LDAP-Benutzung mit Java+Tomcat und Python+Zope

Holger Blasum, Andreas Meisl

6. April 2003

(Vortrag auf dem GUUG-Frühjahrsfachgespräch Bochum am 28. März 2003.)

Zusammenfassung

Wir berichten über Implementation und Installation von Klienten für eDirectory von Novell mit Python und Java via Plaintext und TLS/SSL sowie Performancemessungen. Note: You can find an English version of this paper as well as source code at
http://www.blasum.net/holger/wri/comp/net/7appl/ldap/bochum2003/.

1 Anwendungsfall

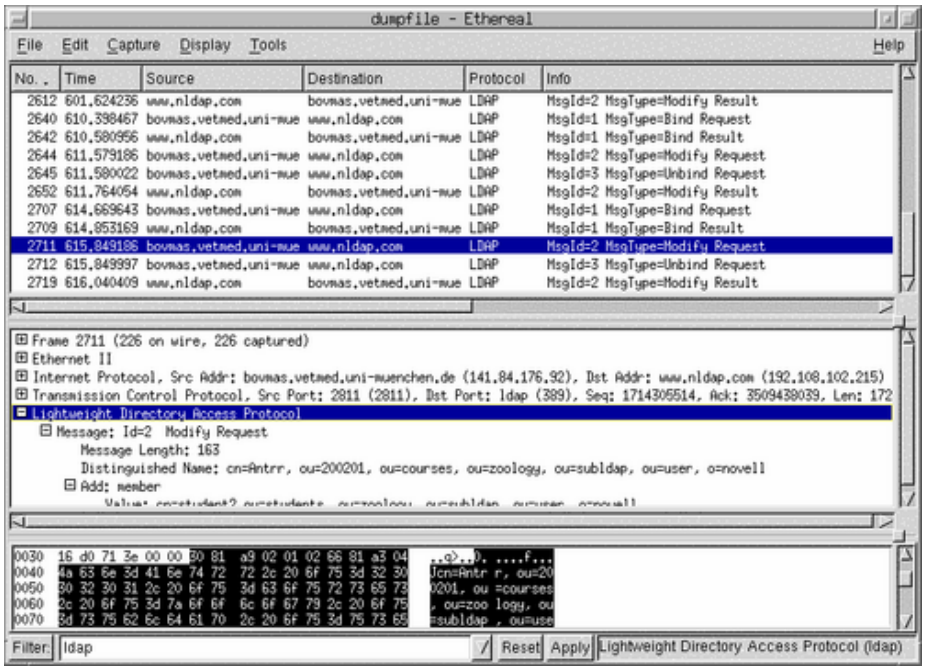
Die Tierärztliche Fakultät der Universität München verwendet Novell eDirectory als Verzeichnisdienst für Studentennutzerdaten[1]. Einmal im Semester tragen sich die Studierenden in Wahlpflichtfächer ein; dies geschieht seit Juli 2002 webbasiert[2].

2 Verzeichnisdienste und LDAP

Verzeichnisse sind etwa Telefonbücher oder das DNS-System, das Datenmodell ist hierarchisch. Von CCITT(ITU) / ISO wurde insbesondere 1988 der Standard X.500 für Verzeichnisdienste verabschiedet [Kla01, Chad96]. LDAP[3] ist eine ein abgespeckte Variante des Verzeichniszugriffsprotokolls (DAP) auf X.500-Verzeichnisse. Gedacht ursprünglich zur Replikation von Verzeichnisdaten, wird LDAP nun von populären Verzeichnisdiensten (also z.B. OpenLDAP[4], Novell eDirectory[5] , oder Microsoft Active Directory[6]) als Schnittstelle angeboten.

In diesen Anwendungen wird LDAP typischerweise über TCP verwendet (obwohl auch ein verbindungsloses UDP spezifiziert wurde[7]).

[1] Diese Nutzung gibt es auch an anderen bayerischen Universitäten, siehe z.B. AK Netz PC, http://www-lan.uni-regensburg.de/ak/netz/.
[2] http://www.vetmed.uni-muenchen.de/studium/wahlpflicht/kurse/index.html
[3] RFC 1777 (Version 2), 2251 (Version 3)
[4] http://www.openldap.org/
[5] zuvor: Novell Directory Services (NDS), http://www.novell.com/products/edirectory/
[6] http://www.microsoft.com/windows2000/technologies/directory/ad/default.asp
[7] RFC 1798



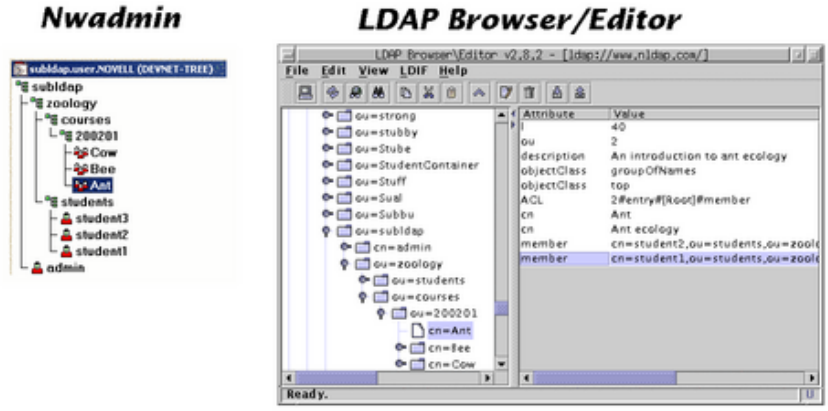
LDAP verwendet die ASN.1/BER-Kodierung. Ein tcpdump (*tcpdump -s 0 -w filename*) einer LDAP-Sitzung kann mit einem Paketanalysierer, der auch ASN.1/BER und LDAP-Anwendungsklassen versteht (zum Debuggen sehr hilfreich) wie Ethereal[8], betrachtet werden. Im Bild sehen wir z.B. eine *modify*-Anfrage von LDAP Version 3.

3 Ein Sandkasten

Sowohl OpenLDAP als auch eDirectory sind frei erhältlich (wie Rede im ersteren, wie Bier im zweiten Fall). Da wir jedoch nur Clients testen wollen, bietet sich hier auch der Test-Site von Novell an[9].

[8] http://www.ethereal.com/

[9] http://www.nldap.com/, in nldapaccount.txt finden sich auch genaue Zugangsdaten (*cn/userPassword*) zu dem derzeitigen Account. Wer sich bei nldap.com selbst einen Account aufsetzen will, folge den Links *eDirectory Account Adminstration->create NDS Container & its admin user*.



Betritt man diesen über ein GUI wie etwa dem *LDAP browser/editor*[10], oder Novells hauseigenen Werkzeugen (nwadmin, Console1), so kann man schnell die einfache Struktur der Sandbox überschauen. Es gibt Zoologiekurse, die von Studenten belegt werden können (Register/Unregister). Alternativ kann über die Kommandozeile ein (auch menschenlesbares) LDIF-File erzeugt werden:

```
ldapsearch -vLx -b "ou=subldap, ou=user,
o=novell" -h www.nldap.com
"(objectclass=*)" > subldap.ldif
```

4 Einfache LDAP-Klienten auf Kommandozeilenebene

4.1 Installation

Java: Wir haben die Blackdown ports[11] verwendet, vorzugsweise Java 1.4.1[12]. Als nächstes besorgen wir uns das Java Novell Developer Kit[13], wirklich gebraucht wird davon jedoch nur die Datei *ldap.jar*, die wir nach */usr/lib/j2sdk1.4.1/ jre/lib/ext/* installieren (oder man erweitere alternativ den Klassenpfad).

Python: Wir verwenden Python 2.1.3 und die python-ldap Bibliothek[14] in Release 2.0.0pre06 (diese geht dann nach */usr/lib/python2.1/site-packages*).

Server; OS: Zusätzlich zur beschriebenen Sandbox haben wird als LDAP Verzeichnis ein eDirectory 8.6.2 auf Netware 5.1[15] verwendet, Python und Java Middleware wurden auf Debian Woody GNU/Linux 2.4.19 i686 installiert.

[10] http://www.iit.edu/ gawojar/ldap/

[11] http://www.blackdown.org/

[12] Für Java 1.3 muss noch die Java Secure Socket Extension (JSSE) 1.0.2 in das Java-Erweiterungs-Verzeichnis installiert werden; JSSE ist in Java 1.4 bereits enthalten.

[13] http://developer.novell.com/ndk/jldapunx.htm, 15. November 2002, auch gelinkt von http://www.openldap.org/jldap/, wobei Dokumentation und Beispiele ebenfalls an dieser Stelle zu finden sind (ausführliche Dokumentation auch in [Har00]).

[14] http://python-ldap.sourceforge.net/

[15] eDirectory läuft auch unter AIX, Linux, Solaris, Windows.

4.2 Kursliste anschauen (anonym)

Wir wollen ein Programm schreiben, mit dem wir anonym den Inhalt des Kursverzeichnisses anschauen können, es gibt also folgende Aufgaben:

- LDAP-Verbindung initialisieren
- LDAP Suchanfrage an Server senden
- die Ergebnisliste *res* auslesen
- Verbindung schließen

Beginnen wir also mit einer Implementation in Python (ListCourses.py):

```
#!/usr/bin/python
import ldap
l=ldap.initialize("ldap://www.nldap.com:389")
res = l.search_s("ou=200201, ou=courses,
  ou=zoology, ou=subldap, ou=user,
  o=novell", ldap.SCOPE_ONELEVEL,
  "objectclass=*")
for r in res:
  print r[0]
c.unbind_s()
```

Dies ergibt dann (offensichtlich):

```
blasum@promitheas:$ ./ListCourses.py
cn=Ant,ou=courses,ou=zoology,ou=user,o=NOVELL
cn=Bee,ou=courses,ou=zoology,ou=user,o=NOVELL
cn=Cow,ou=courses,ou=zoology,ou=user,o=NOVELL
```

Das "_s" heisst hier synchrone Verbindung (Client wartet, bis die Daten da sind). In der JLDAP API wird die Unterscheidung synchron/asynchron einfach durch Methodenüberladung geregelt. Der Java-Client, der hier, wie auch in allen späteren Beispielen, ebenfalls synchron agiert, sieht so aus: (ListCourses.java):

```
import java.io.*;
import com.novell.ldap.*;
public class ListCourses {
  public static void main (String [] args)
    throws LDAPException {
    LDAPConnection c=new LDAPConnection ();
    c.connect ("www.nldap.com", 389);
    LDAPSearchResults res = c.search(
      "ou=200201, ou=courses, ou=zoology,
      ou=subldap, ou=user, o=novell",
      LDAPConnection.SCOPE_ONE, null, null,
      false);
    while (res.hasMore()) {
    System.out.println(res.next().getDN());
    c.disconnect();
```

```
    }
  }
}
```

Abschweifung: Wenn wir Python schreiben, es aber von Java interpretieren lassen wollen, gibt es dafür den Jython-Interpreter[16], ListCourses.jy sieht dann (als Mischung von Python und JLDAP) so aus:

```
from com.novell.ldap import *
c = LDAPConnection ()
c.connect ("www.nldap.com", 389)
res = c.search("ou=200201, ou=courses,
  ou=zoology, ou=subldap, ou=user,
  o=novell",
  LDAPConnection.SCOPE_ONE, None, None, 0)
while res.hasMore():
   print res.next().getDN()
c.disconnect()
```

Beobachtung: der Durchlauf von Java hat 1,0 Sekunden, von Python 0,14 Sekunden, von Jython 4,5 Sekunden gebraucht (+ ca. 0,5-1 Sekunden system response für die remote LDAP-Abfrage). Die Interfaces von JLDAP und python-ldap sind sehr ähnlich, wobei python-ldap etwas schlanker ist.

4.3 Einschreibungsdaten ändern (als Administrator)

Wir wollen mit einer authentifizierten Verbindung Einträge im Verzeichnis verändern. Folgende Schritte sind notwendig:

- LDAP-Verbindung initialisieren
- zum Server authentifizieren
- *modify*-Anfrage senden (bestehend aus einer Liste von Modifikationen)
- Verbindung schließen

Hier ist eine Python-Version (Register.py):

```
import ldap
import sys
if len (sys.argv) != 3:
  print "Usage: Register.py course student"
  sys.exit(1)
ldap.set_option (ldap.OPT_PROTOCOL_VERSION,
ldap.VERSION3)
c=ldap.initialize("ldap://www.nldap.com:389")
c.simple_bind_s("cn=admin, ou=subldap,
  ou=user, o=novell", "secret")
c.modify_s ("cn=" + sys.argv[1] + ", " +
  "ou=200201, ou=courses, ou=zoology,
```

[16] http://www.jython.org/

```
    ou=subldap, ou=user, o=novell",
    [(ldap.MOD_ADD, "member", "cn=" +
    sys.argv[2] + "," + "ou=students,
    ou=zoology, ou=subldap, ou=user,
    o=novell")])
c.unbind_s()
```

Die Umkehroperation Unregister.py sieht genauso aus, nur wurde hier *MOD_ADD* durch *MOD_DEL* ersetzt. Die Java-Versionen Register.java und Unregister.java unterscheiden sich ebenfalls nur durch *ADD* vs *DELETE*.

Beobachtungen: der Zeitverbrauch ist 1,2 Sekunden für JLDAP, 0,11 Sekunden für python-ldap. Wenn man das eDirectory verwendet, muss man explizit die LDAP Protokoll-Version auf 3 setzen (sonst scheitert *simple_bind_s* leise in python-ldap, JLDAP hat dagegen Version 3 als Default). Wenn *simple_bind_s* erfolgreich ist, aber die *modify*-Anfrage scheitert, gibt es eine Ausnahme (die wir hier nicht abfangen, um es einfach zu halten).

4.4 Eine Abstraktionsebene dazwischenquetschen: *ChangeManager*

Damit wir nicht direkt die Bibliotheksspezifika in die Anwendung verdrahten, haben wir eine Abstraktionsklasse (ChangeManager.java bzw ChangeManager.py) erstellt, die sich um die LDAP-Befehle schmiegt (unter der Präsentationslogik), z.B.:

```
Python 2.1.3 (#1, Sep 12 2002, 00:24:18)
[GCC 2.95.4 20011002 (Debian prerelease)] on linux2
>>> import ChangeManager
>>> c = ChangeManager.ChangeManager()
>>> courses = c.getCourseList('03')
```

Indem wir die gesamte direkte LDAP-Interaktion in den ChangeManager verbergen ist es dann auch noch möglich ein ganz anderes (z.B. relationales) Datenrepositorium einzuhängen.

Die Gesamtarchitektur hat also drei Ebenen (three-tiered): LDAP/eDirectory als Speicher, eine Mittelschicht (ChangeManager) für Logik und Zope DTML oder Java Servlets für die Präsentationsschicht. Wenn man das auf das Publisher/Subscriber Muster (etwa bei [Gam94]) abbilden will, so sind Subscriber/Observer Kommandozeilen oder Webinterface-Interaktion, der Publisher liegt im Verzeichnis.

5 Präsentationsschicht

5.1 Zope DTML

Naiver Ansatz: Mit Zope 2.5.1 (läuft auf Python 2.1) haben wir (recht brutal) unser Servlet Courses.py nach */usr/lib/zope/Extensions* kopiert (gesymlinkt), Infrastruktur (alle anderen Pythonklassen) nach */usr/lib/python2.1/* kopiert. In der Zope-Instanz gibt es nun eine DTML-Seite (index.html) die via

```
<dtml-var expr="doGet( term=REQUEST['term'],
course=REQUEST['course'], student=REQUEST['student'],
```

```
password=REQUEST['password'],
raction=REQUEST['raction'])">
```

eine Zope *External Method* aufruft (*doGet*), die einen Verweis auf die Methode *doGet* von Courses.py darstellt. Damit Änderungen im Code von Courses.py wirksam werden, muss Zope neu gestartet werden.

Einbau als Zope-Komponente (”Produkt”): In dem Produkt Courses lassen wir die LDAP-Verbindung nach Benutzung offen und öffnen sie nur dann neu, wenn sie nicht mehr besteht (self.c == None). Wir werden sehen, dass diese Technik (abgeschaut von dem Zope-Produkt LDAPUserFolder) uns TLS/SSL-Handshakes ersparen wird.

5.2 Tomcat Servlets

Der populärste Zugriff auf Java-Funktionalität über das Web sind Servlets oder JSPs. Dazu verwenden wir Tomcat 4.1.18. Bei der Installation und Entwicklung kann Tomcat recht einfach debug-freundlich gesetzt werden (via debug-levels und reloadable in der server.xml), dies ist in der Produktionsversion ggf zurückzusetzen. Je nach den Defaults des Java-Sicherheitsmodells kann es notwendig sein, *SocketPermissions* in policy files zu setzen (das war nötig bei dem Debian Tomcat-Paket, nicht aber bei einem direkten Download von jakarta.apache.org). Courses.java ist das komplette Servlet.

6 TLS/SSL

Nach diesem Proof-of-Concept will man das Ganze natürlich noch sicher machen: dabei gibt es einerseits die Kommunikation zwischen Webserver und Browser, andererseits die Kommunikation zwischen Webserver und LDAP-Datenspeicher (eDirectory-Server).

SSL wurde ursprünglich von Netscape entwickelt und spezifiert, TLS heißen die Weiterentwicklungen seit Übergabe an die gleichnamige IETF Working Group, die Protokolle werden deswegen meist in einem Atemzug genannt [Re01]. Praxis ist, dass die meisten Clients beim Handshake dem Server eine Vielzahl von Übertragungsoptionen anbieten, von denen sich der Server eine heraussucht. Die zunehmende Popularität von TLS-Protokollen sieht man daran, dass TLS Präferenz sowohl von OpenLDAP *slapd* 2.1.12 wie auch eDirectory 8.7 ist (während sich eDirectory 8.6.2 dagegen mit dem SSL-Protokoll begnügt).

Bei LDAP über TLS/SSL wird der Port 636 verwendet, es wird zunächst eine TLS/SSL-Verbindung aufgebaut; ist der Handshake erfolgreich, so wird über TLS/SSL *data transmissions* das LDAP-Protokoll abgewickelt. Genauso ist es natürlich beim HTTPS-Protokoll (Port 443)[17].

6.1 Webserver ↔ Browser

Obwohl sowohl Tomcat als auch Zope SSL-Unterstützung bieten, haben wir beide hinter das SSL Modul von Apache (*lib_modapache_ssl*) geschaltet, dies geht zum Beispiel über die *ProxyPass*-Anweisung (Vorteil: nur eine Stelle für Webserverzertifikate, das SSL Modul von Apache ist erprobt).

[17] Etwas irreführend ist es an dieser Stelle, dass es auch eine RFC 2830 für die TLS-Initialisierung *nach bereits erfolgtem* unverschlüsseltem Verbindungsaufbau über LDAP gibt, diese wird jedoch selten implementiert.

6.2 Webserver ↔ LDAP-Server

Generelles

Um einen Man-in-the-Middle-Attack zu vermeiden, wird das X.509-Zertifikat *der den öffentlichen Schlüssel des LDAP-Servers zertifiziert habenden Zertifizierungsstelle* (CA-Zertifikat; genau diese CA ist nämlich auf dem LDAP-Server-Zertifikat vermerkt, das beim zweiten Schritt des Handshakes übertragen wird) dem Client einmal manuell zugeführt[Mar02], dabei ist zu beachten:

- Im LDAP-Server-Zertifikat muss als *Subject name (cn)* der FQDN (voller Domänenname) des LDAP-Servers stehen (ist genauso wie bei den für HTTPS verwendeten Zertifikaten, aber auch hier haben die Novell-Tools das genausowenig voreingestellt wie die *openssl*-Tools).
- Das CA-Zertifikat (z.B. der *Institutional CA* bei einem eDirectory) muss also in eine Datei exportiert werden (das *b64*-Format bei Novell ist genau das *PEM*-Format bei den OpenLDAP-Tools), welche dann dem LDAP-Klienten in geeigenter Form zugänglich gemacht wird.

Zum Einblick speziell in TLS/SSL-Transaktionen ("wurde das nun verschlüsselt oder unverschlüsselt übertragen, was für ein Protokoll verwenden wir eigentlich gerade ?"), waren äußerst hilfreich: *ssldump*[18] (man sieht alle TLS/SSL-Transaktionen auf einem Interface vorbeilaufen), *ldapsearch -d 7*[19] (im hohen Debuglevel wie auch slapd sehr verbos). X.509 Zertifikate schaut man sich entweder im laufenden Fluss dieser Debugger oder statisch mit "*openssl x509 -noout -text -in cert.pem*" an.

Python-Client

Seit Version 2.0.0pre06 läuft python-ldap auch über TLS/SSL (bei den Protokollverhandlungen im Handshake einigte man sich dann unter in dieser Hinsicht im Default belassenen OpenSSL/eDirectory 8.6/8.7 konkret auf *RSA with 3DES EDE CBC SHA*). Mit

```
ldap.set_option (ldap.OPT_X_TLS_CACERTFILE,
  "nldapcacert.pem")
```

teilen wir dem Client mit, wo sich das CA-Zertifikat befindet.

Dann muss man nur noch die LDAP URL anpassen, also (RegisterSSL.py):

```
c = ldap.initialize ("ldaps://www.nldap.com:636")
```

statt zuvor (Register.py):

```
c = ldap.initialize ("ldap://www.nldap.com:389").
```

Damit das Ganze funktioniert und nicht etwa in einem

```
ldap.SERVER_DOWN: {'desc': "Can't contact LDAP server"}
```

[18] http://www.rtfm.com/ssldump/
[19] http://www.openldap.org/

endet, sind insbesondere zu beachten:

- Die auf dem Klienten installierten openldap-Bibliotheken müssen tatsächlich TLS/SSL unterstützen (unter Debian z.B. muss derzeit auf einem *stable*-System explizit *libldap2-tls* aus *testing* installiert sein).

Java-Client

Das CA-Zertifikat wird hier typischerweise mit dem *keytool* in einer Datei im Keystore-Format gespeichert, die dann von RegisterSSL.java in einem *java.security.Security.addProvider()*-Aufruf ausgelesen wird[20]: Bei den Protokollverhandlungen im Handshake scheint JSSE als Default (das auf Geschwindigkeit optimierte) *RSA with RC4 128 MD5* zu bevorzugen.

7 Diskussion

7.1 Zugänglichkeit von eDirectory via LDAP

Die LDAP-Schnittstelle von eDirectory [Ser02] scheint zu LDAPv3 User Schema (RFC 2256) weitgehend konform; eine (sinnvolle) Besonderheit war, dass das Attribut *userPassword* nur vergleichbar (Zugriff mit *compare*-Operationen), nicht jedoch auslesbar war[21], sonst sind wir in dieser Hinsicht auf keine Abweichungen gestossen.

Ein Unterschied der Bedienung von eDirectory via LDAP-Schnittstelle gegenüber Novells nativeren Schnittstellen (DirXML, NJCL-Javabibliotheken) liegt allerdings darin, dass LDAP als leichtgewichtiges Protokoll allgemein keine Transaktionen zulässt, eDirectory jedoch zum Teil nicht-atomare Operationen verwendet (z.B. ist Gruppenmitgliedschaft in eDirectory sowohl ein Attribut der Gruppe als auch ein Attribut des Users). Wir haben deshalb Gruppenmitgliedschaft (=Kursbelegung) nur als Attribut der Gruppe definiert, was auch ohne weiteres möglich ist.

Bei der Gruppenmitgliedschaft war auch eine Stelle erkenntlich, an der das Mapping des eDirectory-Schemas noch nicht ganz stabil war: in Version 8.6.1 war es das Attribut *member*, in Version 8.6.2 dagegen *uniqueMember* (beides RFC 2256 kompatibel).

7.2 Performance und Usability

Mit 400 gleichzeitigen Benutzern für 60 Sekunden unter *siege*[22] erreichen wir folgende Performance (399MHz PII, 796 bogomips, 128 MB RAM):

Apache direkt	25-35 Transaktionen/Sek.
Apache+Java+Tomcat (ohne TLS/SSL)	10-12 Transaktionen/Sek.
Apache+Zope+Python (ohne TLS/SSL)	10-12 Transaktionen/Sek.
Apache+Java+Tomcat (mit TLS/SSL, kein Pooling)	3,2 Transaktionen/Sek.
Apache+Zope+Python (mit TLS/SSL, kein Pooling)	3,2 Transaktionen/Sek.
Apache+Zope+Python (mit TLS/SSL, Pooling)	10-12 Transaktionen/Sek.

Der bei den Kommandozeilenklienten gemessene initiale Laufzeitüberhang für das Laden der Java Virtual Machine verliert sich also bei Verwendung einer aktuellen

[20] Bei der Installation folgen wir weitgehend den Hinweisen von Susan Perrin, 04. Juni 2002, auf novell.devsup.ldap_j.

[21] Andy Chalarambous, 16. Okt 2001, auf novell.support.ds.nds-general

[22] http://www.joedog.org/siege/index.shtml

Tomcat-Version (wie 4.1.18; sehr viel schlechter schnitt das Java+Tomcat-Duo bei Verwendung der älteren Tomcat Version 4.0.6 auf, hier gab es eine z.T. 2-8 mal schlechtere Performance).

Offensichtlich wird die Performance verbessert, wenn statt einem neuen Verbindungsaufbau für jede Anfrage über TLS/SSL zwischen Webserver und LDAP-Server eine einmal geöffnete Verbindung wiederverwendet wird. Dieses Pooling hatten wir im Zope-Produkt durch Wiederverwertung der Verbindungen erreicht.

In Angesicht der gleichgelagerten Performancedaten kann unsere Nebeneinanderstellung der Bibliotheks-Interfaces (api.txt) wohl hilfreich sein sowohl für Leute, die (wie wir) ihren Java-Prototypen nun in endlich Python implementieren wollen, wie auch andersherum (soll auch es auch geben): dass die Bibliotheken so ähnlich sind, ist in Anbetracht der in RFC 1823 expliziert spezifizierten C API kein Wunder, wobei der Preis in der feineren Typisierung in den JLDAP-Bibliotheken (Rückgabewerte beim Suchen nach Attributen sind z.B. Instanzen von *com.novell.ldap.AttributeSets* in JLDAP, *Hashes von Listen* in python-ldap) für uns erhöhten Aufwand an Typkonversionen im ChangeManager bedeutete.

7.3 TLS/SSL für Python und Zope

In einem früheren - diesem in mancher Hinsicht ähnlichen - Artikel [Ri01a] stellte die Nichtverfügbarkeit von LDAPv3 und damit LDAP über TLS/SSL noch ein Problem für Pythonprogrammierer dar. Dies ist nun gelöst, seit Michael Ströder im Sommer 2002 in die python-ldap-2.0.0pre06 LDADv3-Support eingebaut hat. Da z.B. die meisten Zope-Anwendungen für LDAP (wie etwa *LDAPUserFolder*, *CMFUserFolder*, *LDAPDirectoryManager*, *LDAPAdapter*, *ZopeLDAP*) python-ldap verwenden (eine Alternative wäre ldaptor), ist die Verwendung von TLS/SSL sofort möglich, sobald der Benutzer die openldap-Bibliotheken und python-ldap-Bibliothek entsprechend upgraded und konfiguriert (siehe auch [Chou03]) und für die Verbindungen benötigte CA-Zertifikate zur Verfügung stellt.

7.4 Verzeichnisse vs Datenbanken

In der Frage “Verzeichnis oder Datenbank?” ist unser Anwendungsfall ein Grenzfall: Da sich die Kurswahl gut in die bestehende LDAP-Infrastruktur einfügt (eine einfache Ergänzung von Gruppen und Gruppenattributen in einem bestehenden Userverzeichnis, das Schema wurde z.B. nicht erweitert) und nur leichtgewichtige Daten gespeichert werden, ist die Verwendung des Verzeichnisses hierfür wohl noch vertretbar (ähnlich wie bei [Ri01b]); dies muss nicht für kompliziertere eher relational strukturierte Anwendungsfälle gelten muss [Koehn02, Ma01]. Ein ausdrücklicher Gesichtspunkt bei der Entscheidung, diesen Anwendungsfall in Verzeichnissen zu implementieren, war es, Erfahrungen über die Zugänglichkeit von LDAP-Datenspeichern zu gewinnen, die auch bei typischeren Aufgaben (Authentifizierung, Zertifikatsmanagement etc.) von Nutzen sein können.

8 Danksagungen

Prof Klaus Osterkorn verdanken wir den Anwendungsfall; Christoph Wunder und Helmut Naughton verdanken wir einige “SSL-Sitzungen”.

9 Quellen, Addenda und Corrigenda

http://www.blasum.net/holger/wri/comp/net/7appl/ldap/bochum2003/talkback/.

Literatur

[Chad96] David Chadwick, Understanding X.500 - The Directory, http://www.isi.salford.ac.uk/staff/dwc/Version.Web/Contents.htm.

[Chou03] TC Chou, Setting LDAP, 2003, http://zope.slat.org/Members/tcchou/index_html/setting_ldap.

[Gam94] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns, Addison-Wesley 1994.

[Har00] Robert G. Harrison, Jim Sermersheim, and Steve Trottier, LDAP Developer's Guide, Novell Press 2000.

[Kla01] Norbert Klasen, Directory Services for Linux in comparison with Novell NDS and Microsoft Active Directory, Diploma Thesis, RWTH Aachen 2001, http://www.daasi.de/staff/norbert/thesis/html/ .

[Koehn02] Kristian Köhntopp, Directory Services vs. Relationale Datenbanken, 2002, http://www.koehntopp.de/kris/artikel/dir-vs-rel/.

[Ma01] Vikas Mahajan, Directory, Database, or Both?, http://www.ldapzone.com/general_interest.html#2.

[Mar02] Franck Martin, SSL Certificates HOWTO, http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/SSL-Certificates-HOWTO.html.

[Ra] Raphinou, ldap installation with ssl, http://www.raphinou.com/ldaps/LDAP-SSL.HOWTO.

[Re01] Erich Rescorla, SSL and TLS, Addison-Wesley 2001.

[Ri01a] Mike Richichi, How to Use Perl, Python, and PHP to Access NDS eDirectory 8.5 via LDAP, Novell Appnotes, http://developer.novell.com/research/appnotes/2001/august/05/a010805.pdf, http://developer.novell.com/research/appnotes/2001/august/05/apv.htm.

[Ri01b] Mike Richichi, ATTIC: a case study of directory-enabled course management, Proceedings of 29th ACM SIGUCCS Conf, 2001, pp. 258 - 261.

[Ser02] Jim Sermersheim, LDAP Schema for NDS, 2002, http://www.ietf.org/internet-drafts/draft-sermersheim-nds-ldap-schema-03.txt.