DSCI 519: Foundations and Policy for Information Security

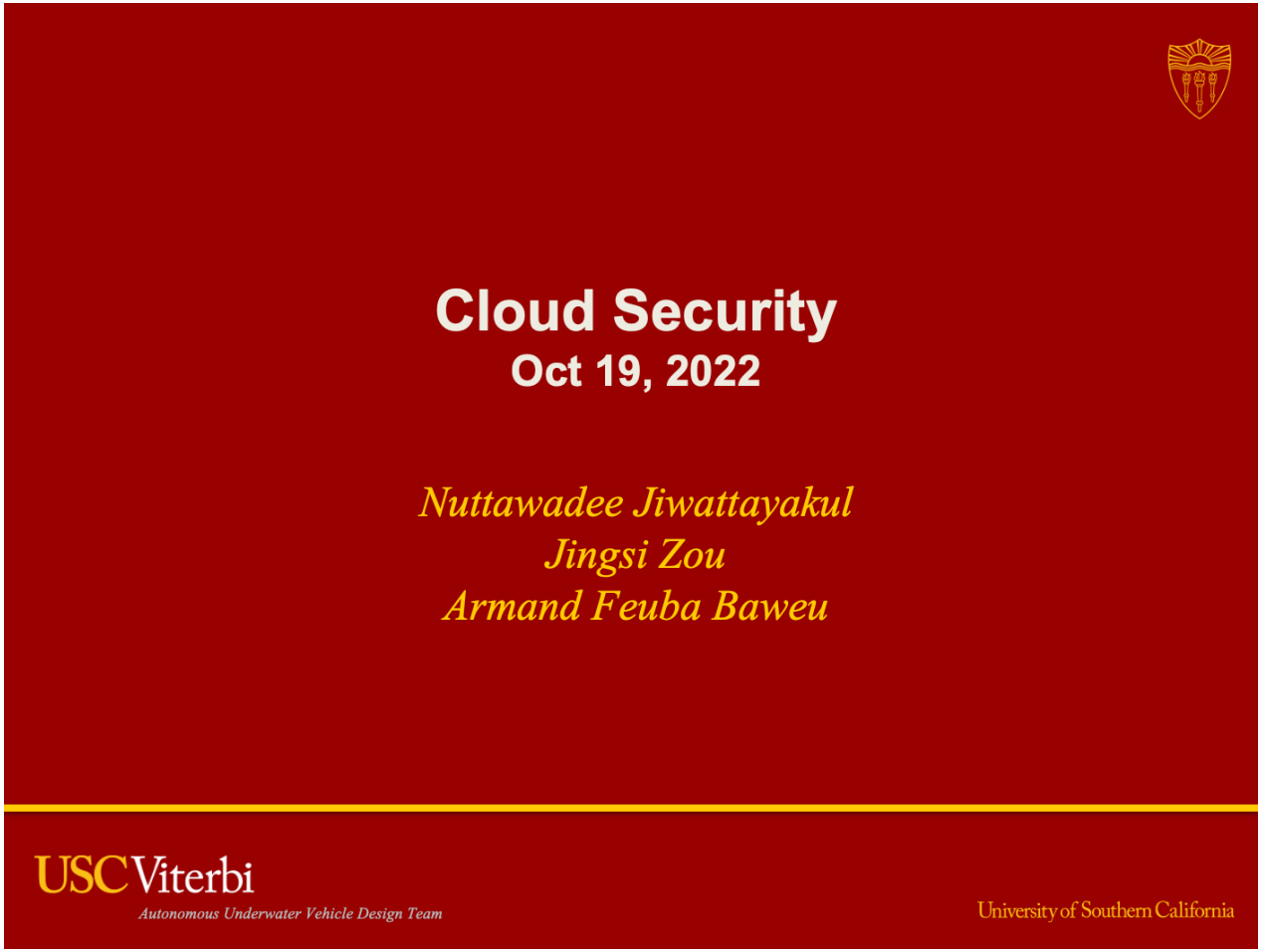
Hybrid Policy Models

Tatyana Ryutov

Outline

- Review
- Lipner Model
- Clark-Wilson Model
- Chinese Wall Model
- Role based access control
- Attribute based access control

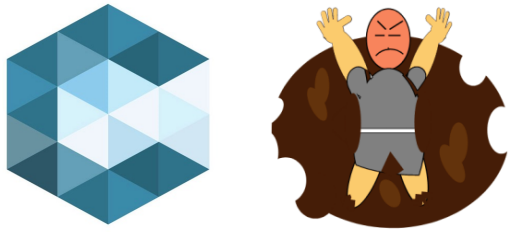
Presentation9



Presentation10

USC Viterbi
School of Engineering

L8.Q6



- Think about all material covered in class today. Identify:
  1. Points that are “crystal clear”
  2. The “muddy points”

Your Questions

- Call brackets please
- If we can go over Access Brackets for Segments again, please
- Can we review the protection rings in Multics again, it was towards the end of lecture and the concepts started to blend together
- The bracket part
- Can we do more practice like Q4?

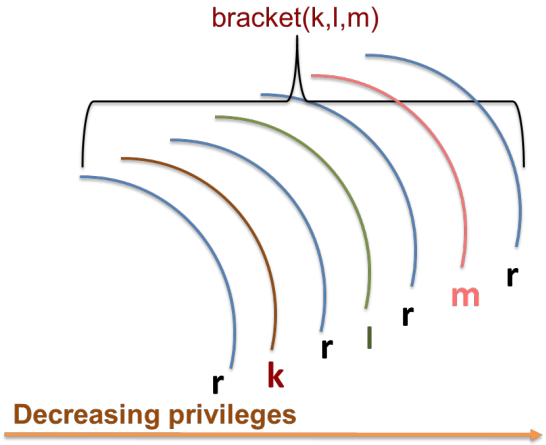
Review: Ring Policy

- Complete bracket is (*k, l, m*):
  - (k,l) – access bracket
  - (l,m) – call bracket
- For ring numbers k, l, m:
  - r < k: w/r/e permitted, for e (execute) ring fault occurs; set *r* = *k*

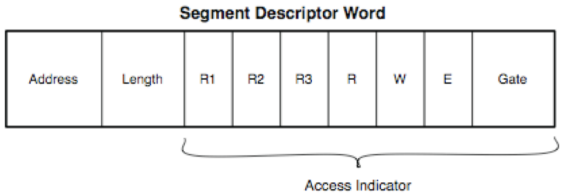
Ring policy for data segments ensures that lower rings have strictly greater access to segments than higher rings

  - r = k: w/r/e permitted
  - k < r ≤ l: r/e permitted; w denied
  - l < r ≤ m: e permitted through gate, transition to more privileged ring
  - m < r: all access denied

Ring policy for procedure segments: do not want to execute lower integrity in more privileged ring, change the current ring



If I have a procedure bracket of (2, 50, 100) and r = 55, what would be an example of an approved gate to be able to execute in the more privileged ring (2), or does it exist in this case?

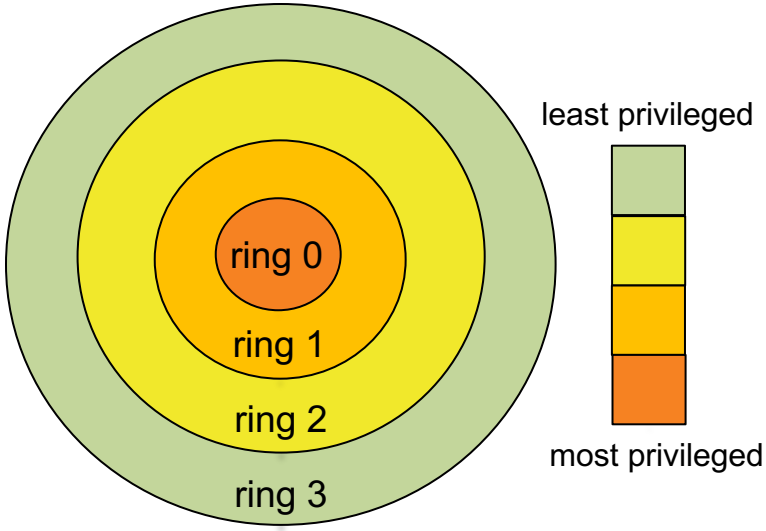


USC Viterbi School of Engineering

Your Questions

- In reference to Q3 part 1 of this lecture, would there be any benefit to placing subjects/objects in their own hardware rings?

1. How can HW rings help support RM implementation?



Outline

- Review
- Lipner Model
- Clark-Wilson Model
- Chinese Wall Model
- Role based access control
- Attribute based access control

Requirements of Commercial Integrity Policies (Lipner 1982)

1. Users will not write their own programs, but use existing production software
2. Programmers develop and test applications on a non-production system using contrived data
3. Moving applications from development to production requires a special process
4. This process must be controlled and audited
5. Managers and auditors must have access to system state and system logs

Principals of Operation

- Separation of duty:
  - If two or more steps are required to perform a critical function, at least two people should perform the steps
  - E.g., a different person installs program on a production system
- Separation of function:
  - Do not use production system for development
  - Do not process production data on development system
- Auditing:
  - Commercial systems emphasize recovery and accountability
  - Ability to analyze who did what

Lipner's Lattice (BLP + Biba)

- A realistic example showing that BLP and Biba can be combined to meet commercial requirements
- How does it combine BLP and Biba?
  - Uses disjoint sets of security levels and integrity levels
  - BLP is used first, and add Biba only when necessary

Lipner's Lattice (BLP + Biba)

- BLP component for confidentiality:
  - 2 security clearances/classifications
    - AM (Audit Manager): system audit, management functions
    - SL (System Low): everything else, any process can read at this level
  - 3 Security categories
    - SP (Production): production code, data
    - SD (Development): production programs under development and testing, but not yet in production use
    - SSD (System Development): system programs under development
  - Security level = (classification, category)
- Biba component for integrity:
  - 3 integrity classifications
    - ISP (System Program): for system programs
    - IO (Operational): production programs, development software
    - ISL (System Low): users get this on log in
  - 2 integrity categories
    - ID (Development): development entities
    - IP (Production): production entities
  - Integrity level = (classification, category)

Subjects' Levels (Clearance)

Subjects	Security Level	Integrity Level
Ordinary users	(SL, { SP })	(ISL, { IP })
Application developers	(SL, { SD })	(ISL, { ID })
System programmers	(SL, { SSD })	(ISL, { ID })
System managers and auditors	(AM, { SP, SD, SSD })	(ISL, { IP, ID})
System controllers	(SL, { SP, SD }) and downgrade privilege	(ISP, { IP, ID})
Repair	(SL, { SP })	(ISL, { IP })

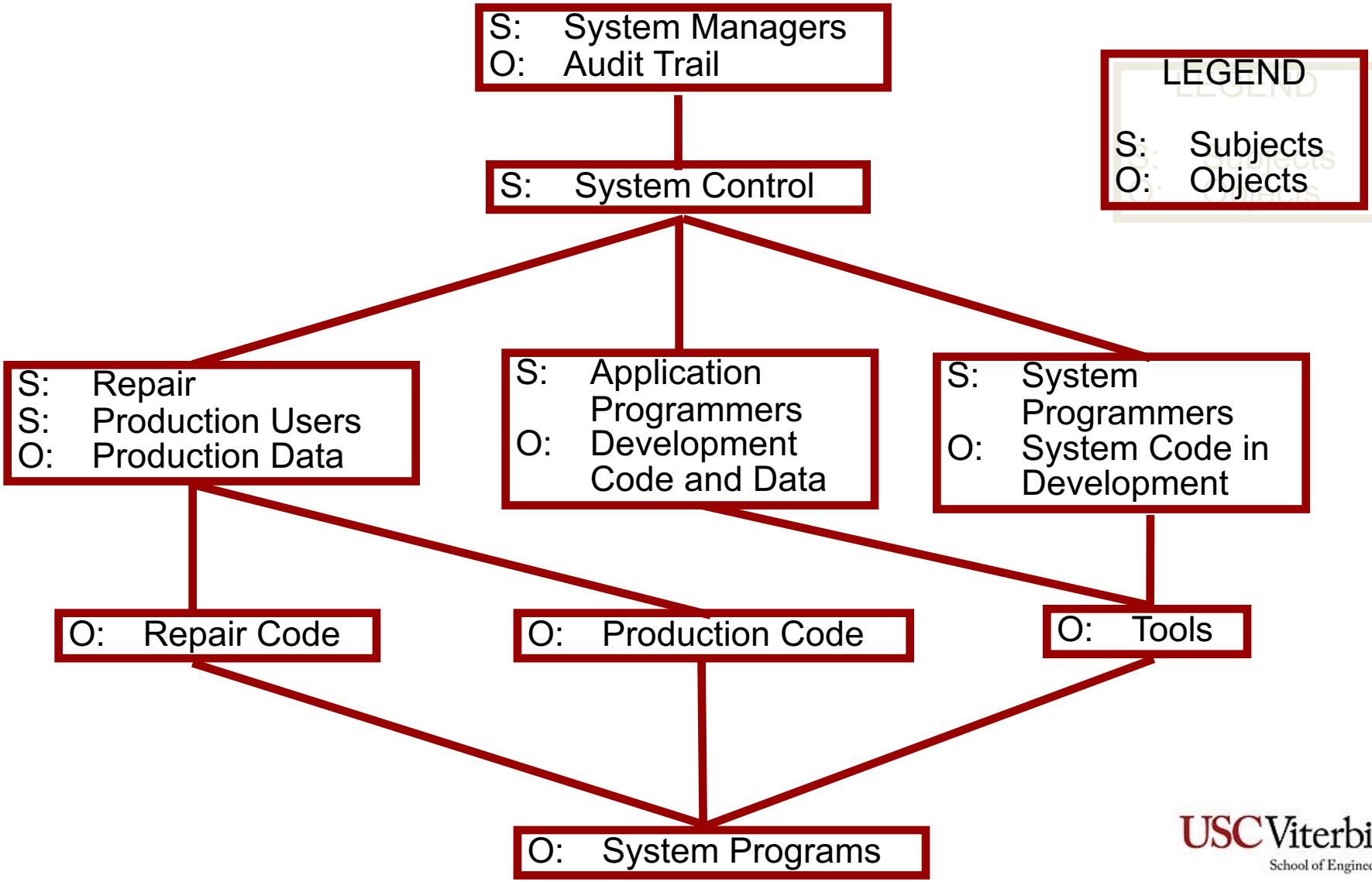
Exempt from *-property: allowed to write down (with respect to confidentiality) and write up (with respect to integrity)

Objects' Levels (Classification)

Objects	Security Level	Integrity Level
Development code/test data	(SL, { SD })	(ISL, { IP})
Production code	(SL, { SP })	(IO, { IP })
Production data	(SL, { SP })	(ISL, { IP })
Software tools	(SL, ∅)	(IO, { ID })
System programs	(SL, ∅)	(ISP, { IP, ID })
System programs in modification	(SL, { SSD })	(ISL, { ID })
System and application logs	(AM, { *appropriate* })	(ISL, ∅)
Repair	(SL, {SP})	(ISL, { IP })

The position of system logs at lowest integrity demonstrates the limitation of info flow policy for integrity

The Lipner Lattice



What does the Lipner Lattice achieve?

- Ordinary users can execute (read) production code but cannot alter it
- Ordinary users can alter and read production data
- System managers need access to all logs but cannot change levels of objects
- System controllers need to install code (hence downgrade capability)
- Subjects need to have append-only access to logs
- These meet the stated requirements

Example: What can an ordinary user do?

- Ordinary users **(SL, { SP }) (ISL, { IP })** can:
  - Read and write production data **(SL, { SP }) (ISL, { IP })**
    - same security integrity levels
  - Read production code **(SL, { SP }) (IO, { IP })**
    - same classification and (ISL, {IP}) *leq* (IO, { IP })
  - Read system program **(SL, ∅) (ISP, { IP, ID })**
    - (SL, {SP}) *dom* (SL, ∅) and (ISL, {IP}) *leq* (ISP, {IP,ID})
  - Repair objects **(SL, { SP }) (ISL, { IP })**
    - same levels
  - Write (not read) system and application log **(AM, { SL }) (ISL, ∅)**
    - (AM, {SP}) *dom* (SL, {SP}) and (ISL, ∅) *leq* (ISL, {IP})

Outline

- Review
- Lipner Model
- Clark-Wilson Model
- Chinese Wall Model
- Role based access control
- Attribute based access control

Clark-Wilson Integrity Model,1987

- Time-proven accounting practices generalized
- Goal: prevent illegal data modification due to fraud and error
- Validation of integrity is done to ensure that:
  - The data is being modified is valid
  - The results of the modification are valid
- Integrity policy is given as high-level rules: a set of constraints
  - Data in a consistent state when it satisfies the constrains
- Example: Bank
  - Objective: today's deposits - today's withdrawals + yesterday's balance = today's balance

Clark-Wilson (CW): Two mechanisms for enforcing Integrity

1. Well-formed transaction
   - Operations that move system from one consistent state to another
     - State before transaction consistent $\Rightarrow$ state after transaction consistent
   - Can manipulate data only through trusted code!
   - Sufficient for ensuring internal consistency, but insufficient for ensuring consistency with physical world
2. Separation of duty
   - Ensure external consistency: data objects correspond to the real world objects
   - Separating all operations into several subparts and requiring that each subpart be executed by a different person
   - E.g., the two-man rule

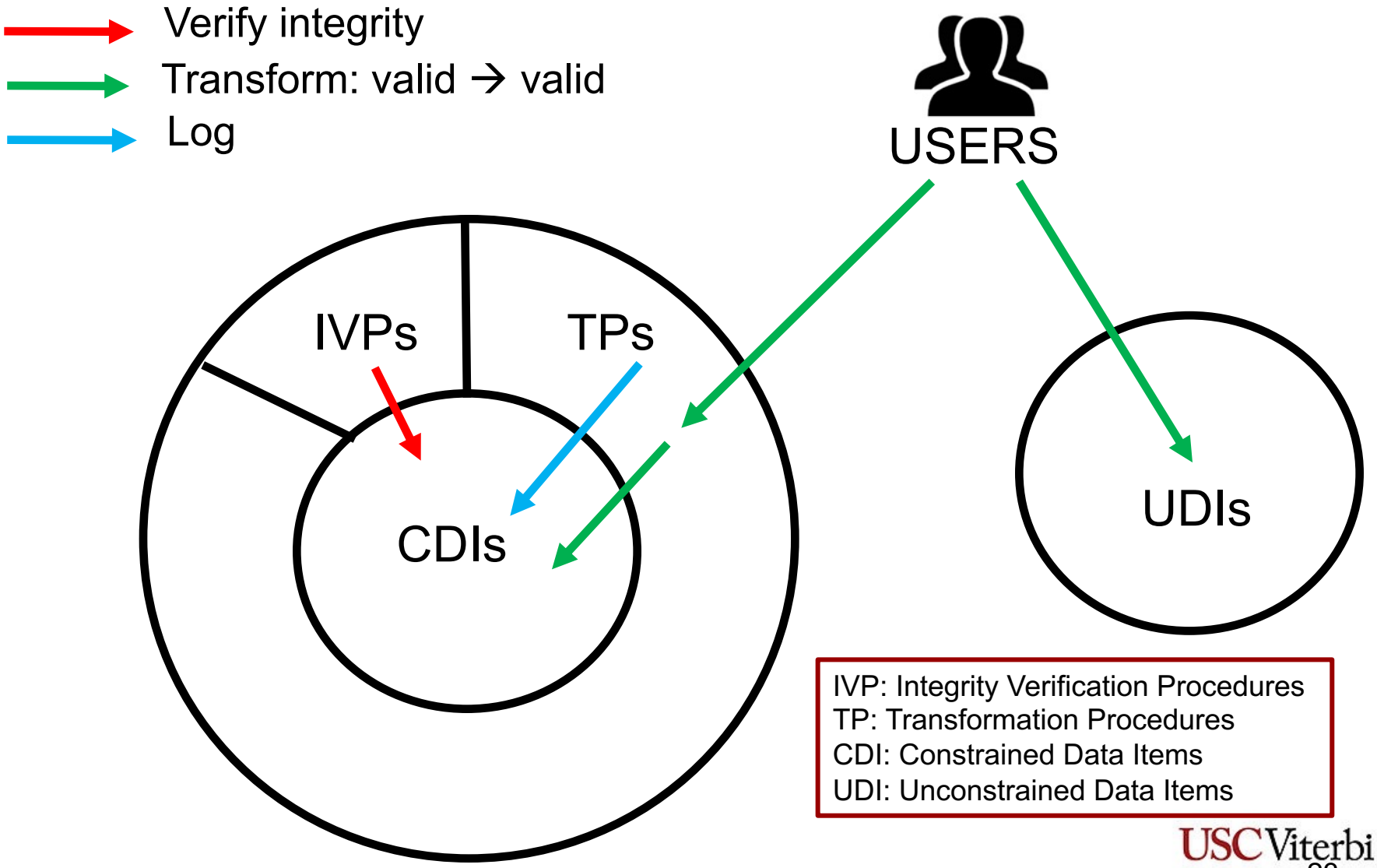
Implementing the Two High-level Mechanisms

- Mechanisms are needed to ensure
  1. controlled access to data: a data item can be manipulated only by a specific set of programs
  2. program certification: programs must be inspected for proper construction, controls must be provided on the ability to install and modify these programs
  3. controlled access to programs: each user must be permitted to use only certain sets of programs
  4. controlled administration: assignment of people to programs must be controlled and inspected

CW Model Elements

- Users – active agents
- **T**ransformation **P**rocedures (**TP**)
  - abstract operations, e.g., debit, credit
  - implement well-formed transactions
- **C**onstrained **D**ata **I**tems (**CDI**): data subject to integrity control
  - Can only be manipulated by Transformation Procedures
- **U**nconstrained **D**ata **I**tems (**UDI**): data not subject to integrity controls
  - Can be manipulated by via primitive read and write operations
- **I**ntegrity **V**erification **P**rocedures (**IVP**)
  - Test CDIs' conformance to integrity constraints at the time IVPs are run
    - E.g., checking account balances

CW Elements Interactions



Rules at a Glance

Certification Rules

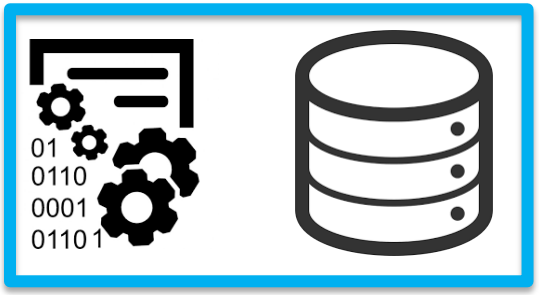
C1 IVPs verify CDI integrity
C2 TPs preserve CDI integrity
C3 Separation of duties for ER2
C4 TPs write to log
C5 TPs upgrade UDIs to CDIs

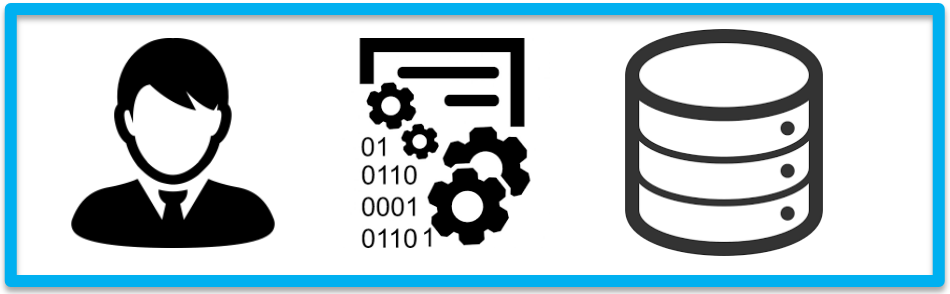
Enforcement Rules

E1 CDIs changed only by authorized TP
E2 TP run only by authorized users
E3 Users are authenticated
E4 Authorizations changed only by certifiers

CW: Certification/Enforcement Rules

- **C1**: When any IVP is run, it must ensure all CDIs are in valid state
- **C2**: A TP must transform a set of CDIs from a valid state to another valid state
- **E1**: System must maintain certified relations
  - TP/CDI mapping enforced
  - TP must not be used on CDIs it is not certified for
- **E2**: System must control users
  - user/TP/CDI mappings enforced
  - the model must account for the person performing the TP





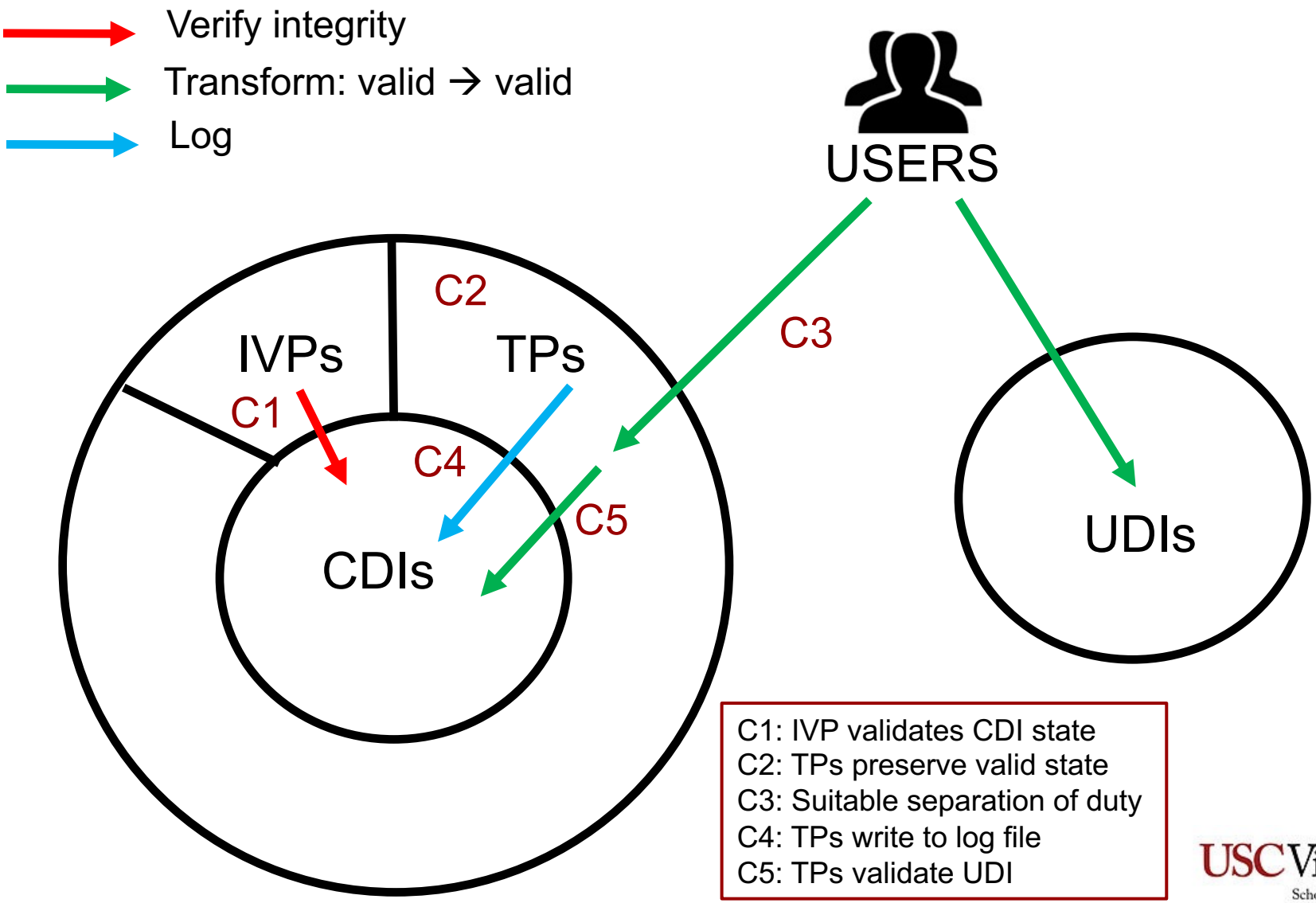
IVP: Integrity Verification Procedures
TP: Transformation Procedures
CDI: Constrained Data Items
UDI: Unconstrained Data Items

CW: Certification/Enforcement Rules

- **C3**: Relations between (user, TP, {CDI}) must support separation of duty
- **E3**: Users must be authenticated to execute TP
- **C4**: All TPs must log undo information to append-only CDI (to reconstruct an operation for review)
  - Log is CDI
- **C5**: A TP taking a UDI as input must either reject it or transform it to a CDI
  - information entering a system need not be trusted
- **E4**: Only certifier of a TP may change the list of entities associated with that TP
  - No certifier of a TP or CDI associated with that TP, may have execute permission on the TP/CDI
  - enforces the separation of duty

IVP: Integrity Verification Procedures
TP: Transformation Procedures
CDI: Constrained Data Items
UDI: Unconstrained Data Items

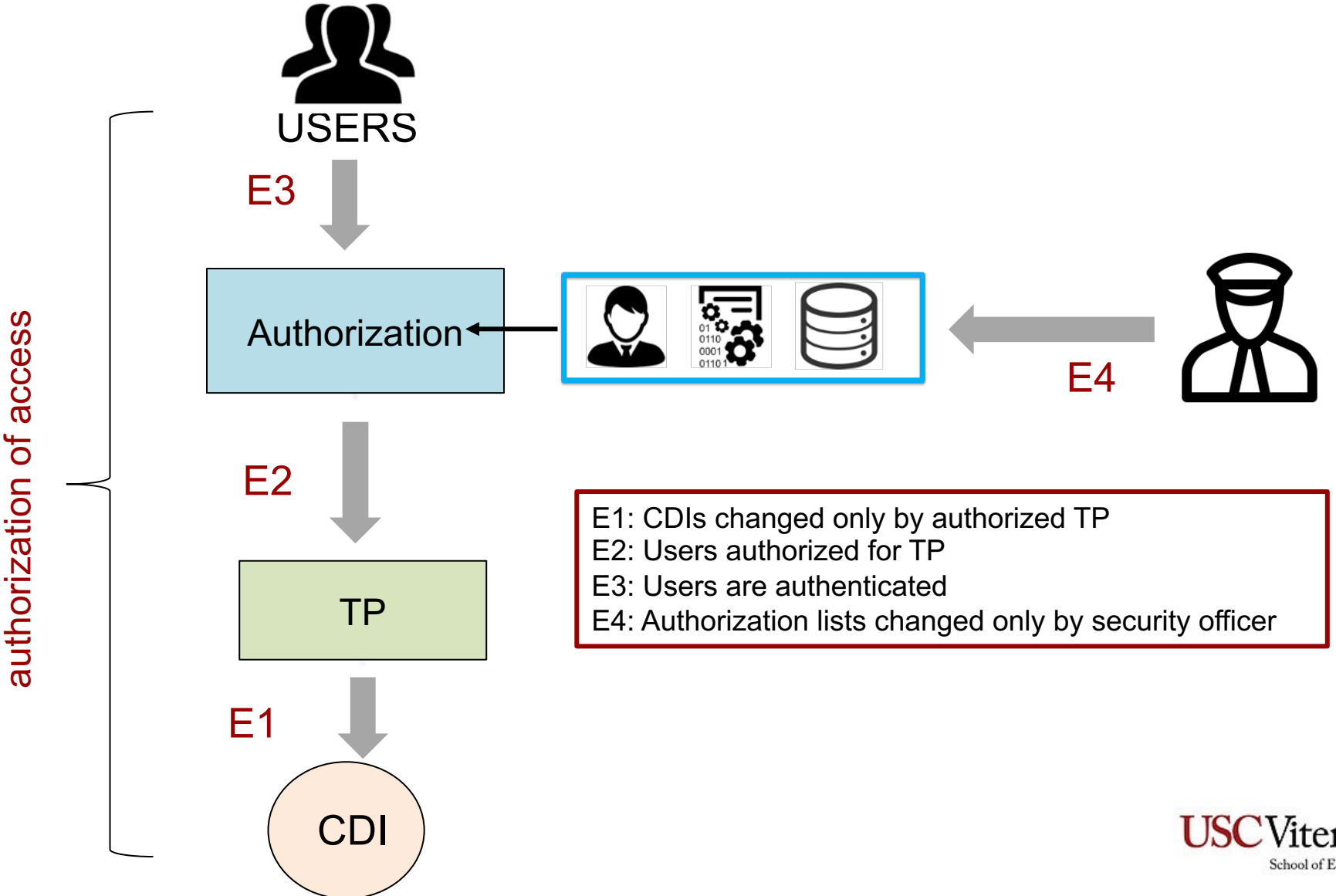
CW Elements and Certification Rules



C1: IVP validates CDI state
C2: TPs preserve valid state
C3: Suitable separation of duty
C4: TPs write to log file
C5: TPs validate UDI

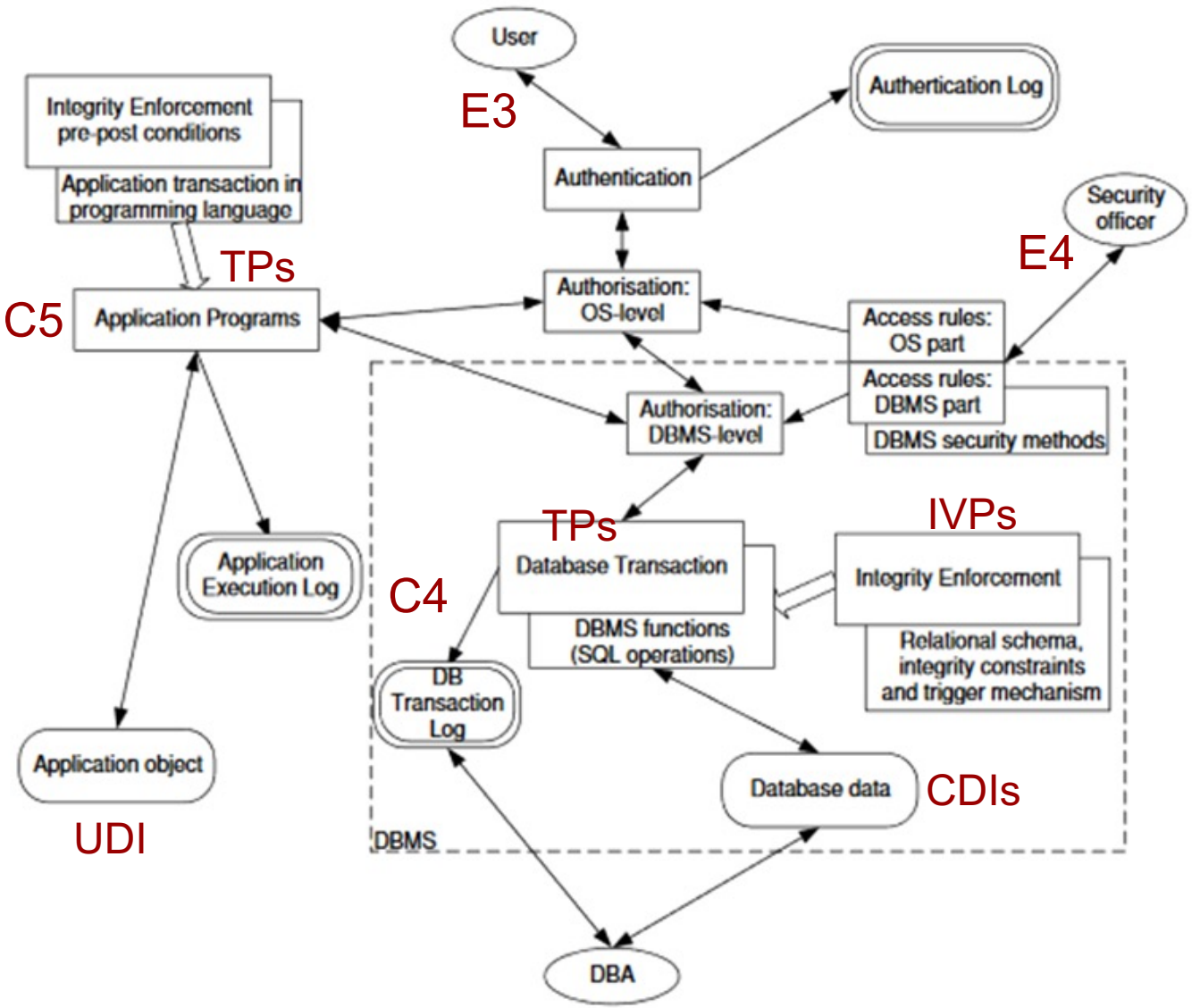
USC Viterbi School of Engineering

CW Elements and Enforcement Rules



Example: Applying CW to a DBMS

C4: TPs write to log file
C5: TPs validate UDI
E3: Users are authorized
E4: Authorization lists changed only by security officer



USC Viterbi School of Engineering

TP in practice: Transaction Concept

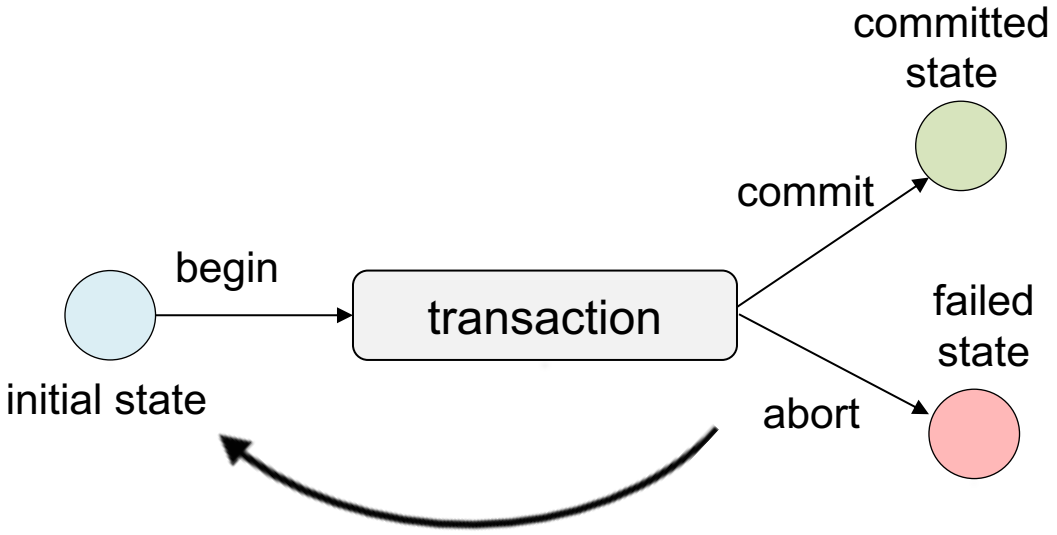
- A transaction is a set of atomic updates to the state of one or more objects
- Transaction can finish in one of two states:
  - **Committed**: If a transaction commits (succeeds) then the new state of the objects will be observed
    - i.e. all updates occur
  - **Rollback (**or **abort)**: If a transaction rolls back (fails) then the object will remain in its original state (as if no updates to any part of the state were made)
    - i.e. no updates occur

```
void threadTask(void* arg)
{
/* Do local computation */
/* checkpoints/saves state */

begin_transaction(val1,val2) {

/* Do some computation/updates */
val1 -= amount;
val2 += amount;
} // end_transaction

abort {
// restore/re-read val1, val2
// release locks
// restart
 }
}
```



USC Viterbi School of Engineering

ACID Properties

- Transactions help achieve the ACID properties:
  - Atomicity: update appears as indivisible (all or nothing), no partial updates are visible (at logical level)
    - At physical level, only single disk/flash write is atomic
  - Consistency: old state and new, updated state meet certain necessary invariants
    - E.g., no orphaned blocks, etc.
  - Isolation: other transactions do not see results of earlier transactions until they are committed
    - Serializability (transaction T1 appears to execute entirely before T2 or vice versa)
  - Durability: committed updates are persistent

Integrity Policy Models: Key Points

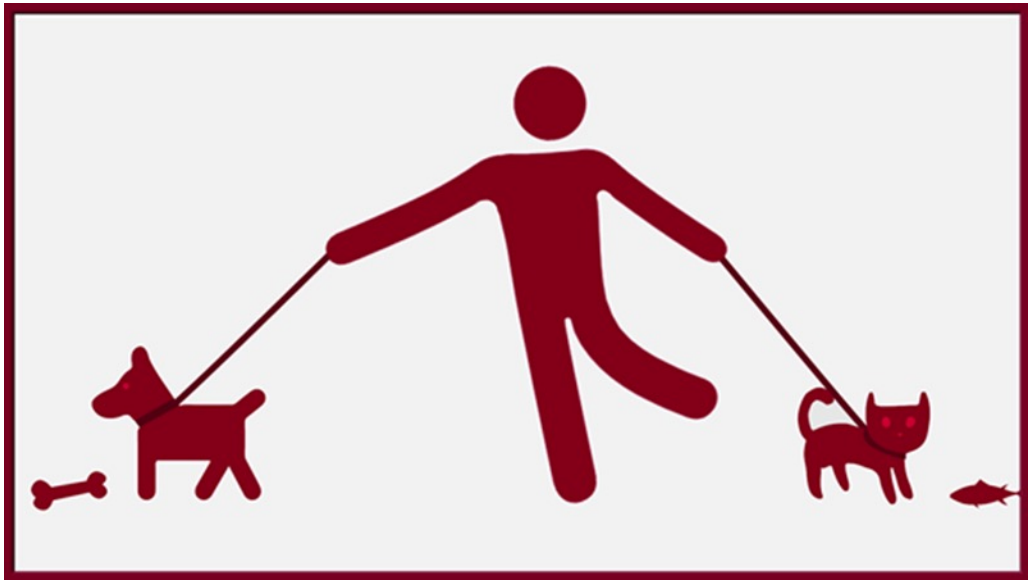
- Commercial world needs integrity
- Biba and Lipner models are based on multilevel integrity
  - Biba model
    - Dual of BLP (or BLP-upside-down)
      - Integrity levels distinct from security levels
      - Information flows differently
    - Can be combined with BLP
  - Lipner's lattice combines the two to meet commercial requirements
    - Pros & cons
  - Integrity problem is more than just restricting information flow
    - For example: concurrency control, integrity constraints, etc.
- Clark-Wilson model
  - Introduces new ideas
    - Enforces integrity by using well-formed transactions (through access triple), separation of user duties, and auditing

Outline

- Review
- Lipner Model
- Clark-Wilson Model
- Chinese Wall Model
- Role based access control
- Attribute based access control

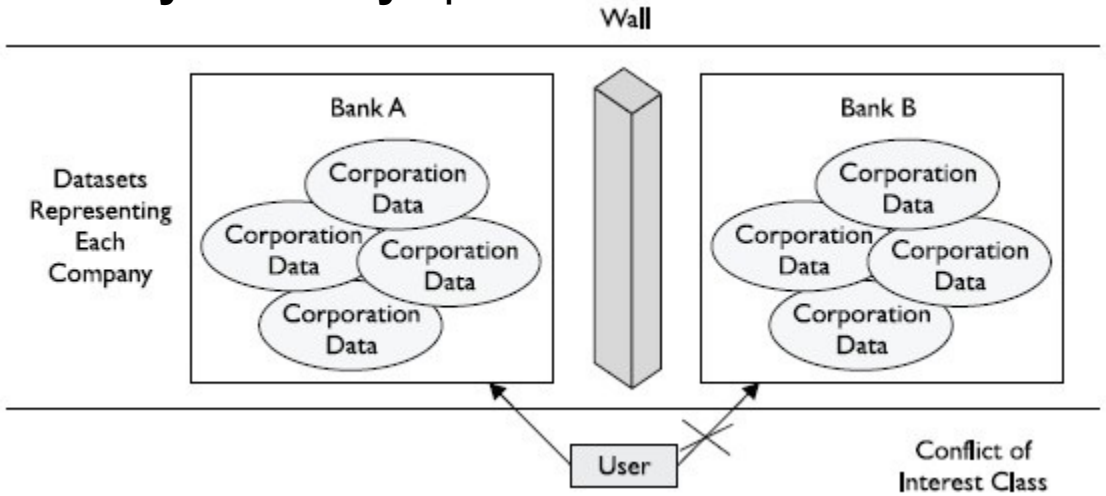
Problem Statement

- A policy for a specific commercial concern:
  - Potential for conflicts of interest and disclosure of information by a consultant or contractor
- Example:
  - A lawyer specializes in product liability and consults for American Airlines
  - Could it be a breach of confidentiality for her to consult also for United Airlines?



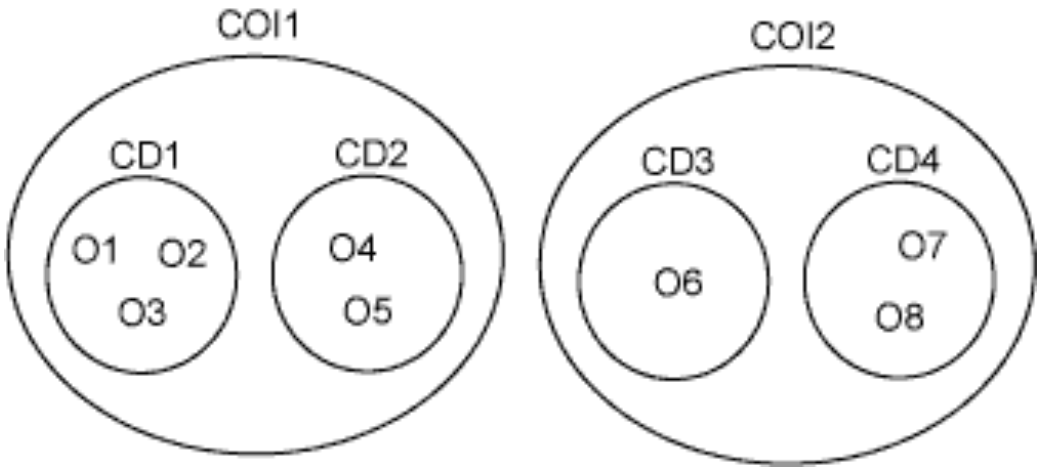
Chinese Wall (ChW) Model Introduction

- Developed by Brewer and Nash in 1989
- A real commercial policy which can be **formally modelled**
  - Addresses confidentiality and integrity
- Basic idea is to prevent conflict of interest:
  - Build a set of Chinese Walls among company datasets
  - Set of rules such that no person (subject) can access data (objects) on the wrong side of that wall
- E.g., UK Stock Exchange: prevent traders to represent clients with conflict interest
- Distinguished from Bell-LaPadula
  - Rights are **dynamically** updated based on actions of the subjects



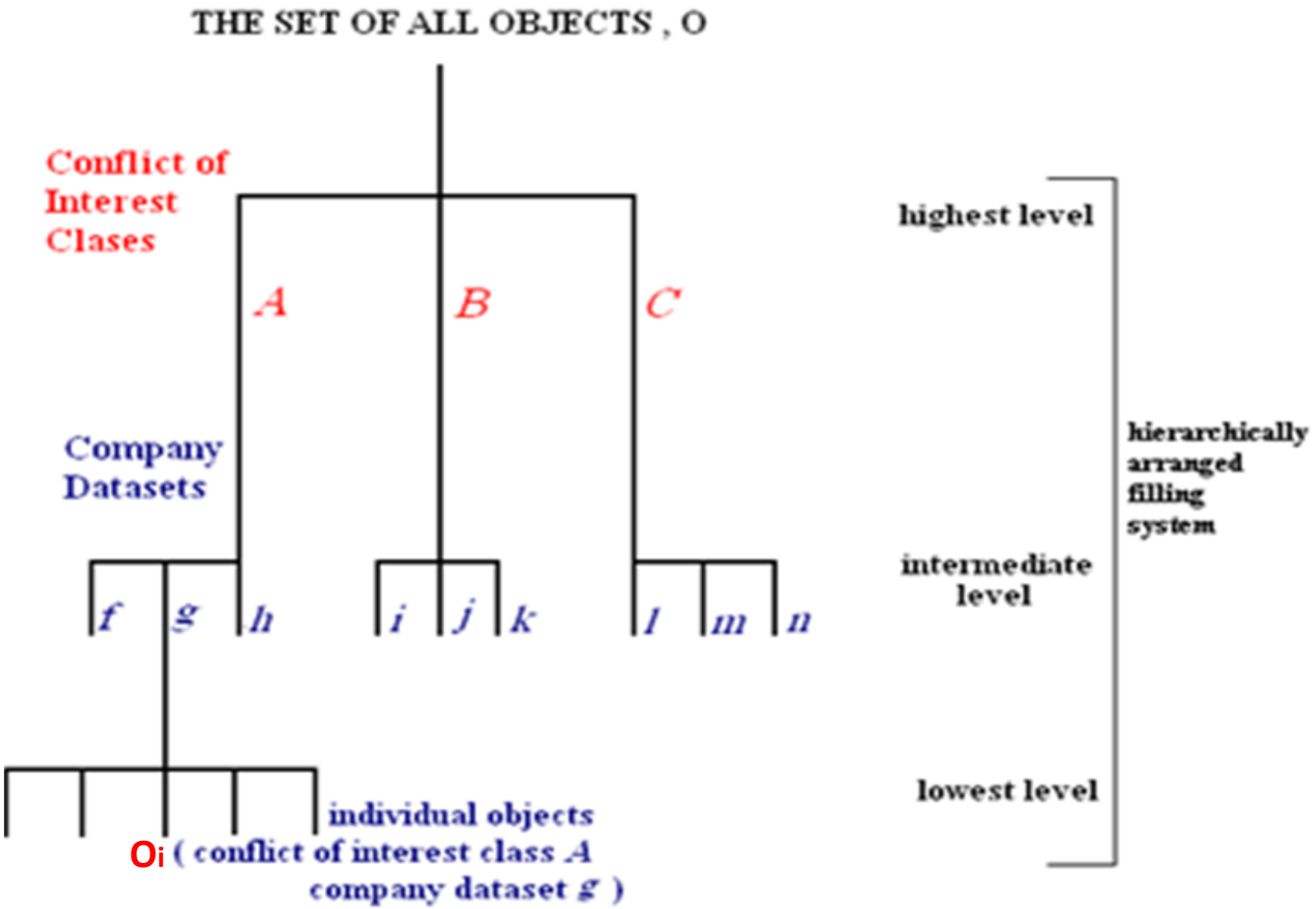
Chinese Wall (ChW) Model Elements

- All corporate information is stored in hierarchically arranged levels (lowest to highest):
  1. **Objects** - individual items of information, each concerning a single corporation
  2. All objects which concern the same corporation are grouped into a **company dataset (CD)**
  3. All company datasets whose corporations are in competition are grouped together. Each group is referred as a **conflict of interest class (COI)**
- Assumption: each object belongs to exactly one COI class



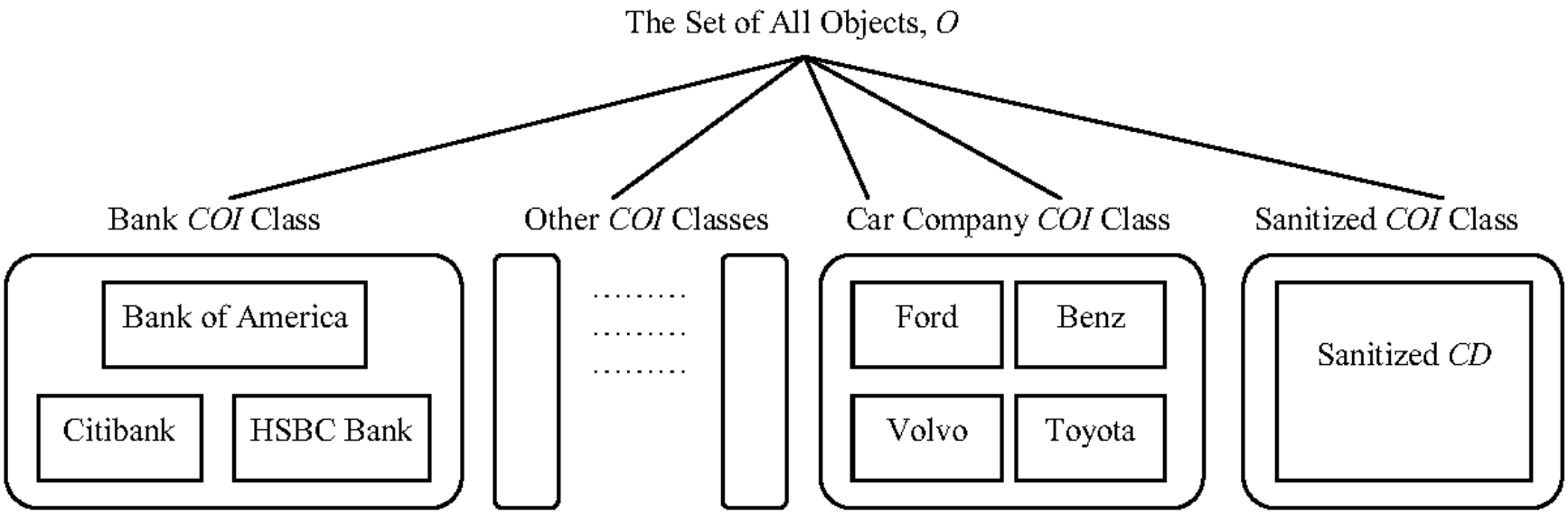
ChW Information Structure

- Associated with each object:
  - the name of the company dataset (CD) to which it belongs
  - the name of the conflict of interest class (COI) to which that company dataset belongs



ChW: Example

- Sanitized CD – only public info contained within a CD

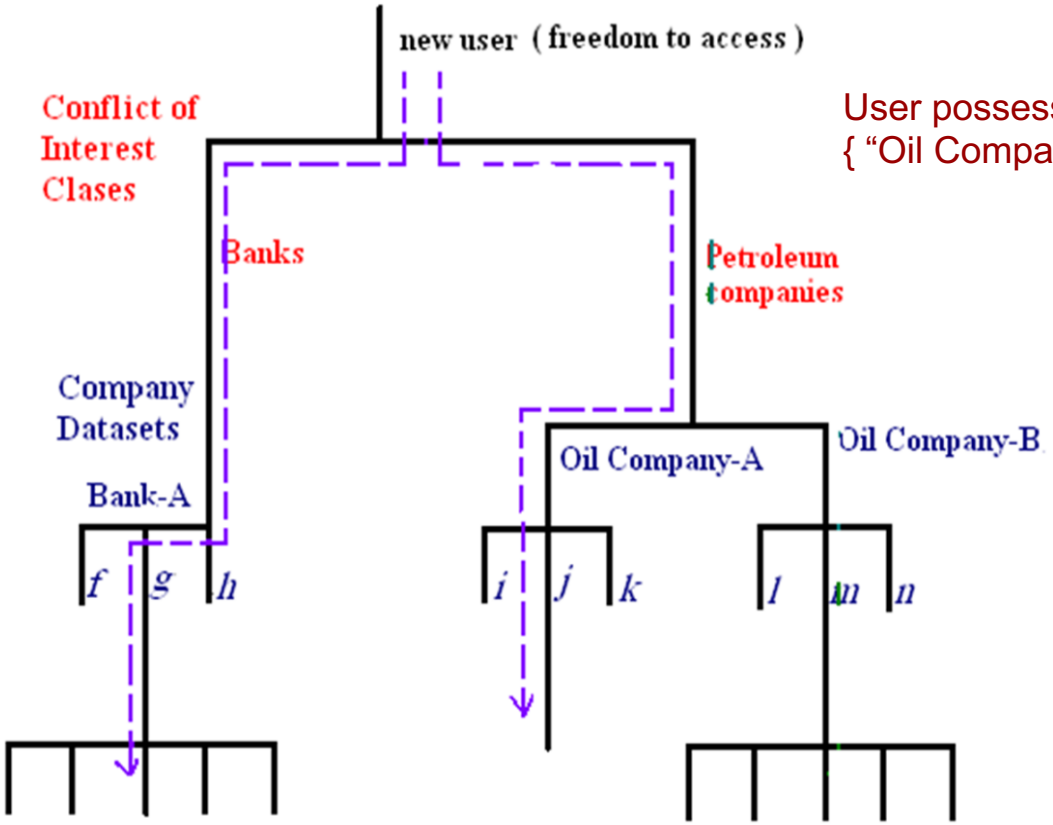


ChW: Simple Security Property

- Subjects are only allowed access to information which is not in conflict with any other information that they already possess
- A subject may access information from any company as long as that subject has never accessed information from a different company in the same conflict class
- Permissions change dynamically
  - The access rights that any subject has depend on the history of past accesses

ChW Model: COI Example 1

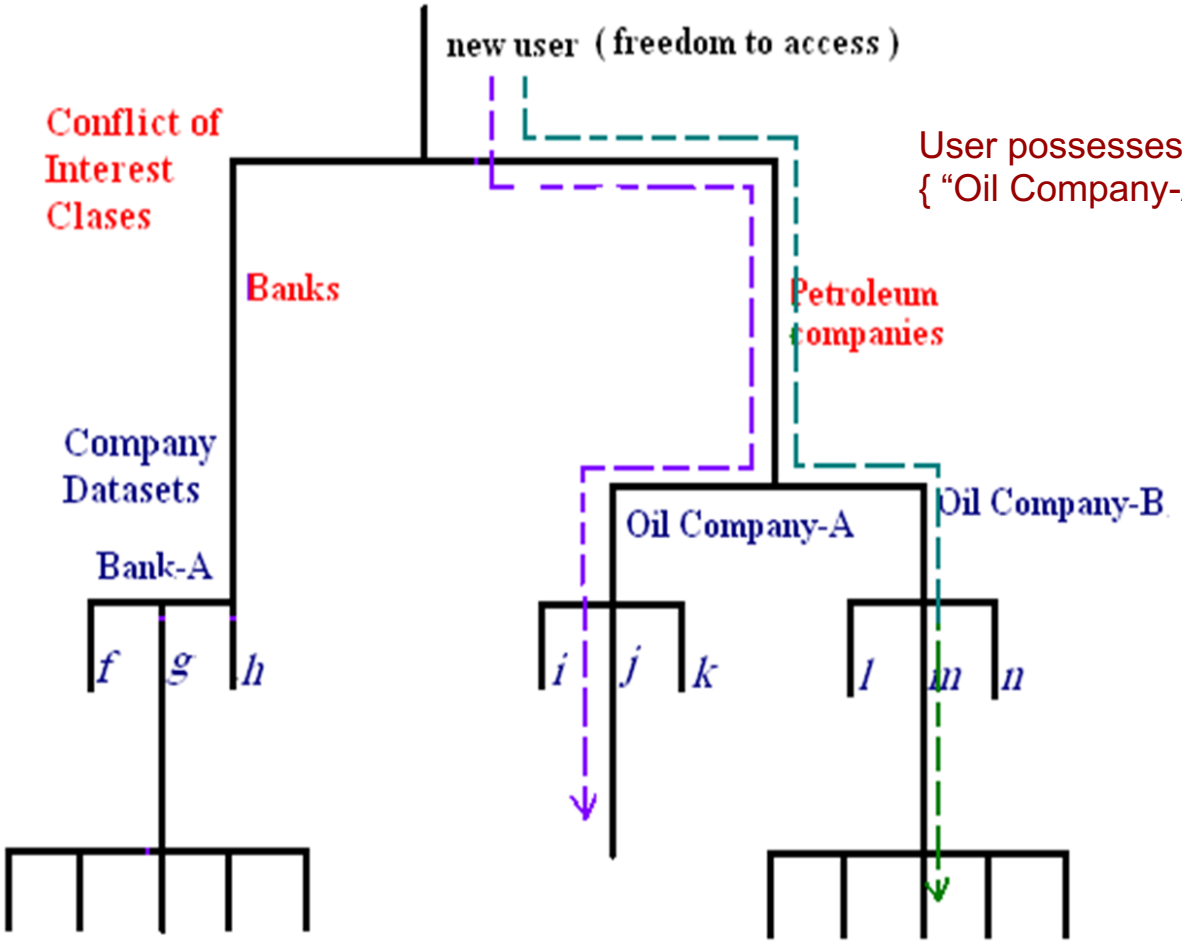
- Suppose a user accesses the Oil Company-A dataset first
  - The user now possesses information concerning the oil company-A dataset
- Later, the user requests access to Bank-A dataset
  - This is allowed since Bank-A and Oil company-A datasets belong to different conflict of interest classes and therefore no conflict exists



User possesses:
{ "Oil Company-A" , "Bank-A" }

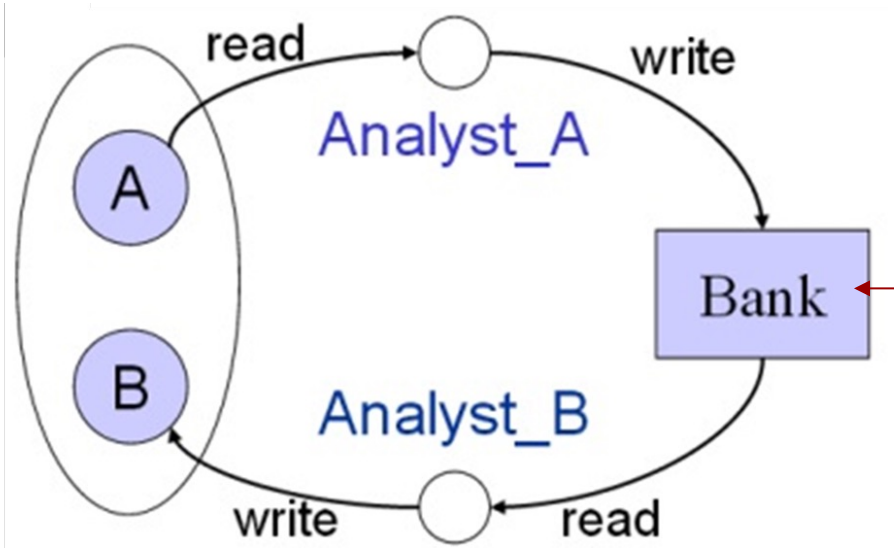
ChW Model: COI Example 2

- The user requests access to oil company-B dataset
  - The request must be denied since a conflict does exist between the requested dataset (Oil Company-B) and one already possessed (Oil Company-A)



ChW: Indirect Information Flow

- Sanitized object is publicly available, no conflicts of interest arises from it
  - So, should not affect read
  - But it does affect write



Conflict Of Interest

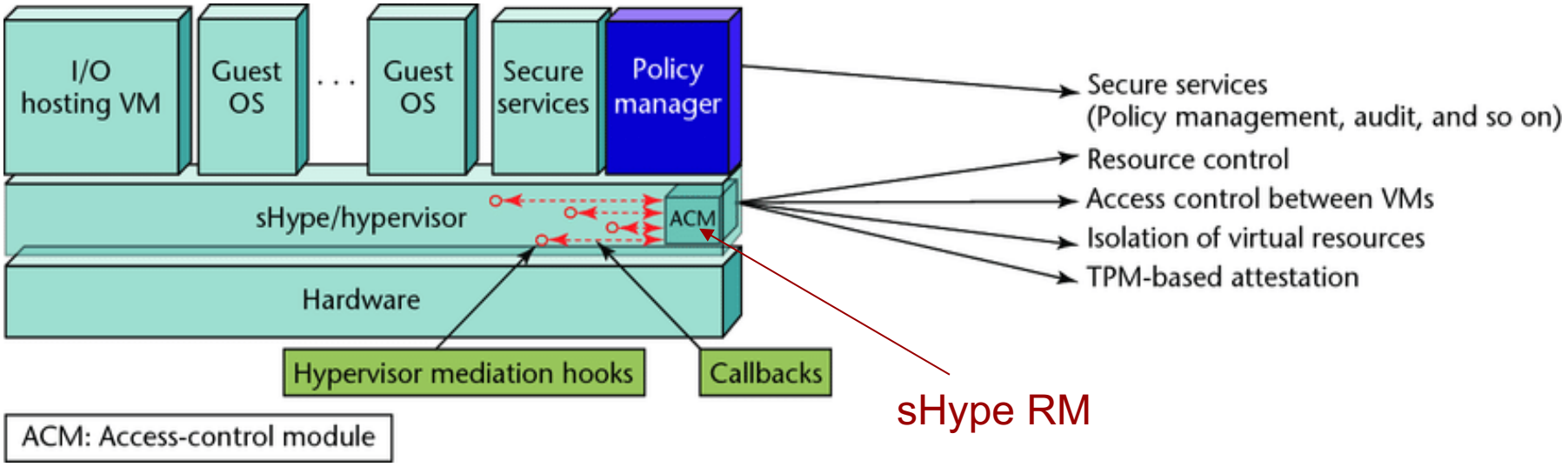
Sanitized object – only public info contained within a CD

ChW Model Rules

- Formally, CW policy restricts access according to the two properties:
  - Simple Security Rule: subject *s* can **read** object *o* only if:
    1. object *o* is in the same company datasets as all objects already accessed by *s*, that is, "within the Wall," **OR**
    2. object *o* belongs to an entirely different conflict of interest class
  - *-property: subject *s* can **write** object *o* only if:
    1. subject *s* can read *o* by simple security rule, **AND**
    2. no object *o'* can be read which is in a **different** company dataset than the one for which write access is requested
- What is the implication of *-property?
  - Either subject *s* cannot write at all OR
  - Subject *s* is limited to reading and writing one company dataset

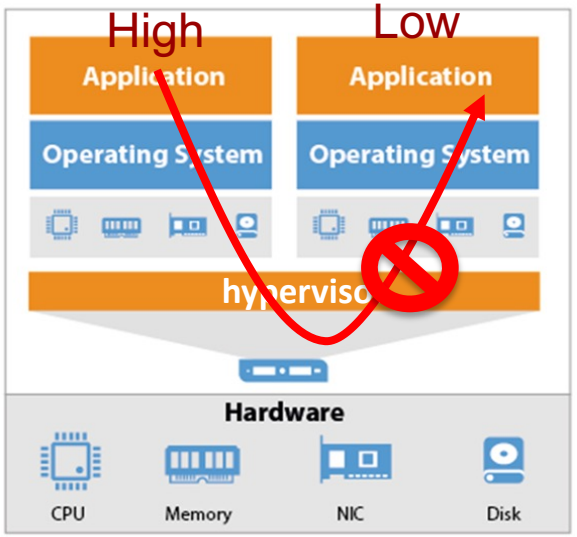
sHype Security Architecture

- The Secure Hypervisor (sHype) is a hypervisor security architecture developed by IBM Research
  - https://researcher.watson.ibm.com/researcher/view_group.php?id=2849
- Enforces a formal security policy (MAC) on information flow between VMs
- Chinese Wall security model is applied in sHype for managing the risk due to potential covert channels

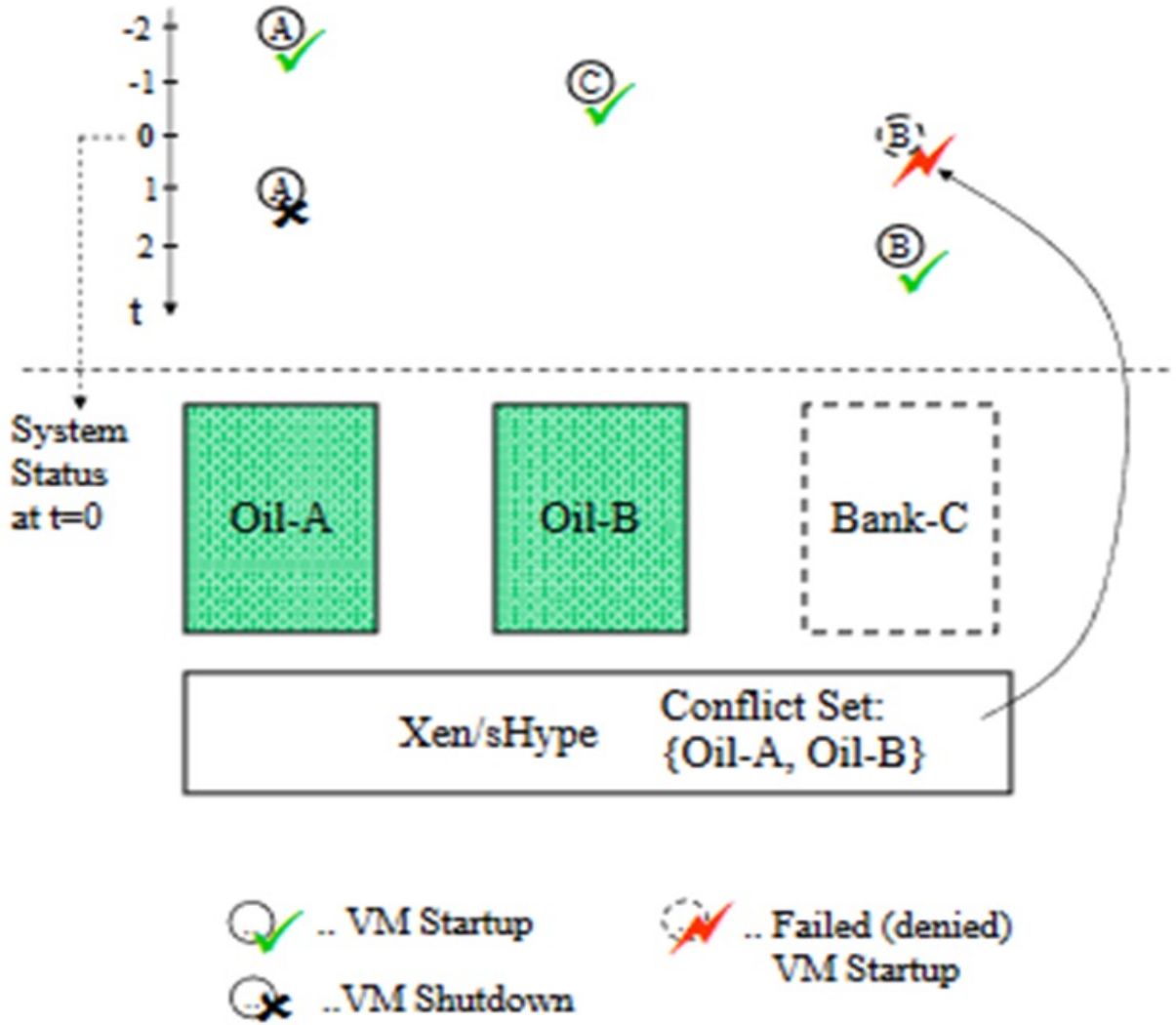


VM Covert Channel Example

- Two VMs could establish a covert channel based on the CPU quantum that each VM received
- System has two VMs
  - Sending machine High, receiving machine Low
  - Round robin scheduling
- To send:
  - For 0, High immediately relinquishes CPU
    - For example, run a process that instantly blocks
  - For 1, High uses full CPU quantum
    - For example, run a CPU-intensive process
- R measures how quickly it gets CPU
  - Time between access to shared resource (CPU)



Example: Xen sHype Chinese Wall Model



Details: https://dominoweb.draco.res.ibm.com/reports/rc24154.pdf

ChW vs. BLP

- Both models control information flow
  - CW: no info flow is allowed that causes COI
- CW is based on access history, BLP is history-less
- BLP can capture CW state at any time, but cannot track changes over time
- BLP security levels would need to be updated each time an access is allowed
  - Example:

Initial user BLP clearance: { “Oil Company-A“ , “Oil Company-B” }

Outline

- Review
- Lipner Model
- Clark-Wilson Model
- Chinese Wall Model
- Role based access control
- Attribute based access control

Principle of Least Privilege

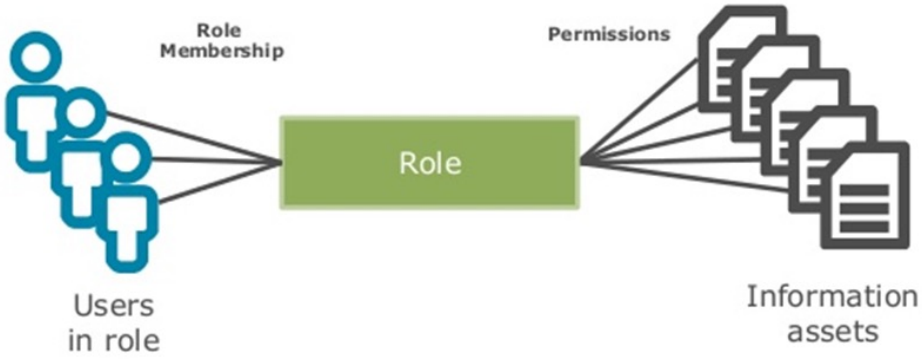
- Each program and user of a computer system should operate with the bare **minimum privileges necessary** to function properly (necessary to complete the job)
- Limits the damage that can result from an accident or error
- The HIPAA Privacy Rule provides guidelines for the establishment of least privilege, such as restricting access to data (i.e. a subset of a patient record as opposed to the entire record) base on the “minimum necessary use” to accomplish a specific purpose

Principle of Separation of Duties (SOD)

- Separation/Segregation of duties (SOD) aims at prevention of fraud and errors
- This objective is achieved by distributing the tasks and associated privileges for a specific business process among multiple users
- Duties – classes or types of operations
- Examples of duties that should be segregated:
  - Authorization or approval of related transactions affecting those assets
  - Custody of assets
  - Recording or reporting of related transactions
  - Verification review the correctness of operations made by other individuals
- Types of SOD: individual-level, unit-level, company level

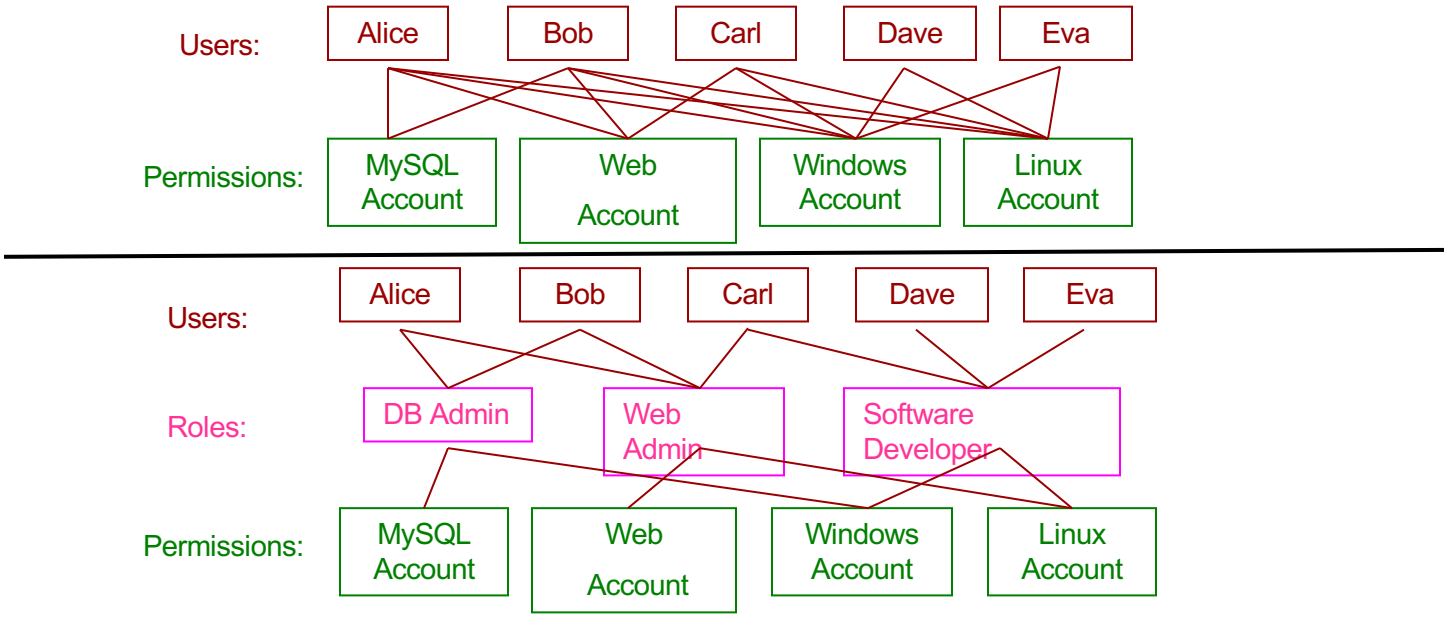
Role Based Access Control (RBAC) Introduction

- Role-Based Access Control (RBAC), access decisions are based on the **roles** that individual users have as a part of an organization
- Role brings together a set of users on one side and a set of permissions on the other
- RBAC model enforces three rules:
  1. Role assignment—a user can access a permission only if they have been assigned a role
  2. Role authorization—a user's active role must be authorized; and
  3. Permission authorization—a user can access a permission only if the permission is authorized for that user's active role



Why Roles?

- Fewer relationships to manage
  - Possibly from *O(mn)* to *O(m+n)*, where *m* is the number of users and *n* is the number of permissions
- Roles add a useful level of abstraction
  - Organizations operate based on roles
- A role may be more stable than the collection of users and the collection of permissions that are associated with it

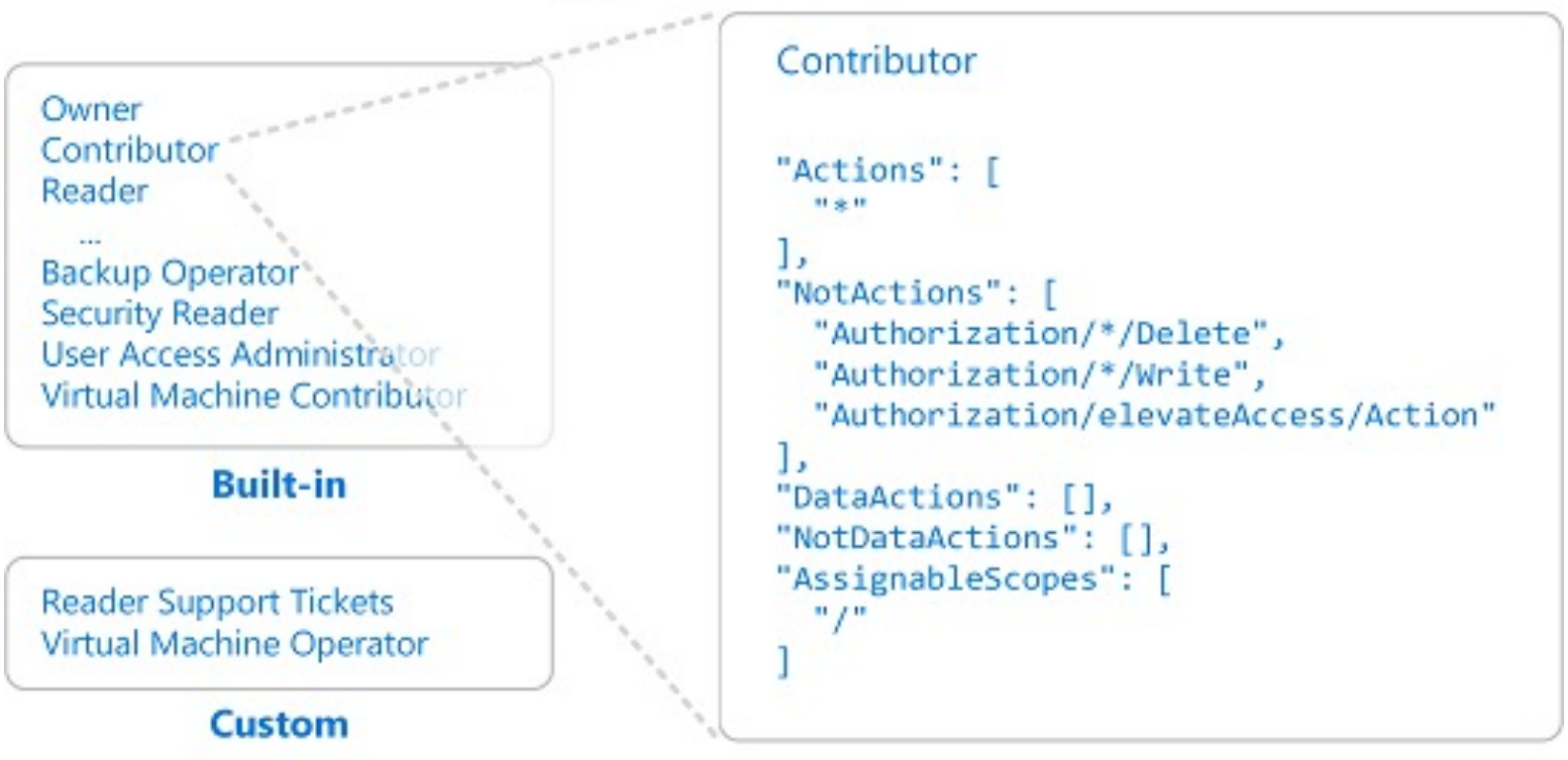


Example: Active Directory RBAC

Class	Roles	Part Purchase Orders	Receiving	Vendor Payments & Checking
IT	System Administrator*	read, write, execute	read, write, execute	read, write, execute
HR	HR			
WH	Procurement Manager	read, write, execute	read	read
	Warehouse Manager		read	
Rec	Receiving Manager	read	read, write, execute	
Sales	Sales			
Log	Logistics Coordinator			
	Logistics Track and Trace			
Actg	Accounts Receivable			
	Accounts Payable	read	read	read, write, execute

Example: Microsoft Azure Role Definition

- Azure includes:
  - Several built-in roles, e.g., Virtual Machine Contributor
  - Custom roles that can be created on demand



Source: docs.microsoft.com

RBAC Elements

- **Object –** data or services
- **User** – entity requesting access to object
  - Has no access as user, only in role
- **Role** – package of permissions assigned to users by association/binding
- **Session** – a mapping between a user and a set of activated roles
- **Permission** – grants access to operation on an object
- **Operation** – specific functions depending on object (e.g., read, write)
- **Associations:**
  - Many-to-many user-to-role assignment
  - Many-to-many permission-to-role assignment
  - Many to many operations to object assignment

A Formal RBAC System

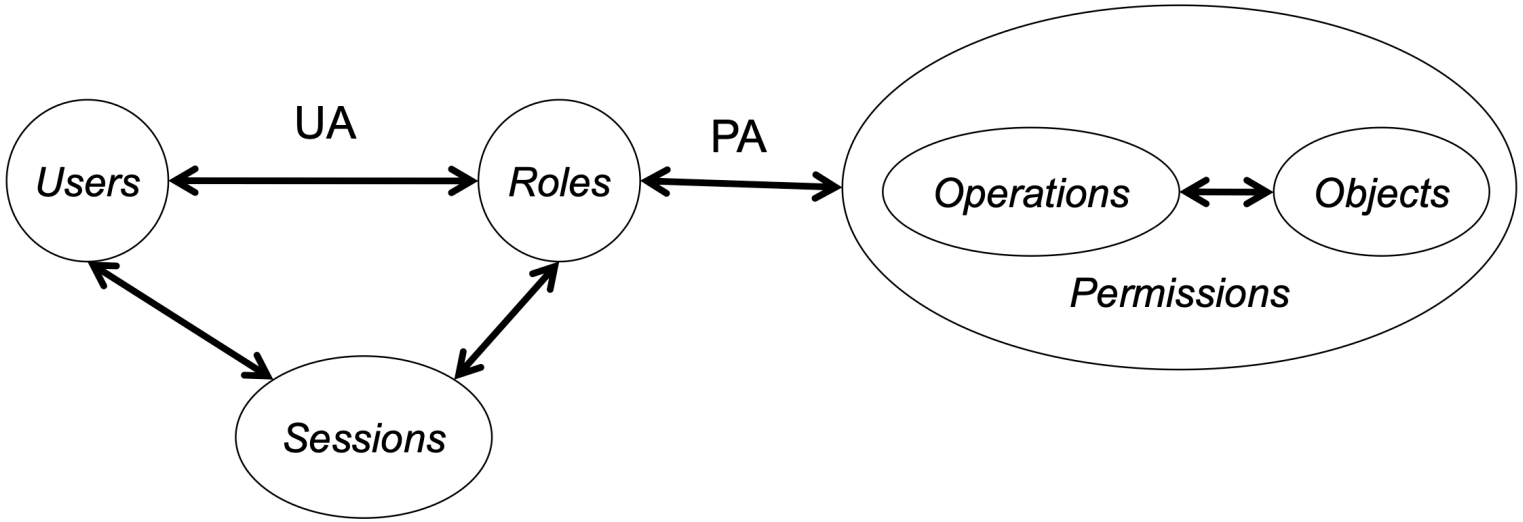
- Defined over the following elements
  - U: user set
  - R: role set
    P: permission set
    S: session set (not always used)
- Relations
  - $UA \subseteq U \times R$ user assignment (which users belong to which roles)
  - $PA \subseteq P \times R$ permission assignment (which permissions belong to which roles)
  - Note: Permissions are positive (not negative) statements
- Functions
  - user: $S \rightarrow U$ (e.g., session s_i belongs to user u_j)
  - roles: $S \rightarrow 2^{|R|}$ (mapping of each session to set of roles)

RBAC Family of Models

- RBAC-0: Flat or Core RBAC
  - Direct assignment to user with “role” or job function
  - No hierarchy
  - Multiple simultaneous active roles per user
- RBAC-1: Hierarchical RBAC
  - Hierarchies
  - Inheritance
- RBAC-2: Constrained RBAC
  - Adds constraints: static separation of duties
- RBAC-3: Symmetric RBAC
  - Adds additional constraints: Dynamic separation of duties
  - NIST model

Flat “Core” RBAC-0

- Assignments (or associations)
  - Users to roles (many-to-many)
  - Permissions to roles (many-to-many)
  - Selective role activation into sessions
  - Multiple simultaneous active roles per user



Hierarchies in RBAC1

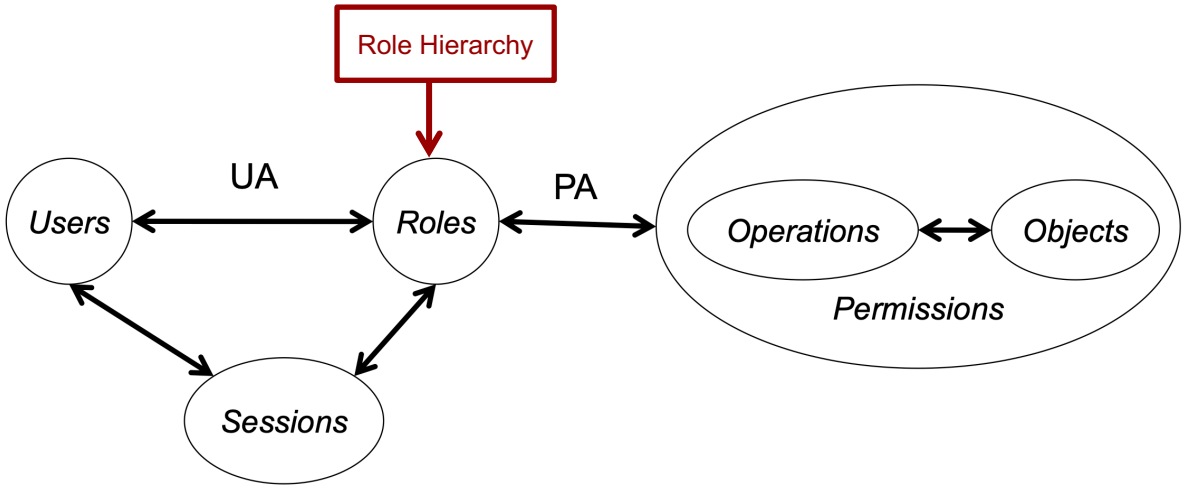
- Defined over (U, R, P, S, PA, UA)
- $H \subseteq R \times R$ (partial order on the set R) called role hierarchy or role dominance relationship
- "$x \geq y$" implies *role x* can dominate *role y*
- "$x \leq y$" implies *role x* can be dominated by *role y*
- roles: $S \rightarrow 2^{|R|}$, such that:
  - $roles(s_i) \subseteq \{r \mid (\exists r' \geq r)\ [user(s_i), r'] \in UA\}$
  - s_i has permissions $\bigcup_{r \in roles(si)} \{p \mid (r'' \geq r)\ [p, r''] \in PA\}$
- Sometimes called "General Hierarchical RBAC"

Notes on Roles

- Roles are a partial order, which means
  - Reflexive ($r \geq r$)
  - Transitive ($x \geq y \wedge y \geq z \Rightarrow x \geq z$)
  - Antisymmetric (if $x \geq y$ and $y \geq x$, then $x = y$)
- Permissions propagate from subordinates (below) to superior roles (above)
- Can leverage hierarchical nature of organizations to more effectively manage roles
- Natural way of reflecting authority, responsibility and competency

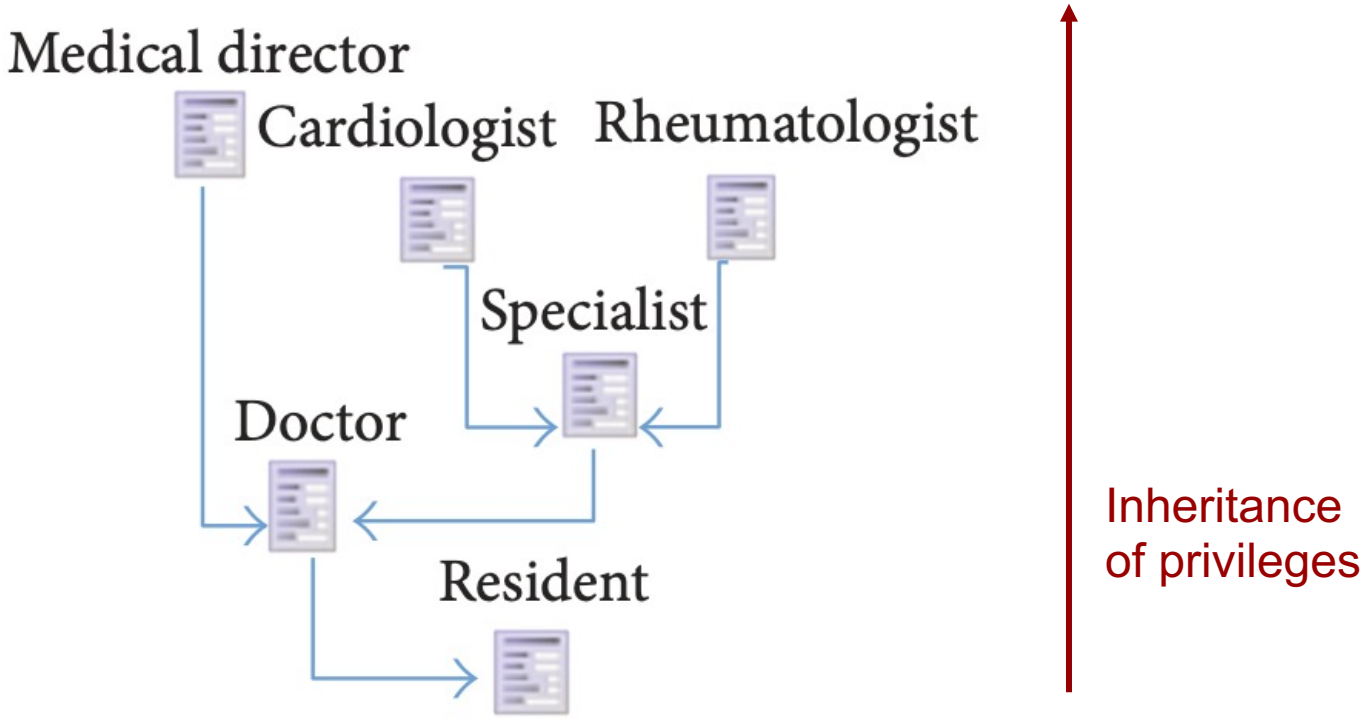
Hierarchical RBAC-1

- All RBAC-0 requirements, plus
- Hierarchy
  - Partial order on roles (seniority)
  - Senior role acquires permissions of its juniors
  - Simplifies role engineering tasks using inheritance of one or more parent roles



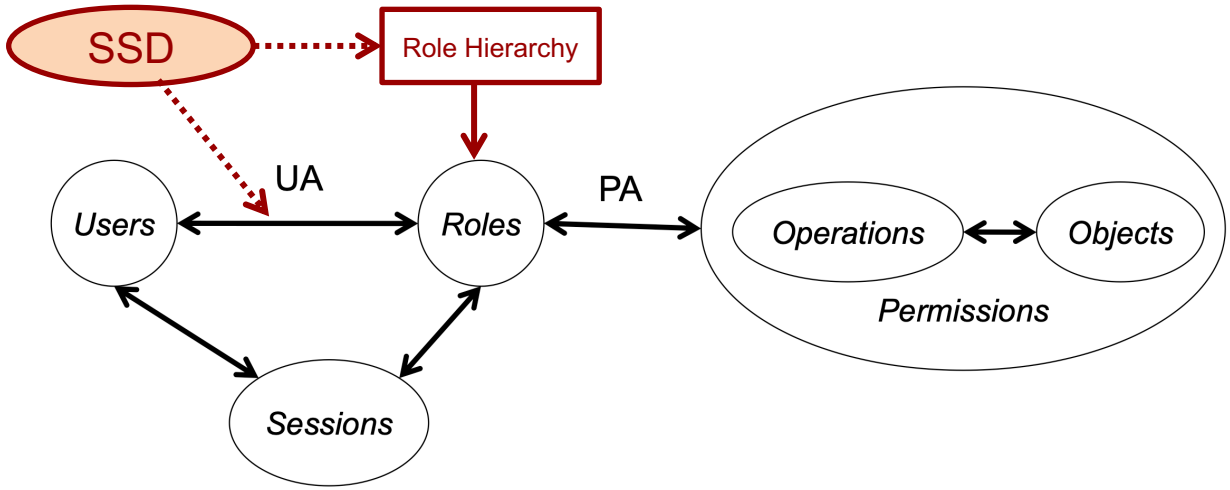
Examples of Role Hierarchies

- Reflect hierarchical structure of roles in organization



Constrained RBAC-2

- All RBAC-1 requirements, plus
- Constraints:
  - Static separation of duties (SSD) is based on user to role assignment
  - Enforce mutual membership exclusions across role assignments
  - Facilitate dual control policies by restricting which roles may be assigned to users in combination
  - SSD provides added granularity for authorization limits which help enterprises meet strict compliance regulations

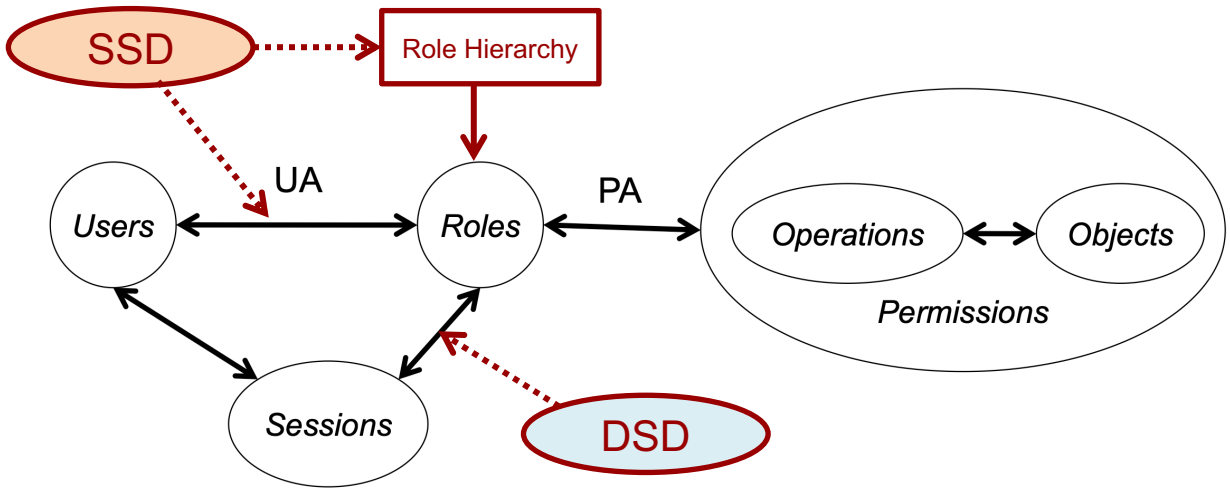


Constraints in RBAC2

- Restrictions on permissible components of RBAC0
- A function that returns acceptable or not acceptable with respect to any assertion
- Can be applied to
  - Relations: UA, PA
  - Functions: *user*, *roles*
- Example: mutually exclusive roles

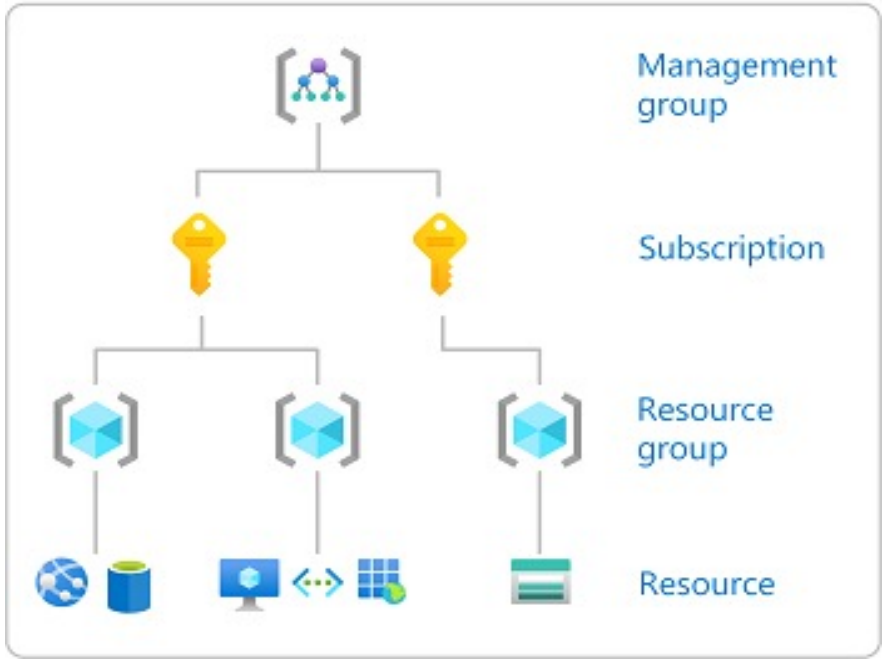
Symmetric RBAC-3

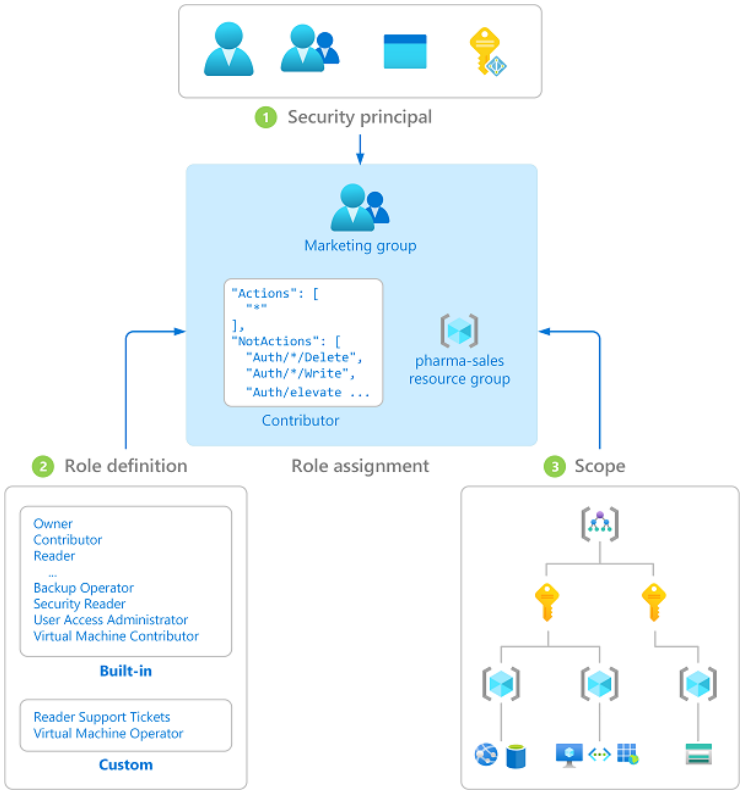
- ANSI INCITS 359
- RBAC-2 requirements, plus
- Additional constraints:
  - Dynamic separation of duties (DSD) is based on role activation
  - DSD policies fine tune role policies that facilitate authorization dual control and two man policy restrictions



Example: Supporting Least Privilege in Azure RBAC

- When you assign a role, you can further limit the actions allowed by defining a **scope**

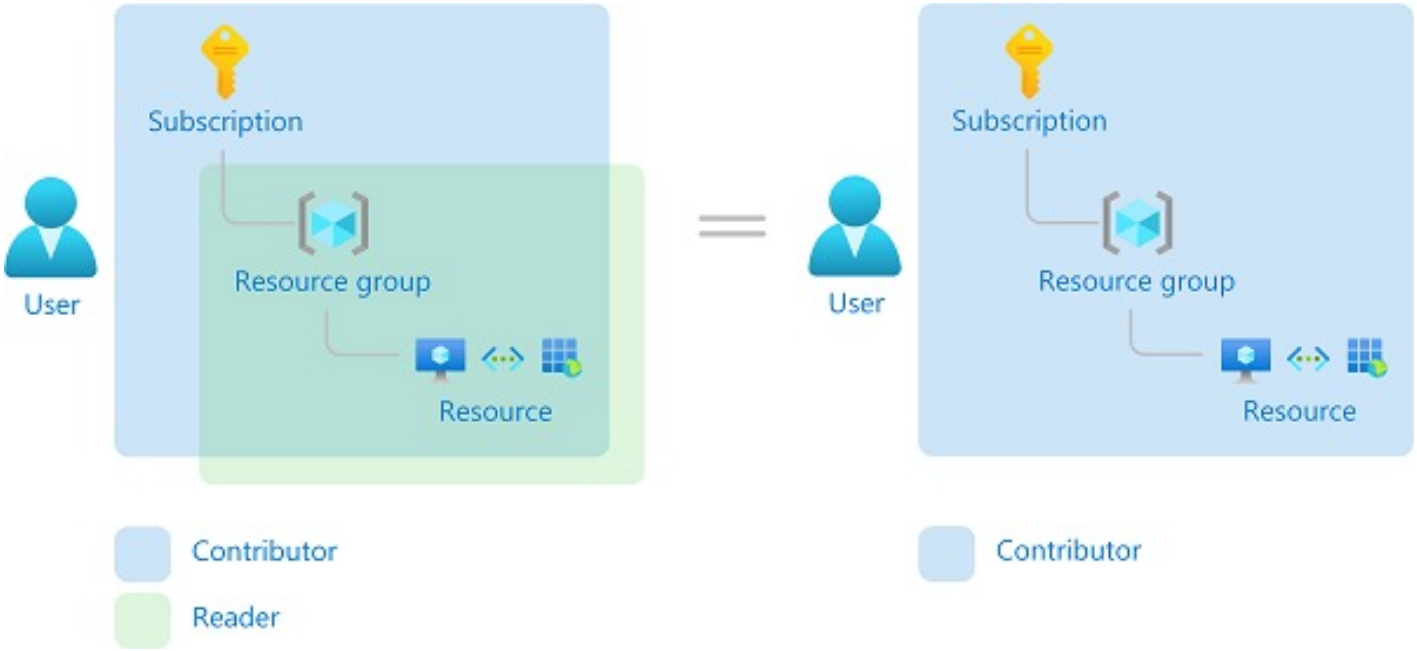




Source: docs.microsoft.com

Example: Multiple Role Assignments in Azure RBAC

- Azure RBAC is an additive model: effective permissions are the sum of the role assignments



Source: docs.microsoft.com

RBAC Properties

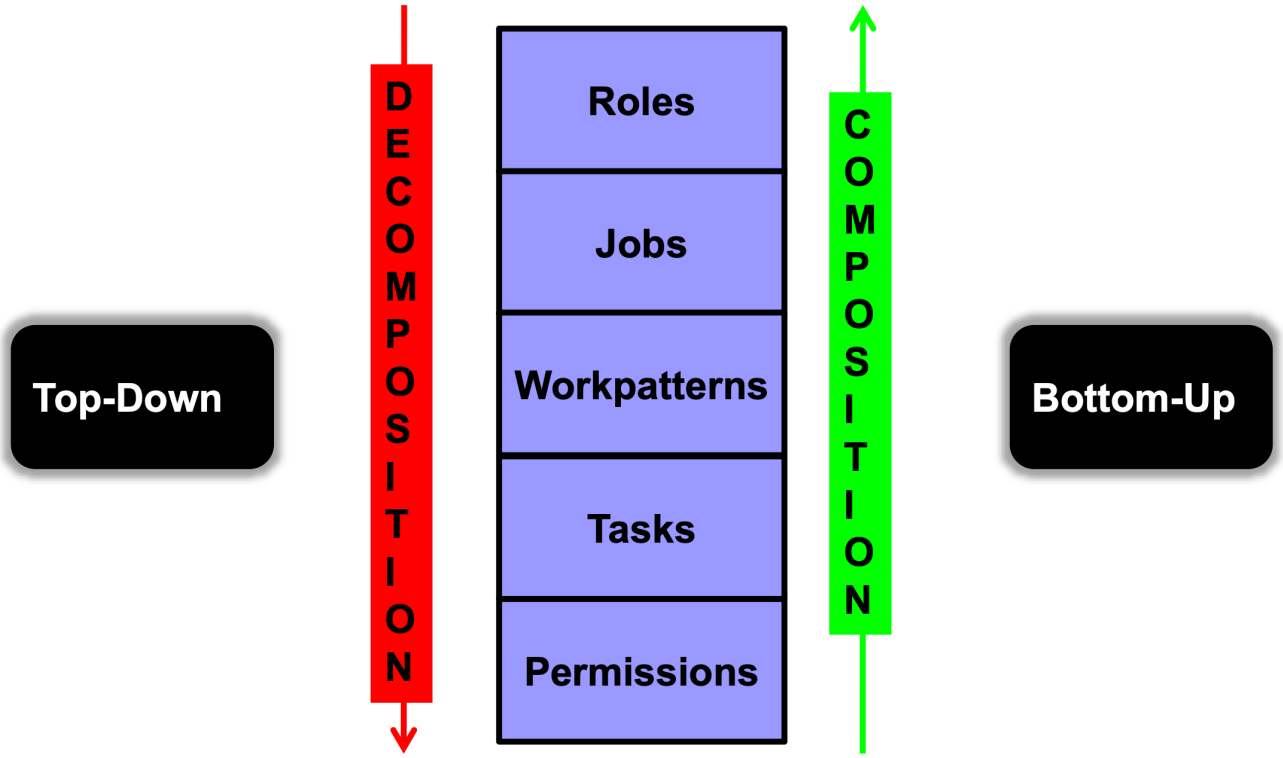
- Roles are collections of permissions, users, and possibly other roles (many-to-many)
- Role hierarchies simplify RBAC management and can be derived from organizational structure
- RBAC implementations simplify access control but may require role engineering
- RBAC supports well-known security principles:
  1. Least Privilege
     - Least Privilege is supported because RBAC can be configured so only those permissions required for tasks conducted by members of the role are assigned to role
  2. Separation of duties
     - Separation of duties is achieved by ensuring that mutually exclusive roles must be invoked to complete a sensitive task

Role Engineering

- Implicit assumptions of RBAC are:
  - Roles exist (!)
  - Roles accurately reflect activities, functions, and responsibilities in the organization
- *Role engineering* is the process of developing an RBAC structure for an organization
- Challenges:
  - NIST identified role engineering as the most costly and time consuming aspect of RBAC execution
    - Even for RBAC0
    - Role specification can take up to 3 – 4 months to establish consensus
    - G. Tassey, “The economic impact of role-based access control”. NIST Report 02-1. 2002

Role Engineering Process

- Top-down is more efficient, but may not be feasible in legacy systems



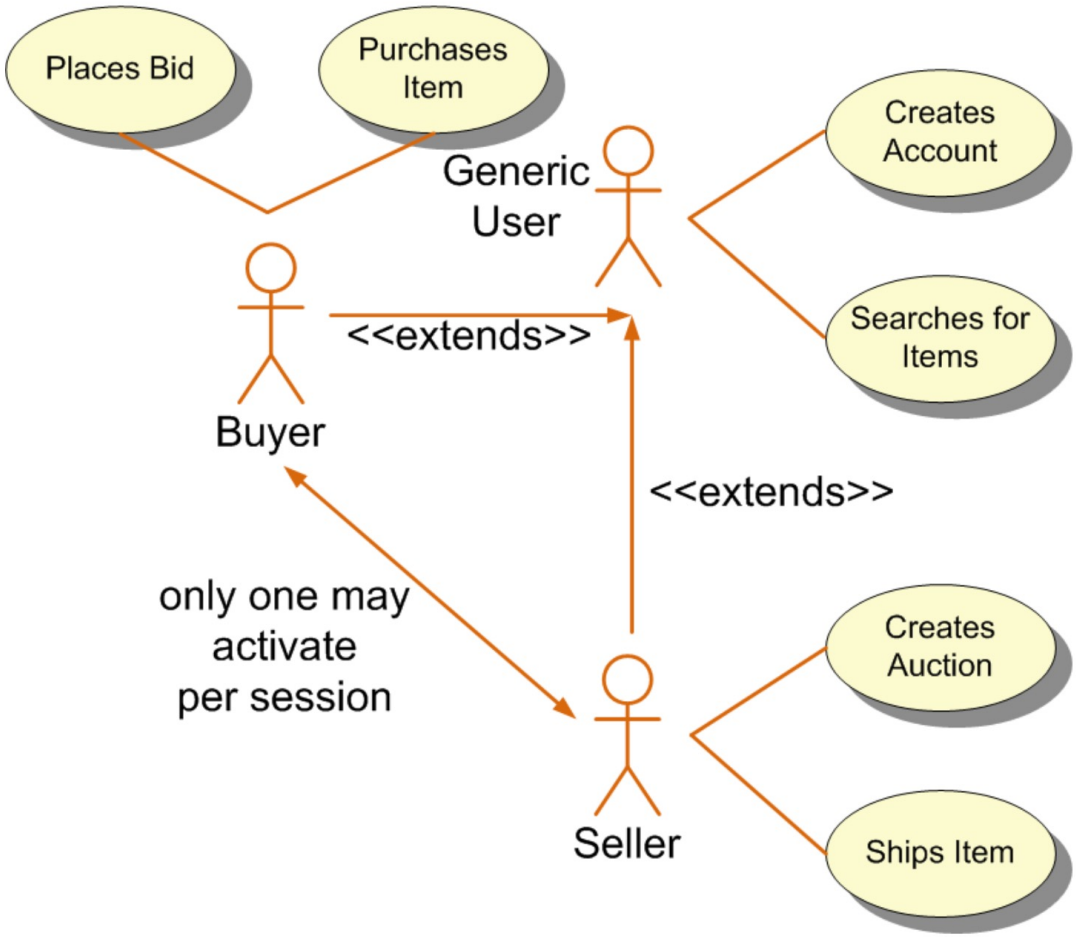
The Hybrid Approach?

Role Engineering Process (Simplified)

1. Define Security Use Cases (Business team)
2. Derive into Entity Lists (Business team or Security team)
   - Define Roles
   - Define the Resources and Operations
   - Map the Roles to the Resources and Operations
3. Assign Users to Roles (HR or Business team)

Source: iamfortress.net

Role Engineering Process Example



Source: iamfortress.net

Define Security Use Cases for the Application

- Legend: red – roles, blue – objects, green – operations
- use case #1 : User must authenticate before landing on the home page
- use case #2 : User must be a Buyer before placing bid on auction or buying an item
- use case #3 : User must a Seller to create an auction or ship an item purchased
- use case #4 : All Users may create an account and search items
- use case #5 : A particular user may be a Buyer, or a Seller, but never both simultaneously

Define Roles

- role name: “Users” description: “Basic rights for all buyers and sellers”
- role name: “Buyers” description: “May bid on and purchase items”
- role name: “Sellers” description: “May setup auctions and ship items”

- Buyers and Sellers inherit from Users as described in use case #4, add Role Inheritance Relationships:
  - child name: “Buyers” parent name: “Users“
  - child name: “Sellers” parent name: “Users“

- Role Activation Constraint (as described in use case #5)
  - role name: “Buyers“
  - role name: “Sellers“
  - type: “DYNAMIC“

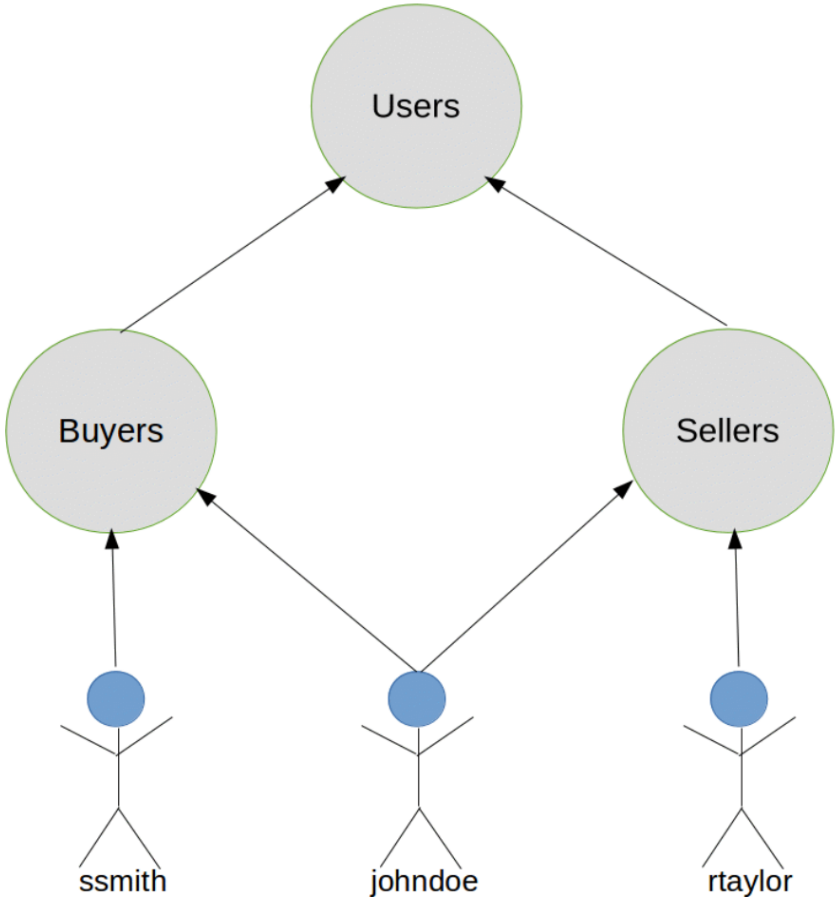
Define the Resources and Operations

- Add Permissions:
  - object name: “Item” description: “This is a product that is available for purchase”
    - operation name: “search” description: “Search through list of items”
    - operation name: “bid” description: “Place a bid on a given product”
    - operation name: “buy” description: “Purchase a given product”
    - operation name: “ship” description: “Ships a given product after sale”
  - object name: “Auction” description: “Controls a particular online auction”
    - operation name: “create” description: “May start a new auction”
  - object name: “Account” description: “Each user must have one of these”
    - operation name: “create” description: “Ability to setup a new account”

Map the Roles to the Resources and Operations

- Grant Permissions to Roles:
  - role: “Buyers” object name: “Item” oper: “bid“
  - role: “Buyers” object name: “Item” oper: “buy“
  - role: “Sellers” object name: “Item” oper: “ship“
  - role: “Sellers” object name: “Auction” oper: “create“
  - role: “Users” object name: “Item” oper: “search“
  - role: “Users” object name: “Account” oper: “create“

Assign Roles to the Users



Outline

- Review
- Lipner Model
- Clark-Wilson Model
- Chinese Wall Model
- Role based access control
- Attribute based access control

The Limitations of RBAC

- RBAC provides coarse-grained, predefined and static access control configurations
- Some limitations:
  - Role explosion
  - Management overhead
  - Lack of context
  - Emergency Provisioning
- Today, defining authorization policies based solely on a user's role is not good enough
  - The context surrounding that user, their data, and the interaction between the two are also important to provide access to
    - the right user
    - at the right time
    - in the right location,
    - and by meeting regulatory compliance
- That means evolving an existing Role Based Access Control model to an Attribute Based Access Control (ABAC) model

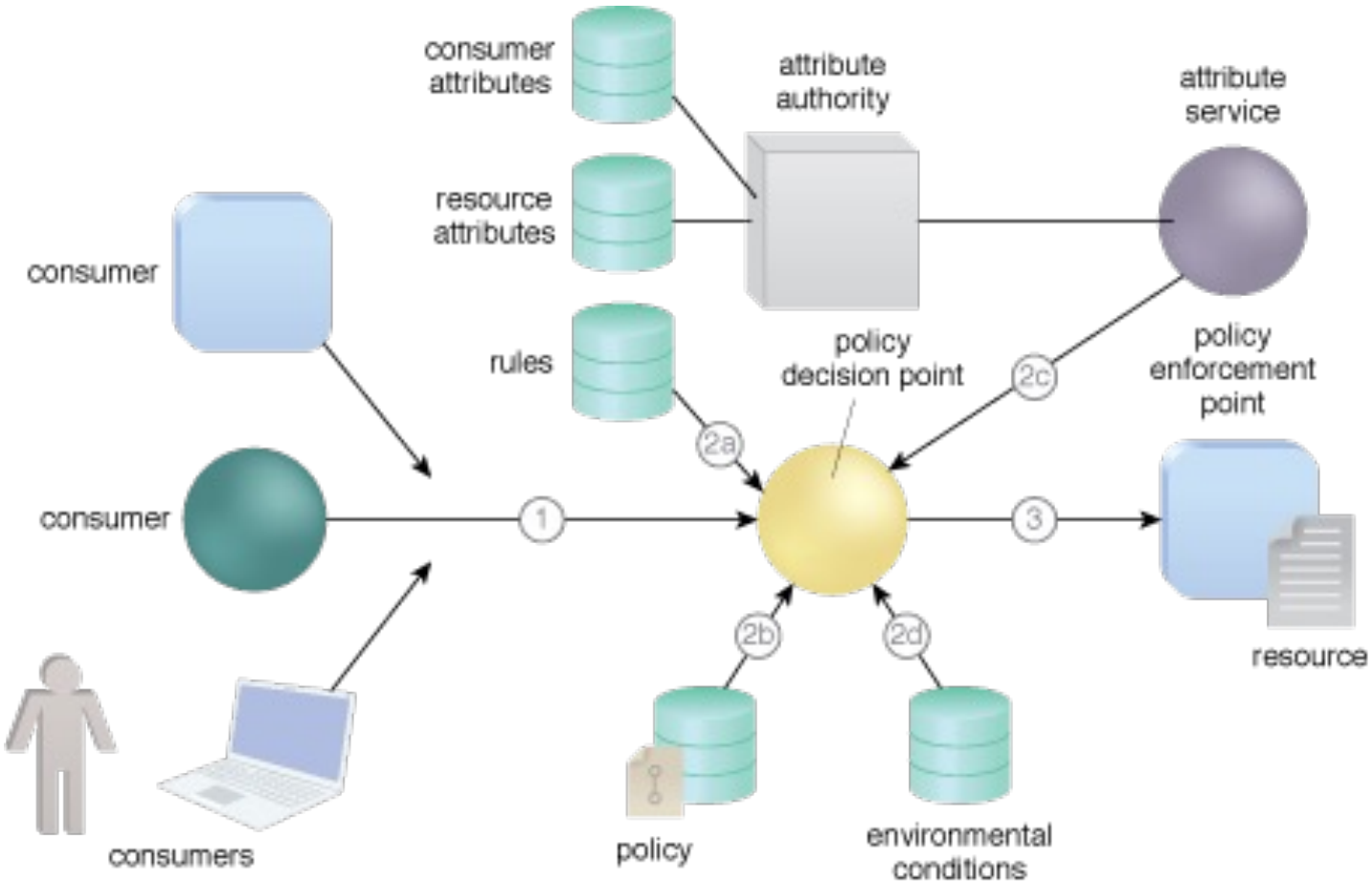
ABAC: Attributes

- Subject Attributes
  - Associated with a subject (user, process) that define the identity and characteristics of the subject
- Resource Attributes
  - Associated with a resource (system function, or data)
- Action Attributes
  - Associated with the requested access rights
- Environment Attributes
  - Describes the operational, technical, or situational environment or context in which the information access occurs
- Attributes can be static and dynamic

Attribute Based Access Control (ABAC)

- *"Access control methodology where authorization to perform a set of operations is determined by evaluating attributes associated with the subject, object, requested operations, and, in some cases, environment conditions against policy rules that describe the allowable operations for a given set of attributes"*
- NIST Special Publication 800-162 "Attribute Based Access Control Definition and Considerations"
- ABAC Rule Example:

ABAC System Architecture Example



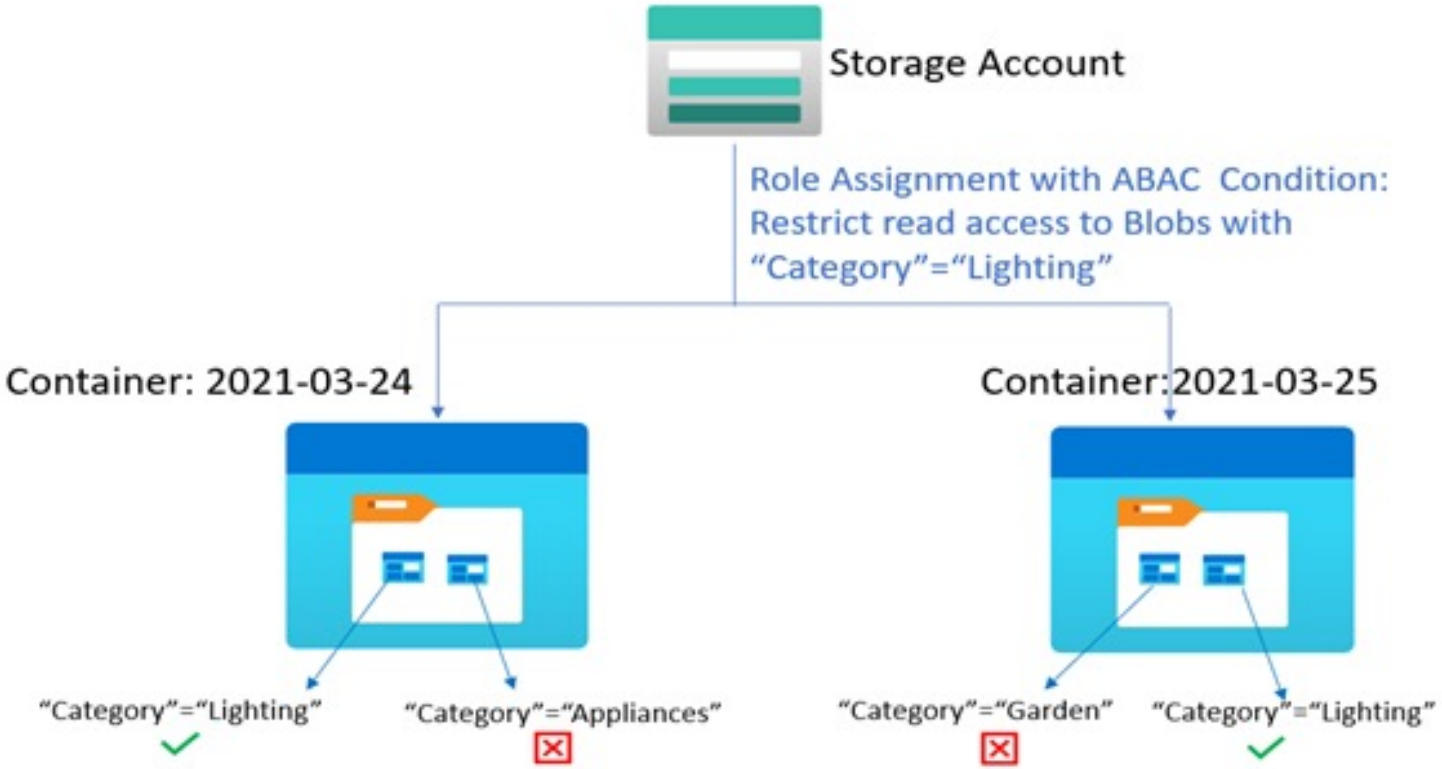
Source: patterns.arcitura.com

Were do we get the attributes?

Attribute sources

Example: Azure ABAC

- Adding ABAC conditions to Azure Blob services



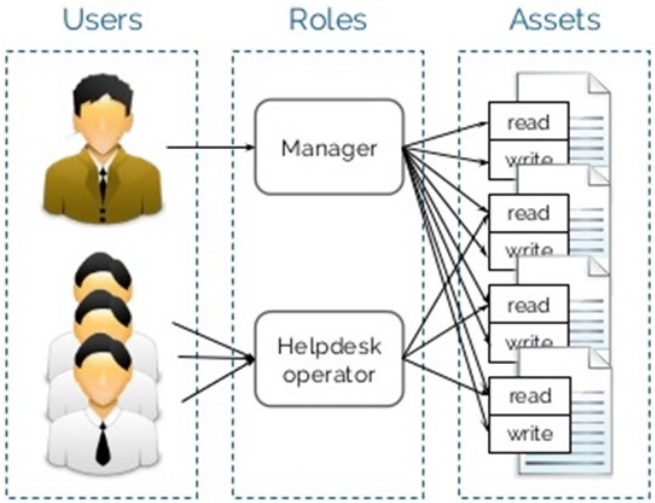
Source: www.drware.com

ABAC Advantages

- ABAC rules can be extremely fine-grained and contextual
- Requires no advance knowledge of requestors
- An individual's attributes can be correlated from multiple sources to create a unified identity
- Highly adaptable to changing needs
  - Efficient for agencies where individuals come and go frequently
  - Support for dynamic groups
- Example: ABAC for AWS
  - https://docs.aws.amazon.com/IAM/latest/UserGuide/tutorial_attribute-based-access-control.html

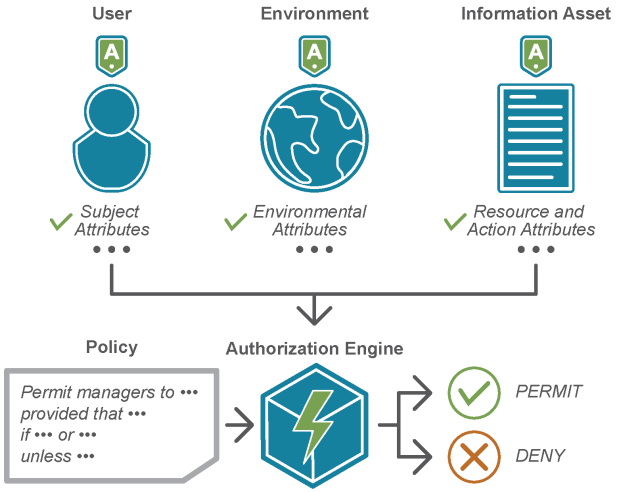
RBAC vs. ABAC

RBAC



- Initial effort of structuring roles
- Advantages in administration and auditability

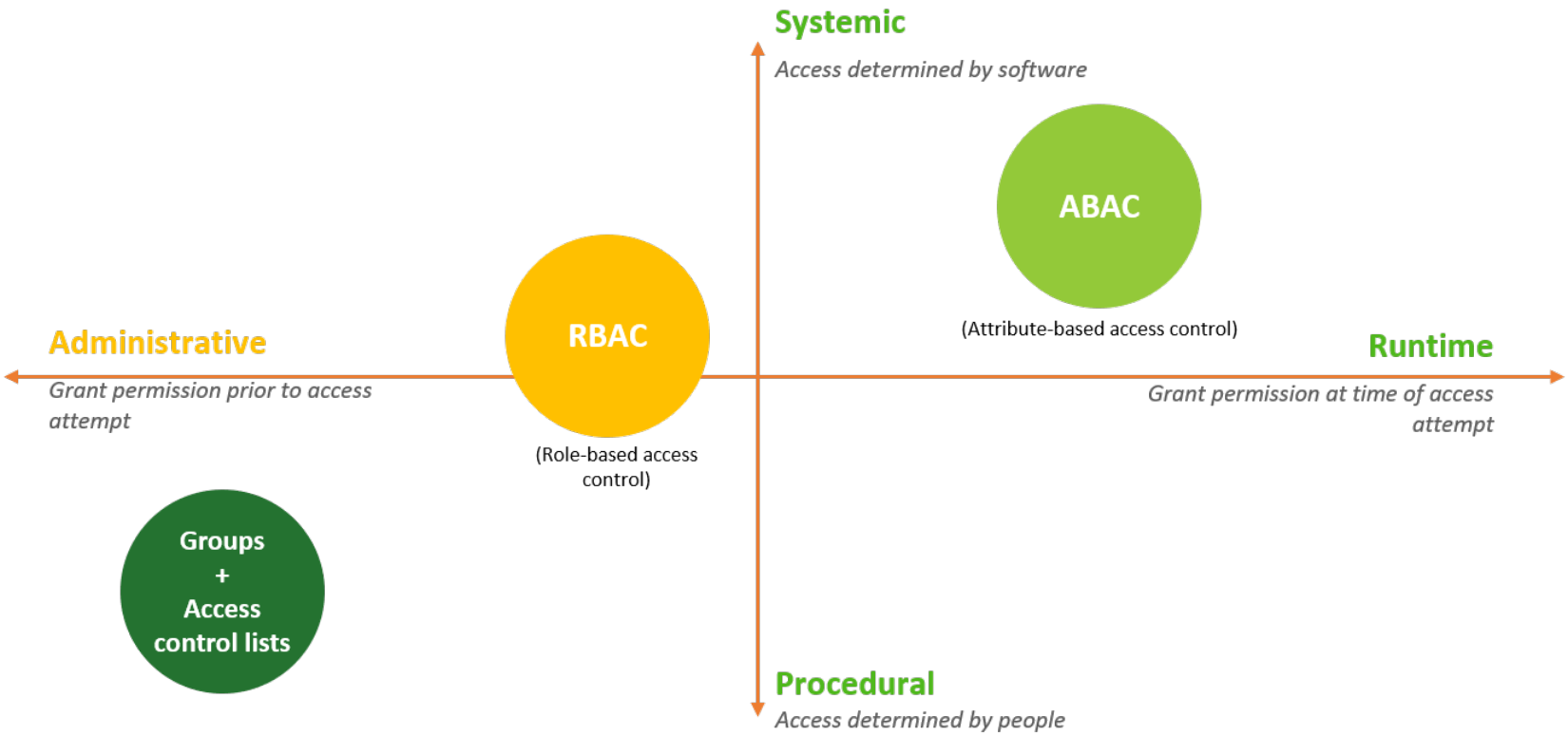
ABAC



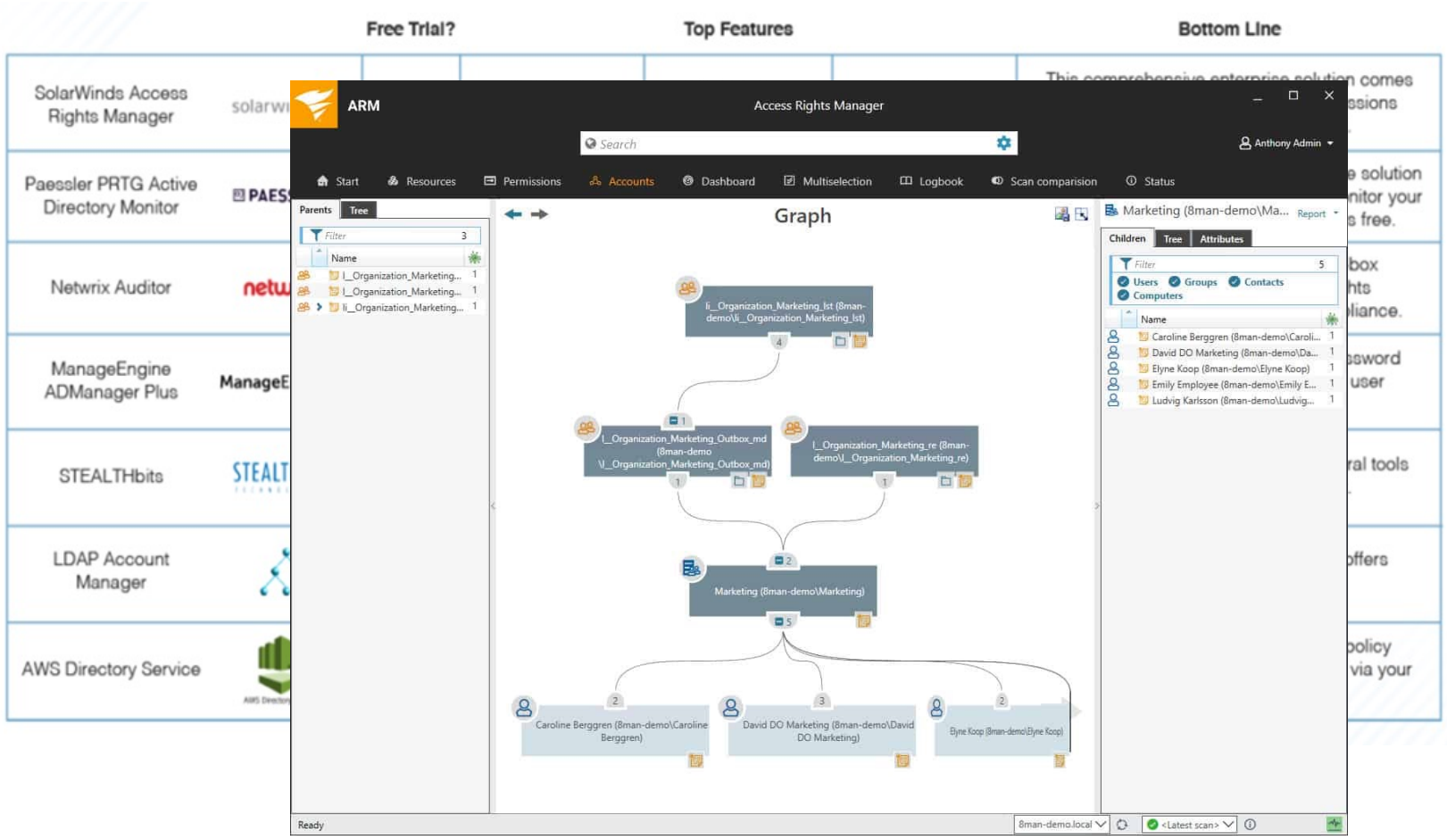
- Immediate creation of rule policies
- Complicated management and audit of user permissions

USC Viterbi School of Engineering

Groups, RBAC, ABAC



Best Access Rights Management Tools



Source: https://www.dnsstuff.com/access-rights-management-tools

USC Viterbi School of Engineering