CSci 5271
Introduction to Computer Security
OS authentication and access control
(combined lecture)

Stephen McCamant

University of Minnesota, Computer Science & Engineering

Outline

OS protection and isolation (cont'd)

OS security: authentication

Basics of access control

Announcements, Ex. 1 debrief

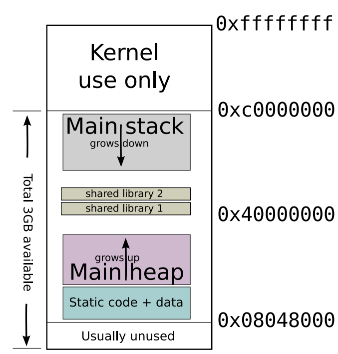
Multilevel and mandatory access control

Capability-based access control

Hardware basis: memory protection

- Historic: segments
- Modern: paging and page protection
  - Memory divided into pages (e.g. 4k)
  - Every process has own virtual to physical page table
  - Pages also have R/W/X permissions

Linux 32-bit example



Hardware basis: supervisor bit

- Supervisor (kernel) mode: all instructions available
- User mode: no hardware or VM control instructions
- Only way to switch to kernel mode is specified entry point
- Also generalizes to multiple "rings"

Outline

OS protection and isolation (cont'd)

OS security: authentication

Basics of access control

Announcements, Ex. 1 debrief

Multilevel and mandatory access control

Capability-based access control

Authentication factors

- Something you know (password, PIN)
- Something you have (e.g., smart card)
- Something you are (biometrics)
- CAPTCHAs, time and location, ...
- Multi-factor authentication

Passwords: love to hate

- Many problems for users, sysadmins, researchers
- But familiar and near-zero cost of entry
- User-chosen passwords proliferate for low-stakes web site authentication

Password entropy

- Model password choice as probabilistic process
- If uniform, $\log_2 |S|$
- Controls difficulty of guessing attacks
- Hard to estimate for user-chosen passwords
  - Length is an imperfect proxy

Password hashing

- Idea: don't store password or equivalent information
- Password 'encryption' is a long-standing misnomer
  - E.g., Unix `crypt(3)`
- Presumably hard-to-invert function h
- Store only $h(p)$

Dictionary attacks

- Online: send guesses to server
- Offline: attacker can check guesses internally
- Specialized password lists more effective than literal dictionaries
  - Also generation algorithms (s $\to$ \$, etc.)
- $\sim$25% of passwords consistently vulnerable

Better password hashing

- Generate random salt s, store $(s, h(s, p))$
  - Block pre-computed tables and equality inferences
  - Salt must also have enough entropy
- Deliberately expensive hash function
  - AKA password-based key derivation function (PBKDF)
  - Requirement for time and/or space

Password usability

- User compliance can be a major challenge
  - Often caused by unrealistic demands
- Distributed random passwords usually unrealistic
- Password aging: not too frequently
- Never have a fixed default password in a product

Backup authentication

- Desire: unassisted recovery from forgotten password
- Fall back to other presumed-authentic channel
  - Email, cell phone
- Harder to forget (but less secret) shared information
  - Mother's maiden name, first pet's name
- Brittle: ask Sarah Palin or Mat Honan

Centralized authentication

- Enterprise-wide (e.g., UMN ID)
- Anderson: Microsoft Passport
- Today: Facebook Connect, Google ID
- May or may not be single-sign-on (SSO)

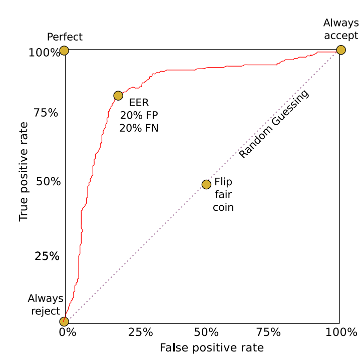
Biometric authentication

- Authenticate by a physical body attribute
- \+ Hard to lose
- − Hard to reset
- − Inherently statistical
- − Variation among people

Example biometrics

- (Handwritten) signatures
- Fingerprints, hand geometry
- Face and voice recognition
- Iris codes

Error rates: ROC curve



Outline

- OS protection and isolation (cont'd)
- OS security: authentication
- **Basics of access control**
- Announcements, Ex. 1 debrief
- Multilevel and mandatory access control
- Capability-based access control

Mechanism and policy

- Decision-making aspect of OS
- Should subject S (user or process) be allowed to access object (e.g., file) O?
- Complex, since admin must specify what should happen

Access control matrix

	grades.txt	/dev/hda	/usr/bin/bcvi
Alice	r	rw	rx
Bob	rw	-	rx
Carol	r	-	rx

Slicing the matrix

- $O(nm)$ matrix impractical to store, much less administer
- Columns: access control list (ACL)
  - Convenient to store with object
  - E.g., Unix file permissions
- Rows: capabilities
  - Convenient to store by subject
  - E.g., Unix file descriptors

Groups/roles

- Simplify by factoring out commonality
- Before: users have permissions
- After: users have roles, roles have permissions
- Simple example: Unix groups
- Complex versions called role-based access control (RBAC)

Outline

- OS protection and isolation (cont'd)
- OS security: authentication
- Basics of access control
- **Announcements, Ex. 1 debrief**
- Multilevel and mandatory access control
- Capability-based access control

Reversing the stack

```
void func(char *str) {
  char buf[128];
  strcpy(buf, str);
  do_something();
  return;
}
```

Payment app

```
void payment(char *name, int amount_jpy,
             char *purpose) {
    float amount_usd = amount_jpy/109.2;
    char memo[32];
    strcpy(memo, "Payment for: ");
    strcat(memo, purpose);
    write_check(name, amount_usd, memo);
}
```

Reverse range

```
void reverse_range(int *a, int from,
                   int to) {
    int *p = &a[from]; int *q = &a[to];
    while (!(p == q + 1 || p == q + 2)) {
        *p += *q;
        *q = *p - *q;
        *p = *p - *q;
        p++; q--;
    }
}
```

Deadlines reminder

- Yesterday: Project progress reports
- Tomorrow: Ex. 2
- Week from today: midterm

Outline

OS protection and isolation (cont'd)

OS security: authentication

Basics of access control

Announcements, Ex. 1 debrief

Multilevel and mandatory access control

Capability-based access control

MAC vs. DAC

- Discretionary access control (DAC)
  - Users mostly decide permissions on their own files
  - If you have information, you can pass it on to anyone
  - E.g., traditional Unix file permissions
- Mandatory access control (MAC)
  - Restrictions enforced regardless of subject choices
  - Typically specified by an administrator

Motivation: it's classified

- Government defense and intelligence agencies use *classification* to restrict access to information
- E.g.: Unclassified, Confidential, Secret, Top Secret
- Multilevel Secure (MLS) systems first developed to support mixing classification levels under timesharing

Motivation: system integrity

- Limit damage if a network server application is compromised
  - Unix DAC is no help if server is root
- Limit damage from browser-downloaded malware
  - Windows DAC is no help if browser is "administrator" user

Bell-LaPadula, linear case

- State-machine-like model developed for US DoD in 1970s

1. A subject at one level may not read a resource at a higher level
   - Simple security property, "no read up"
2. A subject at one level may not write a resource at a lower level
   - * property, "no write down"

High watermark property

- Dynamic implementation of BLP
- Process has security level equal to highest file read
- Written files inherit this level

Biba and low watermark

- Inverting a confidentiality policy gives an integrity one
- Biba: no write up, no read down
- Low watermark policy
- BLP $\wedge$ Biba $\Rightarrow$ levels are isolated

Information-flow perspective

- Confidentiality: secret data should not flow to public sinks
- Integrity: untrusted data should not flow to critical sinks
- Watermark policies are process-level conservative abstractions

Covert channels

- Problem: conspiring parties can misuse other mechanisms to transmit information
- Storage channel: writable shared state
  - E.g., screen brightness on mobile phone
- Timing channel: speed or ordering of events
  - E.g., deliberately consume CPU time

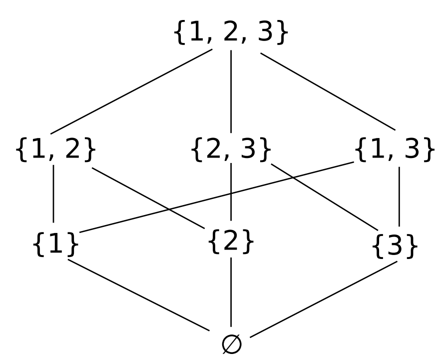
Multilateral security / compartments

- In classification, want finer divisions based on need-to-know
- Also, selected wider sharing (e.g., with allied nations)
- Many other applications also have this character
  - Anderson's example: medical data
- How to adapt BLP-style MAC?

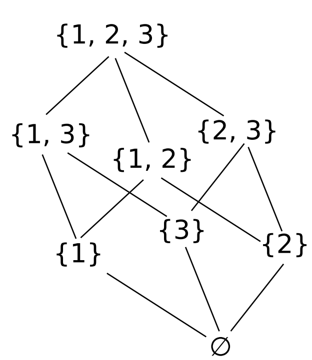
Partial orders and lattices

- $\leq$ on integers is a *total order*
  - Reflexive, antisymmetric, transitive, $a \leq b$ or $b \leq a$
- Dropping last gives a *partial order*
- A *lattice* is a partial order plus operators for:
  - Least upper bound or join $\sqcup$
  - Greatest lower bound or meet $\sqcap$
- Example: subsets with $\subseteq$, $\cup$, $\cap$

Subset lattice example



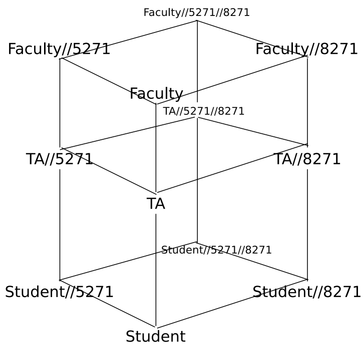
Subset lattice example



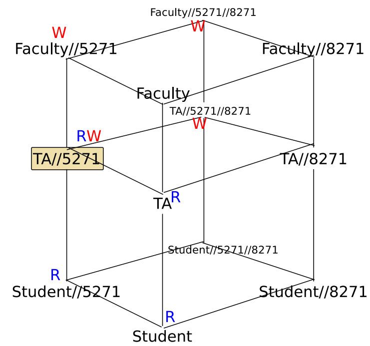
Lattice model

- Generalize MLS levels to elements in a lattice
- BLP and Biba work analogously with lattice ordering
- No access to incomparable levels
- Potential problem: combinatorial explosion of compartments

Classification lattice example



Lattice BLP example



Another notation

Faculty
$\longrightarrow$ (Faculty, $\emptyset$)
Faculty//5271
$\longrightarrow$ (Faculty, $\{5271\}$)
Faculty//5271//8271
$\longrightarrow$ (Faculty, $\{5271, 8271\}$)

MLS operating systems

- 1970s timesharing, including Multics
- "Trusted" versions of commercial Unix (e.g. Solaris)
- SELinux (called "type enforcement")
- Integrity protections in Windows Vista and later

Multi-VM systems

- One (e.g., Windows) VM for each security level
- More trustworthy OS underneath provides limited interaction
- E.g., NSA NetTop: VMWare on SELinux
- Downside: administrative overhead

Air gaps, pumps, and diodes

- The lack of a connection between networks of different levels is called an *air gap*
- A *pump* transfers data securely from one network to another
- A *data diode* allows information flow in only one direction

Chelsea Manning cables leak

- Manning (née Bradley) was an intelligence analyst deployed to Iraq
- PC in a T-SCIF connected to SIPRNet (Secret), air gapped
- CD-RWs used for backup and software transfer
- Contrary to policy: taking such a CD-RW home in your pocket

http://www.fas.org/sgp/jud/manning/022813-statement.pdf

Outline

OS protection and isolation (cont'd)

OS security: authentication

Basics of access control

Announcements, Ex. 1 debrief

Multilevel and mandatory access control

Capability-based access control

ACLs: no fine-grained subjects

- Subjects are a list of usernames maintained by a sysadmin
- Unusual to have a separate subject for an application
- Cannot easily subset access (sandbox)

ACLs: ambient authority

- All authority exists by virtue of identity
- Kernel automatically applies all available authority
- Authority applied incorrectly leads to attacks

Confused deputy problem

- Compiler writes to billing database
- Compiler can produce debug output to user-specified file
- Specify debug output to billing file, disrupt billing

(Object) capabilities

- A *capability* both designates a resource and provides authority to access it
- Similar to an object reference
  - Unforgeable, but can copy and distribute
- Typically still managed by the kernel

Capability slogans (Miller et al.)

- No designation without authority
- Dynamic subject creation
- Subject-aggregated authority mgmt.
- No ambient authority
- Composability of authorities
- Access-controlled delegation
- Dynamic resource creation

Partial example: Unix FDs

- Authority to access a specific file
- Managed by kernel on behalf of process
- Can be passed between processes
  - Though rare other than parent to child
- Unix not designed to use pervasively

Distinguish: password capabilities

- Bit pattern itself is the capability
  - No centralized management
- Modern example: authorization using cryptographic certificates

Revocation with capabilities

- Use indirection: give real capability via a pair of middlemen
- $A \to B$ via $A \to F \to R \to B$
- Retain capability to tell R to drop capability to B
- Depends on composability

Confinement with capabilities

- A cannot pass a capability to B if it cannot communicate with A at all
- Disconnected parts of the capability graph cannot be reconnected
- Depends on controlled delegation and data/capability distinction

OKL4 and seL4

- Commercial and research microkernels
- Recent versions of OKL4 use capability design from seL4
- Used as a hypervisor, e.g. underneath paravirtualized Linux
- Shipped on over 1 billion cell phones

Joe-E and Caja

- Dialects of Java and JavaScript (resp.) using capabilities for confined execution
- E.g., of JavaScript in an advertisement
- Note reliance on Java and JavaScript type safety