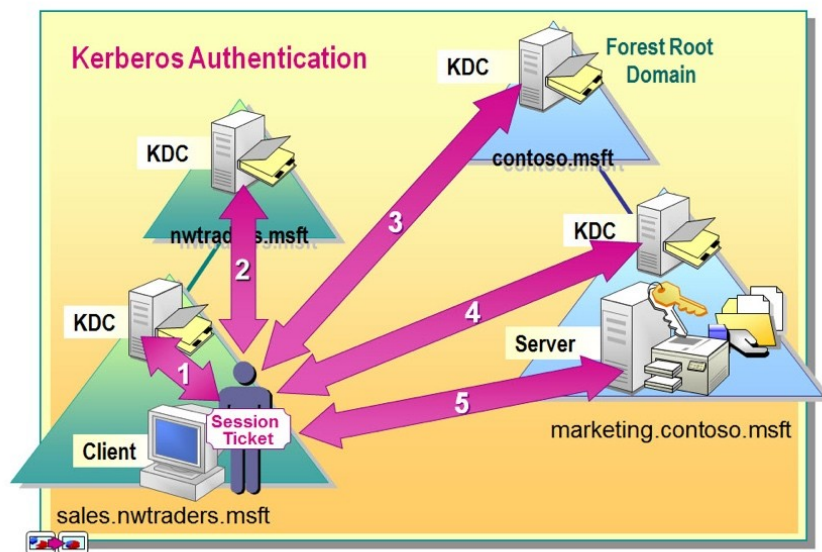




# DIRECTORY SERVICES

## BLOCK 1 - GRUNDLAGEN KERBEROS

### How Kerberos V5 Works



PPT Slide quoted from Microsoft Courseware MOC 2154, Mod 10, page 16

## Inhalt

<b>1 Einführung Directory Services.....</b>	<b>4</b>
<b>1.1 Was ist ein Directory Service ?.....</b>	<b>4</b>
<b>1.2 Wie gehen wir weiter ?.....</b>	<b>5</b>
<b>2 Grundlagen der Netzwerkauthentisierung mit Kerberos.....</b>	<b>6</b>
<b>2.1 Lokale Anmeldung vs. Netzwerkauthentisierung.....</b>	<b>6</b>
<b>2.2 Authentisierung mit Kerberos.....</b>	<b>7</b>
<b>2.3 Kerberos - Begriffe.....</b>	<b>7</b>
2.3.1 KDC.....	7
2.3.2 Realm.....	9
2.3.3 Principals.....	9
2.3.4 Tickets.....	9
2.3.5 Mutual Authentication.....	10
2.3.6 Pluggable Authentication Modules (PAM).....	10
<b>2.4 Delegation.....</b>	<b>11</b>
<b>2.5 Autorisierung, Zugriffskontrolle und Namensdienste.....</b>	<b>11</b>
2.5.1 Authentisierung ist Voraussetzung.....	12
2.5.2 Dienste und Identitäten.....	12
2.5.3 Autorisierung und Kerberos.....	13
<b>2.6 Single Sign-On (SSO).....</b>	<b>13</b>
<b>2.7 Zusammenfassung.....</b>	<b>13</b>
<b>3 Wie funktioniert Kerberos ?.....</b>	<b>13</b>
<b>3.1 Voraussetzungen.....</b>	<b>14</b>
<b>3.2 Das einstufige Kerberos-Verfahren.....</b>	<b>15</b>
<b>3.3 Das zweistufige Kerberos-Verfahren.....</b>	<b>19</b>
3.3.1 Ablauf des zweistufigen Kerberos-Verfahrens:.....	20
<b>3.4 Zusammenfassung.....</b>	<b>23</b>
<b>4 Einstieg in LDAP.....</b>	<b>24</b>
<b>4.1 LDAP.....</b>	<b>24</b>
4.1.1 Das LDAP-Datenmodell.....	24
4.1.2 LDIF-Repräsentation von LDAP-Daten.....	25
<b>4.2 OpenLDAP-Tools.....</b>	<b>26</b>
4.2.1 Suchen mit ldapsearch.....	26
4.2.2 Suchfilter.....	27
4.2.3 Authentisierung.....	27
4.2.4 Änderungen mit LDIF.....	28
4.2.5 Weitere OpenLDAP-Befehle.....	29
<b>5 Aufbau einer eigenen Kerberos Infrastruktur.....</b>	<b>31</b>
<b>5.1 Vorarbeiten für die Beispielumgebung.....</b>	<b>31</b>
5.1.2 DNS-Installation auf vmLS4.....	33
5.1.3 Zeitsynchronisation.....	36
<b>6 Key Distribution Center von MIT Kerberos.....</b>	<b>36</b>
<b>6.1 Installation Kerberos auf vmLS4.....</b>	<b>37</b>
<b>6.2 Konfiguration des KDC.....</b>	<b>38</b>
6.2.1 Initialisierung der KDC-Datenbank.....	39
6.2.2 Mit kadmin.local Principals anlegen.....	41
6.2.3 Das Problem mit dem automatischen Starten des Kerberos-Dienstes.....	42
6.2.4 Das KDC starten und testen.....	44
<b>7 Administration von MIT Kerberos.....</b>	<b>49</b>
<b>7.1 Kadmin-Dienst.....</b>	<b>49</b>
<b>7.2 Administrative Zugriffe kontrollieren.....</b>	<b>49</b>
<b>7.3 Starten der administrativen Dienste.....</b>	<b>50</b>

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 3/64

<b>7.4 Principals verwalten.....</b>	<b>52</b>
7.4.1 Passwortrichtlinien festlegen.....	52
7.4.2 Anwender-Principals anlegen.....	53
7.4.3 Dienste-Principals anlegen.....	54
<b>7.5 Keytabs verwalten.....</b>	<b>56</b>
<b>8 Clientkommandos von MIT Kerberos.....</b>	<b>58</b>
<b>8.1 Die Kommandos kinit und klist.....</b>	<b>58</b>
8.1.1 Tickets holen.....	58
<b>8.2 Das Kommando kvno.....</b>	<b>60</b>
<b>8.3 Das Kommando kpasswd.....</b>	<b>61</b>
<b>8.4 Das Kommando kdestroy.....</b>	<b>61</b>
<b>9 Einfache Kerberos Authentifizierung auf vmLP1.....</b>	<b>62</b>
9.1 Szenario.....	62
9.2 Installation von SSSD auf vmLP1.....	63
<b>10 Aufgabe:.....</b>	<b>64</b>

# 1 Einführung Directory Services

## 1.1 Was ist ein Directory Service ?

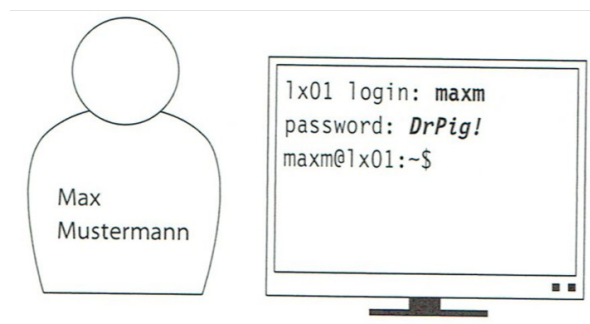
Zuerst einmal zum Begriff "Directory Service". Dies ist der englische Name für Verzeichnisdienst. In diesem Dokument werden beide Begriffe verwendet. Es ist also ein Service, den wir in einem Rechnernetzwerk zur Verfügung stellen.

Wie sie wissen, besteht ein Rechnernetzwerk aus vielen Objekten. Objekte sind zum Beispiel Rechner (PC's oder Server), Benutzer, Drucker, File-Shares, Mail-Dienste, etc. Ein Verzeichnisdienst verwaltet ganz allgemein diese Objekte. Wir als Administratoren, haben dank einem Directory Service, stets die Übersicht über die vorhandenen Objekte. Eine wichtige Anforderung an einen Verzeichnisdienst bildet auch der kontrollierte Zugriff auf diese Objekte. Man spricht auch vom kontrollierten Zugriff auf Netzwerkressourcen. In einer Unternehmung will man zum Beispiel genau wissen, wer auf welche Dateien zugreifen und diese Lesen oder verändern kann. Auch muss sich jeder Benutzer in einem Rechnernetzwerk identifizieren, indem er seinen Benutzernamen und sein Passwort verwendet. Ein Directory Service ist also nichts anderes als eine Datenbank, welche grosse Mengen an Daten aufnehmen und verwalten kann.

Eine Directory Service zur Verwaltung von Objekten eines Rechnernetzwerkes basiert auf der **X.500-Spezifikation**. Dieser Standard ist ein Entwurf eines globalen Verzeichnisdienstes. Viele kommerziell erhältliche Produkte, basieren auf diesem Standard. So zum Beispiel auch die Active Directory Domain Services, ADDS, von Microsoft. Der Zugriff auf die Objekte in dieser X.500-basierten Datenbank erfolgt mit dem **Lightweight Directory Access Protocol (LDAP)**. Bei einem Logon-Prozess (Anmeldevorgang) wird beispielsweise mit LDAP der Verzeichnisdienst abgefragt, ob dieser Benutzer die Anmeldung durchführen darf. Auch Abfragen der Server untereinander, werden mit LDAP gemacht.

Damit der Zugriff auf Ressourcen in einer Netzwerkumgebung gezielt gesteuert werden kann, setzt man **Authentifizierungsprotokolle** ein.

Beispiel: Ein Anwender meldet sich an seinem Arbeitsplatzrechner an. Dabei authentisiert er sich durch die Eingabe seines Benutzernamens und des Passwortes:



Hier spielt das Protokoll **Kerberos** V5 eine wichtige Rolle. Auch Microsoft setzt dieses in ihrem ADDS ein. Ab Windows 2000 und dem damit eingeführten *Active Directory* hat Kerberos v5 auch in die Windows-Welt Einzug gehalten: **In Active-Directory-Domänen wird Kerberos v5 als primärer Authentisierungsdienst verwendet**. Die grosse Verbreitung von Windows und Active Directory hat sicherlich einen wichtigen Beitrag dazu geleistet, dass Kerberos sich noch mehr als Standardauthentisierungsverfahren durchsetzen konnte.

Was macht Kerberos? – ganz vereinfacht formuliert: Kerberos authentifiziert einen Benutzer bei einem Server und es stellt zudem sicher, dass sowohl der User wie auch der Server keine falsche Identität vorgaukeln können – ein sehr wichtiger Punkt. Der authentifizierte Benutzer erhält vom **Key Distribution Center (KDC)** ein Ticket. Dieses Ticket ist der Nachweis seiner Identität und wird als ticketgenehmigendes Ticket oder engl. **Ticket Granting Ticket**, TGT, bezeichnet. Nach der Authentifizierung werden für die Zugriffe auf Netzwerkressourcen

M159\_SKRIPT\_THEMA1\_KERBEROS\_V2.6.ODT

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 5/64

seitens des KDCs dann zusätzlich noch weitere Tickets, sogenannte **Session Tickets** für die weiteren Aktivitäten des Benutzers ausgestellt.

## 1.2 Wie gehen wir weiter ?

Verzeichnisdienste basieren auf einem Authentisierungsdienst (Authentisierungsprotokoll): Kerberos. Diese Basistechnologie wollen wir in diesem ersten Teil beleuchten und verstehen. Im 2. und 3. Teil des Moduls werden wir Fallbeispiele mit den am Markt verfügbaren Produkten realisieren und werden uns dann auf den Microsoft Active Directory Domain Service, ADDS und das OpenSource Produkt Samba4 fokussieren. Beide Produkte verwenden X.500, LDAP und Kerberos zur Realisierung eines Verzeichnisdienstes.

Vorerst widmen wir uns also einer nativen Kerberos - Umgebung.

Was ist mit „nativ“ gemeint?

Wir verwenden Kerberos V5 in seiner Standardform, wie sie vom MIT (Massachusetts Institute of Technology, Boston) entwickelt wurde. Diese wurde der Allgemeinheit zur Verfügung gestellt und ist im Internet-Standard RFC 4120 spezifiziert. Das MIT nennt dieses KRB5. Produkte, wie zum Bsp. das Microsoft ADDS, verwenden diesen Standard.

Als Verzeichnisdienst-Datenbank verwenden wir in Teil 1 die integrierte Datenbank von Kerberos V5.

Noch ein Wort zum Kerberos-Standard:

Neben dem MIT-Kerberos-Standard, welcher weltweit am meisten genutzt wird, existiert auch eine Implementierung, welche sich *Heimdahl* nennt. Diese wurde von einer schwedischen Universität (KTH) entwickelt und basiert nicht auf den Quellen vom MIT. Heimdahl stellt also die europäische Version des Kerberos Standards dar. Dies kommt daher, weil in den USA kryptografische Software strengen Exportbeschränkungen unterlag. Nachdem die US-Regierung die Exportbeschränkungen gelockert hat, kann MIT-Kerberos heutzutage auch ausserhalb der USA eingesetzt werden. Heimdahl-Kerberos wird beispielsweise auch heute noch bei Samba4 eingesetzt.

Eine komplette Neuimplementierung unter der GNU General Public License (GPL) stellt *ShiShi* dar. Diese wird aber in diesem Modul nicht behandelt.

MIT- und Heimdahl-Kerberos sind im Wesentlichen für Unix-artige Betriebssysteme gedacht, obwohl es auch eine MIT-Version für Windows gibt. Beide Implementierungen unterscheiden sich in manchen Punkten, sind aber hinreichend kompatibel, um eine Interoperabilität zu gewährleisten.

## 2 Grundlagen der Netzwerkauthentisierung mit Kerberos

Kerberos ist wie gesagt ein Authentisierungsdienst bzw. ein Authentisierungsprotokoll. Unter **Authentisierung** versteht man den Nachweis der eigenen Identität: Ich bin „Max Mustermann“. Dabei kann es sich um die Identität einer Person oder auch um die eines Computerprogrammes handeln. Wird eine Identität überprüft, spricht man von **Authentifizierung**. Auch hier kann es sich um einen Anwender oder um ein Computerprogramm handeln.

Sehen sie sich das Bild im Kapitel 1.2 an: Durch die Eingabe des Usernamens und des Passwortes **authentisiert** sich der Anwender gegenüber dem Anmeldeprogramm des Computers. Das Anmeldeprogramm überprüft das Passwort und **authentifiziert** den Anwender.

### Lernvideos

Schauen Sie sich folgende Lern-Videos an. Machen Sie dazu Notizen, um den Inhalt mit den Kernaussagen festzuhalten. Denken Sie dabei auch an den Spicker, den Sie für LB1 verwenden dürfen. (Total ca 20 Min.).

Damit sie Zugriff auf Lernvideos unter <https://www.linkedin.com/learning/> erhalten, müssen sie sich einmalig registrieren. Folgen Sie dazu genau (!) der Anleitung unter:

<https://gibb.ch/attachments/attachment-backend/download/1126?fileName=Anleitung+Registrierung+LinkedIn+Learning.pdf>

**Achtung:** Im Bestätigungs **E-Mail** auf [xxx@iet-gibb.ch](mailto:xxx@iet-gibb.ch) den **DEUTSCHEN** Aktivierungsknopf drücken

4 Videos:

The screenshot shows a LinkedIn Learning interface. The search bar contains 'grundlegende informationen ke'. The video player displays 'Video: Grundlegende Informationen zu Netzwerksicherheit Grundkurs'. The right sidebar shows a list of videos under the heading '3. Authentifizierungsdienst Kerberos':

- Grundlegende Informationen zu Kerberos (6 Min. 13 Sek.)
- Funktionsweise von Kerberos (5 Min. 34 Sek.)
- Beispiel einer Kerberos-Authentifizierung (5 Min. 30 Sek.)
- Kerberos implementieren (3 Min. 4 Sek.)

ros

### 2.1 Lokale Anmeldung vs. Netzwerkauthentisierung

Das oben erwähnte lokale Anmeldeprogramm ermöglicht dem Anwender die interaktive Verwendung eines lokalen Rechners. Daneben gibt es aber andere Programme, die nur spezielle Dienste zur Verfügung stellen. Auch diese Dienste verlangen im Allgemeinen, dass sich der Anwender authentisiert.

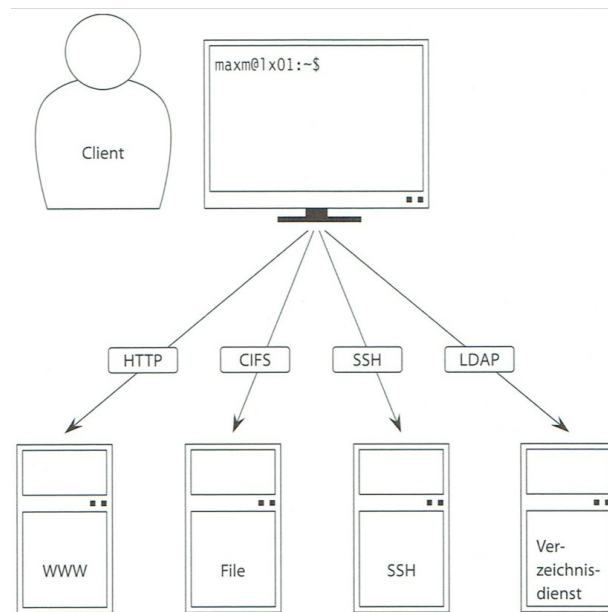
Beispiele solcher Dienste:

Netzwerkdienst	Netzwerkprotokoll
Webserver	HTTP
E-Mail-Dienst	SMTP, POP3, IMAP

Dateidienst	CIFS, NFS, AFS, ...
Druckdienst	IPP
Verzeichnisdienst	LDAP
Host (interaktive Nutzung)	SSH, RSH, Telnet

Diese Dienste laufen in der Regel auf entfernten Rechnern und der Zugriff erfolgt über spezielle Netzwerkprotokolle. Damit kann man also lokale Authentisierung und Netzwerkauthentisierung unterscheiden.

Der Anwender meldet sich also lokal an seinem Arbeitsplatz an und greift dann auf verschiedene Dienste im Netzwerk zu. Auch gegenüber diesen muss er sich authentisieren.



## 2.2 Authentisierung mit Kerberos

Beim Prozess der Netzwerkauthentisierung bzw. -authentifizierung sind mindestens 2 Teilnehmer beteiligt: ein Client, der sich authentisieren muss, und ein Dienst, der den Client authentifizieren möchte. Diese beiden Teilnehmer können sein:

- Anwender - Computerprogramm
- Anwender - Anwender
- Computerprogramm - Computerprogramm

Bildlich kann dies so dargestellt werden:



## 2.3 Kerberos - Begriffe

### 2.3.1 KDC

Für die Kerberos-Authentisierung kommt noch ein dritter Teilnehmer hinzu, der Kerberos-Dienst. Dieser wird als Key Distribution Center (KDC) bezeichnet. Das KDC vermittelt die Authentisierung zwischen Client und Netzwerkdiensten. Dazu müssen alle Clients und alle Dienste ein Vertrauensverhältnis (Trust) zu ihrem KDC haben. Deshalb wird der Kerberos-

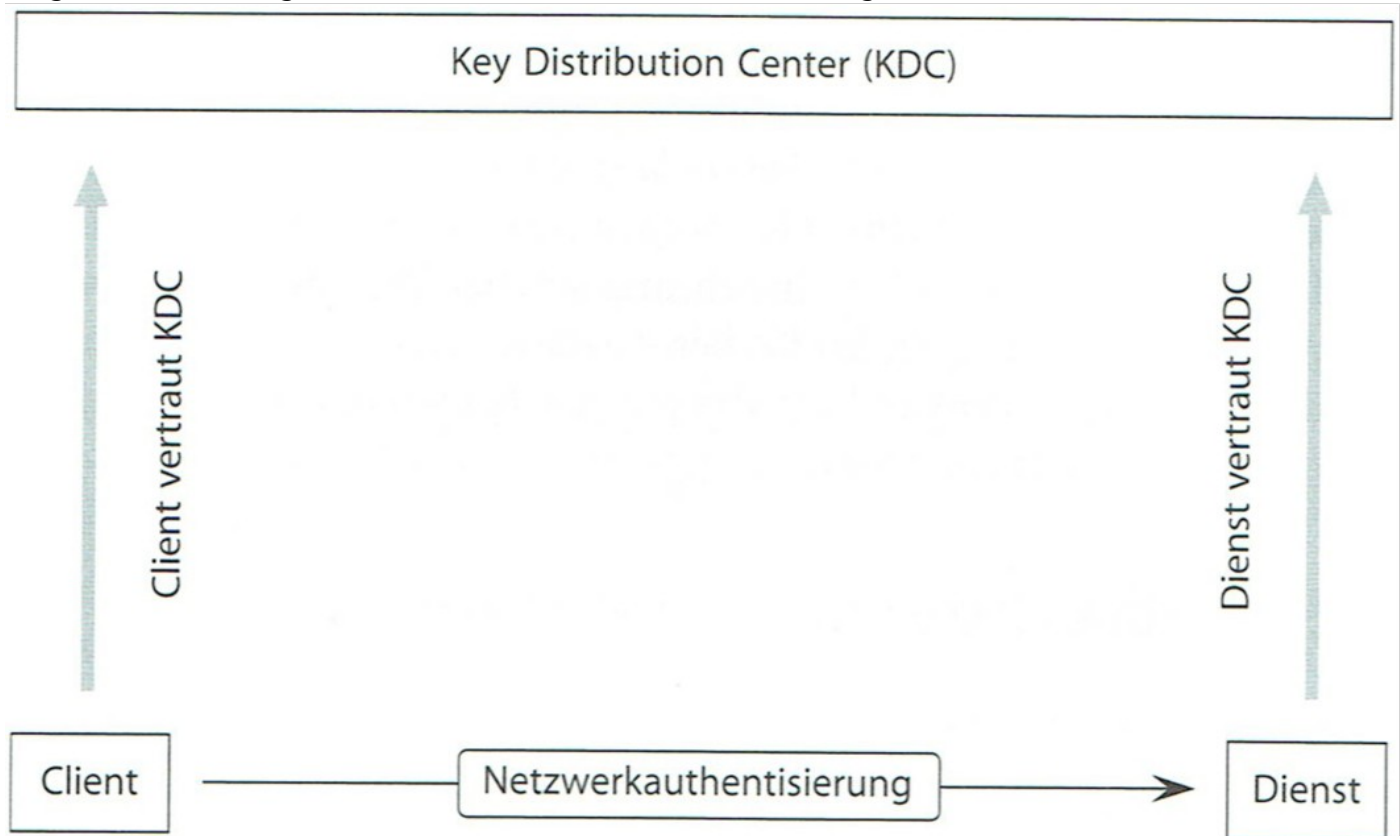


Dienst auch als Trusted Third Party bezeichnet. Damit wird auch klar, warum der Name „Kerberos“ gewählt wurde: Kerberos (Cerberus) ist ein dreiköpfiges Ungeheuer aus der griechischen Mythologie. Siehe dazu: <https://de.wikipedia.org/wiki/Kerberos>  
 Die 3 Köpfe stehen stellvertretend für die 3 Elemente der Authentisierung:

- Der Client
- Der Server
- Die trusted third party oder eben das KDC



Folgende Abbildung illustriert die Kerberos-Authentisierung:





### 2.3.2 Realm

Aus administrativen und organisatorischen Gründen teilt man Kerberos-Umgebungen in sogenannte Realms ein (engl. Realm = Bereich, Reich, Königreich, Gefilde). Kerberos Realms entsprechen typischerweise Organisationen oder Teilen von Organisationen. Jeder Anwender oder Client und jeder Dienst oder Host gehört zu einem Kerberos Realm. Innerhalb dieses Realm gibt es ein KDC, das für die Authentisierungsvorgänge zwischen Clients und Diensten zuständig ist. Die Kerberos-Authentisierung kann auch Authentisierungsvorgänge zwischen einzelnen Realms umfassen. Hier spricht man von Cross-Realm-Authentisierung. Jeder Kerberos-Realm besitzt einen Realm-Namen. Dieser entspricht per Konvention dem DNS-Namen der Umgebung in Grossbuchstaben.

**Im Gegensatz zu DNS-Namen spielt die Gross- und Kleinschreibung für Realm-Namen eine Rolle.** Wir werden eine Umgebung aufbauen mit dem DNS-Namen m159.iet-gibb.ch und dem Realm-Namen M159.IET-GIBB.CH.

Zum Vergleich: In der Microsoft-Welt entspricht eine Kerberos Realm einem Domainnamen.

### 2.3.3 Principals

Clients und Dienste innerhalb eines Kerberos Realm werden durch **Kerberos Principals** repräsentiert. Kerberos Principals sind also die Namen, welche Kerberos für die Darstellung von Client- oder Dienstidentitäten verwendet.

Es gibt folgende Teilnehmer in einer Kerberos Realm:

- Der Client, dargestellt durch den **Client Principal**. Typischerweise ist das der Benutzername und dessen Kerberos Realm, getrennt durch ein @-Zeichen. Der Anwender 'jaeggi' der Realm M159.IET-GIBB.CH würde dann jaeggi@M159.IET-GIBB.CH lauten.
- Der Dienst, dargestellt durch den **Dienste-Principal**. Ein Webdienst auf dem Server www.m159.iet-gibb.ch wird durch den Principal Namen http/www.m159.iet-gibb.ch@M159.IET-GIBB.CH repräsentiert
- Das KDC selbst, also der Kerberos-Dienst

An diesen Beispielen sehen wir, dass Principal-Namen aus mehreren Komponenten bestehen können. Diese werden durch ein /-Zeichen voneinander getrennt. Bei Dienste-Principals besteht die Konvention, den langen DNS-Hostnamen (FQDN) als zweite Komponente zu verwenden.

Allgemeine Syntax für Principal-Namen:

Komponente\_1[/Komponente\_2/.../Komponente\_N]@REALM

Komponente\_2 bis Komponente\_N sind optional, wobei Komponente\_1 als *Primary* und Komponente\_2 als *Instance* bezeichnet wird.

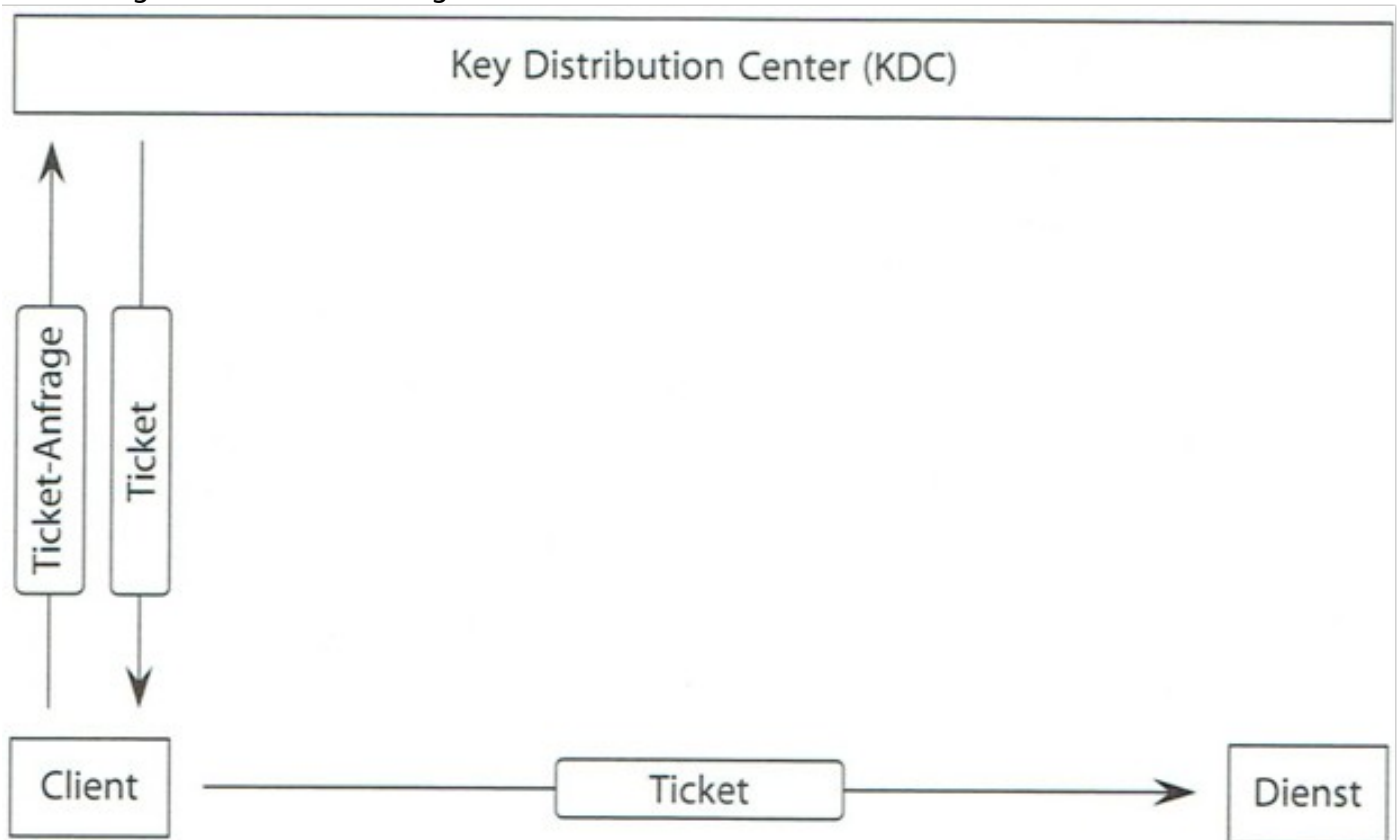
### 2.3.4 Tickets

Die Kerberos Authentisierung basiert auf den Kerberos Tickets. Diese „Eintrittskarten“ sind nur für eine bestimmte Zeit benutzbar. Sie verlieren ihre Gültigkeit in der Regel nach 8-10 Stunden.

Man muss zwischen den zwei Ticketarten **Ticket Granting Ticket (TGT)** und **Session Ticket (ST)** unterscheiden. Nach erfolgreicher Überprüfung der Identität stellt das KDC dem

Client das TGT aus. Mit diesem TGT fordert der Client später Session Tickets an: Wenn er das erste Mal den Webservice verwenden möchte, benötigt er für diese «Web-Session» ein Session Ticket. Der Client fordert das Session Tickets beim KDC an, indem er sein TGT und den gewünschten Dienst mitteilt. Später benötigt der Client womöglich Zugriff aufs Mail, dazu benötigt er ein weiteres Session Ticket für den Zugriff auf den Mail Server. Netzwerkclients erhalten Tickets auf Anfrage vom KDC ausgestellt. Greift ein Client dann auf einen Dienst zu, so legt er ihm ein Ticket vor, das das KDC speziell für diesen Dienst ausgestellt hat und das den Principal-Namen des Clients enthält. Der Dienst erkennt daran die Authentizität des Clients.

Abbildung zur Authentisierung mit Kerberos-Ticket:



### 2.3.5 Mutual Authentication

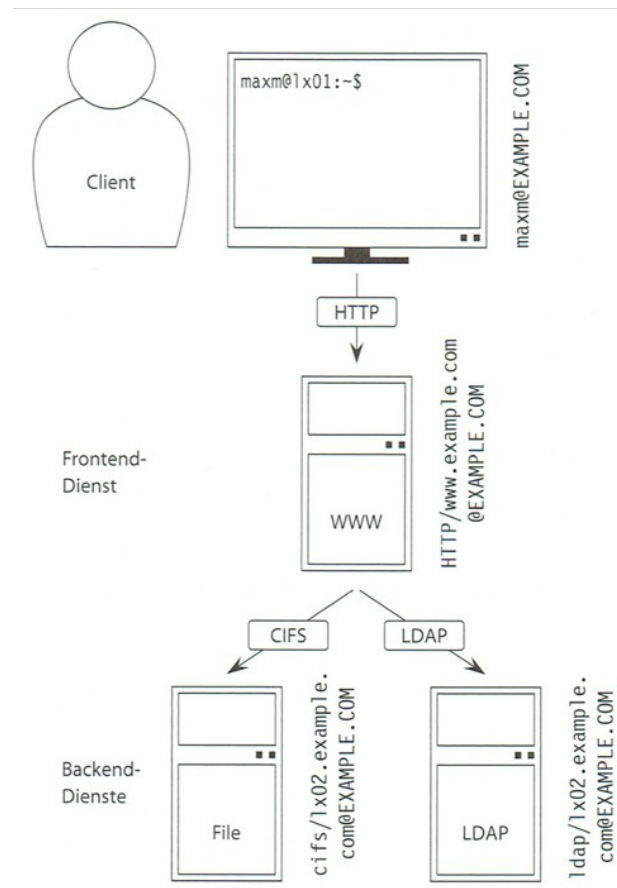
Kerberos bietet auch die Möglichkeit, dass der Client eine Authentisierung des Dienstes verlangen kann. Dies ist dann der Fall, wenn der Client sicher gehen will, dass er z. Bsp. vom „richtigen“ Mail-Server seine Mails bezieht oder dass er seine Dateien auf dem „richtigen“ File-Server ablegt. Wenn der Client eine Authentisierung des Dienstes verlangt, spricht man von Mutual Authentication.

### 2.3.6 Pluggable Authentication Modules (PAM)

Der Kerberos-Authentisierungsdienst wurde in erster Linie für die Authentisierung von Clients gegenüber Netzwerkdiensten geschaffen. Für die lokale Anmeldung existieren Programme, die die Authentifizierung des Anwenders durchführen, wie beispielsweise login oder der grafische gdm unter Linux. Solche Programme verwenden die Pluggable Authentication Modules (PAM), um die lokale Authentifizierung durchzuführen. Anmeldeprogramme lassen sich zur Verwendung des Kerberos-Protokolls konfigurieren. Sie kennen das unter Windows: Das Anmeldeprogramm kann für eine lokale Anmeldung oder für eine Anmeldung in der Domain verwendet werden.

## 2.4 Delegation

Es kommt oft vor, dass Dienste selber auf andere Netzwerkdienste zugreifen. Beispielsweise ein Web-Server auf einen File-Service:



Die Abbildung hier stellt eine typische Multi-Tier-Architektur dar. Der Anwender maxm greift auf den ersten Dienst (Frontend) zu. Der muss seinerseits auf einen weiteren Dienst (Backend) zugreifen, um die Anfrage des Anwenders zu beantworten. Das Frontend tritt gegenüber dem Anwender in einer Service-Rolle und gegenüber dem Backend in einer Client-Rolle auf.

Was bedeutet dies nun für die Kerberos-Authentisierung? Hier gibt es zwei Möglichkeiten:

1. Das Frontend kann unter seiner eigenen Identität auf das Backend zugreifen. Gemäss Abbildung würde das bedeuten, dass der Backend-Dienst einen Zugriff des Principals `http/1x02.example.com@EXAMPLE.COM` registriert.
2. Dem Frontend kann die Benutzung der Clientidentität des Anwenders gestattet werden. Dieser Vorgang heisst **Delegation**. In der Abbildung bedeutet das, dass das Frontend sich dem Backend gegenüber als Anwender maxm ausgeben kann. Der Backend-Dienst registriert dann einen Zugriff des Principals `maxm@EXAMPLE.COM`. Er erlaubt den Zugriff gemäss dessen Berechtigungen.

Kerberos unterstützt Delegation durch die Übermittlung von Tickets an das Frontend. Dabei unterscheidet man zwei Varianten: **Ticket Proxing** und **Ticket Forwarding**.

## 2.5 Autorisierung, Zugriffskontrolle und Namensdienste

Neben der Authentisierung sind auch Begriffe **Autorisierung** und **Zugriffskontrolle** (Access Control) wesentlich für jedes Sicherheitskonzept.

Bei der **Autorisierung** wird festgelegt, mit welchen Berechtigungen Benutzer auf Ressourcen im Netzwerk zugreifen dürfen. Netzwerkdienste, die diese Ressourcen anbieten, führen im Allgemeinen eine **Zugriffskontrolle** (Access Control) durch. Dabei prüfen sie, ob und wie der zugreifende Benutzer autorisiert ist. Dementsprechend wird der Zugriff auf die Ressource erlaubt, verweigert oder nur eingeschränkt gewährt.

Damit das funktioniert, muss ein Dienst wissen, welchem Anwender er auf welches Objekt welche Art von Zugriff erlauben kann. **Welche Autorisierungsinformationen ein Dienst dafür benötigt und wo diese hinterlegt sind, hängt vom betrachteten Dienst ab.**

Beispiele:

- Dateidienste stellen Dateisysteme zur Verfügung. Die Dateien und Verzeichnisse darin sind mit **Access Control Lists (ACLs)** versehen. Anhand dieser kann ein Dateidienst entscheiden, ob ein bestimmter Anwender auf eine Datei oder ein Verzeichnis zugreifen darf und in welcher Form dieser Zugriff erfolgen kann (lesen, lesen und schreiben, ausführend).
- Webanwendungen haben ebenfalls Zugriffsentscheidungen zu treffen. Beispielsweise lässt sich beim Apache-Webserver der Zugriff auf einzelne Verzeichnisse durch **.htaccess**-Dateien einschränken.
- Bei Verzeichnisdiensten (z.Bsp. OpenLDAP) können Zugriffsrechte sehr feingranular vergeben werden.

### 2.5.1 Authentisierung ist Voraussetzung

Damit ein Dienst seine Zugriffskontrolle auf einer sicheren Basis durchführen kann, muss er sich vorher von der Identität des zugreifenden Anwenders überzeugt haben. Die **Authentifizierung des Anwenders ist also eine Vorbedingung für die sichere Zugriffskontrolle.**

Es gibt Dienste, welche diese Regel nicht beachten. Zum Beispiel NFS (Network File System) bis zu Version 3.

### 2.5.2 Dienste und Identitäten

Ein weiterer wesentlicher Punkt ist nun, dass verschiedene Dienste unterschiedliche Darstellungsformen für die Identitäten ihrer Anwender benutzen. Damit ein Dienst eine Zugriffsentscheidung treffen kann, müssen die notwendigen Autorisierungsdaten in der passenden Darstellungsform vorliegen.

Beispiele von Autorisierungsdaten sind:

- Dateidienste verwenden in den ACLs numerische Identifikatoren: Unter Windows ist das der **Security Identifier (SID)**, unter Linux die **User-ID (UID)**. In der Regel werden Benutzer zu Gruppen zusammengefasst, die wiederum mit numerischen Identifikationsnummern bezeichnet werden. Unter Windows wird auch für solche Gruppen eine SID verwendet, unter Linux eine **Group-Id (GID)**.
- Ein E-Mail-Dienst könnte beispielsweise die **E-Mail-Adresse** des Anwenders verwenden
- Der Verzeichnisdienst LDAP benutzt sogenannte **Distinguished Names (DNs)**
- Kerberos verwendet **Principal-Namen** als Darstellungsform für die Identität von Clients und Diensten

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 13/64

### 2.5.3 Autorisierung und Kerberos

Das klassische Kerberos führt **ausschliesslich die Authentisierung** seiner Principals durch.

Kerberos ist also ausschliesslich für die Authentisierung von Diensten und Clients zuständig! Die für die Zugriffskontrolle benötigten Autorisierungsdaten muss ein kerberisierter Dienst aus einer zusätzlichen Datenquelle beziehen.

Der Einsatz von LDAP als Datenbasis für solche Autorisierungsdaten ist sehr verbreitet.

Die **strenge Trennung zwischen Authentisierung (durch Kerberos) und zentraler Bereitstellung von Autorisierungsdaten (durch einen Verzeichnisdienst wie LDAP) ist auch vom Standpunkt eines sicheren Systemdesigns zu befürworten**. Bei sicherheitsrelevanter Software ist es immer sehr wesentlich, den Funktionsumfang einzelner Komponenten zu beschränken und gegeneinander abzugrenzen.

Beispiel:

Die Linux-Datei `/etc/passwd` beinhaltet Authentisierungs- (Passwörter) und Autorisierungsdaten (UIDs, GIDs). Aus Sicht der Sicherheit ist dieses Mischmasch eigentlich nicht gewünscht...

### 2.6 Single Sign-On (SSO)

In vielen Umgebungen ist es üblich, dass einzelne Dienste eigene Passwortdatenbanken verwalten müssen. Bei einem Apache-Webserver ist das beispielsweise die `.htpasswd`- und bei Samba die `smbpasswd`-Datei. In solchen Fällen müssen sich die Anwender für die unterschiedlichen Dienste ihre Passwörter merken. Auch führt dies zu einer umständlichen Benutzerverwaltung, da diese nicht zentral geführt werden kann.

Wenn ein Anwender sein Passwort nur noch einmal eingeben muss, um sich bei allen von ihm verwendeten Diensten zu authentifizieren, spricht man von Single Sign-On (SSO).

Dies geschieht in der Regel bei der Anmeldung am System und wird auch als Primärauthentisierung bezeichnet. Nachdem er angemeldet ist und bis zum Ablauf seiner Sitzung (Session), gilt er für alle Dienste im Netzwerk als authentisiert. Die Anmeldung findet also nicht mehr an einer lokalen Maschine oder an einem speziellen Dienst, sondern **netzwerkweit** statt. Der Anwender muss sein Passwort nur noch einmal eingeben!

Kerberos bietet die Möglichkeit des sicheren Single Sign-On, das auf den beschriebenen Kerberos Tickets beruht. Diese Tickets haben in der Regel eine kurze Gültigkeit. Daraus ergibt sich auch, dass Kerberos SSO zeitlich begrenzt ist.

### 2.7 Zusammenfassung

Kerberos ist also ein **Ticket-basierter, Trusted-Third-Party-Authentisierungsdienst** mit **Single-Sign-On-Funktionalität**. Die Trusted third Party wird auch als **Key Distribution Center** oder **KDC** bezeichnet. Dieses KDC ist für Authentisierungsvorgänge zwischen Client- und Dienste-Principals **innerhalb seines Realm** verantwortlich. Neben der reinen Authentisierung unterstützt Kerberos auch die Delegation im Sinne der Weiterleitung einer Clientidentität.

Autorisierung gehört nicht zu den Aufgaben des klassischen Kerberos-Dienstes.

## 3 Wie funktioniert Kerberos ?

Dieses Kapitel beschreibt die grundlegende Funktionsweise von Kerberos v5. Wie Sie wissen, benutzt Kerberos für die Authentisierung Tickets und Sitzungsschlüssel. Sie erfahren hier, wie Clients an diese Kerberos Credentials gelangen und sich mit deren Hilfe gegenüber kerberisierten Diensten ausweisen können. Dazu lernen Sie die beiden Arbeitsmodi eines

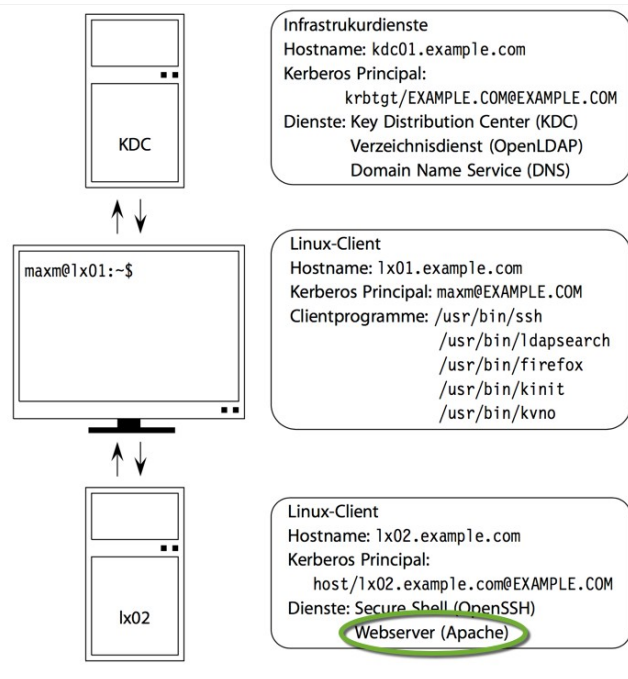
M159\_SKRIPT\_THEMA1\_KERBEROS\_V2.6.ODT

Kerberos-Servers (AS und TGS) kennen und erfahren, was beim Zugriff auf einen kerberisierten Dienst passiert.

Im Folgenden wird das einstufige und das zweistufige Kerberos-Verfahren beschrieben. Bei unserer praktischen Umsetzung und auch für die LB ist das zweistufige Verfahren von Bedeutung.

Der folgende Abschnitt soll Ihnen einen Überblick über das Funktionsprinzip des Kerberos-Protokolls geben. Die Beispiele hier sind vereinfacht dargestellt.

Als Beispiel dient der Anwender Max Mustermann, der auf die kerberisierte Website <https://www.example.com> zugreifen möchte und sich dabei authentisieren muss. Erinnern Sie sich an folgendes Szenario:



Damit das funktioniert, sind zunächst einige Voraussetzungen zu erfüllen.

### 3.1 Voraussetzungen

Jedem Client und jedem Dienst muss eine Kerberos-Identität in Form eines **Principals** zugeordnet sein. Für Max Mustermann lautet der **Client-Principal-Name** im Beispiel maxm@EXAMPLE.COM und der **Service-Principal-Name**

HTTP/lx02.example.com@EXAMPLE.COM.

Ausserdem müssen Client und Dienst jeweils einen oder mehrere kryptografische Langzeitschlüssel besitzen.

Der verwendete **Langzeitschlüssel** des Clients wird im Folgenden als **Client Key** bezeichnet.

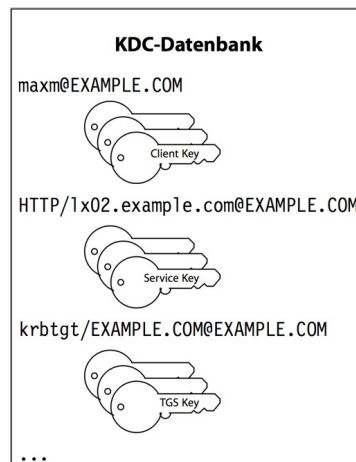
Auch ein Service Principal besitzt in der Regel verschiedene Schlüssel für die unterschiedlichen Verschlüsselungsverfahren. Die Schlüssel des HTTP Principal könnten auch von einem Klartextpasswort abgeleitet sein, sind in der Regel aber zufällig gewählt. Der verwendete Schlüssel des HTTP Principal wird hier als **Service Key** bezeichnet.

**Die Principals müssen ihre Schlüssel kennen: Der Client gelangt an seine Schlüssel über das Anwenderpasswort.**

Dagegen benötigt ein Dienst üblicherweise direkten Zugriff auf seine Schlüssel. Unter Unix erfolgt die **Speicherung der Langzeitschlüssel von Diensten in sogenannten Keytab-Dateien.**

Die nächste Voraussetzung ist, dass es einen (oder mehrere) Kerberos-Server (KDCs) gibt. Eine wichtige Komponente eines jeden KDC ist die **KDC-Datenbank**, in der es zu jedem Kerberos Principal des Realms einen Eintrag geben muss. Insbesondere **sind dort ebenfalls die kryptografischen Langzeitschlüssel aller Principals gespeichert**. Die folgende Abbildung zeigt den Aufbau der KDC-Datenbank:

**Abbildung 3.1**



**Jeder Principal, egal ob Client oder Dienst, muss nur seine eigenen Schlüssel kennen. Nur das KDC muss die Schlüssel sämtlicher Principals kennen.**

Weitere Voraussetzungen für das Funktionieren des Kerberos-Protokolls sind:

- Die **Uhrzeiten** der beteiligten Systeme müssen einigermaßen **synchron** sein. Normalerweise genügt es, dass die Systemzeiten nicht mehr als fünf Minuten voneinander abweichen. Diese maximal erlaubte Zeitabweichung ist konfigurierbar und heißt *Clock Skew*. **Diese Abhängigkeit von den Uhrzeiten der Systeme kommt daher, dass Kerberos für das Erzeugen der Authentikatoren Zeitstempel verwendet.**
- Die **Hostnamensauflösung muss auf allen beteiligten Systemen identisch funktionieren**. Das hängt damit zusammen, dass DNS-Namen in der Regel Bestandteil der Service-Principal-Namen sind und alle Beteiligten des Authentisierungsvorgangs die gleiche Vorstellung von diesen Namen haben müssen.

Die bisher genannten Punkte haben mit dem eigentlichen Kerberos-Protokoll noch nichts zu tun. Sie sind vielmehr Voraussetzungen für das Funktionieren des Protokolls und es ist die Aufgabe der Kerberos-Administration, sie zu erfüllen.

## 3.2 Das einstufige Kerberos-Verfahren

**Abbildung 3.2**



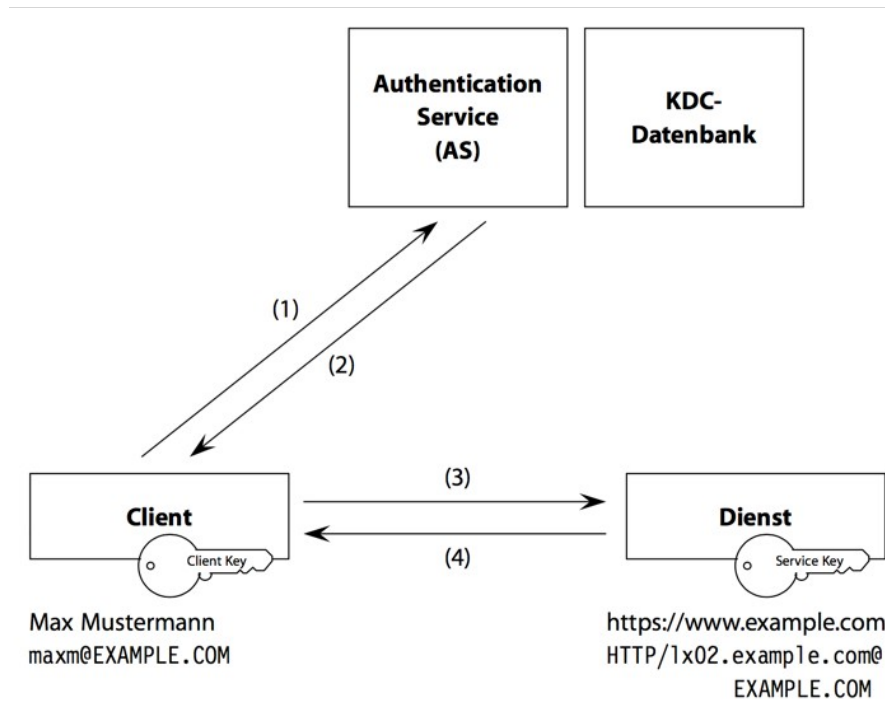


Abbildung 3.2 zeigt eine Variante des Kerberos-Verfahrens. Dort sind die drei Teilnehmer des Authentisierungsaustauschs dargestellt: Client, Dienst und KDC. Letzteres besteht hier aus der KDC-Datenbank und dem sogenannten *Authentication Service (AS)*. Ausserdem sind auch die Nachrichten, die diese Teilnehmer untereinander austauschen, mithilfe der Pfeile 1, 2, 3 und 4 abgebildet. Deren Richtung gibt an, wer eine Nachricht schickt und wer sie empfängt. Nachricht 1 wird beispielsweise vom Client an das KDC geschickt. Abbildung 3.3 gibt einen Überblick über die Inhalte dieser Nachrichten.



### Abbildung 3.3

Das hier beschriebene Verfahren wird auch als *einstufiges* Kerberos-Verfahren bezeichnet. Es ist eine Vorstufe des *zweistufigen* Verfahrens, das im nächsten Abschnitt beschrieben wird, und bietet im Gegensatz dazu noch kein vollständiges Single Sign-On (SSO). Das KDC besteht hier neben der KDC-Datenbank nur aus dem Authentication Service (AS).



### AS-REQ (1)

Client-Principal-Name
Dienste-Principal-Name

### AS-REP (2)

<b>Client-Teil:</b> Dienste-Principal-Name Session Key	
<b>Ticket:</b> Client-Principal-Name Session Key	

### AP-REQ (3)

<b>Ticket:</b> Client-Principal-Name Session Key	
<b>Authentikator:</b> Client-Principal-Name Zeitstempel	

Der AS erfüllt folgende Aufgaben:

- Der AS beantwortet Clientanfragen.
- Der AS erzeugt zufällige Session Keys.
- Der AS erzeugt Kerberos Tickets.

Das Protokoll läuft nun wie folgt ab: Zunächst schickt der Client in Abbildung 3.2 mit der Nachricht 1 eine Anfrage an das KDC, den *Authentication Service Request (AS-REQ)*. Neben einigen anderen Informationen, besteht der AS-REQ im Wesentlichen aus folgendem Inhalt:

- Dem Principal-Namen des Clients
- Dem Principal-Namen des Dienstes, auf den er zugreifen will

Dieser Inhalt ist in der Abbildung 3.3 dargestellt. Auf diese Anfrage reagiert der AS nun folgendermaßen: Zunächst erzeugt er einen zufälligen Sitzungsschlüssel (*Session Key*) für die beiden Principals (Client und Dienst). Dann sucht er deren beiden Langzeitschlüssel (*Client Key* und *Service Key*) in der KDC-Datenbank. Aus diesen Informationen konstruiert er die Nachricht 2 aus Abbildung 3.2: den *Authentication Service Reply (AS-REP)*. Der AS-REP ist eine Datenstruktur, die verschiedene verschlüsselte und unverschlüsselte Anteile enthält. Sein Inhalt besteht aber im Wesentlichen aus zwei verschlüsselten Anteilen, die in der Abbildung 3.3 unten dargestellt sind:

- Der erste Teil besteht aus dem Principal-Namen des Dienstes und dem Session Key, beides verschlüsselt mit dem Langzeitschlüssel des Clients (*Client Key*). Dieser Teil des AS-REP ist für den Client bestimmt, er soll hier als *Client-Teil* bezeichnet werden. Neben dem AS kennt nur der Client den Client Key und kann den Inhalt des Client-Teils lesen.
- Der zweite Teil beinhaltet den Principal-Namen des Clients und den Session Key, beides verschlüsselt mit dem Langzeitschlüssel des Dienstes (*Service Key*). Dieser Teil des AS-REP ist das *Ticket* für den Dienst. Im vorliegenden Beispiel also das HTTP-Ticket für

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 18/64

www.example.com. Neben dem AS kennt nur der Dienst selbst den Service Key. Daher kann nur der Dienst den Inhalt des Tickets lesen.

Wenn der Client die Nachricht 2 erhält, entschlüsselt er den Client-Teil mit dem Client Key. Hat er diesen Langzeitschlüssel nicht zur Hand, so fragt er jetzt den Anwender nach dessen Passwort, welches in einen Client Key umgewandelt wird.

Nach der Entschlüsselung des Client-Teils ist ein wichtiger Schritt getan: Der Client kennt jetzt den Session Key.

Mit diesem Session Key kann der Client nun einen *Authentikator* konstruieren. Die Kerberos-Authentikator basiert auf einem verschlüsselten Zeitstempel. Neben einem aktuellen Zeitstempel des Clientsystems enthält der Authentikator auch den Client-Principal-Namen. Der Client verschlüsselt diese beiden Informationen mit dem Session Key und erhält so einen Authentikator.

Nun hat der Client alles vorbereitet, um auf den Dienst zugreifen zu können. Der kerberisierte Zugriff des Clients auf den Service heisst *Application Server Request (AP-REQ)* und ist in Abbildung 3.2 oben mit der Nummer 3 markiert. Der **AP-REQ** enthält das Ticket und den Authentikator.

Erhält der Dienst die Nachricht 3, so kann er zuerst das Ticket entschlüsseln. Damit ist ein weiterer wichtiger Schritt getan: Der Dienst kennt nun ebenfalls den Session Key. Ausser dem Dienst, dem Client (und dem vertrauenswürdigen KDC) kennt niemand diesen Session Key. Er dient Client und Dienst also als gemeinsames Geheimnis. Der Dienst kann den Client authentifizieren, wenn ihm dieser seine Kenntnis des Session Key beweisen kann.

Dieser Beweis erfolgt über den Authentikator, den der Client ja bereits mitgeliefert hat. Mit dem Session Key kann der Dienst den Authentikator zunächst entschlüsseln. Findet er darin den Principal-Namen des Clients und einen Zeitstempel, so kann er den Client als authentisch betrachten, denn ein Dritter ohne Kenntnis des Session Key hätte den Authentikator nicht erzeugen können. Vorher überprüft der Dienst auch noch, ob der Zeitstempel innerhalb der Clock Skew aktuell ist. Falls der Client auf gegenseitige Authentisierung besteht, so schickt der Dienst dem Client mit Nachricht 4 in Abbildung 3.2 ebenfalls einen Authentikator. Diese Nachricht heisst *Application Service Reply (AP-REP)*.

Eine Sache gilt es noch zu beachten: Angreifer könnten Nachricht 3 auf dem Netzwerk mithören und das erlauschte Ticket-/Authentikator-Paar später selbst verwenden. Gelingt das innerhalb der Clock Skew, so wäre dadurch ein *Replay-Angriff* erfolgreich. Damit das nicht geschehen kann, muss sich der Dienst die bereits akzeptierten Authentikatoren merken. Dazu speichert er diese in einem sogenannten *Replay Cache* und prüft neue Authentikatoren gegen diesen Cache.

Damit ist der Vorgang der Authentisierung mit dem einstufigen Kerberos-Verfahren abgeschlossen.

Das hier beschriebene Verfahren hat folgende Eigenschaften: Wegen der Kurzlebigkeit der Kerberos Credentials darf der Client das Ticket zusammen mit dem Session Key in seinem Credential Cache ablegen. Damit kann er den Vorgang 3 innerhalb der Dauer der Ticket-Gültigkeit beliebig oft neu anstossen und somit auf den Service zugreifen, ohne dabei nochmals das Passwort vom Anwender zu erfragen. Ein erster Schritt in Richtung Single Sign-On ist also getan.

Die langlebigen Authentisierungsmerkmale der Nutzer (Passwörter, Client Keys) dürfen nur auf besonders gesicherten System (den KDCs) gespeichert werden. Passwörter werden dabei nicht über das Netz übertragen. Die Tickets für einen Dienst kann man innerhalb ihrer Lebensdauer wiederholt verwenden und muss somit nicht jedes Mal sein Passwort eingeben, wenn man auf diesen Dienst zugreifen will.

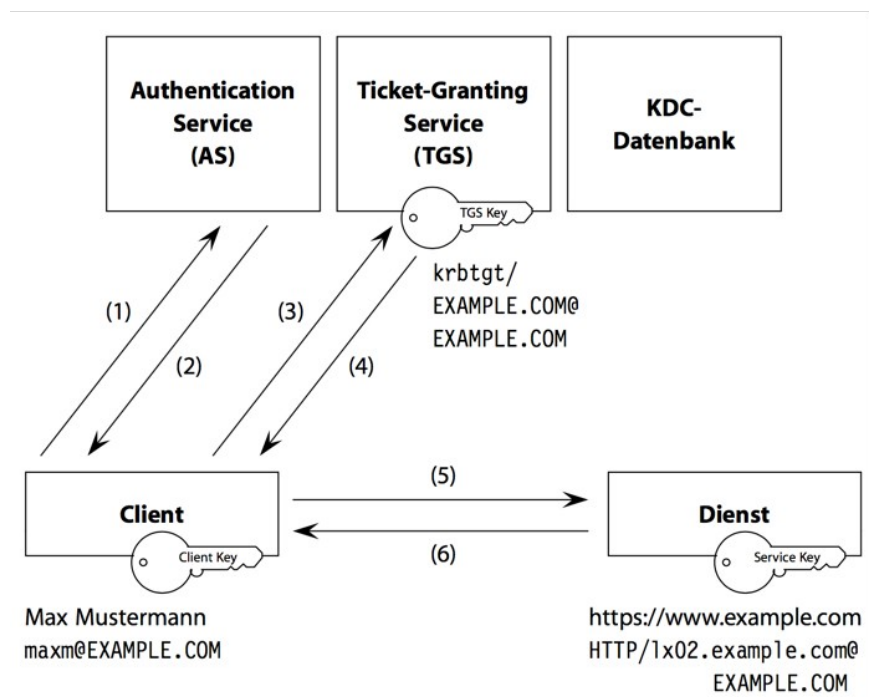
Der Client kann zusätzlich eine gegenseitige Authentisierung verlangen. Dann muss der Dienst seinerseits einen Authentikator konstruieren und mit Nachricht 4 an den Client schicken.

In der Realität kommt dieses Verfahren so allerdings eher selten zum Einsatz. Will ein Anwender nämlich auf weitere Dienste zugreifen, so müsste er das Prozedere 1-3 jeweils

nochmals durchlaufen. Für jeden weiteren Dienst müsste er also insbesondere nochmals sein Passwort angeben. Das gewünschte Single Sign-On, bei dem das Passwort ja nur einmal pro Sitzung einzugeben ist, wird durch das einstufige Verfahren also noch nicht vollständig erfüllt. Letzteres liefert das zweistufige Verfahren und ein spezieller Dienst, der analog zum AS Tickets vergeben kann.

### 3.3 Das zweistufige Kerberos-Verfahren

Abbildung 3.4



In Abbildung 3.4 ist das zweistufige Verfahren dargestellt. Im Gegensatz zu Abbildung 3.2 fällt hier auf, dass das KDC eine weitere Komponente enthält. Dabei handelt es sich um den **Ticket-Granting Service (TGS)**. Der TGS ist ein kerberisierter Dienst: Er besitzt einen Principal-Namen, der hier **krbtgt/EXAMPLE.COM@EXAMPLE.COM** lautet (im Allgemeinen: **krbtgt/REALM@REALM**) und ist mit kryptografischen Langzeitschlüsseln ausgestattet. Der Langzeitschlüssel des TGS wird im Folgenden als **TGS Key** bezeichnet. Wie auf den HTTP-Dienst aus Abbildung 3.2 kann man auch auf den TGS mit Kerberos Credentials zugreifen. Das Ticket für den TGS hat einen speziellen Namen: Es heisst **Ticket-Granting Ticket (TGT)**.

Anders als der HTTP-Dienst aus Abbildung 3.2 ist der TGS aber auch ein Teil des KDC und hat wie der AS Zugriff auf die KDC-Datenbank. Somit kann er die gleichen Aufgaben übernehmen wie der AS. Auch der TGS erfüllt also folgende Aufgaben:

- Der TGS beantwortet Clientanfragen.
- Der TGS erzeugt zufällige Session Keys.
- Der TGS erzeugt Kerberos Tickets.

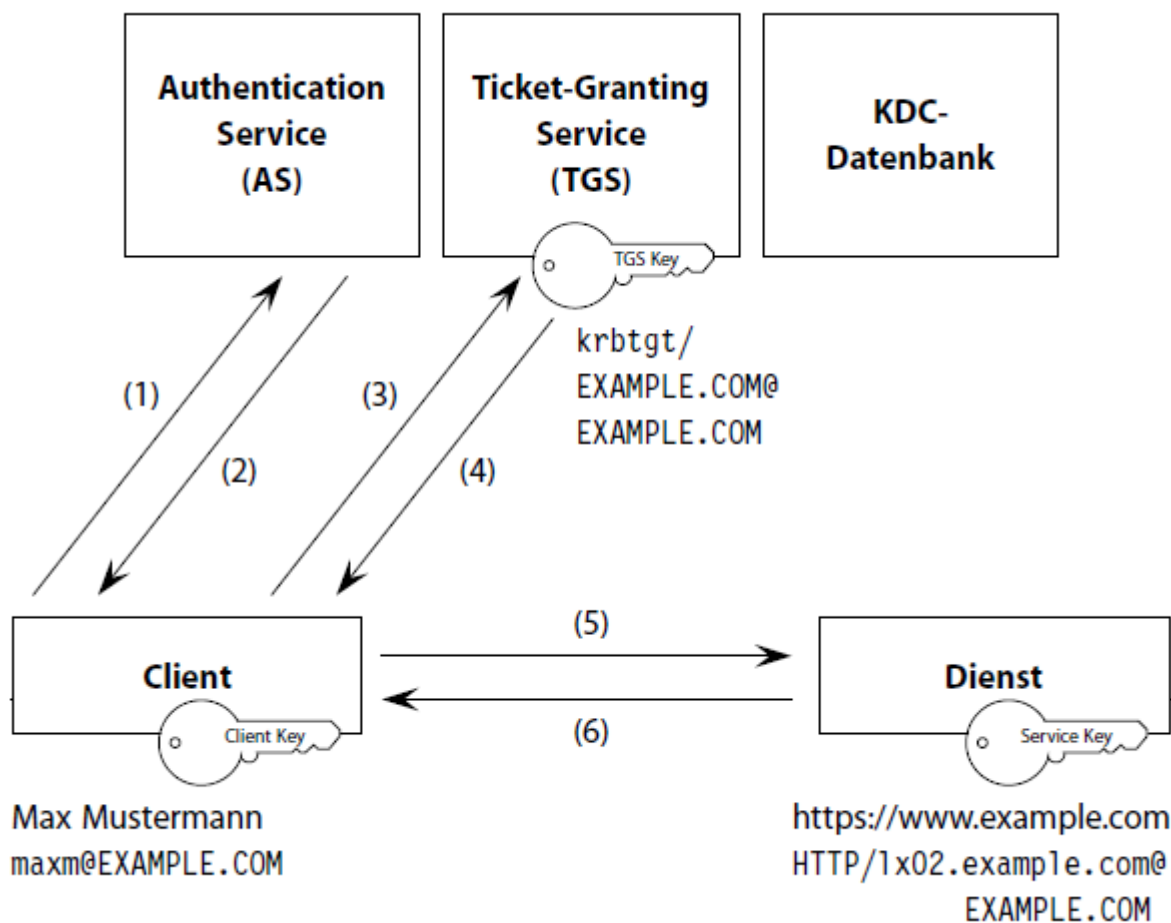
Der einzige Unterschied zum AS ist, dass Clients auf den TGS kerberisiert zugreifen und dazu ein TGT benötigen, während sie für den AS den Client Key (oder das Passwort) brauchen. Um

M159\_SKRIPT\_THEMA1\_KERBEROS\_V2.6.ODT

Single Sign-On zu realisieren, holen sich Clients also zunächst vom AS ein TGT – wozu einmal das Passwort erforderlich ist. Weitere Tickets bekommen sie dann vom TGS – **ohne weitere Anwenderinteraktion**.

### 3.3.1 Ablauf des zweistufigen Kerberos-Verfahrens:

Das zweistufige Verfahren entspricht dem folgenden Bild mit den Schritten (1) bis (6).



#### AS\_REQ (1) und AS\_REP (2)

Der Client aus Abbildung 3.4 greift zunächst wieder auf den AS zu. Anstatt aber direkt von diesem ein Ticket für den HTTP-Dienst anzufordern, bezieht er erst ein TGT.

Die AS-REQ-Nachricht (1 aus Abbildung 3.4) besteht im Wesentlichen aus folgendem Inhalt:

dem Principal-Namen des Clients (`maxm@EXAMPLE.COM`) und dem Principal-Namen des TGS (`krbtgt/EXAMPLE.COM@EXAMPLE.COM`).

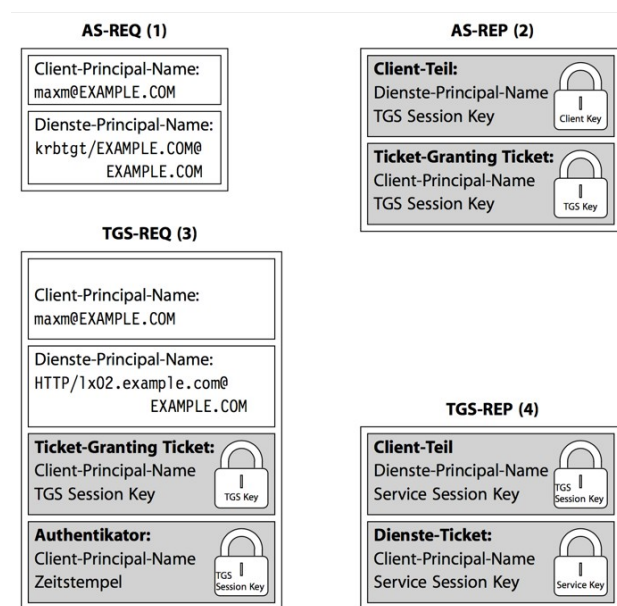
Der AS erzeugt nach Erhalt dieser Nachricht einen neuen zufälligen Session Key, der hier als *TGS Session Key* bezeichnet werden soll.

Der Inhalt des AS-REP (2 in Abbildung 3.4) besteht wiederum aus zwei verschlüsselten Anteilen, die in Abbildung 3.5 (rechts oben) im Detail dargestellt sind:

- Der erste Teil besteht aus dem Principal-Namen des **Ticket-Granting Service** und dem **TGS Session Key**, beides verschlüsselt mit dem Langzeitschlüssel des Clients (Client Key). Dieser Teil des **AS-REP ist der Client-Teil**.
- Der zweite Teil beinhaltet den Principal-Namen des Clients und den **TGS Session Key**, beides verschlüsselt mit dem Langzeitschlüssel des Dienstes (TGS Key). Dieser Teil des **AS-REP ist das Ticket-Granting Ticket (TGT)**.

Wenn der Client die Nachricht 2 erhält, entschlüsselt er den Client-Teil mit dem Client Key und gelangt somit an den **TGS Session Key**. An dieser Stelle ist wiederum das Passwort des Anwenders gefragt, es sei denn, der Client Key würde dem Client bereits vorliegen.

**Abbildung 3.5**



Den **TGS Session Key** und **das Ticket-Granting Ticket** speichert der Client in seinem **Credential Cache**. Der Inhalt des Credential Cache werden wir später anschauen mit dem Befehl `klist`.

Bisher verlief alles analog zum einstufigen Verfahren, nur dass es sich beim Dienst hier um den TGS gehandelt hat.

### **TGS\_REQ (3) und TGS\_REP (4)**

**Möchte der Client nun auf einen kerberisierten Dienst zugreifen** (Web-Service: `http/1x02.example.com@EXAMPLE.COM` im Beispiel), so kann er das dafür nötige HTTP-Ticket vom TGS beziehen. Für den Zugriff auf den TGS muss er zunächst einen **Authentikator** erzeugen. Dazu verschlüsselt er seinen Principal-Namen und einen aktuellen Zeitstempel **mit dem TGS Session Key**.

Die Anfrage des Clients an den TGS heisst **Ticket-Granting Server Request (TGS-REQ)** und ist in Abbildung 3.4 die Nachricht 3. Der Inhalt dieser Nachricht ist in Abbildung 3.5 (unten links) dargestellt. Analog zum AS-REQ enthält er den Namen des Clients, der die Anfrage stellt (`maxm@EXAMPLE.COM`), sowie den Namen des Dienstes, für den das Ticket ausgestellt werden soll (`http/1x02.example.com@EXAMPLE.COM`).

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 22/64

Da es sich anders als beim AS-REQ beim TGS-REQ um einen kerberisierten Zugriff auf das KDC handelt, **enthält der TGS-REQ auch ein Ticket (das TGT) und den Authentikator.**

Erhält der TGS den TGS-REQ, so prüft er das TGT und den Authentikator. Er tut dies analog zum Dienst aus Abschnitt 3.2. **Passt alles, dann hat der TGS den Client authentifiziert. Er erstellt dann einen neuen Session Key für Client und Dienst.** Dieser soll hier **Service Session Key** genannt werden. Ausserdem entnimmt der TGS der KDC-Datenbank den **Langzeitschlüssel des Dienstes** (den **Service Key**).

Die Antwort des TGS an den Client heisst **Ticket-Granting Service Reply (TGS-REP)**. Sie ist in Abbildung 3.5 im Detail dargestellt. Der Inhalt des TGS-REP ähnelt dem des AS-REP. Auch der **TGS-REP** besteht aus zwei verschlüsselten Anteilen:

- Dem *Client-Teil*, also dem Principal-Namen des Dienstes (`HTTP/1x02.example.com@EXAMPLE.COM`) und dem Service Session Key. Dieser Inhalt ist mit dem TGS Session Key verschlüsselt.
- Dem *Service Ticket*, also dem Principal-Namen des Clients (`maxm@EXAMPLE.COM`) und dem Service Session Key, die beide analog zum AS-REP ist mit dem Langzeitschlüssel des Dienstes (Service Key) verschlüsselt sind.

Anders als beim AS-REP ist der Client-Teil im TGS-REP also nicht mit dem Langzeitschlüssel des Clients, sondern mit **einem Kurzeitschlüssel verschlüsselt: dem TGS Session Key. Dies ist der entscheidende Trick.** Denn der Client kann nun ohne weitere Nutzerinteraktion die Entschlüsselung durchführen. Den dazu nötigen TGS Session Key findet er in seinem Credential Cache.

**Durch die Entschlüsselung des Client-Teils gelangt der Client nun auch an den Service Session Key.** Diesen kann er zusammen mit dem Dienste-Ticket in seinem Credential Cache ablegen und damit auch völlig analog zu Abschnitt 3.2 auf den Dienst zugreifen (AP-REQ/AP-REP). Gegenüber dem einstufigen Kerberos-Verfahren hat der Client nun den Vorteil, dass er auf weitere kerberisierte Netzwerkdienste zugreifen kann, ohne dabei 1 und 2 erneut durchlaufen zu müssen. Für 3 und 4 ist keine Anwenderinteraktion nötig; **das einmal beim Erhalt der Nachricht 2 eingegebene Passwort genügt. Single Sign-On ist also möglich.**

Nun hat der Client alles vorbereitet, um auf den Dienst zugreifen zu können.

## AP\_REQ (5) und AP\_REP (6)

Der kerberisierte Zugriff des Clients auf den Service heisst Application Server Request (**AP\_REQ**).

Der AP\_REQ und AP\_REP ist identisch mit dem einstufigen Kerberos Verfahren. Siehe vorhergehendes Kapitel.

Zur Wiederholung:

Der AP\_REQ enthält das Service-Ticket und den Authentikator. Der Service kann das Service-Ticket entschlüsseln mit seinem Server-Schlüssel und den Service-Session-Key lesen. Den Service-Session-Key kennen nur der Client, der Service selber und der KDC.

Mittels Service-Session-Key kann der Service dann den Authentikator entschlüsseln und die Identität des Clients ermitteln. Damit hat der Client seine Authentizität bewiesen weil nur er den Service-Session-Key kennen kann.

Der letzte Schritt des **AP\_REP** ist optional. Dieser findet statt, wenn der Client sicher sein will, dass er mit dem «richtigen» Server spricht. Der Server kann die Meldung (Timestamp) nur korrekt verschlüsseln, wenn dieser den Service-Session-Key kennt, welcher er mit seinem privaten Server-Langzeitschlüssel aus dem Service Ticket des AS\_REQ extrahiert hat.



Modul 159	Skript Directory Services Teil 1 – Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 23/64

### 3.4 Zusammenfassung

Die grundsätzliche Funktionsweise der Kerberos-Authentisierung ist geklärt. **Anstatt Langzeitschlüssel oder Passwörter über das Netzwerk zu senden, kommt Kerberos mit verschlüsselten Nachrichten aus.** Der Kerberos-Server (KDC) übermittelt dem Client Principal und dem Service Principal ein gemeinsames Geheimnis in Form eines Session Key. Diese Übertragung ist mit den jeweiligen Langzeitschlüsseln verschlüsselt. **Die beiden Principals beweisen sich dann gegenseitig ihre Kenntnis dieses gemeinsamen Geheimnisses und authentisieren sich somit.** Ab nun werden kurzlebige Authentisierungsmerkmale eingesetzt, die wiederum zwischengespeichert werden können, um wiederholt auf Dienste zugreifen zu können. Langzeitschlüssel oder Passwörter müssen vom Client dafür nicht gespeichert werden.

**Die gesamte Kommunikation mit dem KDC erledigt der Client. Eine Kommunikation zwischen dem Dienst und dem KDC findet nicht statt.** Das KDC muss für den Dienst also während der Client-Authentifizierung nicht erreichbar sein.

Es gibt zwei Varianten, wie ein Client Tickets beziehen werden kann. Eine davon ist die Verwendung des Authentication Service (AS), bei der anderen bezieht der Client die Tickets vom Ticket-Granting Service (TGS). Beide Dienste sind sich relativ ähnlich, der TGS ist aber ein kerberisierter Dienst.

Der Client benötigt seinen Langzeitschlüssel (also in der Regel das Anwenderpasswort), um den Inhalt der Antworten des AS zu entschlüsseln. **Für die Antworten des TGS braucht der Client nur einen Kurzzeitschlüssel aus seinem Credential Cache.**

Auch wenn man über den AS prinzipiell beliebige Dienste-Tickets beziehen kann, setzt man ihn meistens für TGTs ein. Erst dadurch wird Single Sign-On möglich.

Die bisherige Darstellung hat sich auf die wesentlichen Elemente der Nachrichten zwischen den Authentisierungspartnern beschränkt und auf Details verzichtet.

#### **Anmerkung:**

**Konsultieren Sie zu diesem Thema auch das Zusatzblatt zum Thema Kerberos. Dieses ist auf Laufwerk M: verfügbar**

Lösen Sie die Kerberos-Aufgabe:

03-Kerberos\_Messages\_mitKeys\_AUFGABE.xlsx mit beiliegender Aufgabe in einem txt-File

## 4 Einstieg in LDAP

Das KDC benötigt für seine Daten eine Datenbasis. Diese Datenbasis wird in einer produktiven Umgebung in der Regel von einem LDAP-Dienst zur Verfügung gestellt. In unseren praktischen Aufgaben im Teil 2, werden wir dazu den integrierten LDAP-Dienst von Samba verwenden. Damit wir damit arbeiten können, benötigen wir noch gewisse Grundkenntnisse im Umgang mit Clientkommandos. Das folgende Kapitel führt uns in dieses Thema ein. Sie können dieses als Erstes „überfliegen“ und darauf zurückkommen, wenn wir ldap-Kommandos und ldif-Dateien erstellen müssen.

Hier wird hauptsächlich der Umgang mit den OpenLDAP- Kommandozeilenwerkzeugen beschrieben, wobei ein wesentlicher Punkt in dem Datenformat LDIF besteht.

Wie wir mit SQL das Erstellen von Tabellen, Abfragen, Inserts, etc im Zusammenhang mit relationalen Datenbanken gelernt haben, geht es hier darum zu lernen, wie wir mit einem Verzeichnisdienst (=Directory Service) umgehen.

### 4.1 LDAP

#### 4.1.1 Das LDAP-Datenmodell

LDAP-Verzeichnisse speichern ihre Daten in Form von *Objekten*. Da diese Objekte hierarchisch angeordnet sind, entsteht eine baumartige Struktur, der **Directory Information Tree (DIT)**. Der Baum hat eine Wurzel (root), Knoten und Blätter (leaves).

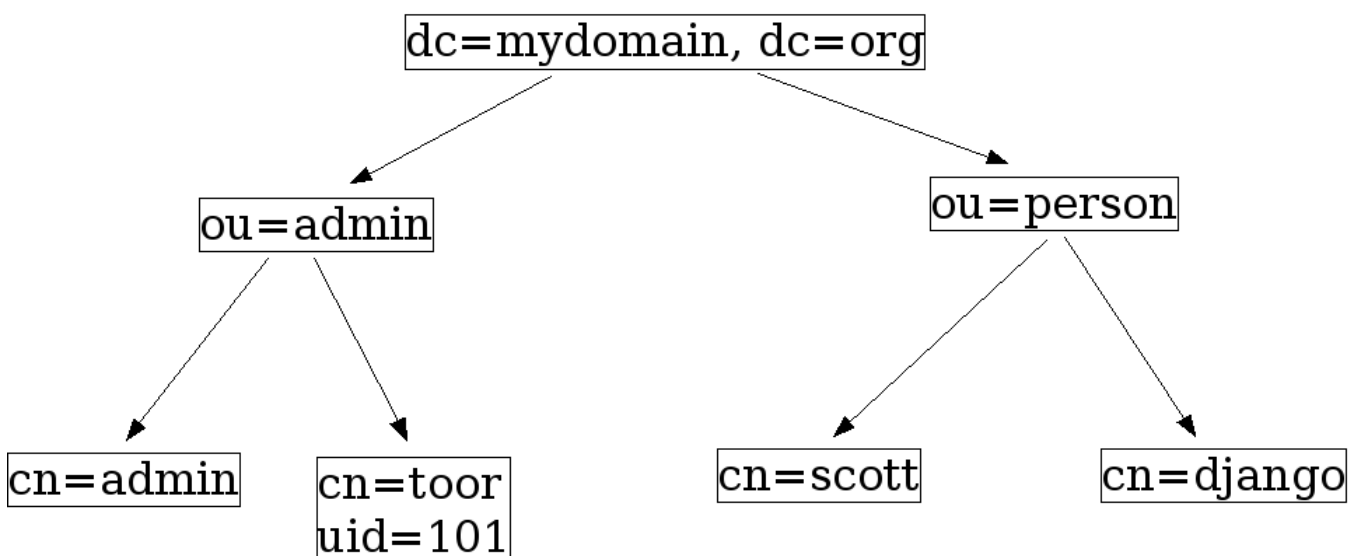
Ein DIT könnte etwa so aussehen:

Exemple de DIT

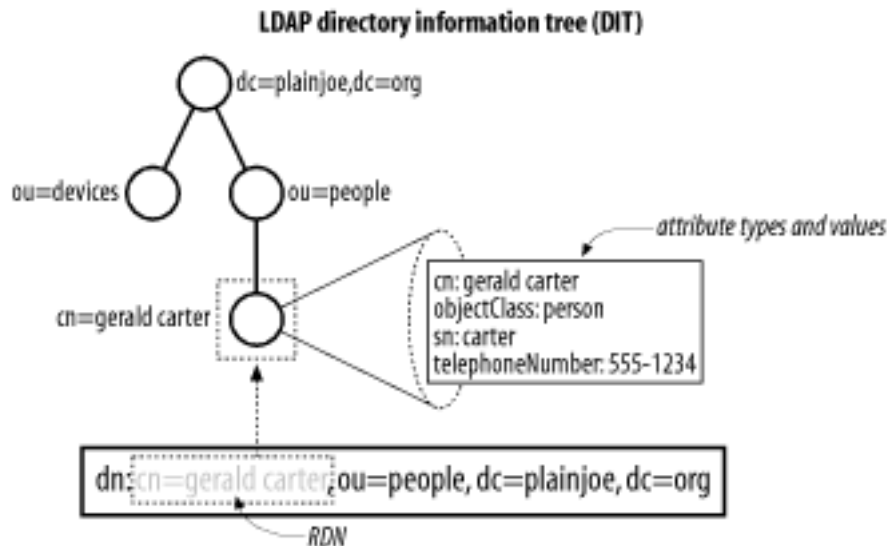
o=mydomain, c=fr notation X500

dc=mydomain.org

dc=mydomain, dc=org notation rfc 2247



Jedes LDAP-Objekt besitzt **Attribute**, denen Werte zugeordnet sein können. Ein Attribut kann einen oder mehrere Werte haben. Hier sehen sie beispielhaft die Attribute des Users Gerald Carter:



LDAP-Objekte gehören einer oder mehreren **Objekt-Klassen** an, die festlegen, welche Attribute das Objekt haben kann und welche es haben muss.

Jedes **LDAP-Objekt** hat einen Namen, den **Distinguished Name oder kurz DN**. Dieser bezeichnet das Objekt eindeutig im gesamten DIT.

LDAP-Objekte kann man sich als Datenstrukturen vorzustellen, die aus dem DN, den Objektklassen und einer Reihe von Attributen bestehen. Sehen Sie sich das folgende Listing für die Benutzerobjekte Max Mustermann und Erika Musterfrau an. cn=Max Mustermann und cn=Erika Musterfrau sind übrigens die kurzen Namen dieser Objekte, die sogenannten **Relative Distinguished Names (RDN)**. Die Attribute im Beispiel sind cn (Common Name), sn (Surname, Nachname) und description (Beschreibung).

#### 4.1.2 LDIF-Repräsentation von LDAP-Daten

Die Objekte eines LDAP Verzeichnisses können als LDIF File exportiert werden.

Bei LDIF handelt es sich um ein einfaches Ascii-Textformat. Dabei gelten folgende Regeln:

- Leerzeilen trennen einzelne Objekte
- Die erste Zeile in einem Objektblock muss mit dn: beginnen
- Kommentarzeilen werden durch das Zeichen # eingeleitet
- 7-Bit-ASCII-Codierung

Als Beispiel können die beiden Objekte in folgendem Listing dienen. Die Beschreibung jedes Objektes beginnt hier mit einer Kommentarzeile. Diese ist allerdings optional. Die erste Zeile in einem Objektblock muss mit dn: beginnen und den DN des Objektes angeben.

Danach folgt eine Reihe von Attribut-Wert-Paaren. Hier sind neben den eigentlichen Daten auch die Objektklassen als Werte des Attributs objectClass enthalten.

```
version: 1
# Max Mustermann
dn: cn=Max Mustermann, ou=people, dc=example, dc=com
objectClass: top
objectClass: person
cn: Max Mustermann
sn: Mustermann

# Erika Musterfrau
dn: cn=Erika Musterfrau, ou=people, dc=example, dc=com
objectClass: top
objectClass: person
```

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 26/64

```
cn: Erika Musterfrau
sn: Musterfrau
description::
RGlIcyBpc3QgZWluIEJlaXNwaWVsIGVpbmVzIEJlbnV0emVyY2JqZWt0ZXMGZnVlciBkYXMGTW9kdWwgMTU5Cg==
```

Von besonderem Interesse im Beispiel aus oberem Listing ist das Attribut `description` bei Frau Musterfrau. Der Wert dieses Attributs lautet eigentlich „Dies ist ein Beispiel eines Benutzerobjektes für das Modul 159“

Beweis:

```
echo RGlIcyBpc3QgZWluIEJlaXNwaWVsIGVpbmVzIEJlbnV0emVyY2JqZWt0ZXMGZs08ciBkYXMGTW9kdWwgMTU5Cg== | base64 -d
Output:
Dies ist ein Beispiel eines Benutzerobjektes für das Modul 159
```

Da darin aber der Umlaut »ü« vorkommt, lässt sich dieser Wert nicht in 7-Bit-ASCII-Codierung darstellen. LDIF sieht für solche Fälle die Codierung als BASE64-String vor, so wie das im Listing oben zu sehen ist. Dass der Wert als BASE64-String zu interpretieren ist, erkennt man übrigens an den zwei Doppelpunkten »::«, die dem Attributnamen `description` folgen.

## 4.2 OpenLDAP-Tools

### 4.2.1 Suchen mit `ldapsearch`

LDAP-Suchen lassen sich auf der Kommandozeile mit `ldapsearch` durchführen. `ldapsearch` ist eines der Werkzeuge der OpenLDAP-Software. Der Aufruf von `ldapsearch` geschieht in der Regel mit einer Reihe von Parametern, von denen die wichtigsten im Folgenden behandelt werden sollen.

- Mit dem Parameter `-h` kann man den *Hostnamen* des LDAP- Servers angeben, den man befragen möchte.
- Alternativ zu `-h` kann man mit `-H` eine LDAP-URI angeben (Beispiel: `ldap://vmLS4.m159.iet-gibb.ch`).
- mit `-b` gibt man die *Suchbasis* an, das ist der Punkt in der hierarchischen LDAP-Datenstruktur, unterhalb dem die Suche beginnen soll. Angegeben wird hier der DN dieses Objektes.
- Optional kann man einen *Suchfilter* anführen. Ohne diese Angabe verwendet OpenLDAP den Suchfilter (`objectClass=*`); dieser Filter trifft auf sämtliche Objekte zu (mehr zu Suchfiltern s.u.).
- Optional kann man eine Liste der Attribute angeben, nach denen man sucht. Ansonsten werden alle Attribute zurückgeliefert, auf die man Leseberechtigung hat.

Nehmen sie folgendes Beispiel-Objekt:

```
# neues Objekt anlegen
dn: cn=Erika Musterfrau,ou=people,dc=example,dc=com
changetype: add
cn: Erika Musterfrau
sn: Musterfrau
objectClass: top
objectClass: person

# ein Attribut hinzufügen
dn: cn=Erika Musterfrau,ou=people,dc=example,dc=com
changetype: modify
add: seeAlso
```

```
seeAlso: cn=Max Mustermann,ou=people,dc=example,dc=com
```

Eine LDAP-Suche nach diesem Objekt könnte so wie in folgendem Listing dargestellt aussehen.

Der Suchfilter lautet hier (`cn=Erika*`). Das Zeichen `*` ist eine sogenannte Wildcard, damit werden alle Objekte gefunden, deren `cn` mit `Erika` beginnt. Der Parameter `-x` wird in im Abschnitt „Authentisierung“ behandelt.

Beispiel einer LDAP-Suche:

```
root@vmLS4:~# ldapsearch -x -h vmLS4 -b dc=example,dc=com '(cn=Erika*)'
dn: cn=Erika Musterfrau,ou=people,dc=example,dc=com
cn: Erika Musterfrau
sn: Musterfrau
objectClass: top
objectClass: person
description: RWluIEJlaXNwaWVsIGVpbmVzIEJlbnV0emVyY2JqZWt0ZXMgZs
O8ciBkYXMgS2VyYmVyY3MtQnVjaAo=
seeAlso: cn=Max Mustermann,ou=people,dc=example,dc=com
```

## 4.2.2 Suchfilter

Suchfilter sind grundsätzlich in runden Klammern anzugeben und können kombiniert werden. Dazu stehen logische Operatoren zur Verfügung: `&` (*und*), `!` (*nicht*) und `|` (*oder*). Etwas gewöhnungsbedürftig ist, dass der Operator nicht zwischen sondern **vor** den Operanden steht. Man schreibt nicht „A AND B“ sondern „AND A B“. Mit dem Symbol `&` für AND. Das folgende Listing zeigt ein weiteres Beispiel. Durch den dort angegebenen Suchfilter wird eine Suche nach allen Objekten ausgeführt, die der Objektklasse `person` angehören und bei denen das Attribut `seeAlso` einen beliebigen Wert aufweist. Durch die zusätzliche Angabe von `cn` auf der Kommandozeile wird lediglich das `cn`-Attribut zurückgeliefert.

Beispiel einer LDAP-Suche mit Attributauswahl

```
root@vmLS4:~# ldapsearch -x -h vmLS4 -b dc=example,dc=com '(&(objectClass=person)(seeAlso=*))' cn
dn: cn=Erika Musterfrau,ou=people,dc=example,dc=com
cn: Erika Musterfrau
```

Wie Sie an diesen Beispielen sehen können, bedienen sich die Open-LDAP-Tools des LDIF-Formats; `ldapsearch` tut dies, um Suchergebnisse darzustellen.

## 4.2.3 Authentisierung

Die beiden LDAP-Suchen aus den zwei vorhergehenden Listings erfolgten anonym, der Client musste sich dabei also nicht authentisieren. Das LDAP-Protokoll sieht verschiedene Arten der Authentisierung vor. Bei den OpenLDAP-Kommandos wird durch die Option `-x` der sogenannte *Simple Bind* durchgeführt, bei dem der Client seinen Namen und ein zugehöriges Passwort an den LDAP-Server schickt. Ohne weitere Optionen geschieht dieser Simple Bind aber in der anonymen Form. Daneben kennt LDAP auch den *SASL Bind* (gängiges Protokoll zur Authentifizierung. Wird unter anderem auch von SMTP, IMAP, POP3, LDAP und XMPP benutzt), der bei OpenLDAP durch Weglassen der Option `-x` eingeleitet wird.

Für einen nicht anonymen Simple Bind sind weitere Parameter erforderlich: `-D` und einer der Parameter `-W`, `-w` oder `-Y`.

- `-D` gibt den Namen eines LDAP-Objektes (*Bind DN*) an. Dieser DN stellt die Identität des Clients («Benutzername als DN») dar, der die Suche durchführt.
- `-W` fordert `ldapsearch` dazu auf, das Passwort für den Simple Bind interaktiv abzufragen.
- `-w` verhält sich wie `-W`, allerdings wird das Passwort hierbei nicht interaktiv abgefragt, sondern direkt auf der Kommandozeile angegeben. (Nicht in produktiven Umgebungen verwenden!)
- `-y` dient schließlich der Angabe einer Datei, die das Passwort für den Simple Bind enthält.

Folgendes Listing zeigt ein Beispiel für einen authentisierten Simple Bind:

```
root@vmLS4:~# ldapsearch -x -h vmLS4 -D 'cn=Erika Musterfrau,ou= people,dc=example,dc=com' -w 'geheim123' -b dc=example,dc=com '( cn=Erika_ )'
dn: cn=Erika Musterfrau,ou=people,dc=example,dc=com
cn: Erika Musterfrau
sn: Musterfrau
objectClass: top
objectClass: person
description:: RWluIEJlaXNwaWVsIGVpbmVzIEJlbnV0emVyY2JqZWt0ZXMgZs
O8ciBkYXMgS2VyYmVyY3MtQnVjaAo=
seeAlso: cn=Max Mustermann,ou=people,dc=example,dc=com
userPassword:: Z2VoZWltMTIz
```

#### 4.2.4 Änderungen mit LDIF

Das LDIF Format kann auch für die Durchführung von Änderungen verwendet werden. Dazu ist zunächst der `changetype` wichtig, der pro Objekt in der Zeile nach dessen `dn` genannt werden muss. Folgende Werte von `changetype` sind möglich:

**changetype: add** führt dazu, dass ein Objekt dem Verzeichnis hinzugefügt wird.

**changetype: modify** führt Änderungen an einzelnen Attributen durch. Hierbei muss man noch genauer spezifizieren, welche Änderung man durchführen möchte:

**add: *Attributname*** fügt dem Objekt ein weiteres Attribut hinzu.

**delete: *Attributname*** löscht ein Attribut aus dem Objekt.

**replace: *Attributname*** ersetzt das genannte Attribut.

**changetype: modrdn** ändert den RDN. Diese Operation spielt hier allerdings keine Rolle.

**changetype: delete** führt dazu, dass ein Objekt aus dem Verzeichnis entfernt wird.

Das folgende Listing stellt die LDIF-Beschreibung mehrerer Änderungen dar. Zunächst wird das Objekt `cn=Erika Musterfrau` angelegt. Der `changetype` ist daher `add`. Diese LDAP-Operation beschreibt der erste Textblock im Listing.

Beispiel: Mehrere LDAP-Modifikationen in einer LDIF-Datei

```
# neues Objekt anlegen
dn: cn=Erika Musterfrau,ou=people,dc=example,dc=com
changetype: add
cn: Erika Musterfrau
sn: Musterfrau
objectClass: top
```

```
objectClass: person

# ein Attribut hinzufügen
dn: cn=Erika Musterfrau,ou=people,dc=example,dc=com
changetype: modify
add: seeAlso
seeAlso: cn=Max Mustermann,ou=people,dc=example,dc=com
```

Im nächsten Block wird eine Änderung am existierenden Objekt durchgeführt (`changetype: modify`). In der darauffolgenden Zeile wird zusätzlich spezifiziert, was geändert werden soll. `add: seeAlso` gibt an, dass dem Objekt das Attribut `seeAlso` hinzuzufügen ist. Die Zeile danach liefert den Wert für dieses neue Attribut.

Mit `changetype: modify` kann man auch mehrere Änderungen auf einmal durchführen, diese sind dann durch eine Zeile mit dem Zeichen »-« voneinander zu trennen. Das folgende Listing zeigt hierfür ein Beispiel. Diese LDIF-Datei enthält nur einen Textblock, beschreibt also nur Änderungen an einem Objekt (`cn=Erika Musterfrau`). Nach der Angabe des `changetype` folgt eine Zeile mit einer `delete`-Anweisung. Diese löscht das Attribut `seeAlso` von `cn=Erika Musterfrau`. Es folgen weitere Änderungen, getrennt durch »-«. Mit `replace` wird der Wert des Attributs `description` ersetzt und mit `add` erhält Frau Musterfrau noch ein Passwort.

#### Beispiel: mehrere Attribute auf einmal ändern

```
dn: cn=Erika Musterfrau,ou=people,dc=example,dc=com
changetype: modify
delete: seeAlso
-
replace: description
description: Eine Beispielanwenderin
-
add: userPassword
userPassword: geheim123
```

Und hier noch ein Beispiel wie man das Objekt wieder löschen kann:

```
# Objekt löschen
dn: cn=Erika Musterfrau,ou=people,dc=example,dc=com
changetype: delete
```

### 4.2.5 Weitere OpenLDAP-Befehle

Im letzten Abschnitt haben Sie eine knappe Einführung in die Handhabung von `ldapsearch` erhalten. Wie Sie gesehen haben, nutzt dieses Tool das LDIF-Format, um die Suchergebnisse darzustellen.

Wir wissen, dass man mit LDIF auch Änderungen an den LDAP-Daten durchführen kann. Dazu dient in erster Linie das Kommando `ldapmodify`, das im Prinzip die gleichen Parameter wie `ldapsearch` unterstützt. Nur die Angabe der Suchbasis mit `-b` kann hierbei entfallen und man benötigt zusätzlich den Parameter `-f`, um den Namen der LDIF-Datei anzugeben, die die durchzuführenden Änderungen beschreibt.

Nehmen wir folgendes Beispiel:

Hier wird ein Objekt angelegt und geändert. Die Beschreibung ist im LDIF-Format.



```
# neues Objekt anlegen
dn: cn=Erika Musterfrau,ou=people,dc=example,dc=com
changetype: add
cn: Erika Musterfrau
sn: Musterfrau
objectClass: top
objectClass: person

# ein Attribut hinzufügen
dn: cn=Erika Musterfrau,ou=people,dc=example,dc=com
changetype: modify
add: seeAlso
seeAlso: cn=Max Mustermann,ou=people,dc=example,dc=com
```

Mit `ldapmodify` können wir nun diese Änderungen direkt vornehmen, indem wir die Datei mit dem Inhalt von oben mit `-f` angeben.

```
ldapmodify -x -D cn=admin,dc=example,dc=com -w 'passwort' -f datei.ldif
```

```
adding new entry "cn=Erika Musterfrau,ou=people,dc=example,dc=com"
```

```
modifying entry "cn=Erika Musterfrau,ou=people,dc=example,dc=com"
```

Mit `ldapadd` können Sie ebenfalls neue Objekte hinzufügen, wobei die Angabe von `changeType: add` im LDIF nicht notwendig ist. Mit `ldapdelete` können Sie existierende Objekte löschen, wobei auch hier die Angabe von `changeType: delete` im LDIF nicht notwendig ist.

## 5 Aufbau einer eigenen Kerberos Infrastruktur

In diesem Kapitel wollen wir nach den Grundlagen von Kapitel 2 Kerberos aus Anwender-Sicht beleuchten. Dazu bauen wir uns eine Umgebung wie folgt:

In dieser Kerberos Realm **M159.IET-GIBB.CH** soll ein Anwender (Sie!) die Authentifizierung via Kerberos demonstrieren können. Sie zeigen, wie mit einer login shell auf **vmLP1** als Client einmal lokal und einmal via Kerberos (**KDC=vmLS4**) die Authentifizierung durchgeführt wird. Die **vmLS4** dient dabei auch als **DNS-Server**.

Von jetzt an arbeiten Sie mit **smartlearn.portable**

### 5.1 Vorarbeiten für die Beispielumgebung

Damit mit den Rechnern eine Internetverbindung aufgebaut werden kann, muss stets auch **vmLF1** gestartet werden !

Reihenfolge des Startens:

vmLF1 - Firewall

vmLS4 Ubuntu Server als KDC: 192.168.210.64

vmLP1 Ubuntu Client: 192.168.210.31

IP-Adressen sind bereits definiert. Sowohl vmLS4 als auch vmLP1 haben Internet-Zugang, sofern vmLF1 gestartet ist.

#### 5.1.1.1 OS aktualisieren

Führen Sie für vmLS4 und vmLP1 ein OS-Update durch:

```
sudo apt update
sudo apt upgrade
```

Achten Sie darauf, dass keine Fehlermeldungen erscheinen.  
**Akzeptieren Sie bei Fragen die Default-Antworten.**

Danach neu starten:

```
sudo reboot
```

Achtung: Ab Ubuntu-Version 18.x **netplan** verwenden und nicht /etc/network/interfaces!  
Siehe auch [https://www.thomas-krenn.com/de/wiki/Netzwerk-Konfiguration\\_Ubuntu\\_-\\_Netplan](https://www.thomas-krenn.com/de/wiki/Netzwerk-Konfiguration_Ubuntu_-_Netplan)

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 32/64

Im Folgenden wird die Netzwerkkonfiguration mit netplan gemacht.

### Für vmLS4:

Arbeiten Sie auf vmLS4 per ssh von vmLP1:

```
vmadmin@vmLP1:~$ ssh vmadmin@vmLS4
```

Ändern Sie das bestehende `/etc/netplan/00-eth0.yaml`, indem Sie bereits die `search-Domain` anpassen. Alles andere können Sie im Moment belassen. Später müssen wir dann den DNS-Server anpassen, sobald DNS auf vmLS4 aktiv ist.

```
nano /etc/netplan/00-eth0.yaml
```

```
network:
  ethernets:
    eth0:
      addresses:
        - 192.168.210.64/24
      nameservers:
        addresses:
          - 192.168.210.1
        search:
          - m159.iet-gibb.ch
      routes:
        - to: default
          via: 192.168.210.1
  version: 2
```

**Hinweis:** Bitte in dieser Datei Leerzeichen zum Einrücken verwenden, keine Tabs. Das Format des yaml-Files entspricht hier Ubuntu Version 22.04.

Danach:

```
sudo netplan apply
```

Damit werden die Einstellungen aktiviert. Achten sie darauf, dass sich beim Ausführen dieses Befehls nur ein \*.yaml im /etc/netplan befindet. Wenn sie mehrere yaml-Files haben, wird die Konfiguration nicht korrekt aktiviert.

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 33/64

### 5.1.1.2 Hostnamen ändern

Arbeiten Sie auf vmLS4 per ssh von vmLP1:

```
vmadmin@vmLP1:~$ ssh vmadmin@vmLS4
```

Aktueller hostname:

```
vmadmin@vmLS4:~$ hostname
vmLS4.smartlearn.lan
```

Der neue Hostname wird gesetzt mit:

```
vmadmin@vmLS4:~$ sudo nano /etc/hostname
Tragen Sie darin den FQDN ein: vmLS4.m159.iet-gibb.ch
```

Danach reboot und Kontrolle mit:

```
vmadmin@vmLS4:~$ hostname
vmLS4.m159.iet-gibb.ch
```

Nun editieren Sie */etc/hosts* und tragen hier ebenfalls den FQDN ein.

```
nano /etc/hosts
127.0.0.1 localhost
127.0.1.1 vmLS4 vmLS4.m159.iet-gibb.ch
192.168.210.64 vmLS4 vmLS4.m159.iet-gibb.ch

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Ändern Sie nun auch den Hostnamen der vmLP1 auf den FQDN vmLP1.m159.iet-gibb.ch auf die gleiche Weise wie vmLS4!

Kontrollieren Sie jeweils die Namen mit dem Befehl: `hostname` (Nach reboot)

### 5.1.2 DNS-Installation auf vmLS4

Unsere vmLS4 werden wir als KDC, LDAP-Server und auch als DNS-Server verwenden. In der Praxis wird das selten der Fall sein. Für unsere Testumgebung ist das aber vertretbar.

Starten Sie von der vmLP1 eine ssh-Session auf vmLS4 mit:

```
ssh vmadmin@192.168.210.64
```

Führen Sie anschliessend alle Befehle mit dieser Remote-Session aus.

Zuerst muss bind mit apt installiert werden.

```
sudo apt install bind9 bind9utils bind9-doc
```

Für die Konfiguration von bind stoppen wir bind9 :  
M159\_SKRIPT\_THEMA1\_KERBEROS\_V2.6.ODT

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 34/64

```
sudo systemctl stop bind9
```

### 5.1.2.1 Forward- und Reverse-Lookup Zonen anlegen

Zuerst muss in der Datei

**/etc/bind/named.conf.local**

die Konfiguration für unser Beispiel-Zone `m159.iet-gibb.ch` eingetragen werden.

In dieser Datei werden zuerst nur die Zonen für BIND bekannt gemacht. Die eigentliche Konfiguration wird dann in den jeweils unter `file` angegebenen Dateien eingetragen, siehe dazu folgende beiden Zonen:

**/etc/bind/db.m159.iet-gibb.ch**

Für den Reverse-Lookup wird noch eine weitere Zonen-Datei

**/etc/bind/db.192.168.210**

angelegt.

Die 3 Dateien finden Sie im Modulordner im Verzeichnis:

**/sh-modules/iet-159/03\_Scripts\_und\_Arbeitsblaetter/Thema\_1/CONFIG-Dateien/DNS-config-files/vmLS4**

**Kopieren Sie diese 3 ins /etc/bind/ Verzeichnis von vmLS4.**

### 5.1.2.2 Weitere Anpassungen

Jetzt müssen wir noch die Datei **/etc/bind/named.conf.options** anpassen. Hier werden unter anderem auch noch die DNS-Forwarders eingetragen.

Diese Datei finden Sie im Modulordner im Verzeichnis:

**/sh-modules/iet-159/03\_Scripts\_und\_Arbeitsblaetter/Thema\_1/CONFIG-Dateien/DNS-config-files/vmLS4**

**Kopieren Sie diese ins /etc/bind/ Verzeichnis von vmLS4.**

Damit die Rechner über ihren DNS-Server informiert sind, sollten Sie das `/etc/netplan/00-eth0.yaml` anpassen, damit neu die vmLS4 als DNS-Server verwendet wird. Machen Sie das auf vmLS4 und vmLP1.

Auch diese Datei finden Sie im Modulordner

**/sh-modules/iet-159/03\_Scripts\_und\_Arbeitsblaetter/Thema\_1/CONFIG-Dateien/DNS-config-files/**

So sieht die Datei

`/etc/netplan/00-eth0.yaml` auf vmLS4 aus:

```
network:
  ethernets:
    eth0:
      addresses:
        - 192.168.210.64/24
      nameservers:
        addresses:
          - 192.168.210.64
      search:
        - m159.iet-gibb.ch
      routes:
        - to: default
          via: 192.168.210.1
  version: 2
```

Anschliessend: `netplan apply` nicht vergessen.

### 5.1.2.3 Konfiguration auf vmLS4 überprüfen

Wurde die Konfiguration angelegt, kann diese mit dem Programm **named-checkconf** geprüft werden.

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 35/64

```
sudo named-checkconf -z
```

Das Ergebnis sollte etwa wie folgt aussehen:

```
root@vmls4:/etc/netplan# named-checkconf -z
zone m159.iet-gibb.ch/IN: loaded serial 2017082401
zone 210.168.192.in-addr.arpa/IN: loaded serial 2017082401
zone localhost/IN: loaded serial 2
zone 127.in-addr.arpa/IN: loaded serial 1
zone 0.in-addr.arpa/IN: loaded serial 1
zone 255.in-addr.arpa/IN: loaded serial 1
root@vmls4:/etc/netplan#
```

#### 5.1.2.4 DNS-Server neu starten

```
sudo systemctl restart bind9
```

Kontrolle mit:

```
sudo systemctl status bind9
```

```
root@vmls4:/etc/netplan# sudo service bind9 status
● bind9.service - BIND Domain Name Server
   Loaded: loaded (/lib/systemd/system/bind9.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2019-09-11 23:53:25 CEST; 8min ago
     Docs: man:named(8)
  Process: 7238 ExecStop=/usr/sbin/rndc stop (code=exited, status=0/SUCCESS)
 Main PID: 7272 (named)
    Tasks: 4 (limit: 401)
   CGroup: /system.slice/bind9.service
           └─7272 /usr/sbin/named -f -u bind

Sep 11 23:53:25 vmls4.m159.iet-gibb.ch named[7272]: managed-keys-zone: loaded serial 2
Sep 11 23:53:25 vmls4.m159.iet-gibb.ch named[7272]: zone 0.in-addr.arpa/IN: loaded serial 1
Sep 11 23:53:25 vmls4.m159.iet-gibb.ch named[7272]: zone 210.168.192.in-addr.arpa/IN: loaded serial 2017082401
Sep 11 23:53:25 vmls4.m159.iet-gibb.ch named[7272]: zone 127.in-addr.arpa/IN: loaded serial 1
Sep 11 23:53:25 vmls4.m159.iet-gibb.ch named[7272]: zone localhost/IN: loaded serial 2
Sep 11 23:53:25 vmls4.m159.iet-gibb.ch named[7272]: zone m159.iet-gibb.ch/IN: loaded serial 2017082401
Sep 11 23:53:25 vmls4.m159.iet-gibb.ch named[7272]: zone 255.in-addr.arpa/IN: loaded serial 1
Sep 11 23:53:25 vmls4.m159.iet-gibb.ch named[7272]: all zones loaded
Sep 11 23:53:25 vmls4.m159.iet-gibb.ch named[7272]: running
Sep 11 23:54:50 vmls4.m159.iet-gibb.ch named[7272]: resolver priming query complete
```

#### 5.1.2.5 DNS-Server durch DNS-Anfragen testen

Die Funktionalität des DNS-Servers kann unter Linux mit dem Tool **dig** getestet werden. Testen Sie DNS auf vmLS4 und vmLP1.

Forward-Lookup testen

```
dig +noall +answer vmls4.m159.iet-gibb.ch
```

Output:

```
vmadmin@vmLS4:/etc/bind$ dig +noall +answer vmls4.m159.iet-gibb.ch
vmls4.m159.iet-gibb.ch. 0          IN      A       127.0.1.1
vmls4.m159.iet-gibb.ch. 0          IN      A       192.168.210.64
```

```
dig +noall +answer vmlp1.m159.iet-gibb.ch
```

Output:

```
vmadmin@vmLS4:/etc/bind$ dig +noall +answer vmlp1.m159.iet-gibb.ch
vmlp1.m159.iet-gibb.ch. 172800   IN      A       192.168.210.31
```

Und auch externe DNS-Zonen müssen aufgelöst werden können, z. Bsp:

```
dig +noall +answer m121.ch
```

Output:

```
vmadmin@vmLP1:~/Dokumente$ dig +noall +answer m121.ch
m121.ch. 3600      IN      A       86.118.61.123
vmadmin@vmLP1:~/Dokumente$
```

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 36/64

Für alle Anfragen sollte der entsprechende DNS-Eintrag mit der korrekten IP-Adresse ausgegeben werden.

### Reverse-Lookup testen

```
dig +noall +answer -x 192.168.210.31
```

Output:

```
vmadmin@vmLS4:/etc/bind$ dig +noall +answer -x 192.168.210.31
31.210.168.192.in-addr.arpa. 86400 IN PTR vmLP1.m159.i-et-gibb.ch.
vmadmin@vmLS4:/etc/bind$
```

```
dig +noall +answer -x 192.168.210.64
```

Output:

```
31.210.168.192.in-addr.arpa. 86400 IN PTR vmLP1.m159.i-et-gibb.ch.
vmadmin@vmLS4:/etc/bind$ dig +noall +answer -x 192.168.210.64
64.210.168.192.in-addr.arpa. 0 IN PTR vmLS4.
64.210.168.192.in-addr.arpa. 0 IN PTR vmLS4.m159.i-et-gibb.ch.
vmadmin@vmLS4:/etc/bind$
```

**Sie dürfen erst weiterfahren, wenn der DNS-Server korrekt funktioniert. Die DNS-Tests müssen auf vmLS4 und vmLP1 ohne Fehler ausgeführt werden können. DNS ist für unsere Kerberos REALM lebenswichtig!**

### 5.1.3 Zeitsynchronisation

Rechnersysteme in einem Kerberos Realm dürfen nur eine kleine Zeitdifferenz haben. Dazu wird in der Praxis ntp (Network Time Protocol) verwendet.

In einer virtualisierten Umgebung wie smartlearn.portable muss dies nicht gemacht werden, da die vm's die Zeit mit dem Host synchronisieren. Die host's ihrerseits sind zeitsynchron.

## 6 Key Distribution Center von MIT Kerberos

In diesem Kapiteln lernen Sie MIT Kerberos kennen, indem Sie den Realm **M159.IET-GIBB.CH** aufbauen.

Sie starten hier mit der Einrichtung des KDC dieses Realm. Dazu gehört die Installation der erforderlichen Software und deren Konfiguration. Danach werden Sie eine initiale Principal-Datenbank erzeugen und darin zusätzliche Principals anlegen. Durch einen Master Key werden die Daten in der Datenbank abgesichert. Am Ende dieses Kapitels läuft das KDC und Sie können mit `kinit` die ersten Tickets beziehen.

Der Aufbau der Kerberos-Infrastruktur beginnt mit dem Linux-Server vmLS4. Sein voller DNS-Hostname (=FQDN) ist **vmLS4.m159.i-et-gibb.ch** und der Name des Kerberos Realm lautet **M159.IET-GIBB.CH**. Das entspricht der Konvention, als Realm den Namen der DNS-Domäne in Grossbuchstaben zu verwenden. Wie Sie bereits wissen, handelt es bei M159.IET-GIBB.CH um die Wurzel der Realm, welche wir hier aufbauen werden.



## 6.1 Installation Kerberos auf vmLS4

Beginnen Sie die Installation mit folgendem Befehl:

```
apt install krb5-user krb5-doc krb5-kdc krb5-admin-server krb5-kdc-ldap
```

### Configuring Kerberos Authentication

When users attempt to use Kerberos and specify a principal or user name without specifying what administrative Kerberos realm that principal belongs to, the system appends the default realm. The default realm may also be used as the realm of a Kerberos service running on the local machine. Often, the default realm is the uppercase version of the local DNS domain.

Default Kerberos version 5 realm:

M159.IET-GIBB.CH

<Ok>

### Configuring Kerberos Authentication

Enter the hostnames of Kerberos servers in the M159.IET-GIBB.CH Kerberos realm separated by spaces.

Kerberos servers for your realm:

vmLS4

<Ok>

### Configuring Kerberos Authentication

Enter the hostname of the administrative (password changing) server for the M159.IET-GIBB.CH Kerberos realm.

Administrative server for your Kerberos realm:

vmLS4

<Ok>

vmLS4 eingeben

### Configuring krb5-admin-server

Setting up a Kerberos Realm

This package contains the administrative tools required to run the Kerberos master server.

However, installing this package does not automatically set up a Kerberos realm. This can be done later by running the "krb5\_newrealm" command.

Please also read the /usr/share/doc/krb5-kdc/README.KDC file and the administration guide found in the krb5-doc package.

<Ok>

Damit sind Kerberos-Client, KDC und Dokumentation von MIT Kerberos installiert. Das Ubuntu-Installationsprogramm hat dabei auch schon Vorlagen für die Konfigurationsdateien von KDC

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 38/64

und Kerberos-Client erzeugt: `/etc/krb5kdc/kdc.conf` und `/etc/krb5.conf`.

Wenn Sie rot-markierte Fehlermeldungen nach der Installation beobachten, ist das kein Grund zur Beunruhigung. Diese werden nach der folgenden Konfiguration verschwinden.

## 6.2 Konfiguration des KDC

Die nötigen Schritte zur Konfiguration des KDC (auf `vmLS4`) sollen hier von Grund auf durchgeführt werden. Löschen Sie daher zunächst die Konfigurationsvorlagen für `/etc/krb5kdc/kdc.conf` und `/etc/krb5.conf`, oder benennen Sie diese beiden Dateien um, so wie das unten dargestellt ist. Stoppen Sie auch die bereits gestarteten Daemons.

Folgende Befehle ausführen:

```
systemctl stop krb5-kdc
systemctl stop krb5-admin-server
mv /etc/krb5kdc/kdc.conf /etc/krb5kdc/kdc.conf.BACKUP
mv /etc/krb5.conf /etc/krb5.conf.BACKUP
```

Legen Sie nun die Konfiguration des kdc an, indem Sie die Datei `/etc/krb5kdc/kdc.conf` erstellen, bzw. anpassen:

```
[libdefaults]
    default_realm = M159.IET-GIBB.CH

[kdcdefaults]
    kdc_ports = 88
    kdc_tcp_ports = 88
    v4_mode = disable

[realms]
    M159.IET-GIBB.CH = {
        database_name = /var/lib/krb5kdc/principal
        acl_file = /etc/krb5kdc/kadm5.acl
        #key_stash_file = /etc/krb5kdc/stash
        max_life = 10h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
        master_key_type = aes256-cts
        supported_encetypes = aes256-cts:normal arcfour-hmac:normal des3-hmac-sha1:normal
        default_principal_flags = +preauth
        #database_module = openldap_ldapconf
    }

[logging]
    kdc = SYSLOG:INFO:AUTH
    admin_server = SYSLOG:INFO:AUTH
```

### Neu mit Ubuntu 22.04

```
[kdcdefaults]
    kdc_ports = 750,88

[realms]
    M159.IET-GIBB.CH = {
        database_name = /var/lib/krb5kdc/principal
        admin_keytab = FILE:/etc/krb5kdc/kadm5.keytab
        acl_file = /etc/krb5kdc/kadm5.acl
        key_stash_file = /etc/krb5kdc/stash
        kdc_ports = 750,88
        max_life = 10h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
```

```

master_key_type = des3-hmac-sha1
#supported_encetypes = aes256-cts:normal aes128-cts:normal
default_principal_flags = +preauth
}

```

### 6.2.1 Initialisierung der KDC-Datenbank

Nun haben Sie also eine fertige KDC- Konfigurationsdatei. Bevor Sie das KDC starten und mit ihm arbeiten können, benötigen Sie noch etwas Wesentliches: **Principals**! Sie müssen noch eine KDC-Datenbank anlegen, die die Principals, deren Flags, Gültigkeitszeiten und vor allem deren Langzeitschlüssel enthält.

MIT Kerberos kennt verschiedene Kommandos zur Administration der KDC-Datenbank:

`kadmin`, `kadmin.local` und `kdb5-util`. Die ersten beiden führen Operationen an einzelnen Principals durch. Dabei muss `kadmin.local` lokal auf der KDC-Maschine ausgeführt werden, da es direkt auf die Datenbank zugreift, wohingegen `kadmin` ein Netzwerkprotokoll verwendet und auch zur Remote-Administration der KDC-Datenbank benutzt werden kann. Im Gegensatz dazu führt `kdb5-util` Operationen an der gesamten KDC-Datenbank durch. Dazu gehört auch die Initialisierung der Datenbank.

Die Initialisierung der Datenbank beginnt hier mit dem *Master Key* der KDC-Datenbank. Neben den bereits bekannten Langzeit- und Kurzzeitschlüsseln kennt MIT Kerberos noch diesen weiteren, sehr zentralen Schlüssel. Welche Bewandnis es damit hat, soll im Folgenden erläutert werden.

Wie Sie wissen, sind in einem Kerberos Realm die kryptografischen Langzeitschlüssel aller Principals nur den jeweiligen Principals selbst und dem KDC des Realm bekannt. Nur das KDC kennt dabei die Schlüssel aller Principals des Realm und diese Kenntnis ist gleichbedeutend mit der Kenntnis der zugehörigen Passwörter. Da die KDC-Datenbank alle diese Schlüssel enthält, muss gerade sie vor Angreifern besonders geschützt werden. In Produktivumgebungen sollte man daher Massnahmen ergreifen, um die KDC-Maschinen abzusichern.

Die KDC-Datenbank mit den Langzeitschlüsseln ist durch die Dateisystemberechtigungen auf den KDC-Maschinen vor unbefugten Zugriffen geschützt: Nur das KDC selbst (die Prozesse `krb5kdc` und `kadmind`) und die Administratoren mit `root`-Berechtigungen haben darauf Zugriff. Dieser Schutz ist aber noch nicht perfekt:

Soll die KDC-Datenbank regelmässig gesichert werden, so bekommen möglicherweise auch die Administratoren der Datensicherung Zugriff auf die zu schützenden Schlüssel.

Gelingt es einem Angreifer, eine KDC-Maschine oder deren Festplatte zu stehlen, dann würden ihn die Dateisystemberechtigungen nicht mehr davon abhalten, an die Schlüssel zu gelangen.

Die Langzeitschlüssel müssen also zusätzlich geschützt werden. MIT Kerberos speichert sie daher nicht im Klartext, sondern legt sie innerhalb der KDC-Datenbank verschlüsselt ab. Dabei werden nur die Schlüssel selbst verschlüsselt, die restlichen Inhalte der Datenbank bleiben

unverschlüsselt.

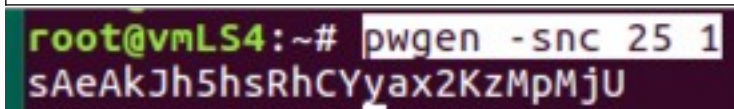
Der dazu verwendete Schlüssel ist der *Master Key*. Ohne ihn ist die KDC-Datenbank praktisch nutzlos. Wird beispielsweise die Festplatte mit der KDC-Datenbank gestohlen, so kann der Dieb zwar die Datenbank öffnen, mit deren Schlüsseln kann er jedoch nichts anfangen. Das gleiche gilt für die Administratoren, die für die Datensicherung zuständig sind.

Zum Master Key gehört ein *Master-Passwort*, von dem sich der Master Key durch eine `string2key`-Funktion ableitet. Programme, die auf die KDC-Datenbank zugreifen müssen, können dieses Passwort interaktiv abfragen. Auch die KDC-Prozesse erwartet beim Starten die Eingabe des Master-Passwortes. Damit ist ein nicht interaktiver Startup des KDC allerdings nicht mehr möglich. In folgenden werden Sie sehen, wie dies mit einer sogenannten **Stash-Datei** zu umgehen ist.

Um ein möglichst sicheres Master-Passwort zu erhalten, empfiehlt es sich, einen Passwortgenerator zu benutzen. Unter Ubuntu-Linux können Sie dafür das Tool `pwgen` verwenden. Um ein 25 Zeichen langes, zufälliges Passwort zu erzeugen, das aus Grossbuchstaben, Kleinbuchstaben und Ziffern besteht, verwenden Sie folgenden Aufruf.

Sehr sicheres Passwort erzeugen mit:

```
pwgen -snc 25 1
```



```
root@vmLS4:~# pwgen -snc 25 1
sAeAkJh5hsRhCYyax2KzMpMjU
```

Falls `pwgen` nicht verfügbar ist, können Sie dieses installieren mit `apt install pwgen`

Um uns hier in diesem praktischen Teil nicht mit komplizierten Passwörtern zu belasten, verwenden wir „unser“ gängiges Passwort `sm112345` auch als Master-Passwort.

### Datenbank initialisieren:

#### Hier initialisieren wir die Datenbank von Kerberos!

Kerberos hat eine eigene DB. In einer produktiven Umgebung wird meist OpenLDAP als DB-Backend von Kerberos verwendet. Für unsere Labor-Umgebung begnügen wir uns mit der DB, welche Bestandteil ist von Kerberos.

```
root@vmLS4:~# kdb5_util -r M159.IET-GIBB.CH create
Loading random data
Initializing database '/var/lib/krb5kdc/principal' for realm 'M159.IET-GIBB.CH',
master key name 'K/M@M159.IET-GIBB.CH'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key: sm112345
Re-enter KDC database master key to verify: sm112345
root@vmLS4:~#
```

Sie haben gerade eine initiale Datenbank für den Realm **M159.IET-GIBB.CH** erzeugt. Darin sind auch schon die wichtigsten Principals enthalten.

Mit dem Kommando `kadmin.local`, das Sie direkt auf dem KDC als Nutzer `root` starten müssen, können Sie alle Principals auflisten. Dazu dient das Subkommando `list-principals`

M159\_SKRIPT\_THEMA1\_KERBEROS\_V2.6.ODT

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 41/64

(oder die Kurzform `listprincs`). Die Befehle unten stellen den Vorgang dar. Zunächst werden Sie nach dem Master-Passwort (`sm112345`) gefragt, dann können Sie alle Principals auflisten. Schliesslich beenden Sie `kadmin.local` mit `quit`.

```
kadmin.local -m -r M159.IET-GIBB.CH
Authenticating as principal root/admin@M159.IET-GIBB.CH with password.
Enter KDC database master key: sm112345
kadmin.local: listprincs
K/M@M159.IET-GIBB.CH
kadmin/admin@M159.IET-GIBB.CH
kadmin/changepw@M159.IET-GIBB.CH
kadmin/vmls4@M159.IET-GIBB.CH
kiprop/vmls4@M159.IET-GIBB.CH
krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
kadmin.local: quit
root@vmLS4:~#
```

Die folgenden Principals sind beim Initialisieren angelegt worden:

**krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH**

(Allgemein: `krbtgt/REALM@REALM`) Hierbei handelt es sich um den wichtigsten und zentralsten Principal eines Kerberos Realm: den Principal des **Ticket-Granting Service** (TGS).

**kadmin/vmls4@M159.IET-GIBB.CH**

(Allgemein: `kadmin/FQDN@REALM`) Dieser Principal wird vom *kadmin-Dienst* für die Authentifizierung verwendet. Dabei ist *FQDN* der vollständige DNS-Hostname derjenigen KDC-Maschine, auf der der Kadmin-Dienst läuft.

**kadmin/admin@M159.IET-GIBB.CH**

(Allgemein: `kadmin/admin@REALM`) Dieser Principal wird ebenfalls vom *kadmin-Dienst* verwendet. Es handelt sich dabei um eine veraltete Form, die aber aus Gründen der Kompatibilität weiter unterstützt wird.

### 6.2.2 Mit `kadmin.local` Principals anlegen

Die initial angelegten Principals reichen aus, um KDC und Admin-Server starten zu können. Viel mehr können Sie aber mit ihnen noch nicht anfangen. Es fehlen immer noch Benutzer-Principals, die als Kerberos-Clients auftreten können. Sie sollten daher jetzt die folgenden beiden Benutzer-Principals anlegen:

**user@M159.IET-GIBB.CH** Dieser Principal dient im Folgenden für normale Arbeiten. Unter seiner Identität können Sie beispielsweise Tickets anfordern.

**user/admin@M159.IET-GIBB.CH** Diesen Principal können Sie für administrative Tätigkeiten benutzen. Er wird die KDC-Datenbank verwalten können.

Das Anlegen eines Principals wird mit dem Subkommando `add_principal` (Kurzform: `addprinc`) gemacht. Für das Anlegen benötigen wir das Masterpasswort, um auf die Datenbank zugreifen zu können. Der Realm-Anteil der Principal-Namen `M159.IET-GIBB.CH` müssen Sie nicht angeben, er wird automatisch angehängt.  
Anmerkung:

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 42/64

Der Befehl `kadmin.local` müssen wir auf der `vmLS4` lokal verwenden, solange der **krb5-admin-server** noch nicht installiert ist. Sobald dies gemacht ist (Kap. 7) kann mit dem Befehl `kadmin` der KDC von einem beliebigen Rechner aus in der Realm administriert werden.

Vorgehen zum Anlegen der Principals:

```
root@vmLS4:~# kadmin.local -m -r M159.IET-GIBB.CH
Authenticating as principal root/admin@M159.IET-GIBB.CH with password.
Enter KDC database master key: sm112345
kadmin.local: addprinc laura
WARNING: no policy specified for user@M159.IET-GIBB.CH; defaulting to no
policy
Enter password for principal "laura@M159.IET-GIBB.CH": sm112345
Re-enter password for principal "laura@M159.IET-GIBB.CH": sm112345
Principal "laura@M159.IET-GIBB.CH" created.
kadmin.local: addprinc laura/admin
WARNING: no policy specified for laura/admin@M159.IET-GIBB.CH; defaulting to
no policy
Enter password for principal "laura/admin@M159.IET-GIBB.CH": sm112345
Re-enter password for principal "laura/admin@M159.IET-GIBB.CH": sm112345
Principal "laura/admin@M159.IET-GIBB.CH" created.
kadmin.local: listprincs (Kontrolle der angelegten Principals)
K/M@M159.IET-GIBB.CH
kadmin/admin@M159.IET-GIBB.CH
kadmin/changepw@M159.IET-GIBB.CH
krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
laura/admin@M159.IET-GIBB.CH
laura@M159.IET-GIBB.CH
kadmin.local: quit
root@vmLS4:~#
```

### 6.2.3 Das Problem mit dem automatischen Starten des Kerberos-Dienstes

Im bisherigen Setup müssen Sie jedem Kommando, das auf die KDC- Datenbank zugreift, das Master-Passwort interaktiv übergeben. Das war beispielsweise beim Aufruf des Kommandos `kadmin.local` notwendig.

Sie mussten `kadmin.local` dazu die Option `-m` mitgeben. Diese Option existiert bei allen MIT-Kerberos-Komponenten, die direkt auf die Datenbank zugreifen müssen: `kadmin.local`, `kdb5-util`, aber auch bei den Daemons `krb5kdc` und `kadmind`. Die Option `-m` führt dazu, dass explizit nach dem Master-Passwort gefragt wird, und bewirkt dadurch eine zusätzliche Interaktion mit dem Administrator. Das führt zu einer wesentlich höheren Sicherheit der KDC-Datenbank, ist in der Praxis aber etwas unbequem, **da die Dienste `krb5kdc` und `kadmind` nicht ohne interaktive Eingabe des Master-Passwortes starten können**. Es ist daher durchaus üblich, den Master Key in einer sogenannten **Stash-Datei** zu speichern. Damit entfällt die Notwendigkeit der interaktiven Passwordeingabe. **Erst damit wird auch ein nicht interaktiver Start des KDC möglich.**

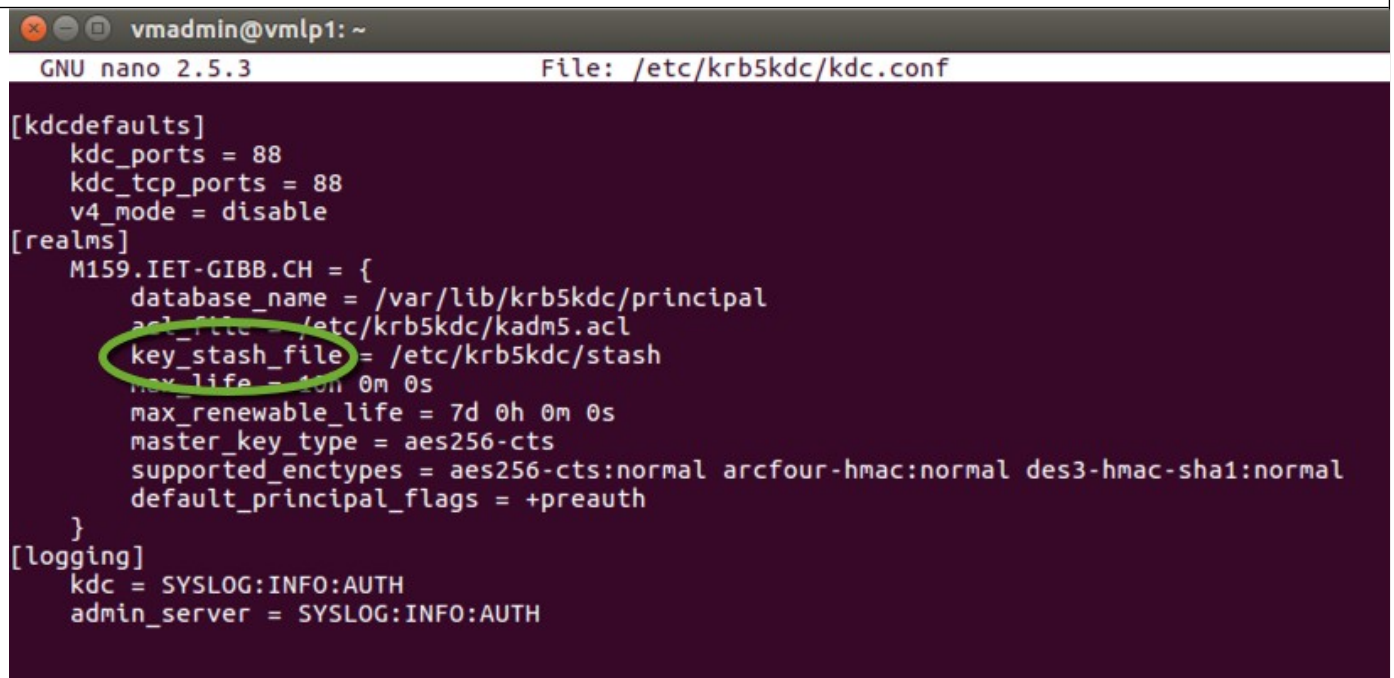
Für die Beispielumgebung soll hier nun eine Stash-Datei angelegt werden. Damit werden Sie mit dem Master Key und dem Master-Passwort nicht mehr in Berührung kommen.

Entfernen Sie zunächst das Kommentarzeichen »#« vor dem Parameter `key-stash-file` in der Datei `/etc/krb5kdc/kdc.conf`. Benutzen Sie dann den Aufruf `kdb5-util -r M159.IET-GIBB.CH stash`, um das Master-Passwort für den Realm `M159.IET-GIBB.CH` in der Stash-Datei abzulegen. Der Master Key befindet sich nun in der Datei `/etc/krb5kdc/stash`. Ab jetzt kann die Option `-m` bei `kadmin.local` und anderen Programmen entfallen. Testen Sie den Erfolg dieses Kommandos, indem Sie den Aufruf wiederholen, diesmal aber das `-m` weglassen.

Hier das Vorgehen der Reihe nach:

Entfernen des # vor `key_stash_file` in Datei `/etc/krb5kdc/kdc.conf`

`nano /etc/krb5kdc/kdc.conf`



```

vmadmin@vmlp1: ~
GNU nano 2.5.3 File: /etc/krb5kdc/kdc.conf

[kdcdefaults]
  kdc_ports = 88
  kdc_tcp_ports = 88
  v4_mode = disable
[realms]
  M159.IET-GIBB.CH = {
    database_name = /var/lib/krb5kdc/principal
    acl_file = /etc/krb5kdc/kadm5.acl
    key_stash_file = /etc/krb5kdc/stash
    max_life = 40h 0m 0s
    max_renewable_life = 7d 0h 0m 0s
    master_key_type = aes256-cts
    supported_encetypes = aes256-cts:normal arcfour-hmac:normal des3-hmac-sha1:normal
    default_principal_flags = +preauth
  }
[logging]
  kdc = SYSLOG:INFO:AUTH
  admin_server = SYSLOG:INFO:AUTH

```

Das Masterpasswort in stash-Datei `/etc/krb5kdc/stash` schreiben mit

```

root@vmLS4:~# kdb5_util -r M159.IET-GIBB.CH stash
kdb5_util: Can not fetch master key (error: No such file or directory). while reading master key
kdb5_util: Warning: proceeding without master key
Enter KDC database master key: sm112345
root@vmLS4:~#
Kontrolle, ob die stash-Datei geschrieben wurde mit:
root@vmLS4:~# ls -l /etc/krb5kdc/stash
-rw----- 1 root root 77 Aug 12 22:07 /etc/krb5kdc/stash

```

Kontrolle, ob nun die Option `-m` weggelassen werden kann. Dazu listen wir uns alle Principals mit dem Kommando `kadmin.local`, Subkommando `listprincs`. Wenn Sie alles richtig gemacht haben, werden Sie jetzt nicht mehr nach dem Master-Passwort gefragt.



```
root@vmLS4:~# kadmin.local -r M159.IET-GIBB.CH
Authenticating as principal root/admin@M159.IET-GIBB.CH with password.
kadmin.local: listprincs
Output:
```

```
kadmin.local: listprincs
K/M@M159.IET-GIBB.CH
kadmin/admin@M159.IET-GIBB.CH
kadmin/changepw@M159.IET-GIBB.CH
krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
laura/admin@M159.IET-GIBB.CH
laura@M159.IET-GIBB.CH
kadmin.local: 
```

```
kadmin.local: quit
root@vmLS4:~#
```

#### 6.2.4 Das KDC starten und testen

Jetzt kann das KDC gestartet werden. Das geschieht unter Linux durch ein Startskript, das im Verzeichnis `/etc/init.d` liegt. Bei Ubuntu-Linux lautet der Name dieses Skripts `/etc/init.d/krb5-kdc`.

Vorher müssen Sie noch in der Datei `/etc/default/krb5-kdc` die Variable `DAEMON_ARGS` setzen: Fügen Sie die Zeile `"DAEMON_ARGS="-r M159.IET-GIBB.CH"` hinzu.

Damit wird dem KDC über die Option `-r` mitgeteilt, für welchen Realm es zuständig ist. Die komplette Datei `/etc/default/krb5-kdc` ist unten dargestellt. Danach können Sie mit dem Kommando `/etc/init.d/krb5-kdc start` das KDC starten.

Edit `/etc/default/krb5-kdc` mit `nano /etc/default/krb5-kdc`

```
# Automatically generated.  Only the value of DAEMON_ARGS will be preserved.
# If you change anything in this file other than DAEMON_ARGS, first run
# dpkg-reconfigure krb5-kdc and disable managing the KDC configuration with
# debconf.  Otherwise, changes will be overwritten.
DAEMON_ARGS="-r M159.IET-GIBB.CH"
```

Danach den KDC starten mit

```
root@vmLS4:~# systemctl start krb5-kdc
root@vmLS4:~# systemctl status krb5-kdc
Output muss jetzt active (running) sein:
```



```

root@vmLS4:/etc/krb5kdc# systemctl status krb5-kdc
● krb5-kdc.service - Kerberos 5 Key Distribution Center
   Loaded: loaded (/lib/systemd/system/krb5-kdc.service; enabled; vendor preset: enabled)
   Drop-In: /usr/lib/systemd/system/krb5-kdc.service.d
            └─slapd-before-kdc.conf
   Active: active (running) since Tue 2022-08-16 23:57:31 CEST; 4s ago
     Process: 3351 ExecStart=/usr/sbin/krb5kdc -P /var/run/krb5-kdc.pid $DAEMON_ARGS (code=exited, status=0/SUCCESS)
    Main PID: 3352 (krb5kdc)
       Tasks: 1 (limit: 1030)
      Memory: 1.1M
         CPU: 22ms
    CGroup: /system.slice/krb5-kdc.service
            └─3352 /usr/sbin/krb5kdc -P /var/run/krb5-kdc.pid -r M159.IET-GIBB.CH

Aug 16 23:57:31 vmLS4.m159.iet-gibb.ch krb5kdc[3351]: Setting up UDP socket for address ::88
Aug 16 23:57:31 vmLS4.m159.iet-gibb.ch krb5kdc[3351]: setsockopt(12,IPV6_V6ONLY,1) worked
Aug 16 23:57:31 vmLS4.m159.iet-gibb.ch krb5kdc[3351]: Setting pktinfo on socket ::88
Aug 16 23:57:31 vmLS4.m159.iet-gibb.ch krb5kdc[3351]: Setting up TCP socket for address 0.0.0.0.88
Aug 16 23:57:31 vmLS4.m159.iet-gibb.ch krb5kdc[3351]: setsockopt(14,IPV6_V6ONLY,1) worked
Aug 16 23:57:31 vmLS4.m159.iet-gibb.ch krb5kdc[3351]: set up 6 sockets
Aug 16 23:57:31 vmLS4.m159.iet-gibb.ch systemd[1]: krb5-kdc.service: Can't open PID file /run/krb5-kdc.pid (yet?) after start: Operation not permitted
Aug 16 23:57:31 vmLS4.m159.iet-gibb.ch systemd[1]: krb5kdc[3352]: commencing operation
Aug 16 23:57:31 vmLS4.m159.iet-gibb.ch systemd[1]: Started Kerberos 5 Key Distribution Center.
root@vmLS4:/etc/krb5kdc#

```

### Herzlichen Glückwunsch!

Sie haben Ihr erstes KDC aufgesetzt und gestartet. Ein kleiner Funktionstest wäre jetzt sicherlich angebracht. Dazu haben Sie ja bereits den Client Principal `laura@M159.IET-GIBB.CH` angelegt und kennen dessen Passwort `sm112345`. Sie können die Funktionalität des KDC also testen, indem Sie mit dem Kommando `kinit username@M159.IET-GIBB.CH` ein TGT (Ticket-Granting Ticket) für `user@M159.IET-GIBB.CH` anfordern. **Dieser Test sollte auf der Clientmaschine `vm1p1` erfolgen.** Gehen Sie wie folgt vor:

Nach dem Starten von `vm1p1`, Kontrolle ob `vmLS4` erreichbar ist mit:

```

vmadmin@vm1p1:~$ ping vmLS4
PING vmLS4.m159.iet-gibb.ch (192.168.210.60) 56(84) bytes of data:
64 bytes from vmLS4.m159.iet-gibb.ch (192.168.210.60): icmp_seq=1 ttl=64 time=0.180 ms
64 bytes from vmLS4.m159.iet-gibb.ch (192.168.210.60): icmp_seq=2 ttl=64 time=0.410 ms
64 bytes from vmLS4.m159.iet-gibb.ch (192.168.210.60): icmp_seq=3 ttl=64 time=0.725 ms

```

Danach muss der Kerberos-**Client** mit root installiert werden. Verwenden Sie also `sudo...`

```

sudo apt install krb5-user

```

vmadmin@vm1p1: ~

Paketkonfiguration

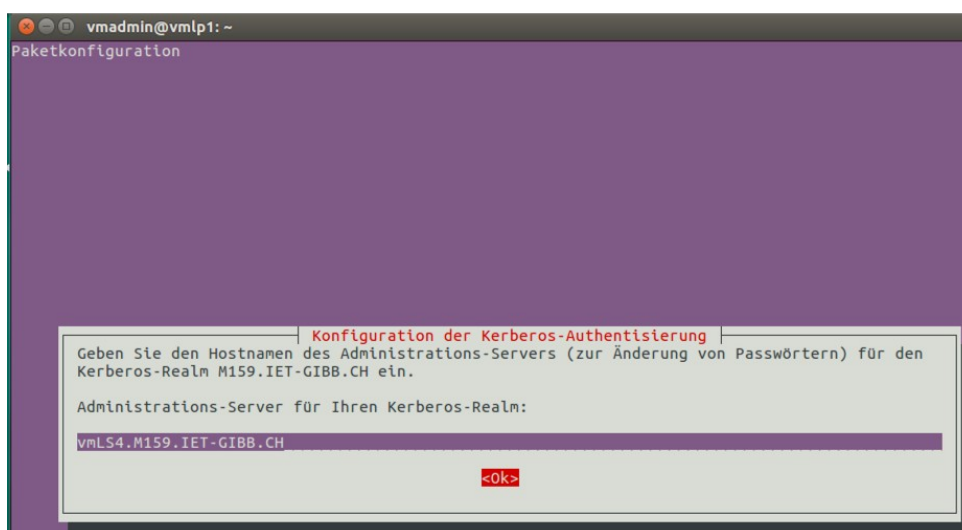
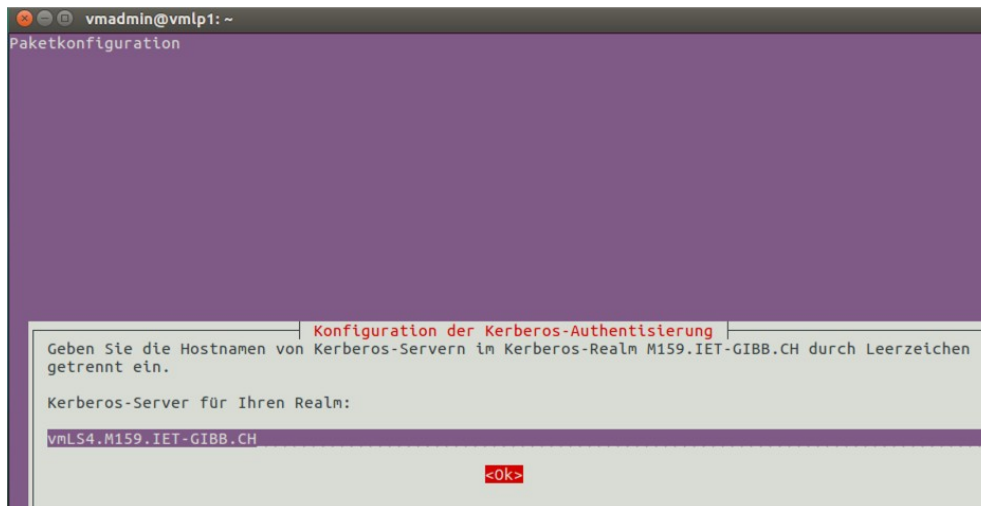
Konfiguration der Kerberos-Authentisierung

Wenn Benutzer versuchen, Kerberos zu nutzen und einen Principal oder Benutzernamen angeben, ohne dabei festzulegen, zu welchem Kerberos-Administrationsbereich (Realm) dieser Principal gehört, dann fügt das System den voreingestellten Realm an. Der voreingestellte Realm kann auch als Realm eines Kerberos-Dienstes verwendet werden, der auf dem lokalen Rechner läuft. Der voreingestellte Realm ist die großgeschriebene Version der lokalen DNS-Domain.

Voreingestellter Realm für Kerberos Version 5:

**M159.IET-GIBB.CH**

<Ok>



Die Konfigurationsangaben wurden in die Datei `/etc/krb5.conf` geschrieben.  
Ergänzen Sie noch folgende Einträge im `/etc/krb5.conf`. Alles andere belassen Sie wie es ist.  
**Machen Sie das mit root oder starten Sie den Editor mit sudo !**

## Inhalt krb5.conf

```
[libdefaults]
    default_realm = M159.IET-GIBB.CH

# The following krb5.conf variables are only for MIT Kerberos.
    kdc_timesync = 1
    ccache_type = 4
    forwardable = true
    proxiable = true

# The following encryption type specification will be used by MIT Kerberos
# if uncommented.  In general, the defaults in the MIT Kerberos code are
# correct and overriding these specifications only serves to disable new
# encryption types as they are added, creating interoperability problems.
#
# The only time when you might need to uncomment these lines and change
# the enctypees is if you have local software that will break on ticket
# caches containing ticket encryption types it doesn't know about (such as
# old versions of Sun Java).

#    default_tgs_enctypes = des3-hmac-sha1
#    default_tkt_enctypes = des3-hmac-sha1
#    permitted_enctypes = des3-hmac-sha1

# The following libdefaults parameters are only for Heimdal Kerberos.
    fcc-mit-ticketflags = true

[realms]
    M159.IET-GIBB.CH = {
        kdc = vmLS4.m159.iet-gibb.ch
        admin_server = vmLS4.m159.iet-gibb.ch
    }

[domain_realm]
    .m159.iet-gibb.ch = M159.IET-GIBB.CH
    m159.iet-gibb.ch  = M159.IET-GIBB.CH

[logging]
    default = SYSLOG:INFO:AUTH
```

Legen Sie diese Datei /etc/krb5.conf nun auf den Maschinen `vm1p1` und `vmLS4` an. Damit funktionieren die Clientkommandos auf beiden Rechnern.

Damit ist der Kerberos-Client nun konfiguriert und wir können nun ein Ticket vom KDC beziehen. Dazu verwenden wir `kinit` und `klist`.

Ticket für den User `laura@M159.IET-GIBB.CH` beziehen mit:

```
vmadmin@vm1p1:~$ kinit laura@M159.IET-GIBB.CH
Password for laura@M159.IET-GIBB.CH: sm112345
```

Tickets anzeigen mit:

```
vmadmin@vmlp1:~$ klist
```

Output:

```
vmadmin@vmlp1:/etc$ klist
```

**Ticketzwischenspeicher:** FILE:/tmp/krb5cc\_1000

**Standard-Principal:** laura@M159.IET-GIBB.CH

Valid starting	Expires	Service principal
17.08.2022 00:08:22	17.08.2022 10:08:22	krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
erneuern bis 18.08.2022 00:08:18		

Gelöste Tickets werden nun auch geloggt. Wie und wo sehen Sie die Logging-Informationen?  
 tail /var/log/auth.log

Mit `klist` sehen wir hier ein Beispiel eines **Credential Caches**, welcher auch für SSO verwendet wird. Diesen Credential Cache haben wir „von Hand“ mit `kinit` kreiert. Bei der Verwendung von Netzwerkdiensten (Login-Prozess, ssh, http, etc) wird dies automatisch gemacht, indem der Dienst „kerberisiert“ wird. Dazu mehr später. Vorerst schauen wir uns diesen Cache genauer an. Was sehen wir?

Wie Sie oben sehen können, liegt der Kerberos-v5-Cache in der Datei `/tmp/krb5cc_1000`. 1000 ist dabei die numerische Benutzer-ID. Eine weitere Information in der `klist`-Ausgabe ist die Angabe des `Default principal`. Der gibt die Identität an, also den Principal-Namen `laura@M159.IET-GIBB.CH`, welchen wir von Hand mit `kinit` erstellt haben. Weiter unten werden dann die Kerberos Tickets gelistet. Zum jetzigen Zeitpunkt liegt nur eines für den Service `krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH` vor. Dies ist das sogenannte **Ticket-Granting Ticket (TGT)**.

In den Ausgaben von `klist` sind noch weitere Angaben zu diesem Ticket zu sehen: In der Spalte `Valid starting` steht der Zeitpunkt, ab wann das Ticket gültig ist, in `Expires`, wann es abläuft. Der Angabe nach `erneuern bis` kann man entnehmen, wie lange es erneuerbar ist.

Mit der Option `-f` zeigt `klist` zusätzlich auch die gesetzten *Ticket Flags* an. Im vorliegenden Beispiel sind das die Flags `FPRIA`, sie bedeuten im Einzelnen: **F**orwardable, **P**roxiabile, **R**enewable, **I**nitial und **P**re-Authenticat**e**d. Geben Sie das Kommando `klist -f` ein. Was es mit diesen Flags genau auf sich hat werden wir hier nicht anschauen.

## 7 Administration von MIT Kerberos

### 7.1 Kadmin-Dienst

Im Kapitel 5.2.2 waren Sie bereits administrativ tätig, als Sie die beiden Anwender-Principals `user` und `user/admin` mit `kadmin.local` angelegt haben. Dort und auch bei der Initialisierung mit `kdb5-util` haben Sie für Operationen an der KDC-Datenbank jeweils Tools verwendet, die einen direkten Datenbankzugriff durchführen. Sie mussten dazu als `root` auf der KDC-Maschine angemeldet sein. Das ist in der Praxis natürlich etwas umständlich, eine Möglichkeit zur Remote-Administration wäre hierfür wünschenswert.

MIT Kerberos sieht für die Kerberos-Administration eine Client- Server-Lösung vor. Der Client besteht aus dem `kadmin`-Kommando, das auf den *Kadmin-Dienst* (`kadmind`) zugreift. Dieser Ansatz besitzt einige Vorteile gegenüber lokalen Tools wie `kadmin.local`:

- Es können mehrere Kerberos-Administratoren bestimmt werden. Diese müssen keine `root`-Rechte auf den KDC-Maschinen erhalten.
- Einzelnen Administratoren kann man ganz bestimmte Aufgaben delegieren. Beispielsweise kann man manchen Administratoren nur die Berechtigung zur Änderung von Passwörtern geben, während andere Admins auch Principals anlegen dürfen.
- Die Kerberos-Administration ist von allen Workstation des Realm möglich, Sie müssen also nicht per SSH auf der KDC-Maschine angemeldet sein.

### 7.2 Administrative Zugriffe kontrollieren

Kerberos regelt nur die Authentisierung, also das »*Wer bin ich?*«. Der `kadmin`-Dienst kann damit die Identität des `kadmin`-Clients (also des zugreifenden Administrators) anhand von dessen Kerberos Principal bestimmen. Um die Zugriffskontrolle, also das »*Was darf ich?*«, muss sich der `kadmind` selbst kümmern. Er benötigt dazu Autorisierungsinformationen, mit denen er entscheiden kann, welche Zugriffsberechtigungen ein zugreifender Client bekommen soll. Diese Informationen entnimmt der `kadmind` einer Datei, die eine Liste von Zugriffsregeln (Access Control List, ACL) enthält. Sie können die Lage der ACL-Datei über den Parameter `acl-file` in der `kdc.conf` vorgeben. Der Default unter Ubuntu ist `/etc/krb5kdc/kadm5.acl`. Die ACL-Datei besteht aus Zeilen, die einzelne Zugriffsregeln beschreiben, oder Kommentare enthalten, die durch ein `#`-Zeichen eingeleitet werden. Eine solche Zugriffszeile besteht aus zwei oder drei Feldern für *Principal*, *Zugriffsmaske* und optional dem *Zugriffsziel*.

Aufbau der `kadm5.acl`:

```
# Kommentarzeile
Principal Zugriffsmaske [Zugriffsziel]
Principal Zugriffsmaske [Zugriffsziel]
[...]
```

In den Beispiel-Realms sollen alle Principals, deren zweite Komponente `admin` lautet, die vollen administrativen Rechte erhalten. Sie können dann beispielsweise den Principal `user/admin`, den Sie vorher angelegt haben, für die Administration benutzen. Legen Sie dazu die Datei `/etc/krb5kdc/kadm5.acl` an.

```
nano /etc/krb5kdc/kadm5.acl      (auf vmLS4)
# Vollzugriff fuer jeden */admin Principal aus der M159.IET-GIBB.CH:
*/admin@M159.IET-GIBB.CH *
```

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 50/64

## 7.3 Starten der administrativen Dienste

Bevor Sie den Kadmin-Dienst starten können, sollten Sie noch dafür sorgen, dass ihm sein Kerberos Realm per Kommandozeilenoption `-r` übergeben wird. Unter Ubuntu geht das am elegantesten durch die Variable `DAEMON_ARGS` in der Datei `/etc/default/krb5-admin-server`. Der komplette Inhalt dieser Datei ist hier dargestellt: (auf **vmLS4** mit root einloggen per ssh von **vm1p1**)

### **nano /etc/default/krb5-admin-server**

```
# Automatically generated.  If you change anything in this file other than
the
# values of DAEMON_ARGS, first run dpkg-reconfigure
# krb5-admin-server and disable managing the kadmin configuration with
# debconf.  Otherwise, changes will be overwritten.

RUN_KADMIND=true
DAEMON_ARGS="-r M159.IET-GIBB.CH"
```

Nun bleibt nur noch ein letzter Schritt zu tun: Starten Sie den `kadmind` mit dem Kommando:

```
sudo systemctl start krb5-admin-server
```

Kontrolle mit:

```
sudo systemctl status krb5-admin-server
```

Output:

```
vmadmin@vmLS4:~$ sudo systemctl status krb5-admin-server
● krb5-admin-server.service - Kerberos 5 Admin Server
   Loaded: loaded (/lib/systemd/system/krb5-admin-server.service; enabled; vendor preset: enabled)
   Drop-In: /usr/lib/systemd/system/krb5-admin-server.service.d
            └─slapd-before-kdc.conf
   Active: active (running) since Wed 2022-08-17 00:38:08 CEST; 37s ago
     Main PID: 3809 (kadmind)
        Tasks: 1 (limit: 1030)
       Memory: 612.0K
          CPU: 21ms
    CGroup: /system.slice/krb5-admin-server.service
            └─3809 /usr/sbin/kadmind -nofork -r M159.IET-GIBB.CH

Aug 17 00:38:08 vmLS4.m159.iet-gibb.ch kadmind[3809]: setsockopt(12, IPV6_V6ONLY, 1) worked
Aug 17 00:38:08 vmLS4.m159.iet-gibb.ch kadmind[3809]: Setting up RPC socket for address 0.0.0.0.749
Aug 17 00:38:08 vmLS4.m159.iet-gibb.ch kadmind[3809]: Setting up RPC socket for address ::.749
Aug 17 00:38:08 vmLS4.m159.iet-gibb.ch kadmind[3809]: setsockopt(14, IPV6_V6ONLY, 1) worked
Aug 17 00:38:08 vmLS4.m159.iet-gibb.ch kadmind[3809]: set up 6 sockets
Aug 17 00:38:08 vmLS4.m159.iet-gibb.ch kadmind[3809]: No dictionary file specified, continuing without one.
Aug 17 00:38:08 vmLS4.m159.iet-gibb.ch kadmind[3809]: No dictionary file specified, continuing without one.
Aug 17 00:38:08 vmLS4.m159.iet-gibb.ch kadmind[3809]: Seeding random number generator
Aug 17 00:38:08 vmLS4.m159.iet-gibb.ch kadmind[3809]: starting
Aug 17 00:38:08 vmLS4.m159.iet-gibb.ch kadmind[3809]: kadmind: starting...
```

Einen ersten Test des kadmin-Dienstes können Sie mit dem `kadmin`-Tool durchführen. Verbinden Sie sich dazu als `laura/admin` mit dem kadmin-Dienst.

**Führen Sie das vom Rechner **vm1p1** aus!**

Die vorläufige `/etc/krb5.conf` auf **vm1p1** enthält bereits den notwendigen Eintrag `admin_server`, sie muss daher nicht erweitert werden:

```

    }
    fcc-mit-ticketflags = true

[realms]
    M159.IET-GIBB.CH = {
        kdc = vMLS4.M159.IET-GIBB.CH
        admin_server = vMLS4.M159.IET-GIBB.CH
    }
/admin

```

Sie müssen dem `kadmin`-Kommando aber mit der Option `-p` den Namen Ihres Admin-Principals mitgeben. Der lautet hier `laura/admin@M159.IET-GIBB.CH`. Sie werden dann nach dessen Passwort gefragt. Danach können Sie beispielsweise das Subkommando `listprincs` aufrufen und anschliessend Ihre Sitzung mit `quit` beenden. Der Vorgang sollte so aussehen:

```

vmadmin@vmlp1:~$ kadmin -p laura/admin@M159.IET-GIBB.CH
Authentifizierung als Principal laura/admin@M159.IET-GIBB.CH mit Passwort
Passwort für laura/admin@M159.IET-GIBB.CH: sm112345
kadmin: listprincs
K/M@M159.IET-GIBB.CH
kadmin/admin@M159.IET-GIBB.CH
kadmin/changepw@M159.IET-GIBB.CH
krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
laura/admin@M159.IET-GIBB.CH
laura@M159.IET-GIBB.CH

```

Von nun an müssen wir für administrative Aufgaben wie das Anlegen von Principals nicht mehr das Kommando `kadmin.local` verwenden. Für alle weiteren administrativen Tätigkeiten verwenden wir nun `kadmin`.

**Anmerkung:** falls `kadmin` nicht funktioniert (Fehlermeldung: `kadmin: GSS-API (or Kerberos) error while initializing kadmin interface`) kann ein Debugging mit

```
env KRB5_TRACE=/dev/stdout kadmin -p user/admin@M159.IET-GIBB.CH
```

eingeschaltet werden. Der Fehler hat mit dem Setup auf einer vm zu tun, welche Ressourcenrestriktionen für das Generieren von Zufallszahlen hat. Kerberos benötigt Zufallszahlen für das Erstellen von Keys. Quelle: <https://serverfault.com/questions/803662/kerberos-error-while-initializing-kadmin-interface-from-admin-server>

Diese Lösung scheint zu funktionieren - **nur anwenden, falls sie wirklich dieses Problem haben!**  
**Vorgehen:**

```

apt-get install rng-tools (auf vMLS4)
Editieren von /etc/default/rng-tools
Einfügen der 2 Zeilen:

```

```

vmadmin@vMLS4:~$ cat /etc/default/rng-tools
HRNGDEVICE=/dev/urandom
HRNGDEVICE=/dev/random
vmadmin@vMLS4:~$

```

```

rng-tools-daemon neu starten:
/etc/init.d/rng-tools stop
/etc/init.d/rng-tools start

```

Neuversuch auf `vmlp1` mit `kadmin -p user/admin@M159.IET-GIBB.CH`  
**Der Fehler sollte jetzt nicht mehr erscheinen.**



## 7.4 Principals verwalten

Bevor wir weitere Nutzer-Principals anlegen, wollen wir noch Richtlinien für deren Passwörter festlegen.

### 7.4.1 Passwortrichtlinien festlegen

Wir wollen 2 Passwortrichtlinien definieren. Diese sind realitätsfremd. Es geht darum zu verstehen, wie man das macht.

- Eine Passwortrichtlinie mit dem Namen `admin`. Diese soll allen Admin-Principals wie beispielsweise dem `user/admin` zugeordnet sein: Passwörter solcher Administratoren müssen alle 3650 Tage geändert werden, frühestens aber nach einem Tag. Die letzten 10 Passwörter dürfen von Admins nicht wiederverwendet werden. Ein Admin-Passwort soll mindestens 4 Zeichen lang sein und aus mindestens drei Zeichenklassen bestehen.
- Eine Passwortrichtlinie mit dem Namen `default` für normale Anwender-Principals der Beispielumgebung. Diese sollen spätestens nach 1800 Tagen ihr Passwort ändern müssen, das aus mindestens 6 Zeichen aus zwei verschiedenen Zeichenklassen bestehen soll. Hier sollen die letzten 8 Passwörter tabu sein.

Die folgenden Befehle zeigen, wie man das macht:

```
vmadmin@vmlp1:~$ kadmin -p laura/admin
Authenticating as principal user/admin with password.
Password for user/admin@M159.IET-GIBB.CH: sm112345
kadmin: add_policy -maxlife 3650days -minlife 1day -minlength 4 -minclasses 3 -history 10 admin
kadmin: add_policy -maxlife 1800days -minlife 1day -minlength 6 -minclasses 2 -history 8 default
kadmin: list_policies
admin
default
kadmin: get_policy admin
Richtlinie: »admin«
Maximale Kennwortdauer: 3650 days 00:00:00
Minimale Kennwortdauer: 1 day 00:00:00
minimale Passwortlänge: 4
minimale Anzahl von Passwortzeichenklassen: 3
Anzahl aufbewahrter alter Schlüssel: 10
maximale Anzahl falscher Passwordeingaben vor dem Sperren: 0
Rücksetzintervall für zu viele falsch eingebene Passwörter: 0 days 00:00:00
Passwortsperredauer: 0 days 00:00:00

kadmin: get_policy default
Richtlinie: »default«
Maximale Kennwortdauer: 1800 days 00:00:00
Minimale Kennwortdauer: 1 day 00:00:00
minimale Passwortlänge: 6
minimale Anzahl von Passwortzeichenklassen: 2
Anzahl aufbewahrter alter Schlüssel: 8
maximale Anzahl falscher Passwordeingaben vor dem Sperren: 0
kadmin: quit
```

Setzen sie die Richtlinien wieder zurück, damit die Eingabe von **sm112345** möglich ist:  
In diesem Beispiel muss für `admin` nur `-minclasses` auf 1 gesetzt werden. Den notwendigen Befehl zum Modifizieren einer Policy finden sie mit '?' im `kadmin: -` Prompt.



Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 53/64

**Tipp:** Löschen Sie die Policy «admin» und legen Sie diese neu an oder modifizieren Sie die bestehende:

```
modify_policy, modpol      Modify policy
delete_policy, delpol      Delete policy
```

Anschliessend Kontrolle mit:

```
get_policy admin
```

Hier ein Beispiel, da würde auch 1234 oder abcd als Passwort funktionieren:

```
kadmin: get_policy admin
```

```
Richtlinie: »admin«
```

```
Maximale Kennwortdauer: 3650 days 00:00:00
```

```
Minimale Kennwortdauer: 1 day 00:00:00
```

```
minimale Passwortlänge: 4
```

```
minimale Anzahl von Passwortzeichenklassen: 1
```

```
Anzahl aufbewahrter alter Schlüssel: 10
```

```
maximale Anzahl falscher Passwordeingaben vor dem Sperren: 0
```

```
Rücksetzintervall für zu viele falsch eingebene Passwörter: 0 days 00:00:00
```

```
Passwortsperredauer: 0 days 00:00:00
```

#### **Hinweis:**

*Wenn Sie eine Policy mit `add_policy` hinzufügen wollen, welche schon besteht, erscheint die Fehlermeldung "add\_policy: Unknown code adb 1". Diese Meldung wird im `/var/log/auth.log` aufgeschlüsselt in Klartext.*

### **7.4.2 Anwender-Principals anlegen**

Wir wollen Principals für zwei Beispielnutzer anlegen:

Max Mustermann (`maxm@M159.IET-GIBB.CH`)

Erika Musterfrau (`erim@M159.IET-GIBB.CH`)

Für Max Mustermann soll zusätzlich auch der administrative Principal `maxm/admin@M159.IET-GIBB.CH` angelegt werden.

Die drei Principals werden mit dem `kadmin`-Subkommando `add_principal` angelegt. Über die Option `-policy` bekommen sie ihre jeweilige Passwortrichtlinie zugeordnet. Die Option `-pw` setzt das initiale Passwort auf den Wert `sml12345`.

Das Vorgehen ist wie folgt:

```
vmadmin@vmlp1:~$ kadmin -p laura/admin
Authenticating as principal laura/admin with password.
Password for laura/admin@M159.IET-GIBB.CH:sml12345
kadmin: add_principal -policy default -pw sml12345 maxm
Principal "maxm@M159.IET-GIBB.CH" created.
kadmin: add_principal -policy default -pw sml12345 erim
Principal "erim@M159.IET-GIBB.CH" created.
kadmin: add_principal -policy admin -pw sml12345 maxm/admin
Principal "maxm/admin@M159.IET-GIBB.CH" created.
kadmin:quit
```

Test, ob mit den angelegten User-Principals ein TGT angefordert werden kann:

```

vmadmin@vmlp1:~$ kinit erim@M159.IET-GIBB.CH
Password for erim@M159.IET-GIBB.CH: sml12345
Ticket anzeigen:
vmadmin@vmlp1:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: erim@M159.IET-GIBB.CH

Valid starting      Expires            Service principal
13.08.2017 12:18:38  13.08.2017 22:18:38  krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
        renew until 14.08.2017 12:18:33

vmadmin@vmlp1:~$ kinit maxm@M159.IET-GIBB.CH
Password for maxm@M159.IET-GIBB.CH: sml12345
vmadmin@vmlp1:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: maxm@M159.IET-GIBB.CH

Valid starting      Expires            Service principal
13.08.2017 12:19:15  13.08.2017 22:19:15  krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
        renew until 14.08.2017 12:19:10

vmadmin@vmlp1:~$ kinit maxm/admin@M159.IET-GIBB.CH
Password for maxm/admin@M159.IET-GIBB.CH: sml12345
vmadmin@vmlp1:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: maxm/admin@M159.IET-GIBB.CH

Valid starting      Expires            Service principal
13.08.2017 12:19:40  13.08.2017 22:19:40  krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
        renew until 14.08.2017 12:19:35

```

**Sie sehen, dass mit jedem kinit das bestehende TGT mit dem neuen überschrieben bzw. ersetzt wird.  
Sie sehen auch, dass der Credential Cache im File /tmp/krb5cc\_1000 angelegt wird**

### 7.4.3 Dienste-Principals anlegen

Als erster Dienst in der KDC-Datenbank soll der Principal des Hostdienstes für die vmlp1 eingetragen werden. Dieser Principal dient allen Diensten, die die interaktive Verwendung eines Rechners zur Verfügung stellen. Sie werden den Hostdienst für die vmlp1 später benötigen, beispielsweise wenn Sie die lokale Anmeldung unter Linux an Kerberos anbinden oder wenn Sie die Remote-Zugriffe über Telnet oder OpenSSH kerberisieren. Der Principal-Name des Hostdienstes für die vmlp1 lautet host/vmlp1.m159.iet-gibb.ch@M159.IET-GIBB.CH. Das Anlegen dieses Principals ist hier dargestellt:

```

vmadmin@vmlp1:~$ kadmin -p laura/admin
Anzeigen der bekannten Principals mit:
kadmin: listprincs

```

Nun erzeugen wir den neuen Hostdienst:

```

kadmin: add_principal -clearpolicy -randkey +requires_preauth host/vmlp1.m159.iet-gibb.ch@M159.IET-GIBB.CH
M159_SKRIPT_THEMA1_KERBEROS_V2.6.ODT

```

```
Principal "host/vmlp1.m159.iet-gibb.ch@M159.IET-GIBB.CH" created.
kadmin: listprincs
K/M@M159.IET-GIBB.CH
erim@M159.IET-GIBB.CH
host/vmlp1.m159.iet-gibb.ch@M159.IET-GIBB.CH
kadmin/admin@M159.IET-GIBB.CH
kadmin/changepw@M159.IET-GIBB.CH
krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
laura/admin@M159.IET-GIBB.CH
laura@M159.IET-GIBB.CH
maxm/admin@M159.IET-GIBB.CH
maxm@M159.IET-GIBB.CH
kadmin: quit
```

Beachten Sie die Synthax des soeben angelegten Host-Principals. Dieser besteht aus dem DNS FQDN und der grossgeschriebenen REALM, getrennt mit einem @.

Zur Erklärung des Kommandos `add_principal`:

Die Option `-clearpolicy` sorgt dafür, dass dem neuen Dienste-Principal keine Passworrichtlinie zugeordnet wird. Durch `-randkey` werden zufällige Schlüssel erzeugt. Nun gibt es also den Principal `host/vmlp1.m159.iet-gibb.ch@M159.IET-GIBB.CH` in der KDC-Datenbank des Realm `M159.IET-GIBB.CH`. Das KDC kann von jetzt an Host Tickets für die `vmlp1.m159.iet-gibb.ch` ausstellen. Wir testen das im Folgenden. Dazu dient der Befehl `kvno`.

Das Listing unten zeigt, wie Sie mit `kvno` testen können, ob es den Service Principal `host/vmlp1.m159.iet-gibb.ch@M159.IET-GIBB.CH` gibt und ob das KDC für ihn Tickets ausstellen kann: Mit `kinit` holen Sie ein TGT für einen beliebigen Client Principal, im Beispiel für `erim`. Das darauf folgende `kvno` holt das Host Ticket, was auch der anschliessende Aufruf von `klist` belegt.

```
vmadmin@vmlp1:~$ kinit erim
Password for erim@M159.IET-GIBB.CH: sml12345
vmadmin@vmlp1:~$ kvno host/vmlp1.m159.iet-gibb.ch
host/vmlp1.m159.iet-gibb.ch@M159.IET-GIBB.CH: kvno = 1
vmadmin@vmlp1:~$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: erim@M159.IET-GIBB.CH

Valid starting      Expires            Service principal
13.08.2017 12:50:23  13.08.2017 22:50:23  krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
    renew until 14.08.2017 12:50:14
13.08.2017 12:51:02  13.08.2017 22:50:23  host/vmlp1.m159.iet-gibb.ch@M159.IET-
GIBB.CH
    renew until 14.08.2017 12:50:14
vmadmin@vmlp1:~$
```

Zusätzlich zum TGT von `erim` sehen wir nun auch das Serviceticket für den Host `vmlp1`!

## 7.5 Keytabs verwalten

Sie haben gerade den ersten Dienste-Principal in der KDC-Datenbank des Realm `M159.IET-GIBB.CH` angelegt. Dort sind nun die kryptografischen Langzeitschlüssel für `host/vmlp1.m159.iet-gibb.ch@M159.IET-GIBB.CH` gespeichert. Diesen Langzeitschlüssel haben wir mit dem Parameter `-randkey` beim Anlegen des Hostprincipals im Hintergrund erzeugt. Wie Sie wissen, müssen sowohl das KDC als auch der Dienst diese Schlüssel kennen.

### Wichtig zu verstehen hier:

Beim Anlegen des Hostdienst-Principals haben wir mit dem **randkey**-Parameter ein zufälliges **Langzeit-Service-Passwort** erstellt. In unserer Kerberos-Protokoll-Analyse im Excel-Sheet haben wir diesen Schlüssel **K<sub>s</sub>** genannt. Bis jetzt kennt das aber nur der KDC. Der HostService muss dieses aber auch kennen, damit beim `AP_REQ` (Schritt 5) der Service Session Key aus dem Service Ticket entschlüsselt werden kann.

Wie teilen wir dem HostService sein Langzeitschlüssel mit ?

Wir erstellen auf `vmLS4` eine sogenannte **keytab-Datei**, welche den Langzeitschlüssel des Hostdienstes enthält !

### Zwischenfrage:

**In welchem Schritt des Kerberos Protokolls ist dies relevant bzw. kommt diese Tatsache zum Tragen?**

Im Moment ist dieser Langzeitschlüssel des Service nur dem KDC bekannt! Wir müssen nun dafür sorgen, dass dieser Schlüssel auch dem Service bekannt gemacht wird. Dazu verwenden wir sogenannte **keytab**-Dateien.

Kerberisierte Dienste unter MIT Kerberos und anderen Unix-artigen Kerberos-Implementierungen beziehen ihre kryptografischen Langzeitschlüssel aus **keytab-Dateien**. Keytabs sind Dateien, die Schlüssel für verschiedene Principals enthalten können. Pro Principal können in einer Keytab mehrere Schlüssel gespeichert sein, die sich in der Key Version Number (KVNO) und im Verschlüsselungstyp unterscheiden können. Die einer `kadmin`-Sitzung (`kadmin -p laura/admin`) auf der `vmlp1` und dem Subkommando `ktadd` können Sie nun die Schlüssel von `host/vmlp1.m159.iet-gibb.ch@M159.IET-GIBB.CH` in eine Keytab exportieren. Benutzen Sie dazu die Datei `/etc/krb5.keytab`. Das ist unter Linux die Default-Keytab und sie sollte die Host Keys enthalten. Das Listing unten stellt diesen Vorgang dar:

Damit der `ktadd`-Befehl die Keytab-Datei in `/etc` anlegen kann, stellen Sie sicher, dass Sie `kadmin -p user/admin` mit `root` ausführen. Wechseln Sie also von User `vmadmin` auf `user root` mit dem Befehl `sudo su`

```
root@vmlp1:~# kadmin -p laura/admin
Authenticating as principal user/admin with password.
Password for user/admin@M159.IET-GIBB.CH: sm112345
kadmin: ktadd -k /etc/krb5.keytab host/vmlp1.m159.iet-gibb.ch
Der Eintrag für Principal host/vmlp1.m159.iet-gibb.ch mit KVNO 2 und Verschlüsselungstyp aes256-cts-hmac-shal-96 wurde der
Schlüsselstabelle WRFIL:/etc/krb5.keytab hinzugefügt.
Der Eintrag für Principal host/vmlp1.m159.iet-gibb.ch mit KVNO 2 und Verschlüsselungstyp aes128-cts-hmac-shal-96 wurde der
Schlüsselstabelle WRFIL:/etc/krb5.keytab hinzugefügt.
kadmin: quit
root@vmlp1:~#
```

Damit haben Sie eine Keytab mit dem Host Key der `vmlp1` angelegt:  
Kontrolle:

```
root@vmlp1:/home/vmadmin# ls -l /etc/*.keytab
-rw----- 1 root root 192 Sep 21 22:40 /etc/krb5.keytab
M159_SKRIP1_THEMA1_KERBEROS_V2.0.ODT
```

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 57/64

Kerberisierte Services, werden diese dann benutzen, um das Serviceticket ihrer Clients zu überprüfen.

**Zusammenfassung:**

***Diese Keytab-Files sind eine Voraussetzung, dass User via Kerberos von Services des Hosts authentifiziert werden können. In welchem Umfang der Service den Zugang bereitstellt (Authorisierung!), ist nicht mehr Aufgabe von Kerberos. Diese Informationen werden ausserhalb von Kerberos konfiguriert, z. Bsp in einem LDAP-Verzeichnis oder in einer Konfigurationsdatei des Service.***

## 8 Clientkommandos von MIT Kerberos

Das KDC von MIT Kerberos und seine administrativen Dienste sind nun betriebsbereit und Sie haben bereits Erfahrungen mit den Administrationswerkzeugen sammeln können. Damit wird es höchste Zeit, sich mit den MIT-Clientkommandos vertraut zu machen. Kommandos wie `kinit` und `klist` wollen wir als erstes genauer anschauen.

### 8.1 Die Kommandos `kinit` und `klist`

#### 8.1.1 Tickets holen

Das MIT-Kommando `kinit` dient dazu, den Client mit Kerberos Tickets zu versorgen.

**Hauptanwendungszweck ist das Holen von Ticket-Granting Tickets (TGTs).** Dazu führt `kinit` den Austausch mit dem Authentication Server AS (=Komponente des KDC) durch. Der Benutzer benötigt dazu zwei Informationen: den **Principal-Namen des Clients** und dessen **Passwort**. Erhält man beim Aufruf von `kinit` keine Fehlermeldung, dann liegt anschliessend ein neues Ticket im **Credential Cache**. Im Listing unten sind drei Varianten dargestellt, wie Sie ein TGT für die Benutzerin `erim` holen können.

```
root@vmlp1:~# kinit erim@M159.IET-GIBB.CH
Password for erim@M159.IET-GIBB.CH:
root@vmlp1:~# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: erim@M159.IET-GIBB.CH

Valid starting      Expires            Service principal
13.08.2017 17:33:20  14.08.2017 03:33:20  krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
        renew until 14.08.2017 17:33:15

root@vmlp1:~# kinit erim
Password for erim@M159.IET-GIBB.CH:
root@vmlp1:~# kinit
Password for erim@M159.IET-GIBB.CH:
root@vmlp1:~# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: erim@M159.IET-GIBB.CH

Valid starting      Expires            Service principal
13.08.2017 17:33:57  14.08.2017 03:33:57  krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
        renew until 14.08.2017 17:33:51

root@vmlp1:~#
```

An diesen Aufrufen erkennt man die Heuristik, nach der `kinit` den Client Principal bestimmt. Der erste Aufruf enthält den vollständigen Principal-Namen `erim@M159.IET-GIBB.CH`. Dass die Angabe des Realm dabei optional ist, zeigt der zweite Aufruf, der lediglich den kurzen Namen `erim` verwendet. `kinit` ergänzt den Realm dabei automatisch durch den eigene Realm, also den in `/etc/krb5.conf` konfigurierten default-realm.

Aber auch der dritte Aufruf führt zu einem Request für `erim@M159.IET-GIBB.CH`, obwohl dieser Name nicht auf der Kommandozeile angegeben wird. Genau genommen müssen Sie keinen Principal angeben. Wenn es noch keinen Credential Cache gibt, dann nimmt `kinit` einfach Ihren Nutzernamen als erste Komponente des Principal-Namens und ergänzt diesen um den Default-Realm.

Das wäre hier allerdings `root` und den Principal `root@M159.IET-GIBB.CH` gibt es in der Realm `M159.IET-GIBB.CH` nicht. Existiert aber bereits ein Credential Cache, dann wird beim Aufruf von `kinit` ohne weitere Angaben der Client Principal aus diesem Cache bestimmt.

Genau das ist beim dritten Aufruf der Fall: Der zweite Aufruf hat einen Credential Cache hinterlassen, der den Principal-Namen des Clients enthält.

**Achtung:** *kinit* ersetzt einen bereits vorhandenen Credential Cache ohne Rückfrage. Darin enthaltene Tickets gehen dabei verloren. In der Regel ist das kein Problem, denn es wird ja ein neues TGT im Cache abgelegt – die Grundvoraussetzung für die Kerberos-Funktionalität bleibt dadurch bestehen.

Waren die Tickets im alten Cache allerdings für eine andere Clientidentität ausgestellt worden, dann können durch den Wechsel der Identität Probleme entstehen, da nun alle Zugriffe auf kerberisierte Netzwerkdienste mit der neuen Identität erfolgen.

Mit dem Kommando *klist* können Sie sich den Inhalt des Credential Cache ansehen. Das folgende Listing stellt den Inhalt des Credential Cache von erim direkt nach dem Ausführen von *kinit* dar.

```
root@vmlp1:~# klist -f -e
Ticketzwischenspeicher: FILE:/tmp/krb5cc_0
Standard-Principal: erim@M159.IET-GIBB.CH

Valid starting      Expires            Service principal
21.09.2022 22:51:36  22.09.2022 08:51:36  krbtgt/M159.IET-GIBB.CH@M159.IET-
GIBB.CH
           erneuern bis 22.09.2022 22:51:32, Schalter: FPRIA
           Etype (Skey, TKT): aes256-cts-hmac-sha1-96, aes256-cts-hmac-sha1-96
root@vmlp1:~#
```

Dabei gilt:

**klist -f**

zeigt die gesetzten Flags der Tickets im Credential Cache. Die Flags werden durch eine Kombination der Buchstaben R, D, d, i, F, f, P, p, O, I, A, T und H dargestellt.

Sie bedeuten im Einzelnen:

**R** Renewable

**D** Allow-Postdate

**d** Postdated

**i** Invalid

**F** Forwardable

**f** Forwarded

**P** Proxiabale

**p** Proxy

**O** Ok-As-Delegate

**I** Initial

**A** Pre-Authenticated

**T** Transited-Policy-Checked

**H** HW-Authent

**klist -e** Auch die verwendeten Verschlüsselungstypen für die Ticket- Verschlüsselung und für den Session Key können Sie mit *klist* bestimmen. Dazu dient die Option *-e*, die beides anzeigt. Im Listing oben werden die Verschlüsselungstypen in der Zeile *Etype* (*skey*, *tkt*) angezeigt. In diesem Beispiel liegt also der Session Key (*skey*) im Format AES-256 vor und das Ticket (*tkt*) ist ebenfalls mit AES-256 verschlüsselt.

#### **Bemerkung:**

Microsoft verwendet im Standard RC4 und nicht AES, obwohl Microsoft selber schreibt, dass RC4 weniger sicher ist als AES. Microsoft sagt auch, wie man von RC4 auf AES wechselt, wenn man die Sicherheit erhöhen will:

*RC4 encryption is considered less secure than the newer encryption types, AES128-CTS-HMAC-SHA1-96 and AES256-CTS-HMAC-SHA1-96. Security guides such as the Windows 10 Security Technical*

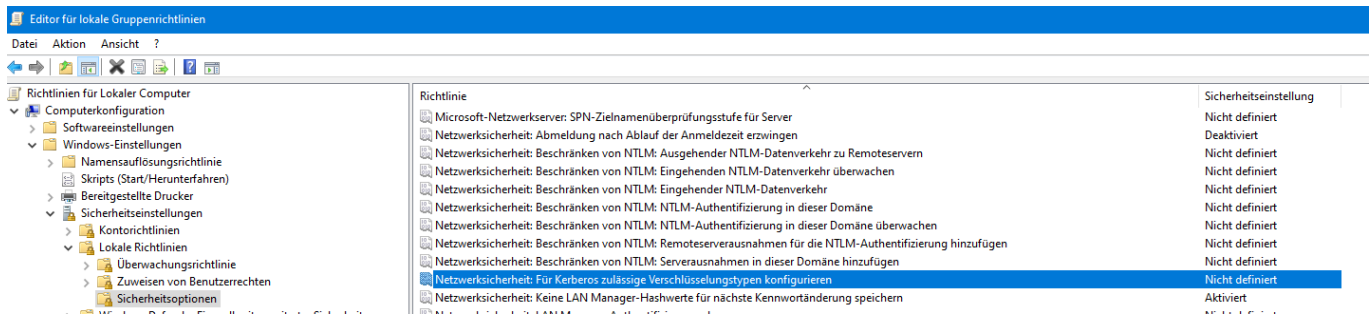
*Implementation Guide provide instructions for improving the security of a computer by configuring it to use only AES128 and/or AES256 encryption (see Kerberos encryption types must be configured to prevent the use of DES and RC4 encryption suites).*

Gelesen hier:

<https://support.microsoft.com/en-us/help/4492348/kerberos-unsupported-etype-error-when-authenticating-across-trust>

Dass RC4 bei Microsoft immer noch im Standard ist, hat wohl mit Kompatibilitätsgründen zu tun...

Der Verschlüsselungstyp kann übrigens in den GPOs eingestellt werden :



## 8.2 Das Kommando kvno

Befindet sich bereits ein TGT im Credential Cache, so kann man damit weitere Tickets beziehen. Das ist typischerweise die Aufgabe des Clients einer kerberisierten Applikation, der dazu den TGS-REQ ausführt.

Mit dem Kommando `kvno` bietet MIT Kerberos die Möglichkeit, einen solchen kerberisierten Client zu simulieren: `kvno` bezieht Dienste-Tickets vom TGS und speichert sie im Credential Cache. Nützlich ist das insbesondere, um die Existenz von Dienste-Principals zu testen.

Das Listing zeigt, wie Sie mit `kvno` testen können, ob es den Service Principal

`host/vmlp1.m159.i-et-gibb.ch` gibt und ob das KDC für ihn Tickets ausstellen kann: Mit

`kinit` holen Sie ein TGT für einen beliebigen Client Principal, im Beispiel für `erim`. Das darauf folgende `kvno` holt das Host Ticket, was auch der anschließende Aufruf von `klist` belegt.

Siehe dazu das Vorgehen in Kap. 7.4.3 am Ende.

```
root@vmlp1:~# kinit erim
Password for erim@M159.IET-GIBB.CH: sml12345
root@vmlp1:~# kvno host/vmlp1.m159.i-et-gibb.ch
host/vmlp1.m159.i-et-gibb.ch@M159.IET-GIBB.CH: kvno = 3
root@vmlp1:~# klist
Ticketzwischenspeicher: FILE:/tmp/krb5cc_0
Standard-Principal: erim@M159.IET-GIBB.CH

Valid starting    Expires          Service principal
21.09.2022 22:54:25  22.09.2022 08:54:25  krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
    erneuern bis  22.09.2022 22:54:22
21.09.2022 22:54:31  22.09.2022 08:54:25  host/vmlp1.m159.i-et-gibb.ch@M159.IET-GIBB.CH
    erneuern bis  22.09.2022 22:54:22
root@vmlp1:~#
```

**Achtung:** `kvno` holt kein neues Dienste-Ticket, wenn dieses bereits im Credential Cache vorliegt. In diesem Fall gibt der Aufruf von `kvno` nur die Versionsnummer des vorhandenen Tickets aus, ohne den TGS-REQ (Schritt 3) erneut auszuführen. Das bedeutet aber auch, dass `kvno` einen alten Wert der Versionsnummer anzeigt, auch wenn sich diese inzwischen



Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 61/64

geändert hat. Für die zuverlässige Bestimmung von Versionsnummern mit `kvno` sollten Sie daher immer zuerst eine Neuinitialisierung des Credential Cache mittels `kinit` durchführen.

### 8.3 Das Kommando `kpasswd`

Das Kommando `kpasswd` dient dazu, die KDC-seitig gespeicherten kryptografischen Langzeitschlüssel eines Principals zu ändern. Dies wird im Hintergrund ausgeführt, wenn Sie unter Windows in einer Domäne das Passwort ändern.

### 8.4 Das Kommando `kdestroy`

Zum Ende einer Kerberos-Sitzung gehört das Aufräumen des Credential Cache. Dieser lässt sich sicher durch das Kommando `kdestroy` entfernen. Damit sind alle Tickets im aktuellen Credential Cache vernichtet. Also:

```

root@vmLP1:/home/vmadmin# klist
Ticketzwischenspeicher: FILE:/tmp/krb5cc_0
Standard-Principal: erim@M159.IET-GIBB.CH

Valid starting    Expires          Service principal
21.09.2022 22:54:25  22.09.2022 08:54:25  krbtgt/M159.IET-GIBB.CH@M159.IET-GIBB.CH
    erneuern bis 22.09.2022 22:54:22
21.09.2022 22:54:31  22.09.2022 08:54:25  host/vmlp1.m159.iet-gibb.ch@M159.IET-GIBB.CH
    erneuern bis 22.09.2022 22:54:22
root@vmLP1:/home/vmadmin# kdestroy
root@vmLP1:/home/vmadmin# klist
klist: No credentials cache found (filename: /tmp/krb5cc_0)

```

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 62/64

## 9 Einfache Kerberos Authentifizierung auf vmLP1

### 9.1 Szenario

#### **Ziel der folgenden Übung:**

Ein User auf vmLP1 soll einmal lokal und einmal vom kdc, also vmLS4, authentifiziert werden. Wir vergeben dem lokalen User ein anderes Passwort als dem gleichen User, welcher als Principal auf dem KDC angelegt ist.

Dabei soll sichtbar werden, dass dieser bei Eingabe des Principal-Passwortes automatisch ein TGT erhält. Wenn das lokale Passwort eingegeben wird, ist kein TGT vorhanden; die Authentifizierung wurde nur lokal auf vmLP1 durchgeführt.

Vorgehen:

Wir verwenden den Principal [maxm@M159.IET-GIBB.CH](mailto:maxm@M159.IET-GIBB.CH) , welcher einmal als Kerberos-Prinzipal und einmal als lokaler Linux-User auf vmLP1 authentifiziert werden soll.

Hier ist festzuhalten, dass Kerberos alleine nicht genügt, damit ein User auf einem Rechner existieren kann. Oder anders ausgedrückt: Die Tatsache, dass Kerberos User-Prinzipals kennt, heisst nicht automatisch, dass sich diese User auf einem System anmelden können. Kerberos ist nur zuständig für die Authentisierung. Kerberos kennt keine Home-Verzeichnisse, User-Gruppen, User-IDs, Filesysteme, etc. All diese Informationen werden anderswo verwaltet, z. Bsp. im LDAP oder in Konfigurationsdaten vom Betriebssystem.

Um unser beschriebenes Szenario durchführen zu können, müssen wir unseren Testuser maxm auch auf vmLP1 physisch anlegen. Was wir dann aber überprüfen können, ist die Instanz der Authentifikation des Users: Wenn er ein TGT-Ticket nach dem Login hat, war der KDC im Spiel; wenn er dieses nicht hat, war das lokale System für die Authentifizierung zuständig.

Damit die vmLP1 während einem login-Vorgang eine Authentifizierung bei einem KDC einer Kerberos REALM vornehmen kann, wird der SSSD-Dienst benötigt. SSSD ist der *System Security Services Daemon*. Der SSSD-Dienst kann weiter auch für LDAP- und Windows ADDS-Abfragen verwendet werden. Wir weisen SSSD an, die Authentifizierung bei einem User-Login beim KDC (vmLS4) vorzunehmen.

## 9.2 Installation von SSSD auf vmLP1

```
sudo apt install sssd-krb5
```

Jetzt muss sssd noch konfiguriert werden. Erstellen Sie die Konfigurationsdatei

```
sudo nano /etc/sss/sss.conf
```

```
[sss]
config_file_version = 2
services = pam
domains = m159.iet-gibb.ch

[pam]

[domain/m159.iet-gibb.ch]
id_provider = proxy
proxy_lib_name = files
auth_provider = krb5
krb5_server = vmls4.m159.iet-gibb.ch
krb5_kpasswd = vmls4.m159.iet-gibb.ch
krb5_realm = M159.IET-GIBB.CH
```

Die Konfiguration verwendet Kerberos für die Authentifizierung ( **auth\_provider** ). Sie verwendet jedoch das lokale System für die Zuweisung der User und Gruppen-Informationen ( **id\_provider** ).

Jetzt müssen noch die Berechtigungen der sssd.conf angepasst werden. Der Daemon wird dann neu gestartet:

```
$ sudo chown root:root /etc/sss/sss.conf
$ sudo chmod 0600 /etc/sss/sss.conf
$ sudo systemctl start sssd
```

SSD ist nun so konfiguriert, dass bei einem Fehler bei der Kerberos-Authentifizierung automatisch auf die lokale Authentifizierung (PAM) geschaltet wird (Fallback). Dies wollen wir gleich testen.

Zuerst müssen wir aber noch den User maxm lokal erstellen, damit dieser ein Home-Verzeichnis, und IDs besitzt. Wichtig ist nun, dass wir dem lokalen User maxm ein anderes Passwort geben als dem User-Principal [maxm@M159.IET-GIBB.CH](mailto:maxm@M159.IET-GIBB.CH) ! **Also NICHT sml12345.**

Modul 159	Skript Directory Services Teil 1 - Grundlagen Kerberos	
tja / GIBB	V2.6	Seite 64/64

maxm lokal auf vmLP1 erstellen mit Befehl:

```
sudo adduser maxm
```

```
vmadmin@vmLP1:~$ sudo adduser maxm
Benutzer »maxm« wird hinzugefügt ...
Neue Gruppe »maxm« (1004) wird hinzugefügt ...
Neuer Benutzer »maxm« (1003) mit Gruppe »maxm« wird hinzugefügt ...
Persönliche Ordner »/home/maxm« wird erstellt ...
Dateien werden von »/etc/skel« kopiert ...
Geben Sie ein neues Passwort ein: welcome123
/var/cache/cracklib/cracklib_dict.pwd: Datei oder Verzeichnis nicht gefunden
Unsicheres Passwort: Passwort scheitert beim Wörterbuchtest - Fehler beim Laden des Wörterbuchs
Geben Sie das neue Passwort erneut ein: welcome123
passwd: Passwort erfolgreich geändert
Benutzerinformationen für maxm werden geändert.
Geben Sie einen neuen Wert an oder drücken Sie ENTER für den Standardwert
    Vollständiger Name []: Max Mustermann
    Zimmernummer []: 007
    Telefon geschäftlich []: 999 999 99 99
    Telefon privat []: 000 000 00 00
    Sonstiges []:
Ist diese Information richtig? [J/N] J
```

Öffnen Sie nun eine Login shell mit

```
sudo login maxm
```

Verwenden Sie dabei einmal das Passwort des User-Principals [maxm@M159.IET-GIBB.CH](mailto:maxm@M159.IET-GIBB.CH) (**sm112345**) und einmal das lokale Passwort (**welcome123**).

Was stellen Sie nach dem login mit **klist** fest? Immer vorausgesetzt, Sie haben das Passwort korrekt eingegeben.

## 10 Aufgabe:

Diese Aufgabe ist optional.

Kerberisieren Sie einen Apache Web-Service. Quelle, z. Bsp:

[http://www.microhowto.info/howto/configure\\_apache\\_to\\_use\\_kerberos\\_authentication.html](http://www.microhowto.info/howto/configure_apache_to_use_kerberos_authentication.html)