Kerberos: An authentication service for open network systems

J. G. Steiner, C. Neuman and J. I. Schiller


Passwords are ineffective in open network systems because they are vulnerable to snooping. Kerberos provides secure authentication without exchanges of passwords and without using public-key encryption. It assumes that the network can contain intruders who can listen to messages, store them and replay them later.

1. General Organization

A Kerberos system consists of a Kerberos server and a ticket-granting server (TGS). Once users are authenticated by Kerberos, they receive a TGS ticket. This ticket will allow them to request from the TGS tickets for all the other servers.

Kerberos keeps keys for (a) all the system users and (b) the ticket-granting server (TGS). It assumes that all clients can trust their workstation. Hence it does not work well with public workstations.

2. The Kerberos Protocol

Assume that a client c, logged on a trusted workstation WS, wants to access a server s through an insecure network.

1. Client provides the workstation WS with its name c:

   c → WS: c

   and WS sends to Kerberos a request for a ticket for the TGS:

   WS → K: c, tgs

2. Kerberos sends to WS a ticket T_{c,tgs} and a random session key K_{c,tgs}:

   K → WS: \{K_{c,tgs}, T_{c,tgs}\}K_{tgs}K_{c}

   The ticket T_{c,tgs} authenticates the client for the TGS. It contains the client's name c, the name of the ticket-granting service tgs, the IP address of the client addr, the current time timestamp, a lifetime life and the random session key K_{c,tgs}:

   T_{c,tgs} = c, tgs, addr, timestamp, life, K_{c,tgs}

   To make the ticket tamperproof, Kerberos encrypts it with K_{tgs} the secret key of the TGS.

   As the whole message is encrypted with the user key K_{c}, it makes no sense to anyone else.

3. When WS receives this message it prompts the client c for his or her password, computes from it the user key K_{c} = fn(password) and uses K_{c} to decrypt the message. It then sends a message to the TGS with the name of the service s it wants to utilize, the still encrypted ticket T_{c,tgs} and an authenticator A_{c,tgs} containing the client name c, its address addr and the current time timestamp encrypted with the session key K_{c,tgs}:

   WS → TGS: s, \{T_{c,tgs}\}K_{tgs}, \{A_{c,tgs}\}K_{c,tgs}

   where

   A_{c,tgs} = c, addr, timestamp

4. The TGS will reply by sending to the workstation a ticket T_{c,s} for the service s and a new random session key K_{c,s}:

   TGS → WS: \{K_{c,s}, T_{c,s}\}K_{c,tgs}

   T_{c,s} authenticates the user for server s. It contains, the client's name c, the name of the service s, the IP address of the client addr, the current time timestamp, a lifetime life and the random session key K_{c,s}:

   T_{c,s} = c, s, addr, timestamp, life, K_{c,s}

   T_{c,s} is encrypted with K_{s} the secret key of server S.

5. WS sends to the server s a message with the still encrypted ticket T_{c,s} and an authenticator A_{c,s} containing the client’s name c, its address addr and the current time timestamp encrypted with the session key K_{c,s}:

   WS → s: \{T_{c,s}\}K_{s}, \{A_{c,s}\}K_{c,s}

   where

   A_{c,s} = c, addr, timestamp

6. If the client asked the server to authenticate itself, the server will reply with the authenticator time stamp minus one encrypted with the session key:

   s → WS: \{timestamp \− 1\}K_{c,s}

3. Observations

   • The critical part of the system is the Kerberos server. If it is compromised, all user keys are lost.
   • There is a trade-off in selecting the optimal ticket lifetime: short ticket lifetimes make the system more secure but also less convenient for its users.
   • The Kerberos server is normally replicated on several sites. There is a single primary site and it is the only than can accept key change requests.
   • Kerberos does not protect clients and servers against denial of service attacks.
   • Newer versions of Kerberos eliminate the double encryption of tickets. It now replies with:

   K → WS: \{K_{c,tgs}\}K_{c}, \{T_{c,tgs}\}K_{tgs}.