Single-Sign-On mit Java und Kerberos

Michael Wiesner, SOFTCON IT-Service GmbH

jax 2006
Konferenz für Java™, XML, Web Services

Über mich

- Softwareentwickler und Sicherheitsexperte bei der Firma SOFTCON
- Projekte: Enterprise Software, Webportale, Sicherheitslösungen, ...
- Trainings und Vorträge über Security
- michael.wiesner@softcon.de

Agenda

- Single Sign-On
- Kerberos Grundlagen
- Kerberos und Java
- Beispiele
- Ausblick

- Single Sign-On
  - Was bedeutet Single Sign-On?
  - Wieso brauchen wir das?
  - Lösungsansätze
- Kerberos Grundlagen
- Kerberos und Java
- Beispiele
- Ausblick

Identitätsmanagement Begriffe

- Authentifizierung (Authentication)
  - Identitätsfeststellung
  - z.B. Personalausweis, Pass, ...

- Autorisierung (Authorization)
  - Zuweisung von Rechten
  - z.B. Mitgliedsausweis, Access Control Lists

Was bedeutet Single Sign-On?

- Benutzerauthentifizierung nur einmal
- Anschließend Zugriff auf (alle) Systeme
- Führt i.d.R. keine Autorisierung durch
- Eine Stufe darunter: Single Credential
  - Ein Benutzername/Kennwort für alle Systeme

Wieso brauchen wir das?

- Produktivitätssteigerung
  - Arbeitsfluss wird nicht gestört
  - Einfachere Administration
- Erhöhte Sicherheit, denn ohne SSO...
  - wird oft das gleiche Kennwort benutzt
  - wird das Kennwort auf mehreren Systemen gespeichert
  - geht das Kennwort mehrmals über das Netzwerk

Wieso brauchen wir das?

- Nachteile
  - Bei einer Kompromittierung erhält der Angreifer Vollzugriff.
  - Die Anwendung ist an das SSO-System gebunden und läuft nicht „standalone“

Lösungsansätze

- „Lokale Lösungen“
  - Benutzername/Kennwort wird lokal gespeichert (z.B. durch Browser, Outlook, ...)
- Portale
  - Einloggen bei einem zentralen Portal
  - Portal steuert weitere Zugriffe

Lösungsansätze

- Ticket Systeme
  - Zentraler Authentifizierungsserver
  - Ticket mit Identitätsnachweis wird bei der ersten Anmeldung erstellt
  - Authentifizierung anschließend über dieses Ticket

- Single Sign-On
- Kerberos Grundlagen
  - Was ist Kerberos?
  - Funktionsweise
  - Vor-/Nachteile
- Kerberos und Java
- Beispiele
- Ausblick

Was ist Kerberos?

Kerberos...

- ... ist ein verteilter Authentifizierungsdienst.
- ... ist konzipiert für offene und unsichere Netze.
- ... ermöglicht Single Sign-On.
- ... ist Plattform- und Systemunabhängig.
- ... verwendet symmetrische Verschlüsselung.
- ... ist in RFC 1510 beschrieben.

Principals & Realms

- Kerberos Principals bestehen aus:

  component[/component]...@REALM

  z.B.: mike@SECPOD.DE
- Der Realm gibt die Authentifizierungs-Domäne an.
- Component kann aber auch einen Service sein:

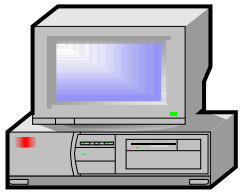
  z.B.: java/server.secpod.de@SECPOD.DE

KDC

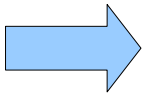
- KDC = Key Distribution Center

- Besteht aus:
  - Principal Database (z.B. Active Directory)
    - Speichert Principals und Keys
  - Authentication Server (AS)
    - Erstellt das Ticket Granting Ticket (TGT)
  - Ticket Granting Server (TGS)
    - Erstellt Service Tickets für ein TGT

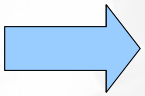
Authentication Server

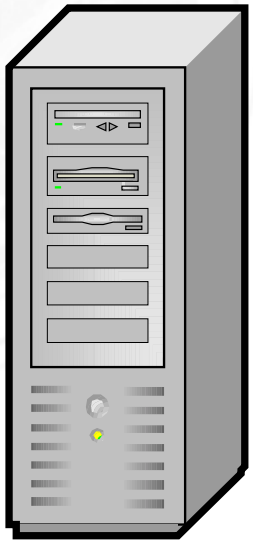


Client



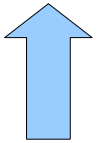
Client Principal
Client timestamp
krbgt principal name
Requested lifetime





AS session key
krbtgt principal name
Ticket lifetime

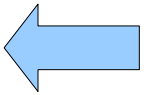
User's key



AS session key
Client principal
Ticket lifetime
KDC timestamp
Client IP address

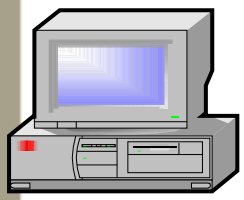
TGT

TGS key

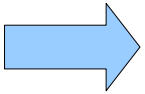


Authentication Server

Ticket Granting Server



Client



Service principal

AS session key

Client principal

Ticket lifetime

KDC timestamp

Client IP address

TGT

TGS key

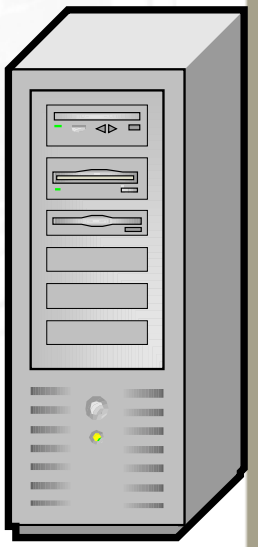
Client principal

Client timestamp

Authenticator

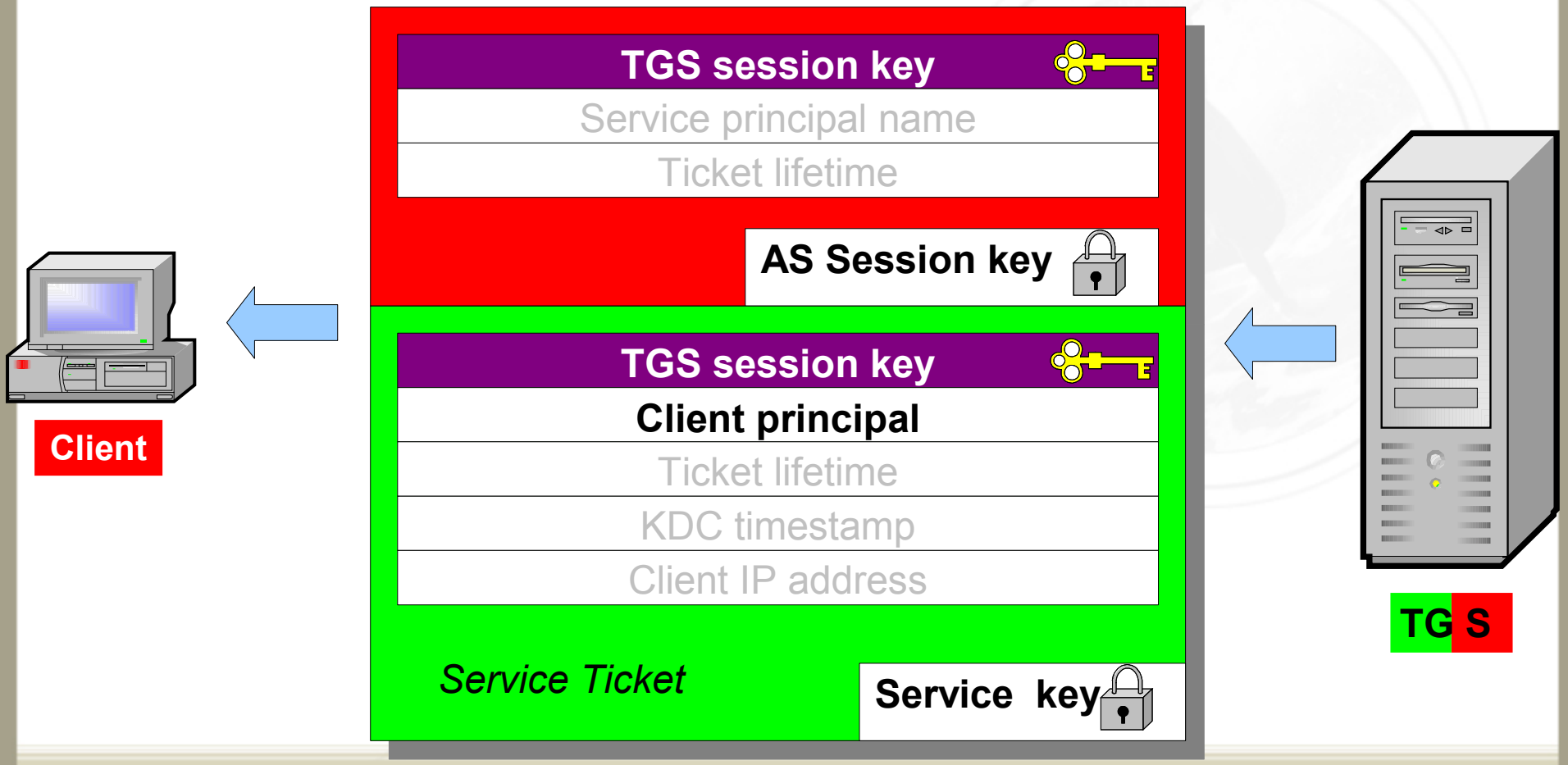
AS Session key

Requested lifetime

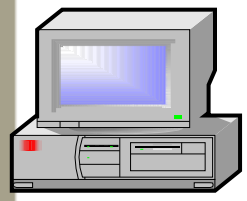


TGS

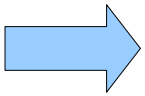
Ticket Granting Server



Service



Client



Service Specific Information

TGS session key

Client principal

Ticket lifetime

KDC timestamp

Client IP address

Service Ticket

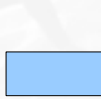
Service key

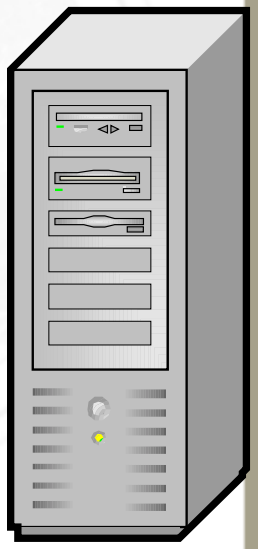
Client principal

Client timestamp

Authenticator

TGS Session key





Service

weitere Protokolle

- Generic Security Services API (GSSAPI):
  - generisches Interface zur Unterstützung von „strong Authentication“, wie z.B. Kerberos
- Security Support Provider Interface (SSPI):
  - Microsoft Pendant zu GSSAPI
- Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO):
  - Übermittlung eines Kerberos-Tickets zwischen Browser und Webserver

Vorteile von Kerberos

- Sichere, gegenseitige Authentifizierung aller Beteiligten
- Geringe Belastung des KDCs
- Autorisierung wird dem Service überlassen
- Plattform- und Systemunabhängig

Nachteile von Kerberos

- Keys müssen zu den Services verteilt werden
- Uhrzeiten müssen synchronisiert sein
- nicht ohne weiteres mit Firewalls einsetzbar (NAT)
- mangelnde Unterstützung bei den Clients (derzeit)
- verschiedene APIs bei Unix und Windows

Agenda

- Single Sign-On
- Kerberos Grundlagen
- Kerberos und Java
  - JAAS
  - Java GSS-API
- Beispiele
- Ausblick

JAAS

- JAAS = Java Authentication and Authorization Service
- Java Standard API zur Authentifizierung und Autorisierung (seit JDK1.4)
- Konfiguration und Providerauswahl über eine Konfigurationsdatei
- Enthält diverse Login-Module (JNDI, Keystore, Kerberos, ...), ist aber erweiterbar.

JAAS und Kerberos

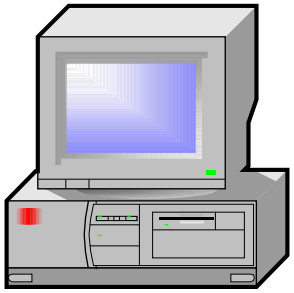
- Einfachste Möglichkeit Kerberos zu benutzen
- Ermöglicht Single-Sign-On auf dem Client (TGT holen/prüfen)
- Anwendungscode ist frei von Kerberos/GSSAPI
- Native Ticket-Caches können ausgelesen werden
- Kann kein Service Ticket holen und delegieren
- Konfiguration über Umgebungsvariablen und Konfigurationsdateien

Java GSS-API

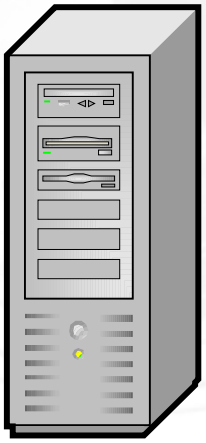
- Java Binding zu GSSAPI (RFC 2853)
- Ermöglicht Zugriff auf Authentifizierungssysteme (z.B. Kerberos)
- Aufwendiger als „reines“ JAAS
- Benutzt intern das JAAS Kerberos Login Module
- Konfiguration über Umgebungsvariablen und optional JAAS-Konfigurationsdateien
- Kann TGT und Service Ticket holen und delegieren

- Single Sign-On
- Kerberos Grundlagen
- Kerberos und Java
- Beispiele
  - JAAS Login
  - Client/Server-AW mit Java GSS-API
  - Pitfalls
- Ausblick

JAAS Login



winxp.secpod.de
(Windows XP)



winkdc.secpod.de
(Windows 2003 Server)

JAAS Login

- Aufruf:

```
lc = new LoginContext("SampleClient", new
   TextCallbackHandler());
lc.login();
```

- Konfigurationsdatei (jax.conf):

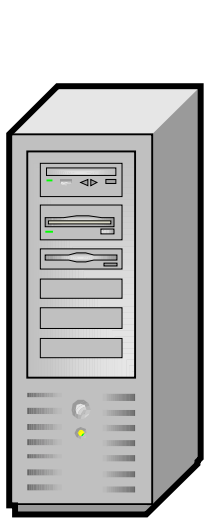
```
SampleClient {
com.sun.security.auth.module.Krb5LoginModule
   required useTicketCache=true;
};
```

JAAS Login

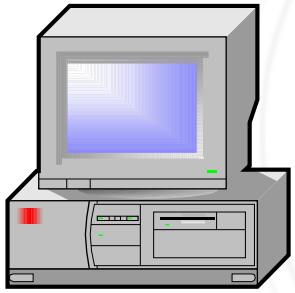
- VM-Parameter:
  - -Djava.security.krb5.**realm=SECPOD.DE**
  - -Djava.security.krb5.**kdc=winkdc.secpod.de**
  - -Djava.security.auth.login.**config=jax.conf**
- Benutzername:

  lc.getSubject().getPrincipals()

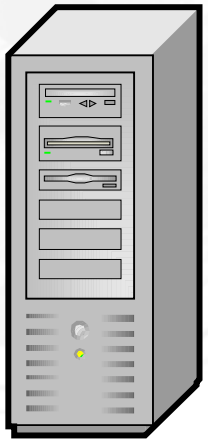
Client/Server-AW mit GSS-API



winkdc.secpod.de
(Windows 2003 Server)



winxp.secpod.de
(Windows XP)



server.secpod.de
(Java TCP/IP)

Client/Server-AW mit GSS-API

- Service-Principle im KDC anlegen:

  Benutzer java/server.secpod.de im Active Directory anlegen

- Key(s) aus dem Active Directory exportieren:

  ktpass -out java.keytab -princ java/server.secpod.de@SECPOD.DE -pass * -mapuser java/server.secpod.de

- Keytab auf den Server kopieren

Client mit GSS-API

```
String server = java/server.secpod.de
GSSManager manager = GSSManager.getInstance();
GSSName serverName = manager.createName(server, null);
GSSContext context = manager.createContext(
    serverName, "1.2.840.113554.1.2.2", null,
    GSSContext.DEFAULT_LIFETIME);
context.requestMutualAuth(true);
context.requestConf(true);
context.requestInteg(true);
```

Client mit GSS-API

```
byte[] token = new byte[0];
while (!context.isEstablished())  {
    token = context.initSecContext(token, 0, token.length);
    if (token != null) {
        // token zum Server schicken
    }
    if (!context.isEstablished()) {
        // token vom Server abholen
    }
}
```

Client mit GSS-API

- Konfigurationsdatei (jax_c.conf):

  com.sun.security.jgss.initiate {

  - com.sun.security.auth.module.**Krb5LoginModule** required **useTicketCache=true**;};

- VM-Paramter:

  -Djava.security.krb5.**realm=SECPOD.DE**

  -Djava.security.krb5.**kdc=winkdc.secpod.de**

  -Djava.security.auth.login.**config=jax_c.conf**

  -Djavax.security.auth.**useSubjectCredsOnly=false**

Client mit GSS-API

- Benutzername:

  context.getSrcName()

- Servername (Rückantwort):

  context.getTargName()

- Ver-/Entschlüsselung:

  context.wrap() / context.unwrap()

Server mit GSS-API

```
GSSManager manager = GSSManager.getInstance();
GSSContext context = manager.createContext((GSSCredential) null);
while (!context.isEstablished()) {
    // token vom Client lesen
    token = context.acceptSecContext(token, 0, token.length);
    if (token != null) {
        //token zum Client schicken
    }
}
```

Server mit GSS-API

- Konfigurationsdatei (jax_s.conf):

```
com.sun.security.jgss.accept {
    com.sun.security.auth.module.Krb5LoginModule
      required useKeyTab=true keyTab=java.keytab
      storeKey=true principal="java/winkdc.secpod.de"
      doNotPrompt=true; };
```

Pitfalls

- Neuere Windowsversionen verhindern das auslesen des Session-Keys aus dem Ticketcache
  - Registry-Eintrag anpassen
- Sun unterstützt derzeit nicht die RC4-Verschlüsselung.
  - DES beim KDC einstellen
- Mehr Informationen dazu unter:
  - http://java.sun.com/j2se/1.5.0/docs/guide/security/jgss/tutorials/Troubleshooting.html

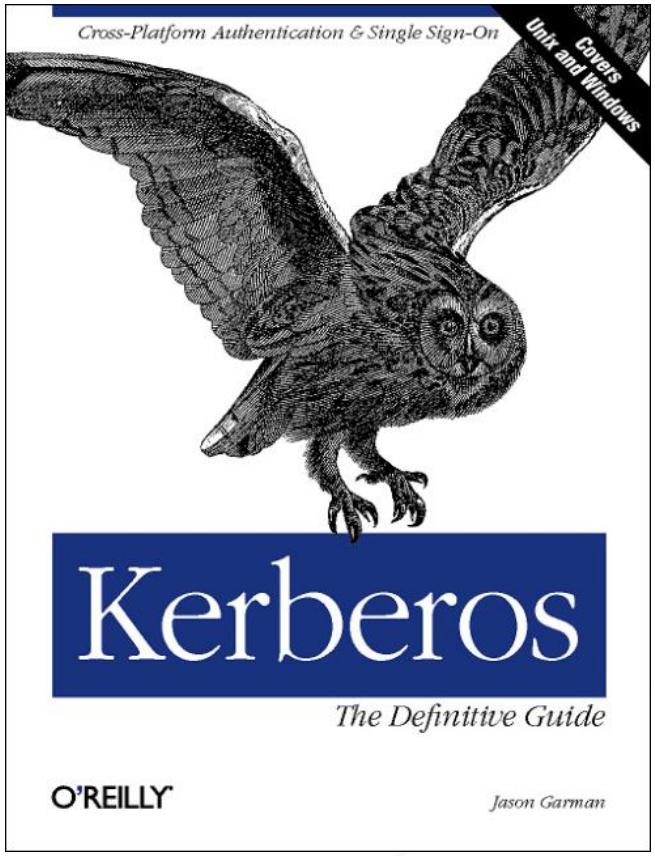
Agenda

- Single Sign-On
- Kerberos Grundlagen
- Kerberos und Java
- Beispiele
- Ausblick

Java 6

- Native platform GSS/Kerberos integration
- SPNEGO HTTP authentication

Buchempfehlung



Kerberos

The Definitive Guide

Jason Garmen

ISBN: 0596004036