The Authentication Problem

- Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network.
- Restrict access to authorized users and to be able to authenticate requests for service.
Can we rely on workstation for authentication service?

- Three threats:
  - A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
  - A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
  - A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

Authentication Service Provided by Kerberos

- A centralized authentication service
  - Authenticate users to services
  - Authenticate services to users
  - Servers are relieved of the burden of maintaining authentication information.

- Facts about Kerberos
  - Rely exclusively on conventional encryption.
    - Public key based Kerberos has been considered.
  - Stateless: Kerberos server doesn’t need to maintain the state information about any entities being authenticated.
Requirements for Kerberos

- **Secure**
  - A network eavesdropper should not be able to obtain the necessary to impersonate a user.
- **Reliable**
  - Kerberos should be highly available and should employ a distributed server architecture.
- **Transparent**
  - The user shouldn’t be aware that authentication is taking place.
- **Scalable**
  - The system should be capable of supporting large numbers of clients and servers.

The Kerberos Protocol

- **Outline of the introduction to the Kerberos protocol**
  - A simple authentication protocol
  - A more secure authentication protocol
  - Kerberos Version 4 authentication protocol
A Simple Authentication Protocol

- Use an authentication server (AS)
- Basic idea: use a ticket to authenticate a user to a server.
- Protocol
  1. C → AS: \( ID_C \| P_C \| ID_V \)
  2. AS → C: Ticket
  3. C → V: \( ID_C \| Ticket \)
     - \( Ticket = E_{K_V}[ID_C \| AD_C \| ID_V] \)

A Simple Authentication Protocol (Cont’d)

- Advantages
  - A centralized authentication service
- Weaknesses
  - A user needs to enter a password for every different service.
  - Password is transmitted in plaintext.
A More Secure Authentication Protocol

- A new server: ticket-granting server (TGS)
- Protocol
  - Once per user logon session
    1) $C \rightarrow AS$: $ID_C \ || \ ID_{tgs}$
    2) $AS \rightarrow C$: $E_{K_C}[Ticket_{tgs}]$
  - Once per type of service
    3) $C \rightarrow TGS$: $ID_C \ || \ ID_V \ || \ Ticket_{tgs}$
    4) $TGS \rightarrow C$: $Ticket_V$
  - Once per service session
    5) $C \rightarrow V$: $ID_C \ || \ Ticket_V$
  - $Ticket_{tgs} = E_{K_{tgs}}[ID_C \ || \ AD_C \ || \ ID_{tgs} \ || \ TS_1 \ || \ lifetime_1]$
  - $Ticket_V = E_{K_V}[ID_C \ || \ AD_C \ || \ ID_V \ || \ TS_2 \ || \ lifetime_2]$

A More Secure Authentication Protocol (Cont’d)

- Ticket-granting ticket (TGT): $Ticket_{tgs}$
- Service-granting ticket: $Ticket_V$
- Weaknesses
  - Replay attack: No authentication of the valid ownership of the tickets.
  - No authentication of the servers.

- What are the components in the tickets?
- Why do we have them?
Kerberos Version 4 Protocol

- Basic idea to address the previous weaknesses
  - Session key
    - Authentication of the valid ownership of the tickets
    - Provide authentication of servers.

Kerberos Version 4 Protocol (Cont’d)

- Authentication Service Exchange: to obtain ticket-granting ticket
  1) C → AS: ID_C || ID_tgs || TS_1
  2) AS → C: E_{K_c} [K_{C,tgs} || ID_tgs || TS_2 || Lifetime_2 || Ticket_tgs]
     - Ticket_tgs = E_{K_{tgs}} [K_{C,tgs} || ID_C || AD_C || ID_tgs || TS_2 || Lifetime_2]
Kerberos Version 4 Protocol (Cont’d)

- Ticket-Granting Service Exchange: to obtain service-granting ticket
  3) C → TGS: ID_V || Ticket_{tgs} || Authenticator_c
  4) TGS → C: E_{K_{C,tgs}}[K_{C,V} || ID_V || TS_4 || Lifetime_4 || Ticket_V]
     - Ticket_{tgs} = E_{K_{tgs}}[K_{C,tgs} || ID_C || AD_C || ID_{tgs} || TS_2 || Lifetime_2]
     - Ticket_V = E_{K_{V}}[K_{C,V} || ID_C || AD_C || ID_V || TS_4 || Lifetime_4]
     - Authenticator_c = E_{K_{C,tgs}}[ID_C || AD_C || TS_3]

Kerberos Version 4 Protocol (Cont’d)

- Client/Server Authentication Exchange: to obtain service
  5) C → V: Ticket_V || Authenticator_c
  6) V → C: E_{K_{C,V}}[TS_5 + 1]
     - Ticket_V = E_{K_{V}}[K_{C,V} || ID_C || AD_C || ID_V || TS_4 || Lifetime_4]
     - Authenticator_c = E_{K_{C,V}}[ID_C || AD_C || TS_5]
The Whole Picture

Keberos
Authentication Server (AS)

Ticket-Granting Server (TGS)

Server

1. Request TGT
2. TGT + session key
3. Request SGT
4. Ticket + session key
5. Request service
6. Server authenticator

Kerberos Deployment

- The Kerberos server must have the user ID and hashed password of all participating users in its database.
- The Kerberos server must share a secret key with each server.
- Kerberos are “physically” secured
- Kerberos libraries are distributed on all nodes with users, applications, and other Kerberos-controlled resources
Replicated Kerberos

- Multiple replica of Kerberos - availability and performance
- Keeping Kerberos databases consistent
  - Single master Kerberos as the point of direct update to principals’ database entries
  - Updated database is downloaded from the master to all replica Kerberos
  - Periodic download or on-demand

Kerberos Realms and Multiple Kerberi

- Kerberos realm
  - A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers
- Inter-realm authentication
  - The Kerberos server in each interoperating realm shares a secret key with the server in the other realm. The two Kerberos servers are registered with each other.
Inter-realm Authentication

1. Request ticket for local TGS
2. Ticket for local TGS
3. Request ticket for remote TGS
4. Ticket for remote TGS
5. Request ticket for remote server
6. Ticket for remote server
7. Ticket for remote server
8. Remote server authenticator