Overview

- Introduction
  - Security threads
  - Security policy and mechanism
- Secure channels
  - Authentication
  - Message Integrity and confidentiality

Security

Hongfei Yan
School of EECS, Peking University
5/19/2008

Security Threads

- Subject: Entity capable of issuing a request for a service as provided by objects
- Channel: The carrier of requests and replies for services offered to subjects
- Object: Entity providing services to subjects.

<table>
<thead>
<tr>
<th>Thread</th>
<th>Channel</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interception</td>
<td>Preventing message transfer</td>
<td>Denial of service</td>
</tr>
<tr>
<td>Inspection</td>
<td>Reading the content of transferred messages</td>
<td>Reading the data contained in an object</td>
</tr>
<tr>
<td>Modification</td>
<td>Changing message content</td>
<td>Changing an object's encapsulated data</td>
</tr>
<tr>
<td>Fabrication</td>
<td>Inserting messages</td>
<td>Spoofing an object</td>
</tr>
</tbody>
</table>

Security policies

- Policy: Prescribes how to use mechanisms to protect against attacks. Requires that a model of possible attacks is described (i.e., security architecture).
- Example: Globus security architecture
  - There are multiple administrative domains
  - Local operations subject to local security policies
  - Global operations require requester to be globally known
  - Interdomain operations require mutual authentication
  - Global authentication replaces local authentication
  - Users can delegate privileges to processes
  - Credentials can be shared between processes in the same domain
Design Issue: Focus of Control

- **Essence**: What is our focus when talking about protection: (a) data, (b) invalid operations, (c) unauthorized users

- **Note**: we generally need all three, but each requires different mechanisms

Design Issue: Layering of Mechanisms and TCB

- **Essence**: At which logical level are we going to implement security mechanisms?

- **Important**: Whether security mechanisms are actually used is related to the trust a user has in those mechanisms. No trust => implement your own mechanisms.

- **Trusted Computing Base**: What is the set of mechanisms needed to enforce a policy. The smaller, the better.

Cryptography

- **Symmetric system**: Use a single key to (1) encrypt the plaintext and (2) decrypt the ciphertext. Requires that sender and receiver share the secret key.

- **Asymmetric system**: Use different keys for encryption and decryption, of which one is private, and the other public.

- **Hashing system**: Only encrypt data and produce a fixed-length digest. There is no decryption, only comparison is possible.

Cryptographic Functions

- **Essence**: Make the encryption method $E$ public, but let the encryption as a whole be parameterized by means of a key $S$ (Same for decryption)

- **One-way function**: Given some output $m_{out}$ of $E_P$, it is (analytically or) computationally infeasible to find $m_{in}$: $E_S(m_{in}) = m_{out}$

- **Weak collision resistance**: Given a pair $<m, E_S(m)>$, it is computationally infeasible to find an $m* \neq m$ such that $E_S(m*) = E_S(m)$

- **Strong collision resistance**: Given $E_S$, it is computationally infeasible to find any two different inputs $m$ and $m*$ such that $E_S(m) = E_S(m*)$

- **Note**: Not all cryptographic functions have keys (such as hash functions)

Secure Channels

- **Authentication**

- **Message Integrity and confidentiality**

Secure Channels

- **Goal**: Set up a channel allowing for secure communication between two processes.

- They both know who is on the other side (authenticated).

- They both know that messages cannot be tampered with (integrity).

- They both know messages cannot leak away (confidentiality).
**Authentication versus Integrity**

- **Note:** Authentication and data integrity rely on each other: Consider an active attack by Trudy on the communication from Alice to Bob.

- **Authentication without integrity:** Alice’s message is authenticated, and intercepted by Trudy, who tampers with its content, but leaves the authentication part as is. Authentication has become meaningless.

- **Integrity without authentication:** Trudy intercepts a message from Alice, and then makes Bob believe that the content was really sent by Trudy. Integrity has become meaningless.

**Authentication: Secret Keys**

1. Alice sends ID to Bob
2. Bob sends challenge RB (i.e. a random number) to Alice
3. Alice encrypts RB with shared key KA,B. Now Bob knows he’s talking to Alice
4. Alice send challenge RA to Bob
5. Bob encrypts RA with KA,B. Now Alice knows she’s talking to Bob

**Note:** We can “improve” the protocol by combining steps 1 & 4, and 2 & 3. This costs only the correctness.

**Authentication: Public Key**

1. Alice sends a challenge RA to Bob, encrypted with Bob’s public key KB.
2. Bob decrypts the message, generates a secret key KA,B, proves he’s Bob (by sending RA back), and sends a challenge RB to Alice. Everything’s encrypted with Alice’s public key KA.
3. Alice proves she’s Alice by sending back the decrypted challenge, encrypted with generated secret key KA,B

**Note:** KA,B is also known as a session key.

**Authentication: KDC (1/3)**

- **Problem:** With N subjects, we need to manage N(N -1)/2 keys, each subject knowing N-1 keys.
- **Essence:** Use a trusted Key Distribution Center that generates keys when necessary.

- **Question:** How many keys do we need to manage?

**Authentication: KDC (2/3)**

- **Inconvenient:** We need to ensure that Bob knows about KA,B before Alice gets in touch.
- **Solution:** Let Alice do the work and pass her a ticket to set up a secure channel with Bob

**Note:** This is also known as the Needham-Schroeder authentication protocol, and is widely applied (in different forms).
Needham-Schroeder: Subtleties

- **Q1**: Why does the KDC put Bob into its reply message, and Alice into the ticket?
- **Q2**: The ticket sent back to Alice by the KDC is encrypted with Alice’s key. Is this necessary?
- **Security flaw**: Suppose Chuck finds out Alice’s key => he can use that key anytime to impersonate Alice, even if Alice changes her private key at the KDC.
- **Reasoning**: Once Chuck finds out Alice’s key, he can use it to decrypt a (possibly old) ticket for a session with Bob, and convince Bob to talk to him using the old session key.
- **Solution**: Have Alice get an encrypted number from Bob first, and put that number in the ticket provided by the KDC => we’re now ensuring that every session is known at the KDC.

Authentication: KDC (3/3)

- Protection against malicious reuse of a previously generated session key in the Needham-Schroeder protocol

References

- Chapter 8 of [Tanenbaum, 2002]
  - book1, [Coulouri, 2005]
  - book2, [Birman, 2005]
  - book3, [Tanenbaum, 2006]
  - book4, [Tanenbaum, 2002]
  - Book5, [Xining Li, 2006]