Kerberos – Private Key System

Ahmad Ibrahim
History

- Cerberus, the hound of Hades, (Kerberos in Greek)
- Developed at MIT in the mid 1980s
- Available as open source or supported commercial software
- Combination of topics covered previously in class
What do we want to do?

- Want to be able to access all resources from anywhere on the network.
- Don't want to be entering password to authenticate for each access to a network service.
  - Time consuming
  - Insecure
Ingredients
Cryptology is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, authentication, and non-repudiation.
Review: Cryptology (cont)

- Private Key Mechanism
  - A single **secret key** (Y) is used for both encryption and decryption by the parties
  - **Symmetric Algorithm**

\[
\begin{align*}
M & \rightarrow E_Y(M) \rightarrow D_Y(C) \\
E_Y(M) & \rightarrow D_Y(C) \\
\end{align*}
\]
Review: Authentication

- **Authentication** is a mechanism that verifies a claim of identity

- Various systems provide means to reliably authenticate
  - Difficult to reproduce artifact; *digital signatures*
  - Shared secret; *symmetric key systems*
  - Electronic signature; *private key infrastructure*

- Needham-Schroeder with Denning-Sacco modification
Authorization is the process of giving individuals access to system objects based on their identity.
Putting it all together

- User's passwords are never sent across the network
- Secret keys are only passed across the network in encrypted form
- Client and server systems mutually authenticate
- It limits the duration of their users’ authentication
- Authentications are reusable and durable
Kerberos Terminology

- **Realm**: Kerberos “site”
- **Process**: client
- **Principle**: basic entity: user, service, host
  - **Associated with a key**
- **Instance**: optional additional identifier to make associated principles unique within a realm
- **Verifier**: application server
- **Authenticator**: encrypted data structure that confirms identity
- **Ticket**: a block of data sent to a service containing a user id, server id, and timestamp and time-to-live, encrypted with secret key
Kerberos Structure

**Requirements:**
- Each user has a private password known only to the user.
- A user’s secret key can be computed by a one-way function from the user’s password.
- The Kerberos server knows the secret key of each user and the TGS.
- Each server has a secret key known by itself and the TGS.
Key Distribution Center (KDC)

- Client (C)
- User (U)
- Kerberos Server (K)
- Ticket Granting Server (tgs)
- Application Server (S)

Authentication:
- User secret key database

Authorization:
- Server secret key database
Ticket

- Encrypted certificate issued by KDC
  - name of the principle (C)
  - name of server (S)
  - random session key (KC,S)
  - expiration time (lifetime)
  - timestamp

Ticket Structure:

\[
E_{K(s)} \{C, S, K_{C,S}, \text{timestamp, lifetime}\}
\]
Kerberos Protocol Simplified

- **Client to Authentication Server**
  - Authentication request

- **Authentication to Server**
  - Reply with ticket and session key

- **Client to Verifier**
  - User authenticates to verifier
  - Communicates with session key

- **Verifier to Client**
  - Optional, mutual authentication

1. `as_req`: `c, v, timesexp, n`
2. `as_rep`: `{Kc,v, v, timesexp, n, ...}Kc, {Tc,v}Kv`
3. `ap_req`: `{ts,cK, Ksubsession, ...}Kc,v {Tc,v}Kv`
4. `ap_rep`: `{ts}Kc,v (optional) Tc,v = Kc,v(c, timesexp ..."}
Protocol Overview

1. U: user id
2. $T_{u,tgs}$
3. $(T_{u,tgs}, S)$
4. $T_{C,S}$
5. $(T_{C,S}, \text{request})$
6. $T'$

Kerberos Server (K) → User (U) → Client (C) → Ticket Granting Server (tgs)
Kerberos: Phase 1

1. The user logs on to the client and the client asks for credentials for the user from Kerberos

   U --> C : U (user id)
   C --> K: (U, tgs)

2. Kerberos constructs a ticket for U and tgs and a credential for the user and returns them to the client

   \[ T_{u,tgs} = E_{K(tgs)} \{ U, tgs, K_{u,tgs}, ts, lt \} \]
   K --> C: E_{K(u)} \{ T_{u,tgs}, K_{u,tgs}, ts, lt \}

   The client obtains the user's password, P, and computes:

   \[ K'(u) = f(P) \]

   The user is authenticated to the client if and only if \( K'(u) \) decrypts the credential.
3. The client constructs an “authenticator” for user U and requests from TGS a ticket for server, S:

\[ A_U = E_{K(u,tgs)} \{ C, ts \} \]

\[ C \rightarrow TGS : (S, T_{u,tgs}, A_U) \]

4. The ticket granting server authenticates the request as coming from C and constructs a ticket with which C may use S:

\[ T_{c,s} = E_{K(s)} \{ C, S, K_{c,s}, ts, lt \} \]

\[ TGS \rightarrow C : E_{K(u,tgs)} \{ T_{c,s}, K_{c,s}, ts, lt \} \]
5. The client builds an “authenticator” and send it together with the ticket for the server to $S$:

$$A_c = E_{K(c,s)} \{ C, ts \}$$

$C \rightarrow S : (T_{c,s}, A_c)$

6. The server (optionally) authenticates itself to the client by replying:

$$S \rightarrow C: E_{K(c,s)} \{ts + 1 \}$$
Final Product

1. User logs on to workstation and requests service on host.

2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.

3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.

4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested service.

5. Workstation sends ticket and authenticator to server.

6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.
Limitations

- Every network service must be individually modified for use with Kerberos
- Doesn't work well in time sharing environment
- Requires a secure Kerberos Server
- Requires a continuously available Kerberos Server
- Stores all passwords encrypted with a single key
- Assumes workstations are secure
- May result in cascading loss of trust
- Scalability
Further Reading

- RFC 1510
- Kerberos web site
  http://web.mit.edu/kerberos/www
- O'Reilly Kerberos The Definitive Guide by Jason Garman
- Video on Kerberos from Oslo University College
Questions

?