Kerberos

Peter Sjödin
psj@kth.se

Based on material by Vitaly Shmatikov, Univ. of Texas, and by the previous course teachers
Kerberos

• Many-to-many authentication?
• Kerberos
Many-to-Many Authentication

- How can we authenticate users when they connect to machines on the network?
  - Separate logins for each machine
  - Distribute login information from one machine to every other machine
  - Insecure, management burden
Many-to-Many: Requirements

- **Scalable**
  - Must cope with large numbers of users and systems
- **Secure**
  - Must withstand both passive and active attacks
- **Transparent to users**
  - Should not require entering password all the time
  - Should not be noticed by, or hamper, users
Many-to-Many: Threats

• User impersonation
  - Malicious user with access to a workstation pretends to be another user from the same workstation
    • Can’t trust workstations to verify users’ identities

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

• Eavesdropping, tampering and replay
  - Malicious user eavesdrops on, tampers with or replays other users’ conversations to gain unauthorized access
Many-to-Many: Trusted Third Party

- Trusted authentication server:
  - Manages all passwords; grants access to systems
  - Key Distribution Center, essentially

- Single point of failure. Require strong physical security

User requests access, providing authentication

Use ticket to get access

Authentication server

Tickets

Servers & Workstations
What is a Ticket?

- Proof of authentication and authorization
  - Identity is confirmed
  - Access rights are confirmed
- Should have limited reusability
  - Should not contain server's password in plain text
  - Information in ticket is encrypted
    - Opaque to user
  - Server decrypts ticket when information is needed

**Ticket:** gives user access to server

User ——— Ticket: gives user access to server ——— Server
Ticket content

- User name
- Server name
- User's current network address

- Ticket lifetime
- Session key
- ...
- All of which are essential for functionality and security properties!
Authenticating

- Need to protect authentication
- Repeated authentication!
  - Once for every (non-local) service we need access to
  - Inconvenient and increases risk to permanent secret
    - User typing it or workstation remembering for user
  - Should only need to authenticate once!

Deja vu?
Solution: Two-step authentication

- Authenticate once and receive a special Ticket Granting Ticket (TGT)
  - System uses TGT to re-authenticate user
  - Like a login cookie on a web site.
Still Not Quite Done Yet

Don’t transmit password in cleartext

Ticket can be stolen

Authentication server

Server

Creats TGT using its own key

Server isn’t authenticated

Encrypted TGT

Encrypted TGT

Encrypted ticket

Encrypted ticket

Encrypted ticket

Encrypted ticket

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication

User authentication
Practical Authentication

- Many-to-many authentication?
- Kerberos
Kerberos

• Greek and Roman mythology
  - Cerberus, three-headed dog guarding the gates to Hades
• “Kerberos, originally developed for MIT's Project Athena, has grown to become the most widely deployed system for authentication and authorization in modern computer networks”
  • MIT Kerberos Consortium Web site
    – http://www.kerberos.org
Aims

• Single Sign-On
• User passwords never sent across network
• User passwords never stored on clients
• User passwords never stored in unencrypted form on authentication servers
• Mutual authentication
  - Application server can authenticate itself to client as well
• Support encrypted communication between client and application server
Summary of Kerberos

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 server.

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.
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 ticket granting service TGS
  - Known to KDC and TGS
- $K_V$ is **long-term** key of network service V
  - Known to V and TGS; separate key for each service
- $K_{C-TGS}$ is **short-term** key between C and TGS
  - Created by KDC, known to C and TGS
- $K_{C-V}$ is **short-term** key between C and V
  - Created by TGS, known to C and V
“Single Logon” Authentication

- Client only needs to obtain TGS ticket **once** (say, every morning)
  - Ticket is encrypted; client cannot forge it or tamper with it
Obtaining A Service Ticket

- Client uses TGT to obtain a service ticket and a **short-term key** for each network service
  - One encrypted, unforgeable ticket per service (printer, email, etc.)

```
System command, e.g. “lpr –Pprint”
```

```
K_{C,TGS}(ID_C, Addr_C, time_C)
Proves that client knows key $K_{C,TGS}$ contained in encrypted TGS ticket
```

```
K_{C,TGS}(K_{C-V}, ID_V, time_{TGS}, ticket_V)
Fresh key to be used between client and service
```

```
K_V(K_{C-V}, ID_C, Addr_C, ID_V, time_{TGS}, lifetime)
Client will use this unforgeable ticket to get access to service V
```

```
User
```

```
Client
Knows $K_{C,TGS}$ and ticket_{TGS}
```

```
Ticket Granting Service (TGS)
usually lives inside KDC
```

```
ID_V, ticket_{TGS}, auth_C
```

```
Knows key $K_V$ for each service
```
Obtaining Service

For each service request, client uses the short-term key for that service and the ticket he received from TGS.

User

Client

Knows $K_{C,V}$ and ticket$_V$

System command, e.g. “lpr –Pprint”

Server V

$K_{C,V} (ID_C, Addr_C, time_C)$
Proves that client knows key $K_{C,V}$ contained in encrypted ticket

ticket$_V$, auth$_C$

Authenticates server to client

Reasoning:

Server can produce this message only if server knows key $K_{C,V}$.

Server can learn key $K_{C,V}$ only if server can decrypt service ticket ticket$_V$.

Server can decrypt service ticket only if server knows correct key $K_v$.

If server knows correct key $K_v$, then server is the right server.
Important Ideas in Kerberos

- Use of short-term session keys
  - Minimize distribution and use of long-term secrets; use them only to derive short-term session keys
  - Separate short-term key for each user-server pair
- But multiple user-server sessions reuse the same key!
- Proofs of identity are based on authenticators
  - Client encrypts his identity, address and current time using a short-term session key
  - Also prevents replays (if clocks are globally synchronized)
  - Server learns this key separately (via encrypted ticket that client can’t decrypt) and verifies user’s identity
- Symmetric cryptography only
Kerberos in Large Networks

- One KDC isn’t enough for large networks (why?)
- Network is divided into realms
  - KDCs in different realms have different key databases
- We cannot assume a direct relationship between users in our realm and services in another
  - Organisations may have a relationship – DSV & KTH
- Want user's realm to manage access to foreign realm
Realms

• To access a service in another realm, users must...
  - Get ticket for home-realm TGS from home-realm KDC
  - Get ticket for remote-realm TGS from home-realm TGS
    • As if remote-realm TGS were just another network service
  - Get ticket for remote service from that realm’s TGS
  - Use remote-realm ticket to access service
  - \( N(N-1)/2 \) key exchanges for full \( N \)-realm interoperation
Principals and Credentials

- Principal: identifier, entry in authentication data base
  - Client: client@REALM
  - Service: service/hostname@REALM (hostname optional)
  - TGS: krbtgt/REALM@REALM

- Credential cache: where tickets and session keys are stored
- klist: Unix command to view credentials

```
peter@gopher$ klist
Credentials cache: FILE:/tmp/krb5cc_Zr6288
    Principal: peter@KTH.SE

    Issued     Expires                   Principal
    Nov 21 08:53:57 Nov 21 18:53:56       krbtgt/ICT.KTH.SE@KTH.SE
    Nov 21 08:53:56 Nov 21 18:53:56       afs@ICT.KTH.SE
peter@gopher$
```
Problematic Issues

- Password dictionary attacks on client master keys
- Replay of authenticators
  - 5-minute lifetimes long enough for replay
  - Timestamps assume global, secure synchronized clocks
  - Challenge-response would be better
- Same user-server key used for all sessions
- Homebrewed PCBC mode of encryption
  - Tries to combine integrity checking with encryption
- Extraneous double encryption of tickets
- No ticket delegation
  - Printer can’t fetch email from server on your behalf
Kerberos Version 5

- Better user-server authentication
  - Separate subkey for each user-server session instead of re-using the session key contained in the ticket
  - Authentication via subkeys, not timestamp increments
- Authentication forwarding
  - Servers can access other servers on user’s behalf
- Realm hierarchies for inter-realm authentication
- Richer ticket functionality
- Explicit integrity checking + standard CBC mode
- Multiple encryption schemes, not just DES
Delegation

- It is often desirable to allow one user, Alice, to delegate access to another user, Bob.
  - Alice cannot just give Bob her ticket, nor would it be prudent for her to give Bob her secret.
  - In Kerberos version 5, Alice can ask for ticket for Bob.
  - …or a TGT with a different network address.
  - Need to consider expiration time!
Expiration

- Kerberos v4 has an upper limit to expiration
  - Start time
  - 8 bit field counting 5 minute intervals = 21 hours
- Kerberos v5 has no limit
  - Start and end time
  - Timestamp of TGT used to create this ticket
  - Also allows renewable and postdated tickets
- Security issues with too liberal (unlimited?) ticket lifetimes!
Kerberos History

- Developed at MIT within Project Athena
  - 1980s
  - Still maintained by MIT
  - Windows, MacOS, Unix/Linux
- Subject to U.S. export restrictions
  - DES implementation
- Non-US version at KTH
  - KTH-KRB (Kerberos 4)
  - Heimdal (Kerberos 5)
    - Actively maintained and developed
    - http://www.h5l.org/
Practical Uses of Kerberos

• Email, FTP, network file systems and many other applications have been kerberized
  - Use of Kerberos is transparent for the end user
  - Transparency is important for usability!
• Microsoft’s Active Directory
• AFS (Andrew File System)
  - Distributed file system
  - Widely used at KTH
• Local authentication
  - login and su in Linux, OpenBSD
• Authentication for network protocols
  - Ssh, rlogin, rsh, telnet
• Secure windowing systems
  - xdm, kx
Other Authentication Server Systems

- Network Access Control
  - Remote Authentication Dial In User Service (RADIUS)
  - Diameter
  - TACACS (Terminal Access Controller Access-Control System)
  - TACACS+
- Web authentication ("single sign on")
  - Central Authentication Service (CAS)
  - Shibboleth
  - OpenID