Orientation in Objects

JAAS Tutorial

Single Sign-On mit Kerberos

) Schulung)

) Beratung)

) Entwicklung)

) Artikel)

AUTOREN

Andreas Spall
Orientation in Objects GmbH

Dirk M. Sohn
Orientation in Objects GmbH

Josef Lüth
Orientation in Objects GmbH

Veröffentlicht am: 12.12.2002

ABSTRACT

Ein Verzeichnisdienst ermöglicht die zentrale Verwaltung von Ressourcen und ein Single Sign-On über verschiedene Systeme. Die dafür notwendigen Schnittstellen bietet der Java Authentication and Authorization Service (JAAS) und der Java Naming and Directory Service (JNDI).

Dieses Tutorial beschreibt, wie ein Single Sign-On mit JAAS und dem Kerberos Version 5 Authentifizierungsprotokoll umgesetzt werden kann. Als Verzeichnisdienst wurde der Active Directory Service (ADS) von Microsoft verwendet. Mit dem Beispiel wurde das folgende Szenario abgebildet

- Ein User authentifiziert sich über das Kerberos Version 5 Authentifizierungsprotokoll bei einer Applikation.
- Anschließend wird mittels LDAP aus dem Active Directory des Domänencontrollers die Benutzerrollen des Users ausgelesen.
- Als Beispiel für eine Autorisierung wird, abhängig von den Rollen des Users, die Umgebungsvariablen **java.home** und **user.home** ausgegeben.

Orientation in Objects GmbH

Weinheimer Str. 68
D-68309 Mannheim

Tel. +49 (0) 6 21 - 7 18 39 - 0
Fax +49 (0) 6 21 - 7 18 39 - 50

www.oio.de info@oio.de

JAVA AUTHENTICATION AND AUTHORIZATION SERVICE (JAAS)

Den Quellcode zum Beispiel können Sie als Zip-Archiv herunterladen:

OIO JAAS Demo

JAAS wurde mit der Version 1.4 in die Java 2 Standard Edition aufgenommen, um eine standardisierte Schnittstelle zur Benutzeranmeldung (Authentifizierung) und Berechtigung (Autorisierung) bereitzustellen.

Die pluggable-Fähigkeit von JAAS ermöglicht es, Änderungen an der Technologie für die Authentifizierung zu konfigurieren, ohne dass die zugehörige Anwendung modifiziert werden muss. Dies geschieht durch die Einbindung sogenannter *LoginModule*. Durch die stackable-Fähigkeit ist es möglich, mehrere Authentifizierungsmodule aufzuschichten und so einen Single Sign-On gegen mehrere Systeme zu realisieren.

Hintergrund

JAAS stellt eine Java-Implementierung des aus der Unix-Welt bekannten PAM (Pluggable Authentication Module) dar. PAM signalisiert die Möglichkeit zur Verwendung von unterschiedlichen Modulen zur Authentifizierung (Authentication) und eine hohe Konfigurierbarkeit (Pluggable) der Gesamtlösung durch den System-Administrator in einem Unix-System an.

MODULE FÜR DIE AUTHENTIFIZIERUNG (LOGINMODULE)

In der J2SE Version 1.4 sind bereits einige LoginModule für Unix, NT, JNDI und Kerberos enthalten.

In diesem Artikel werden wir ein Modul für Kerberos einsetzen. Das Modul "Krb5LoginModule" bietet die Möglichkeit, Benutzer über deren Windows2000 oder Solaris-Systemlogin zu authentifizieren.

AUTHENTIFIZIERUNG

Bei der Authentifizierung muß der Benutzer einen Beweis (z.B. Username und Password) für seine Identität erbringen.

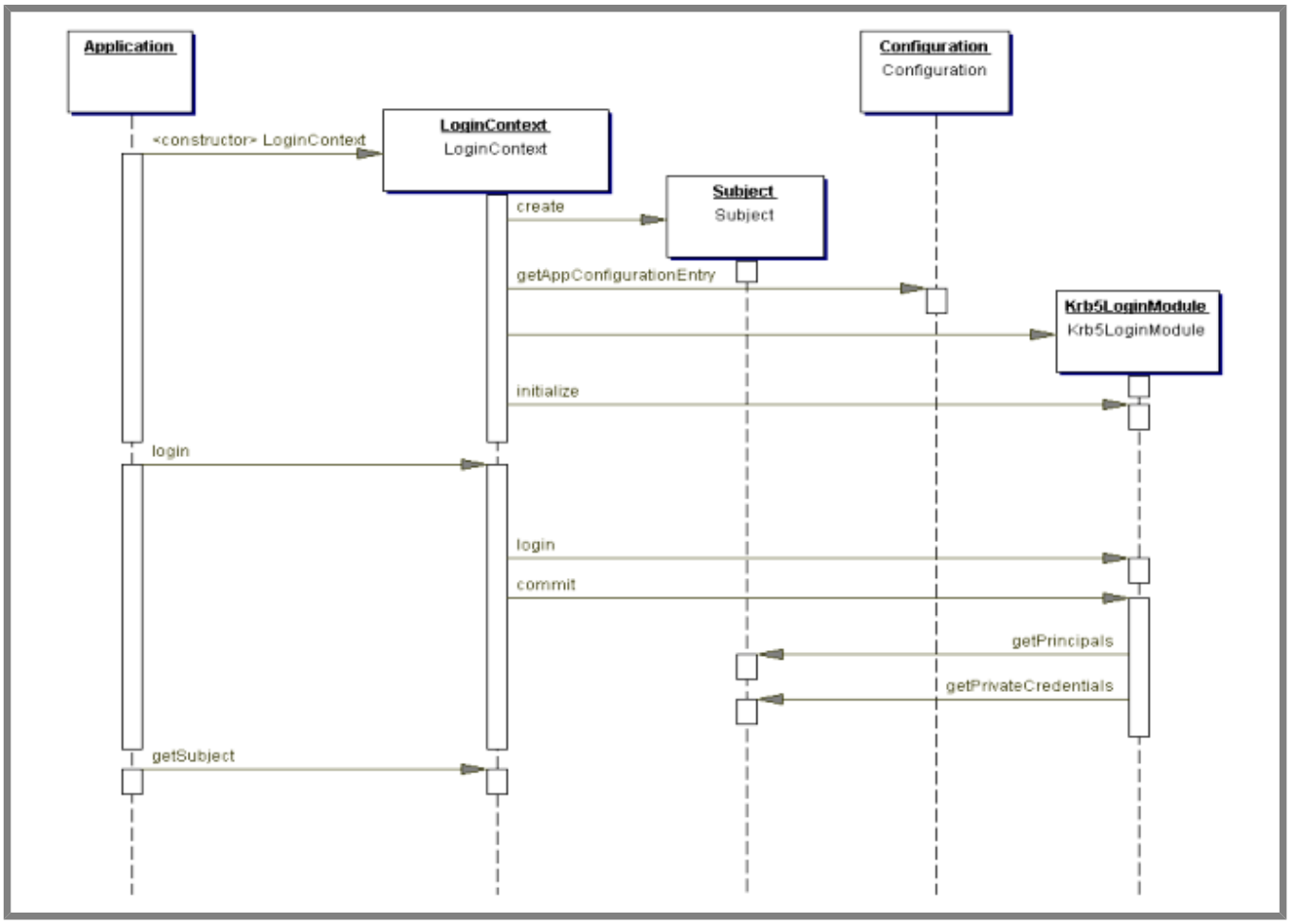

Abbildung 1: Ablauf einer Authentifizierung

JAAS-basierte Anwendungen starten die Authentifizierung durch Instanziierung eines LoginContextes. Bei der Instanziierung kann ein Parameter übergeben werden, der auf eine JAAS-Konfiguration verweist. Listing 2 zeigt die Konfigurationsdatei oiojaas.conf mit der Konfiguration LoginJass. In der JAAS-Konfiguration werden die verwendeten LoginModule festgelegt. Der virtuellen Maschine kann diese Datei beim Start als Parameter übergeben werden.

Parameter für die JVM:

```
$ java ... -Djava.security.auth.login.config=oiojaas.conf
```

Beispiel 1: Angabe der Konfigurationsdatei für JAAS

```
LoginJaas {
   com.sun.security.auth.module.Krb5LoginModule required
   debug=false
   useTicketCache=false;
};
```

Beispiel 2: Die Konfigurationsdatei oiojaas.conf

Hintergrund

Beim Login eines Users auf einem Windows 2000 Rechner (Client) wird innerhalb einer Windows 2000 Domäne von einem Kerberos Server, den sogenannten Key Distrution Center (KDC), ein Ticket erstellt. Der Client speichert dieses Ticket in einem sichern Bereich des RAM.

Über `useTicketCache=true` kann man angeben, ob das Kerberos LoginModul zunächst versuchen soll, den Benutzer über dessen gespeichertes Ticket zu identifizieren (Ticket-Granting Ticket).

Ist dies nicht vohanden oder wird die Variable `useTicketCache` auf `false` gesetzt, erfragt sich das Kerberos LoginModul vom User dessen Username und Password und lässt sich vom Key Distribution Center (KDC) ein Session Ticket erstellen.

Nach der Instanziierung des LoginContextes ermittelt dieser die Loginkonfiguration (siehe *getAppConfigurationEntry* in der Abbildung 1) und setzt den für die Kommunikation mit dem User möglicherweise notwendigen CallbackHandler. Dieser muß das Interface `javax.security.auth.callback.CallbackHandler()` implementieren. Die Aufgabe des CallbackHandler ist es, die erforderlichen Login-Informationen vom User zu erfragen. Im Beispiel wird der TextCallbackHandler von Sun verwendet.

```
LoginContext lc;
try {
  lc = new LoginContext(
    "LoginJaas",
    new TextCallbackHandler());
  } catch (LoginException le) {
     le.printStackTrace();
     System.exit(-1);
  } catch (SecurityException se) {
     se.printStackTrace();
     System.exit(-1);
  }
```

Beispiel 3: Initialisierung des LoginContext

DAS SUBJECT

Alle benötigten Benutzerinformationen wie Loginnamen, Gruppen, Tickets und Passwörter werden in JAAS in einem sogenannten *Subject* hinterlegt. Die im Subject enthaltenen Informationen dienen später beispielsweise dazu, eine Anmeldung gegen ein Backendsystem durchzuführen.

Wie gelangen die Informationen aber ins Subject?

Für das Füllen des Subjects sind die LoginModule zuständig. Der LoginContext erzeugt die LoginModule und übergibt diesen das Subject. Jedes LoginModule kann der Reihe nach im Subject die Informationen hinterlegen, die für das jeweilige System benötigt werden. Beispielsweise Benutzername und Passwort für ein ERP System.

Abbildung 1 zeigt den Ablauf im Detail. Zunächst erzeugt der *LoginContext* durch Aufruf des Default-Konstruktors von jedem benötigten *LoginModule* eine Instanz. Die LoginModule werden mit einem darauf folgenden Aufruf von `initialize` initialisiert. Dei der Initialisierung wird den Login-Modulen unter anderem das Subject (enthält alle benötigten bzw. ermittelten Benutzerinformationen) bekannt gegeben.

Durch Aufruf der Methode `login()` wird die Authentifizierung unter Verwendung der konfigurierten Login-Module gestartet.

Nach einer erfolgreichen Anmeldung kann über den LoginContext das Subject ermittelt werden, welches die ermittelten Benutzerinformationen enthält. Dies sind der Username in Form von einem *KeberosPrincipal* und das Ticket in Form von einem *KerberosTicket*.

Misslungene Anmeldeversuche werden durch eine LoginException quittiert.

ZUSÄTZLICHE USER-ROLLEN DEM SUBJECT HINZUFÜGEN

Aus dem LoginContext entnehmen wir das für den User ermittelte Subject als mySubject.

```
setMySubject(lc.getSubject());
```

Aus diesem Subject entnehmen wir den ersten Principal. Da wir das *Krb5LoginModule* verwenden enthält unser Subject den vom Kerberos LoginModule erstellten Principal. Dieses beinhaltet als Wert den *userPrincipalName* des authentifizierten Users.

```
loginname@domain
```

Wir lassen uns vom Active Directory Service über das Lightweight Directory Access Protocol (LDAP) die mit diesem *userPrincipalName* verbunden Rollen geben (siehe memberOf in GetRole.java).

Das LDAP liefert uns die Rollen wie folgt:

```
CN=ROLE1,CN=Users,DC=Domain
```

Wir benötigen aber nur den ersten *Common Name (CN)* der ermittelten Rolle und lösen uns den ersten CN (z.B. ROLE1) als *roleName* aus. Diesen fügen wir dem mySubject als neuen `JaasRolePrincipal()` hinzu.

Warum benötigen wir den ersten CN?

Damit wir über die User-Rollen die Ausführungsberechtigungen steuern können.

```
JaasRolePrincipal jrp = new JaasRolePrincipal(roleName);
getMySubject().getPrincipals().add(jrp);
```

AUTORISIERUNG

Bei der Autorisierung wird überprüft, ob der Benutzer die benötigten Zugriffsrechte zum Lesen, Schreiben oder Ausführen bestimmter Ressourcen besitzt.

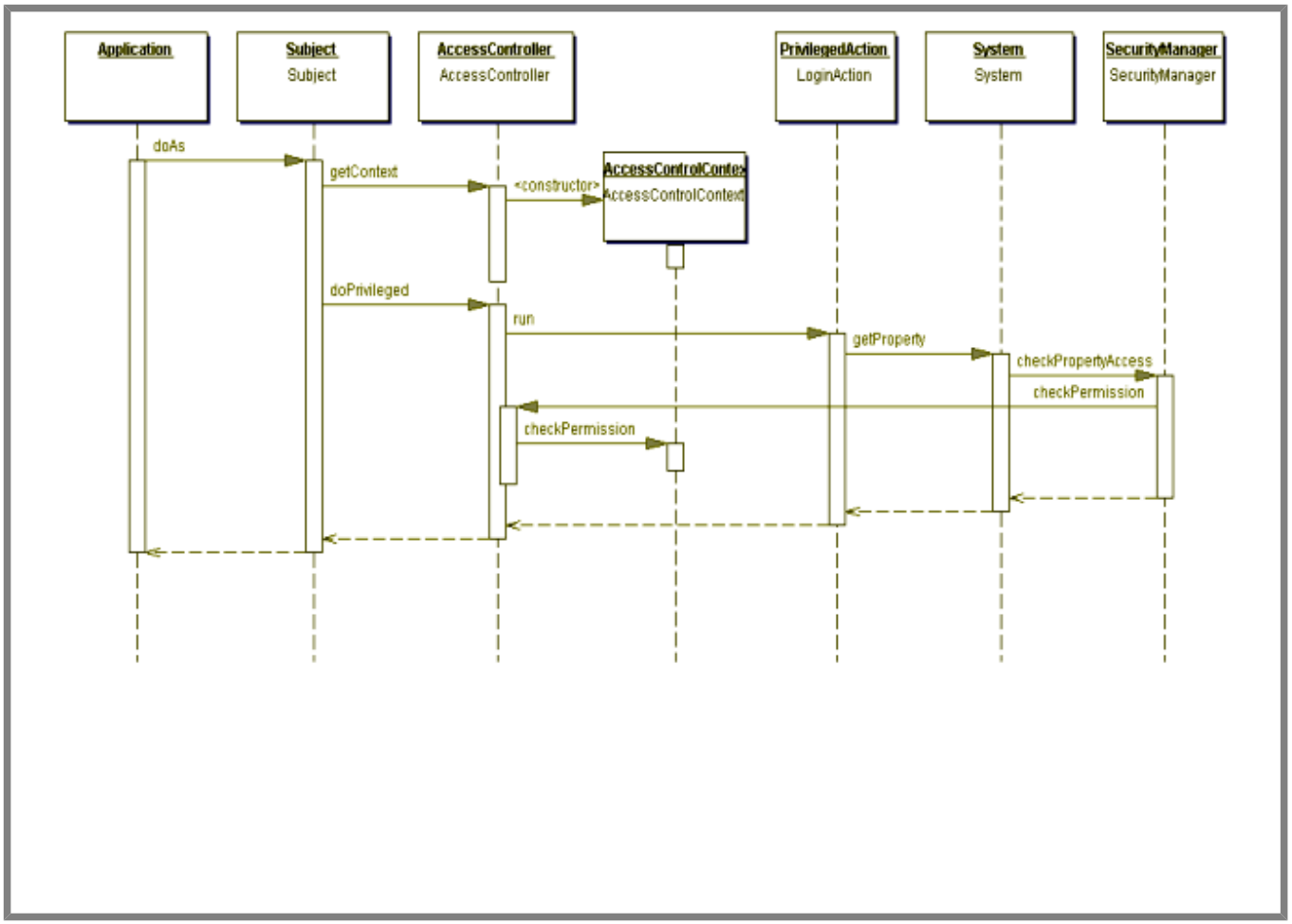

Abbildung 2: Ablauf einer Autorisierung

Die während der Authentifizierung ermittelten Benutzerinformationen und hinzugefügten Rollen können zur Realisierung von anwendungsspezifischen Zugangsbeschränkungen oder zur Weitergabe an ein Backend-System genutzt werden.

Mit JAAS können darüber hinaus diese Informationen auch für die Vergabe von Ausführungsberechtigungen genutzt werden, deren Einhaltung durch die Java-Runtime sichergestellt wird.

Für die Einhaltung der Berechtigungen in der Java-Runtime ist ein *SecurityManager* verantwortlich. Dieser wird durch Start der JVM mit dem Parameter `-Djava.security.manager` oder durch explizites Setzen des *SecurityManagers* in der Anwendung mittels `System.setSecurityManager(new SecurityManager());` geladen. Der *SecurityManager* kann über eine Policydatei konfiguriert werden.

```
$ java ... -Djava.security.manager -Djava.security.policy=oiojaas.policy
```

```
oiojaas.policy:
....
grant codebase "file:./LoginAction.jar",
Principal principal.JaasRolePrincipal "yourRole" {
  permission java.util.PropertyPermission "java.home", "read";
  ...
};
...
```

Beispiel 4: Konfiguration über Policydatei

Innerhalb der Policydatei können für jedes jar und jeden möglichen User (oder Rolle) die erlaubten Aktionen festgelegt werden.

Damit der *SecurityManager* die Möglichkeit hat, die Einhaltung der Ausführungsberechtigungen zu überprüfen, muss ihm bekannt sein, welcher Benutzer ein Programm ausführt. Zu diesem Zweck wird das im Rahmen des Login-Prozesses ermittelte Subject mit dem AccessControlContext des laufenden Threads assoziiert. Hierfür stellt die Klasse Subjekt über ihre statischen Methoden `doAs` und `doAsPrivileged` verschiedene Varianten zur Verfügung.

```
...
PrivilegedAction action = new LoginAction();
Subject.doAs(getMySubject, action);
...
```

Beispiel 5: Statischen Methoden doAs und doAsPrivileged

Eine derart aufgerufene Aktion wird nur ausgeführt, wenn der User die in der Policy spezifizierten Bedingungen erfüllt. Ist dies nicht der Fall wirft die JVM zur Laufzeit eine `java.security.AccessControlException`.

Geschützte Aktionen müssen das Interface PrivilegedAction (`java.security.PrivilegedAction`) implementieren.

LOGOUT

Principals und Credentials, die im Rahmen der Authentifizierung mit einem Subject verknüpft wurden, werden nach dem Aufruf der `logout`-Methode wieder gelöscht.

NEU IN J2SDK 1.4.1

Seit der Version 1.4.1 des J2SDK gibt es drei neue Sicherheits-Tools. Die Tools `kinit`, `klist`und `ktab` unterstützen den User beim Erhalten, Auflisten und Verwalten von Kerberos-Tickets.

Tool	Beschreibung
kinit	Kerberos TGTs (Ticket-Granting Ticket) Beziehen und Cachen
klist	Teile des gecachten TGT ausgeben
ktab	Prinzipalnamen und Passwörter in einer lokalen Schlüsseltabelle speichern. Mit der *key table* ist es möglich, dass sich Services beim Key Distribution Center (KDC) autonom anmelden können

Tabelle 1: Sicherheits-Tools

FAZIT

Bei der Verwendung von JAAS mit dem Kerberos LoginModule kann man die Authentifizierungsdaten im zentralen Active Directory Service belassen und eine Authentifizierung über das Key Distribution Center durchführen. Über LDAP ist es möglich, die Rollen des Users auszulesen und seinem Subject hinzuzufügen. Über Domänen-Benutzer oder den Domänen-Rollen können Ausführungsberechtigungen erteilt werden.

Nach vielen herstellerabhängigen Lösungen kann mit JAAS jetzt auf standardisierte Weise ein Single-Sign-On realisiert werden. JAAS hat nicht nur in der Java Standard Edition sondern seit der Version 1.3 auch in der Java Enterprise Edition eine zentrale Aufgabe bei Realisierung von sicheren unternehmensweiten Lösungen.

Wir hoffen, die Einführung hat Ihnen gefallen und freuen uns über Ihr Feedback.

KONFIGURATION DES BEISPIELS

Zur Ausführung des Beispiels sind verschiedene Anpassungen vorzunehmen.

Datei: oiojaas.conf

Wenn Sie möchten, dass das Kerberos Login-Modul zunächst im Sytem Cache sucht, setzen sie `useTicketCache=true`.

Wenn Sie möchten, dass der User sofort sein Loginname und Password eingeben muss, setzen sie `useTicketCache=false`.

Datei: oiojaas.policy

Ersetzten Sie die `yourRole` Parameter durch die in ihrer Windows 2000 Domäne verwendeten Rollennamen (memberOf).

Datei: GetRole.java

Ändern Sie folgende Parameter (fragen Sie notfalls Ihren Windows-Administrator):

Parameter	Beschreibung	Beispiel
yourPrincipal	Der Principal, welcher von GetRole verwendet wird um vom LDAP die benötigten Userinformationen zu erfragen. Legen Sie besser einen neuen User in ihrer Domäne an, da das Password des Principals der Klasse angegeben werden muss	user@develop.yourComp.com
yourCredential	Das Password des Principals	
yourServer	Server (Active Directory Services), welcher über LDAP angesprochen wird	192.168.1.1
yourPort	Port auf welchem das LDAP des Servers reagiert	389
yourDomain	Der Name Ihrer Domäne (siehe auch `run.bat` weiter unten)	dc=develop,dc=yourComp,dc=com

Tabelle 2: Parameter von GetRole.java

run.bat

Ändern Sie folgende Parameter (fragen Sie notfalls Ihren Administrator):

Parameter	Beschreibung	Beispiel
yourRealm	Der Name Ihrer Domäne (können Sie über Einstellungen => Systemsteuerung => System => Netzwerkidentifikation (Domäne:...) in Erfahrung bringen)	DEVELOP.YOURCOMP.COM
yourKDC	Der Name des Key Distrution Centers (oft ist dies der Ihnen bekannte Domänencontroller ihrer Domäne)	YOURSERVERNAME.DEVELOP.YOURCOMP.COM

Tabelle 3: Parameter der run.bat

DAS BEISPIEL AUSFÜHREN

Mit folgenden Kommandos kann das Beispiel übersetzt und ausgeführt werden:

- Öffnen Sie die Eingabeaufforderung
- Starten Sie buildLoginAction.bat
- Starten Sie buildLoginJaas.bat
- Starten Sie run.bat

VERWENDETE BEGRIFFE

Authentifizierung

- Der Benutzer erbringt einen Beweis für seine Identität z.B. durch Username und Password, oder über eine Chipkarte

Autorisation

- Überprüfung der Zugriffsrechte des Benutzers
- Darf er das Lesen, Schreiben oder Ausführen was er möchte?

Subject

- Repräsentiert eine Person oder ein System

Principal

- Repräsentiert eine Identität einer Person oder eines Systems in der Form: `username@DEVELOP.YOURCOMP.COM` (bei einem KerberosPrincipal)

public Credential

- Öffentliche Schlüssel

- Zertifikate

private Credential

- Enthält ein KerberosTicket, welches sich auf den KerberosPrincipal bezieht
- Private Schlüssel und Passwörter

REFERENZEN

- Making Login Services Independent of Authentication Technologies
  Samar, Vipin , Sun Microsystems; Lai, Charlie , Sun Microsystems
  (http://java.sun.com/security/jaas/doc/pam.html)
- Security
  (http://java.sun.com/j2se/1.4.1/docs/guide/security/)
- JavaTM 2 Platform Security Architecture
  (http://java.sun.com/j2se/1.4.1/docs/guide/security/spec/security-spec.doc.html)
- JavaTM Authentication and Authorization Service (JAAS)
  (http://java.sun.com/products/jaas/)
- Standardmäßige Active Directory-Attribute im Windows 2000-Schema
  (http://support.microsoft.com/kb/257218/)
- USER AUTHENTICATION AND AUTHORIZATION IN THE JAVA(TM) PLATFORM
  (http://java.sun.com/security/jaas/doc/acsac.html)
- PAM - Pluggable Authentication Modules
  (http://wwwbs.informatik.htw-dresden.de/svortrag/ai96/Baehr/)
- Kerberos Papers and Documentation
  (http://web.mit.edu/kerberos/www/papers.html)
- Windows 2000 Kerberos Authentication
  (http://www.microsoft.com/windows2000/docs/kerberos.doc)
- Single Sign-On in Windows 2000 Networks
  (http://www.microsoft.com/windows2000/docs/SSO.doc)