Authentication Service

(Kerberos)

Chun-Jen (James) Chung

Arizona State University
Identity Management System

- **Identity Management (IdM)**
  - Manage identities for authentication & authorization within or across systems

- **Authentication (AuthN)**
  - A process of verifying the *identity* of a user or system
  - Verifying that “you are who you say you are”

- **Authorization (AuthZ)**
  - A process of determine the *privileges* the user or system has
  - Verifying that “you are permitted to do what you are trying to do”

- **Directory Service**
  - A software system that stores, organizes and provides access to information in a directory.
Many-to-Many Authentication

How do users prove their identities when requesting services from machines on the network?

**Naïve solution:** every server knows every user’s password

- **Insecure:** break into one server \(\Rightarrow\) compromise all users
- **Inefficient:** to change password, user must contact every server
Requirements

• Security
  – … against attacks by passive eavesdroppers and actively malicious users

• Transparency
  – Users shouldn’t notice authentication taking place
  – Entering password is Ok, if done rarely

• Scalability
  – Large number of users and servers
Scalability Issue

• Generalizing the model for $m$ users and $n$ services, requires a priori distribution of $m \times n$ shared keys.

• Possible improvement:
  – Use trusted 3rd party, with which each user and service shares a secret key: $m+n$ keys.
  – Also has important security advantages.
Threats

• **User impersonation**
  – Malicious user with access to a workstation pretends to be another user from the same workstation

• **Network address impersonation**
  – Malicious user changes network address of his workstation to impersonate another workstation

• **Eavesdropping, tampering, replay**
  – Malicious user eavesdrops, tampers, or replays other users’ conversations to gain unauthorized access
Solution: Trusted Third Party

- Trusted authentication service on the network
  - Knows all passwords, can grant access to any server
  - Convenient (but also the single point of failure!)
  - Requires high level of physical security
Kerberos

- A Network Authentication Protocol.
- Developed at MIT in the mid 1980s.
- A secret-key based service for providing strong authentication for client/server applications.
- Authentication mediated by a trusted 3rd party
  - Key Distribution Center (KDC)
- Available as open source or in supported commercial software.
Kerberos Objectives

• Standards based *strong authentication system*
• Prevents transmission of passwords over the network
• Servers can build *authorization and access control services* on top of Kerberos
• Provides “single-sign-on” (SSO) capability
  – Only 1 password to remember
  – Only need to enter it once per day (typically)
Kerberos Design Goals

• **Impeccability**
  – no cleartext passwords on the *network*
  – no client passwords on *servers* (server must store secret server key)
  – minimum exposure of client key on workstation (smartcard solution would eliminate this need)

• **Containment**
  – compromise affects only one client (or server)
  – limited authentication lifetime (8 hours, 24 hours, more)

• **Transparency**
  – password required only at login
  – minimum modification to existing applications
Trust: Consolidated Kerberos Model

- Centralized Trust model (vs. decentralized / hierarchical)
Trust: Consolidated Kerberos Model

- Breaking into one host provides a cracker no advantage in breaking into other hosts
- Authentication systems can be viewed as trust propagation systems
  - the Kerberos model is a centralized star model

Users and service providers
Kerberos Architecture

3 major components
3 main exchanges

Client Server Exchange

Authentication Service

Ticket Granting Service

KDC

Authentication Server (AS)

Ticket Granting Server (TGS)

Client

Service Server (SS)
Kerberos Protocol Overview

1. User ID password
2. ticket granting ticket (TGT)
3. service principal name
4. service ticket, optional session key
5. ticket verification request

Information flow for client server in same Kerberos realm
Key Distribution Center

- Responsible for maintaining *master keys* for all *principles* and issuing Kerberos *tickets*
  - Master key: the key shared by user and KDC
- Authentication Service (AS) gives the client a *session key* and a *Ticket Granting Ticket (TGT)*
  - Session key: the key shared by client, server and AS
- Distributes service session keys and ticket for the service via a Ticket Granting Service (TGS)
KDC (cont.)

- Everyone trusts the KDC
- Each user and service registers a secret key with KDC
- KDC holds a database of clients and servers (principals) and their private keys
- principal: a client of the AS, a user, or a service
  - Format: name/instance@realm
  - Examples:
    - user: alice@asu.edu
    - service: printing/cise.asu.edu@asu.edu
Mediated Authentication

- Between Two parties

\[ \text{KDC generates key } K_{ab} \]

A and B perform mutual authentication step to prove that both of you know \( K_{ab} \)

\[ \{K_{ab}, B\}K_a \rightarrow \{K_{ab}, A\}K_b \]
Mediated Authentication

- Involve more parties

\[ \text{A wants to talk to B} \]

\[ \{K_{ab}, B\}K_a \]

\[ \text{Ticket}_b = \{K_{ab}, A\}K_b \]

\[ \text{KDC generates key} \]

\[ K_{ab} \]

\[ \text{I’m A, Ticket}_b = \{K_{ab}, A\}K_b \]

\[ \text{A and B perform mutual authentication step} \]

\[ \text{to prove that both of you know } K_{ab} \]
Replay Attack

- A replay attack occurs when an intruder steals the packet and presents it to the service as if the intruder were the user.

\[
A \xrightarrow{\{K_{ab}, B\}K_a} KDC \text{ generates key } K_{ab} \xrightarrow{\text{Ticket}_b=\{K_{ab}, A\}K_b} B
\]

\[
C \xrightarrow{\text{I'm A, Ticket}_b=\{K_{ab}, A\}K_b} \text{C pretends A to perform mutual authentication step with B}
\]
Kerberos (simplified)

- If a packet is replayed, the timestamp is checked. If the timestamp is earlier or the same as a previous authenticator, the packet is rejected because it's a replay.
- In addition, the time stamp in the authenticator is compared to the server time. It must be within five minutes (by default in Windows).

\[
\begin{align*}
\text{A} & \quad \text{I want to talk to B} \\
\{K_{ab}, B\}K_a & \quad \text{KDC generates key } K_{ab} \\
\text{Ticket}_b = \{K_{ab}, A, \text{lifetime}\}K_b & \quad \text{B}
\end{align*}
\]

\[
\begin{align*}
\text{I'm A, } \{\text{TimeStamp}\}K_{ab}, \text{Ticket}_b = \{K_{ab}, A, \text{lifetime}\}K_b & \\
\{\text{TimeStamp} + 1\}K_{ab} &
\end{align*}
\]
Two-Step Authentication

- Prove identity **once** to obtain special TGS ticket
- Use TGS to get tickets for any network service
What Should a Ticket Include?

- User name
- Server name
- Address of user’s workstation
  - Otherwise, a user on another workstation can steal the ticket and use it to gain access to the server
- Ticket lifetime
- A few other things (session key, etc.)
TGS Benefits

• Single Sign-on (SSO) capability
• Limits exposure of user’s password
  – Client’s workstation can forget the password immediately after using it in the early stages of the protocol.
  – Less data encrypted with the user’s secret key travels over the network, limiting attacker’s access to data that could be used in an offline dictionary attack.
  – Dictionary attack:
    • tries only those possibilities which are most likely to succeed, typically derived from a list of words for example a dictionary
Ticket Granting Service

- **Problem: Transparency**
  - user should provide password once upon initial login, and should not be asked for it on every service request
  - workstation should not store the password, except for the brief initial login

- **Solution: Ticket-Granting Service (TGS)**
  - store session key on workstation in lieu of password
  - TGS runs on same host as Kerberos (needs access to $K_c$ and $K_s$ keys)
Kerberos Protocol

Authenticating the identity and obtaining TGT
1. \([ID_C, ID_{TGS}, TS_1]\)
2. \(\{K_{C,TGS}, ID_{TGS}, TS_2, lifetime_1, Ticket_{TGS}\}K_C\)
   \(Ticket_{TGS} = \{K_{C,TGS}, ID_C, AD_C, ID_{TGS}, TS_2, lifetime_1\}K_{TGS}\)

Obtaining Service-granting-ticket
3. \([ID_S, Ticket_{TGS}, Auth_{C1}]\), \(Auth_{C1} = \{ID_C, AD_C, TS_3\}K_{C,TGS}\)
4. \(\{K_{C,S}, ID_S, TS_4, Ticket_S\}K_{C,TGS}\)
   \(Ticket_S = \{K_{C,S}, ID_C, AD_C, ID_S, TS_4, lifetime_2\}K_S\)

Obtaining service
5. \([Ticket_S, Auth_{C2}]\), \(Auth_{C2} = \{ID_C, AD_C, ID_S, TS_S, lifetime_2\}K_{C,S}\)
6. \(\{TS_{S+1}\}K_{C,S}\)
Symmetric Keys in Kerberos

- $K_c$ is long-term key of client $C$
  - Derived from user’s password
  - Known to client and key distribution center (KDC)

- $K_{TGS}$ is long-term key of TGS
  - Known to KDC and ticket granting service (TGS)

- $K_s$ is long-term key of network service $S$
  - Known to $V$ and TGS; each service $S$ has a separate key

- $K_{c,TGS}$ is short-term session key between $C$ and TGS
  - Created by KDC, known to $C$ and TGS

- $K_{c,s}$ is short-term session key between $C$ and $S$
  - Created by TGS, known to $C$ and $V$
Kerberos Dictionary Attack

- First two messages reveal known plaintext for dictionary attack
- First message can be sent by anyone
- Kerberos v5 has pre-authentication option to prevent this attack
Pre-authentication

- Kerberos 5 added pre-authentication
  - Client is required to prove it’s identity to the Kerberos AS in the first step.
  - By supplying an *encrypted timestamp* (encrypted with users secret key)
  - This prevents an active attacker being able to easily obtain data from the KDC encrypted with any user’s key, then able to mount an offline dictionary attack.

**Key distribution center (KDC)**

**USER=Joe; service=TGS, {TS}K_e**

Joe the User
Realms

- A Realm is an *authentication domain*
  - one Kerberos database and a set of KDCs
- Hierarchical organization (new in v5)
  - Active Directory in Windows
- One or two way authentication
- Cross-realm authentication
  - transitive cross-realm
  - direct between realms
Kerberos Design Features

• Uses timestamps to avoid replay.
  – Requires time synchronized within a small window (5 minutes)

• Uses DES-based symmetric key cryptography for user authentication
  – Converting user’s password to a DES key

• Stateless
Kerberos - Summary

• Authentication method:
  – User’s enter password on local machine only
  – Authenticated via central KDC once per day
  – No passwords travel over the network

• Single Sign-on (via TGS)
  – KDC gives you a special “ticket”, the TGT, usually good for rest of the day.
  – This ticket can be used to get other service tickets allowing user to access them.
Advantages of Kerberos (1)

• Passwords aren’t exposed to eavesdropping

• Password is only typed to the local workstation
  – It never travels over the network
  – It is never transmitted to a remote server

• Password guessing more difficult

• Single Sign-on
  – More convenient. Only one password, entered once
  – Users may be less likely to store passwords
Advantages of Kerberos (2)

• Stolen tickets hard to reuse
  – Need authenticator as well, which can’t be reused

• Much easier to effectively secure a small set of limited access machines (the KDC’s)

• Easier to recover from host compromises

• Centralized user account administration
Kerberos Caveats

• Kerberos server can impersonate anyone

• KDC is a single point of failure
  – Can have replicated KDC’s

• KDC should be a performance bottleneck
  – Everyone needs to communicate with it frequently
  – Not a practical concern these days
  – Having multiple KDC’s alleviates the problem

• If local workstation is compromised, user’s password could be stolen
  – Only use a desktop machine or laptop that you trust
  – Use hardware token pre-authentication
Kerberos, LDAP, and AD

- Kerberos is for authentication only and provides Single Sign-on (SSO)
- LDAP can be used for authentication, authorization, and name services (no SSO)
- Active Directory is a kerberized directory service with an LDAP interface
- Use Kerberos for authN, LDAP for authZ and name services