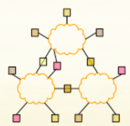
15-446 Distributed Systems Spring 2009



L-14 Security

1

Important Lessons - Security

- Internet design and growth → security challenges
- Symmetric (pre-shared key, fast) and asymmetric (key pairs, slow) primitives provide:
  - Confidentiality
  - Integrity
  - Authentication
- "Hybrid Encryption" leverages strengths of both.
- Great complexity exists in securely acquiring keys.
- Crypto is hard to get right, so use tools from others, don't design your own (e.g. TLS).

2

Today's Lecture

- Access Control
- Identity and Trust
- Project 2

3

Security Threats, Policies, and Mechanisms

- Security implies dependability, confidentiality, and integrity.
- Types of security threats to consider:
  - **Interception** – an unauthorized party gains access to data or service
  - **Interruption** – situation where data or service becomes unavailable
  - **Modification** – unauthorized changing of data or tampering with a service so that it no longer adheres to its spec.
  - **Fabrication** – situation where data or activity generated that normally would not exist.

4

Denial of Service attacks

- Bandwidth depletion
  - Typically accomplished by sending many message to a single machine, making it difficult for the normal messages to be processed.
- Resource depletion
  - Attempting to tie up resources that are needed by normal processes.
- One thing that makes the problem particularly difficult is that attackers use innocent users by secretly installing code on their machine.
- Detecting/stopping DoS attacks typically involves monitoring of message traffic.

5

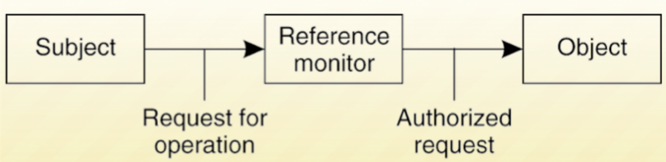
Security Threats, Policies, and Mechanisms

- Security policy – describes which actions the entities in a system are allowed to take (and which are prohibited)
- Security mechanism – way to enforce policy
  1. Encryption – data confidentiality, data integrity
  2. Authentication – verify the claims of a user, client, server or host
  3. Authorization – see if an authenticated client is allowed to perform the requested action
  4. Auditing – logging access

6

Access Control (1)

- Once secure communication between a client and server has been established, we now have to worry about access control – when the client issues a request, how do we know that the client has **authorization**?



7

The Access Control Matrix (ACM)

A model of protection systems

- Describes who (subject) can do what (rights) to what/whom (object/subject)
- Example
  - An instructor can assign and grade homework and exams
  - A TA can grade homework
  - A Student can evaluate the instructor and TA

8

An Access Control Matrix

- Allowed Operations (Rights): r,x,w,o

	File*1*	File2	File*3*
Ann	*rx*	*r*	rwx
Bob	*rwxo*	r	--
Charlie	*rx*	rwo	w

ACMs and ACLs; Capabilities

- Real systems have to be fast and not use excessive space

What's Wrong with an ACM?

- If we have 1k 'users' and 100k 'files' and a user should only read/write his or her own files
  - The ACM will have 101k columns and 1k rows
  - Most of the 101M elements are either empty or identical
- Good for theoretical study but bad for implementation
  - Remove the empty elements?

Two ways to cut a table (ACM)

- Order by columns (ACL) or rows (Capability Lists)?

	File*1*	File2	File*3*
Ann	*rx*	*r*	rwx
Bob	*rwxo*	r	--
Charlie	*rx*	rwo	w

ACLs

Capability

Access Control Lists

- An ACL stores (non-empty elements of) each column with its object
- Columns of access control matrix

	File1	File2	File3
Andy	*rx*	*r*	rwx
Betty	*rwxo*	r	--
Charlie	*rx*	rwo	w

- ACLs:
- file1: { (Andy, rx) (Betty, rwxo) (Charlie, rx) }
- file2: { (Andy, r) (Betty, r) (Charlie, rwo) }
- file3: { (Andy, rwo) (Charlie, w) }

13

Capability Lists

- Rows of access control matrix

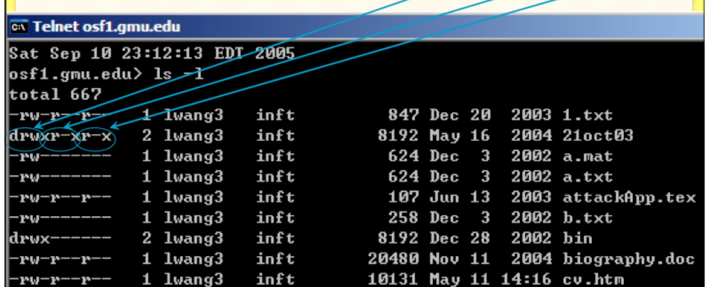
	File1	File2	File3
Andy	*rx*	*r*	rwx
Betty	*rwxo*	r	--
Charlie	*rx*	rwo	w

- C-Lists:
  - Andy: { (file1, rx) (file2, r) (file3, rwo) }
  - Betty: { (file1, rwxo) (file2, r) }
  - Charlie: { (file1, rx) (file2, rwo) (file3, w) }

14

ACL:Default Permission and Abbreviation

- Example: UNIX ➔
- Three classes of users: owner, group, all others



15

ACL Abbreviations

- Augment abbreviated lists with ACLs
  - Intent is to shorten ACL without losing the granularity
- Example ➔ IBM AIX
  - ACL overrides base permission
  - Denial takes precedence

16

Permissions in IBM AIX

```
attributes:
base (traditional UNIX) permissions
  owner(bishop):      rw-
  group(sys):         r--
  others:       ---
extended permissions enabled
  specify     rw-     u:holly              [override]
  permit      -w-     u:heidi, g=sys       [Add]
  permit      rw-     u:matt
  deny        -w-     u:holly, g=faculty   [Remove right]
```


Semantics of Capability

- Like a bus ticket
  - Mere possession indicates rights that subject has over object
  - Object identified by capability (as part of the token)
    - Name may be a reference, location, or something else
  - The key challenge is to prevent process/user from altering capabilities
    - Otherwise a subject can augment its capabilities at will

Implementation of Capability

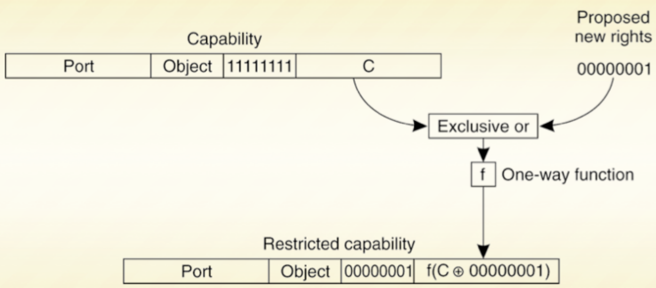
- Tagged architecture
  - Bits protect individual words
- Paging/segmentation protections
  - Like tags, but put capabilities in a read-only segment or page
- Cryptography
  - Associate with each capability a cryptographic checksum enciphered using a key known to OS
  - When process presents capability, OS validates checksum

Capabilities and Attribute Certificates (1)

48 bits	24 bits	8 bits	48 bits
Server port	Object	Rights	Check

- Owner capability in Amoeba.

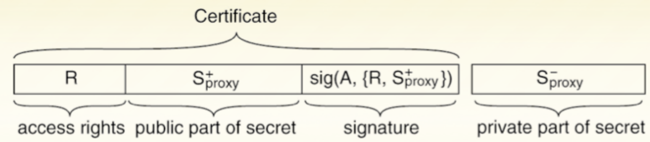
Capabilities and Attribute Certificates (2)



- Generation of a restricted capability from an owner capability.

21

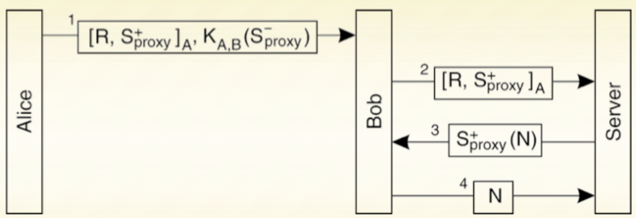
Delegation (1)



- The general structure of a proxy as used for delegation.

22

Delegation (2)



- Using a proxy to delegate and prove ownership of access rights.

23

ACLs vs. Capabilities

- They are equivalent:
  1. Given a subject, what objects can it access, and how?
  2. Given an object, what subjects can access it, and how?
  - ACLs answer second easily; C-Lists, answer the first easily.
- The second question in the past was most used; thus ACL-based systems are more common
- But today some operations need to answer the first question

24

Today's Lecture

- Access Control
- Identity and Trust
- Project 2

Fault Tolerance

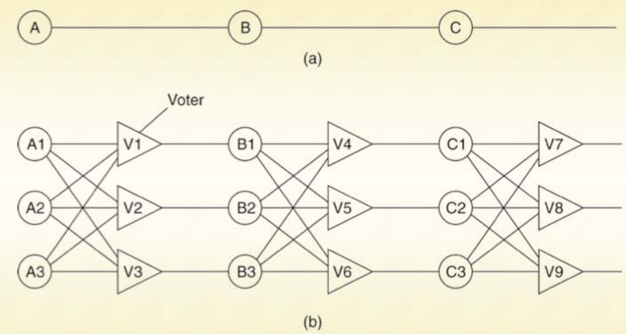
- Being fault tolerant is strongly related to what are called dependable systems. Dependability implies the following:
- **Availability**: probability the system operates correctly at any given moment
- **Reliability**: ability to run correctly for a long interval of time
- **Safety**: failure to operate correctly does not lead to catastrophic failures
- **Maintainability**: ability to "easily" repair a failed system

Failure Models

Type of failure	Description
Crash failure	A server halts, but is working correctly until it halts
Omission failure	A server fails to respond to incoming requests
Receive omission	A server fails to receive incoming messages
Send omission	A server fails to send messages
Timing failure	A server's response lies outside the specified time interval
Response failure	A server's response is incorrect
Value failure	The value of the response is wrong
State transition failure	The server deviates from the correct flow of control
Arbitrary failure	A server may produce arbitrary responses at arbitrary times

- A system is said to fail if it cannot meet its promises. An error on the part of a system's state may lead to a failure. The cause of an error is called a fault.

Failure Masking by Redundancy



- Triple modular redundancy. For each voter, if two or three of the inputs are the same, the output is equal to the input. If all three inputs are different, the output is undefined.

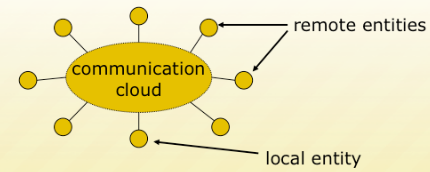
Sybil Attack undermines assumed mapping between identity to entity and hence number of faulty entities

- A Sybil attack is the forging of multiple identities for malicious intent -- having a set of faulty entities represented through a larger set of identities.
- The purpose of such an attack is to compromise a disproportionate share of a system.
- Result is overthrowing of any assumption of designed reliably based on a limited proportion of faulty entities.

29

Model

- Model in Douceur(2002):
- Set E of entities e; two disjoint subsets C (c is correct) and F (f is faulty).
- Broadcast communication cloud, pipe connecting each entity to the cloud.
- Entities communicate by broadcast messages, all messages received within bounded time, not necessarily in order.
- Assume local entity l is correct.



30

Goal: accept all legitimate identities, but no counterfeits

- Model (continued)
- Identity i is abstract representation of entity e which persists across multiple messages.
- 3 sources of info for which a local entity can accept identity i of remote e :
  - Trusted agency
  - Itself
  - Other entities
- Two ways to validate entities not received from trusted agency:
  - Direct validation
  - Indirect validation; accept identities vouched for by already accepted identities

31

Sybil Attack

- Result: for direct or indirect validation, a set of faulty entities can counterfeit an unbounded number of identities
- Method: for direct and and indirect validation (not using trusted agency), utilize computational tasks to validate distinctness;
  - basically, validate distinctness of two entities by getting them to perform some task (computational puzzle) that a single entity could not.
  - can not assume homogeneous resources, only minimum; faulty entity could have more than minimum
  - practical impossibility of having challenges issued simultaneously.

32

Sybil Attack

- Douceur's Conclusion: A centralized authority is required to realize a reliable distributed system
- Validation which does not use a trust agency can't provable meet the identity goal;
  - Identification based on local-only information not practical (remember days when DNS was a file on your system?)
  - PGP-style web of (certification) trust not adequate; is indirect-validation.

33

Today's Lecture

- Access Control
- Identity and Trust
- Project 2

34

Project 2

- 3 person groups
- Build a real, useable distributed application
  - Should have some distributed systems component
    - E.g. routing, replication, consistency, …
  - Should make use of Android
- Project proposals due 3/19
  - 1pg writeup
  - Expected timeline for progress
- Project meetings
  - 1st – 3/24 → refine/approve project idea
  - 2nd – 4/6 → progress update
  - 3rd – 4/23 → progress update
  - 4th – 4/30 → project demos
- Final report due 5/3

35

Collaboration Applications

- Shared event recording
  - Class note taking
  - Photos
  - Audio
- Collaborative editing
- Reviewing/reputation designs
  - Photos, files, songs
  - 802.11 access points and other services

36

Games

- Strategy games
  - Slow-paced, move exchange on meetings
- Position based
- Standard FPS
  - Managing BW more aggressively than wired games

37

Application Distribution

- Get updates and find new applications from your friends
- P2P filesharing
- Running code in an distributed experiment
- P2P routing of messages through phones

38

Social Networks

- Locating friends via local multi-hop
  - Finding friends of friends without revealing relationships
- Sybil attack prevention
- Shared “bulletin boards”
  - Leaving notes for your friends
- Trajectory reporting

39

Sensor Networks Applications

- How many people are near you
  - Can we measure density in different parts of campus?
  - Privacy preserving
  - Synopsis diffusion
- Image or audio sensing
  - Distributing sensing requests
- User entered data
  - Item prices
  - Weather

40