R|R|Z|N
Regionales Rechenzentrum für Niedersachsen

Leibniz Universität Hannover

Kerberos: Prinzip und Umsetzung

Sascha Klopp

Inhalt

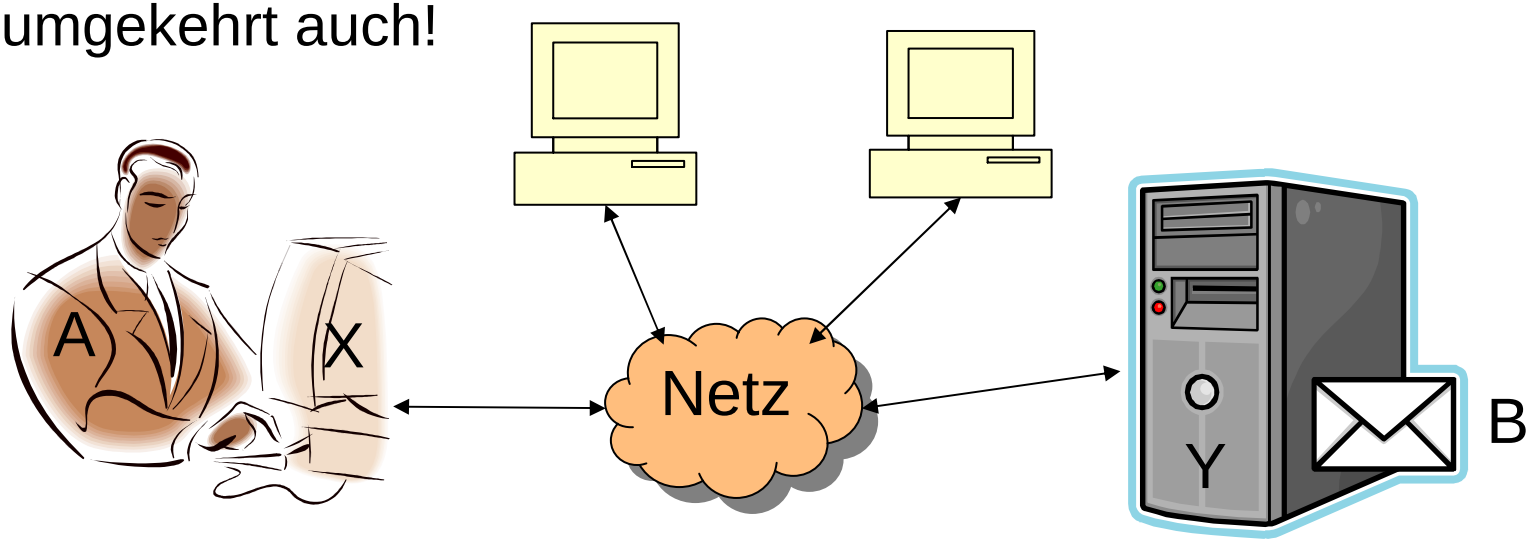
- Prinzip
- Umsetzung
- Anwendungen
- Vor- und Nachteile

Historie

- Das Kerberos-Protokoll wurde im Xerox PARC entwickelt
- Erste öffentliche Version: Kerberos IV
  - wird noch unterstützt, es fehlen aber viele wichtige Features
- Zur Zeit aktuell: Kerberos V (1993)

Prinzip: Problemstellung

- Es liegt ein ungesichertes Netzwerk vor
  - d.h. jeder kann alle Nachrichten abhören und jeder kann jedem beliebige Nachrichten senden.
- Benutzer A am Client X möchte Dienst B auf Server Y nutzen.
- B muss sich über die Identität von A sicher sein, aber umgekehrt auch!



Prinzip: Ticket Granting Ticket

(1) „Benutzer A möchte Kerberos-Dienste nutzen“

Nonce

TGS-Session-Key in zwei Ausfertigungen:

(2)

KDC

1. Verschlüsselt mit User-Key („Key“)
2. Verschlüsselt mit TGS-Key („TGT“)

Prinzip: Service-Authentifizierung

(3) „Benutzer A möchte Service S nutzen“

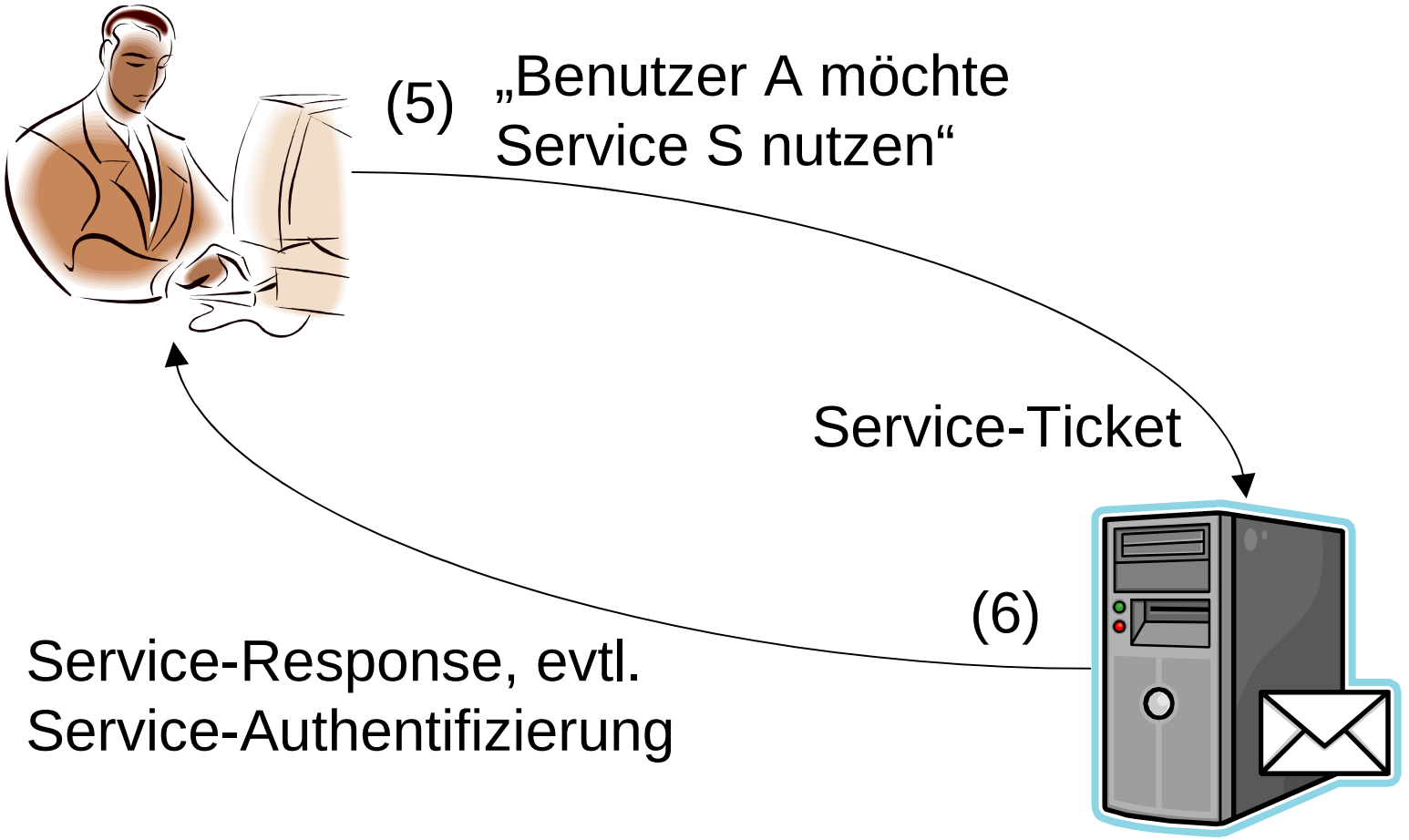
TGT

KDC

(4) Service-Session-Key in zwei Ausfertigungen:

1. Verschlüsselt mit TGS-Session-Key
2. Verschlüsselt mit Service-Key („Service-Ticket“)

Prinzip: Service-Nutzung



Umsetzung: Prinzipale und Realms

- Die beteiligten Nutzer und Services werden unter dem Oberbegriff „Prinzipal“ zusammengefasst
- Alle Prinzipale, die von einem Kerberos-Server verwaltet werden, bilden einen Realm
- Realmnamen entsprechen meistens dem DNS-Namen in Großbuchstaben, also z. B. UNI-HANNOVER.DE

Umsetzung: Prinzipale

- Jedes Prinzipal hat einen Namen und einen Schlüssel
- Der Name besteht aus:
  - primary
  - instance
  - realm
  - (ggf. weitere Felder)
- Geschrieben als
  - *<primary>/<instance>@<realm>*
- Schlüssel sind üblicherweise AES-256-Keys

Umsetzung: Benutzer-Prinzipale

- Benutzer haben üblicherweise Prinzipale in der Form
  - *<primary>*@*<realm>*
- also mit leerer instance
- der primary-Teil entspricht üblicherweise dem Benutzernamen
- Weitere Konvention: Administrationsprinzipale haben die Form
  - *<primary>*/admin@*<realm>*
- Der Schlüssel wird üblicherweise aus einem Passwort generiert

Umsetzung: Service-Prinzipale

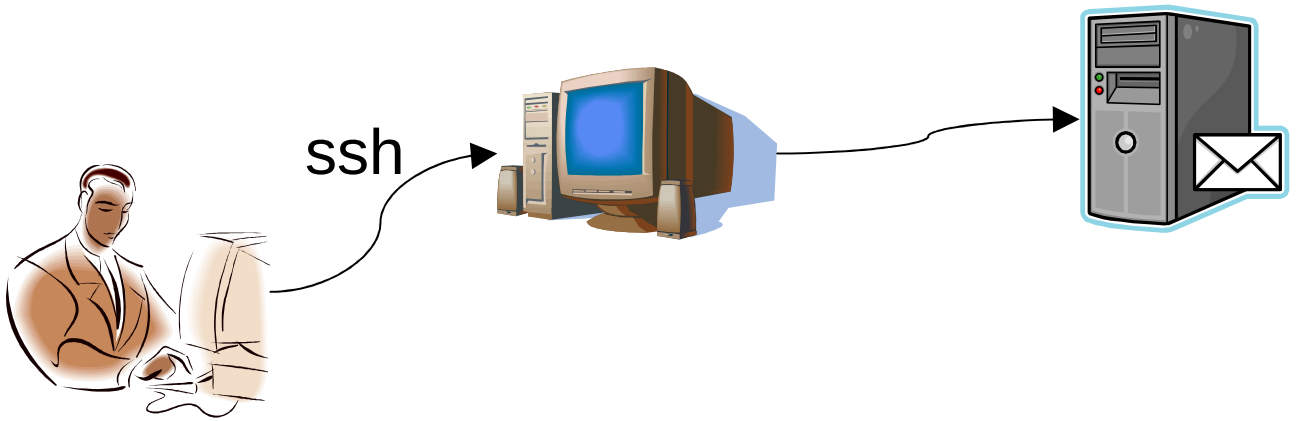
- Service-Prinzipale haben meist die Form
  - *<service>/<hostname>@<realm>*
- Beispielsweise gibt es für den Anmelde“service“ für jeden Rechner ein Prinzipal
  - host/*<hostname>@<realm>*
- Schlüssel werden zufällig generiert und auf die entsprechenden Hosts übertragen

Umsetzung: Berechtigungen

- Kerberos ist prinzipiell nur zur Authentifizierung vorgesehen, Benutzungsberechtigungen (Authorisierung) müssen weiterhin vom Dienst selbst oder anderweitig zentral verwaltet werden.
- Für die Kerberos-Server-Administration (kadmin) selbst gibt es aber eine Rechteverwaltung

Umsetzung: Forwardable Tickets

- Das TGT ist prinzipiell nur für den Client gültig, von dem aus es angefordert wurde.
- Wenn der KDC es erlaubt, und der Benutzer es angefordert hat, kann ein sogenanntes „Forwardable Ticket“ ausgestellt werden. Es erlaubt die automatische Erzeugung eines TGT für einen Remote-Client.



Umsetzung: Nötige Infrastruktur

- Funktionierende Namensauflösung (auch reverse)
- Synchron laufende Uhren auf allen beteiligten Systemen
- Absicherung des Kerberos-Servers, er enthält alle Schlüssel
- Clientsoftware muss das Protokoll unterstützen, „kerberisiert“ sein
  - Standard-Tools (telnet, ftp) werden von der MIT-Implementierung mitgeliefert
  - Auch sonst vielfach vorhanden, z. B. in Standard-Browsern

Umsetzung: Angebotene Implementierungen

- Freie Implementierung wird vom MIT angeboten, für Unix/Linux
- Weitere Implementierungen für Unix/Linux sind Shishi (GNU, aktuelle Version: 0.0.37) und Heimdal (KTH, Schweden)
- Microsoft benutzt das Kerberos-Protokoll für die Authentifizierung ab Windows 2000, hat aber eine Authorisierungsschicht dazuimplementiert

Umsetzung: MIT vs. Heimdal

- Heimdal:
  - Völlig freie Implementierung (wg. ehem. Ausfuhrbeschränkungen)
  - Sehr gute AFS Unterstützung
  - schlechte Dokumentation
  - wenige Anwendungen lassen sich direkt gegen Heimdal bauen
  - Features teilweise noch nicht implementiert
- MIT:
  - Ausgereifte Distribution
  - Viele Anwendungen nur hiermit kompilierbar
  - Weit verbreitet (besonders in USA)
  - AFS Unterstützung nur mit Zusatztools

Implementierung: krb5kdc

- krb5kdc ist die Server-Implementierung für das KDC.
- Erlaubt die Verwaltung mehrerer Realms (bis zu 32) innerhalb eines Prozesses
- Es kann einen Master- und mehrere Slave-KDCs für den gleichen Realm geben.

Implementierung: kadmin

- Benutzer mit admin-Prinzipal können das Tool kadmin benutzen, um den KDC zu administrieren
  - Host-Keys hinzufügen, auf die entsprechenden Hosts übertragen
  - Benutzer anlegen, löschen
  - etc.
- Alternative: Auf KDC einloggen (z. B. ssh), dort kadmin.local benutzen

Implementierung: kinit, klist, kdestroy

- Von der Shell aus kann sich der Nutzer mit kinit ein TGT vom KDC holen.
- klist dient zum Überprüfen der Tickets im Ticket-Cache.
- Mit kdestroy schließlich können Tickets gelöscht werden.

Anwendungen: PAM

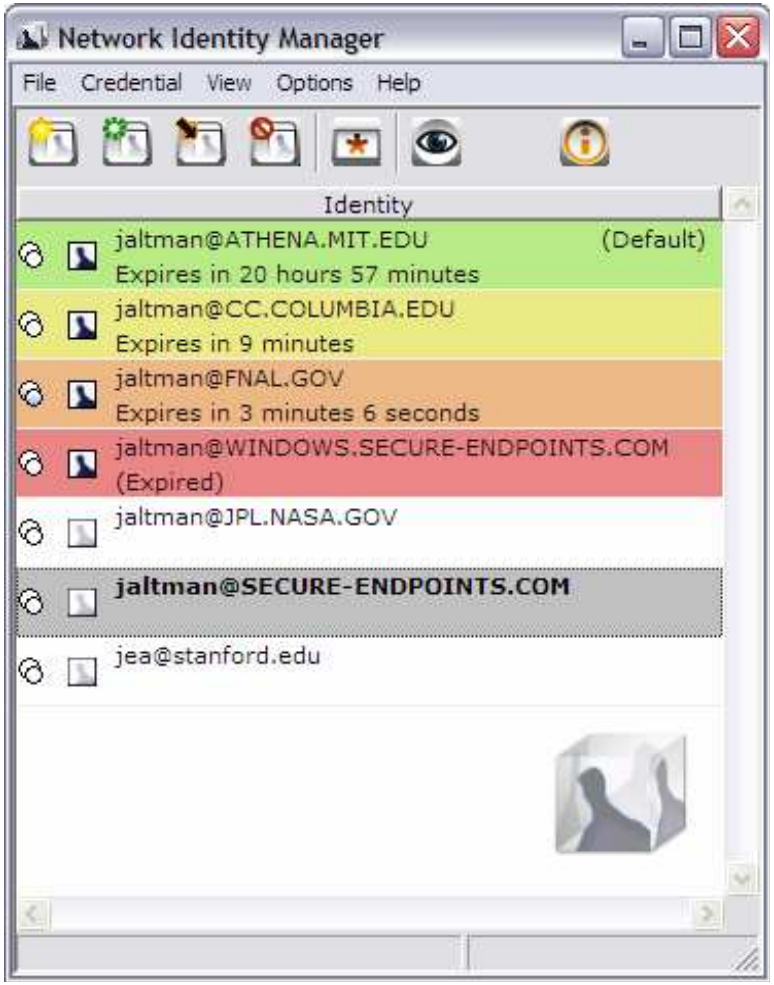
- Für Unix/Linux-Umgebungen
- Alle Dienste, die PAM für die Authentifizierung verwenden, können damit indirekt Kerberos nutzen.
- Beim Einsatz im Login-Prozess ermöglicht dieses Verfahren ein Single-Sign-On.

Anwendungen: Webserver

- Webserver können Zugriffsbeschränkungen durchsetzen, indem der Benutzer ein Kerberos-Ticket vorweisen muss
  - Es gibt ein entsprechendes apache-Modul (mod_auth_kerb)
- Muss natürlich vom Browser unterstützt werden
  - Firefox kann das, in der Grundeinstellung allerdings nur für https

Anwendungen: Network Identity Manager

- wird vom MIT als Teil des Kerberos-for-Windows-Clients angeboten
- verwaltet Kerberos-Tickets unter Windows-Systemen
- weitere Software und Einstellungen sind nötig, um die Tickets für Dienste zu nutzen.



Stärken

- sicher gegen Replay-Attacken und Abhören
- breite Anwendungsunterstützung
  - PAM-Modul existiert
  - Browser, Webserver
  - Windows-Client zur Verwaltung der Tickets
- Ermöglicht prinzipiell ein Single-Sign-On
- „Gereiftes“ Protokoll
  - ziemlich alt
  - wird vielfach eingesetzt (z. B. Windows-Netzwerke)

Nachteile

- KDC ist single point of failure
  - sowohl bzgl. Ausfall (Redundanz ist aber möglich)
  - als auch bzgl. Kompromittierung
- Clients müssen sicher sein (also nicht im Internet-Café)
- Gestaffelte Administration in demselben Realm wird nicht unterstützt
  - z. B. lokale Pool-Admins, die eigene Host-Keys erzeugen und runterladen können
- beteiligte Rechner müssen zeitlich synchronisiert sein

Vielen Dank für Ihr Interesse!

Fragen?