Verzeichnis-Baum (*Directory T*ree) besteht immer aus einer Wurzel (generell mit „Root“ bezeichnet) und darunter eingehängte Einträge mit gewissen Eigenschaften (welche oft noch weiter gegliedert und strukturiert sind). Diese Datenbank muss nicht zwingend auf einem zentralen Server liegen, sie kann auch auf mehrere Rechner verteilt werden (*Distributed Directory*). Weiters kann sie auch repliziert werden, um die Ausfallssicherheit und Leistung (Performance) zu erhöhen. Auf diese letzten Punkte wird aber später noch genauer eingegangen.

Die Funktionen, die ein Verzeichnisdienst bereitstellt, kann man wie folgt grob einteilen:

Speichern von Informationen:

- Daten strukturiert speichern (Directory Tree)
- Daten schnell allen Benutzern zur Verfügung zu stellen

Verwenden von Informationen: neben dem einfachen lesenden Zugriff auf die Daten, ist es meist noch viel wichtiger, sie direkt und automatisiert zu verwenden. In

diese Kategorie fallen z.B. Dienste wie die zentrale Benutzerauthentifikation oder eine einheitliche Konfiguration des Network File Systems (NFS).

Anwendungen eines Verzeichnisdienstes:

Grafische Anwendungen: sind oftmals notwendig um auch normalen PC-Anwendern ein einfaches Interface zur Anwendung des Verzeichnisdienstes zur Verfügung zu stellen (z.B. KLdap oder GQ unter Linux)

Kommandos: werden häufig für administrative Zwecke eingesetzt, da sie sich über Skripts leicht automatisieren lassen.

Funktionen: werden benötigt und eingesetzt um eigene Programme mittels Directory Services anzureichern.

Routinen: viele Hersteller stellen schon fertig implementierte Komponenten, Frameworks oder sogar ganze Programmpaketen zur Nutzung von Verzeichnisdiensten bereit.

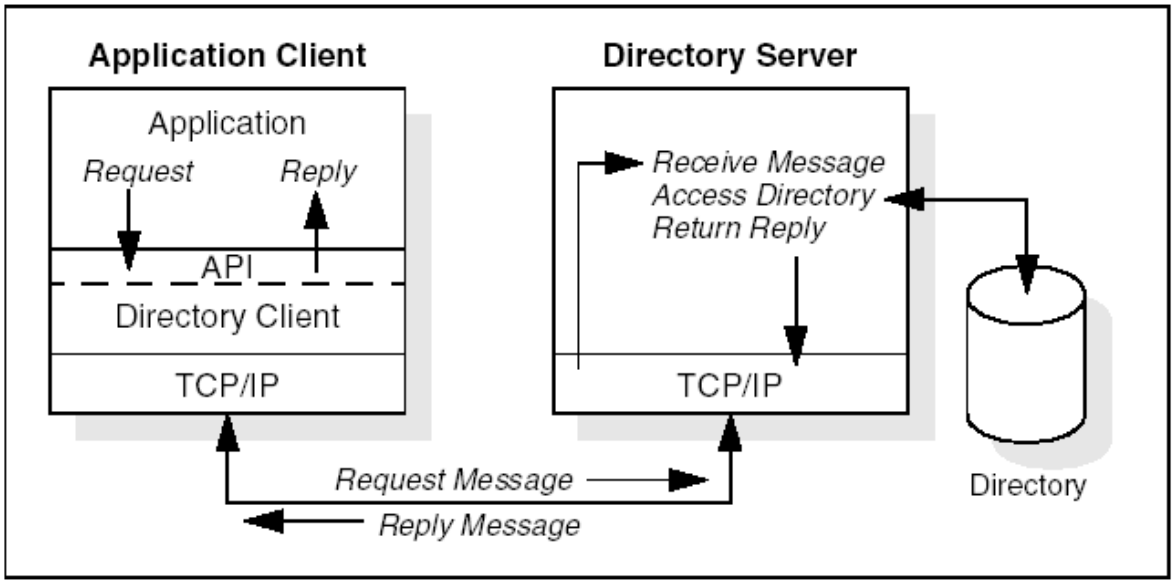


Abb.1.1: Kommunikation im Verzeichnisdienst [Johner98 – Fig.1].

Die Kommunikation zwischen Client und Server funktioniert meist auf Basis des Standard-Netzwerkprotokolls TCP/IP. Der Datenfluss basiert demnach auf Anfrage- und Antwort-Nachrichten im Sinne von TCP/IP. Es ist zu beachten, dass die Datenbank (das Directory) nicht unbedingt am gleichen Rechner laufen muss wie das Directory Service (der Server). Abbildung 1 verdeutlicht dieses Prinzip ganz gut.

2

2 LDAP und X.500

[vgl. Banning01, Johner98]

2.1 Geschichte und Standards:

Im Jahre 1988 standardisierte die CCITT (Consultative Committee on International Telephony and Telegraphy) den *X.500 Standard*, welcher in ISO 9594 zum *Data Communications Network Directory* zertifiziert wurde.

X.500 dient der Organisation von großen Datenmengen (im Sinne von Verzeichniseinträgen) in einem hierarchischen Namensraum. Der Standard definiert auch effiziente Such-Funktionen um den lesenden Zugriff möglichst performant zu gestalten. Die Kommunikation zwischen Verzeichnis-Client und Verzeichnis-Server ist durch das *Directory Access Protocol (DAP)* standardisiert.

DAP war aber zu mächtig und ressourcenintensiv, denn es benötigte den gesamten OSI-Protokollstapel um zu funktionieren, somit entschloss man sich ein kleineres und einfacheres Zugriffsprotokoll zu entwickeln. Somit war *LDAP (Lightweight Directory Access Protokoll)* im entstehen, welches am einfacheren TCP/IP-Protokollstapel aufsetzt und somit weitgehend überall eingesetzt werden kann (wo es TCP/IP gibt, und dies ist heutzutage fast überall) und auch wesentlich einfacher zu implementieren ist.

Bis zur endgültigen Version von LDAP entstanden mehrere Entwürfe und Vorschläge. Anfänglich tauchten die zwei RFCs 1202 (Directory Assistance Service) und 1249 (DIXIE Protocol Specification) als Ideenskizzen auf. Danach kam der 1.LDAP-Vorschlag in RFC 1487 als X.500 Lighweight Access Protocol, was im folgenden dann zur LDAP-Version 1 verfeinert und teilweise noch vereinfacht wurde und in RFC 1777 sozusagen standardisiert wurde.

Diese LDAP-Grundversion wurde im Laufe der Jahre durch mehr Funktionalität erweitert und verbessert, was sich in den Versionsnummern v2 und v3 (RFC 2251) niederschlägt. Diese 2 Versionen sind auch heute noch gängige Standards (obwohl RFC 3494 vorschlägt Version 2 durch neuere Versionen zu ersetzen).

LDAP v3 unterstützt nun schon folgende Funktionen:

Weiterleitung: ein Server der die angeforderten Daten nicht besitzt, kann den Client an einen anderen Server weiterleiten

Sicherheit: Simple Authentication and Security Layer (SASL)

Internationalisierung: UTF-8 Unterstützung

Erweiterbarkeit: neue Objekttypen und Operationen können dynamisch definiert und per Schema-Definition eingesetzt werden.

Noch kurz einige Erklärungen zu X.500:

X.500 gilt als der OSI-Standard für weltweit verteilte Verzeichnisdienste.

X.500 hilft Ihnen Adressen von Personen und Organisationen zu finden

X.500 hilft nützliche Ressourcen im Netz zu finden

X.500 dient Anwendungen als globaler Namensdienst

2.2 Grundeigenschaften von LDAP:

basiert auf dem TCP/IP-Protokoll: LDAP funktioniert somit in fast jeder Netzwerkumgebung und nutzt die bestehenden Verbindungen und Grundstrukturen

leicht zu implementieren

funktionell und schnell: wegen dem einfachen Protokollaufbau funktioniert die Kommunikation und der Datenzugriff wesentlich schneller als bei der alten Implementierung DAP.

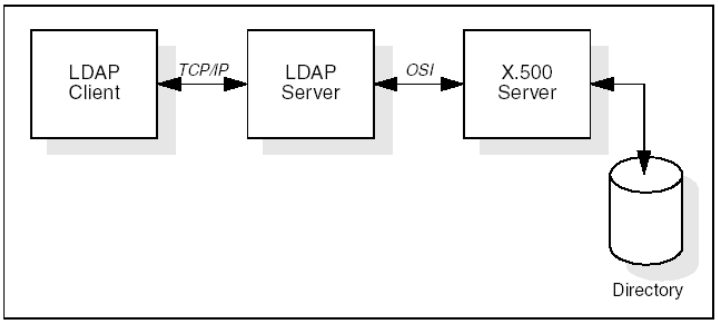


Abb.2.1: LDAP-Server als Gateway zu einem X.500-Server [Johner98 – Fig.2]

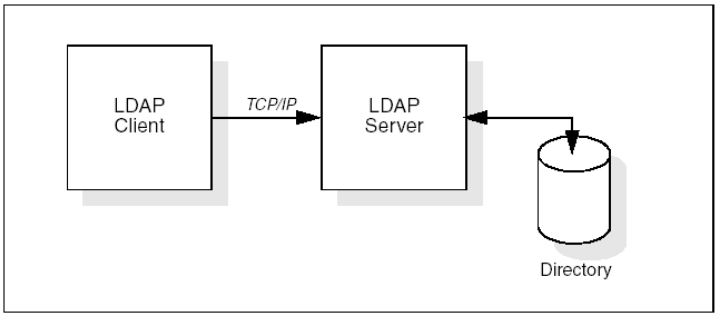
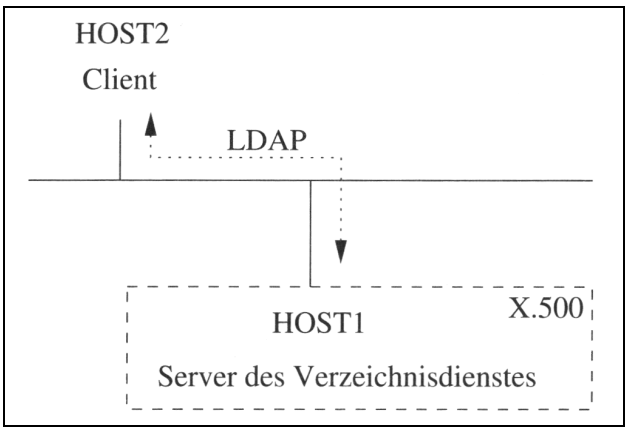


Abb.2.2: Stand-alone LDAP-Server [Johner98 – Fig.3]

LDAP arbeitet nach dem klassischen Client/Server-Modell (vgl. TCP/IP), nämlich nachrichtenorientiert. Der Client sendet eine Anfrage an den LDAP-Server, dieser stellt seinerseits eine Anfrage an die X.500-Datenbank (entweder lokal oder auf einem anderen Server – vgl. Abb.2 u. 3) und liefert die erhaltenen Daten wieder zurück an den Client, wobei die gesamte Kommunikation dem LDAP-Protokoll unterliegt.



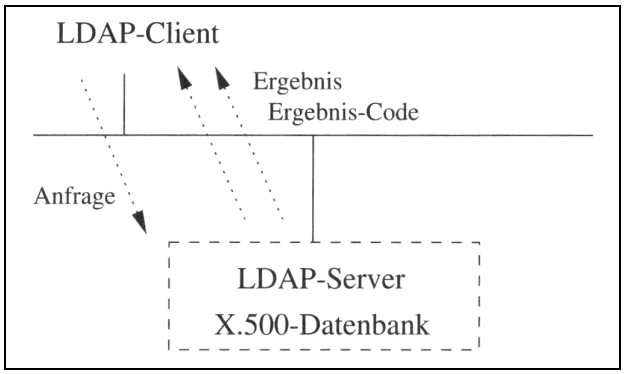


Abb.2.3 u. 2.4: Kommunikation zwischen LDAP-Client und -Server [Banning01 – Abb.3.1 u. 3.2]

Die Nachrichten die gesendet werden, werden Mitteilungen (Messages) genannt, wobei der Client seine Anfrage-Daten in eine solche Message packt und verschickt und dann auf die Antwort des Servers wartet. Er kann in der Zwischenzeit auch noch weitere Anfragen senden, und bekommt dann zu jeder dieser Anfragen eine eigene Antwort zurück. Der Server liefert zu jeder Anfrage eine Ergebnis-Message (bzw. mehrere Teil-Messages) und einen Ergebnis-Code.

2.3 Aufbau und Eigenschaften

LDAP-Aktionen lassen sich in 3 Kategorien einteilen:

Beglaubung: ist eine der wichtigsten Zugriffsarten, wobei der LDAP-Client eine Authentifizierungs-Mitteilung (meist bestehend aus Benutzername und Passwort) an

den Server schickt, wobei dieser dann den Client, für die in der Datenbank definierten, Berechtigungen authentifiziert oder ihn zurückweist.

Lesender Zugriff: für diese Form stehen meist 2 Aktionen zur Auswahl:

- *suchen* in der Datenbank
- *vergleichen* von Inhalten der Datenbank

Schreibender Zugriff: für Anwender mit entsprechenden Rechten stehen folgende Arten des schreibenden Zugriffes zur Verfügung:

- *hinzufügen*
- *löschen*
- *ändern*
- *umbenennen*

... von Einträgen.

Die Verbindungseigenschaften des LDAP-Protokolls lassen sich wie folgt beschreiben: die Kommunikation erfolgt über das Transmission Control Protokoll (TCP) und bietet somit folgende Eigenschaften:

bidirektionale Verbindung zwischen Client und Server

verbindungsorientierte Datenübertragung

Integritätsprüfung über Checksummen

3-way-Handshake Verbindungsaufbau

Datenzuordnung über TCP-Ports, d.h. zu einem eindeutigen Dienst

Der LDAP-Server wartet auf Anfragen generell am **TCP-Port 389**.

2.4 LDAP-Server

2.4.1 Datenhaltung

Zur Planung der Datenbank bedarf es zuerst einmal einiger Überlegungen über die Art und die Menge der zu speichernden Daten.

Was soll gespeichert werden?

Um diese Frage zu beantworten, sieht man sich am besten an, was für Funktionen man benötigt und überlegt sich dann, welche Daten man minimal dazu braucht. Auch in Hinblick

auf die Datenpflege, muss man sich gut überlegen, wie man welche Daten am besten up-to-date hält.

Welches Format haben die Daten?

Für jedes Attribut muss man sich einen Datentyp und die Menge des reservierten Speichers dafür überlegen, wobei letzteres besonders schwierig ist. Zum Beispiel wie entscheidet man die Maximallänge eines Namens und was macht man wenn ein Name dann doch länger ist?

Wen man alle diese Entscheidungen getroffen hat, kann man abschätzen, wie viel Speicherplatz man für die Datenbank vorbereiten sollte.

$$Gesamtbedarf = Anz\_Obj * Anz\_Attr * Feldgröße$$

... Anz_Obj = Anzahl der Objekte in der gesamten DB

... Anz_Attr = Anzahl der Attribute je Objekt

... Feldgröße = Größe der Attribute (bei unterschiedlichen Größen über Summe getrennt berechnen)

2.4.2 Das Schema

Der Begriff Schema bezeichnet die Struktur einer Datenbank eines Verzeichnisdienstes.

Das Schema lässt sich in folgende 3 Teile gliedern:

Objekte: ... sind Einträge im Verzeichnisbaum

Attribute: ... sind die Eigenschaften der Objekte

Regeln: ... sie legen fest, welche Objekte wo und mit welchen Attributen vorkommen dürfen.

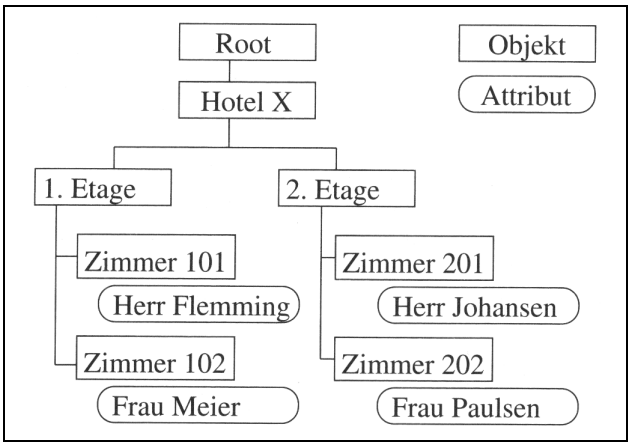


Abb. 2.5: Beispiel für einen Verzeichnisbaum [Banning01 – Abb.3.5]

Objekte können in Objektklassen eingeteilt werden, welche folgende Eigenschaften aufweisen:

Eindeutiger Name: Objektklassen werden durch einen, im ganzen Verzeichnisbaum, eindeutigen (nicht case-sensitive) Namen repräsentiert. Somit kann eine Objektinstanz vom Typ der durch den Objektnamen repräsentierten Klasse sein.

Eindeutige Identifikation: ... ist das tatsächliche Identifikationsfeld, welches als numerische Kennung (Object Identifier = OID) eine Objektklasse eindeutig identifiziert. Der oben erwähnte Name dient nur der einfacheren Verwendung.

Art des Objekts

Notwendige Attribute: diese Attribute muss jede Instanz dieser Klasse aufweisen.

Mögliche Attribute: zusätzliche Attribute, die auch vorkommen können, aber nicht zwingend müssen.

Objekte unterscheidet man grundsätzlich in *Containerobjekte* und *Blattobjekte*, wobei letzteres die Blätter des Baumes bezeichnen und Containerobjekte, wie schon der Name sagt, weitere Objekte beinhalten können, also mitten im Baum stehen.

Zur hierarchischen Gliederung des Baumes teilt man die Objekte in Klassen ein.
Für Container stehen folgende 4 Klassen zur Verfügung:

Root: die einzigartige Wurzel

Country (c)

Organization (o)

Organizational Unit (ou)

Blatt-Objekte sind generell vom Typ:

Common Name (cn)

Attribute dienen der Beschreibung der Objekte und folgen auch einer vordefinierten Struktur:

Eindeutiger Name

Eindeutige Identifikation

Syntax: … definiert die möglichen Werte eines Attributs (z.B. : max. Länge, nur Ziffern, nur Zeichen, case-sensitv, ...)

Werte des Attributs: ein Attribut kann einen oder mehrere (Liste) Werte besitzen

Attribute können in mehreren Hierarchiestufen im Verzeichnisbaum auftauchen, d.h. ein Attribut ist einem anderen untergeordnet, was die Ordnung und die Suche erleichtert. Z.B.: könnte das Attribut Name die Unterattribute Vorname und Nachname besitzen.

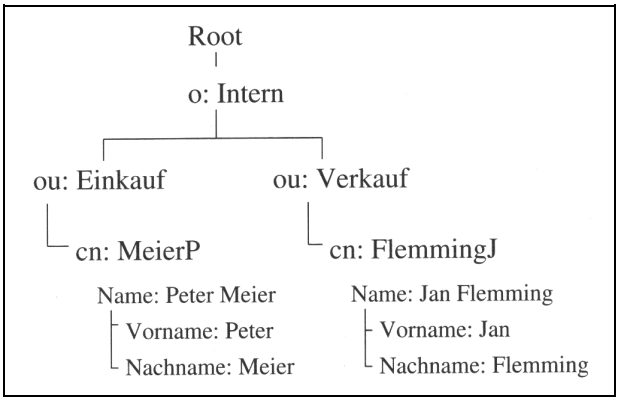


Abb. 2.6: Ein attributierter Verzeichnisbaum [Banning01 – Abb.3.8]

Regeln:

Es gibt einige Grundregeln, die in jedem Verzeichnisbaum beachtet werden sollten:

... übernommen aus [Banning01 – S.23]

1. Als oberstes Element eines Verzeichnisdienstes existiert das Objekt Root. Es ist grundsätzlich vorhanden und kann weder verschoben noch verändert werden.
2. Unterhalb von Root ist es möglich, entweder ein Country-Objekt (c) oder eine Organisation (o) anzulegen.
3. Das Country-Objekt muss nicht existieren
4. Das Country-Objekt darf nur einmal existieren
5. Die Organisation muss direkt unter dem Country-Objekt stehen. Fehlt das Land, so muss die Organisation direkt unter Root stehen.
6. Es dürfen mehrere Organisationen auf gleicher Ebene existieren.
7. Unterhalb des Objekts o dürfen entweder Blattobjekte (cn) oder organisatorische Einheiten (ou) existieren.
8. Den organisatorischen Einheiten können nur weitere Objekte vom Typ ou oder Blattobjekte folgen.
9. Es dürfen mehrere Objekte der Klasse ou auf gleicher Ebene existieren.
10. Blattobjekte (cn) dürfen nur unter o und ou vorkommen.

2.4.3 Der Namensraum

Eindeutigkeit von Namen:

Wie oben bereits erwähnt wurde, müssen Objekte eindeutig bezeichnet werden. Dafür gibt es zwei Möglichkeiten: eine absolute und eine relative Benennung ...

Distinguished Names (dn)

Relative Distinguished Names (rdn)

Der *Relative Distinguished Name* ist kontextabhängig, d.h. er bezieht sich auf den übergeordneten Container bzw. den gesamten Pfad bis zur Wurzel.

Der *Distinguished Name* enthält all diese Informationen für jedes Objekt explizit, d.h. er setzt sich zusammen aus dem zu bezeichnenden Objekt selbst und den darüberliegenden Objekten in Richtung Baumwurzel.

Dieser Weg, von der Wurzel bis zu einem gewissen Punkt, kann als Kontext (context) bezeichnet und auch so verwendet werden. Man kann zum Beispiel in einer Filiale den Kontext genau auf den Teilbaum dieser Filiale setzen und vereinfacht somit die Benennung über die relative Benennung von diesem Punkt aus, da der Pfad bis zur Wurzel dann implizit angenommen wird.

Distinguished Name = Context + Relative Distinguished Name

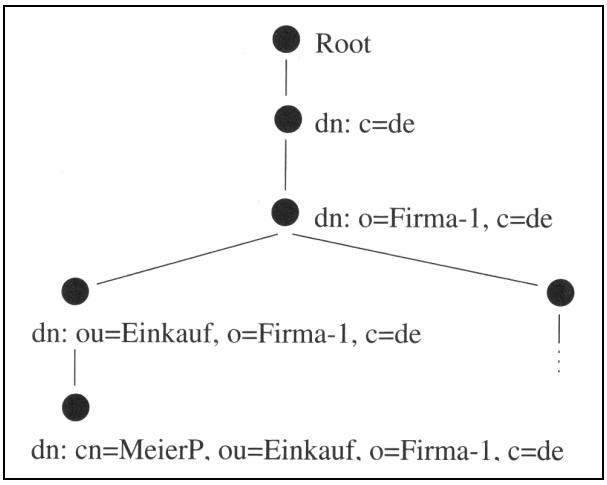


Abb. 2.7: Distinguished Names im Verzeichnisbaum [Banning01 – Abb.3.12]

Namensgebung & Strukturierung:

Als *Namensraum* bezeichnet man die Aufteilung des Verzeichnisbaums in Container- und Blattobjekte und deren Benennung. Ein Verzeichnisbaum sollte von Anfang an gut durchdacht und geplant werden, er muss kreisfrei sein, also eine echte Baumstruktur aufweisen und für die Objekte sollten logische, einfache und funktionelle Begriffe gewählt werden.

Die gängigste Strukturierungsform ist die Anlehnung an die Organisationsstruktur der Firma, aber je nach Anwendung sind auch andere Arten denkbar und oft sinnvoller:

Organisation

Objektarten (eher ungebräuchlich)

Zugriffskontrolle

Anwendungen

Partitionierung

Replizierung

Meist sollte man die anderen Strukturierungseigenschaften zumindest berücksichtigen und die Planung je nach Priorität danach ausrichten. [vgl. Banning01]

2.4.4 Replizierung & Partitionierung

Die *Replizierung* bedeutet, dass die Datenbank auf beliebig vielen Rechnern existieren kann, womit gemeint ist, dass die gleichen Daten mehrfach, also redundant vorhanden sind. Diese Methode wird aus folgenden Gründen eingesetzt:

Ausfallssicherheit: dies ist meist der Hauptgrund für die Replizierung. Man kann auf den Master genauso gut wie auf die einzelnen Replicas zugreifen und bekommt in jedem Fall die gleichen Daten (falls alles am aktuellen Stand ist)

Lastverteilung: es werden nicht alle Anfragen an eine Datenbank (Master) gerichtet, sondern diese werden auf alle vorhandenen Replicas verteilt.

Zugriffsgeschwindigkeit: dieser Punkt kommt dann zur Geltung, wenn die LDAP-Datenbank sich weit entfernt vom Client befindet und die Netzwerkverbindung dorthin langsam ist (z.B. über ein WAN), dann wird meist ein Replica des weit entfernten Masters im lokalen Netz eingerichtet.

Für die Replizierung gibt es mehrere Vorgehensweisen:

Intervall: das Intervall bestimmt die Zeitspanne innerhalb der alle Replicas mit dem Master abgeglichen werden (aktuelle Daten stehen immer am Master). Es muss gut auf die Bedürfnisse abgestimmt sein, denn ein zu langes Intervall führt zu schlecht übereinstimmenden Replicas und ein zu kurzes Intervall zu viel Synchronisationsaufwand.

Komplette Replizierung: der komplette Datenbestand des Masters wird an alle Replicas zum Abgleich geschickt – wird meist durchgeführt wenn ein neuer Replica-Rechner installiert wird.

Inkrementelle Replizierung: Es werden nur Änderungen an die Replica-Datenbanken geschickt, um diese auf den neuesten Stand zu bringen. (Problem von misslungenen Updates führen zu beschädigten Datenbanken)

Man kann die Datenbank auch partitionieren, um folgende Probleme aus dem Weg zu schaffen:

Zu viele Objekte: Je nach Implementierung des Verzeichnisdienstes ist dieser in der Anzahl der zu speichernden Objekte beschränkt. (die Zahlen liegen bei rund 1000 Objekten je Container und an die 100.000 Objekte in der gesamten Baumstruktur)

Zugriffsgeschwindigkeit: je größer und komplexer die Datenbank ist, desto zeitintensiver werden Such-Anfragen darin.

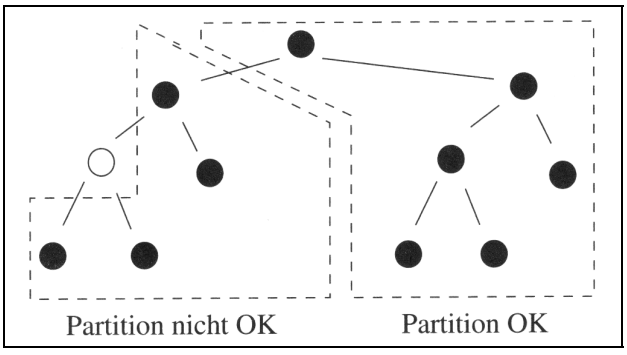
Aufwand der Replizierung: der Replizierungsaufwand steigt natürlich auch mit der Anzahl der Objekte in der Datenbank.

Zum Beispiel ist es oft sinnvoll, einzelne Länder oder Filialen einer Firma zu trennen um somit kleinere und effizientere Verzeichnisdienst-Partitionen zu erhalten.

Grundregeln der Partitionierung:

Die Wurzel einer Partition muss entweder ein Container sein, oder es handelt sich um die Root-Wurzel des gesamten Verzeichnisbaums

Eine Partition muss sich immer auf einen kompletten Teilbaum beziehen



Partitionen dürfen keine Löcher aufweisen. *Abb. 2.8: Regeln der Partitionierung* [Banning01 – Abb.3.24]

2.5 LDAP-Client

Die Hauptfunktionalität von LDAP-Systemen stellt der Server(-dienst) zur Verfügung. Der LDAP-Client muss/kann sich nun um die Nutzung der bereitgestellten Informationen kümmern.

2.5.1 Datenzugriff

Der Datenzugriff, als elementarer Bestandteil eines Verzeichnisdienstes, basiert auf dem Senden von Anfragen an den „Server“ (damit ist der Server-Dienst gemeint), wobei man unterscheidet zwischen:

Anfragen mit *einem* Ergebnis (Anfrage Ergebnis Ergebnis-Code)

Anfragen mit *mehreren* Ergebnissen: alle Teilergebnisse werden hintereinander an den Client zurückgeschickt

(Anfrage Erg.-Teil 1 Erg.-Teil 2 ... Erg.-Teil n Ergebnis-Code)

Parallele Anfragen: LDAP bietet, anders als andere Protokolle, die Möglichkeit Anfragen parallel zu senden, d.h. ein Client kann nach dem Senden der ersten Anfrage und bevor dem Erhalt der Antwort darauf, weitere Anfragen losschicken. Natürlich können auch mehrere Clients gleichzeitig Anfragen an den Server schicken, dies hängt dann vom Server ab, wie viele dieser „parallel“ beantworten kann.

2.5.2 Verwendung der LDAP-Daten

Erhaltene Daten sollten auch einen Zweck erfüllen. Man kann die Verwendung in zwei große Klassen einteilen:

Information: hierunter versteht man die manuelle oder automatische Informationsabfrage wie zum Beispiel Telefonnummern eines Mitarbeiters oder IP-Adressen von PCs oder ähnlichem.

Konfiguration: in den heutigen großen Netzwerklandschaften ist oft sehr wichtig viele Dienste zu konfigurieren, hierbei kann LDAP seine Dienste erweisen um dies zentral und effizient zu lösen. Dies kann je nach Anwendung und Konfiguration von manuell bis vollautomatisch geschehen (näheres dazu im Praxisteil)

2.5.3 Sicherheit von LDAP

Dieses Thema möchte ich nur kurz erwähnen, da es den Rahmen dieses Projektes bei weitem sprengen würde.

Netzwerksicherheit: ... es ist darauf zu achten, dass die LDAP-Dienste richtig konfiguriert sind (Port-, IP-Beschränkungen) – eine richtig angepasste Firewall ist auf dieser Ebene sicherlich die beste Lösung.

Datensicherheit: ... ist am besten über das „Transport Layer Security“ (**TLS**)-Protokoll zu erreichen

Authentifizierung: zur sicheren Authentifizierung stellt LDAP den „Simple Authentication and Security Layer“ (**SASL**) Mechanismus zur Verfügung

Weiters sollten natürlich Punkte wie ...

Zugriffsschutz

Rechtevergabe

Protokollierung

... nicht fehlen.

Genaueres zu diesem Thema findet man u.a. in [Johner98], [OpenLDAP-Admin], oder per Web-Recherche ... unter den Schlagwörtern „Security“, SASL, TLS, ... u.ä.

3

3 Varianten von LDAP

3.1 OpenLDAP

OpenLDAP ist eine Opensource Implementierung des Lightweight Directory Access Protokolls und besteht aus

slapd: Stand-alone LDAP Server

slurpd: Stand-alone LDAP Replication-Server

Bibliotheken

Utilities, Tools, Beispielimplementierungen.

Derzeit liegt die Software in der Version 2.2.11 vor.

Die erste Version dieser Software (1.0) wurde am 28. August 1998 released. Zwei Monate später wurde die Version 1.1 veröffentlicht. Diese Version unterschied sich von ihrer Vorgängerversion vor allem durch zusätzliche Features wie extern konfigurierbare Clients (mit Hilfe der Datei ldap.conf) und Unterstützung des GTK- und PHP3 Interfaces. Auch gab es einige Verbesserungen in Punkto Sicherheit, wie z.B. "ACL by group" Unterstützung, Passwörter wurden nicht mehr in Klartext übertragen, sondern SHA1, MD5 oder crypt(3) verschlüsselt und erweiterten TCP-Wrapper Support. Ab dieser Version gab es auch eine automatisierte Konfiguration und eine Plattform für Upgrades. Zu Beginn des Jahres 1999 wurde die Version 1.2 veröffentlicht. Diese enthielt neben zahlreicher Bugfixes und Updates auch Neuerungen wie eine TCL SDK (von NeoSoft), ein LDAP-Passwort-Tool sowie die Unterstützung von "salted passwords". Eine große Neuerung stand im August 2000 auf dem Programm, als OpenLDAP in der Version 2.0 veröffentlicht wurde. Eine der wichtigsten Neuerungen war, dass nun LDAPv3 (RFC3377,2251-2256,2829-2830) voll unterstützt wurde. Dazu gehört die Unterstützung von: [Open LDAP]

"Strong Authentication" (SASL) (RFC2829),

"Start TLS" (RFC2830),

"Language Tags" (RFC2596),

"DNS-based service location" (RFC2247+RFC3088) und

“Password Modify” (RFC3062).

Am Stand-Alone LDAP-Server wurden bei Version 2.0 auch einige Erweiterungen vorgenommen. Zu den wichtigsten zählen: [Open LDAP]

“Named References/ManageDsaIT” (RFC3296),

Ein verbessertes Zugriffskontrollsystem,

“Thread pooling”,

Unterstützung von preemptiven Threading und

“Multiple listener”- Unterstützung

Außerdem wurde LDIFv1 (RFC2849) unterstützt und die Plattform/Subsystemerkennung wurde entscheidend verbessert.
Nicht ganz 2 Jahre später (Juni 2002) wurde die Version 2.1 released. Sie glänzte durch verbesserte Stabilität und zusätzlichen Funktionen wie: [Open LDAP]

Transaktion-orientiertes Datenbankbackend,

Verbesserte Unicode/DN Unterstützung,

SASL authentication/authorization mapping,

SASL Authentikationsgeheimnisse werden direkt im Verzeichnis gespeichert,

Verbesserte administrative Zugriffssteuerung,

Verbesserte Systemwartungsmöglichkeiten,

LDAP C++ API

Verbesserte LDAP C & TCL APIs

Meta Backend

Monitor Backend

Virtual Context "glue" Backend

Außerdem wurden einige LDAPv3 Erweiterungen vorgenommen. Zu ihnen zählen: [Open LDAP]

Verbesserte Sprach Tag/Range Option Unterstützung,

“objectClass”-basierende Attributlisten

Zusätzliche Operation: LDAP Who am I?

“LDAP no-op Control”

“Matched Values Control”

Zusätzlich wurden einige LDAP-Features entscheidend erweitert.
Die **aktuelle Version** ist LDAP 2.2 und erschien im Dezember 2003. Zu den aktuellen Neuerungen zählen: [Open LDAP]

"LDAP Sync"-based lightweight replication

Proxy Cache Support

Hierarchical Backend

NS-SLAPI Support

Backend Layering

Access Control extensions including dynamic group support

LDAPv3 extensions:

- ACID extensions
- Cancel Operation
- Content Synchronization Operation
- DIT Content Rules
- Duplicate Entry Extension
- Simple Paged Results Extension
- Proxy Authorization Extension

Für **zukünftige Realeases** sind Erweiterungen wie [Open LDAP]

Configuration Backend

Built-in SASL/EXTERNAL support

Internationalization/Localization

Adaptive Cache Tuning

Improved Internationalization/Localization

LDAPv3 extensions:

- Distributed Operations
- LDAP Transactions
- LDAP Turn Operation for SyncRepl
- Collective Attributes
- Component Matching

Da OpenLDAP ein Opensource Projekt und eine gute Implementierung des LDAP Stands ist, haben wir uns entschieden dieses Paket für unsere Arbeit zu verwenden. Mehr zur Installation Inbetriebnahme und Wartung finden Sie im Praktischen Teil unserer Ausarbeitung.

3.2 IMBs LDAP

Der **Tivoli Directoryserver** von IBM ist eine kommerzielle Implementation des LDAP Protokolls.

Produktfeatures: [IBM]

LDAP V3 Unterstützung garantiert Kompatibilität mit industriellen Anwendungen, die auf den LDAP Standard beruhen.

Die IBM DB2® Universal Database V8.1 Engine bietet Skalierbarkeit sowohl für Millionen von Einträgen als auch Gruppen von hunderttausenden von Mitgliedern.

Breite Plattformunterstützung: Sowohl AIX®, SolarisTM, Microsoft® Windows® 2000, und HP-UXTM, als auch Linux Distributionen für Intel, und IBM eServer iSerie, pSerie und zSerie Plattformen werden unterstützt.

Leichte Webadministration mit Hilfe einer übersichtliche GUI.

Leicht einzubinden in IBM Betriebssysteme, WebSphere® Middleware und Tivol Identitymanagement- und Securityprodukten.

3.3 Suns LDAP

Der **System Directory Server** von Sun ist eine weitere kommerzielle Implementierung des Lightweight Directory Access Protokolls. Sun schreibt auf der unter Quelle Sun angegebenen Produktseite:

> *"The Sun Java System Directory Server 5.2 (formerly Sun ONE Directory Server 5.2) is software that provides a central repository for storing and managing identity profiles, access privileges, application and network resource information. With over 1.5 billion licenses sold, the Java System Directory Server is the most widely deployed general-purpose, LDAP-based directory server. The Java System Directory Server is rated #1 in the Directory Server Magic Quadrant by Gartner Research, #1 market share leader by Radicati Research, #1 market penetration by The Burton Group."* [Sun].

Produktfeatures: [Sun]

High-Performance Design optimiert für 64 bit Architekturen

Vierpass Multimaster Replikation sowohl über LAN als auch über WAN Netzwerke.

Gutes Passwortmanagement

Protokollunabhängige Architektur.

3.4 Novells LDAP

Novell eDirectory ist eine Implementierung des LDAP Protokolls von Novell. Auf der Produktseite von Novell ist folgende Kurzbeschreibung des Produktes zu finden:

> *"For over a decade Novell eDirectory has defined and reinvented the role of a directory service. More than just an LDAP data store, eDirectory is the identity foundation that links your users and their access rights with corporate resources, devices and security policies. Among directory services, eDirectory is uniquely capable of meeting the demands of large-scale, high-end directory deployments. It offers the compatibility, security, reliability, scalability and manageability required for internal and Internet deployments supporting millions of identities."*
> [Novell].

Produktfeatures: [Novell]

Kompatibilität mit offenen Standards und vielen Plattformen. eDirectory ist voll kompatibel zum LDAPv3 Standard. Außerdem unterstützt eDirectory Extensible Markup Language (XML), Directory Services Markup Language (DSML), Simple Object Access Protocol (SOAP) und einige andere offene Standards. Unterstützt werden Plattformen wie Linux, Windows, Solaris, AIX, NetWare und HP-UX. Daher ist eDirectory auch für heterogene Netzwerke einsetzbar.

Echtzeitkontrolle über die Zugriffe von Benutzern auf Ressourcen. Benützer können so jederzeit aus dem Netzwerk ausgeschlossen werden.

Sehr gute Unterstützung von Wachstum im Netzwerk. Es funktioniert auch noch mit über einer Billion Identitäten. Laut Novell ist das Wachstum mit keinen spürbaren Einbußen an Performance verbunden.

Zentrale, webbasierende Wartung über die Managementkonsole “Novell Manager”, über die auf alle Funktionen zentral zugegriffen werden kann.

4

4 Vergleich LDAP vs. Active Directory

Eigentlich gibt es zwischen Microsofts Active Directory und dem LDAP Standard keine all zu großen Unterschiede. Es liegt jedoch auf der Hand dass Active Directory in die Betriebssysteme von Windows-Servern sehr leicht einzubinden ist. Wenn bis jetzt alle Benutzerdaten in einer Windowsumgebung verwaltet wurden, gibt es eigentlich nicht viele Gründe, die gegen Active Directory sprechen. Das frühere Argument, dass Active Directory nicht sehr performant ist, ist inzwischen überholt.

Auf der anderen Seite ermöglicht das Verwenden von reinen LDAP jedoch optimale Kompatibilität zu vielen Opensourceprodukten, Java-Applikationen, IBM-, Sun- oder Novellservern uvm. Wenn man primär Java Applikationen verwendet, die auf das Directory zugreifen, ist reines LDAP die beste Lösung. Außerdem verspricht LDAP eine bessere Skalierbarkeit, besonders wenn es auf High-End-Hardware läuft, die oft in Zusammenarbeitung mit Active Directory nicht funktioniert.

5

5 PAM (Pluggable Authentication Modules)

Eine ganz gute Definition dazu gibt [Morgan04]:

> *Linux-PAM (Pluggable Authentication Modules for Linux) is a suite of shared libraries that enable the local system administrator to choose how applications authenticate users.*
>
> *In other words, without (rewriting and) recompiling a PAM-aware application, it is possible to switch between the authentication mechanism(s) it uses. Indeed, one may entirely upgrade the local authentication system without touching the applications themselves.*

Dabei handelt es sich um ein komplettes Sicherheitskonzept, welches im Prinzip programmunabhängig ist. Der Sinn und Vorteil dieses ganzen Konzeptes ist, dass sich nicht jeder Programmierer um die Implementierung eines eigenen Authentifikationsverfahren kümmern muss, sondern bereits existierende und somit meist effiziente und gut getestete Module, in sein Programm einbauen bzw. eigentlich als Modul nur einstecken braucht. Weiters gibt es auch von Anwenderseite einige gute Gründe die für PAM sprechen. Wie schon der Name sagt, sind die Module austauschbar (pluggable), also kann sich der Anwender (hier der Administrator) für das für ihn am besten passende Modul entscheiden, sei es aus Sicherheits-, Kompatibilitäts- oder anderen Gründen. Der Computeranwender profitiert auch davon, da er sich mit nur einem Account an allen Unterstützen Anwendungen anmelden kann.

5.1 Funktionsweise von PAM

PAM besteht aus 2 Grundbausteinen:

PAM-Bibliothek (Grundbibliothek)

PAM-Modulen (spezifisch für jeden Einsatztyp)

Die Offenheit und die Austauschbarkeit von Komponenten verdankt PAM seinem Aufbau: es ist ein **rein passives System**, d.h. die Applikation muss sich um den Aufruf von PAM kümmern. Grund dafür ist, dass nur die Anwendung selbst weiß, wie die Authentifizierungsdaten eingelesen und verarbeitet werden können – z.B. können die Benutzerdaten über die Tastatur daherkommen oder auch von einer Chipkarte eingelesen werden. Somit kümmert sich die Anwendung darum, dass die Daten eingelesen werden und

ruft mit diesen Daten dann das PAM-System auf, welches anhand von vordefinierten Einstellungen diese Daten überprüft und dann eine Antwort zurückliefert. Die Abbildung 5.1 zeigt den System-Aufbau ganz schematisch.

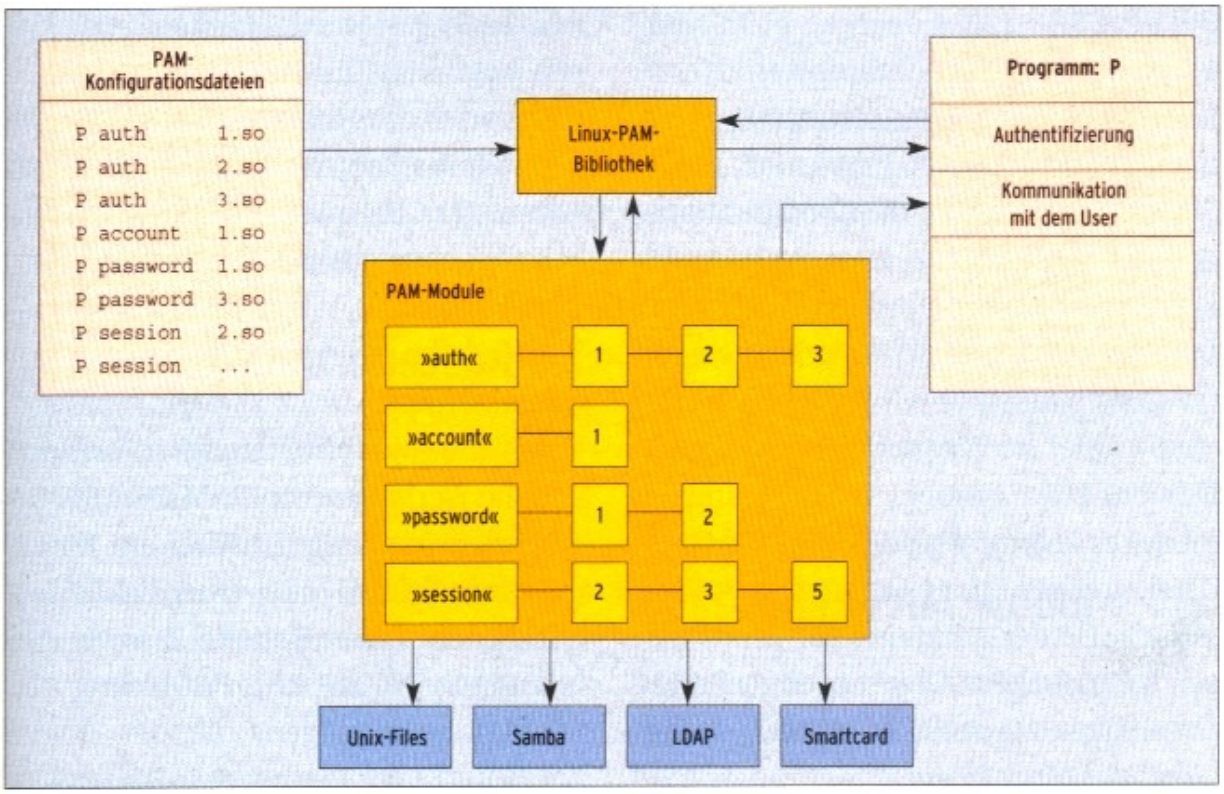


Abb. 5.1: Systemaufbau von PAM [Suchodoletz04 - Abb.1]

Die Programmdateien von PAM befinden sich in der Regel unter */lib/security* und die Konfigurationsdateien entweder in */etc/pam.conf* oder als separate Datei für jeden Dienst unterhalb von */etc/pam.d/* - je nach Anzahl der Dienste und Komplexität der Einstellungen. Folgendes Beispiel zeigt den Aufbau einer solchen Config-Datei:

```
01 # Modultyp  Modulsteuerung  Modulpfad       Argumente
02 auth        required        pam_unix2.so    nullok
03 account     required        pam_unix2.so
04 password    required        pam_unix2.so
   strict=false
05 session     required        pam_unix2.so    debug
06 session     required        pam_devperm.so
07 session     required        pam_resmgr.so
```

Abb. 5.2: Eine Typische PAM-Konfigurationsdatei [Suchodoletz04 – Listing 1]

Modultypen: die PAM-Module sind ihrer Funktionalität entsprechend in verschiedene Kategorien oder Modultypen eingeteilt:

Auth: für Benutzeridentifizierung und –authentifizierung – unterschiedlichste Arten dafür verfügbar (Name+Passwort, Smart-Card, Biometrie, …)

Account: für Accountrichtlinien wie Zugriffsrechte, Limits, etc. zuständig (unabhängig von Authentifizierung)

Password: dient der Passwortverwaltung (Änderung, Richtlinien für gute Passwörter, u.ä.)

Session: um Sitzungseinstellungen der jeweiligen Benutzer zu verwalten. Z.B.: mounten von zusätzlichen Verzeichnissen, Logging, ...

Modulsteuerung: die einzelnen Module werden der Reihenfolge abgearbeitet und anhand folgender Parameter wird entschieden was getan wird: Erfolg (*success*) oder Misserfolg (*fail*)

required: …ist entscheidend für das Gesamtergebnis der Auswertung. Ein „fail“ bei einem required-Modul führt zu einem finalen Misserfolg nach Abarbeitung aller anderen Module.

requisite: ähnlich wie required, nur dass dabei die Kontrolle sofort nach einem „fail“ an die Anwendung zurückgegeben wird.

sufficient: nach einem erfolgreichen sufficient-Modul wird sofort success an die Anwendung zurückgegeben, falls alle vorherigen required-Module success gemeldet haben.

optional: optionale Module dienen nur im Notfall, d.h. wenn sonst keine Entscheidung getroffen werden konnte, zur Entscheidungsfindung.

Das PAM-Standardpaket beinhaltet schon eine Menge Module für die häufigsten Anwendungen (wie z.B. *pam_unix*, das für die Unix-Standard-Password-Authentifizierungs über /etc/passwd und /etc/shadow zuständig ist), weitere interessante und wichtige Module sind zum Beispiel:

pam_access: für Access Control Lists

pam_cracklib: setzt Richtlinien für gute Passwörter durch

pam_ldap: Benutzerauthenifizierung gegen eine LDAP-Datenbank

pam_permit: liefert immer success zurück (Achtung: nur für Spezialfälle einsetzen!)

pam_deny: liefert immer fail zurück, verhindert also immer den Zugriff auf das Modul

... eine umfangreichere Modulliste ist u.a. bei [Suchodoletz04] und bei [Morgan04] zu finden. [Suchodoletz04]

5.2 Vorteile von PAM:

Austauschbare Authentifizierungsverfahren

Nur eine Benutzer- und Account-Verwaltung für alle Systeme

Zentral konfigurierbar

Es existiert bereits eine große Menge an fertiger PAM-Modulen für alle gängigen Anwendungen

Unterschiedliche Front- (Smartcards, Biometrie-basiert, ...) und Back-Ends (alle gängigen Dienste) möglich.

Weniger Fehlerquellen

Teil II – Praktische Konfiguration von LDAP unter Linux:

6 Grundkonfiguration

Als Basissystem für den **Server** haben wir uns für ein ***Gentoo-Linux*** entschieden (das reine Debian-System war mit dem vorhandenen Notebook nicht besonders gut kompatibel). Im Grunde genommen spielt das Basissystem aber kaum eine Rolle, solange es auf Linux basiert, da maximal ein paar Flags und Verzeichnisse anders sein dürften.
Das **Client**-System, mit dem wir auf den LDAP-Server zugreifen ist der Einfachheit halber ein ***Knoppix-System*** (V.3.4 mit Kernel 2.4.23).

Als Software haben wir folgende Komponenten und Versionen verwendet:

Open LDAP v.2.2.11 [OpenLDAP]

Berkeley DB (BDB) v.4.1.25 [Sleepycat]

Pam_Ldap v.169 [PADL]

7 Installation von LDAP

7.1 Installation der benötigten Datenbank (hier BDB – auch LDBM ist möglich)

(1) Sourcecode von der Herstellerseite herunterladen
ftp://sleepycat1.inetu.net/releases/db-4.1.25.tar.gz

(2) Sourcecode in ein temporäres Verzeichnis entpacken

```
[/tmp]$ tar -xvzf db-4.1.25.tar.gz   /tmp/db-4.1.25/
```

(3) Sourcecode mit *configure* für Installation einrichten

```
[/tmp/db-4.1.25/build_unix]$ ../dist/configure
```

(4) Programm mit *make* kompilieren

```
[/tmp/db-4.1.25/build_unix]$ make
```

(5) Programm mit *make install* installieren (SU-Rechte notwendig!)

```
[/tmp/db-4.1.25/build_unix]# make install
```

7.2 Installation von Open LDAP

(1) Sourcecode von der Herstellerseite herunterladen
ftp://ftp.openldap.org/pub/OpenLDAP/openldap-release/openldap-2.2.11.tgz

(2) Sourcecode in ein temporäres Verzeichnis entpacken

```
[/tmp]$ tar -xvzf openldap-2.2.11.tgz   /tmp/openldap-2.2.11/
```

(3) Sourcecode mit *configure* für Installation einrichten

```
[/tmp/openldap-2.2.11]$ ./configure
```

(4) Programm mit *make* kompilieren

```
[/tmp/openldap-2.2.11]$ make
```

(5) Programm mit *make install* installieren (SU-Rechte notwendig!)

```
[/tmp/openldap-2.2.11]# make install
```

7.3 Installation von PAM_LDAP

(das PAM-Grundsystem mit den meisten Standardmodulen ist bei den meisten Linux-Distribution bereits integriert und muss somit nicht extra installiert werden)

(1) Sourcecode von der Herstellerseite herunterladen ftp://ftp.padl.com/pub/pam_ldap.tgz

(2) Sourcecode in ein temporäres Verzeichnis entpacken

(3) `[/tmp]$ tar -xvzf pam_ldap.tgz  /tmp/pam_ldap-169/`

(4) Sourcecode mit *configure* für Installation einrichten

(5) `[/tmp/pam_ldap-169]$ ./configure`

(6) Programm mit *make* kompilieren

(7) `[/tmp/pam_ldap-169]$ make`

(8) Programm mit *make install* installieren (SU-Rechte notwendig!)

(9) `[/tmp/pam_ldap-169]# make install`

7.3 Testen der Installation

(1) Das LDAP-Source-Package stellt Test-Skripte bereit, welche die vollständige Grundfunktionalität des Servers sicherstellen sollen. Diese sind über das Kommando *make test* nach dem vollendeten Konfigurieren und Kompilieren (nach Schritt (7) siehe oben)

(2) Starten des Servers (slapd) mit Debug-Informationen `sudo /usr/local/libexec/slapd -d <debug-level>` (z.B **–d 63** für Basisdebug-Informationen; genaueres siehe [Banning01, S.56]

(3) Konfigurieren des LDAP-Server, hinzufügen von Testdaten (ldapadd) und suchen nach den Daten (ldapsearch)

eine umfangreiche Installation und Testanleitung ist unter [OpenLdap-Quick] zu finden.

8 Serverkonfiguration

8.1 LDAP-Server konfigurieren

8.1.1 Server-Dienst konfigurieren

Der LDAP Daemon wird primär über die Datei slapd.conf (bei mir im Verzeichnis /etc/openldap/) konfiguriert.
Diese Datei sieht bei mir folgendermaßen aus:

[/etc/openldap/]$ cat slapd.conf

```
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.

# -> Integration von Schemen zur Definition der LDAP-Baum-Struktur
include             /etc/openldap/schema/core.schema
include             /etc/openldap/mail-routing.oc.conf
include             /etc/openldap/schema/cosine.schema
include             /etc/openldap/schema/nis.schema
include             /etc/openldap/schema/inetorgperson.schema

# -> dort werden die Run-Time Infos abgelegt
pidfile             /var/run/openldap/slapd.pid
argsfile            /var/run/openldap/slapd.args

# -> ACLs sollten aus Sicherheitsgründen immer eingerichtet werden,
# aber der Einfachheit halber haben wir darauf  verzichtet.
# Das Beispiel zeigt aber ganz gut wie dies gemacht werden sollte:
# Sample access control policy:
#      Root DSE: allow anyone to read it
#      Subschema (sub)entry DSE: allow anyone to read it
#      Other DSEs:
#             Allow self write access
#             Allow authenticated users read access
#             Allow anonymous users to authenticate
#      Directives needed to implement policy:
# access to dn.base="" by * read
# access to dn.base="cn=Subschema" by * read
# access to *
#      by self write
#      by users read
#      by anonymous auth
#
# if no access controls are present, the default policy is:
#      Allow read by all
#
# rootdn can always write!

########################
# database definitions

database     bdb
suffix       "dc=my-domain,dc=com"

# -> die Rood-DN wird benötigt um Veränderungen an der Datenbank vorzunehmen
```

```
# Diese Rechte für Root können nicht über ACLs eingeschränkt werden!
# Cleartext passwords, especially for the rootdn, should
# be avoid.  See slappasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootdn       "cn=Manager,dc=my-domain,dc=com"
rootpw       secret

# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory    /var/lib/openldap-data

# Indices to maintain
index  objectClass  eq
```

Dabei werden im oberen Abschnitt der Datei die Includes definiert. Durch diese können vordefinierte Schemata verwendet werden. Ein weiters wichtiger Teil dieser Datei schreibt die verwendete Datenbank. In diesen Abschnitt muss man sowohl das Installationsverzeichnis der Datenbank (bei uns /var/lib/openldap-data), als auch verschiedene andere Dinge wie z.B. Domainname oder Administrator (hier Manager) anpassen. Die Schreibweise ist teilweise etwas eigentümlich, da man statt z.B. my-domain.com „dc=my-domain,dc=com“ schreiben muss. Als letztes legt man mit rootpw noch das Rootpasswort fest. Wir haben uns der Einfachheit halber für ein Klartextpasswort entschieden. Weitere Informationen zu dieser Datei so wie andere mögliche Ausprägungen und Kommandos finden Sie unter der Quelle [OpenLDAP-Admin].

Nach Anpassen der Datei slapd.conf kann die Konfiguration mit dem Befehl /usr/lib/openldap/slapd -t getestet werden. Ist die Konfiguration in Ordnung erscheint "success" in der Shell und man kann den Daemon mit /usr/lib/openldap/slapd starten.

8.1.2 Einfügen von Daten in den LDAP Server

Zum Auslesen von Benutzerdaten von lokalen Rechnern haben wir die OpenSource Migration Tools von Padl Software Pty Ltd [PADL] verwendet. Diese lädt man sich einfach von der Firmenhomepage herunter und entpackt sie. Es handelt sich dabei um eine Reihe von Perl-Scripts, die das Auslesen von Daten aus z.B. /etc/passwd recht einfach gestalten. Jedoch muss man sich zusätzlich zu den Tools auch noch die Pakete "nssldap" und "pamldap" von der gleichen Homepage runterladen, um die Daten später über LDAP auch verwenden zu können. Diese werden mit ./configure konfiguriert und mit make und make install installiert. Nach der

Installation muss man noch in einigen Dateien Änderungen vornehmen. So ersetzt man in /etc/ldap.conf die Standard Pfad- und LDAP-Angaben durch die eigenen Verzeichnisnamen. Außerdem muss (falls nicht vorhanden) die Datei /etc/lib-nss-ldap.conf erstellt werden und mit den Zeilen

```
host <ldap hostname>
base <dn>
```

gefüllt werden, wobei bei mir der LDAP-Server auf dem "localhost" rennt und dn gleich "dc=my-domain,dc=com" ist.
Sind diese Konfigurationen getätigt kann man ganz einfach mit z.B.

```
./migrate_passwd.pl /etc/passwd pass.ldif
```

die Userdaten des lokalen Rechners auslesen und in der Datei pass.ldif speichern. Es gibt hier einige nützliche Scripts:

migrate_all_*.sh: mehrere Skripte um die ganze Migration automatisch zu erledigen

migrate_slapd_conf.pl: liefert eine Beispiel-Config-Datei für einen PAM-fähigen LDAP-Server aus

migrate_base.pl: Basis-Verzeichnis-Struktur

migrate_passwd.pl: Extraktion der Passwort-Daten (incl. der Shadow-Passwörter)

weitere Netzwerk- und Host-Daten lassen sich auch extrahieren

migrate_fstab.pl

migrate_group.pl, migrate_hosts.pl

migrate_netgroup.pl, migrate_networks.pl

etc.

Nähere Informationen zu jeden einzelnen Script finden Sie auf der unter [PADL] angegebenen Firmenhomepage.

Wie fügt man nun diese Daten in die LDAP Datenbank ein?
Als erstes muss die Datei pass.ldif mit einen Texteditor geöffnet werden und man ersetzt (am besten mit der Find-Replace Funktion des Texteditors) jede „padl“-Domain durch seine eigene (in unseren Fall „dc=my-domain,dc=com“). Ist man damit fertig speichert man die Datei und fügt die Daten mit dem Befehl „ldapadd -x -D "cn=Manager,dc=my-domain,dc=com“ -W -f /home/florian/uni/seos/pass.ldif" in die LDAP Datenbank ein.

Dabei steht x für „Simple Authentication“, D für „bind dn“, W für „prompt for bind password“ und f dafür, dass ldapadd die Informationen aus der angegebenen Datei bezieht.

Somit wurde der LDAP Server richtig konfiguriert und einige Daten erfolgreich eingefügt.

8.1.3 Konfiguration des SSH-Servers

Die Konfiguration des SSH-Servers passiert großteils über die Datei /etc/pam.d/sshd. Darin muss man festlegen, dass die Authentifizierung über LDAP abläuft. Das macht man mit den Zeilen

```
auth        sufficient        /lib/security/pam_ldap.so

account     sufficient        /lib/security/pam_ldap.so
```

Serverseitig ist es also nicht wirklich herausfordernd einen SSH Server zu konfigurieren.
Gestartet wird der Server übrigens mit „sshd“. Man muss jedoch als root eingeloggt sein bzw. sich mit dem „su“ Befehl Superuserrechte holen um den Server starten zu können.

8.1.4 Einbinden von LDAP in den Apache Webserver

LDAP kann in Zusammenarbeit mit dem Apache Webserver recht nützlich sein, um z.B. verschiedene Verzeichnisse eines Webservers verschiedenen Benutzern (oder Gruppen) zugänglich zu machen, oder zu sperren.
Damit wir eine Authentifizierung vornehmen können müssen wir uns erst das „mod_ldap“ [mod_ldap] unter der in den Quellen angegebenen Homepage runterladen. Es handelt sich hierbei wiederum um ein OpenSource Projekt. Nach dem Download wird die Datei entpackt. Es gibt zwei verschiedene Arten Module in den Apache Webserver einzubinden [King01], dynamisch oder statisch. Meist wird das dynamische Laden bevorzugt, und so haben uns auch wir entschieden das LDAP-Modul dynamisch zu laden. In [King01] sind beide Arten des Einbindens in den Apache Webserver beschrieben.

Dynamisches Einbinden von mod_ldap in den Apache Webserver

Durch das Entpacken des Moduls wurde ein Verzeichnis „modauthldap“ erstellt, in dieses wechseln wir vorerst. Nun wird folgender Befehl in die Shell eingegeben:
apxs -L/usr/lib -lldap -llber -i -a -c mod_auth_ldap.c
Dadurch wird das Modul kompiliert und alle Dateien werden schon an die richtigen Verzeichnisse kopiert.

Das war es im Prinzip schon. Will man nun ein Verzeichnis schützen, wird in ihm die Datei **„.htaccess“** erstellt. Eine geeignete Beispielkonfiguration ist:

```
AuthName "Gesichertes Verzeichnis"
AuthType Basic
LDAP_Server 127.0.0.1
LDAP_Port 389
Base_DN "ou=apache,dc=my-domain,dc=com"
require valid-user
```

Erweiterte Optionen können Sie der Tabelle 1 aus [King01] entnehmen.

Damit ist das Verzeichnis nur Benutzern der Gruppe apache, der Firma "my-domain.com" zugänglich.

8.1.5 Konfiguration des „ProFTP“ FTP Servers

Im Folgenden wird das Einbinden von LDAP Benutzerauthentifizierung in den FTP Server proFTPD [proftpd] demonstriert. Dieser FTP Server unterstützt LDAP, es ist aber standardmäßig nicht aktiviert. Man ladet sich als erstes ProFTPD [proftpd] von der Produkthomepage runter und entpackt die Datei. Anschließend wird das Programm mit den Befehlen

```
./configure with-modules=mod_ldap
make
make install
```

installiert. Wichtig ist beim ./configure die Option „with-modules=mod_ldap“ nicht zu vergessen! Für make install bedarf es wie gewöhnlich root-Rechte.

Die Hauptkonfigurationsdatei heißt **/etc/proftpd.conf**. Eine Beispielkonfiguration: [King01]

```
ServerName "FTP-Server"
ServerAdmin king@t-king.de
ServerType Stand-alone
DefaultServer on
Port 21
RequireValidShell off
AuthPAM off
LDAPServer ldap.mycompany.de
LDAPDNInfo cn=Manager,o=MyCompany,c=DE secret
LDAPDoAuth on ou=proftpd,o=MyCompany,c=DE
```

Ein Benutzer, der Zugriff auf den FTP Server haben soll, muss nur der Gruppe proftpd zugeordnet werden.

8.1.6 Adressbuch über LDAP einrichten

Um ein Adressbuch über LDAP einzurichten benötigt man zuerst die richtigen Schemata, denn umfangreiche Personeninformationen lassen sich nicht im Standard-Schema speichern. Somit haben wir das Schema `/etc/openldap/schema/inetorgperson.schema` zur Server-Konfiguration hinzugefügt. Nun muss man eine Gruppe für alle Adresseinträge definieren: `ou=Addressbook,dc=my-domain,dc=com` unter welcher man nachher alle Personen hinzufügt. Das Hinzufügen der Daten kann man entweder wieder, wie bereits zuvor erwähnt über das Kommando ldapadd mit zuvor erstellten LDIF-Dateien erledigen, oder man fügt die Daten einfach über einen GUI-Client in die Datenbank ein. Ein ganz geeigneter dazu ist zum Beispiel der unter der GPL laufende GQ. Um aber darüber Daten bearbeiten zu können muss man diesen mit ausreichenden Rechten ausstatten. Dies geschieht über eine BIND-DN mit dazugehörigem Passwort. Entweder man gibt den Root-User an, oder einen anderen dazu berechtigten LDAP-User. Die eigentlichen User-Daten sind dann eine Sammlung aus zig Einträgen mit Name, Adresse, Telefonnummer, uvm.

8.2 Server-Dienste konfigurieren (über PAM)

Für einfache Konfigurationen mit wenig Diensten ist es sinnvoll alles in der Konfigurationsdatei **`/etc/pam.conf`** einzustellen. Für komplexere Server ist es aber anzuraten je Dienst eine eigene Konfigurationsdatei (entsprechend dem Dienstnamen) im Verzeichnis **`/etc/pam.d/`** einzurichten (diese Standard-Config-Files sind meist schon von der PAM-Standardinstallation vordefiniert eingerichtet). Somit haben wir uns auch für letzteren Fall mit getrennten Config-Files entschieden. Im folgenden nun eine Auflistung interessanten Dienste mit deren Config-File:

login: der Standard-Unix-Login-Prozess `/etc/pam.d/login`

passwd: Std.-Unix-Password-Utility `/etc/pam.d/passwd`

Die folgenden Dienste sind aber Programm-abhängig und können sich in der Config-Datei unterscheiden:

ssh: die Secure Shell `/etc/pam.d/ssh (sshd)`

ftp: FTP-Protokoll `/etc/pam.d/ftp`

cvs: Concurrent Versions System `/etc/pam.d/cvs`

smtp: Simple Mail Transfer Protokoll `/etc/pam.d/smtp`

pop3: Post Office Protokoll V.3 `/etc/pam.d/pop3`

imap: der Standard-Unix-Login-Prozess `/etc/pam.d/imap`

http: Standard Web-Protokoll `/etc/pam.d/http`

8.3 Testen der PAM-Funktionalität

Um sicher zu gehen, dass die Server-Dienste funktionieren und mit den PAM-Modulen richtig zusammenspielen (z.B. dass nicht die falschen Config-Files verwendet wurden) sollte man dies testen.

Wir werden dies anhand des SSH-Dienstes kurz durchspielen:

1. **Test auf normale Funktionalität**
   1.1. Einrichten des SSH-Servers
   1.2. PAM auf Standard-Konfiguration belassen (über pam_unix.so authentifizieren)
   1.3. Starten des SSH-Servers (/sbin/sshd)
   1.4. Testen der SSH-Verbindung vom Client aus
   1.5. Ergebnis: *es wird das am Server gesetzte Passwort benötigt!*
2. **Test mit dem auth-Modul <pam*_permit.so*>** *(jede Verbindung sollte erlaubt sein, unabhängig vom Passwort)*
   2.1. im PAM-Config-File folgende Zeile an oberster Stelle einfügen

   ```
   auth      sufficient  pam_permit.so
   ```

   (Steuerung sufficient, um nach erfolgreichen permit sofort success zurückzugeben)
   2.2. Starten des SSH-Servers (/sbin/sshd)
   2.3. Testen der SSH-Verbindung vom Client aus
   2.4. Ergebnis: *egal welches Passwort eingegeben wird, wird die Verbindung immer akzeptiert!*
3. **Test mit dem auth-Modul <pam_deny.*so*>** *(keine Verbindung sollte erlaubt sein)*
   3.1. im PAM-Config-File folgende Zeile an oberster Stelle einfügen

   ```
   auth      requisite   pam_deny.so
   ```

   (Steuerung requisite, um nach Fehlschlag sofort fail zurückzugeben)
   3.2. Starten des SSH-Servers (/sbin/sshd)
   3.3. Testen der SSH-Verbindung vom Client aus

3.4. Ergebnis: *egal welches Passwort eingegeben wird, wird die Verbindung immer zurückgewiesen!*

9 Clientkonfiguration

9.1 LDAP-Client konfigurieren

Der Client muss für den Zugriff auf Serverdienste lediglich wissen, wo sich der LDAP-Server befindet und wie er darauf zugreift. Dies wird über die Datei <ldap.conf> eingerichtet.

```
[/etc/openldap/] $ cat ldap.conf
# See ldap.conf(5) for details
# This file should be world readable but not world writable.

BASE   dc=my-domain, dc=com
#HOST  193.171.40.9 bzw. über hostname wenn in /etc/hosts definiert
HOST   server
PORT   389
#URI   ldap://193.171.40.9
URI    ldap://server

#SIZELIMIT   50
#TIMELIMIT   30
#DEREF       never

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

# The distinguished name to bind to the server with.
# Optional: default is to bind anonymously.
BINDDN cn=LDAPAuth,dc=my-domain,dc=com

# The credentials to bind with.
# Optional: default is no credential.
BINDPW bindsecret
```

Die Verbindung zwischen PAM und LDAP stellt das pam_ldap-Modul her, welches über die Datei **pam_ldap.conf** wie folgt konfiguriert wird:

```
HOST   server
BASE   dc=my-domain,dc=com

# legt fest ob das Passwort bereits lokal verschlüsselt wird oder erst vom Server
```

```
#pam_crypt   local

# Steuerung des Datenbank-Zugriffs:
# Filter to AND with uid=%s
pam_filter objectclass=account

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

es gibt hier noch eine Unmenge an Einstellungen, welche aber nicht unbedingt benötigt werden und in dem bei der Installation erstellten Default-Config-File gut erklärt sind.

Für das lokale Login muss am Client, aber das Login-PAM-Modul konfiguriert werden, sodass dieses die Authentifizierung über den LDAP-Server erledigt. Dazu konfiguriert man die Datei </etc/pam.d/login> entsprechend:

/etc/pam.d/login

... analog zum SSH-Server müssen die pam_ldap.so Module hinzugefügt werden.

Um die Netzwerkinformationen LDAP-gerecht aufzubereiten bedarf es einer kleinen Änderung in **nsswitch.conf:**

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

9.2 LDAP-Client Anwendungen

Authentifizierung:

Alle Authentifizierungsmechanismen sollten transparent vom Benutzer erfolgen, d.h. er sollte nicht bemerken und keine Umstände damit haben, dass er sich gerade über den LDAP-Server anmeldet und nicht über lokale System-Einstellungen. Je nach Konfiguration kann eingerichtet sein, dass bei einem Fehlschlagen der LDAP-Authentifizierung sofort lokal geprüft wird, ob dieser Benutzer erlaubt ist, ohne dass dieser nochmals seine Anmeldungsdaten eingeben muss (dies kann über die PAM-Config-Files mit Zusatzparametern wie *use_first_pass* eingerichtet werden)

Davon profitieren wird hauptsächlich aber nur der Administrator, der das ganze dadurch zentral warten und managen kann, was bei sehr großen Netzwerken eine enorme Erleichterung ist!

Verwenden der LDAP-Adressbuch-Informationen:

Nutzen kann man diese Adress-Informationen nun über diverse Clients. Die meisten Adressbücher von Mail-Clients unterstützen LDAP-Informationen. Der Web-Browser kann auch meist LDAP-Informationen anzeigen (URI = ldap://<server>/<dn>). Und spezielle LDAP-Clients lassen sich natürlich auch für diese Aufgabe einsetzen.

Verwenden sonstiger Informationen:

Ein zentraler Verzeichnisdienst kann selbstverständlich noch wesentlich mehr Informationen speichern und bereitstellen. Welche dann über verschiedenste Anwendungen verwendet oder auch nur eingesehen werden können. Beispiele hierfür wären:

Netzwerkinformationen: IP-Adressen, DNS-Server, Gateways, Drucker

Personaldatenbank

Unternehmensstruktur

Raumpläne

10 Quellen:

[Radtke04] - RADTKE Stefan, REPP Andreas: *„Wanderkarte – Migration von NIS zu LDAP“*, aus IX Magazin 4/2004 (S.132-135), Hannover: Heise Zeitschriftenverlag, 2004.

[Zerbst04] - ZERBST Carsten.: *„Baum-Pflege – LDAP-Server mit Tcl bearbeiten“*, aus Linux Magazin 3/2004 (S.105-109), München: Linux New Media AG, 2004.

[King01] – KING Thomas: *“Workshop: LDAP Teil 1”,* aus Linux Magazin 6/2001, München: Linux New Media AG, 2001.

- http://www.linux-magazin.de/Artikel/ausgabe/2001/06/ldap/ldap.html

[Suchodoletz04] - SUCHODOLETZ Dirk von, WALTER Martin: „*PAM-Grundlagen – Die Pforten zur Linux-Maschine: Pluggable Authentication Modules*“ aus Linux Magazin 5/2004 (S.38-45), München: Linux New Media AG, 2004.

[Banning01] - BANNING Jens: "*LDAP unter Linux - Netzwerkinformationen in Verzeichnisdiensten verwalten*", Addison-Wesley Verlag, 2001.

[Johner98] - JOHNER Heinz, BROWN Larry, u.a.: *“Understanding LDAP”*, 1.Auflage, Austin/TX: IBM Redbooks, Juni 1998.

- http://www.redbooks.ibm.com/redbooks/pdfs/sg244986.pdf

[Morgan04] – MORGAN Andrew G.: “*The Linux-PAM System Administrators' Guide*”, Stand: 2004.

- http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html

[Williams01] - WILLIAMS Adam Tauno: *“LDAP and OpenLDAP – on the Linux Platform”*, 2001. (herausgegeben unter GNU Free Documentation Licence, gewartet von der Kalamazoo Linux User's Group)

- ftp://ftp.kalamazoolinux.org/pub/pdf/ldapv3.pdf

Standards, Produkte und Online-Quellen: (Stand: April 2004)

(herausgegeben bzw. bereitgestellt von Organisationen, Verbänden oder freien User-Gruppen)

[IBM] - IBMs LDAP Standard: http://www.ibm.com/software/tivoli/products/directory-server/

[SUN] - SUNs LDAP Standard: http://wwws.sun.com/software/products/directory_srvr/home_directory.html

[Novell] - Novells LDAP Standard: http://www.novell.com/products/edirectory/

[RFC]: http://rfc.net/ , http://www.faqs.org/rfcs/

[OpenLDAP] – Open LDAP Standard: http://www.openldap.org/

[OpenLDAP-Quick] – Open LDAP Quick Starter Guide: http://www.openldap.org/doc/admin22/quickstart.html

[OpenLDAP-Admin] – Open LDAP Administration Guide: u.a. LDAP-Server:

- http://www.openldap.org/doc/admin22/
- http://www.openldap.org/doc/admin22/slapdconfig.html

[Rage.net] - http://www.rage.net/ldap/

[Debian] – Debian Linux: http://www.debian.org/

[Knoppix] - Knoppix: http://www.knopper.net/

[Gentoo] – Gentoo Linux: http://www.gentoo.org/

[Sleepycat] – Berkeley Database: http://www.sleepycat.com/

[PADL] – PAM-LDAP-Modul, Migrationsskripte: http://www.padl.com/

[mod_ldap] - LDAP Authentication module for Apache 1.3.x: http://www.muquit.com/muquit/software/mod_auth_ldap/mod_auth_ldap.html

[proftpd] - Highly configurable GPL-licensed FTP server software

- http://www.proftpd.org
- http://www.proftpd.org/docs