L15: Ring-based Access Control

Hui Chen, Ph.D.
Dept. of Engineering & Computer Science
Virginia State University
Petersburg, VA 23806

Acknowledgement

- Many slides are from or are revised from the slides of the author of the textbook
  - Matt Bishop, Introduction to Computer Security, Addison-Wesley Professional, October, 2004, ISBN-13: 978-0-321-24774-5. Introduction to Computer Security @ VSU's Safari Book Online subscription
  - http://nob.cs.ucdavis.edu/book/book-intro/slides/

Outline

- Locks and keys
- Rings-based access control
- Propagated access control lists

Locks and Keys

- Associate information (*lock*) with object, information (*key*) with subject
  - Latter controls what the subject can access and how
  - Subject presents key; if it corresponds to any of the locks on the object, access granted
- Can be dynamic
  - ACLs, Capability-Lists are static and must be manually changed
  - Locks and keys can change based on system constraints, other factors (not necessarily manual)

Cryptographic Implementation

- Enciphering key is lock; deciphering key is key
  - Encipher object o; store $E_k(o)$
  - Use subject's key k' to compute $D_{k'}(E_k(o))$
  - Any of n can access o: store
$$o' = (E_1(o), \ldots, E_n(o))$$
  - Requires consent of all n to access o: store
$$o' = (E_1(E_2(\ldots(E_n(o))\ldots))$$

Example: IBM 370

- IBM 370: process gets access key; pages get storage key and fetch bit
  - Fetch bit clear: read access only
  - Fetch bit set, access key 0: process can write to (any) page
  - Fetch bit set, access key matches storage key: process can write to page
  - Fetch bit set, access key non-zero and does not match storage key: no access allowed

Example: Cisco Router

- Dynamic access control lists

```
access-list 100 permit tcp any host 10.1.1.1 eq telnet
access-list 100 dynamic test timeout 180 permit ip any host \
  10.1.2.3 time-range my-time
time-range my-time
  periodic weekdays 9:00 to 17:00
line vty 0 2
  login local
  autocommand access-enable host timeout 10
```

- Limits external access to 10.1.2.3 to 9AM–5PM
  - Adds temporary entry for connecting host once user supplies name, password to router
  - Connections good for 180 minutes
    - Drops access control entry after that

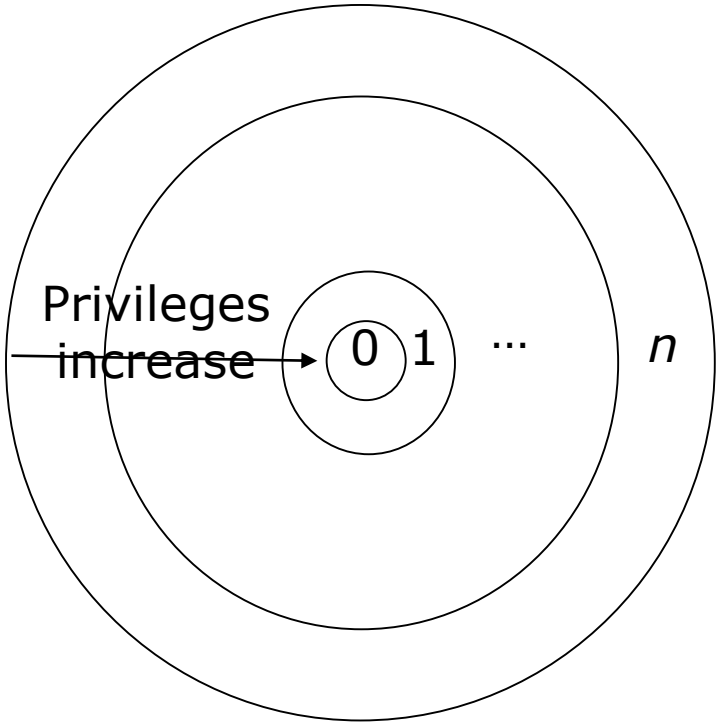
Type Checking

- Lock is type, key is operation
  - Example: UNIX system call *write* can't work on directory object but does work on file
  - Example: split I&D space of PDP-11
  - Example: countering buffer overflow attacks on the stack by putting stack on non-executable pages/segments
    - Then code uploaded to buffer won't execute
    - Does not stop other forms of this attack, though …

More Examples

- LOCK system:
  - Compiler produces “data”
  - Trusted process must change this type to “executable” becore program can be executed
- Sidewinder firewall
  - Subjects assigned domain, objects assigned type
    - Example: ingress packets get one type, egress packets another
  - All actions controlled by type, so ingress packets cannot masquerade as egress packets (and vice versa)

Ring-Based Access Control



- Process (segment) accesses another segment
  - Read
  - Execute
- *Gate* is an entry point for calling segment
- Rights:
  - *r* read; *w* write; *a* append; *e* execute

Reading/Writing/Appending

- Procedure executing in ring r
- Data segment with *access bracket* (a_1, a_2)
- Mandatory access rule
  - $r \leq a_1$ allow access
  - $a_1 < r \leq a_2$ allow r access; not w, a access
  - $a_2 < r$ deny all access

Executing

- Procedure executing in ring r
- Call procedure in segment with *access bracket* (a_1, a_2) and *call bracket* (a_2, a_3)
  - Often written (a_1, a_2, a_3)
- Mandatory access rule
  - $r < a_1$ allow access; ring-crossing fault
  - $a_1 \leq r \leq a_2$ allow access; no ring-crossing fault
  - $a_2 < r \leq a_3$ allow access if through valid gate
  - $a_3 < r$ deny all access

Versions

- Multics
  - 8 rings (from 0 to 7)
- Digital Equipment's VAX
  - 4 levels of privilege: user, monitor, executive, kernel
- Older systems
  - 2 levels of privilege: user, supervisor

Propagated Access Control List

- PACLs
  - Implements ORGON
- Creator kept with PACL, copies
  - Only owner can change PACL
  - Subject reads object: object's PACL associated with subject
  - Subject writes object: subject's PACL associated with object
- Notation: $PACL_s$ means *s* created object; PACL(*e*) is PACL associated with entity *e*

Multiple Creators

- Betty reads Ann's file *dates*

  $$\text{PACL(Betty)} = \text{PACL}_{\text{Betty}} \cap \text{PACL}(\textit{dates}) = \text{PACL}_{\text{Betty}} \cap \text{PACL}_{\text{Ann}}$$

- Betty creates file *dc*

  $$\text{PACL}(\textit{dc}) = \text{PACL}_{\text{Betty}} \cap \text{PACL}_{\text{Ann}}$$

- $\text{PACL}_{\text{Betty}}$ allows Char to access objects, but PACL_{Ann} does not; both allow June to access objects
  - June can read *dc*
  - Char cannot read *dc*

Summary

- Access control mechanisms provide controls for users accessing files
- Many different forms
  - ACLs, capabilities, locks and keys
    - Type checking too
  - Ring-based mechanisms (Mandatory)
  - PACLs (ORCON)