

CS574

Computer Security

San Diego State University

Spring 2008

Lecture #12

Today's Structure

- Administrivia
- Recent News
- Questions
- Lecture

Administrivia

- Potential Job Opening
(Small real estate office, managing small network, keeping anti-virus software up to date, etc.)

Recent News

- Phishing scam: fake assassins.
- Fingerprinting all visitors in the name of “Security”.
- Crypto-Gram and RISKS digest.
- Anonymous posting to be illegal in Kentucky?
- Chinese “hackers”.
- Promised article by Neal Stephenson.

Questions?

Lecture

Authentication Recap

- authentication = identity + proof
- three ways: (1) know (2) have (3) about
- TANSTAAFL - always costs and risks
- multi-factor >> single-factor

On Obscuring Identity

- Why?
 - No need for anyone to know (least priv)
 - Whistleblowers
 - Fear of reprisal
- Where?
 - Web / Network
 - Email / Usenet

On Obscuring Identity

- Issues - Two-way versus One-way
- Proxies
- Remailers
 - type 0: psuedo-anonymous
 - type 1: cypherpunk
 - type 2: mixmaster
 - type 3: mixminion

Type 0

- Psuedo-anonymous remailer
- Allows for two-way communication.
- Replace header information with “anonymous” information, but keep original-to-anonymous mapping in a database.
- Weakness: seize server and reverse mapping.

Type I

- Cypherpunk remailer
- One-way communication only.
- Encapsulate header and message in body; remailer removes outermost header and forwards on the remainder.
- Encipherment basically required.
- Weakness: monitoring server's input/output can de-anonymize communication.

Type 2

- Mixmaster remailer
- One-way communication only.
- Take a type 1 remailer and fragment and pad messages to fixed sizes, then randomize sending order.
- Final receiver must reconstitute message.

Type 3

- Mixminion remailer
- Allows for replies through the mix network.
- Similar to a type 2 remailer, with additional features and protections against more attacks.
- See <http://mixminion.net/>

Other Anonymizing Networks

- Anonymized TCP/IP traffic
 - Tor (<http://tor.eff.org/>)
- P2P Document redistribution
 - Freenet (<http://freenetproject.org/>)
- etc.

Review

Access Control Recap

- Access Control Matrix
- Access Control Lists
- Capability Lists

Access Control Recap

- Access control
- Authorization
- Policy
- Subject
- Object
- Mediator
- Right

Operating System Features

Operating System Protection Features

- Access controls rely on the OS to protect resources.
- We want to share resources in a controlled manner, as specified by our policies.
- OS acts as (or delegates to) a trusted mediator to enforce access control.

Operating Systems

- How did we get here?
 - one computer, one program
 - program loaders / *executives*
 - monitors (*multiprogramming/timeslicing*)
- Hardware was expensive, so there was a desire to share cost over lots of users.
- Hardware is now cheap, but also fast, so lots of independent processes and networking.

OS Protection

- What are we protecting?
 - memory and CPU
 - shared devices (tape, disk, network, ...)
 - reusable devices (printers, cameras, ...)
 - sharable data (files, directories, ...)
 - sharable software (libs, programs, OS, ...)
- Protecting from whom?
 - other users
 - ourselves
 - intruders

OS Protection

- Primary protection principle is *separation*.
- Keep a user's objects away from other users.
- Four basic categories/mechanisms of separation.

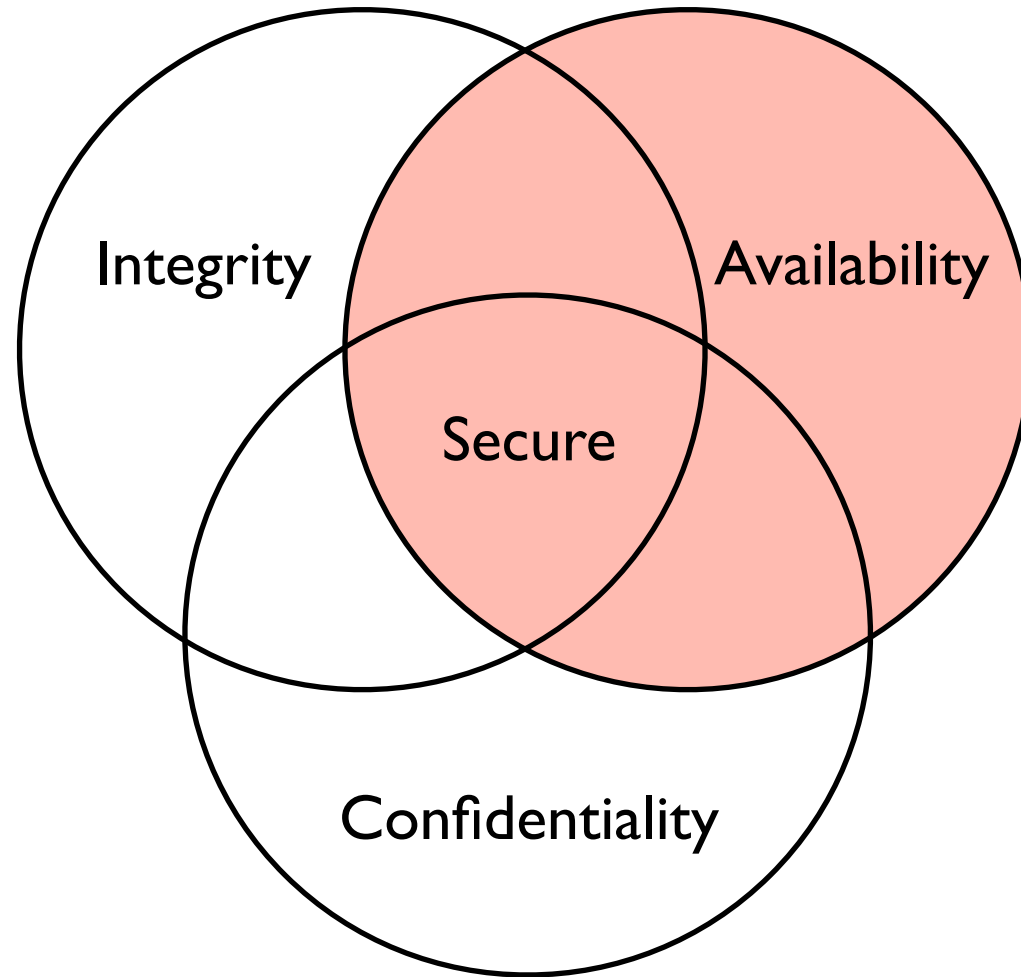
Separation

- Physical
 - use different machines / devices.
- Temporal
 - only allow serial, not concurrent, use.
- Logical
 - virtualization: the illusion of physical or temporal separation.
- Cryptographic
 - even though data can be seen, it can't be read (decrypted) or modified.

Sharing

- Mostly computers process information; sharing resources and information is what makes computers useful for most people.
- There are many different types of sharing.
- ‘Sharing’ is making resources *available*.

Sharing



Types of Sharing

- Share everything, all the time (no protection)
- Share nothing, all the time (isolation)
- Share all, or share nothing (public/private)
- Access Control (the DAC/MAC approach)
- Capabilities (dynamic rights, maybe transferrable)
- Limited use (app-specific; e.g., “view-not-print”)

Granularity of Sharing

- How big are the pieces being shared/controlled?
- Tradeoffs: the finer the granularity –
 - ...the better the security.
 - ...the more expensive it is to implement.
 - ...the more confusing to users it becomes.

Granularity of Sharing

- Objects
 - all or nothing; share whole object or don't.
 - offsets, pages, segments, or words
 - table, row, column, or cell in databases
- Subjects
 - everyone or nobody
 - user OR group
 - user AND group
 - user AND role

Protecting Memory

- Fences
- Base/Bounds
- Tagged
- Segmentation
- Paging
- Paging + Segmentation

Fences

- A lower bound in memory.
- Protects just the OS from program(s).
- Program must fit in contiguous memory.
- Primitive, but easy to implement.
- Hardware Support?
 - no – *relocate* the program at load time
 - yes – use a (variable) *fence register*

Base and Bounds

- Have a lower and an upper bound.
- Protects OS from programs and programs from other programs.
- Hardware support required – need a *base register* and a *limit register*.
- Each process gets own register values.
- Registers set during a *context switch*.

Base and Bounds

- Each memory access checked against base and bound registers.
- Programs still must be in contiguous memory.
- Break up programs – add more registers!
 - instructions and data
 - instructions, read-only data, writable data
etc.

Tagged Memory

- Use a 1-bit tag for capabilities.
- Every memory “unit” (typically a word) has some number of tag-bits.
- Protects OS from programs and programs from other programs.
- May possibly protect a program from itself.

Tagged Memory

- Tags may indicate ownership (user id, process number, role, etc.).
- Tags may indicate expected use (code, data, constants, stack, etc.).
- Tags may indicate access rights (read, write, execute, etc.).
- Tags may indicate data type (capability, pointer, string, integer, array, etc.).

Segmentation

- Divides a program (process) into pieces.
- Each piece is a *segment*.
- Can be thought of having an unlimited (or nearly so) number of base/bound registers.
- Each segment has its own access-control policy (i.e., a really big memory “unit”).

Segmentation

- Each segment is a chunk of contiguous memory.
- Each segment has a variable size.
- Each segment has a unique name.
- Memory addressing is by segment name + offset within the segment.
- Memory access is converted from (name,offset) tuple via the *segment translation table*.

Segmentation Problem

- What happens if an offset is too large?
- Three approaches:
 - do nothing
 - $\text{offset} = \text{offset} \bmod \text{size}(\text{segment})$
 - raise an exception

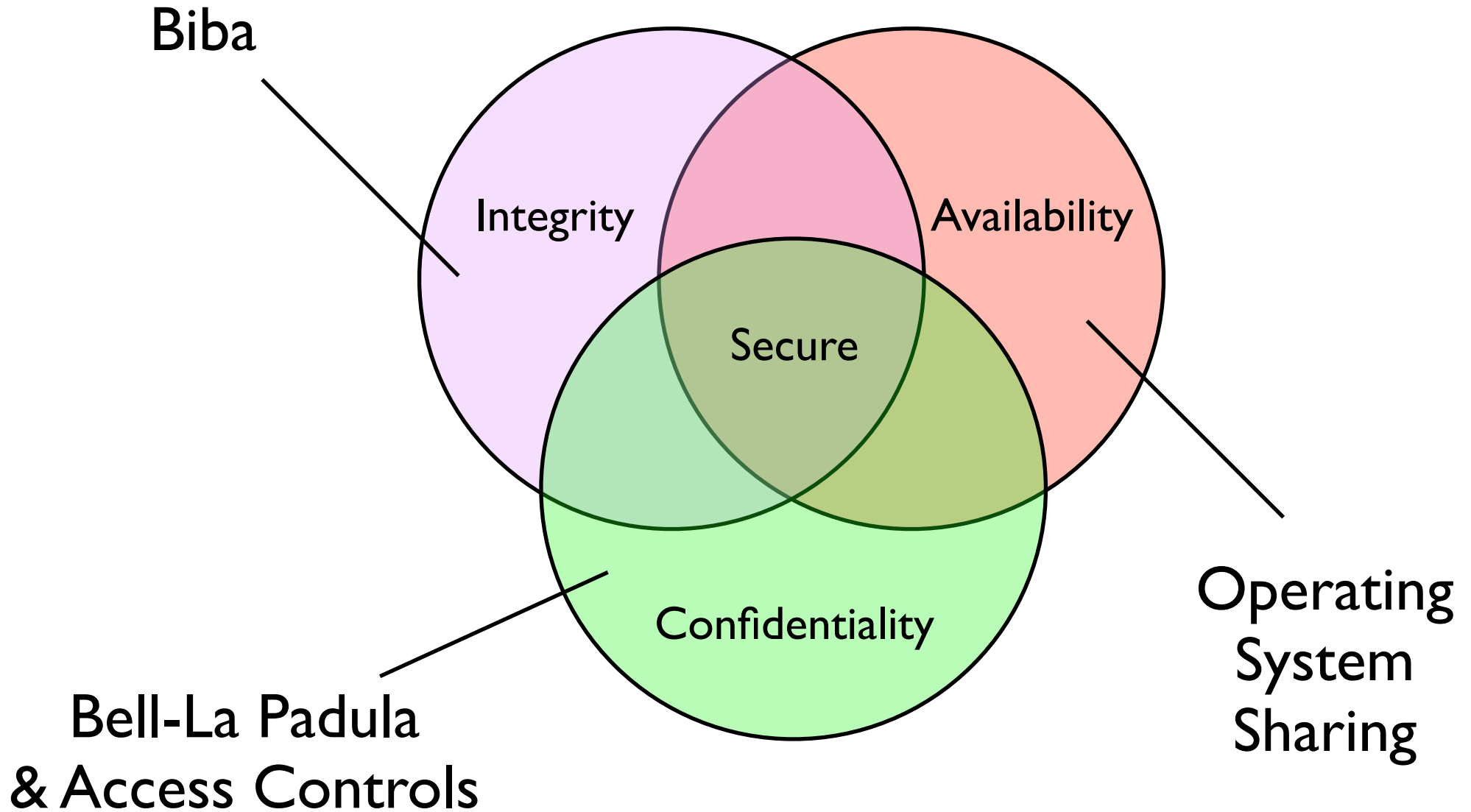
Paging

- Memory is divided into *page frames* and programs/processes are divided into *pages*.
- Size of a page frame is generally a power of 2.
- Use a *page translation table* to find actual memory location.
- Coarse granularity for access control – several pages may belong to a process.
- Sufficient to protect OS from programs.

Paging + Segmentation

- Why not run a segmentation system on top of a paging system?
- Segmentation gives you per-segment access control policies (e.g., r,w,x).
- Paging gives you protection from large offsets (plus cool stuff like virtual memory).

Security



Lecture References

- Pfleeger, *Security in Computing*, 3rd & 4th edition
- Bishop, *Introduction to Computer Security*
- Tor website, March 2008, <http://tor.eff.org>
- Freenode website, March 2008, <http://freenetproject.org>
- Mixminion website, March 2008, <http://mixminion.net>

End Lecture

Reading

- Chapter 4
- Chapter 5

Finis