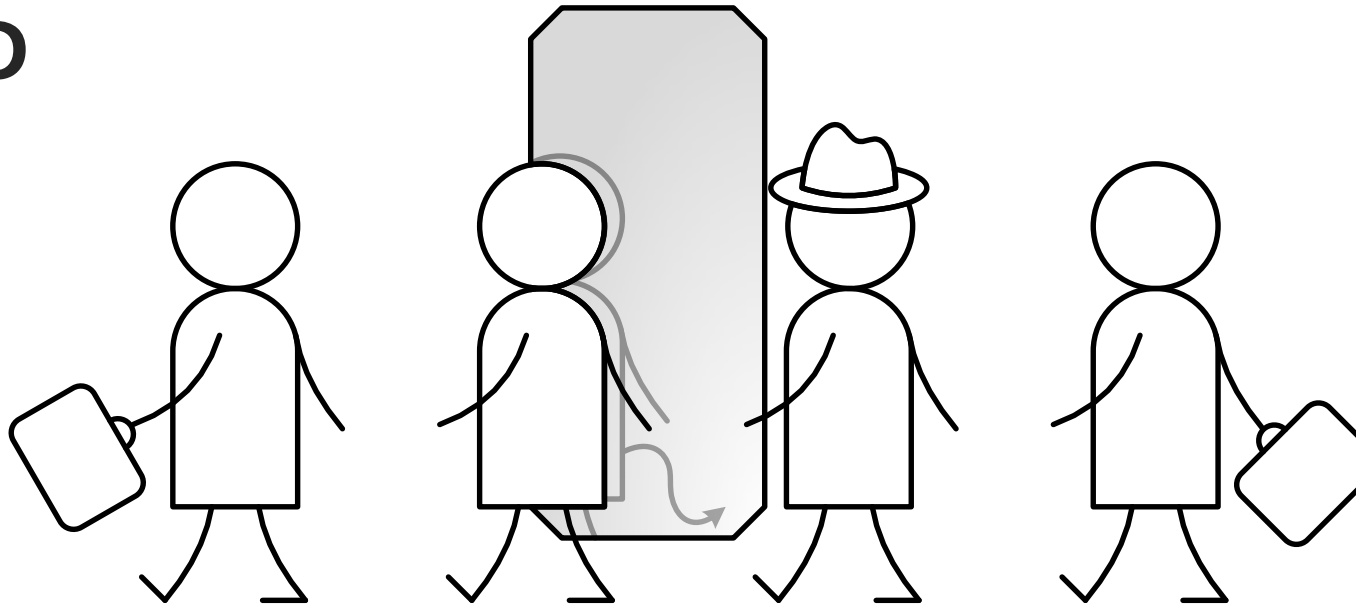
WE HAVE MET THE ENEMY AND HE IS US



UC DAVIS
MATT BISHOP
SOPHIE ENGLE
SEAN PEISERT
SEAN WHALEN

CA LABS
CARRIE GATES

LAKE TAHOE, CA
NSPW 09.23.2008

WHAT WE SAW	Binary, perimeter-based definition of insiders hinder threat analysis
WHAT WE SHOW	How to define and analyze the insider problem
WHAT WE DON'T SHOW	How to detect, deter, mitigate, or solve the insider problem
WHY IT'S IMPORTANT	Identifies highest-risk resources and highest-threat insiders

NAVIGATION

Main Sections:

- Part 1: Unifying Policy Hierarchy
- Part 2: Existing Insider Definitions
- Part 3: Attribute-Based Group Access Control

Supplemental:

- Definitions

PART 1

Understanding Insiders and Insider Threat

CLAIMS

- **The complexity of security policy is key to understanding the insider problem.**
- **Binary or perimeter-based definitions of an insider impede threat analysis.**
- **The ABGAC model identifies “insiderness” with respect to a resource and allows for insider threat analysis.**

SECURITY POLICY

The Complexities

POLICY EXAMPLE

The Scenario:

- Yasmin, a doctor, is only authorized to read and append medical records of her patients for the purpose of treating them.

POLICY EXAMPLE

The Scenario:

- Yasmin, a doctor, is only authorized to read and append medical records of her patients for the purpose of treating them.

The Ideal Policy:

- Yasmin is authorized to read {…} records for the purpose of treating {…} patients.
- Yasmin is authorized to append {…} records for the purpose of treating {…} patients.

Feasible?

POLICY EXAMPLE

The Scenario:

- Yasmin, a doctor, is only authorized to read and append medical records of her patients for the purpose of treating them.

The ~~Ideal~~ Policy:

- Yasmin is authorized to authenticate as `yasmin`.
- `yasmin` is authorized to read {···} records.
- `yasmin` is authorized to append {···} records.

POLICY EXAMPLE

The Scenario:

- Yasmin, a doctor, is only authorized to read and append medical records of her patients for the purpose of treating them.

The ~~Ideal~~ Policy:

- Yasmin is authorized to authenticate as `yasmin`.
- `yasmin` is authorized to read {···} records.
- `yasmin` is authorized to append {···} records.

Practical?

POLICY EXAMPLE

The Scenario:

- Yasmin, a doctor, is only authorized to read and append medical records of her patients for the purpose of treating them.

The ~~Ideal~~ Policy:

- Yasmin is authorized to authenticate as `yasmin`.
- `yasmin` is authorized to read all records.
- `yasmin` is authorized to write all records.

Possible?

POLICY EXAMPLE

The Scenario:

- Yasmin, a doctor, is only authorized to read and append medical records of her patients for the purpose of treating them.

The ~~Ideal~~ Policy:

- Yasmin is authorized to authenticate as `yasmin`.
- `yasmin` is authorized to read all records.
- `yasmin` is authorized to write all records.
- **`yasmin` can delete all records.** ← **Exploit!**

POLICY EXAMPLE

The Scenario:

- Yasmin, a doctor, is only authorized to read and append medical records of her patients for the purpose of treating them.

The Different Policies:

- What is ideal?
- What is feasible?
- What is practical?
- What is possible?

SECURITY POLICY

The Unifying Policy Hierarchy

UNIFYING POLICY HIERARCHY

What is the Unifying Policy Hierarchy?

- Introduced by Carlson in 2006:
  - Carslon, Adam, “The Unifying Policy Hierarchy Model,” Master’s Thesis, UC Davis, June 2006.
- A hierarchical model of security policy at different levels of abstraction.

What is it good for?

- Analyzing gaps in the hierarchy lead to insight to where and why problems occur

EXAMPLE SCENARIO

The Scenario:

- Yasmin, a doctor, is only authorized to read and append medical records of her patients for the purpose of treating them.

EXAMPLE SCENARIO

Oracle Policy *(Ideal)*

OP(subject, object, action, environment/intent **) =**
{ *authorized*, *unauthorized* **}**

OP(*s,o,a,e*) = *authorized*

- Yasmin, `yasmin`, authenticate, any
- `yasmin`, {···} records, read, treating {···} patients
- `yasmin`, {···} records, append, treating {···} patients

EXAMPLE SCENARIO

Feasible Policy *(Feasible)*

FP(subject, object, action **) =**
{ *authorized*, *unauthorized, unknown* **}**

- **FP(** `yasmin`, {···} records, read **) =** *authorized*
- **FP(** `yasmin`, {···} records, append **) =** *authorized*

- **FP(** Yasmin, `yasmin`, authenticate **) =** *unknown*
- **FP(** Xander, `yasmin`, authenticate **) =** *unknown*

EXAMPLE SCENARIO

Configured Policy *(≈Practical)*

CP(subject, object, action **) =**
{ *authorized*, *unauthorized, unknown* **}**

- **FP(** `yasmin`, {···} records, read **) =** *authorized*
- **FP(** `yasmin`, {···} records, append **) =** *authorized*

- **CP(** `yasmin`, all records, read **) =** *authorized*
- **CP(** `yasmin`, all records, write **) =** *authorized*

EXAMPLE SCENARIO

Real-Time Policy *(Possible)*

RP(subject, object, action **) =**
{ *possible*, *impossible* **}**

- **OP(** Xander, `yasmin`, authenticate **) =** *unauthorized*
- **CP(** `yasmin`, all records, delete **) =** *unauthorized*

- **RP(** Xander, `yasmin`, authenticate **) =** *possible*
- **RP(** `yasmin`, all records, delete **) =** *possible*

POLICY GAPS

Oracle/Feasible Gap

- Technology Limitations
  Ex: user versus user account, user intent

Feasible/Configured Gap

- Configuration Errors
  Ex: slow removal of terminated employees

Configured/Real-Time Gap

- Implementation Errors and Vulnerabilities
  Ex: buffer overflow, runtime vulnerability

POLICY GAPS

Action	OP	FP	CP	RP
Xander authenticates as `xander`.	✓	?	?	✓
`xander` accesses a website…	✗	✓	✓	✓
…to check the weather	✓	?	?	✓
…to expose system to exploit	✗	?	?	✓
Web browser leaks user password	✗	✗	✗	✓
Yasmin authenticates as `xander`.	✗	?	?	✓

UNIFYING POLICY HIERARCHY

Understanding Insiders and Insider Threat

DEFINITIONS

Who are the Insiders?

- Anyone with more privileges in a lower level of policy than at a higher level of policy.

What is the Insider Problem?

- Insiders have more permissions than necessary to perform their jobs.
- Insiders must be trusted not to misuse these permissions for other purposes.

PRIMITIVE INSIDER MISUSES

- **Violate OP using privileges in CP or FP**
  - *Ex: Misuse privileges for perso*

"Legitimate" Access Misuse

- **Violate FP using privileges in CP**
  - *Ex: Fired employee logs on and*

Assume FP = CP?

- **Violate CP using privileges in RP**
  - *Ex: Exploit buffer overflow insid increase privileges.*

"Illegitimate" Access Misuse

EXAMPLE OF INSIDER MISUSE

Scenario:

Yasmin sells information from all medical records to insurance companies.

- Intent unauthorized in OP
- Intent unrecognized in FP
- Access to all records unauthorized in FP
- Access to all records authorized in CP

Potential for misuse!

INSIDERNESS

Definition:

- A “measure” of an insider’s potential for misuse
- Loosely based on “size of gaps” for an insider

Example:

- Programmer with read and commit access to svn for a specific project
- System administrator for SVN with root access for all company projects

WHAT DO WE LEARN?

There are different categories of insider misuse

- OP/CP Misuse (Legitimate Privilege Misuse)
- CP/RP Misuse (Illegitimate Privilege Misuse)

Insider misuse is not always linked to cyber access

- Some misuse occurs at higher levels of the hierarchy.
- Some misuse is the result of social or physical factors.
- *The Insider Problem predates computers anyway!*

WHAT DO WE LEARN?

Some insiders have higher degree of "insiderness"

- How big are the gaps?
- How much access does the insider have?
- How do we measure or capture "insiderness"?

We need to perform insider threat analysis!

PART 2

Existing Definitions of Insiders

CLAIMS

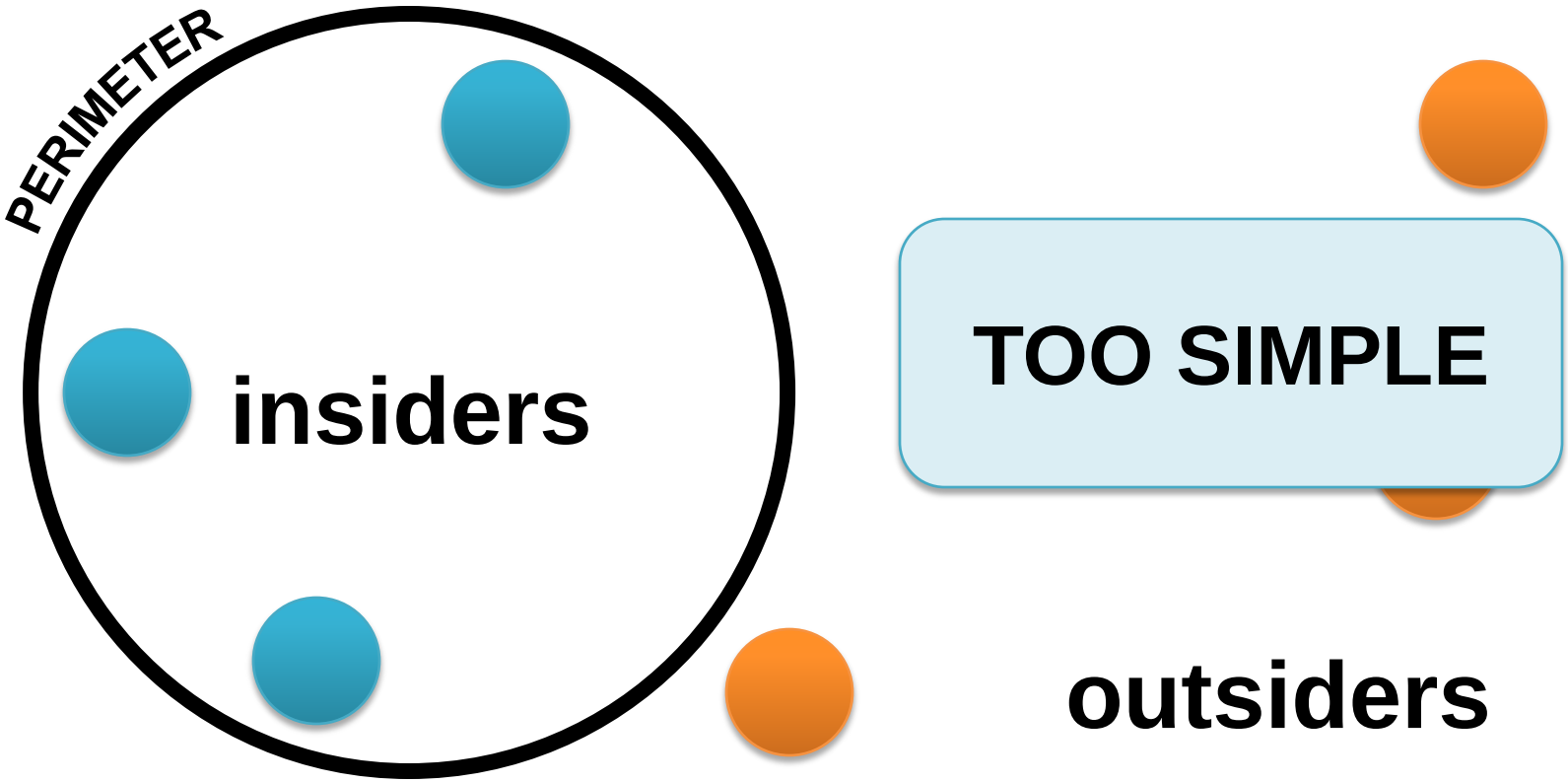
- **The complexity of security policy is key to understanding the insider problem.**
- **Binary or perimeter-based definitions of an insider impede threat analysis.**
- **The ABGAC model identifies “insiderness” with respect to a resource and allows for insider threat analysis.**

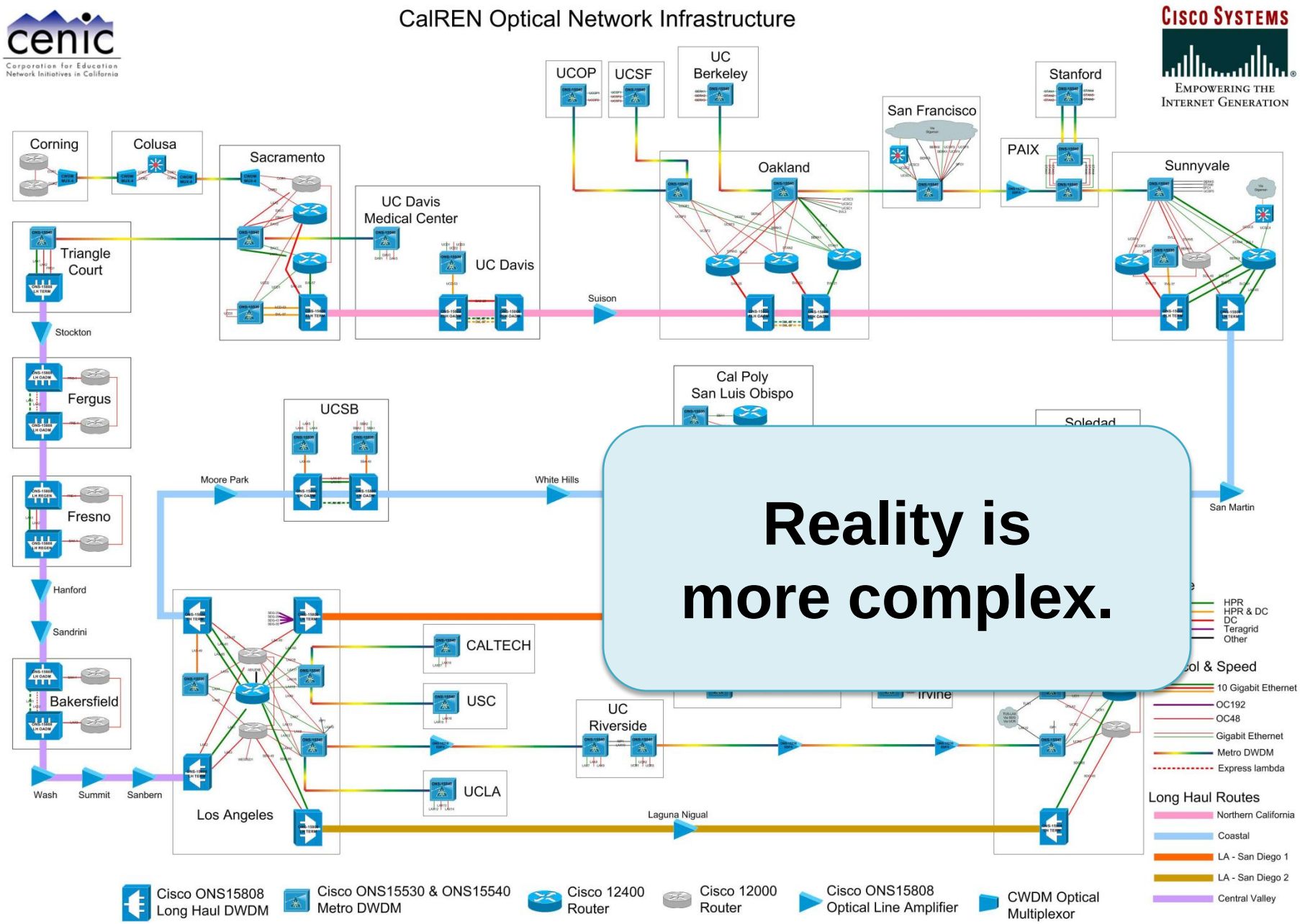
EXISTING DEFINITIONS

Insider:

Anyone operating inside the security perimeter.

(*Patzakis, "New Incident Response Best Practices," 2003.)*





http://www.cenic.net/operations/documentation/CENIC-Design.jpg

INSIDER

Someone with access, privileges, or knowledge of information systems and services.
(RAND, "Understanding the Threat," 2004.)

Binary Classification

- Insider(Name) = { Yes, No }

- Xander, has access and knowledge
- Yasmin, has just knowledge
- Insider(Xander) = Insider(Yasmin) = Yes

INSIDER

Someone with access, privileges, or knowledge of information systems and services.

(RAND, "Understanding the Threat," 2004.)

What type of access?

- Cyber only?
- Saw how other types of access lead to insider problems in the policy hierarchy

OUR APPROACH

OUR APPROACH

Avoid perimeters

- Define an insider with respect to a resource

Avoid binary classification

- Assign “insiderness” based on level of access

Avoid cyber-only access

- Include physical, cyber, and social access
- Include subjects, objects, actions from Oracle Policy

PART 3

Identifying Insiders and Analyzing Insider Threat

CLAIMS

- **The complexity of security policy is key to understanding the insider problem.**
- **Binary or perimeter-based definitions of an insider impede threat analysis.**
- **The ABGAC model identifies “insiderness” with respect to a resource and allows for insider threat analysis.**

ACCESS CONTROL

Identifying Insiders

USING RBAC

Definition:

- Role-Based Access Control
- Create roles based on job function
- Assign permissions to roles
- Assign roles to users

Usage:

- Identify all roles with access to resource
- Identify all users with those roles

RBAC SCENARIO

	Attribute		
Name	Job Function	Building Access	Server Access
Wilma	System Admin	Before 5pm	Both
Xander	Help Desk	After 5pm	Remote
Yasmin	Janitor	Before 5pm	Physical
Zane	Janitor	After 5pm	Physical

RBAC SCENARIO

	Attribute		
Name	Job Function	Building Access	Server Access
Wilma	System Admin	Before 5pm	Both
Xander	Help Desk	After 5pm	Remote
Yasmin	Janitor	Before 5pm	Physical
Zane	Janitor	After 5pm	Physical

Insiders With:	Remote access to servers.
RBAC Role:	System Admin, Help Desk

RBAC SCENARIO

	Attribute		
Name	Job Function	Building Access	Server Access
Wilma	System Admin	Before 5pm	Both
Xander	Help Desk	After 5pm	Remote
Yasmin	Janitor	Before 5pm	Physical
Zane	Janitor	After 5pm	Physical

Insiders With:	Physical access after 5pm
RBAC Role:	Janitor

RBAC SCENARIO

	Attribute		
Name	Job Function	Building Access	Server Access
Wilma	System Admin	Before 5pm	Both
Xander	Help Desk	After 5pm	Remote
Yasmin	Janitor	Before 5pm	Physical
Zane	Janitor	After 5pm	Physical

Insiders With:	Physical access before 5pm
RBAC Role:	*Unclear*

ABGAC

Attribute-Based Group Access Control

INTRODUCING ABGAC

Attribute-Based Group Access Control

- Generalization of RBAC
- Assigns rights based on general attributes, which may or may not include job function
- Inherits features of RBAC such as:
  - “role containment” as “group containment”
  - “separation of duty” becomes “conflicts of interest”

CONFLICTS OF INTEREST

Scenario:

- Xander, an executive at a company, is married to Yasmin.
- Xander has insider information that company stock will increase.
- There is a conflict of interest if Xander advises Yasmin to invest.

Groups:

- Group 1: Those given the insider information.
- Group 2: Those related to group 1.

Separation:

- Members of group 2 are forbidden to do anything forbidden to members of group 1.

ABGAC

Building Blocks

RESOURCE PAIR

Definition:

A pair consisting of a resource (entity) and an access mode describing one way in which that entity can be accessed.

** *Access mode not restricted to cyber access!*

The resource or access may come from *any* level in the policy hierarchy.

RESOURCE PAIR

Example:

(backups, erase) : ability to erase backup files

Access includes anyone with:

- Privileges to delete files on the server
- Physical access to the hard drive

- Include what is *possible* (RP) not *authorized* (CP+)

RESOURCE DOMAIN

Definition:

A set of resource pairs.

(similar to a protection domain, but includes physical, procedural, and cyber access and resource-oriented)

Example:

{ (backups, modify), (backups, erase) }

RD-GROUP

Definition:

A set of (one or more) resource domains.

(can group domains required for multi-stage attacks, or domains with similar risk values)

Example:

{ { (backups, modify), (backups, erase) },
{ (servers, login), (servers, configure) } }

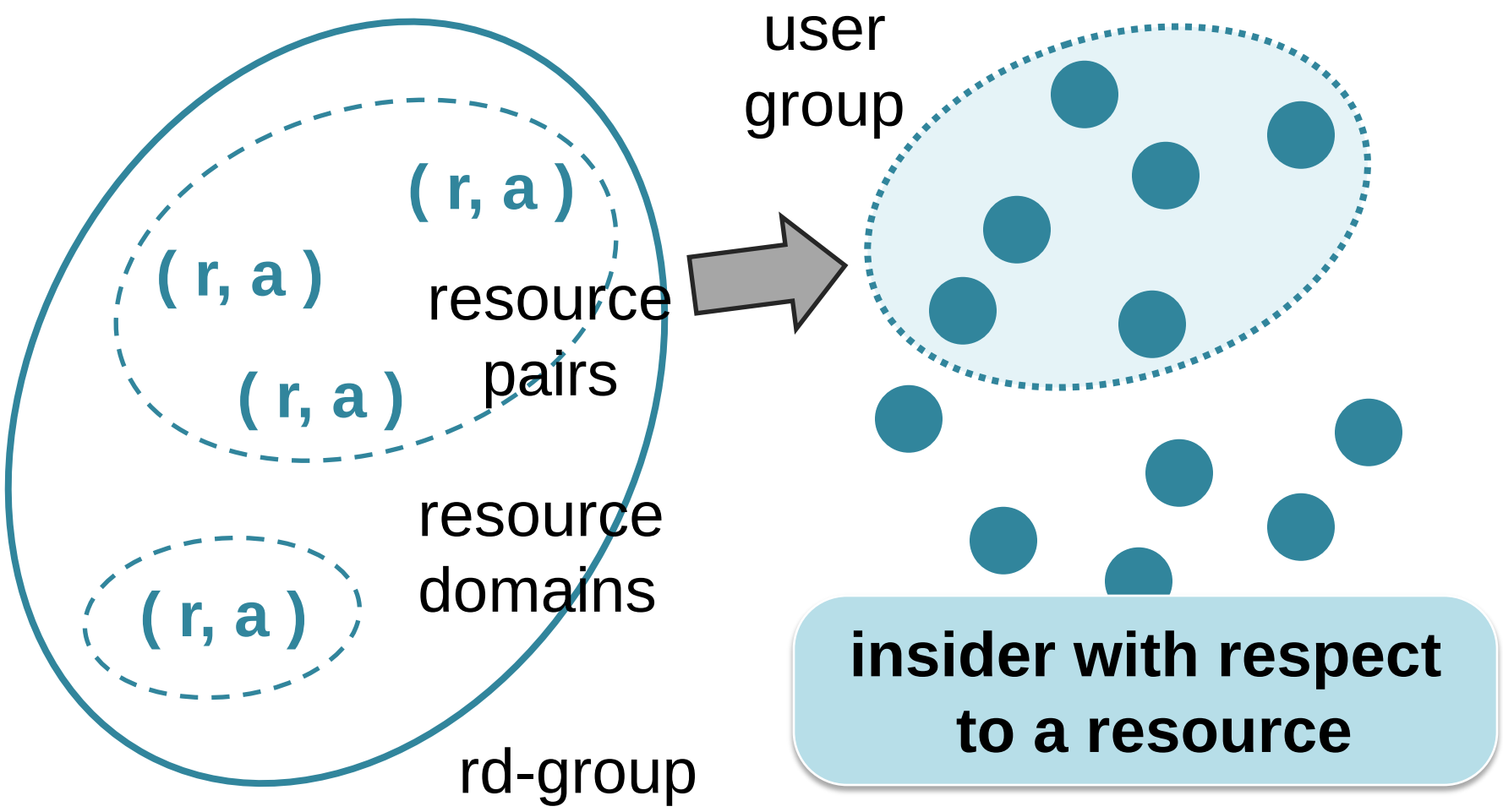
USER GROUP

Definition:

The set of all subjects whose protection domains are a (possibly improper) superset of the associated **rd-group**.

** *Protection domain is used broadly to include possible access from cyber, physical, and social domains.*

ABGAC BUILDING BLOCKS



ANALYZING THREAT

A Simplified Example

ANALYZING THREAT

General Goals:

- Minimize impact of an insider attack
- Minimize number of known insiders

General Approach:

- Provide an ordering of resource domains
- Results in ordering of rd-groups
- Identify user groups for high-value rd-groups
- Users with highest value represent greatest risk

ANALYSIS EXAMPLE

The Scenario

ANALYSIS EXAMPLE

Scenario:

- Multinational company based in the US is developing software for recording real-estate ownership over the Internet

Priorities:

- Preserve integrity and accountability

ANALYSIS EXAMPLE

Environment:

- Developers create and edit software on home systems across the world
- Software is downloaded and uploaded over VPN
- Code resides on servers located in Iowa
- Server backed up daily by corporate office

ANALYSIS EXAMPLE

Resources:

- Developer Workstations (DWS)
- VPN Connection (VPN)
- Server (SVR)
- Backup Files (BAK)

Goal:

- Identify insiders that might insert trap doors
- Identify insiders that could debilitate company
  - Destroy the code and its backups

ANALYSIS EXAMPLE

Worried About:

- Ability to alter code on DWS (directly or indirectly)
- Ability to alter or destroy code on SVR
- Ability to alter or destroy code on BAK
- Ability to alter code in transmission (mitm VPN)

RD-Groups:

- { (DWS: login, tamper) }
- { (SVR: write, destroy) }
- { (BAK: write, destroy) }
- { (VPN: configure) }

ANALYSIS EXAMPLE

Identify User Groups

USER GROUPS: DETAILED

User Group: { (DWS: login, tamper) }

- Developers
- Anyone with physical access to the workstation
  - Developers family
  - Housekeepers
  - Etc.
- Computer repair technicians
- Anyone with remote access to workstation
  - Rogue websites
  - Etc.

USER GROUPS: SIMPLIFIED

Actors:

- Vernon, a developer
- Wilma, Vernon's nosey wife
- Xander, a system administrator
- Yasmin, president at corporate office
- Zane, janitor at corporate office

PROTECTION DOMAINS

	DWS		VPN	SVR		BAK	
	log	tamp	config	write	dest	write	dest
Vernon *(developer)*	●	●		●		●	
Wilma *(wife)*	●	●		●		●	
Xander *(sysadmin)*			●	●	●	●	●
Yasmin *(president)*						●	●
Zane *(janitor)*					●		●

PROTECTION DOMAINS

	DWS		VPN	SVR		BAK	
	log	tamp	config	write	dest	write	dest
Vernon *(developer)*	●	●		●		●	
Wilma *(wife)*	●	●		●		●	
Xander *(sysadmin)*			●	●	●	●	●
Yasmin *(president)*						●	●
Zane *(janitor)*					●		●

ANALYSIS EXAMPLE

Assign and Evaluate Metrics

VALUE RESOURCES

Assign metrics to rd-groups:

40 ← { (SVR: write, destroy), (BAK: write, destroy) }

24 ← { (SVR, destroy), (BAK, destroy) }

16 ← { (SVR, write), (BAK, write) }

8 ← { (SVR, write) }

2 ← { (DWS, tamper) }

VALUE RESOURCES

	DWS		VPN	SVR		BAK	
	log	tamp	config	write	dest	write	dest
Vernon: 18 *(developer)*	**0**	**2**		**8**		**8**	
Wilma: 18 *(wife)*	**0**	**2**		**8**		**8**	
Xander: 44 *(sysadmin)*			**4**	**8**	**12**	**8**	**12**
Yasmin: 20 *(president)*						**8**	**12**
Zane: 24 *(janitor)*					**12**		**12**

PROTECTION DOMAINS

	DWS		VPN	SVR		BAK	
	log	tamp	config	write	dest	write	dest
Vernon *(developer)*	●	●		●		●	
Wilma *(wife)*	●	●		●		●	
Xander *(sysadmin)*			●	●	●	●	●
Yasmin *(president)*						●	●
Zane *(janitor)*					●		●

VALUE ACCESS ATTRIBUTES

Assign metric to attribute groups:

4 ← upper management access

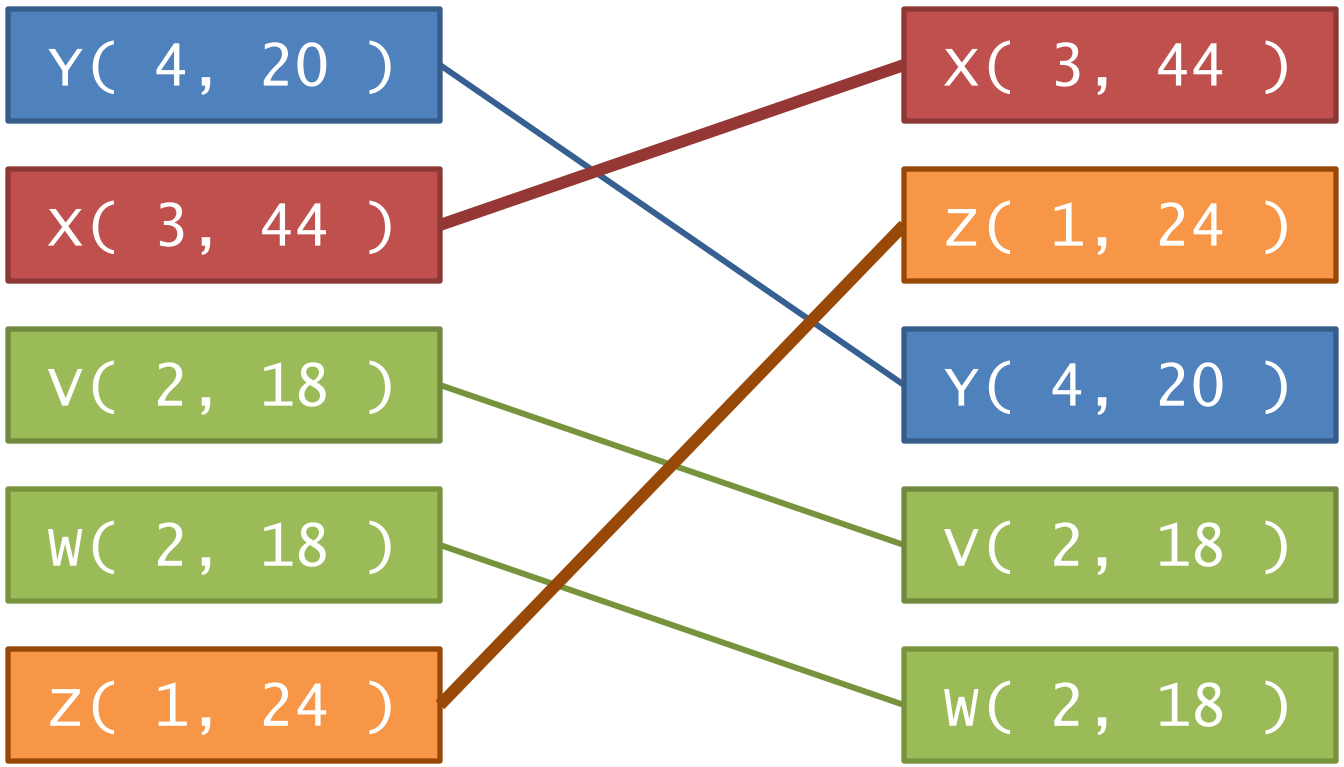
3 ← system administrator access

2 ← developer access

1 ← other staff access

EVALUATE METRICS

Name(user metric, resource metric)



ANALYSIS EXAMPLE

Reality Check

REALITY CHECK

- **Simplified Scenario**
  - Simplified resources
  - Simplified user groups
  - Simplified metrics
- **The Reality**
  - Difficult to anticipate avenues of attack
  - Cost functions difficult to create
  - Analysis possible for high-value resources and high-risk insiders?

CLAIMS

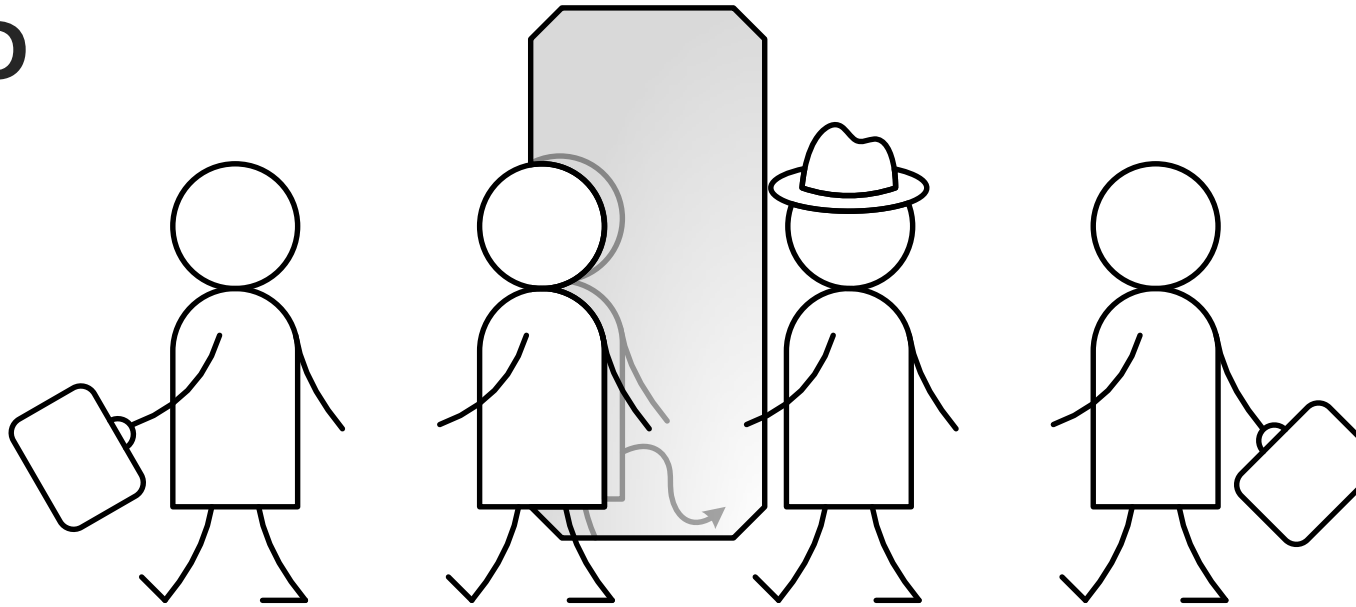
A Review

CLAIMS

- **The complexity of security policy is key to understanding the insider problem.**
- **Binary or perimeter-based definitions of an insider impede threat analysis.**
- **The ABGAC model identifies “insiderness” with respect to a resource and allows for insider threat analysis.**

QUESTIONS?

WE HAVE MET THE ENEMY AND HE IS US



UC DAVIS
MATT BISHOP
SOPHIE ENGLE
SEAN PEISERT
SEAN WHALEN

CA LABS
CARRIE GATES

LAKE TAHOE, CA
NSPW 09.23.2008

SUPPLEMENTAL

Definitions

INDEX

INSIDER

Anyone with more privileges in a lower level of policy than at a higher level of policy.

INSIDER PROBLEM

Insiders have more permissions than necessary to perform their jobs. Insiders must be trusted not to misuse these permissions for other purposes.

INSIDERNESS

A “measure” of an insider’s potential for misuse.

UNIFYING POLICY HIERARCHY

A hierarchical model of security policy at different levels of abstraction, introduced by Adam Carlson in his Master's Thesis.

ORACLE POLICY

Ideal policy, even if not explicitly defined.

OP(subject, object, action, environment/intent **) =**
{ *authorized*, *unauthorized* **}**

FEASIBLE POLICY

Attempts to approximate the Oracle Policy while taking into account the limitations of policy technology. Only able to understand system-definable subjects, objects, and actions, and returns unknown for anything outside its domain.

FP(subject, object, action **) =**
{ *authorized*, *unauthorized, unknown* **}**

CONFIGURED POLICY

Policy as configured on the system.

CP(subject, object, action **) =**
{ *authorized*, *unauthorized, unknown* **}**

REAL-TIME POLICY

Reflects what is possible on the system.

RP(subject, object, action **) =**
{ *possible*, *impossible* **}**

LEGITIMATE ACCESS MISUSE

Violating Oracle Policy using access granted in Feasible Policy or Configured Policy.

ILLEGITIMATE ACCESS MISUSE

Violating Configured Policy using access granted in the Real-Time Policy.