LDAP

Philipp Wendler

IEEE Student Branch Passau

28. Mai 2009

Inhalt

Was ist LDAP?

Lightweight Directory Access Protocol

- Lightweight? eher nicht
- Protokoll und Anfragesprache zum Zugriff auf „Verzeichnisse“
- interoperabel (theoretisch)
- Schreib- und Lesezugriff
- gedacht für viele kleine Anfragen
- nur begrenzt komplexe Anfragen möglich

Geschichte

- Standard X.500 sollte eine all-umfassende Verzeichnislösung bieten
- viel zu komplex...
- es gibt keine einzige vollständige Implementierung
- LDAP als vereinfachte Alternative zum X.500 DAP
- daher das Lightweight
- RfC 1487 im Jahr 1993
- einige Implementierungen: OpenLDAP, Microsoft Active Directory, Novell NDS, Oracle, Siemens, SGI, ...

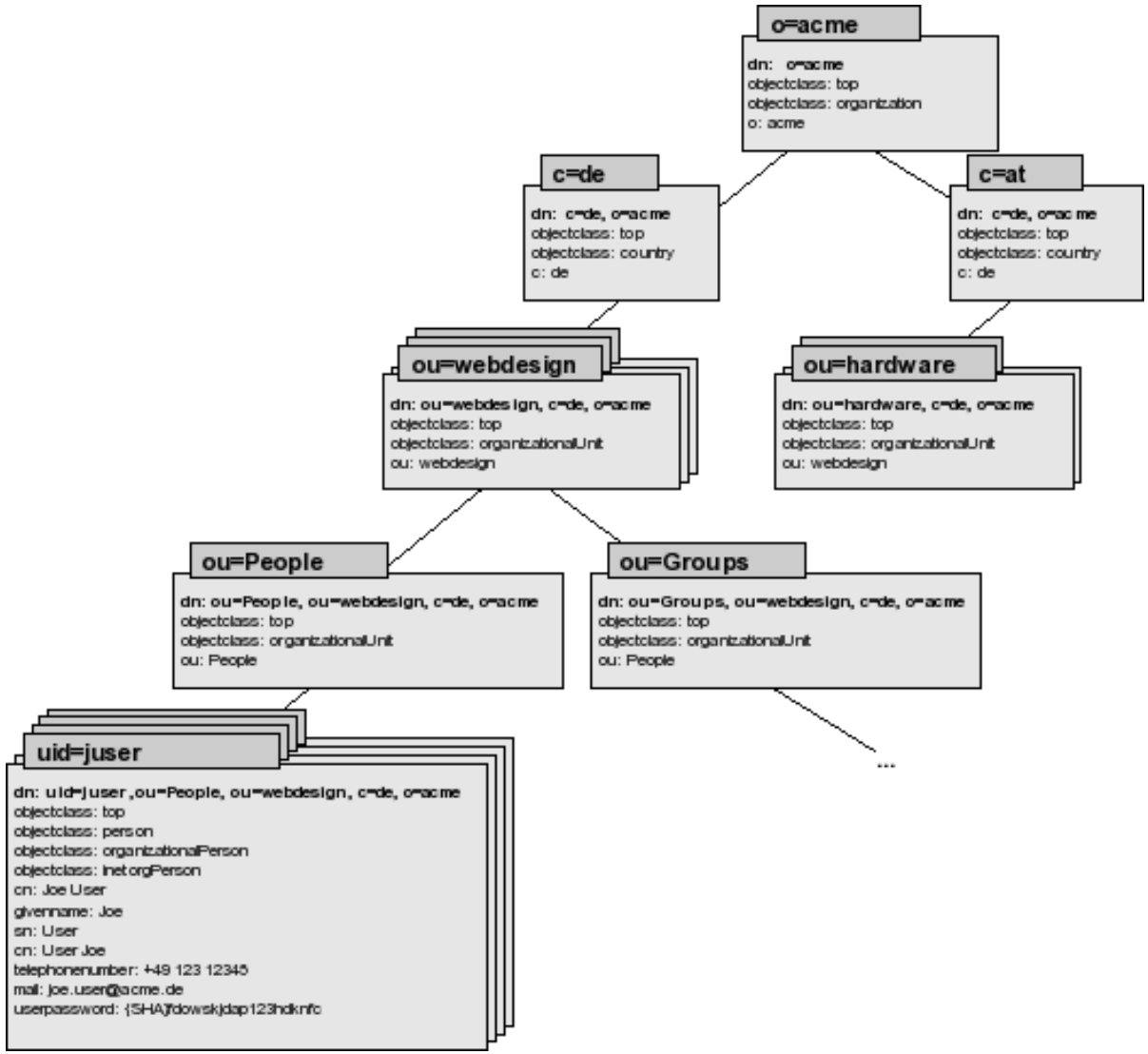
Das Verzeichnis dahinter

LDAP-Verzeichnis in einem Satz

Baumartig angeordnete Objekte verschiedener Klassen besitzen getypte Attribute.

+ baumartig, hierarisch
+ „objektorientiert“, Klassen mit Vererbung, Objekte
+ streng getypt
– keine Methoden, reine Datenobjekte
– Klassen sind keine Typen: nur primitive Typen

Das Verzeichnis dahinter



Bildquelle: Wikipedia

Verzeichnis

- ein Baum pro Server
  (evtl. auch auf mehrere Server verteilt)
- in der Theorie: ein Baum pro Organisation
- in der Ideologie: ein globaler Baum
- jeder Knoten ist ein Objekt
- kein Unterschied zwischen inneren Knoten und Blättern

Schema

- bildet den Baukasten für das Verzeichnis
- definiert:
  - Typen
  - (abstrakte) Klassen
  - Attribute
- wird beim Starten in den Server geladen
  (keine Modifikation zur Laufzeit)
- kann aber über LDAP ausgelesen werden
- macht keine Aussagen über Struktur des Baums und Typen der Objekte!
  ($\rightarrow$ Unterschied zum Schema relationaler Datenbanken!)

Schema-Definition

- Schema ist nicht Teil des Standards
- einige Schemata sind recht weitverbreitet (z.B. für Unix-Benutzer, Personen etc.)
- manchmal vorgegeben (z.B. beim Active Directory)
- Anwendungen liefern Schema-Definitionen, mit den Attributen die sie brauchen (z.B. Samba)
- kann für den Anfang als gegeben angenommen werden

Klassen

Eine Klasse definiert eine Reihe von Attributen, die ein Objekt dieser Klasse haben kann oder muss.

- strukturelle Klassen / Hilfsklassen
  (1 strukturelle pro Objekt,
  beliebig viele Hilfsklassen)
- Klassen können von anderen Klassen erben
- abstrakte Klassen
- schreibt Namen und Typen der Attribute vor
- vorgeschriebene / optionale Attribute

Verbreitete Klassen

top Superklasse von allen

dcObject, organization, organizationalUnit
zum Strukturieren (Attribute `dc`, `o` bzw. `ou`)

simpleSecurityObject
Oberklasse von allem was ein Passwort hat

person Person mit Name, Adresse etc.

posixAccount, shadowAccount, posixGroup
Unix-Benutzer-Verwaltung

Objekte

- hat eine strukturelle Klasse
  (Oberklassen dürfen ebenfalls explizit aufgeführt werden, müssen aber nicht)
- hat beliebig viele Hilfsklassen (auxiliary classes)
- Klassen eines Objekts sind im mehrwertigen Attribut `objectClass` gespeichert
- Objekt ist nur die Summe seiner Attribute

Objekt-Namen

- ein beliebiges Attribut wird als „**relative distinguished name**“ ausgewählt
- Form: `key=value` (z.B. `cn=Philipp Wendler`)
- keine zwei Objekte mit dem selben Vaterobjekt dürfen den selben RDN besitzen
- global eindeutiger Name („**distinguished name**“) wird aus den RDN aller Objekte bis zur Wurzel gebildet
- Beispiel für DN:

```
cn=Philipp Wendler, ou=People, dc=ieee,
        dc=students, dc=uni-passau, dc=de
```

Beispiel-Objekt

im LDIF (LDAP Data Interchange Format):

```
dn: cn=Philipp Wendler, ou=People, dc=ieee,
        dc=students, dc=uni-passau, dc=de
objectClass: person
cn: Philipp Wendler
sn: Wendler
```

Base DN

- Wieso diese seltsamen Namen mit `dc=...`?
- Jeder LDAP-Baum hat eine Wurzel, den „Base DN“ (Suffix für die DNs aller Objekte)
- theoretisch beliebig wählbar (genau wie die restliche Struktur)
- Wir erinnern uns: ein globaler Baum in der Ideologie
- Konvention: DNS-Name in der Form `dc=ieee, dc=students, dc=uni-passau, dc=de` (`dc` = domain component)
- sorgt für Konfliktfreiheit bei Merges // Datenaustausch
- veraltete Konvention: `ou=Branch, o=Company`

Das Protokoll

- spezifiziert nicht nur Anfragesprache (wie SQL), sondern auch das drumherum (Verbindungsaufbau, Authentifizierung etc.)
- aktuell ist Version 3 (Version 2 sollte nicht mehr benutzt werden)
- standardmäßig auf TCP-Port 389
- mit SSL auf TCP-Port 636
- TLS möglich
- lokal auch über Unix-Sockets

Authentifizierung

jetzt: **Authentifizierung beim LDAP-Server**

- Zugreifender User ist immer ein DN im Verzeichnis
- Authentifizierungsphase wird „bind“ genannt
- anonyme Authentifizierung
- „simple authentication“:
  Username und Passwort (im Klartext übertragen)
- SASL-Authentifizierung
  (mit allen möglichen Verfahren wie CRAM-MD5, Kerberos etc.)

Suchanfragen

folgende Dinge sind enthalten:

- **base DN** (Teil-)Baum für die Suche
- **scope** rekursiv ja/nein: `base`, `sub`, `one`, `children`
- **filter** Suchausdrücke über die Attributwerte
- **attributes** Liste der Attribute, die im Ergebnis auftauchen sollen

Komplexere Suchen sind nicht möglich!

Suchfilter

- vollständig geklammerte Prefix-Notation
- boolesche Operatoren: `&` `|` `!`
- die üblichen Vergleichsoperatoren
- genauer: RfC 4515
- Beispiel:

```
(&
    (objectClass=posixAccount)
    (uid=w*ndler)
    (!(disabled=true))
)
```

slapd

- der Server des OpenLDAP-Projekts
- bietet viele verschiedene Backends:
  BDB, **HDB**, DNS, LDAP, Ldif, passwd, perl, shell, SQL
- Rechte können fein konfiguriert werden:
  - pro Objekt/Teilbaum und Attribut
  - Rechte: none||auth||compare||search||read||write
- unterstützt Slave-Replikation

Clients

- Kommandozeile: `ldap-utils`:
  ldap(search||add||modify||delete||...)
- Skripte zum Accounts verwalten: `ldapscripts`
- graphisch: `gq` (Gnome), `jxplorer` (Java)
- Web: `phpldapadmin`
- selber schreiben mit `libldap`,
  Bindings in fast jeder Sprache

Benutzer-Authentifizierung mit LDAP

Standardverfahren:

1. Zugriff mit Suchanfrage, um zu einem Usernamen den DN zu ermitteln
   - je nach Suchfilter gleich Einschränkung der erlaubten User möglich
   - dieser Zugriff kann entweder anonym oder mit einem in der Konfiguration angegebenem Account erfolgen
2. Zugriff zur Überprüfung des Passworts
   - Anwendung versucht sich mit obigem DN und Passwort beim LDAP-Server einzuloggen
   - LDAP-Server überprüft Passwort
   - Passwort(-hash) verlässt Server nie!

Benutzer-Verzeichnis LDAP

Vorschlag / weitverbreitete Konfiguration:

- 2 Objekte: `ou=People` und `ou=Groups` mit Klasse `organizationalUnit`
- Gruppen wie im Beispiel:

```
dn: cn=admins, ou=Groups, <BASE DN>
objectClass: posixGroup
objectClass: top
cn: admins
gidNumber: 2100
memberUid: wendler
```

Accounts im LDAP-Verzeichnis

```
dn: uid=wender, ou=People, <BASE DN>
objectClass: person
objectClass: posixAccount
objectClass: top
sn: Wendler
cn: Philipp Wendler
uid: wendler
gidNumber: 2000
uidNumber: 2000
loginShell: /bin/bash
homeDirectory: /home/test
userPassword: MD5(geheim)
```

Statt `person` auch andere `*Person`-Klassen möglich je nach gewünschten Zusatzdaten.

User-Daten in Linux übernehmen

- `libnss-ldap` installieren
- `/etc/ldap.conf` anpassen (Server-URI, Base DN)
- `/etc/nsswitch.conf` (Auszug):

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

- fertig
- für Legacy-Clients: Skript `ldiftopasswd` aus OpenLDAP erzeugt passwd, group & shadow, die z.B. via NIS verteilt werden können

Authentifizierung: PAM

- `libpam-ldap` installieren
- `/etc/ldap.conf` ist ja schon angepasst
- `/etc/pam.d/common-(account|auth)` anpassen je nach Erfordernissen: `account required pam_ldap.so`

  `auth required pam_ldap.so`
- Ubuntu (Debian auch?) macht das automatisch

Authentifizierung: Apache 2.2

- Module `ldap` und `mod_authnz_ldap` laden
- Konfiguration:

```
AuthType Basic
AuthBasicProvider ldap
AuthLDAPGroupAttribute memberUID
AuthLDAPGroupAttributeIsDN off
AuthLDAPUrl ldapi:///ou=People,ou=Auth,dc=ieee,
dc=students,dc=uni-passau,dc=de?uid?one
Require valid-user
Require ldap-user wendler
Require ldap-group cn=admins, ou=Groups, ...
```

Samba

- braucht eigenes Schema im LDAP-Server eingebunden
- Accounts brauchen Klasse `sambaSamAccount`,
  Gruppen brauchen Klasse `sambaGroupMapping`
- Maschinen-Accounts mit Klasse `sambaSamAccount`
- smb.conf:

```
ldap admin dn = <Samba-User mit vollen Rechten>
ldap suffix = <Base DN>
ldap group suffix = ou=Groups
ldap user suffix = ou=People
ldap machine suffix = ou=Computers
ldap ssl = No
ldap passwd sync = Yes
add machine script = <Script> %u
```

Passwortänderungen

- z.B. über PAM, Weboberfläche etc.
- falls Samba benutzt wird: nur über Samba!
- Grund: Samba-eigene Passwort-Hashes müssen synchron gehalten werden
- Samba kann zum Glück normalen Hash auch setzen
- Tool `smbpasswd` unter Linux
- Trick: `/usr/local/bin/passwd`:

```
#!/bin/sh
smbpasswd -r <SAMBA Server> $@
```

- oder einfach Windows benutzen...