Security Part 4
Serious Security

- National defense
- Proprietary information
- Personal privacy
Mandatory vs. Discretionary Access Control

• Discretionary
  – ACLs, capabilities, etc.
    - access is at the discretion of the owner

• Mandatory
  – government/corporate security, etc.
    - access is governed by strict policies
Mandatory Access Control (1)

- Top Secret
- Secret
- Confidential
- Unclassified
Mandatory Access Control (2)

• Privacy/confidentiality policies
  – compartmentalization

- student records
  registrar

- faculty salaries
  dean of the faculty

- medical records
  University-affiliated hospitals
Mandatory Access Control (3)

• Local computer policy
  – web-server
    - may access only designated web-server data
  – administrators
    - may execute only administrative programs
    - (may not execute code supplied by ordinary users)
Bell-LaPadula Model

1) Simple security property
   - no subject may read from an object whose classification is higher than the subject’s clearance

2) *-property
   - no subject may write to an object whose classification is lower than the subject’s clearance
Information Black Hole

?  
Top Secret  
Secret  
Confidential  
Unclassified  

Attack!

Not cleared for top-secret orders
Managing Confidentiality

• Black-hole avoidance
  – trusted vs. untrusted subjects
  – trusted subjects may write down
Espionage

agent X learns of invasion plans

communication not possible

agent Y can send email to spymaster (but doesn’t know what to send)
Covert Channels

agent X runs resource-intensive program

sneaky communication possible

agent Y monitors load sends email to spymaster
Defense

- Identify all covert channels
  - (good luck …)
- Eliminate them
  - find a suitable scheduler
    - eliminates just one channel
Multi-Level Directories (1)

root directory

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>tmp</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>plan x</th>
<th>invade Canada</th>
<th>plan z</th>
</tr>
</thead>
<tbody>
<tr>
<td>confidential</td>
<td>top secret</td>
<td>confidential</td>
</tr>
</tbody>
</table>
```
Multi-Level Directories (2)
Orange Book

• Evaluation criteria for secure systems
  – D: minimal protection
  – C: discretionary protection
    - C1: discretionary security protection
    - C2: controlled access protection
  – B: mandatory protection
    - B1: labeled security protection
    - B2: structured protection
    - B3: security domains
  – A: verified protection
    - A1: verified design
Integrity

Top Secret → Secret → Confidential → Unclassified

Interstate highway Database

?
Biba Model

• Integrity is what’s important
  – no-write-up
  – no-read-down
Windows and MAC

• Concerns
  – viruses
  – spyware
  – etc.

• Installation is an integrity concern

• Solution
  – adapt Biba model
Windows Integrity Control

- No-write-up
- All subjects and objects assigned a level
  - untrusted
  - low integrity
    - Internet Explorer
  - medium integrity
    - default
  - high integrity
  - system integrity
- Object owners may lower integrity levels
- May set *no-read-up* on an object
Industrial-Strength Security

• Target:
  – embezzlers
Clark-Wilson Model

- Integrity and confidentiality aren’t enough
  - there must be control over how data is produced and modified
    - well formed transactions

- Separation of duty
  - steps of transaction must involve multiple people

Cash account
Withdrawals here

Accounts-payable account
Must be matched by entries here
Implementing MAC

- Label subjects and objects
- Security policy makes decisions based on labels and context

- registrar person
- d.o.f. person
- CS person
- web-server process

- student record
- salary record
- password file
- public database
• Security-Enhanced Linux
  – MAC-based security
  – labels on all subjects and objects
  – policy-specification language
SELinux Examples (1)

- Publicly readable files assigned type `public_t`
- Subjects of normal users run in domain `user_t`
- `/etc/passwd`: viewable, but not writable, by all
- `/etc/shadow`: protected
- SELinux rules
  
  ```
  allow user_t public_t : file read
  - normal users may read public files
  allow passwd_t passwd_data_t : file {read write}
  - `/etc/shadow` is of type `passwd_data_t`
  - subjects in `passwd_t` domain may read/write `/etc/shadow`
  ```
SELinux Examples (2)

• How does a program get into the `passwd_t` domain?
  – assume passwd program is of type `passwd_exec_t`

allow passwd_t passwd_exec_t : file entrypoint
allow user_t passwd_exec_t : file execute
allow user_t passwd_t : process transition
type_transition user_t passwd_exec_t : process
  passwd_t
SELinux Examples (3)

• Accounting example
  – one person requests a purchase order; another approves it
  – files containing accounting data are of type account_data_t
  – subjects accessing data are in two domains
    - account_req_t
    - account_approv_t

allow account_req_t account_data_t : file {read write}
allow account_approv_t account_data_t : file {read write}
SELinux Examples (4)

- Must specify which programs must be used to manipulate accounting data
  - requestPO
    - used to request a purchase order
    - type `account_req_exec_t`
  - approvePO
    - used to approve purchase order
    - type `account_approv_exec_t`
SELinux Examples (5)

• Who may run these programs?

  allow user_t account_req_t : process transition
  allow user_t account_approv_t : process transition
    - normal users may, but ...
SELinux Examples (6)

• Restrict usage to those users in appropriate roles
  
  role POrequester_r types account_req_t
  role POapprover_r types account_approv_t

  user mary roles {user_r POrequester_r}
  user robert roles {user_r POapprover_r}
  allow user_r {POrequester_r POapprover_r}
  role_transition user_r account_req_exec_t
    POrequester_r
  role_transition user_r account_approv_exec_t
    POapprover_r
SELinux Examples (7)

- Finally ...

```plaintext
allow user_t {account_req_exec_t
    account_approv_exec_t} : file execute
  - allow mary and robert to execute programs they need to run
```
Off-the-Shelf SELinux

• Strict policy
  – normal users in user_r role
  – users allowed to be administrators in staff_r role
    - but may run admin commands only when in sysadm_r role
  – policy requires > 20,000 rules
  – tough to live with

• Targeted policy
  – targets only “network-facing” applications
  – everything else in unconfined_t domain
  – ~11,000 rules
Confused-Deputy Problem

• The system has a pay-per-use compiler
  – keeps billing records in file /u/sys/comp/usage
  – puts output in file you provide
    - /u/you/comp.out
• The concept of a pay-per-use compiler annoys you
  – you send it a program to compile
  – you tell it to put your output in /u/sys/comp/usage
  – it does
    - it’s confused
    - you win
Unix and Windows to the Rescue

• Unix
  – compiler is “su-to-compiler-owner”
• Windows
  – client sends impersonation token to compiler
• Result
  – malicious deputy problem
• Could be solved by passing file descriptors
  – not done
  – should be ...
Authority

• Pure ACL-based systems
  – authority depends on subject’s user and group identities

• Pure capability-based systems
  – authority depends upon capabilities possessed by subject
ACLs vs. C-Lists

Mary's Process
Mary: rw
Robert: r
ACL

File X

Robert's Process
Mary: r
Robert: rw
ACL

File Y

Mary's Process
Mary: rw
Robert: r
ACL

C-List

C-List
More General View

- Subjects and resources are *objects* (in the OO sense)
Copying Capabilities (1)

Object A

write cap
read

Object B

Object C
Copying Capabilities (2)

Object A → write cap
   ↘
   read
   ↗
   read
   ↘
Object B

Object C
“Directories”

Object A
- read cap

Object B
- read cap

Directory
- read
- write
- append

Object X

Object Y

Object Z
Least Privilege (1)

- Login Process
  - read cap
  - write cap
- Directory
  - read
  - write
  - read
- Public Data
- System File
- Credit Card Info
- Suspect Code
Least Privilege (2)

- Login Process
  - read cap
  - write cap

- Suspect Code
  - read

- Directory
  - read
  - write

- Public Data
  - read

- System File

- Credit Card Info
Issues

• Files aren’t referenced by names. How do your processes get capabilities in the first place?
  – your “account” is your login process
    - created with all capabilities it needs
    - persistent: survives log-offs and crashes
Issues

• Can MAC be implemented on a pure capability system?
  – proven impossible twice
    - capabilities can be transferred to anyone
      • wrong: doesn’t account for write-capability and read-capability capabilities
    - capabilities can’t be retracted once granted
      • wrong:
Do Pure Capability Systems Exist?

• Yes!
  – long history
    - Cambridge CAP System
    - Plessey 250
    - IBM System/38 and AS/400
    - Intel iAPX 432
    - KeyKOS
    - EROS
A Real Capability System

• KeyKOS
  – commercial system
  – capability-based microkernel
  – used to implement Unix
    - (sort of defeating the purpose of a capability system ...)
  – used to implement KeySafe
    - designed to satisfy “high B-level” orange-book requirements
    - probably would have worked
    - company folded before project finished
KeySafe