Kerberos V4

November 16, 2000
Kerberos

- network authentication using Needham-Schroeder
- insecure network: listen, modify
- secret key

- login session: from login to logout

- Version 5: more complex, not just TCP/IP, greater functionality

- KDC + libraries (e.g., GSS API)
  - telnet
  - rlogin, rcp, rsh, ...
  - NFS

November 16, 2000
Tickets and Ticket-Granting Tickets

- users, resources: *principal* ➞ share masterkey with KDC
- KDC sends to $A$: $K_A\{K_{AB}\}$; ticket: $K_B\{K_{AB}, Alice\}$
- tickets expire in 21 hours
- thus: knowledge of $K_{AB}$ proves identity + use for encryption
- *credentials*: $K_{AB}$ and ticket
- password generates master key
- workstation asks for session key $S_A$ (time-limited)
- *ticket-granting ticket* (TGT): $K_{KDC}\{S_A, \ldots\}$
- workstation forgets master key, uses TGT
- KDC: authentication server (AS) + ticket-granting server (TGS)
Configuration

- \textit{KDC master key} encrypts KDC database, TGT
- DES-based
- principals need to remember pw (humans) or key (machines)
Logging In

- send username
- get credentials
- ask for password (minimum residency!)
- but: can do password-guessing by sending user name
- TGT ➠ state-less server (crashes, replication)
Communicating with Remote Node

rlogin Bob:

- authenticator = timestamp (Δ N-S)
- limit replay: allow skew of 5 min. → time synchronization
- construct ticket to Bob
Replicated KDCs

- KDC: single PoF (in addition to NFS...)
- replication with master copy
- performance scaling: service location protocol?
- exchange master database in clear, protected by secure hash
Realms

- can’t have single (replicated) KDC: need to limit trust
- limit compromise
- principal: name (service), instance (host, human role), realm
- each realm carries others as principals
- no chaining of realms: prevent rogue KDC impersonating everybody
- V4: DNS names
Key Version Numbers

- allow unsynchronized changes of master keys
- remember several versions of past keys
- replication → new passwords may fail
Privacy and Integrity

- encrypt and protect (e.g., CBC with residue ✤ two passes)
- plain-text cipher block chaining (PCBC)
  
  CBC: $c_{n+1} = E(m_{n+1} \oplus c_n)$
  
  PCBC: $c_{n+1} = E(m_{n+1} \oplus m_n \oplus c_n)$
  
- corrupt $c_i$: all data $> i$ will be changed
- put recognizable string at end
- but: can swap two adjacent $c_i$’s
Integrity

- DES CBC residue “too expensive”
- algorithm not documented (but not broken)
- hash over session key and message; transmit message, checksum
- may allow to get session key
Network Layer Addresses

- TGT, ticket contains Alice’s network layer address
- Bob checks connection
- Alice can’t hand off ticket to Ted
- can’t steal session key and use it from elsewhere
- prevent eavesdropping/replay within 5 min. window
- does not work with firewalls, mobile nodes
- does not support delegation
- addresses easily spoofable
Message Formats

**timestamp:** seconds since 1970-1-1; expires in 2038

**D bit:** direction to avoid reflection attack

**lifetime:** units of 5 minutes (21 hours)

**5 ms timestamp:** or sequence number

**session key:** 8 byte DES key

**B bit:** byteorder (little/big-endian)
### Kerberos vs. NT4.0

<table>
<thead>
<tr>
<th>Kerberos</th>
<th>NT 4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDC</td>
<td>PDC (primary domain controller)</td>
</tr>
<tr>
<td>replicated KDC</td>
<td>BDC (backup domain controller)</td>
</tr>
<tr>
<td>realm</td>
<td>domain (= 1 PDC, ≥ 1 BDC)</td>
</tr>
<tr>
<td>interrealm auth.</td>
<td>trust between domains</td>
</tr>
</tbody>
</table>