DISTRIBUTED SYSTEMS (COMP9243)

Lecture 11 (B): Security

**Slide 1**
A Introduction
A Cryptography
B Secure protocols and communication
B Authentication
B Authorisation

**Slide 2**
Secure Protocols
Protocol: rules governing communication
Security protocol: protocol that performs a security-related function (usually authentication)
Goal: Survive malicious attacks:
- Lies
- Modifying data
- Injecting data
- Malicious behaviour
Threat Assumptions:
- Can communication channel be intercepted?
- Can data stream be modified?
- Are participants malicious?

**Slide 3**
How to Build a Cryptographic Protocol
Use:
- encryption
- secure digest
- signatures
- random number generators
Protocol mechanisms:
- Challenge-Response
  - nonce – used to uniquely relate two messages together
  - What properties should a nonce have?
- Ticket – secured information to be passed to another party
  - Why is this useful?
- Session keys – for secure communication
  - Why is this useful?
Principles:
- A message must contain all relevant information
- Don’t allow parties to do things identically
- Don’t give away valuable information to strangers

**Slide 4**
A Simple Protocol
Authentication
- Alice knows it’s Bob; Bob knows it’s Alice
**How to Break a Protocol**

**Slide 5**

**Man-in-the-Middle:**
- Take on the role of Alice to Bob and Bob to Alice
- Alice → Eve: challenge
- Eve → Bob: challenge
- Eve ← Bob: response
- Alice ← Eve: response

**Slide 6**

**Reflection:**
- Use Alice to respond to Alice’s challenge
- Alice → Eve: challenge
- Alice ← Eve: challenge
- Alice → Eve: response
- Alice ← Eve: response

**Slide 7**

**Replay:**
- Re-use Bob’s old message to respond to Alice’s challenge
- Alice → Bob: challenge
- Alice ← Eve ← Bob: response
- Alice ← Eve: challenge
- Alice ← Eve: response

**Slide 8**

**Message Manipulation:**
- Change the message from Alice to Bob
- Alice sends: let’s meet at 3pm by the bridge
- Eve intercepts and changes
- Bob receives: let’s meet at 2pm by the oak

**Changed Environment/Assumptions:**
- Bob is no longer trustworthy
- Bob sells Alice’s secrets to the tabloid press!
A Simple Protocol: Revisited

Authentication

Slide 9

Optimising the Protocol

Slide 10

Is this different from Man-in-the-middle?

Slide 11

Key Distribution

A set of keys provides a secure channel for communication.

How does the secure channel get established in the first place?

- Use separate channel to establish keys
- Use key distribution protocols
- Protocols vary depending on whether symmetric or asymmetric encryption is used
- Often symmetric keys are communicated over a channel using an asymmetric cipher

Slide 12

Oops!

→ Vulnerable to reflection attack
**Distribution of Symmetric Keys (Needham-Schroeder)**

- Central key distribution centre $D$
- Each agent $A$ shares a (symmetric) key $K_A$ with $D$
- $A$ wants to communicate with $B$, asks $D$ for session key $K_{AB}$
- After key distribution protocol, both $A$ and $B$ know that they share a key provided by $D$

Properties of the symmetric key distribution protocol:
- Ticket and challenge implicitly authenticate $A$ and $B$
- Nonce and challenge protect against replay attacks
- $D$ is centralised resource (hierarchical scheme possible)
- Every agent must trust $D$
- $D$ maintains highly sensitive information (secret keys), compromising $D$ compromises all communication
- Large number of keys required (one per pair of agents), manufactured by $D$ on-the-fly
- $D$ must take care to make key sequence non-predictable

Any vulnerabilities?

**Secure Communication**

Properties of a Secure Channel:
- Authentication
- Message confidentiality
- Message integrity

**Example: SSL (and TLS)**

Secure Socket Layer:
- Application level protocol for secure channel
- Handshake protocol: establish and maintain session
- Authentication
- Record protocol: secure channel
- Confidentiality, Integrity
- Flexible: can choose ciphers to use
- Most widely used to secure HTTP (https: URLs)
- TLS (Transport Layer security): IETF standard based on SSL 3.0
- TLS 1.0: RFC 2246, TLS 1.2: RFC RFC 5246, TLS 1.3 proposed standard
SSL Handshake Protocol:

1. ClientHello
2. ServerHello
3. Certificate
4. Certificate Verify
5. Change Cipher Spec
6. Finished
7. Change Cipher Spec
8. Finished
9. Server Key Exchange
10. Certificate Request
11. Certificate
12. Client Key Exchange
13. ServerHelloDone

SSL Record Protocol:

1. Application Data
2. Record protocol units
3. Compressed units
4. Hash
5. Message Authentication Code
6. Encrypted
7. Transmit
8. TCP packet

Secure Group Communication

Two types:

- Confidential group communication:
  - All group members share the same secret key
  - Need to trust all members
  - Separate keys for each pair
  - Scalability problem
  - Public key cryptography
  - Everyone knows each others keys

Secure replicated servers:

- Secure Replicated Servers: protecting from malicious group members
- Collect responses from all servers and authenticate each
- Not transparent
- Secret sharing:
  - All group members know part of a secret.
  - Recipient combines answers from k members, decrypts with special decryption function D.
  - If successful: these k members are honest.
  - If not: try other combination of answers.
**AUTHENTICATION**

Verify the claimed identity of an entity (principal)

**Authentication Requires:**
- Representation of identity
  - Unix user id, email address, student number, bank account
- Some way to verify the identity
  - Password, reply to email, student card, PIN
- Different levels of authentication

**Credentials:**
- Speaks for a principal
- Example: certificate stating identity of a principal
- Combine credentials
- Role-based credentials

**Approaches to Authentication:**

**Password:** provide some secret information

**Shared secret key:** challenge and response encoded with shared secret key

**Key distribution centre:** keys stored at KDC, never sent over network

**Public key:** exchange session key encoded with public keys

**Hybrid:** use public keys to set up a secure channel and then authenticate

**KERBEROS**

Commercial authentication system developed at MIT

Based on Needham and Schroeder protocol

Integrates symmetric key encryption, distribution and authentication into commercial computer systems.

**Assumptions:**
- Secure central server
- Insecure network
  - Never transmit cleartext passwords
- Insecure workstations (shared between users)
  - Hold user passwords on workstations for very short periods only
  - Hold no system keys on workstations

**Kerberos Authentication:**

```
Key Distribution Centre
Authentication Service A
Ticket Granting Service B
```

```
Client C

Login Session Setup

Server Session Setup

Work

m1 = \{G \cdot T\}_{K_{C}} \cdot \{G, S\}_{K_{S}} \cdot request, n

m4 = \{K_{T}, n\}_{K_{C}} \cdot \{G, T\}_{K_{S}}

m2 = \{K_{C}, n\}_{K_{T}} \cdot \{G, T\}_{K_{S}}

m3 = \{K_{C}, n\}_{K_{T}} \cdot \{G, T\}_{K_{S}}

```

```
Server S
```

```
Client C
```

```
Server S
```
Central KDC contains
- Authentication service \( A \),
  knows all user logins and their passwords (secret keys)
  as well as identity and key of \( T \);
- Ticket granting service \( T \),
  knows all servers and their secret keys

Kerberos protocol has three phases:
1. login session setup (user authentication)
2. server session setup (establishing secure channel to server)
3. client-server RPC

Uses time-limited tickets

**Distribution of Public Keys**

Major weakness of Needham-Schroeder and Kerberos:
- Key distribution centre as a central authority
- Compromised keys can be used to decrypt past communication

Public Key Infrastructure (PKI):
- Public keys can be exposed without risk
- Distribution centre only establishes link between identities and public keys

Certificates and certification authorities:
- A certificate links an identity with a public key
- Distribution centres are called certificate servers or certificate directories

Checking of certificates is recursive:
- To establish trust in Alice’s certificate signed by \( C_2 \), Bob may need to obtain \( C_2 \)’s certificate
- Bob uses the public key of \( C_2 \) to validate Alice’s certificate
- \( C_2 \) is signed by \( C_1 \)
- This may lead to a chain of certificates
- Terminated by self-signed certificate of a root certification authority (who Bob trusts)

How to communicate certificates to clients?
- Secure channel between certificates server and client?
- Digital signatures establish the validity of certificates
- Formatted according to X509.1 standard or PGP format

Whose signature?
- Certification authorities sell certification as a service
- Alternatively, web of trust avoids any central authority
Are certificates valid forever?

- Certificates may have an expiry date to reduce risk of security breach
- After a certificate expires, a new one must be generated and signed
- Alternatively, certificates may be revoked
- Revocation is only effective if receiver regularly checks the certificate server

**Authorisation and Access Control**

Determine what actions an authenticated entity is authorised to perform

**Access Rights:**

- The rights required to access (perform an operation on) a given resource

**Access Control:**

- verify access rights
- grant access rights

**Non-distributed Protection:**

- Global mechanisms
- Global policies
- Examples:
  - Users
  - File permissions
  - Separate address spaces

**Distributed Protection:**

- Service specific
  - Web servers and .htaccess: authentication, access control
  - Application specific

**Access Control Matrix**

<table>
<thead>
<tr>
<th>Subjects</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O_1$</td>
<td>terminate</td>
<td>wait, signal, terminate</td>
<td>wait, signal, receive</td>
<td>control</td>
</tr>
<tr>
<td>$O_2$</td>
<td>wait, signal, send</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$O_3$</td>
<td>read</td>
<td></td>
<td></td>
<td>execute</td>
</tr>
<tr>
<td>$O_4$</td>
<td></td>
<td></td>
<td></td>
<td>write</td>
</tr>
</tbody>
</table>

- Access permissions of a given subject to a given object
- Specifies allowed operations

Ensuring that authorisation and access control are respected
Properties of the access matrix:

- Rows define subjects’ protection domains
- Columns define objects’ accessibility
- Dynamic data structure: frequently changes
  - permanent changes (e.g., chmod)
  - temporary changes (e.g., setuid flag)
- Matrix is very sparse with many repeated entries
  - usually not stored explicitly

Design considerations in a protection system:

- **Propagation** of rights:
  - Can someone act as an agent’s proxy?
- **Restriction** of rights:
  - Can an agent propagate a subset of their rights?
- **Amplification** of rights:
  - Can an unprivileged agent perform some privileged operations?
- **Revocation** of rights:
  - Can a right, once granted, be remove from an agent?
- **Determination of object accessibility**
  - Who has which rights on an object?
- **Determination of agent’s protection domain**
  - What is the set of objects an agent can access?

Access control lists (ACLs):

<table>
<thead>
<tr>
<th>Object</th>
<th>Subjects</th>
</tr>
</thead>
<tbody>
<tr>
<td>/etc/passwd</td>
<td>S₁</td>
</tr>
<tr>
<td></td>
<td>read</td>
</tr>
</tbody>
</table>

- Column-wise representation of the access matrix
- Each object associated with a list of (subject, rights) pairs
  - requires explicit authentication
- Usually supports concept of group rights (domain classes)
  - (granted to each agent belonging to the group)
- Often simplified to a simple fixed-size list (e.g., UNIX user-group-others or VMS system-owner-group-world)
- Can have negative rights as well (e.g., to simplify exclusion from groups)

Properties of ACLs:

- **Propagation**: meta-right to change ACL (e.g., owner can chmod)
- **Restriction**: meta-right to change ACL
- **Amplification**: (e.g., setuid)
- **Revocation**: remove from ACL
- **Object accessibility**: explicit in ACL
- **Protection domain**: hard (if not impossible)
Slide 37

Capabilities:
- An element of access matrix
- Capabilities list (C-list) associated with each subject, which defines a protection domain
- Each capability can confer a single or a set of rights
- Capabilities can confer negative rights
- Capabilities must be protected against forgery and theft
- Capability used as an object name:
  - evidence of access permission
  - independent of authentication
  - don’t need to trust intermediary

Slide 38

Properties of capabilities:
- Propagation: copy capability (but need to be careful about confinement)
- Restriction: may be supported by derived capabilities
- Amplification: may have amplification capabilities
- Revocation: difficult, requires invalidation
- Object accessibility: hard (if not impossible)
- Protection domain: explicit in C-list

Slide 39

Three basic approaches to making caps tamper-proof:
- Tagged capabilities:
  - protected by hardware (tag bit)
  - controlled by OS (only kernel can turn on tag bit)
  - used in most historical capability systems (Plessey 250, CAP, Hydra, System/38)
- Partitioned (segregated) capabilities:
  - protected by OS: Capabilities kept in kernel space
  - used in Mach, Grasshopper, EROS, seL4
- Sparse capabilities:
  - protected by sparseness (obscurity)
  - used in Monash Password Capability System, Amoeba, Mungi

Slide 40

Signature capabilities:
- Encrypted capability
- Object ID Access rights
- Signature

✓ tamper proof via encryption with secret kernel key
✓ can be freely passed around
✗ need to encrypt on each validation
Password capabilities:

- Invented for Monash U's Password Capability System
- "Random" bitstring is password, not derived from other parts of capability.
- Validation requires checking against global object table.

Firewalls

Properties:

- When communicating with untrusted clients/servers
- Disconnects part of system from outside world
- Incoming communication inspected and filtered

Two types:

- Packet-filtering gateway
- Application-level gateway

Three Myths of Firewalls:

① We've got the place surrounded
② Nobody here but us chickens
③ Sticks and Stones may break my bones, but words will never hurt me

How to Break Security?

Encryption:

- find weaknesses in algorithms
- find weaknesses in implementations
- attack underlying intractable problem
- brute force

Protocols:

- find weakness in protocol design (try MitM, reflection attacks)
- find vulnerability in implementation

Authentication:

- find keys or passwords
- social engineering

Authorisation and Access Control:

- find problems with Access Control Matrix
- find and exploit bugs to escalate privileges

Reading List

HOMEWORK

Look up how protocols have been broken in the past. Find examples where:

- the protocol was broken
- the cryptography was broken
- the implementation was broken