User Authentication

- refers to the authentication functions that have been developed
- to support network-based use authentication

- one of the earliest and
- also one of the most widely used authentication services: Kerberos.

Kerberos

- Kerberos is a key distribution and user authentication service developed at MIT

- The problem that Kerberos addresses is this:
  - Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network.
  - We would like for servers to be able to restrict access to authorized users and to be able to authenticate requests for service.
  - In this environment, a workstation cannot be trusted to identify its users correctly to network services

Kerberos

- In particular, the following three threats exist:
  - 1. A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
  - 2. 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.
  - 3. A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

provides a centralized authentication server and two versions

Kerberos

- Rather than building in elaborate authentication protocols at each server,
- Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.
- Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption
- Two versions of Kerberos are in common use.
  - Version 4,
    - implementations still exist.
    - Version 4 of Kerberos makes use of DES, in a rather elaborate protocol, to provide the authentication service
  - Version 5
    - corrects some of the security deficiencies of version 4 and
    - has been issued as a proposed Internet Standard

Motivation for the Kerberos approach

- In today's distributed architecture consisting of
  - dedicated user workstations (clients) and
  - distributed or centralized servers,
- three approaches to security can be envisioned.
- 1. Rely on each individual client workstation to assure the identity of its user or users and rely on each server to enforce a security policy based on user identification (ID).
- 2. Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user.
- 3. Require the user to prove his or her identity for each service invoked. Also require that servers prove their identity to clients.

Motivation for the Kerberos approach

- In a more open environment in which network connections to other machines are supported, the third approach is needed to protect user information and resources housed at the server.

- Kerberos supports this third approach.

- Kerberos assumes a distributed client/server architecture and employs one or more Kerberos servers to provide an authentication service

Motivation for the Kerberos approach

- Requirements for Kerberos
- Secure: A network eavesdropper should not be able to obtain the necessary information to impersonate a user.
  - More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
- Reliable: For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services.
  - Hence, Kerberos should be highly reliable and should employ a distributed server architecture with one system able to back up another.
- Transparent: Ideally, the user should not be aware that authentication is taking place beyond the requirement to enter a password.
- Scalable: The system should be capable of supporting large numbers of clients and servers.
  - This suggests a modular, distributed architecture.

Motivation for the Kerberos approach

- Requirements for Kerberos
- To support these requirements,
- the overall scheme of Kerberos is that of a
- trusted third-party authentication service

- It is trusted in the sense that clients and servers trust Kerberos to mediate their mutual authentication.

- Assuming the Kerberos protocol is well designed, then the authentication service is secure if the Kerberos server itself is secure

confirming the identities of clients for each client/server interaction is burden on each server

A Simple Authentication Dialogue as protocol of Kerberos

- In an unprotected network environment, any client can apply to any server for service.
- The obvious security risk is that of impersonation.
    - An opponent can pretend to be another client and obtain unauthorized privileges on server machines.
- To counter this threat, servers must be able to confirm the identities of clients who request service.
- Each server can be required to undertake this task for each client/server interaction,
- but in an open environment, this places a substantial burden on each server.

A Simple Authentication Dialogue

- An alternative is to use an authentication server (AS)
  - that knows the passwords of all users and stores these in a centralized database.
  - shares a unique secret key with each server.
  - These keys have been distributed physically or in some other secure manner.

A Simple Authentication Dialogue

- Consider the following hypothetical dialogue:

$$\begin{aligned}
&\textbf{(1)}\ \mathrm{C} \rightarrow \mathrm{AS}: \quad ID_C \| P_C \| ID_V \\
&\textbf{(2)}\ \mathrm{AS} \rightarrow \mathrm{C}: \quad Ticket \\
&\textbf{(3)}\ \mathrm{C} \rightarrow \mathrm{V}: \quad ID_C \| Ticket \\
&Ticket = E(K_v, [ID_C \| AD_C \| ID_V])
\end{aligned}$$

- where
  - C = client
  - AS = authentication server
  - V = server
  - ID_C = identifier of user on C
  - ID_V = identifier of V
  - P_C = password of user on C
  - AD_C = network address of C
  - K_v = secret encryption key shared by AS and V

- In this scenario, the user logs on to a workstation and requests access to server V.
- The client module C in the user's workstation requests the user's password and then sends a message to the AS that includes the user's ID, the server's ID, and the user's password.
- The AS checks its database to see if the user has supplied the proper password for this user ID and whether this user is permitted access to server V.
- If both tests are passed, the AS accepts the user as authentic and must now convince the server that this user is authentic.
- To do so, the AS creates a ticket that contains the user's ID and network address and the server's ID.
- This ticket is encrypted using the secret key shared by the AS and this server.
- This ticket is then sent back to C.
- Because the ticket is encrypted, it cannot be altered by C or by an opponent.

A Simple Authentication Dialogue

- Consider the following hypothetical dialogue:

$$\begin{aligned}
&\textbf{(1)}\ \mathrm{C} \rightarrow \mathrm{AS}: && ID_C \| P_C \| ID_V \\
&\textbf{(2)}\ \mathrm{AS} \rightarrow \mathrm{C}: && Ticket \\
&\textbf{(3)}\ \mathrm{C} \rightarrow \mathrm{V}: && ID_C \| Ticket \\
&Ticket = E(K_v, [ID_C \| AD_C \| ID_V])
\end{aligned}$$

- where
  - C = client
  - AS = authentication server
  - V = server
  - ID_C = identifier of user on C
  - ID_V = identifier of V
  - P_C = password of user on C
  - AD_C = network address of C
  - K_v = secret encryption key shared by AS and V

- With this ticket, C can now apply to V for service.
- C sends a message to V containing C's ID and the ticket.
- V decrypts the ticket and verifies that the user ID in the ticket is the same as the unencrypted user ID in the message.
- If these two match, the server considers the user authenticated and grants the requested service.

A Simple Authentication Dialogue

- Consider the following hypothetical dialogue:

$$\begin{aligned}
&\textbf{(1)}\ \mathrm{C} \rightarrow \mathrm{AS}: && ID_C \| P_C \| ID_V \\
&\textbf{(2)}\ \mathrm{AS} \rightarrow \mathrm{C}: && Ticket \\
&\textbf{(3)}\ \mathrm{C} \rightarrow \mathrm{V}: && ID_C \| Ticket \\
&Ticket = E(K_v, [ID_C \| AD_C \| ID_V])
\end{aligned}$$

- where
  - C = client
  - AS = authentication server
  - V = server
  - ID_C = identifier of user on C
  - ID_V = identifier of V
  - P_C = password of user on C
  - AD_C = network address of C
  - K_v = secret encryption key shared by AS and V

- Each of the ingredients of message (3) is significant.
- The ticket is encrypted to prevent alteration or forgery.
- The server's ID (ID_V) is included in the ticket so that the server can verify that it has decrypted the ticket properly.
- ID_C is included in the ticket to indicate that this ticket has been issued on behalf of C.
- Finally, AD_C serves to grant access to the user on that other workstation.
- Now the ticket is valid only if it is transmitted from the same workstation that initially requested the ticket.

A More Secure Authentication Dialogue

- Although the foregoing scenario solves some of the problems of authentication in an open network environment, problems remain.
- Two in particular stand out.
- First, we would like to minimize the number of times that a user has to enter a password.
- Suppose each ticket can be used only once.
- If user C logs on to a workstation in the morning and wishes to check his or her mail at a mail server, C must supply a password to get a ticket for the mail server.
- If C wishes to check the mail several times during the day, each attempt requires reentering the password.
- We can improve matters by saying that tickets are reusable.
- For a single logon session, the workstation can store the mail server ticket after it is received and use it on behalf of the user for multiple accesses to the mail server.

A More Secure Authentication Dialogue

- The second problem is that the earlier scenario involved a plaintext transmission of the password [message (1)].

- An eavesdropper could capture the password and use any service accessible to the victim.

- To solve these additional problems, we introduce a scheme for avoiding plaintext passwords and a new server, known as the ticket-granting server (TGS).

A More Secure Authentication Dialogue

- The new scenario is as follows.

Once per user logon session:

(1) $C \rightarrow AS: \quad ID_C \| ID_{tgs}$

(2) $AS \rightarrow C: \quad E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS: \quad ID_C \| ID_V \| Ticket_{tgs}$

(4) $TGS \rightarrow C: \quad Ticket_v$

Once per service session:

(5) $C \rightarrow V: \quad ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$
$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

- The new service, TGS, issues tickets to users who have been authenticated to AS.
- Thus, the user first requests a ticket-granting ticket ($Ticket_{tgs}$) from the AS.
- The client module in the user workstation saves this ticket.
- Each time the user requires access to a new service, the client applies to the TGS, using the ticket to authenticate itself.
- The TGS then grants a ticket for the particular service.
- The client saves each service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested

A More Secure Authentication Dialogue

Once per user logon session:

(1) C → AS: $ID_C \| ID_{tgs}$

(2) AS → C: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) C → TGS: $ID_C \| ID_V \| Ticket_{tgs}$

(4) TGS → C: $Ticket_v$

Once per service session:

(5) C → V: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$
$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

- Let us look at the details of this scheme:
- 1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with the TGS ID, indicating a request to use the TGS service.
- 2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password (K_c), which is already stored at the AS.
- When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message.
- If the correct password is supplied, the ticket is successfully recovered.

- Because only the correct user should know the password, only the correct user can recover the ticket.
- Thus, we have used the password to obtain credentials from Kerberos without having to transmit the password in plaintext.

A More Secure Authentication Dialogue

Once per user logon session:

(1) C → AS: $ID_C \| ID_{tgs}$

(2) AS → C: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) C → TGS: $ID_C \| ID_V \| Ticket_{tgs}$

(4) TGS → C: $Ticket_v$

Once per service session:

(5) C → V: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$

$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

- The ticket itself consists of the ID and network address of the user, and the ID of the TGS.
- The idea is that the client can use this ticket to request multiple service-granting tickets.
- So the ticket-granting ticket is to be reusable.
- If the opponent would be able to reuse the ticket
  - the ticket includes a timestamp, indicating the date and time at which the ticket was issued, and a lifetime, indicating the length of time for which the ticket is valid (e.g., eight hours).
- Thus, the client now has a reusable ticket and need not bother the user for a password for each new service request.
- Finally, note that the ticket-granting ticket is encrypted with a secret key known only to the AS and the TGS.
  - This prevents alteration of the ticket.
- The ticket is reencrypted with a key based on the user's password.
  - This assures that the ticket can be recovered only by the correct user, providing the authentication.

A More Secure Authentication Dialogue

Once per user logon session:

(1) C → AS: $ID_C \| ID_{tgs}$

(2) AS → C: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) C → TGS: $ID_C \| ID_V \| Ticket_{tgs}$

(4) TGS → C: $Ticket_v$

Once per service session:

(5) C → V: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$

$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

- Now that the client has a ticket-granting ticket, access to any server can be obtained with steps 3 and 4.
- 3. The client requests a service-granting ticket on behalf of the user.
  - For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.
- 4. The TGS decrypts the incoming ticket using a key shared only by the AS and the TGS (K_{tgs}) and verifies the success of the decryption by the presence of its ID.
- It checks to make sure that the lifetime has not expired.
- Then it compares the user ID and network address with the incoming information to authenticate the user.
- If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.

A More Secure Authentication Dialogue

Once per user logon session:

(1) $C \rightarrow AS: \quad ID_C \| ID_{tgs}$

(2) $AS \rightarrow C: \quad E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS: \quad ID_C \| ID_V \| Ticket_{tgs}$

(4) $TGS \rightarrow C: \quad Ticket_v$

Once per service session:

(5) $C \rightarrow V: \quad ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$
$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

- The service-granting ticket contains a timestamp and lifetime.
- If the user wants access to the same service at a later time, the client can simply use the previously acquired service-granting ticket and need not bother the user for a password.
- Note that the ticket is encrypted with a secret key (K_v) known only to the TGS and the server, preventing alteration.

- Finally, with a particular service-granting ticket, the client can gain access to the corresponding service with step 5.
- 5. The client requests access to a service on behalf of the user.
- For this purpose, the client transmits a message to the server containing the user's ID and the service granting ticket.
- The server authenticates by using the contents of the ticket.

- This new scenario satisfies the two requirements of only one password query per user session and protection of the user password.

A More Secure Authentication Dialogue

Once per user logon session:

(1) $C \rightarrow AS: \quad ID_C \| ID_{tgs}$

(2) $AS \rightarrow C: \quad E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS: \quad ID_C \| ID_V \| Ticket_{tgs}$

(4) $TGS \rightarrow C: \quad Ticket_v$

Once per service session:

(5) $C \rightarrow V: \quad ID_C \| Ticket_v$

$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$

$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$

Message (1)	Client requests ticket-granting ticket.
ID_C	Tells AS identity of user from this client.
ID_{tgs}	Tells AS that user requests access to TGS.
TS_1	Allows AS to verify that client's clock is synchronized with that of AS.
Message (2)	AS returns ticket-granting ticket.
K_c	Encryption is based on user's password, enabling AS and client to verify password, and protecting contents of message (2).
ID_{tgs}	Confirms that this ticket is for the TGS.
TS_2	Informs client of time this ticket was issued.
$Lifetime_2$	Informs client of the lifetime of this ticket.
$Ticket_{tgs}$	Ticket to be used by client to access TGS.

(a) Authentication Service Exchange

A More Secure Authentication Dialogue

Once per user logon session:

(1) C → AS: $ID_C \| ID_{tgs}$

(2) AS → C: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) C → TGS: $ID_C \| ID_V \| Ticket_{tgs}$

(4) TGS → C: $Ticket_v$

Once per service session:

(5) C → V: $ID_C \| Ticket_v$

$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$

$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$

Message (3)	Client requests service-granting ticket.
ID_V	Tells TGS that user requests access to server V.
$Ticket_{tgs}$	Assures TGS that this user has been authenticated by AS.

Message (4)	TGS returns service-granting ticket.
ID_V	Confirms that this ticket is for server V.
TS_4	Informs client of time this ticket was issued.
$Ticket_V$	Ticket to be used by client to access server V.
$Ticket_{tgs}$	Reusable so that user does not have to reenter password.
K_{tgs}	Ticket is encrypted with key known only to AS and TGS, to prevent tampering.

ID_C	Indicates the rightful owner of this ticket.
AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket.
ID_{tgs}	Assures server that it has decrypted ticket properly.
TS_2	Informs TGS of time this ticket was issued.
$Lifetime_2$	Prevents replay after ticket has expired.

ID_C	Must match ID in ticket to authenticate ticket.
AD_C	Must match address in ticket to authenticate ticket.
TS_3	Informs TGS of time this authenticator was generated.

(b) Ticket-Granting Service Exchange

A More Secure Authentication Dialogue

Once per user logon session:

(1) C → AS: $ID_C \| ID_{tgs}$

(2) AS → C: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) C → TGS: $ID_C \| ID_V \| Ticket_{tgs}$

(4) TGS → C: $Ticket_v$

Once per service session:

(5) C → V: $ID_C \| Ticket_v$

$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$

$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$

Message (5)	Client requests service.
$Ticket_V$	Assures server that this user has been authenticated by AS.

(c) Client/Server Authentication Exchange