ECE 455: CYBERSECURITY

Lecture #2

Daniel Gitzel

In the news...

- **LastPass bug leaks credentials from previous site (9/16/2019)**
  - bug relies on executing malicious JavaScript code alone, with no other user interaction
  - lure users on malicious pages to steal passwords
  - https://bugs.chromium.org/p/project-zero/issues/detail?id=1930

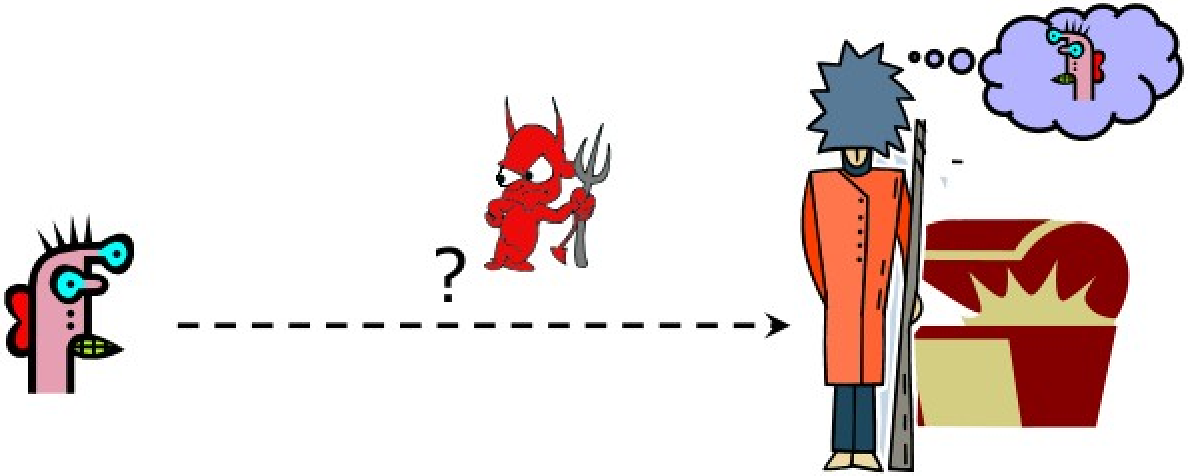
Announcement

- **Read papers for quiz next week**
- **Security Review due next week**
- **Working on uLab images for lab assignments**
  - Limited set of VMs for those of you w/o Unix-like system

REVIEW

Basic Problem

- **How do you prove to someone that you are who you claim to be?**
  - Any system with access control must solve this problem.



Security Goals

- **Accountability** is the ability to identify and authenticate users and audit actions.
- **Non-repudiation** is unforgeable evidence that a specific action has occurred.

How to Prove Who You Are

- **What you know**
  - Passwords
  - Answers to questions that only you know
- **Where you are**
  - IP address, geolocation
- **What you are**
  - Biometrics
- **What you have**
  - Secure tokens, mobile devices

Simple Idea!

- Password & user created
- System stores list of users & passwords
- System checks credentials at logon
- User authenticated

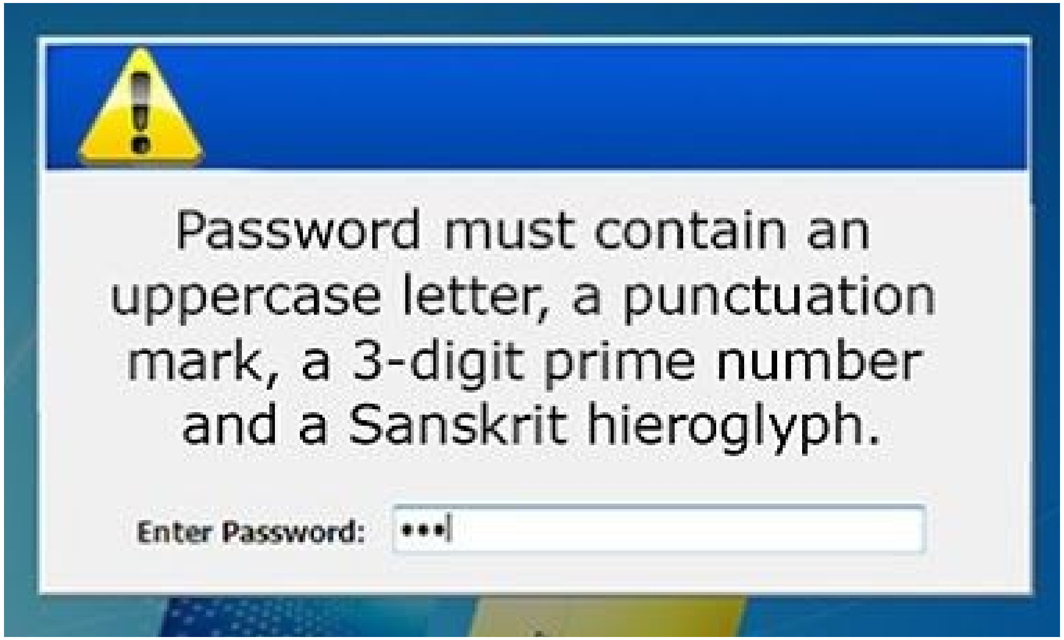
- Can you think of any issues?

Password Storage

- **Protecting the password file**
  - Don’t store plain-text passwords (obviously)
  - Don’t use encrypted passwords (dictionary attacks)
  - Use *hashed* passwords
- **Hash a salt along with the password**
  - Store the salt and the hashed salt+password on the server
  - Users with same password will have different password+salt!
- **Hash function (simple definition)**
  - Given x, $f(x)$ is **easy** to compute
  - Given $f(x)$, x is **hard** to compute

Password Usability

- **Classic recommendation:**
  - > 8 characters
  - At least 3: digits, lower/upper case, symbol
- **Backfires!**
  - Paper notes
  - Loopholes in rules



More Password Issues

- **Credential Stuffing**
  - using stolen credentials on other sites
- **No rate limiting**
  - Website allows brute force (automated guesses)
- **No multi-factor authentication**
  - Just password is enough
- **Weak password recovery mechanisms**
  - Remember the Palin email hack?
- **Application timeouts too long**
  - Did you know that sudo lasts 15 minutes?

Even More Password Issues

- **Keystroke loggers**
  - Hardware
  - Software (spyware)
- **Shoulder surfing**
- **Same password at multiple sites**
  - One breach becomes many!
- **Broken implementations**
  - TENEX timing attack

Improving Passwords

- **Add biometrics**
  - For example, keystroke/mouse dynamics or voice print
- **Graphical passwords**
  - Goal: easier to remember? no need to write down?
- **Password managers**
  - Examples: LastPass, built into browsers
  - Can have security vulnerabilities…
- **Two-factor authentication**
  - Leverage phone (or other device) for authentication

Improving Passwords (More)

- **Mutual Authentication**
  - User authenticates *and* site authenticates (prevent phishing)
- **Trusted Path**
  - Guarantee user only communicates with OS (CTRL+ALT+DEL)
- **Display number of failed attempts**
  - First try fails, second succeeds
  - But OS shows one login attempt!
- **Timeouts and Limits**
  - Prevent online guessing

ACCESS CONTROL

Introduction

- **User is authenticated**
  - I know who you are!
- **Who is allowed to do what?**
  - What privileges, permissions, power do you have?
- **Traditionally, consists of an operation performed on a resource**
  - read, write, execute on a file, directory, or port
- **Today, this can be more abstract**

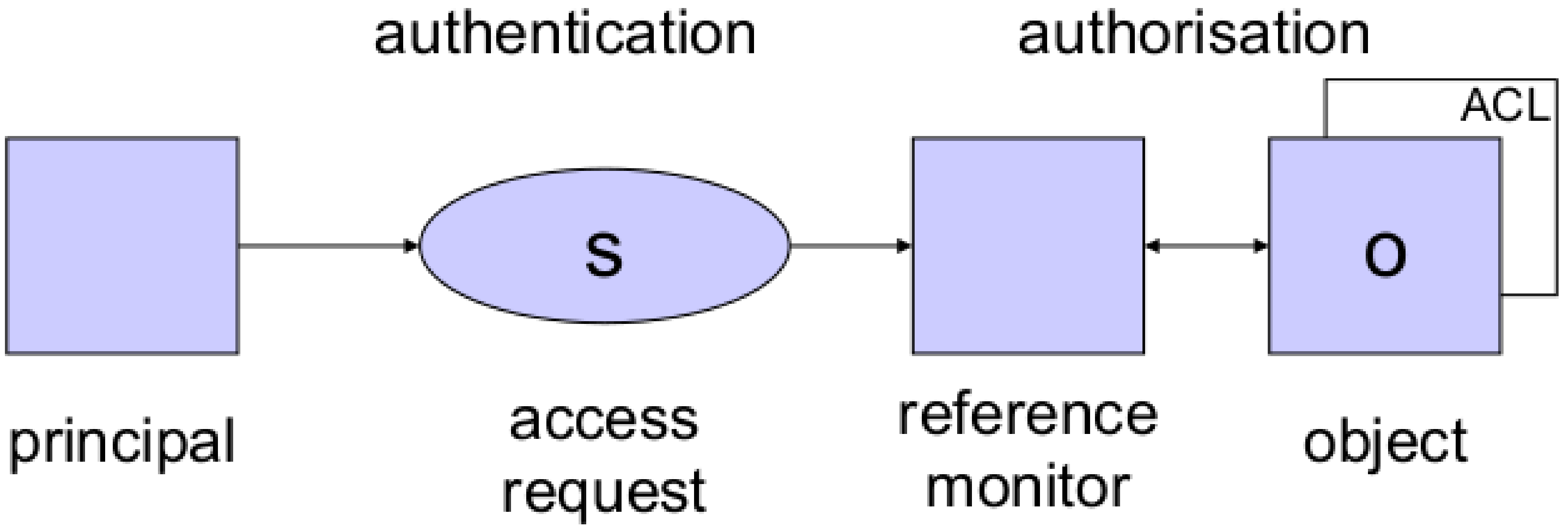
Agenda

- **Fundamental terminology**
  - Principals & subjects, access operations
- **Authentication & authorization**
- **Policies**
  - Capabilities & access control list
  - Discretionary & mandatory access control
  - Role Based Access Control
  - Policy instantiation
- **Structuring policies**
  - Partial orderings & lattices

Security Policies

- **Access control enforces operational security policies.**
- **A *policy* specifies who is allowed to do what.**
- **The active entity requesting access to a resource is called the *principal***
- **The resource access is requested for is called the *object*.**
- **Traditionally, policies refer to the requester's identity and decisions are *binary* (yes/no).**

Authentication vs. Authorization



B. Lampson, M. Abadi, M. Burrows, E. Wobber: Authentication in Distributed Systems: Theory and Practice, ACM Transactions on Computer Systems, 10(4), pages 265-310, 1992

Authentication vs. Authorization

- **Authentication: reference monitor verifies the identity of the principal making the request.**
  - A user identity is one example for a principal.
- **Authorization: reference monitor decides whether access is granted or denied.**
- **Reference monitor has to find and evaluate the security policy relevant for the given request.**
- **“Easy” in centralized systems.**
- **In distributed systems,**
  - How do we find all relevant policies?
  - How do we make decisions if policies may be missing?

Post-authentication

- **User enters username and password.**
  - If the values entered are correct, the user is "authenticated".
- **"The machine now runs on behalf of the user".**
  - This might be intuitive, but it is imprecise.
- **Log on creates a process that runs with access rights assigned to the user.**
- **Typically, the process runs under the *user identity* of the user who has logged on.**

Users & User Identities

- **Requests to reference monitor do not come directly from a user or a user identity, but from a *process*.**
- **The process "speaks for" the user (identity).**
- **The active entity making a request within the system is called the *subject*.**
- **Three concepts:**
  - User: person (Peter Cooper);
  - User identity (principal): name used in the system, possibly associated with a user (pcooper);
  - Process (subject) running under a given user identity (ls).

Principals & Subjects

- ***Policy*: A *principal* is an entity that can be granted *access* to objects or can make statements affecting access control decisions.**
  - Example: user ID
- ***System*: *Subjects* operate on behalf of *principals*; access is based on the principal's name bound to the subject in some un-forgeable manner at authentication time.**
  - Example: process (running under a user ID)

Principals & Subjects

- ***Principal*** **and** ***subject*** **are both used to denote the entity making an access request.**
- **The term** ***principal*** **can have different connotations, causing confusion.**
- **M. Gasser (1990): Because access control structures identify principals, it is important that principal names be** ***globally unique, human-readable and memorable*****, easily and reliably associated with known people.**
- **We will examine later whether this advice is still valid.**

Basic Terminology – Recap

***Subject/Principal*: Active entity – user or process.**

***Object*: Passive entity – file or resource.**

***Access operations*: basic memory access (read, write), method calls, push to network, etc.**

Comparable systems may use different access operations or attach different meanings to operations which appear to be the same.

Access Operations

- ***Access right***: right to perform an access or operation
- ***Permission***: typically a synonym for access right.
- ***Privilege***: typically a ***set of access rights*** given directly to roles like administrator, operator, ...
- These terms may have specific meanings in different systems.

Access Operations

- **On the most elementary level, a subject may**
  - observe an object, or
  - alter an object.
- **Some policies can be expressed with these *access modes*.**
- **A richer set of operations is more convenient.**

Elementary Access Operations

- **Bell-LaPadula model (see chapter 11) has four *access rights*:**
  - execute
  - read
  - append, also called blind write
  - write
- **Mapping between *access rights* and *access modes*:**

	execute	append	read	write
observe			X	X
alter		X		X

Rationale

- **Multi-user O/S:**
  - users open files to get access
  - files are opened for read or for write access
  - O/S can avoid conflicts like two users simultaneously writing to the same file.
- **Write access usually includes read access**
  - user editing a file should not be asked to open it twice
  - write includes observe and alter mode.
- **Few systems implement append**
  - altering an object without observing its content is rarely useful
- **A file can be used without being opened (read)**
  - example: running a binary or using a secret key

Access Rights (Unix/Linux)

- **Three access operations on files:**
  - read: from a file
  - write: to a file
  - execute: a file
- **Access operations on directories:**
  - read: list contents
  - write: create or rename files in the directory
  - execute: search directory
- **Deleting files/sub-directories handled by access operations in the directory.**

Administrative Access Rights

- **Policies for creating and deleting files expressed by**
  - access control on the directory (Unix)
  - specific create and delete rights (Windows, OpenVMS)
  - get, set, use, manage (in CORBA)
- **Policies for defining security settings such as access rights handled by:**
  - access control on the directory
  - specific rights like grant and revoke

ACCESS CONTROL STRUCTURES

Policy Focus

- **Principals & objects provide a different focus of control:**
  - What is the principal allowed to do?
  - What may be done with an object?
- **OS provides infrastructure for managing files and resources, i.e. objects**
  - Access control takes the second approach.
- **Application oriented systems, (e.g. database) provide services to the user**
  - Control actions of principals.
- **Note: some sources use authorization to denote the process of setting policies.**

Access Control Structure

- **Policy is stored in an *access control structure*.**
  - Captures your desired access control policy.
  - You should be able to check that your policy has been captured correctly.
- **Access rights can be defined individually for each combination of subject and object.**
- **For large numbers of subjects and objects, such structures are cumbersome to manage; *intermediate levels of control* are preferable.**

Access Control Matrix

- **At runtime, we could specify for each combination of subject and object the operations that are permitted.**
  - S ' set of subjects
  - O ' set of objects
  - A ' set of access operations
- **Access control matrix: $M = (M_{so})_{s \in S, o \in O}$**
- **Matrix entry $M_{so} \subseteq A$ the operations subject *s* may perform on object *o***
- **You can visualize the matrix as a (big) table.**

Access Control Matrix

- ***Access control matrix*** **has a row for each subject and a column for each object.**
  - The control matrix is an abstract concept,
  - not very suitable for direct implementation,
  - not very convenient for managing security.
- **How do you answer the question: Has your security policy been implemented correctly?**

	bill.doc	edit.exe	fun.com
Alice	-	{exec}	{exec,read}
Bob	{read,write}	{exec}	{exec,read,write}

Capabilities

- **Focus on the subject**
  - access rights stored with the subject
  - capabilities = rows of the access control matrix
  - Subjects may *grant rights* to other subjects; subjects may *grant the right to grant rights*.
- **How to check who may access a specific object?**
- **How to revoke a capability?**
- **Distributed system security has created renewed interest in capabilities.**

Alice	edit.exe: {exec}	fun.com: {exec,read}

Access Control Lists (ACLs)

- **Focus on the object**
  - access rights of principals stored with the object
  - ACLs = columns of the access control matrix
- **How to check access rights of a specific subject?**
- **ACLs implemented in most commercial operating systems but their actual use is limited.**

fun.com	Alice: {exec}	Bill: {exec,read,write}

Who Sets the Policy?

- **Security policies specify how principals are given access to objects.**
- **Responsibility for setting policy could be assigned to:**
  - the *owner* of a resource, who may decree who is allowed access; such policies are called *discretionary* as access control is at the owner's discretion.
  - a *system wide policy* decreeing who is allowed access; such policies are called *mandatory*.
- ***Warning*: other interpretations of discretionary and**
- **mandatory access control exist.**

DAC & MAC

- Access control based on policies that refer to user identities was historically called *discretionary access control* (DAC).
- Referring to individual users in a policy works best within closed organizations
- Access control based on policies that refer to security labels (confidential, top secret, ') was historically called *mandatory access control* (MAC).
- DAC and MAC have survived in computer security text books, but not very much in the wild.

Intermediate Levels

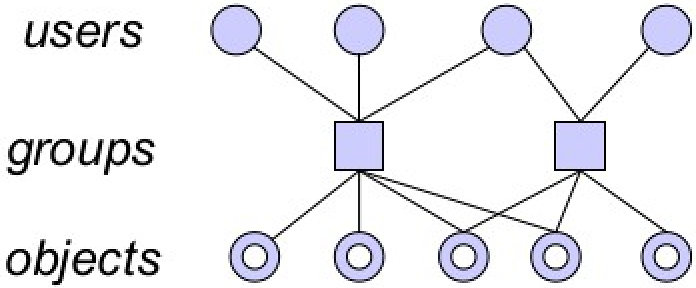
- “In computer science, problems of complexity are solved by adding another level of indirection.” [David Wheeler]
- Introduce intermediate layers between users and objects
- Represent policies in a more manageable fashion

IBAC & Groups

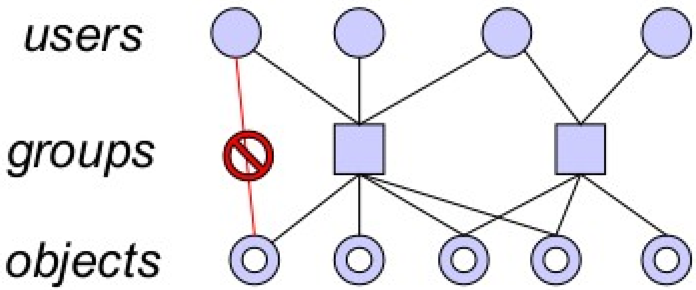
- **Identity based access control (IBAC) instead of DAC.**
  - IBAC does not scale well and will incur an “identity management” overhead.
- **Teacher wants to give students access to some documents.**
  - Putting names into several ACLs is tedious
  - Teacher defines a group
  - Declares the students to be members of group
  - Puts group into the ACLs
- **Access rights are often defined for groups:**
  - Unix: owner, group, others (3x octal format)

Groups & Negative Permissions

Groups: intermediate layer between users and objects.



To handle exceptions, negative permissions withdraw rights



Roles

- **Alternatively, we could have created a role ‘student’.**
- **A *role* is a collection of *procedures* assigned to users**
  - A user can have more than one role and more than one user can have the same role.
- **Teacher creates a procedure for reading course material, assigns this procedure to the role ‘student’.**
- **A role ‘course tutor’ could be assigned a procedure for updating documents.**

RBAC

- **Role Based Access Control**
- ***Procedures*: 'High level' access operations**
  - more complex semantic than read or write
  - procedures can only be applied to objects of certain data types
  - Example: Funds transfer between bank accounts.
- **Roles are a good match for typical access control requirements in business.**
- **RBAC typical found at the application level.**
- **Difference between groups and roles??**

More on RBAC

- **Role hierarchies define relationships between roles:**
  - Senior role has all access rights of the junior role.
- **Do not confuse the role hierarchy with the hierarchy of positions (superior – subordinate) in an organization**
  - These two hierarchies need not correspond.
- **Separation of duties is an important security principle**
  - numerous flavors of static and dynamic separation of duties policies exist.
  - Example: a manager is given the right to assign access rights to subordinates, but not the right to exercise those access rights
  - Example: accountant can submit expenses, but only treasurer can sign-off on spending

NIST: RBAC Levels

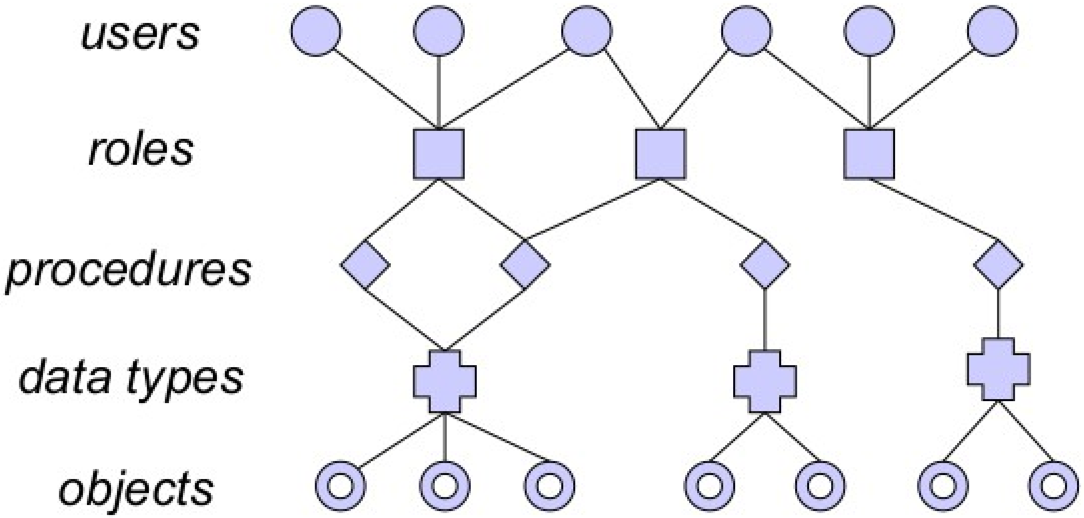
- **Flat RBAC:**
  - users are assigned to roles,
  - permissions are assigned to roles,
  - users get permissions via role membership;
  - support for user-role reviews.
- **Hierarchical RBAC: adds support for role hierarchies.**
- **Constrained RBAC: adds separation of duties.**
- **Symmetric RBAC: support for permission-role reviews (can be difficult to provide in large distributed systems).**

Role Based Access Control

- **Standard: American National Standards Institute: Role Based Access Control, ANSI-INCITS 359-2004.**
- **RBAC itself does not have a generally accepted meaning, and it is used in different ways by different vendors and users.**
- **[R. Sandhu, D. Ferraiolo, and R. Kuhn: The NIST Model for Role-Based Access Control: Towards a Unified Standard, Proceedings of the 5th ACM Workshop on Role-Based Access Control, Berlin, Germany, July 26-27, 2000**

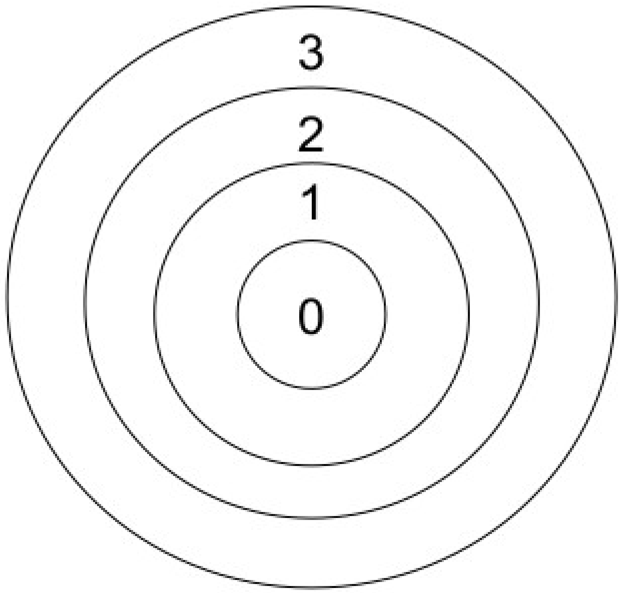
Lesson: Intermediate Controls

Intermediate controls for better security management; to deal with complexity, introduce more levels of indirection.



Protection Rings

Protection rings are mainly used for integrity protection.



Protection Rings

- **Each subject (process) and each object is assigned a number, depending on its ‘importance’, e.g.**
  - -1 – firmware or BIOS or hardware
  - 0 – operating system kernel
  - 1 – operating system
  - 2 – utilities
  - 3 – user processes
- **Numbers correspond to concentric protection rings, ring 0 in center gives highest degree of protection.**
- **If a process is assigned number i, we say the process “runs in ring i”.**
- **Access control decisions are made by comparing the subject’s and object’s ring.**

Policy Instantiation

- When developing software you will hardly know who will eventually make use of it.
- At this stage, security policies cannot refer to specific user identities.
- A customer deploying the software may know its “authorized” users and can instantiate a generic policy with their respective user identities.
- Generic policies will refer to ‘placeholder’ principals like owner, group, others (world, everyone).
- Reference monitor resolves values of ‘placeholders’ to user identities when processing an actual request.

STRUCTURING POLICIES

Structuring Access Control

- Some resources in an academic department can be accessed by all students, other resources only by students in a particular year.
- Department creates groups like ‘All-Students’ and ‘Y1-Students’.
- The two groups are related, Y1-Students is a subgroup of All-Students; if All-Students has access to a resource, so has Y1-Students.
- No such direct relationship between Y1-Students and Y2-Students.

Partial Orderings

- **We now can use *comparisons* in security policies:**
  - Is the user's group a subgroup of the group permitted to access this resource?
- **Some groups are related but others are not (e.g. Y1-Students and Y2-Students).**
- **Relationships are transitive: CS102-Students ⊆**
- **Y1-Students ⊆ All-Students**
- **In mathematical terms, we are dealing with a *partial ordering*.**

Mathematical Definition

- **A partial ordering ≤ (‘less or equal’) on a set L is a relation on L×L that is**
  - reflexive: for all $a \in L$, $a \leq a$
  - transitive: for all $a,b,c \in L$, if $a \leq b$ and $b \leq c$, then $a \leq c$
  - anti-symmetric: for all $a,b \in L$, if $a \leq b$ and $b \leq a$, then $a = b$
- **If a≤b, we say ‘b dominates a’ or ‘a is dominated by b’.**

Examples

- **Integers with the relation "divides by":**
  - We can order 3 and 6 (3 divides 6); we cannot order 4 and 6.

  Integers with the relation ≤ ("less or equal"):
  - We can order any two elements (total ordering).
- **Strings with the prefix relation:**
  - We can order AA and AABC (AA is a prefix of AABC) but not AA and AB.
- **Power set P(C) with subset relation ⊆:**
  - We can order {a,b} and {a,b,c} ({a,b} ⊆ {a,b,c}) but not {a,b} and {a,c}.

Example: VSTa Microkernel

- **Groups in Unix are defined by their group ID and are not ordered.**
- **VSTa uses capabilities to support hierarchies:**
  - VSTa capability is a list of integers $.i_1.i_2. \cdots .i_n$, e.g. .1, .1.2, .1.2.3, .4, .10.0.0.5
- **Abilities are ordered by the prefix relation:**
  - a_2 is a prefix of a_1 (written as $a_2 \leq a_1$) if there exists a_3 s. t. $a_1 = a_2a_3$.
  - The empty string ε is the prefix of any ability.
- **For example: .1 ≤ .1.2 ≤ .1.2.4 but not .1 ≤ .4 !**

Abilities and our Example

- **Assign abilities to groups:**
  - All-students: .3
  - Y1-Students: .3.1
  - CS102-Students: .3.1.101
  - ECE130-Students .3.1.130
- **Label objects with appropriate abilities**
- **Access is given if the object's label is a prefix of the subject's label**
  - CS102-Students have access to objects labeled .3.1.102 or .3.1 or .3 but not to objects labeled .3.1.130

Null Values

- **Consider the dual of the previous policy:**
  - access is granted if the subject's ability is a prefix of the ability of the object.
- **A subject without an ability has access to every object.**
- **Frequent problem: when an access control parameter is missing the policy is not evaluated and access is granted.**
- **NULL DACL problem in Windows:**
  - Nobody has access to a file with an empty ACL but everyone has access to a file with no ACL.

Towards Lattices

- How should we label objects that may be accessed both by CS102-Students and ECE130-Students?
- How should we label a subject that may access resources earmarked for CS102-Students and resources earmarked for ECE130-Students?
- To answer both questions, we need more structure than just partial orderings.

Towards Lattices

- **Assume that a subject may observe an object only if the subject's label is higher than the object's label. We**
- **can ask two questions:**
  - Given two objects with different labels, what is the minimal label a subject must have to be allowed to observe both objects?
  - Given two subjects with different labels, what is the maximal label an object can have so that it still can be observed by both subjects?
- **A lattice is a mathematical structure where both questions have unique 'best' answers.**

Lattice (L,≤) (The slide on lattices you must not memorize)

- **A lattice (L,≤) is a set L with a partial ordering ≤ s.t. for every two elements a,b ∈ L there exists**
  - a least upper bound $u \in L$: $a \leq u$, $b \leq u$, and for all $v \in L$: $(a \leq v \wedge b \leq v) \Rightarrow u \leq v$.
  - a greatest lower bound $l \in L$: $l \leq a$, $l \leq b$, and for all $k \in L$: $(k \leq a \wedge k \leq b) \Rightarrow k \leq l$.
- **Lattices come naturally whenever one deals with hierarchical security attributes.**

System Low & System High

- A label that is dominated by all other labels is called System Low.
- A label that dominates all other labels is called System High.
- System Low and System High need not exist; if they exist, they are unique.
- When L is a finite set, the elements System Low and System High exist.

Lattices - Example 1

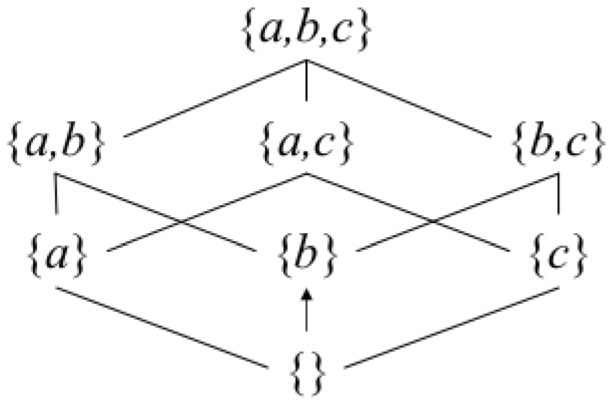
- **The natural numbers with the ordering relation ‘divides by’ form a lattice:**
  - The l.u.b. of a,b is their least common multiple.
  - The g.l.b. of a,b is their greatest common divisor.
  - There exists an element System Low: the number 1.
  - There is no element System High.

Lattices - Example 2

- **The integers with the ordering ≤ form a lattice:**
  - The l.u.b. of a,b is the maximum of a and b.
  - The g.l.b. of a,b is the minimum of a and b.
  - Elements System Low and System High do not exist.
- **The integers with the ordering ≤ are a total ordering.**

Lattices - Example 3

- **P({a,b,c}), ⊆), i.e. the power set of {a,b,c}, with the subset relation as partial ordering:**
  - least upper bound: union of two sets.
  - greatest lower bound: intersection of two sets.



Lines indicate the subset relation.

Summary

- Security terminology is ambiguous.
- Distinguish between access control as a security service and its various implementations.
- Policies expressed in terms of principals and objects.
- In identity-based access control, users are principals.
- Deployed in practice: RBAC, ACLs to a minor extent.
- More sophisticated policies draw you into mathematics.
- We have covered ‘classical’ access control; we return to current trends later.