Week 8 - Wednesday

COMP 4290

Last time

- Web security
- Web attacks

Questions?

Project 2

Assignment 3

Abiral Pokarel Presents

Back to E-mail Attacks

E-mail spoofing

- SMTP is the protocol for sending e-mail
- It's very straight-forward
- The **from** field is easy to spoof
- There are protocols with authentication built in, but regular SMTP is entrenched how
- You can never trust header information in an e-mail

Phishing

- **Phishing** is when an e-mail tries to trick someone into giving out private data or doing something else unsafe
- **Spear phishing** is phishing that targets a specific individual
  - Details about that user's life or accounts might be included
- **Whaling** is a term used for spear phishing of rich people or celebrities
  - They have more money
  - Many of their personal details could be public

Secure e-mail systems

- PGP (Pretty Good Privacy) is a system that uses the encryption mechanisms we described to send safe e-mails
  - The public key system uses a decentralized web of trust where you add your friends' keys to your web and get keys for their friends and friends of friends
- S/MIME is a standard that is like PGP, but it uses hierarchies of trust based on certificates from central authorities instead of a web

OS Security

OS security

- The OS has to enforce much of the computer security we want
  - Multiple processes are running at the same time
- We want protection for:
  - Memory
  - Hard disks
  - I/O devices like printers
  - Sharable programs
  - Networks
  - Any other data that can be shared

OS issues

- The OS has many functions that involve computer security
  - Enforced sharing
  - Interprocess communication
  - Protection of OS data
  - Guaranteed fair service
  - Interface to hardware
  - User authentication
  - Memory protection
  - File and I/O device access control
  - Allocation and access control of general objects

OS history

- Originally, there were only single users
  - Users programmed directly on the hardware
- Multitasking
  - Modern systems run multiple **processes** (which **do not** share memory)
  - Each process can have multiple **threads** (which **do** share memory)
    - The book also mentions **tasks**, which are the same as threads on many systems
  - We don't want processes to interfere with each other

Virtualization

- **Virtualization** is key to OS security, particularly of memory and disk usage
- Virtualization means providing one visible set of resources that are actually built on other resources
  - User A only sees user A resources because it sees a virtual machine tailored for it
- A **hypervisor** is the program that implements the virtual machine, acting as mediator between virtual resources and real resources
- A **sandbox** is a virtual machine that prevents programs from endangering the rest of the system

Separation

- OS security is fundamentally based on separation
  - **Physical separation:** Different processes use different physical objects
  - **Temporal separation:** Processes with different security requirements are executed at different times
  - **Logical separation:** Programs cannot access data or resources outside of permitted areas
  - **Cryptographic separation:** Processes conceal their data so that it is unintelligible

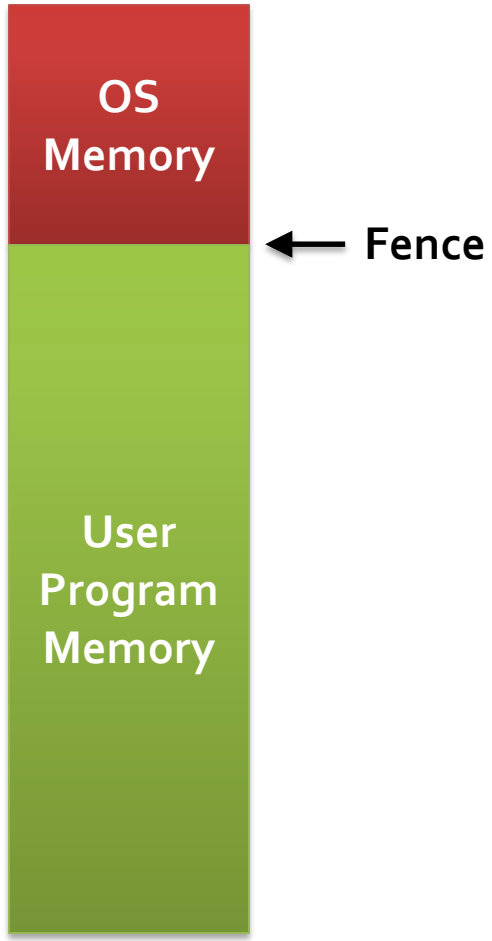
Memory Protection

Memory protection

- Protecting memory is one of the most fundamental protections an OS can give
  - All data and operations for a program are in memory
  - Most I/O accesses are done by writing memory to various locations
- Techniques for memory protection
  - Fence
  - Base/bounds registers
  - Tagged architectures
  - Segmentation
  - Paging

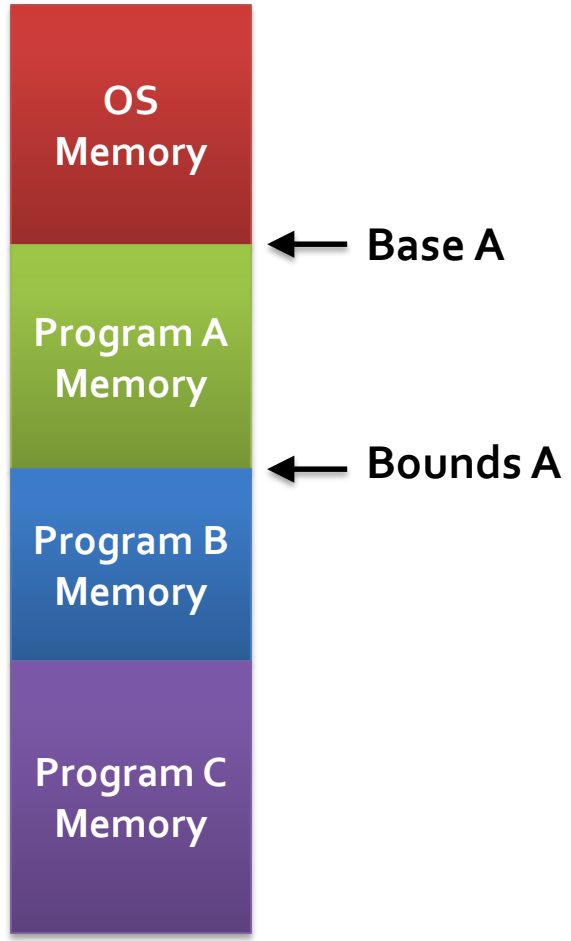
Fence

- A fence can be a predefined or variable memory location
- Everything below the fence is for the OS
- If a program ever tries to access memory below the fence, it either fails or is shut down
- As with many memory schemes, code needs to be relocatable so that the program is written as if it starts at memory location 0, but actually can be offset to an appropriate location



Base/bounds registers

- In modern systems, many user programs run at the same time
- We can extend the idea of a fence to two registers for each program
  - The **base register** gives the lowest legal address for a particular user program
  - The **bounds register** gives the highest legal address for a particular user program

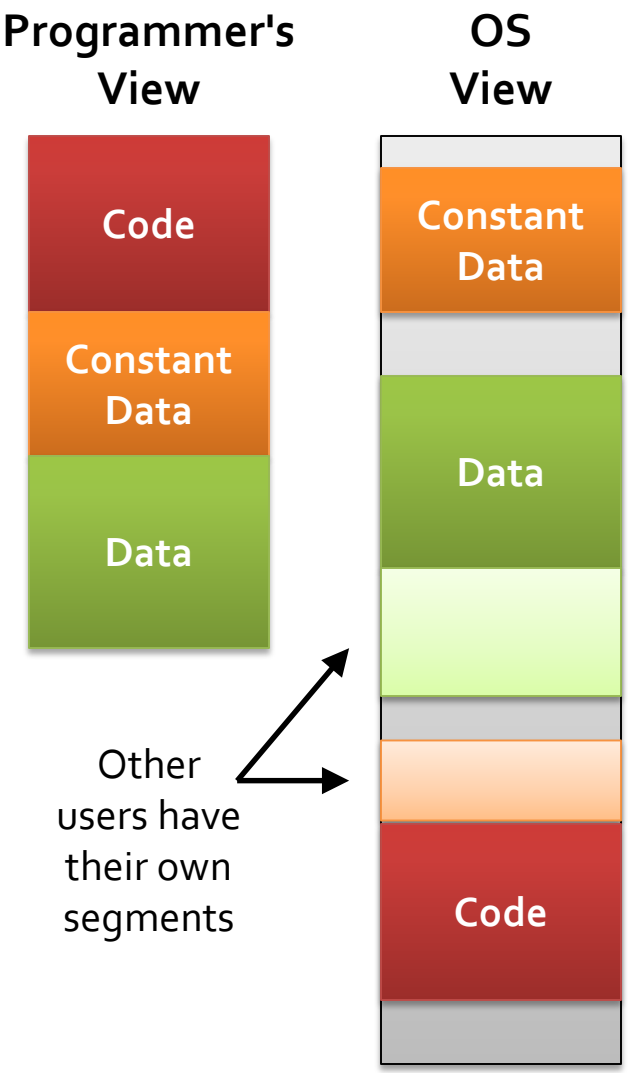


Tagged architectures

- The idea of base and bounds registers can be extended so that there are separate ranges for the program code and for its data
- It is possible to allow data for some users to be globally readable or writable
  - But this makes data protection all or nothing
- **Tagged architectures** allow every byte (or perhaps defined groups of bytes) to marked read only, read/write, or execute only
- Only a few architectures have used this model because of the extra overhead involved

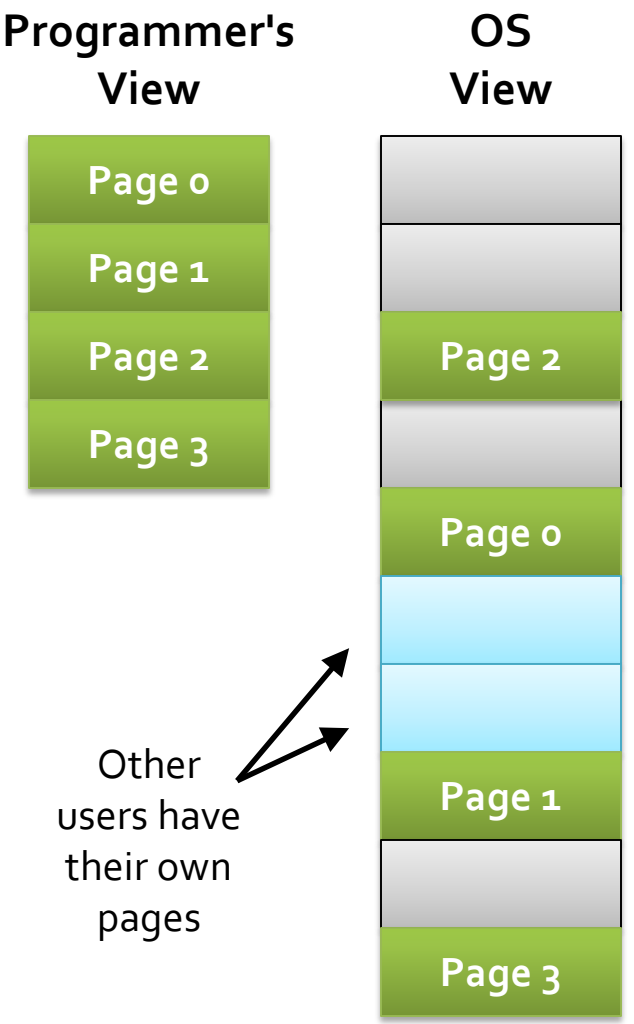
Segmentation

- **Segmentation** has been implemented on many processors including most x86 compatibles
- A program sets up several segments such as code, data, and constant data
  - Writing to code is usually illegal
  - Other rules can be made for other segments
- A memory lookup is both a segment identifier and an offset within that segment
- For performance reasons, the OS can put these segments wherever it wants and do lookups
  - Segments can be put on secondary storage if they are not currently in use
- The programmer sees a solid block of memory



Paging

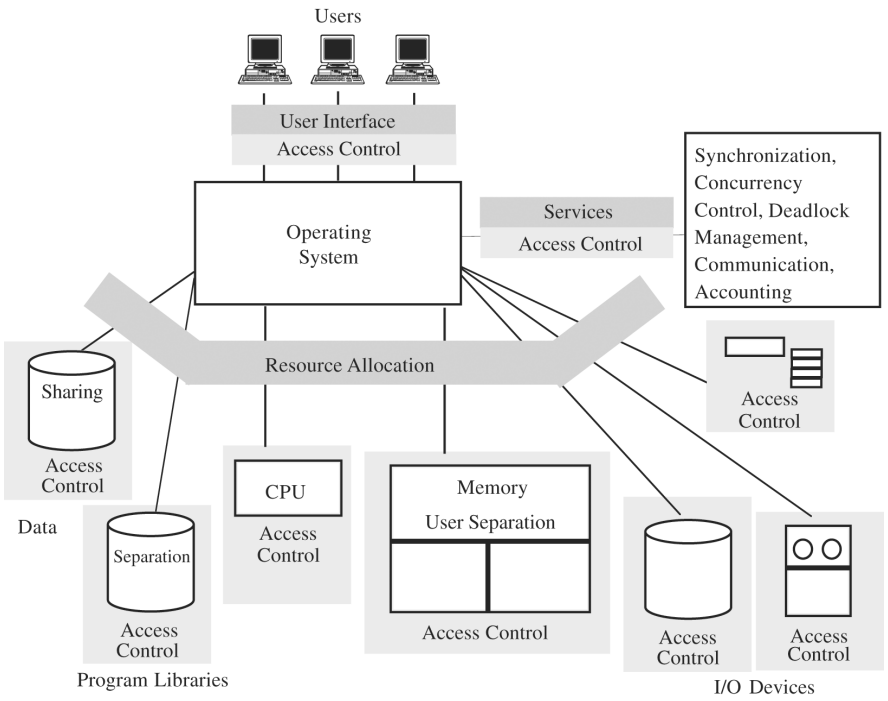
- **Paging** is a very common way of managing memory
- A program is divided up into equal-sized pieces called pages
  - An address is page number and an offset
- Paging doesn't have the fragmentation problems that segmentation does
  - It also doesn't specify different protection levels
- Paging and segmentation can be combined to give protection levels



Trusted Operating Systems

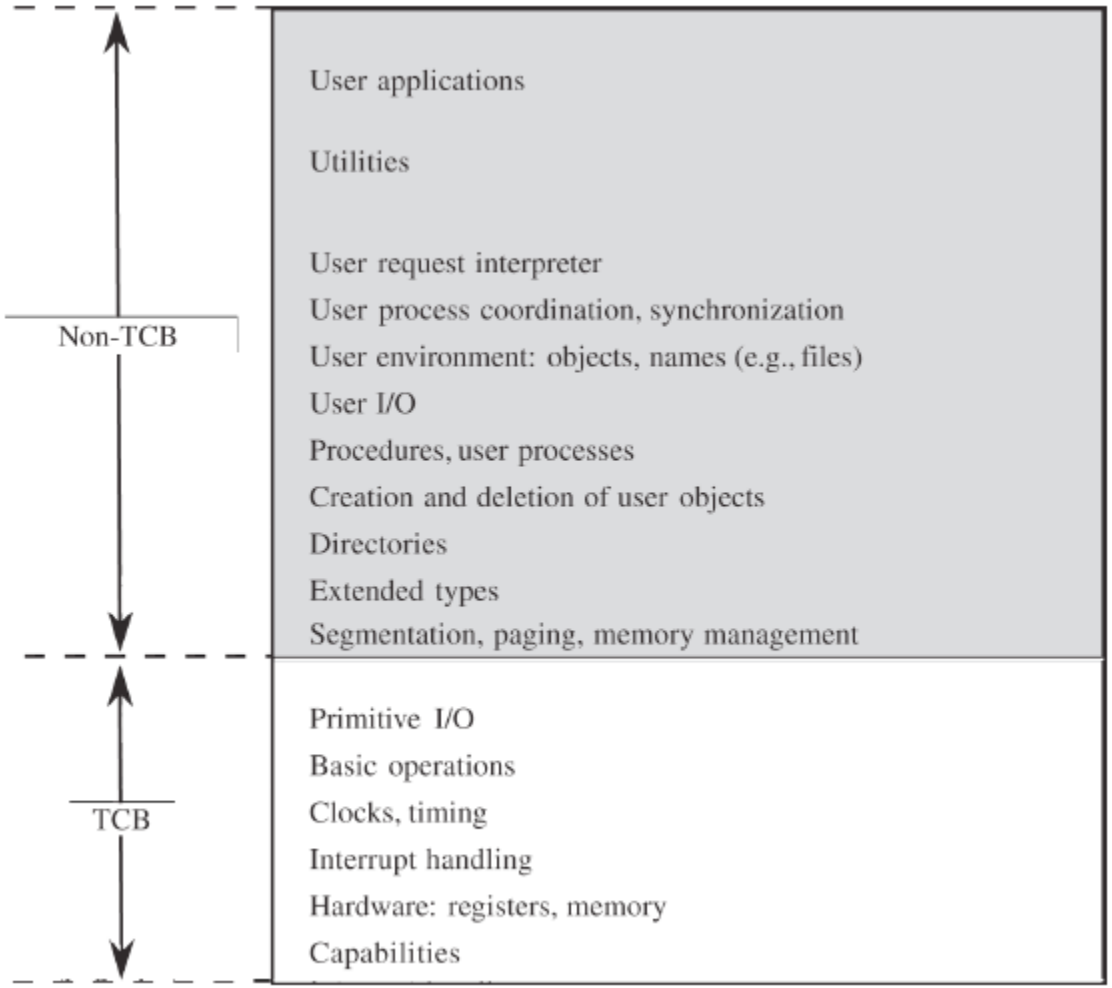
Trusted OS

- A trusted OS is similar to a normal OS, except that it puts a layer of access control around everything
- A trusted OS will typically be careful about:
  - User identification and authentication
  - Mandatory access control
  - Discretionary access control
  - Object reuse protection
  - Complete mediation
  - Trusted paths
  - Auditing
  - Intrusion detection



Trusted Computing Base

- The **trusted computing base** or **TCB** is the parted of a trusted OS that enforces the security policy
- It has to handle the most hardcore stuff



Mandatory and discretionary access control

- **Mandatory access control** (**MAC**) means that the controls are enforced by rules in the system, not by user choices
  - In a few slides, we'll talk about Bell-La Padula, a classic MAC system
- **Discretionary access control** (**DAC**) means that the user has control over who can access the objects he or she owns
  - Linux and Windows are largely DAC systems
- Many real systems have elements of both

Object reuse

- When a file is deleted, it isn't actually deleted
  - It's blocks are unlinked from the file system
- When you create a new file, it usually uses a block from an old deleted file
- You can examine the contents of that block and reconstruct some or all of the deleted file
  - Software is available for home users to undelete files
  - Digital forensics experts use more powerful tools in criminal investigations
- The problem is that **object reuse** allows for security violations
- A regular OS often does this and other kinds of object reuse for efficiency
- A trusted OS will sacrifice efficiency for security

Complete mediation and trusted paths

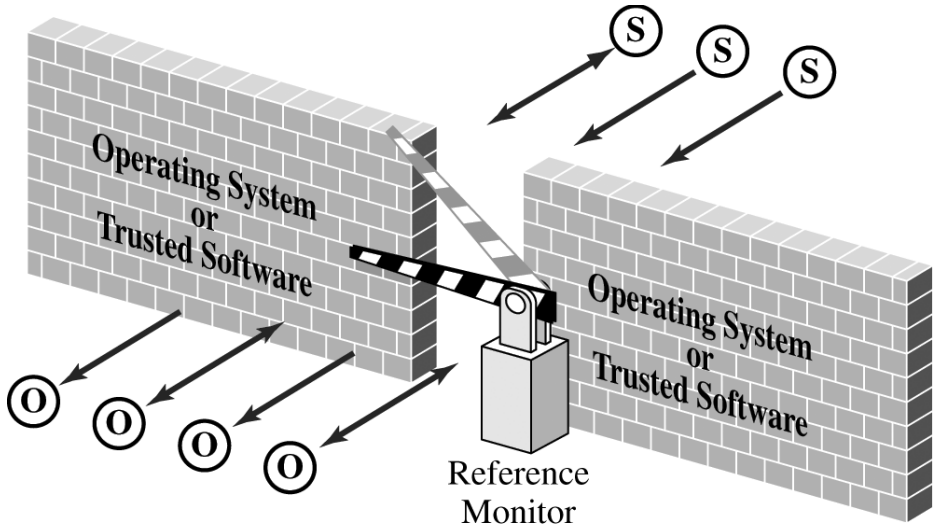
- **Complete mediation** means that every access goes through the system
  - All resources are checked
  - Past permissions are no guarantee of future permissions
- A **trusted path** means an unmistakable process for performing protected tasks
  - Phishing is the opposite of a trusted path
  - Some attacks on OS users rely on getting them to download a file with the same name as a system command, which will then be run instead if they execute from the same directory

Auditing

- Trusted systems also keep an audit log of all security-relevant actions that have been taken
- Unfortunately, audit logs can become huge
- Even if an illegal access is known to have happened, it might be impossible to find it in the logs
- Audit reduction is the process of reducing the size of the log to critical events
  - This may require sophisticated pattern recognition software

Kernelized design

- One approach to making a trusted system is a **kernelized design**
- A **security kernel** is the low level part of the OS that enforces security mechanisms
  - It can be a unified layer sitting between hardware and the rest of the OS
  - Or it can be spread throughout the entire OS
- The **reference monitor** is the most important part of the security kernel
  - It controls accesses to objects
  - It should be **tamperproof**, **unbypassable**, and **analyzable**

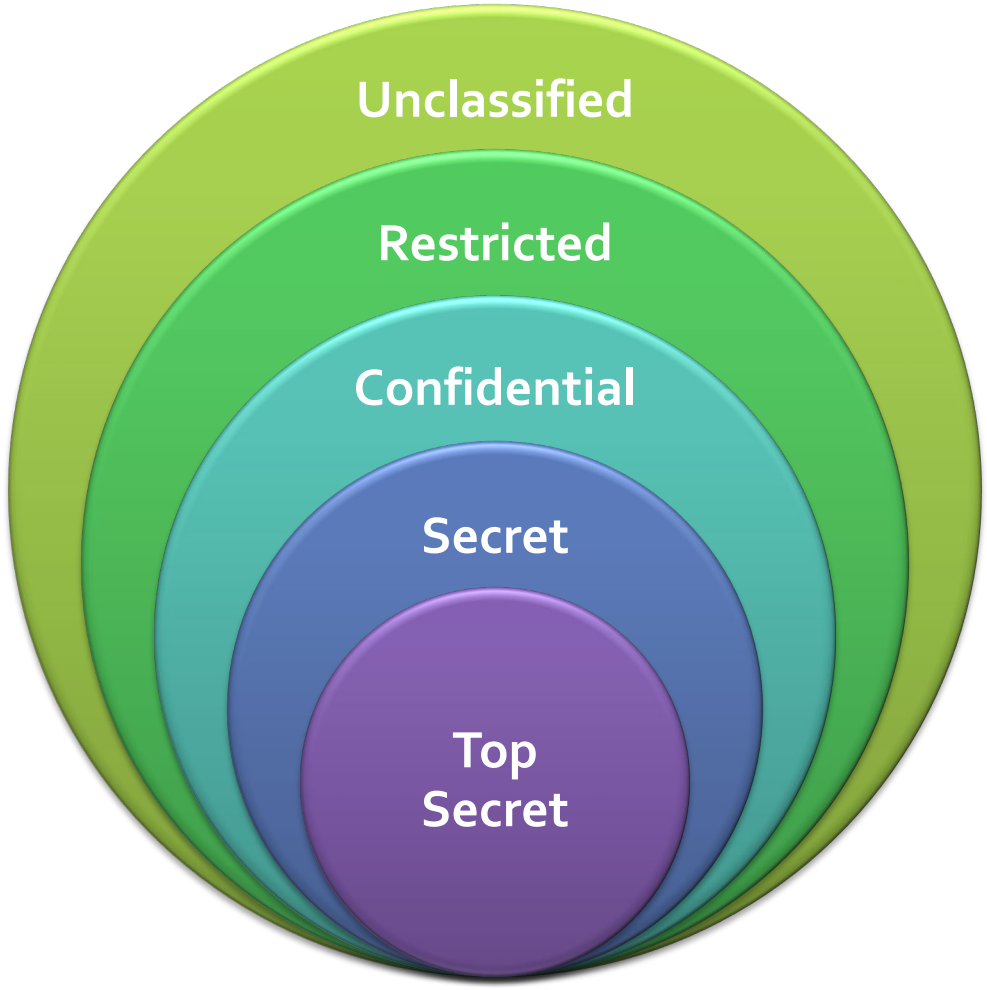


Mandatory Access Control

Bell-La Padula Model

Bell-La Padula overview

- Confidentiality access control system
- Military-style classifications
- Uses a linear clearance hierarchy
- All information is on a **need-to-know** basis
- It uses **clearance** (or **sensitivity**) levels as well as project-specific **compartments**



Security clearances

- Both subjects (users) and objects (files) have security clearances
- Below are the clearances arranged in a hierarchy

Clearance Levels	Sample Subjects	Sample Objects
Top Secret (TS)	Tamara, Thomas	Personnel Files
Secret (S)	Sally, Samuel	E-mail Files
Confidential (C)	Claire, Clarence	Activity Log Files
Restricted (R)	Rachel, Riley	Telephone List Files
Unclassified (UC)	Ulaley, Ursula	Address of Headquarters

Simple security condition

- Let $level_O$ be the clearance level of object ***O***
- Let $level_S$ be the clearance level of subject ***S***
- The simple security condition states that ***S*** can read ***O*** if and only if the $level_O \leq level_S$ and ***S*** has discretionary read access to ***O***
- In short, you can only read down
- In a few slides, we will expand the simple security condition to make the concept of level

*-Property

- The *-property states that ***S*** can write ***O*** if and only if the $level_S \leq level_O$ and ***S*** has discretionary write access to ***O***
- In short, you can only write up

Basic security theorem

- Assume your system starts in a secure initial state
- Let ***T*** be all the possible state transformations
- If every element in ***T*** preserves the simple security condition and the *-property, every reachable state is secure
- This is sort of a stupid theorem, because we define "secure" to mean a system that preserves the security condition and the *-property

Upcoming

Next time…

- Finish access control models
- Rootkits
- Network basics
- Network threats
- Network attacks
- Adam Garantche presents

Reminders

- Read sections 6.1 and 6.2
- Keep working on Project 2
- Work on Assignment 3
  - Due on Friday