Access Control

Policy Composition, XACML, and Evaluation Metrics – oh my

Dr. Hale
University of Nebraska at Omaha
Information Security and Policy– Lecture 8 & 9

Today's topics:

- Access control basics
  - Access Control Model
  - Matrix and protection states
  - Access control lists and capability model
- Role Based Access Control
  - Definitions and components
  - Reference Model
  - Policy composition
- NISTIRS
  - Policy Ontology
  - Implementing AC policy
- XACML
- NISTIRS Access Control System Metrics and assigning responsibility
  - Terminology
  - Metrics by type

Definition

A state of access control is said to be *safe* if no permission can be leaked to an unauthorized or uninvited individual

- Access control systems come with a wide variety of features and administrative capabilities
- *Security models* are formal presentations of the security policy enforced by the access control system and are useful for proving theoretical limitations of a system

Types of Access Control Polices

- Discretionary Access Control (DAC, IBAC)
  - individual user sets access control mechanisms to allow or deny access to an object
  - Based on identity of subject and object involved
  - e.g. Diary
- Mandatory Access Control (MAC)
  - system controls access to objects and *individual cannot alter that access*
  - e.g. public court information, military systems
- Originator Controlled Access Control (ORCON)
  - originator (creator) of information controls who can access and disseminate information, not the owner
  - e.g. NDAs on code changes, licensing agreements
- Role Based Access Control (RBAC)
  - access control decisions based on the a user's role in an Organization
  - Roles may be expressed hierarchically
  - Can implement DAC and MAC
- Attributed Based Access Control
  - logical access control based on collections of attributes of objects and users
  - authorization to perform a set of operations is determined by evaluating attributes associated with the subject, object, requested operations, and environment conditions against policy, rules, or relationships that describe the allowable operations for a given set of attributes
- Others exist that are domain specific or are used for solutions to specific access problem

Access Control Models

- Regulate the logical access to information with the system
- Maintained by a collection of policies and enforcement mechanisms
- 4 processes that build on each other:
  - identification: Obtain the identity of the entity requesting access
  - authentication: Confirm the identity of the entity
  - authorization: Determine which actions the entity can perform
  - accountability: Document the activities of the entity and system
- Built on principles for
  - Least privilege – minimum access required for duties
  - Need to know – specific data at specific times
  - Separation of duties – segregating access responsibilities to limit powers

Definition

Access *control lists, matrices, and capability tables* are formal mechanisms that govern the rights and privileges of users

- Can control access to file storage systems, object brokers, or other network communications devices.

A capability table specifies which subjects and objects that users or groups can access

- Often considered user profiles or user policies
- Can take the form of complex matrices

Access Control Tables

- Restrict access according to user, time, duration, and file to regulate the following
  - Who can use the system
  - What authorized users can access in the system
  - Where authorized users can access the system from
  - When authorized users can access the system
  - How authorized users can access the system
- Administrators assign user privileges as rights
- Rights can include
  - Generic access (read, write, execute)
  - Domain specific
  - Functions that determine rights given the current state or historical access or states
  - Functions that determine rights given other current rights

Access Control Matrix

- Tool to describe current protection state
  - Privileges possessed by *subjects* (active entity) with respect to other entities
    - State transitions change elements of matrix
      - Matrix evolves by the autonomous activities of the subjects
    - The set of protection states of the system is represented by the triple (S, O, A) where S is the set of *Subjects*, O is the set of *Objects*, and A is the *matrix of rights*
  - Relies on an authorization scheme
    - Rules that direct how the protection state can be changed

Access Control Matrix as an Abstract Model of the Protection State

	Objects (O)						
Subjects (S)		O_1	…	O_m	S_1	…	S_n
	S_1						
	S_2						
	…						
	S_n						

Matrix A

- Subjects $S = \{ s_1,\ldots,s_n \}$
  - each are subjects and objects that own themselves
- Objects $O = \{ o_1,\ldots,o_m \}$
  - *Could be devices, processes, messages, systems*
  - Subjects are objects (active) but not vice versa
- Rights $R = \{ r_1,\ldots,r_k \}$
  - *r* (read), *w* (write), *x* (execute), *a* (append), *o* (own)
  - meaning of a right may vary depending on the object involved
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_x, \ldots, r_y \}$ means subject s_i has rights $r_x, \ldots, r_y$ over object o_j

can think of R in terms of reachability as well (a different R, from before)

Access Control by Boolean Expression Evaluation

- ACM controls access to objects
  - Objects are records and fields
  - Subjects are authorized users with attributes
  - Verbs define type of access (rights)
  - Rules associated with objects, verb pair
- Subject attempts to access object
  - Rule for object, verb evaluated, grants or denies access

Example

- Subject (s) **Abe**
  - role (clerk), group (courthouse)
- Verb (activity) **sign**
  - Default: Deny
- Object **tax-doc**
  - Access Rule for tax-doc

    sign: 'clerk' in s.role and
    'courthouse' in s.group and
    0800 ≤ hour ≤ 1700 and
    "Monday" ≤ day ≤ "Friday"

Activity	**Default Access**
Read	Granted
Write	Deny
Sign	Deny

maps to policy:

$\forall s \in Subjects, t \in Times, d \in Days,$

$sign(s) \Leftrightarrow (role(s) = clerk) \wedge (0800 \leq t \leq 1700) \wedge d \in \{M, T, W, Th, F\}$

Access Control Matrix for Abe

- Protection state changes according to hour and day

- At 1am on Monday

- At 3pm on Wednesday

- At 3pm on Saturday

	…	tax_doc	…
…			
Abe		read	
…			

	…	tax_doc	…
…			
Abe		read, sign	

	…	tax_doc	…
…			
Abe		read	
…			

State Transitions

- Change the protection state of system –
  - $X_0 = (S_0, O_0, A_0)$ be the initial state
  - $T = [\tau_1, \tau_2, \ldots]$ commands
- Commands are transformation procedures that follow the authorization scheme
    - Change the triple
      - Alter subject or object set based on τ
      - Change entries in the access control matrix rights
    - Use parameters to state how the change is made
- Given the initial state and the authorization scheme, it is a formal process to characterize all of the protection states that are reachable

Primitive Commands, τ

- To maintain proper logical values for pre- and post-conditions
  - Protection *before* state: (S,O,A)
  - Protection *after* state: (S′, O ′, A ′)
- **create subject** s
  - Creates new **row** and **column** in ACM, but does not alter rights
  - Precondition (subject does not exist) : $s \notin S$
  - Postconditions:

$S' = S \cup \{ s \} \wedge$	[subject exists]
$O' = O \cup \{ s \} \wedge$	[subject object exists]
$(\forall y \in O)[A'[s, y] = \varnothing] \wedge$	[initialize access to all objects to null, i.e. deny]
$(\forall x \in S)[A'[x, s] = \varnothing] \wedge$	[ensure no other subject has access to the new subject object]
$a'[s, s] = \{\text{“own”}\} \wedge$	[establish ownership of self]
$(\forall x \in S)(\forall y \in O)[A'[x, y] = A[x, y]]$	[everything else stays the same as it was before]

- **subject s creates object** o
  - Creates new **column** in ACM and assigns ownership to subject s
- **destroy subject** s
  - Deletes **row**, **column** from ACM
- **destroy object** o
  - Deletes **column** from ACM

Sample Command Logic

- Allows for provability
- enter *r* into A[*s*, *o*]
  - Adds *r* rights for subject s over object *o*
  - Precondition: $s \in S$, $o \in O$
  - Postconditions:

    $S' = S \wedge O' = O \wedge$

    $A'[s, o] = A[s, o] \cup \{ r \} \wedge$

    $(\forall x \in S')(\forall y \in O' - \{ o \})\ [A'[x, y] = A[x, y]] \wedge$

    $(\forall x \in S' - \{ s \})(\forall y \in O')\ [A'[x, y] = A[x, y]]$
- delete r from A[s, o]
  - Removes r rights from subject s over object o
- Make subject *p* the owner of file *g*

```
command make-owner(p, g)
        enter own into A[p, g];
end
```

- Conditional commands
  - Let *p* give *q* *r* and *w* rights over *f*, if *p* owns *f* and *p* has *copy* (*c*) rights over *q*

```
command grant-read-file(p, f, q)
        if own in A[p, f] and c in A[p, q]
        then
            enter r into A[q, f];
            enter w into A[q, f];
end
```

Copying Rights

- Allows possessor to give rights to another
- Often attached to only the applicable right
  - r is read right that cannot be copied
  - rc is read right that can be copied
- Depending on the model, the copy flag may copied when giving r rights

Owning Rights

- Usually the possessor (owner) can change entries in ACM column by adding and deleting rights for others with respect to that object
  - May depend on what system allows
    - Can't give rights to specific (set of) users
    - Can't pass copy flag to specific (set of) users

Principle: Attenuation of Privilege

- says you can't give rights you do not possess
  - Restricts addition of rights within a system
  - Usually *ignored* for owner since owner gives self rights, gives them to others, deletes self rights.

Two Approaches

- ACL – Access Control List for specifying object access
- Capability Lists - for specifying subject capabilities

Access Control Lists

	Obj_1	Obj_2	Obj_3
Allen	rwxo	r	rw
Bea	rx	rwo	
Cody	rx	r	rwo

- Uses the columns of access control matrix
- ACLs:
  - Obj_1: { (Allen, *rwxo*) (Bea, *rx*) (Cody, *rx*) }
  - Obj_2: { (Allen, *r*) (Bea, *rwo*) (Cody, *r*) }
  - Obj_3: { (Allen, *rw*) (Cody, *rwo*) }
- The normal use is if not named, *no* rights over file
  - Based on Principle of Fail-Safe Defaults
  - Extended to composed policies

ACL Usage

- Who can modify the ACL?
  - Creator is given *own* right for modification
  - Can be a something available like a copy flag that allows a right to be transferred, so ownership not needed
- ACL application to privileged users varies across vendors and with respect to abbreviated or full blown entries
- Denying access
  - If ACL entry denies user access, then deny access
  - If the user is not in file's ACL nor in any group named in file's ACL then deny access
  - If there are conflicts, the norm is to deny access if any entry denies access

Capability Lists

- Rows of access control matrix
- C-Lists:
  - Allen: { (Obj_1, rwxo) (Obj_2, r) (Obj_3, rw) }
  - Bea: { (Obj_1, rx) (Obj_2, rwo) }
  - Cody : { (Obj_1, rx) (Obj_2, r) (Obj_3, rwo) }

	Obj_1	Obj_2	Obj_3
Allen	rwxo	r	rw
Bea	rx	rwo	
Cody	rx	r	rwo

ACLs vs. Capabilities

- Theoretically equivalent
  1. Given a subject, what objects can it access, and how? (answered by C-Lists)
  2. Given an object, what subjects can access it, and how? (answered by ACLs)
- Second question has in past been of most interest making ACL-based emerge as more common
- First question becomes more important for incident response

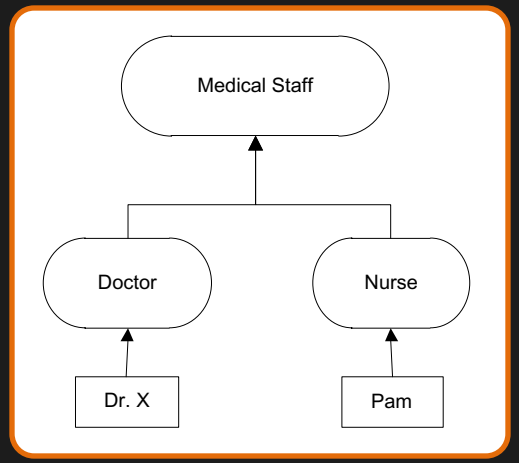
Exercise

- Formally write the state changes required for the primitive command: **subject s creates object** *o*

Looking at RBAC in particular

Role-Based Access Control

- Access control model specified in terms of roles and role hierarchies, role activation, and constraints on user/role membership and role set activation
- Ease of Role Change
  - Allison, bookkeeper for Math Dept, has access to financial records.
  - She leaves.
  - Betty hired as the new bookkeeper, so she now has access to those records
  - The role of "bookkeeper" dictates access, not the identity of the individual
- Role Containment
  - Trainer can do all transactions that trainee can do (and then some). This means role *r* can contain another role *r′* where r dominates *r′*.



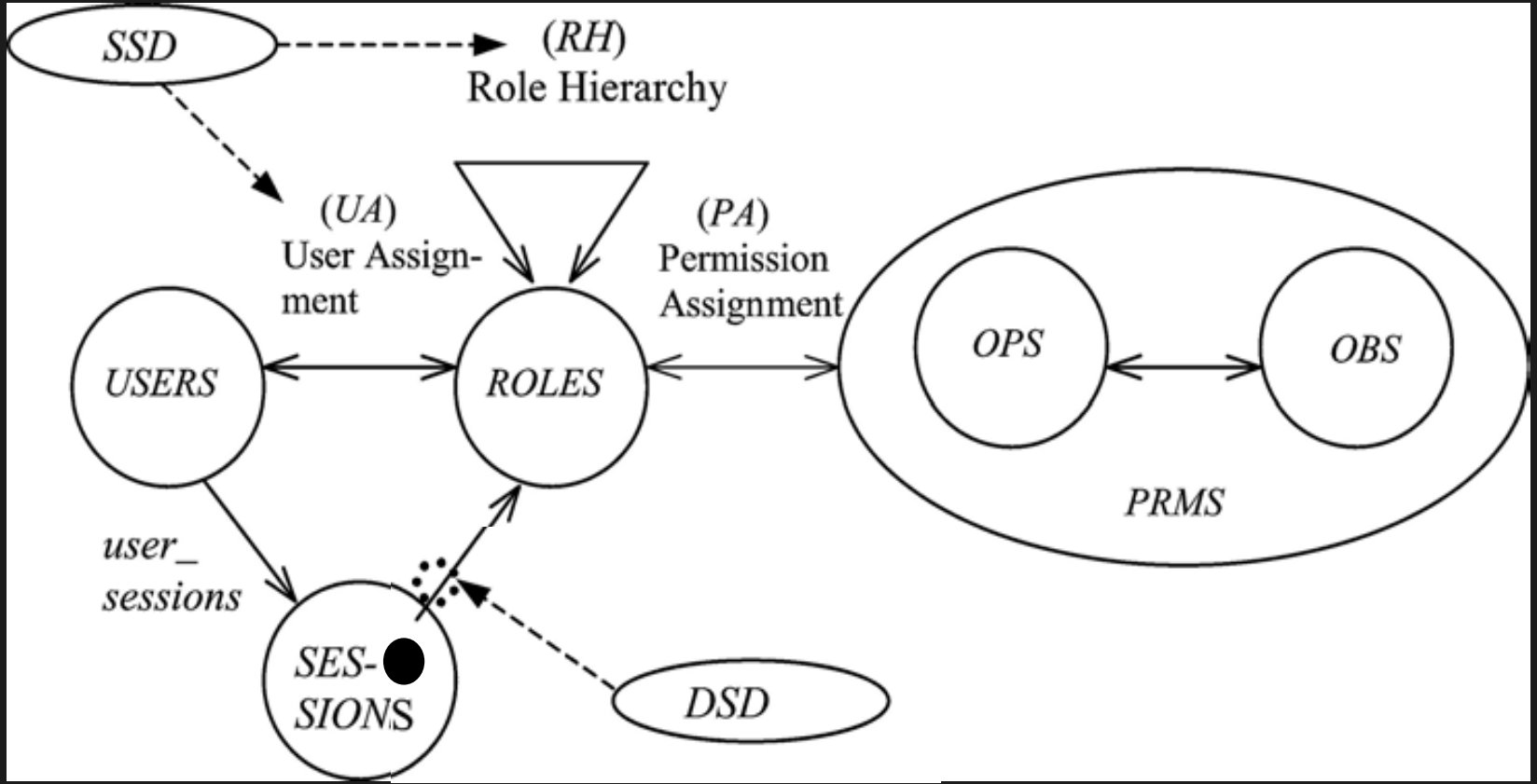
Role-Based Access Control (ANSI INCITS 359-2004)

- *Users*
  - humans but can be extended to generic subjects
- *Objects*
- *Operations*
  - program, which upon invocation executes a function for a user
- *Permissions*
  - approval to perform an operation on one or more RBAC protected objects
- *Role*
  - job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role

RBAC Reference Model – 4 Model Components

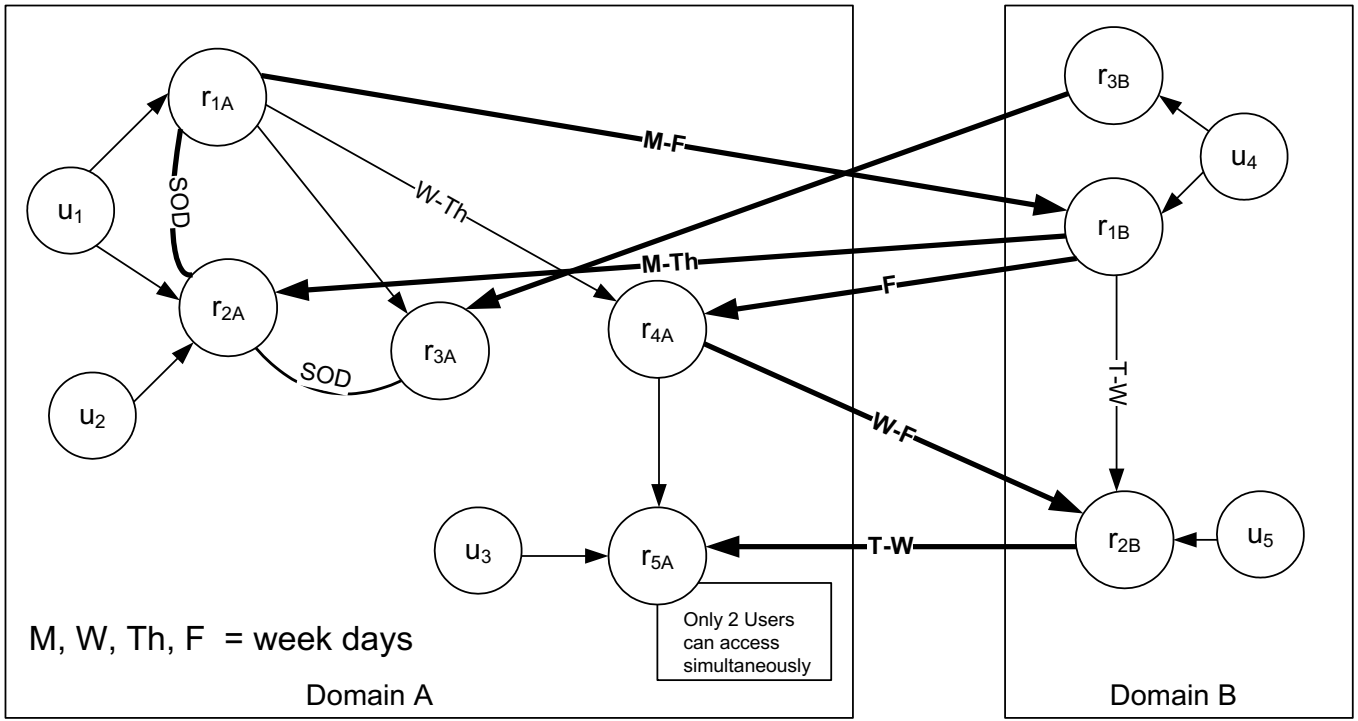
- Core RBAC
  - Minimum collection of RBAC elements
  - User-role and permission-role assignment relations
  - Role activation as part of a user's session
  - Required in any RBAC system
- Hierarchical RBAC
  - Adds role hierarchies as a partial order of seniority among roles
  - Role has a set of authorized users and authorized permissions
- Static Separation of Duty Relations (SSD)
  - Adds relations among roles with respect to user assignments
  - Defines relations both in the presence and absence of role hierarchies
- Dynamic Separation of Duty Relations (DSD)
  - Defines exclusivity relations with respect to roles when activated as part of a user's session

Reference Model



RBAC

Example Policy Composition and Temporal Properties



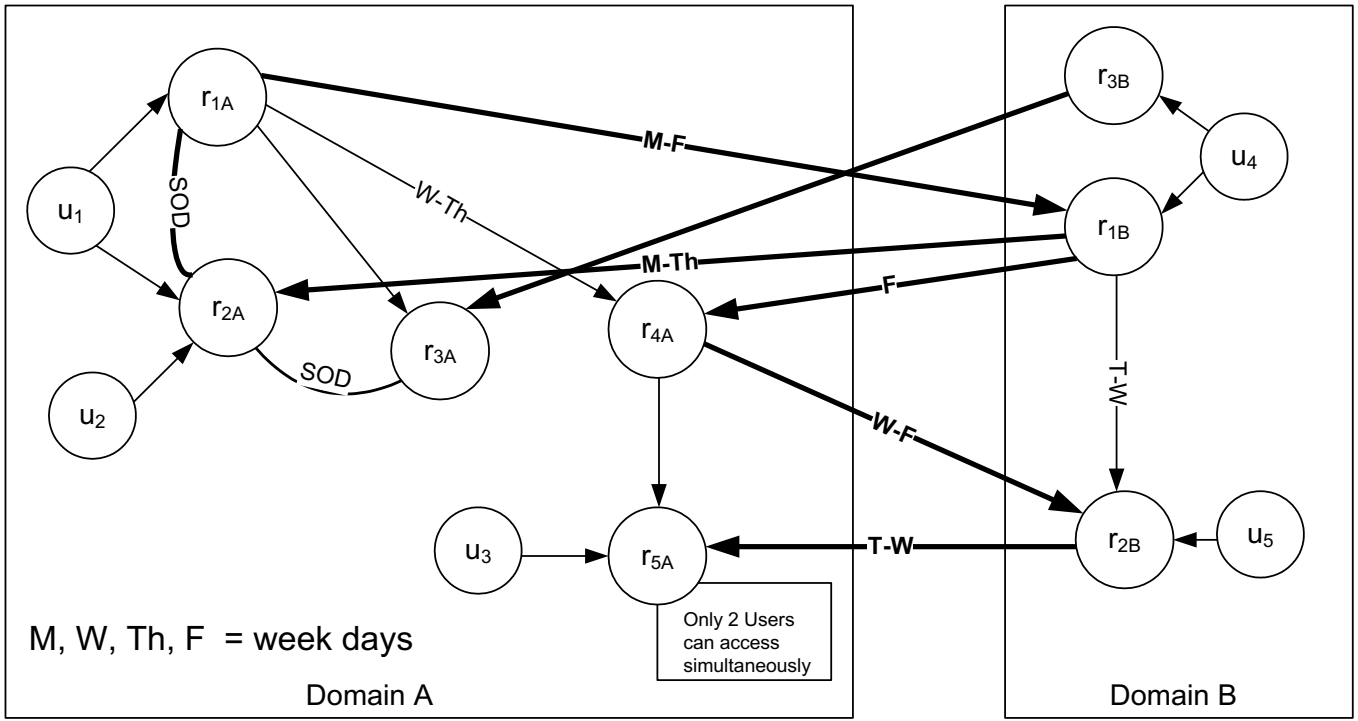
RBAC

Definitions for Example

Let A and B be two different domains (e.g. two systems or organizations)
Let U be the set of all users $\{u_1 \ldots u_n\}$ who have access to a system at any given time
Let R be the set of roles in a given system/organization X $\{r_{1X} \ldots r_{nX}\}$ where X is A or B

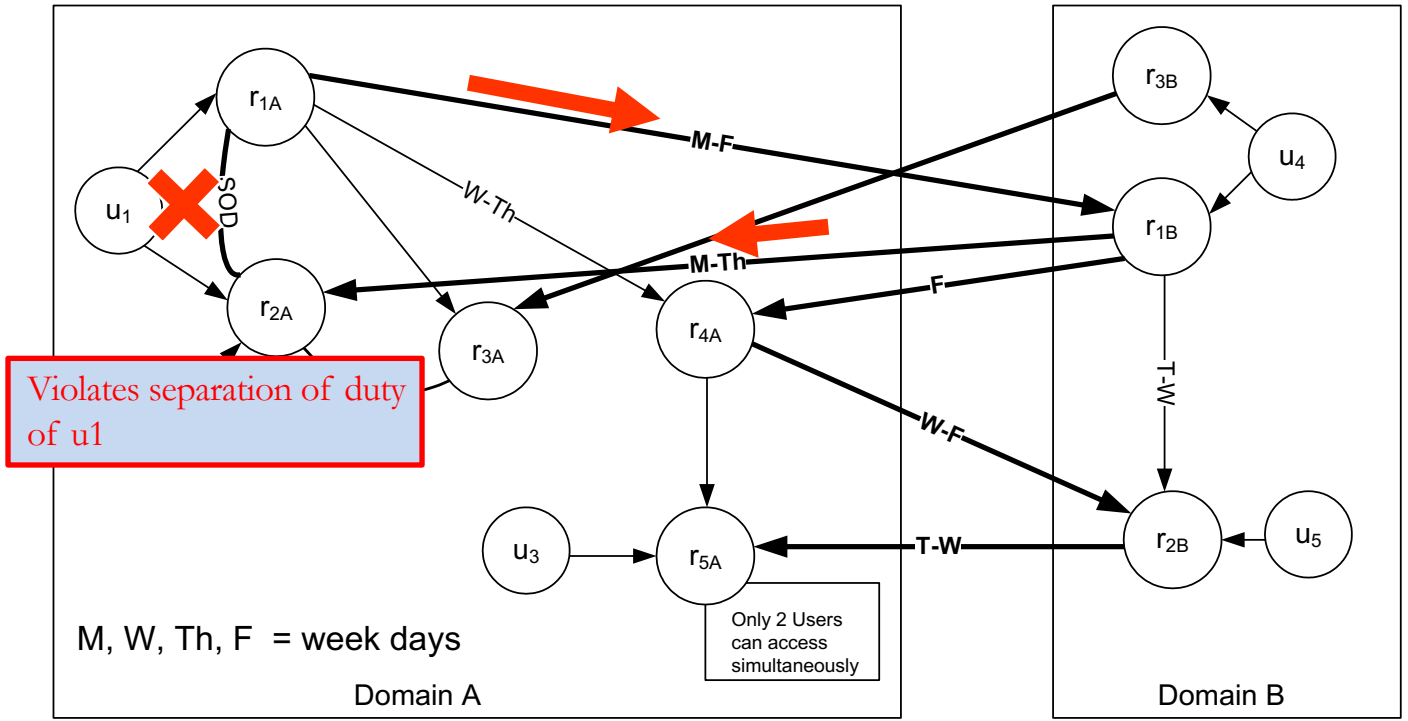
Let directed lines between roles denote role mappings (bold for inter-domain)
Let directed lines between users to roles denote role assignment of a given u_i to a role

Example Policy Composition and Temporal Properties

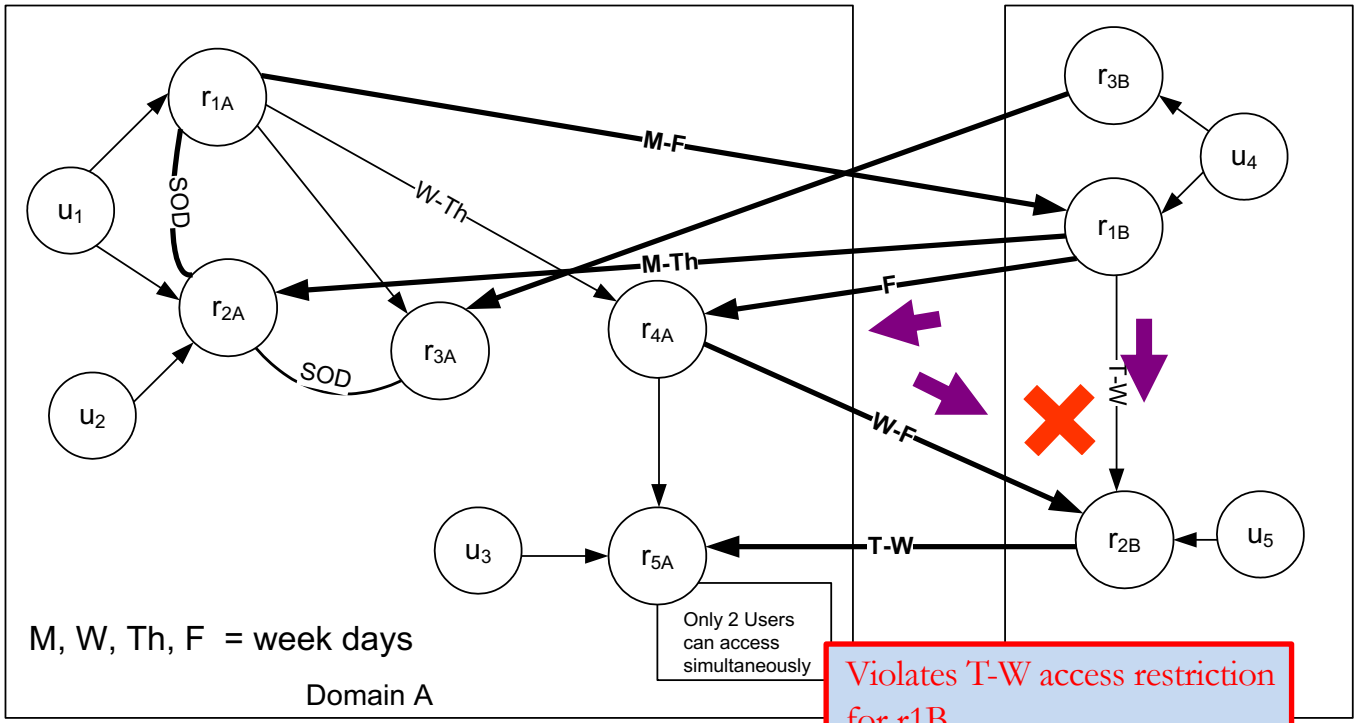


RBAC

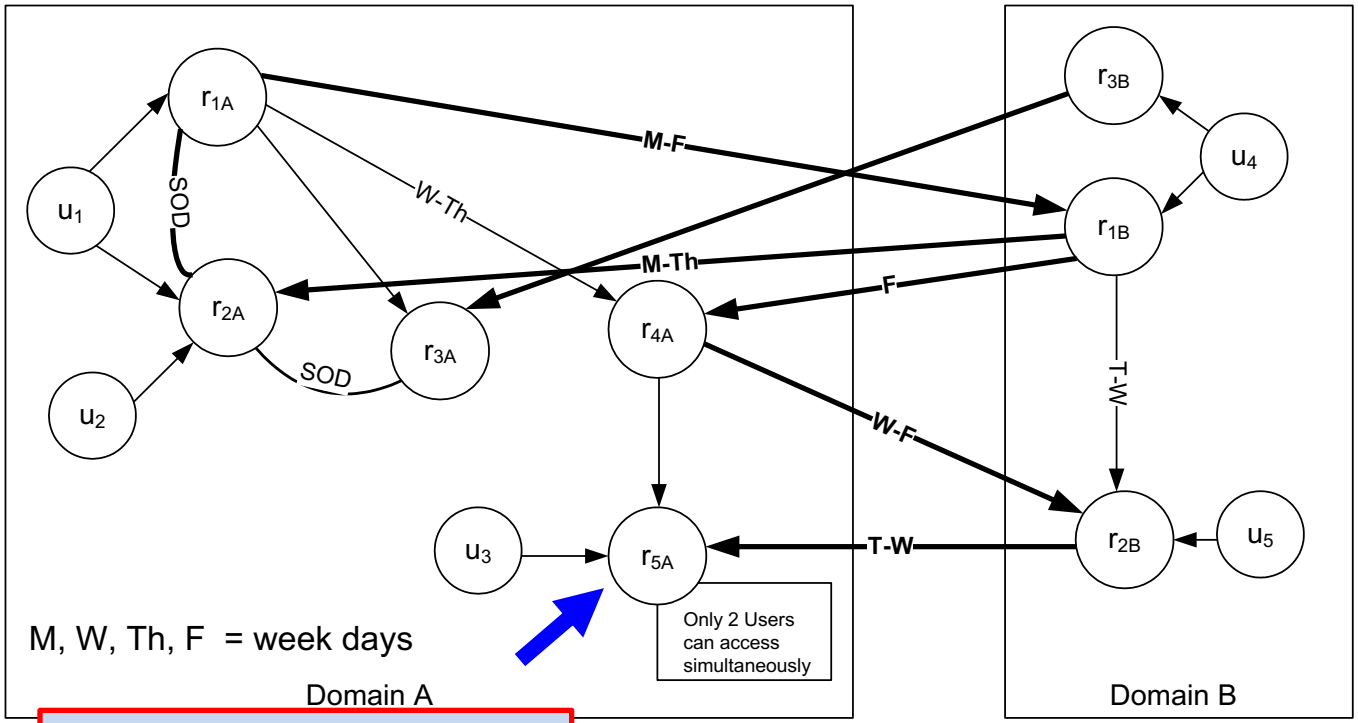
Example Policy Composition and Temporal Properties



Example Policy Composition and Temporal Properties



Example Policy Composition and Temporal Properties



r1A, r1B, and r2B can access on W violating # constraint

RBAC

Looking at NISTIRS, XACML, and metrics

Implement Policy

Monitoring / Audit

NISTIRS

NISTIRS (Interagency Reports)

Another tool in your toolbag

- Describe research or technical information related to information security produced by NIST
- Typically focus on security topics at a much greater level of detail than seen in the SP 800-53 or FIPS series documents
- Are best used in combination with other things like security controls.
- 7874 – focuses on
  "Guidelines for Access Control System Evaluation Metrics"

http://csrc.nist.gov/publications/PubsNISTIRs.html

AC System is initiated here

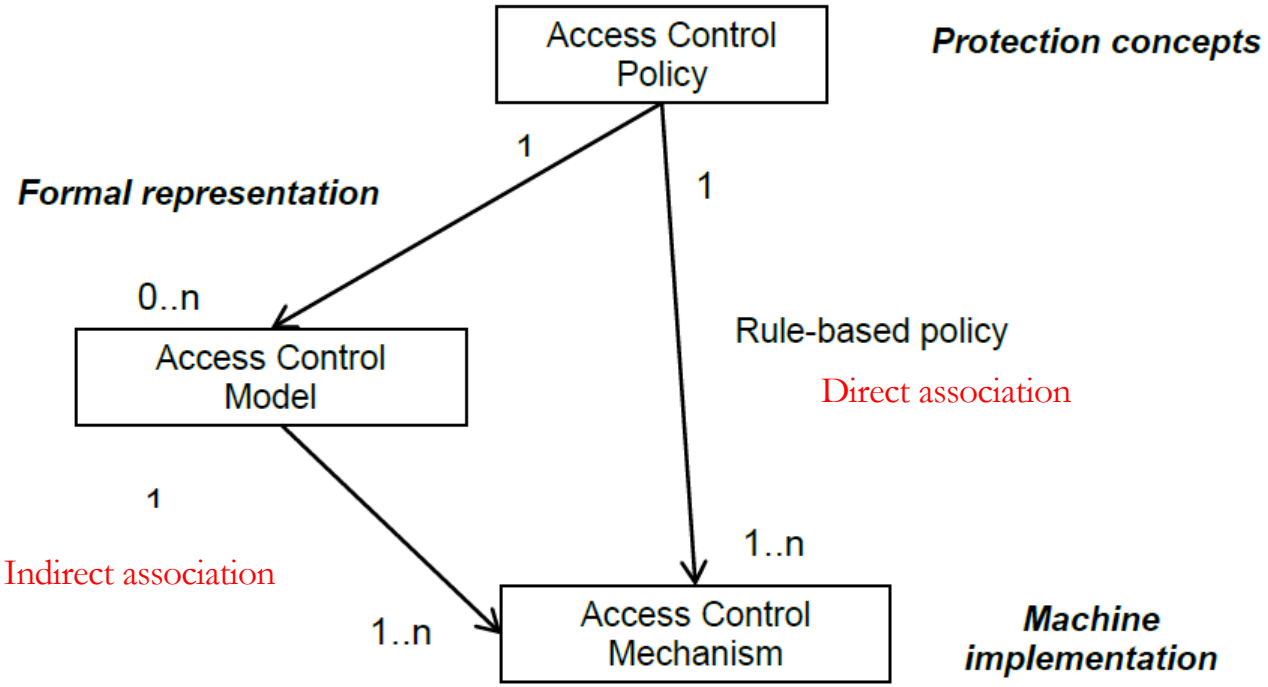


Figure 1 – Mapping of AC policy, model, and mechanism of AC systems

http://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7874.pdf

NIST IRs

Definitions in NIST IR 7874

AC Policies are high level requirements that specify how access is managed and who, under what circumstances, may access what information. To enforce policies, organizations are required to codify their internal privacy and security policies into machine-enforceable algorithms or AC policy languages to govern the exchange of data within their organizations.

AC Models are formal presentations of the security policies enforced by AC systems, and are useful for proving theoretical limitations of systems. AC models bridge the rather wide gap in abstraction between policy and mechanism

AC Mechanisms provide a way to enforce AC policies by translating a user's access request into terms of a system provided structure (e.g. Access control matrix). Access control mechanisms can be designed to adhere to the properties of the model by machine implementation using protocols, architecture, or formal languages such as program code.

RBAC

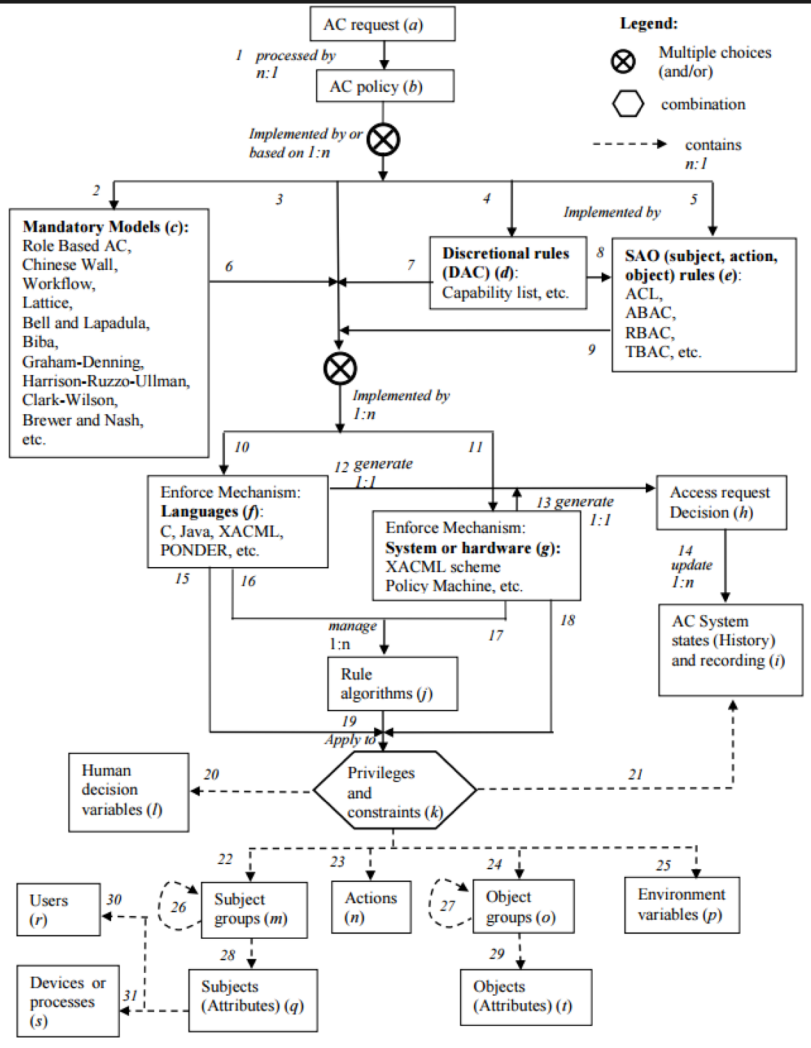


Figure 3 – AC policy ontology

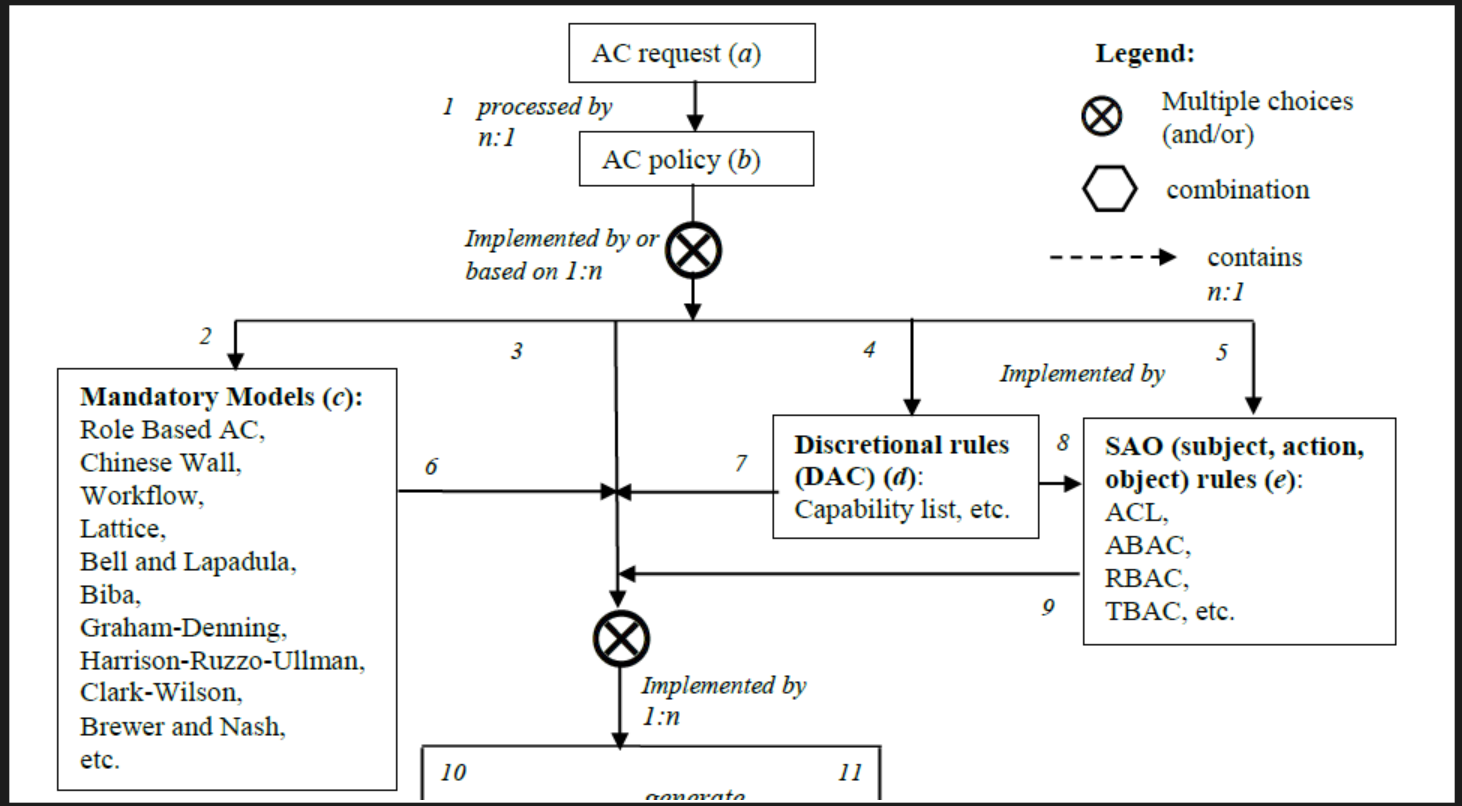
Policy Ontology

onto- comes from the Greek *ὤν, ὄντος* meaning "being" / "that which is"

-logy from Greek *–λογία* meaning "the character of one who speaks of a certain subject" (branch of knowledge)

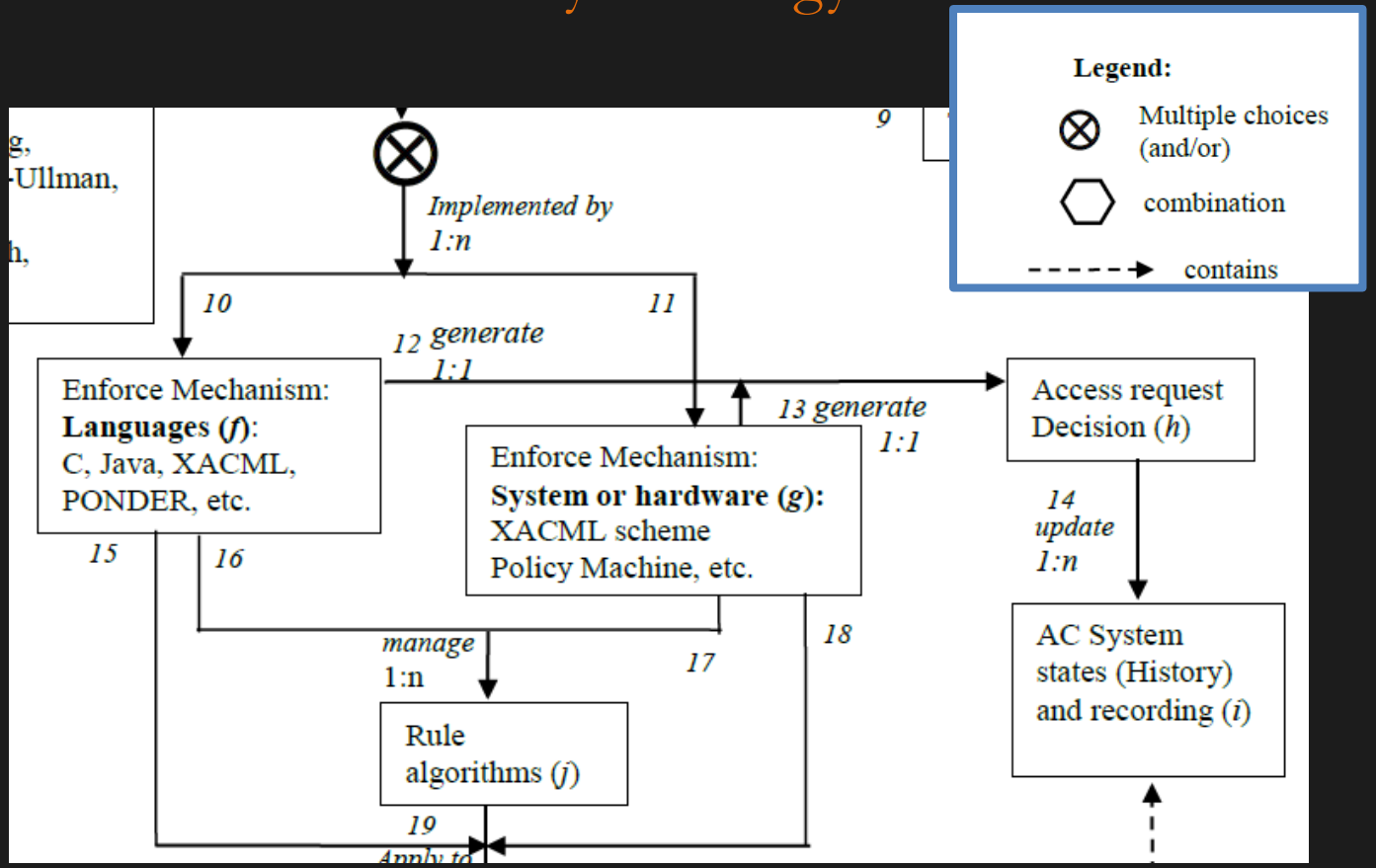
NIST IRs

Policy Ontology



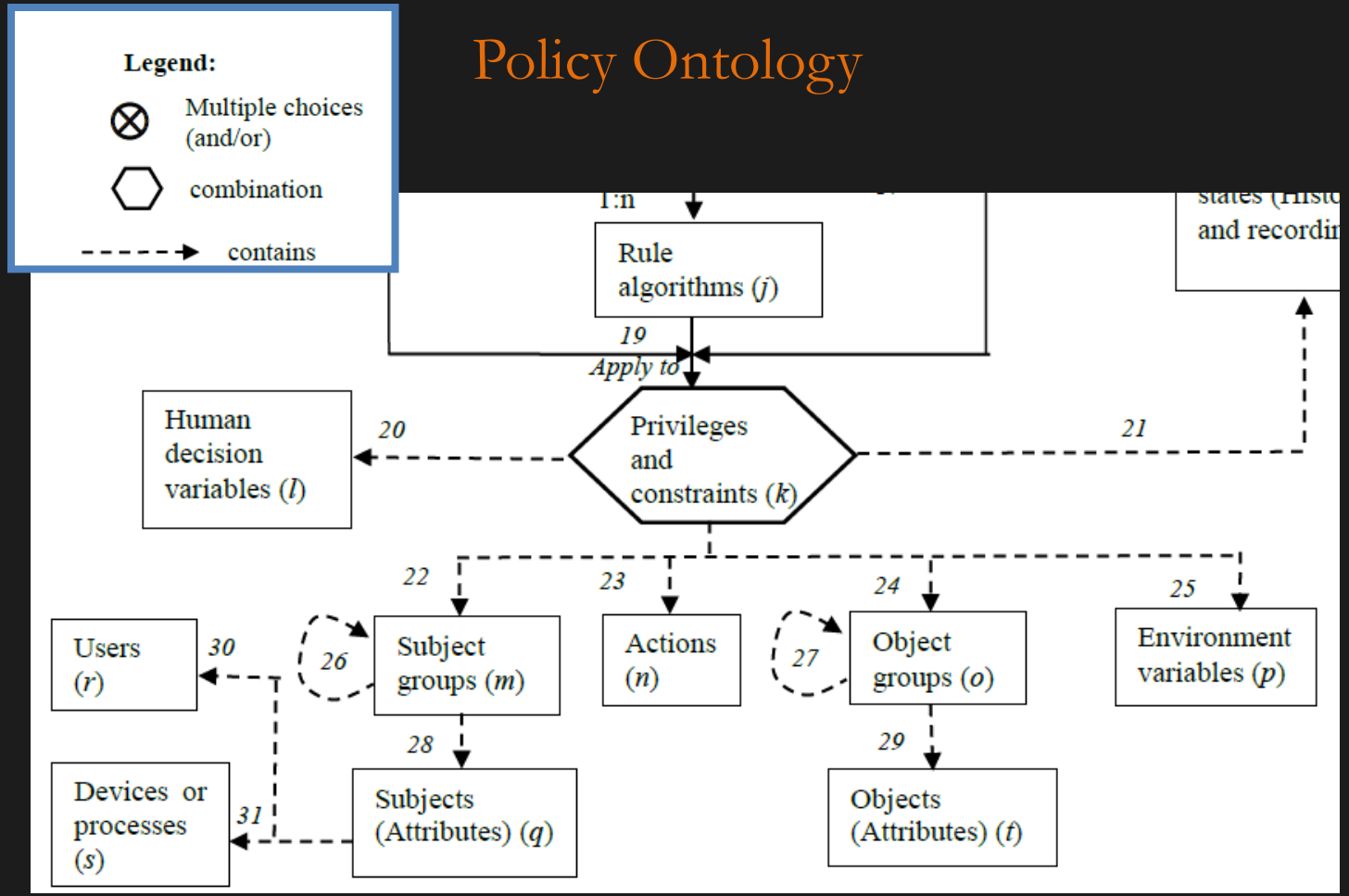
NIST IRs

Policy Ontology



NIST IRs

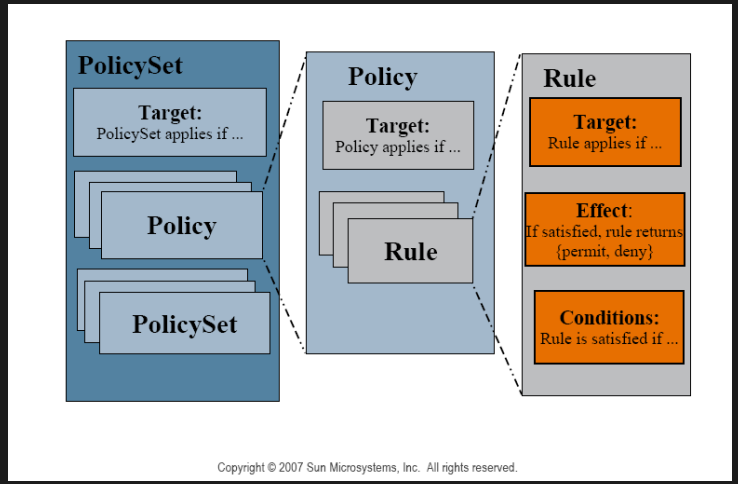
Policy Ontology



NIST IRs

XACML

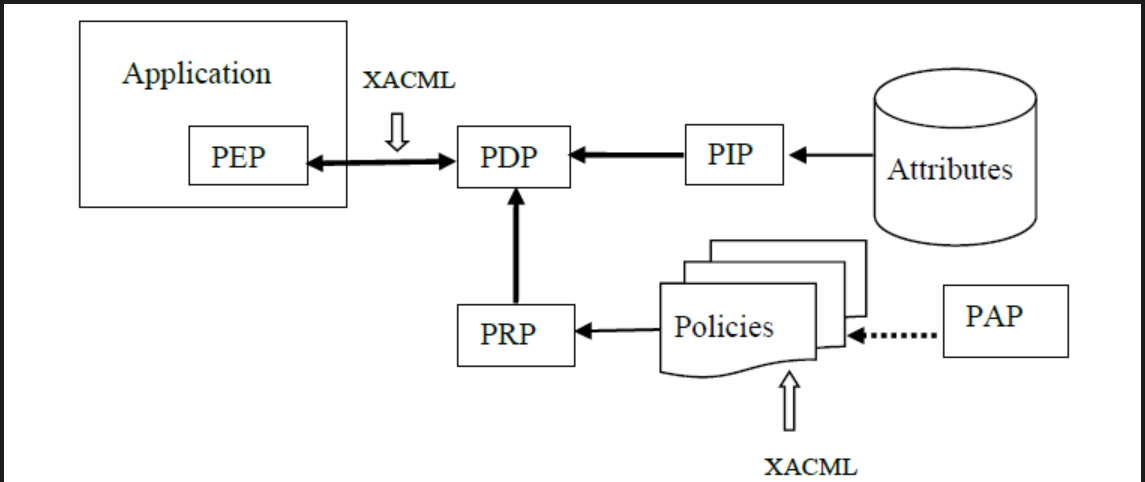
- An authorization-related standard created by the Organization for the Advancement of Structured Information Standards (OASIS)
- XML-based general-purpose language used to describe policies, requests, and responses for AC policies
  - Input: policies, request
  - Output: permit, deny, not applicable, indeterminate
  - Flexible and system-independent representation of access rules that vary in granularities
- Five basic elements of XACML policies
  - *PolicySet* - a container that holds other policies or policy sets
  - *Policy* - policy is expressed through a set of rules
  - *Rule* – implement authorization logic using a target, condition, and effect
  - *Target* – subjects, resources and actions that a rule applies to
  - *Condition* – applies restrictions to the target attributes and refines rule applicability



XACML

XACML Architecture

- **Policy Decision Point (PDP):** Makes the access decisions by evaluating the applicable policy. PDP implements the decision procedures according to the XACML specification.
- **Policy Administration Point (PAP):** Provides a user interface for creating, testing, and debugging XACML policies, and storing these policies in the appropriate repository.
- **Policy Enforcement Point (PEP):** Performs AC by making decision requests made by the PDP and enforcing authorization decisions.
- **Policy Information Point (PIP):** Serves as the source of attribute values, or the data required for policy evaluation to provide the information needed by the PDP to make the decisions.
- **Policy Retrieval Point (PRP):** Where the policies are stored and fetched by the PDP.



XACML

XACML Policy Example

```
<Policy PolicyId="ExamplePolicy"
        RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
   <Description>A policy to specify read privileges on a document called “some-document.pdf”</Description>
   <Target>
      <Resources>
       <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">some-document.pdf</AttributeValue>
          <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
        </ResourceMatch>
       </Resource>
      </Resources>
   </Target>
   <Rule RuleId="ReadRule" Effect="Permit">
                  …
   </Rule>
 </Policy>
```

XACML Policy Example

```
<Rule RuleId="ReadRule" Effect="Permit">
   <Description> Matt can view the document</Description>
      <Target>
            <Actions>
                  <Action>
                     <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">read</AttributeValue>
                           <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
                           AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/>
                     </ActionMatch>
                  </Action>
            </Actions>
      </Target>
      <Condition>
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <SubjectAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" >
               urn:oasis:names:tc:xacml:1.0:subject:subject-id
            </SubjectAttributeDesignator>
            <AttributeValue DataType = "http://www.w3.org/2001/XMLSchema#string">Matt</AttributeValue>
         </Apply>
    </Condition>
</Rule>
```

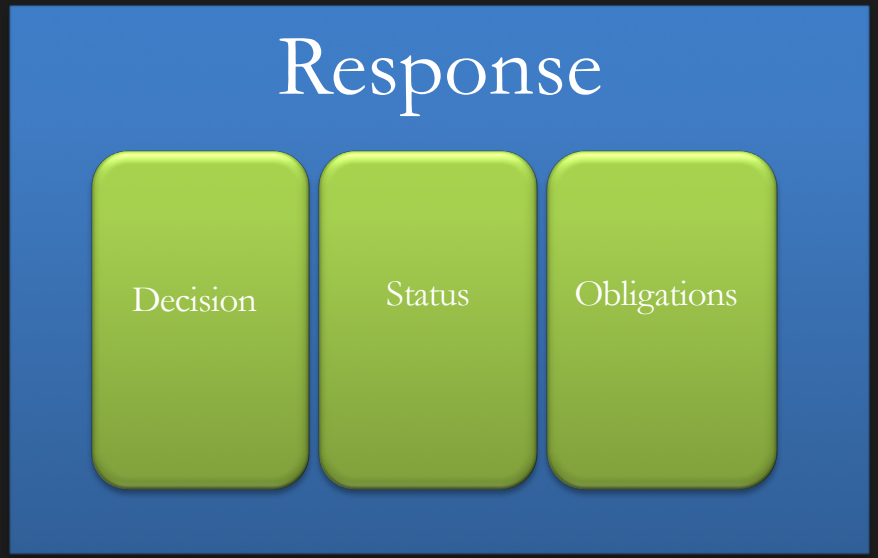
XACML Request Structure

credit:
www.cs.odu.edu/~mukka/cs795sum14.net/Lecturenotes/day7/xacmltutorial.ppt

Request Example

```
<Request>
    <Subject>
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
            DataType=" http://www.w3.org/2001/XMLSchema#string ">
                Matt
        </Attribute>
    </Subject>
    <Resource>
        <Attribute AttributeId = "urn:oasis:names:tc:xacml:1.0:resource:resource-id“
            DataType="http://www.w3.org/2001/XMLSchema#string">some-document.pdf</AttributeValue>
                some-document.pdf
        </Attribute>
    </Resource>
    <Action>
        <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        DataType="http://www.w3.org/2001/XMLSchema#string">
                read
        </Attribute>
    </Action>
</Request>
```

XACML Response Structure



credit:
www.cs.odu.edu/~mukka/cs795sum14.net/Lecturenotes/day7/xacmltutorial.ppt

XACML Response Example

```
<Response>
    <Result>
        <Decision>Permit</Decision>
        <Status>
            <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
    </Result>
</Response>
```

Effect:

Permit/Deny/Not Applicable/Indeterminate

XACML: Benefits/Drawbacks

- Benefits:
  - Allows for combining policies for different authoritative domains into one policy set for making AC decisions in a widely distributed system environment.
  - Reconcile conflicting rules using a collection of combining algorithms Flexible and highly expressive
  - Clear and interchangeable once created
  - system independent
- Drawbacks:
  - extremely verbose – making simple rules many lines long
  - makes first order logic look easy
  - heavy handed for small policy applications
- Its just one tool in the shed – not the only one!

Now that you know what XACML is:
Looking at back at the NISTIRS

NISTIRS

7874 Defines Responsible Principals

- **Organization CIO (Chief Information Officer) (*OC*):** oversee the establishment of information systems from the cost, service, and security perspectives of the organization's policy
- **AC policy authors (*PA*):** define or design security policies for the organization's information system according to business practices and security requirements
- **AC system implementers (*SI*):** install, configure and/or implement the AC system in accordance with the PA's design
- **AC system administrators, (operators, or maintainers) (*SA*):** facilitate building, networking, deploying, administrating, and maintaining the AC system
- **Authentication system managers (*ASM*):** responsible for connecting authentication or other service functions for the AC system
- **AC system users (*SU*):** access information through the AC system

…and Properties for Quality Metrics of Access Control Systems

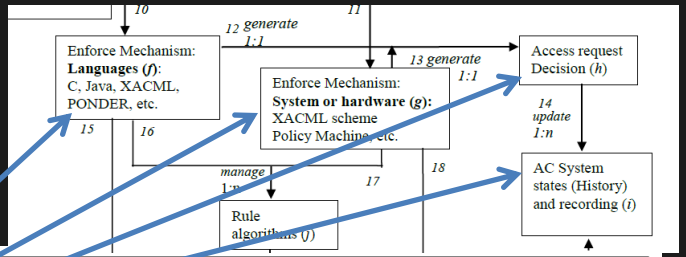
- Categories
  - **Administration** properties impact the cost, efficiency, and performance of an AC system's administration
  - **Enforcement** properties relate to the mechanisms or algorithms that the AC system uses to enforce the embedded AC models and rules - affect the efficiency of rendering AC decisions
  - **Performance** properties are in addition to the enforcement of the AC system's processes
  - **Support** properties may not be essential but can increase the usability and portability of an AC system
- Criticality
  - Questions from the metric items should match the organization's requirements for the AC system.
  - Selected AC metric items are categorized as
    - *Critical* - are necessary for the system
    - *Optional* - desirable but not essential (e.g., improve performance)
    - *Supplemental* - will not affect the normal AC operation, but might be required for extension or future services.

Administration Properties

- Auditing
- Privileges/capabilities discovery
- Ease of privilege assignments
- Syntactic and semantic support for specifying AC rules
- Policy management
- Delegation of administrative capabilities
- Flexibilities of configuration into existing systems
- The horizontal scope (across platforms and applications) of control
- The vertical scope (between application, DBMS, and OS) of control

Administration – Function: Auditing

- **Organization CIO (*OC*):** oversee the establishment of information systems from the cost, service, and security perspectives of the organization's policy
- **AC system administrators (*SA*):** facilitate building, networking, deploying, administrating, maintaining the AC system
- **Policy Enforcement Point (*PEP*):** Performs AC by making decision requests made by the PDP and enforcing authorization decisions.



Responsible principals: ***OC*** PA SI ***SA*** ASM SU	
Required policy ontology elements: a b c d e ***f*** ***g*** ***h*** ***i*** j k l m n o p q r s t 1 2 3 4 5 6 7 8 9 10 11 ***12*** ***13*** ***14*** 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 N/A	
Applicable XACML architecture components: ***Application*** ***PEP*** PDP PRP PIP PAP N/A	
Metric Items to Evaluate	**Description**
☐ Does the AC system log system failure?	Log for source of errors records when the AC system fails to make grant decisions.
☐ Does the AC system log denied access requests?	Log for attempted policy violations records the denied user request with respect to the AC policies involved.
☐ Does the AC system log granted access requests?	Log for access tracking records the granted capabilities of a subject. Because objects can be renamed, copied, and given away, tracking the dissemination and retention of access is difficult or impossible to achieve through privilege expressions alone.
☐ Does the AC system provide additional log functions required by the organization?	Customize the audit information-providing capabilities for managing log data (e.g., set the maximum size of audit logs).

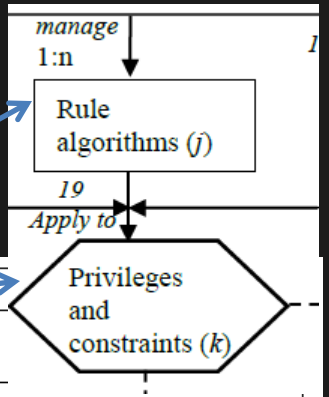
NIST IRs: Metrics

Enforcement Properties

- Policy combination, composition, and constraint
- Bypass
- Separation of Duty (SoD)
- Safety (confinements and constraints)
- Conflict resolution or prevention
- Operational/situational awareness
- Granularity of control
- Expression (policy/model) properties
- Adaptable to the implementation and evolution of AC policies

Enforcement – Function: *Conflict Resolution/Prevention*

- **AC policy authors (*PA*):** define or design security policies for the organization's information system according to business practices and security requirements
- **AC system implementers (*SI*):** install, configure and/or implement the AC system in accordance with the PA's design
- **Policy Administration Point (*PAP*):** Provides a user interface for creating, testing, and debugging XACML policies, and storing these policies in the appropriate repository.



Responsible principals: OC **PA** **SI** SA ASM SU

Required policy ontology elements: a b c d e f g h i **j** **k** l m n o p q r s t 1 2 3 4 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 N/A

Applicable XACML architecture components: Application PEP PDP PRP PIP **PAP** N/A

Metric Items to Evaluate	Description
☐ Is the AC system capable of preventing policy rule conflicts? ☐ Is the AC system capable of resolving conflict policy rules? ☐ Is the AC system capable of preventing policy conflicts (if multiple policies enforcement is available)? ☐ Is the AC system capable of resolving policy conflicts (if multiple policies enforcement is available)?	Policy rule conflicts appear when the specifications of two or more access rules result in the conflict decision of granting a subject's access request by either direct or indirect access assignments. Policy rule conflicts can also be a result of the deadlock of access rules specification. Deadlock can be defined as: a *rule r* has a dependency on other *rule(s)*, which eventually depend back on *r* itself such that the subject's request will never reach a decision because of the cyclic referencing. In addition to policy rules, when multiple policies are evoked for granting permission, conflicts of policy may occur between policy *X* and policy *Y*. To support conflict resolution, an AC system may provide automatic conflict identification with suggested corrections.

Performance Properties

- Response time
- Policy repository and retrieval
- Policy distribution
- Integrated with authentication function

Performance – Function: *Policy Repository & Retrieval*

- **Organization CIO (*OC*):** oversee the establishment of information systems from the cost, service, and security perspectives of the organization's policy
- **AC system administrators (*SA*):** facilitate building, networking, deploying, administrating, maintaining the AC system
- **Authentication system managers (*ASM*):** responsible for connecting authentication or other service functions for the AC system
- **Policy Administration Point (PAP):** Provides a user interface for creating, testing, and debugging XACML policies, and storing these policies in the appropriate repository.
- **Policy Information Point (PIP):** Serves as the source of attribute values, or the data required for policy evaluation to provide the information needed by the PDP to make the decisions.
- **Policy Retrieval Point (PRP):** Where the policies are stored and fetched by the PDP.

Responsible principals: ***OC*** PA SI ***SA*** ***ASM*** SU	
Required policy ontology elements: a b c d e f g h i j k l m n o p q r s t 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 ***N/A***	
Applicable XACML architecture components: Application PEP PDP ***PRP*** ***PIP*** ***PAP*** N/A	
Metric Items to Evaluate	**Description**
☐ Does the AC policy retrieval and deposit meet the organization's safety, operation and cost requirements?	An AC system may store and retrieve AC policies in different forms of repositories. Policies can be stored and retrieved in local, global, federated, or subscribed (e.g., grid and cloud environment) repositories. Also, some AC policies might require connecting to multiple repositories simultaneously. It is important to balance the cost of hardware and software with efficiency based on the organization's requirements.

NIST IRs: Metrics

Support Properties

- Policy import and export
- OS compatibility
- Policy source management
- User interfaces and API
- Verification and compliance function support

Support –Function: *Policy Source Management*

- **AC system implementers (*SI*):** install, configure and/or implement the AC system in accordance with the PA's design
- **AC system administrators (*SA*):** facilitate building, networking, deploying, administrating, maintaining the AC system
- **Authentication system managers (*ASM*):** responsible for connecting authentication or other service functions for the AC system
- **Policy Administration Point (PAP):** Provides a user interface for creating, testing, and debugging XACML policies, and storing these policies in the appropriate repository.
- **Policy Information Point (PIP):** Serves as the source of attribute values, or the data required for policy evaluation to provide the information needed by the PDP to make the decisions.

Responsible principals: OC PA ***SI*** ***SA*** ***ASM*** SU	
Required policy ontology elements: a b c d e f g h i j k l m n o p q r s t 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 ***N/A***	
Applicable XACML architecture components: ***Application*** PEP PDP PRP ***PIP*** ***PAP*** N/A	
Metric Items to Evaluate	**Description**
☐ Does the AC system provide a function to maintain or manage the source and destination of AC policies?	For the integrity of the resource, some AC systems, especially those that handle multiple AC policies, are required to manage the authoritative policy source(s). Management functions include identifying authoritative sources of record (creator), identifying authoritative sources of reference (distributor), establishing authoritative source authority, delegating authoritative source authority, updating existing authoritative source authority, terminating authoritative source authority, and maintaining an enterprise authoritative source authority. It is also required to use secure communication channel functions if information exchange among these management functions is necessary.

Next Time

No Class

Then

Practice with XACML and System State Modeling

Security Training, Awareness, and Social engineering

Then

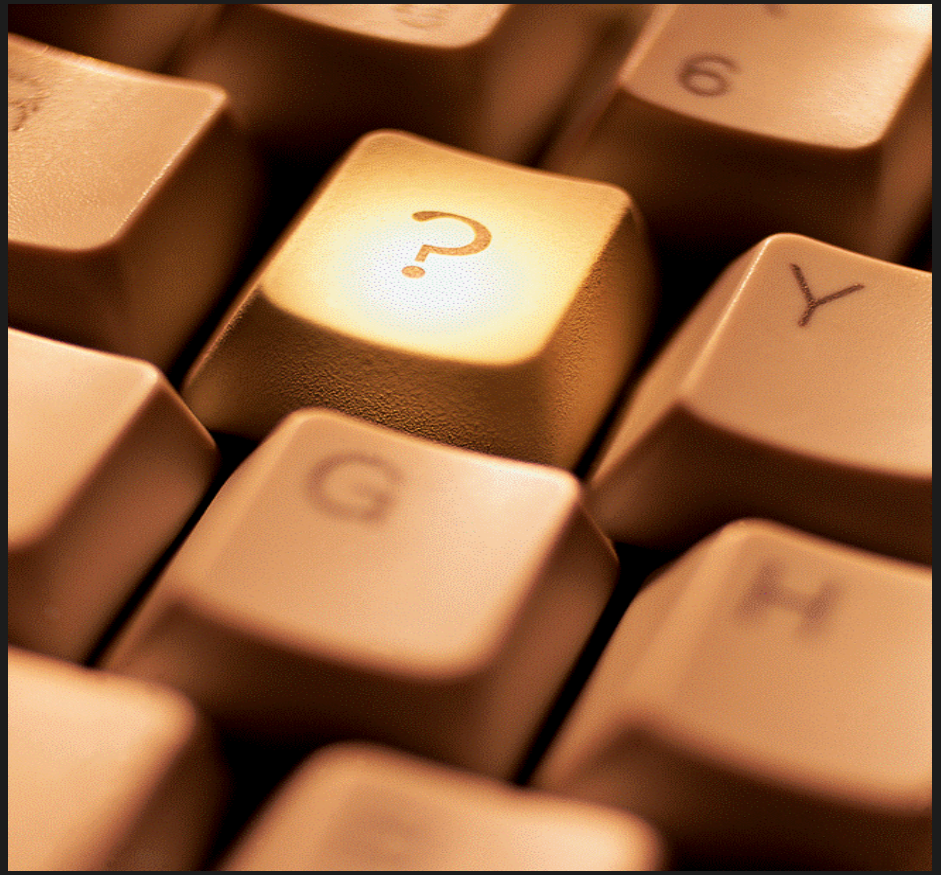
Security Training and Awareness

Homework

Read (i.e. glance) at the NISTIR 7874

http://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7874.pdf

Questions?



Matt Hale, PhD

University of Nebraska at Omaha

Interdisciplinary Informatics

mlhale@unomaha.edu

Twitter: @mlhale_