
University of Pittsburgh
Giving a Research Presentation
Fall 2019
A tale of two talks…
School of Computing and Information
Department of Computer Science

Announcement

Two weeks: Student technical presentations

Logistics:

- 6 groups
- Presentations split up over both days
- Each group will receive reviews from:
  - All classmates ("anonymous")
  - Me (not anonymous)

All talks must be emailed to me by start of class on Tuesday September 24th!

Giving a talk isn't easy the first time around…

Talk 1: Less than stellar talk

Discussion: Knee-jerk reactions

- What made that talk bad?
- What could be improved?
- Structural elements of a good talk

Talk 2: A (hopefully) improved talk

Discussion: Presentation elements

- Style and delivery
- Slide layout and effects

Brace yourselves for mediocrity…

Adam J. Lee and Ting Yu, "Towards Quantitative Analysis of Proofs of Authorization: Applications, Framework, and Techniques," in Proceedings of the 23rd IEEE Computer Security Foundations Symposium (CSF 2010), July 2010.

Towards Quantitative Analysis of Proofs of Authorization: Applications, Framework, and Techniques

Adam J. Lee (University of Pittsburgh)
Ting Yu (North Carolina State University)

Proofs of Authorization

- Trust management systems are used for access control in open systems
- Logical proofs are constructed at runtime to determine whether a given principal is allowed to access some specific resource
- Rather than simply interpreting a proof as a binary decision, we aim to analyze these proofs in a more quantitative manner

Framework

Conceptually, a trust management system contains

- A set *P* of principals
- A set *S* of resources
- A set *C* of credentials that make policy statements
  - Abstraction: s <- q, signed by p
    - P says that anyone that satisfies q can access s
    - P must control s
- An inference scheme $F : P \times S \times 2^C \rightarrow \{true, false\}$

Views

- We assume principals have some *view* of the system.

$$res(s \leftarrow q) \mapsto s \quad (1)$$

$$ac(s) \mapsto \{c \in \mathcal{C} \mid res(c) = s\} \quad (2)$$

Definition 1 (View): The *view* that some principal $p \in \mathcal{P}$ has of the protection state of a trust management system is defined as a three tuple $v_p = \langle S \subseteq \mathcal{S}, C \subseteq \mathcal{C}, \mathcal{A} \rangle$, where for each $s \in S$, $ac(s) \subseteq C$, and $\mathcal{A}$ is the abstraction of any auxiliary information that p has about the system.

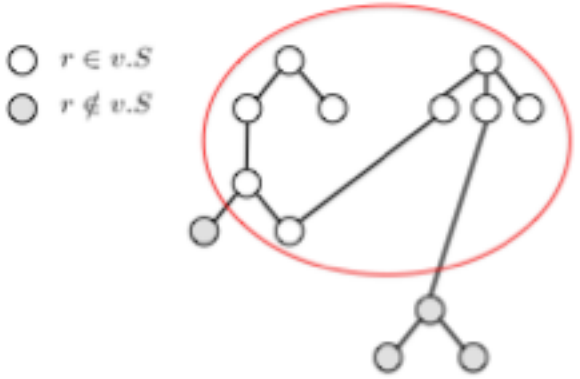


Figure 1. Graphical representation of a system view.

- This allows us to define proof scoring functions, score: *P* x *S* x *V* -> *T*

Properties of Scoring Functions

Required Properties

1. Deterministic
2. Simple ordering
   - $F(A,s,C)=T \wedge F(B,x,C)=F \rightarrow score(A,s,v) > score(B,s,v)$
3. Authorization relevant

Optional Properties

4. Interpretable
5. Bounded
6. Monotonic

Overview of RT_0

Basics

- Public keys identify users
- Roles group users

Four types of rules

- Simple member: A.R <- B
- Simple containment: A.R <- B.R'
- Linking containment: A.R <- A.R1.R2
- Intersection containment: A.R <- $B_1.R_1 \cap \ldots \cap B_n.R_n$

Policies built up using combinations of these rules

Scoring Functions: Take 1

- Assumptions
  - Simplified model
  - User designing function only knows about A.R
    - Knows all rules defining A.R
    - Understands semantics of every role “used” in these rules
  - Each credential associated with a vector w_i
    - All entries > 0
    - $||w_i||_1 = 1$

Scoring Functions: Take 1

Algorithm 1 A simple recursive scoring scheme.

Function score($p \in \mathcal{P}, A.R \in \mathcal{R}, v \subseteq \mathcal{V}$) : $\mathbb{R}$
// Filter credentials and initialize storage vector
$C = \{c_i \mid c_i \in v.C \wedge \mathsf{head}(c) = A.R\}$
Discard all $c_i \in C$ of the form $A.R \leftarrow P', P' \neq P$
$\overline{s} = [1, 0, \ldots, 0]$ // vector in $\mathbb{R}^{|C|+1}$

for all $c_i \in C$ **do**
    $\overline{w_i} = v.\mathcal{A}.\mathsf{weight}(c_i)$ // weight vector for c_i
    if $c_i = A.R \leftarrow P$ **then**
        $\overline{t} = [1, 1]$
    else if $\mathsf{body}(c_i) = B_1.R_1 \cap \cdots \cap B_k.R_k$ **then**
        $\overline{t} = [1, B_1.\mathsf{score}(p, B_1.R_1), \ldots, B_k.\mathsf{score}(p, B_k.R_k)]$
    else if $\mathsf{body}(c_i) = A.R_1.R_2$ **then**
        Find $B \subseteq A.R_1$ such that $\forall B_j \in B : P \in B_j.R_2$
        $\overline{t} = [1, max_{B_j \in B}(B_j.\mathsf{score}(p, B.R_2))]$
    if $\overline{t}$ contains any 0 entries **then**
        $\overline{s}[i] = 0$
    else
        $\overline{s}[i] = \overline{t} \cdot \overline{w_i}$

// Get master weight vector and combine all weights
$\overline{w} = v.\mathcal{A}.\mathsf{weight}(A.R)$
return $\overline{s} \cdot \overline{w}$

Scoring Functions: Take 1

Theorem 1: The function score : $\mathcal{P} \times \mathcal{R} \times \mathcal{V} \rightarrow \mathbb{R}$ defined in Algorithm 1 satisfies the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties.

Proof sketch:

- Deterministic: Obvious
- Simple ordering: Members scored with a positive value, non-members not scored (Line 16)
- Authorization relevant: Only credentials defining A.R used when computing a score (Line 3)
- Bounded: $||w_i||_1 = 1$ for all credentials c_i, so bounded above by 1. All entries in each $w_i > 0$, so bounded below by 0.
- Monotonic: No negative entries in any w_i, so score can never decrease by getting more information

Scoring Functions: Take 2

- Assumptions
  - More general system model
  - User knows nothing about policies
    - Structural information is discovered at runtime
    - Like RT, SecPAL, Gray, etc.
- Basic idea: Compute score based on number of ways that a policy can be satisfied

Scoring Functions: Take 2

$$\text{score}(p, A.R, v) = \sum_{(C_i, w_i) \in \text{osets}_\omega(v.C, A.R)} w_i \cdot \frac{1}{2}^i$$

Weighting functions ω : 2^C x 2^{2C} -> [0,1] weight the contribution of each proof

$$\omega_{len}(C_s, \_) = \gamma^{\max_{p \in \text{paths}(C_s)}(\text{length}(p))}$$

$$\omega_{ind}(C_s, C) = 1 - \frac{\max_{C_i \in C \setminus \{C_s\}}(|C_s \cap C_i|)}{|C_s|}$$

$$\omega_{li}(C_s, C) = \alpha \cdot \omega_{len}(C_s, \_) + \beta \cdot \omega_{ind}(C_s, C)$$

Scoring Functions: Take 2

Theorem 2: The class of scoring functions score : $\mathcal{P} \times \mathcal{R} \times \mathcal{V} \rightarrow \mathbb{R}$ represented by Equation 6 satisfies the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties, provided that the scaling function $\omega : 2^{\mathcal{C}} \times 2^{2^{\mathcal{C}}} \rightarrow [0, 1]$ used to parameterize the osets function is deterministic.

- Proof sketch
  - Deterministic: ω is deterministic, so score is too
  - Simple ordering: Same as function #1
  - Authorization relevant: trivial by def'n of proofs of authorization
  - Bounded: Based on geometric series in score converging to 1 when summed infinitely

Scoring Functions: Take 2

- Proof Sketch (cont)

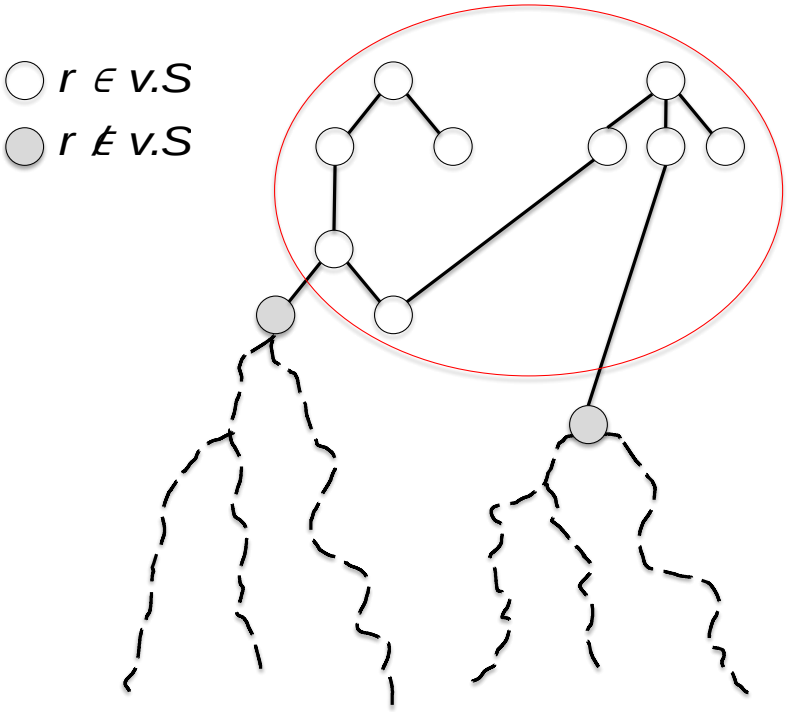
Monotonic. To prove the monotonicity of Equation 6, we proceed by induction. We first assume that principal p has previously discovered the (ordered) collection of proofs and weights $(C_1, w_1), \ldots, (C_n, w_n)$ for the role $A.R$. The base case that we must consider is that a new pair (C_s, w_s) is discovered such that no weight w_i is less than w_s. In this case, this new pair will introduce a new term to the end of the summation calculated by Equation 6, thereby increasing principal p's score for the role $A.R$.

Assume that (C_s, w_s) can be inserted before up to n terms in the sequence of (c_i, w_i) pairs while still preserving the monotonicity requirement. Now, assume that p has previously found proofs of authorization with the sequence of weights $S = (C_1, w_1), \ldots, (C_i, w_i), \ldots, (C_{i+n}, w_{i+n})$ and has now discovered a (C_s, w_s) pair such that $w_s > w_i$, thereby needing to be inserted before $n + 1$ terms in the sequence S. We first note that replacing (C_i, w_i) with (C_s, w) will generate a sequence S' that—when used in conjunction with Equation 6—will produce a score greater than that produced using S, since $w_s > w_i$ and all other terms are the same. By the inductive hypothesis, (C_i, w_i) can then be re-inserted before the n final terms of S' while still preserving monotonicity.

Composing Scoring Functions

Motivation

- Perfect information known within a security domain
- Less information known outside of security domain



Definitions

Definition 7 (Horizon): The *horizon* of a view v is defined as the set of resources $\mathsf{horizon}(v) = \{r \mid \exists c \in ac(v.S) : r \in \mathsf{body}(c) \wedge r \notin v.S\}$. That is, $\mathsf{horizon}(v)$ contains all resources mentioned in the body of policies protecting resources in $v.S$ that are not themselves in $v.S$.

Definition 8 (Sequential Composition): Assume that we have a view v, a principal p, a resource r, and two authorization scoring functions score_1 and score_2. We say that score_1 is *sequentially composed* with score_2 if there exists a resource $r' \in \mathsf{horizon}(v)$, a principal p', and a view v' such that $\mathsf{score}_2(p', r', v')$ is calculated when calculating $\mathsf{score}_1(p, r, v)$.

Definition 9 (Order-Preserving Homomorphism): Let $\mathsf{score}_1 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_1$ and $\mathsf{score}_2 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_2$ be two authorization scoring functions. Let $t_1 \in \mathcal{T}_1$ (resp. $t_2 \in \mathcal{T}_2$) be a threshold such that if $\mathsf{score}_1(p, s, v) \leq t_1$ (resp. $\mathsf{score}_2(p, s, v) \leq t_2$) then p cannot access resource s. Similarly, if $\mathsf{score}_1(p, s, v) > t_1$ (resp. $\mathsf{score}_2(p, s, v) > t_2$) then p can access resource s. A function $f : \mathcal{T}_2 \rightarrow \mathcal{T}_1$ is an *order-preserving homomorphism* from $\mathcal{T}_2$ to $\mathcal{T}_1$ if and only if (i) $t \leq t_2 \rightarrow f(t) \leq t_1$, (ii) $f(t_2) = t_1$, and (iii) $t > t_2 \rightarrow f(t) > t_1$.

Composition Theorem

Theorem 3: Let $\mathsf{score}_1 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_1$ and $\mathsf{score}_2 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_2$ be two authorization scoring functions that satisfy the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties. If there exists an order-preserving homomorphism f between $\mathcal{T}_2$ and $\mathcal{T}_1$, then the sequential composition of score_1 with score_2 is also *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic*.

Neat Corollaries

Corollary 1: The result of sequentially composing the authorization scoring functions defined by Algorithm 1 and Equations 6–9 using the order-preserving homomorphism $f(x) \mapsto x$ is an authorization scoring function that is also *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic*.

Corollary 2: Let $\mathsf{score} : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_1$ be a authorization scoring function that satisfies the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties and let v be a view. The result of sequentially composing score with an arbitrary any number of other *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* authorization scoring functions along $\mathsf{horizon}(v)$ is an authorization scoring function that is also *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic*.

Arbitrary composition along horizon

Corollary 3: Let $\mathsf{score}_1 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_1, \ldots, \mathsf{score}_n : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_n$ be proof scoring functions that satisfy the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties, and let $f_{n-1} : \mathcal{T}_n \rightarrow \mathcal{T}_{n-1}, \ldots, f_1 : \mathcal{T}_2 \rightarrow \mathcal{T}_1$ be order-preserving homomorphisms mapping between the ranges of these functions. The result of sequentially composing $\mathsf{score}_1, \ldots, \mathsf{score}_n$ using $f_1, \ldots, f_{n-1}$ is also authorization scoring function that is also *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic*.

Arbitrary depth of composition

Scoring Functions: Take 3

- Preliminaries

Definition 10 (Canonical Proof of Authorization): A *canonical proof of authorization* for a principal p and a role $A.R$ is a minimal set of credentials C from the universe of all possible credentials such that $F(p, A.R, C) = TRUE$. We denote by $\mathsf{csets}(p, A.R)$ the set of all canonical proofs for the principal p and the role $A.R$.

$$\mathsf{psets}(p, A.R, v) = \{(C_p, C_c) \mid C_c \in \mathsf{csets}(p, A.R) \\ \wedge C_p = v.C \cap C_c \wedge C_p \neq C_c\}$$

$$\mathsf{leaves}(C) = \{c \in C \mid c \text{ of the form } A.R \leftarrow p\}$$

$$\psi(C_p, C_c) = \frac{|\mathsf{leaves}(C_p \cap C_c)|}{|\mathsf{leaves}(C_c)|}$$

Scoring Functions: Take 3

Goal: Score role membership, as well as non-membership

- Membership: Obvious reasons
- Non-membership: Approximate pricing

$$\phi(x) = \begin{cases} 1 & \text{if } x \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

$$\mathsf{score}(p, A.R, v) = \phi(|\mathsf{sets}(v.C)|) \tag{21}$$
$$+\alpha \sum_{(w_i, C_i) \in \mathsf{osets}_\omega(v.C, A.R)} w_i \cdot \frac{1}{2}^i$$
$$+\beta \sum_{(w, C_p, C_c)_i \in \mathsf{opsets}(p, A.R, v)} w \cdot \frac{1}{2}^i$$

Scoring Functions: Take 3

Theorem 4: The class of non-member scoring functions score : $\mathcal{P} \times \mathcal{R} \times \mathcal{V} \rightarrow \mathbb{R}$ represented by Equations 17–21 satisfies the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties, provided that the scaling function $\omega : 2^{\mathcal{C}} \times 2^{2^{\mathcal{C}}} \rightarrow [0, 1]$ used to parameterize the osets function is deterministic.

Proof is similar to previous case

Interesting observation: Meets properties needed by composition theorem

Conclusions

- Proofs have a lot more information than the binary yes/no decision that we use them for
- We developed a formal framework for scoring these proofs of authorization
- Cases explored
  - Perfect information a priori
  - No information a priori
  - Arbitrary combinations
  - Incomplete proofs

Discuss: What was wrong with that talk?

Issues with content:

- Why should we care about the problem?
- How will the results be useful in practice?
- Had no idea where talk was going!
- Missing context to understand problem setup

Issues with delivery:

- Lack of eye contact
- Lecturing to the board/laptop, not the audience
- Blurry fonts
- Too much text
- …

Structure your talk based on your audience and the time that you have

Your audience: Generally smart individuals

- Computer Scientists? Yes
- In your area? Maybe
- Knowledgeable about your problem? Probably not

Time is usually limited

- Conference talk: 20 minutes or so
- Job talk: < 1 hour

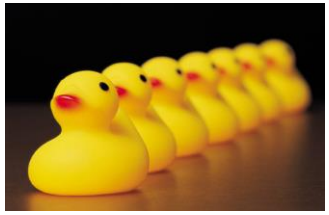
This is not a lot of time…

Bottom line: Your talk should be an advertisement for your paper(s)

That's not a lot of time, how should I structure my talk to relate to these people?

This is a hard problem…

… with interesting applications…



… that builds on prior work…

… in a verifiable way

Two sub-parts:

- You do something that has not been done
- You use neat technological advancements to do this

Hint: Try to give audience one good take-home point

It's not just *what* you say, but *how* you say it

Body language says a lot

- Make eye contact with your audience
  - *Corollary*: Face your audience
- Some movement is good
- Don't speak too fast (or too slow!)

Make useful slides

- Provide a topic outline to structure your talk
- One primary idea per slide
- Use slide titles to convey take-away message
- Do not read your slides!
- A picture is worth a thousand words…

Let's try to put some of this into practice…

Towards Quantitative Analysis of Proofs of Authorization: Applications, Framework, and Techniques

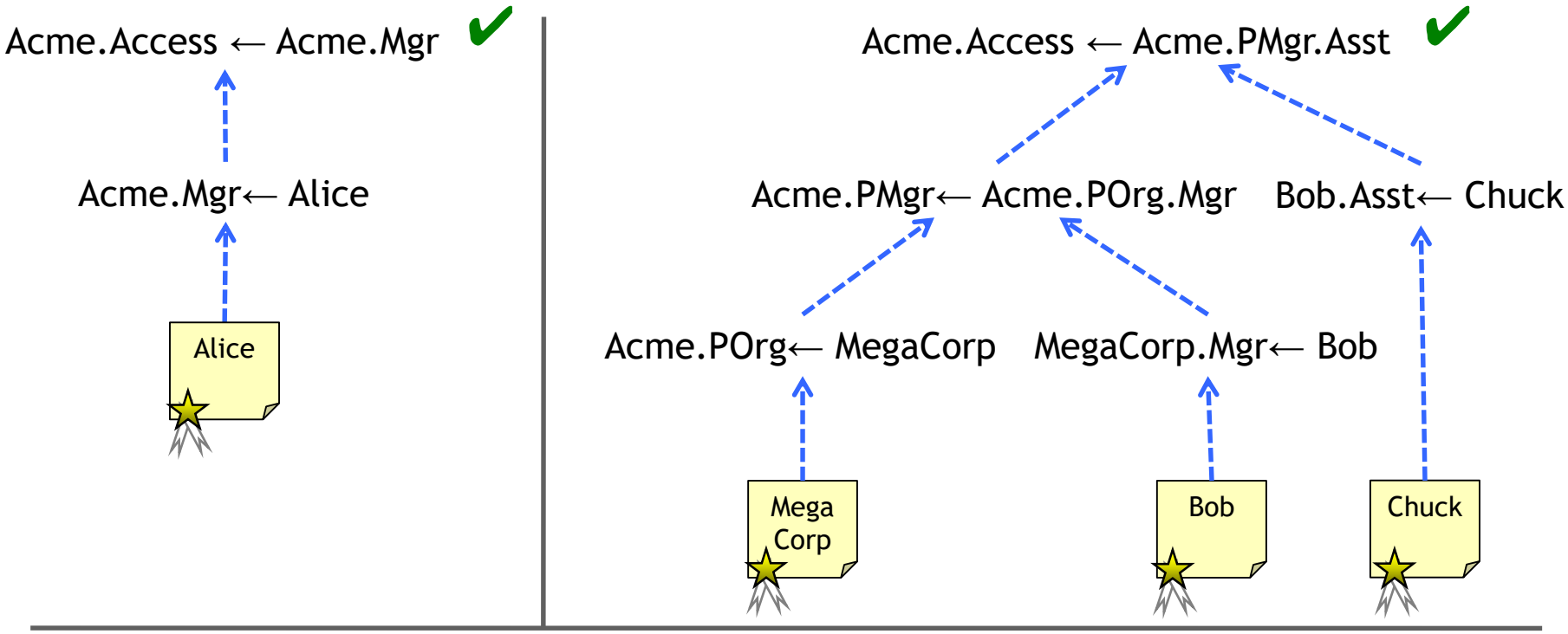
Adam J. Lee
adamlee@cs.pitt.edu
Department of Computer Science
University of Pittsburgh

Ting Yu
yu@csc.ncsu.edu
Department of Computer Science
North Carolina State University

University of Pittsburgh

Like most access control systems, distributed proof construction systems are typically used to support binary decisions

Example: Access to a company database



Note that…

- Both proofs are valid
- The first proof is far simpler than the second
- Why focus only on the *destination* (validity)? What about the *journey* (context)?

Proofs of authorization reveal a great deal of information about the conditions under which some access was granted

Authorization robustness

- How many proofs can some user generate?
- Are these proofs concise, or do they use odd delegations?
- How dependent on system state are these proofs?
- Applications: Anomaly detection, policy audit

vs.

User-to-user comparison

- Policies are requirements
- How well do various individuals satisfy them?
- Applications: Top-*k* analysis, group formation

Examination of incomplete proofs

- Policies aren't always perfect…
- How close is an unauthorized user to accessing a resource?
- Applications: Risk assessment, policy revision

What are we not doing?

Point-based access control & trust management

- E.g., Yao et al. 2006
- Privacy-preserving compliance checking of point-based policies

Reputation-based trust management

- E.g., Kamvar et al. 2003, Xiong and Liu 2003, Josang et al. 2007
- Aggregation-based trust, different than credential-based proofs

Risk-based access control

- E.g., MITRE 2003, Aziz et al. 2006, Cheng et al. 2007
- More on this later…

Reasoning under uncertain information

- E.g., Dempster 1976, Shafer 1976, Cox 2004
- Focus is on uncertain information and/or inference rules

Talk Outline

- Model for quantitative proof analysis
- Proof scoring functions
  - Desiderata
  - An example scoring construction
  - Functional composition
- Scoring incomplete proofs of authorization
- (Lots of) future directions

RT_0 is the simplest language in the *RT* family

Principals are represented by public keys

Policies are constructed using four basic types of assertion

1. Simple membership: Alice.Friend ← Bob
   - Bob is a member of Alice's "Friend" role

2. Simple containment: Acme.Contractor ← WidgetTech.Employee
   - WidgetTech employees are "Contractors" at Acme

3. Intersection containment: Tech.Disct ← StateU.Student ∩ IEEE.member
   - Students at Univ who are IEEE members are eligible for a discount

4. Linking containment: Acme.PMgr← Acme.POrg.Mgr
   - Members of the "Mgr" role defined by any member of "Acme.POrg" are members of Acme's "PMgr" role

Modeling Authorization Scoring Functions

An RT_0 trust management system consists of:

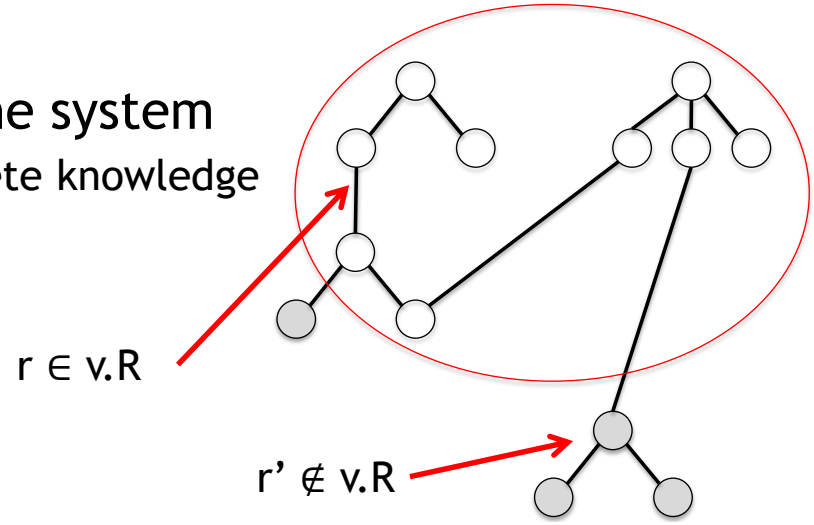
- A set $\mathcal{P}$ of principals
- A set $\mathcal{R}$ of roles/resources
- A set C of credentials

Very large…

- An inference scheme F : $\mathcal{P} \times \mathcal{R} \times 2^C \rightarrow$ {True, False}

Each principal has their own view of the system

- A set R $\subseteq \mathcal{R}$ for which they have complete knowledge
- A set C $\subseteq C$ of credentials
  - ac(r) ≡ { c ∈ C | head(c) = r }
  - ∀ r ∈ R : ac(r) ⊆ C
- A store of auxiliary information $\mathcal{A}$
  - Ignored in this talk, see paper for details



Proofs are scored relative to some principal's view

- score : $\mathcal{P} \times \mathcal{R} \times \mathcal{V} \rightarrow \mathcal{T}$

What properties should a proof scoring function have?

Necessary properties ensure that proof scores "make sense"

- Deterministic
- Simple ordering:
  - $\forall v \in \mathcal{V} : F(p_1, r, v.C) \wedge \neg F(p_2, r, v.C) \rightarrow score(p_1, r, v) \geq score(p_2, r, v)$
- Authorization relevant:
  - if F(p, r, C) = True, then C is a proof for p to access r
  - if F(p, r, C') = False for all $C' \subset C$, C is a minimal proof
  - Only credentials belonging to some minimal proof influence score

Desirable properties are beneficial, but not strictly necessary

- Bounded: $\exists\ b_1, b_2 : \forall\ p, r, v : b_1 \leq score(p, r, v) \leq b_2$
- Monotonic: $v \subseteq v' \rightarrow score(p, r, v) \leq score(p, r, v')$

What might some interesting classes of authorization scoring functions look like?

Scoring proofs generated with incomplete views

Assumption: Principals start with empty views and discover minimal proofs of authorization at runtime

- Credential chain discovery in *RT*
- Distributed proof construction in, e.g., Grey or Cassandra
- Etc.

Let sets(C, r) represent the minimal proofs for r contained in C

One simple scoring construction is the following:

$$\text{score}(p, r, v) = \sum_{i=1}^{|\text{sets}(v.C, r)|} \frac{1}{2}^{i}$$

This function:

- Defines robustness as the number of proofs that a principal can generate
- Exponentially decays the contribution of proofs as they are discovered

This simple notion of robustness is not very exciting, but can easily be tuned

Consider a function $\omega : 2^{\mathcal{C}} \times 2^{2^{\mathcal{C}}} \rightarrow [0, 1]$ that weights a minimal proof (possibly) by comparing it with other minimal proofs

Examples:

- $\omega_{len}(C_s, \cdot) = \gamma^{\max_{p \in \text{paths}(C_s)}(\text{length}(p))}$
- $\omega_{card}(C_s, \cdot) = \gamma^{|C_s|}$
- $\omega_{ind}(C_s, C) = 1 - \dfrac{\max_{C_i \in C \setminus C_s}(|C_s \cap C_i|)}{|C_s|}$
- Linear combinations of the above

Prefer limited delegation

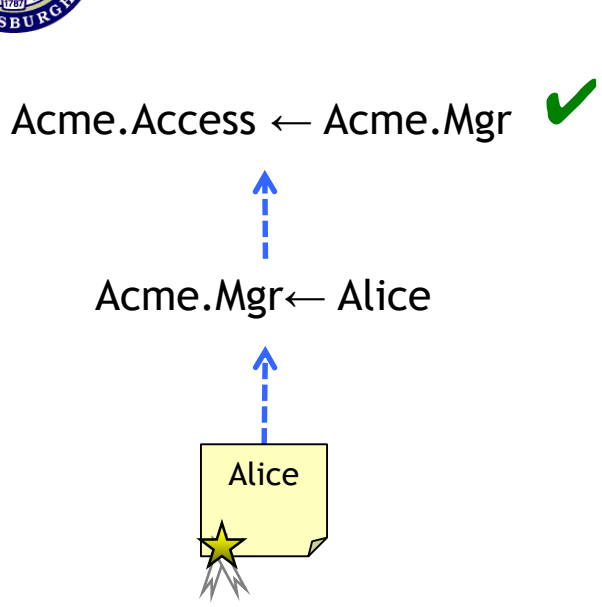
Prefer simple structure

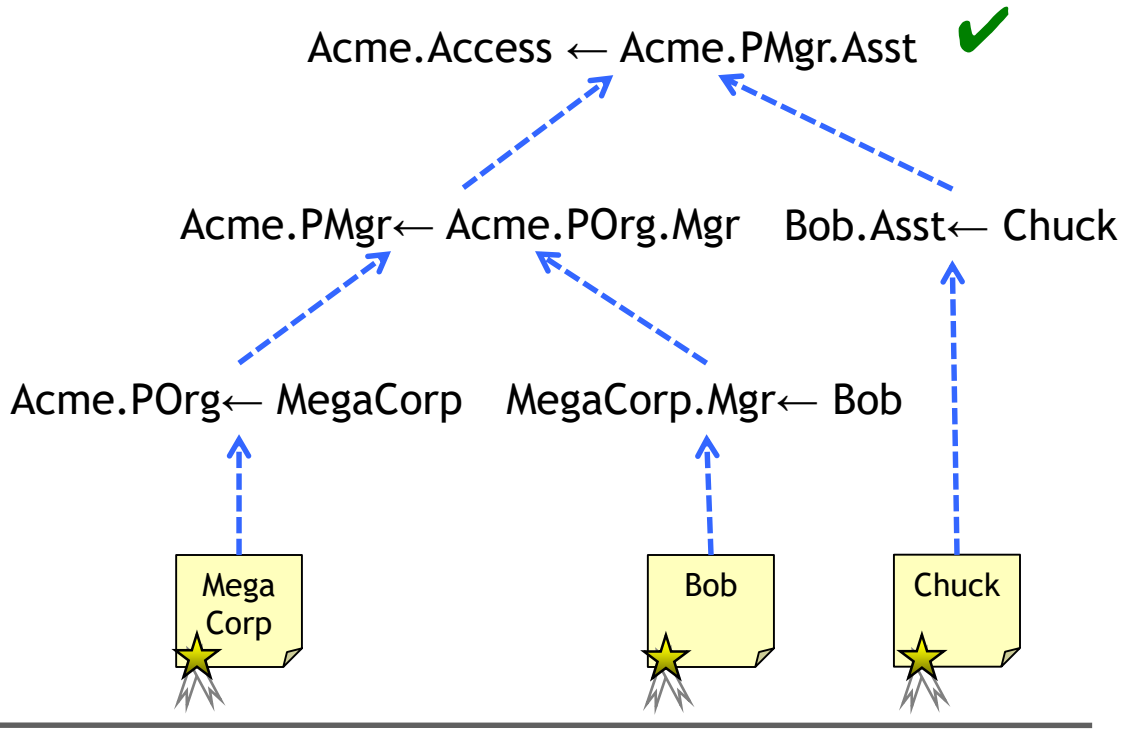
Prefer multiple, largely independent proofs

Our scoring construction can then be rewritten as:

$$\text{score}(p, r, v) = \sum_{(C_i, w_i) \in \text{osets}_\omega(v.C, r)} w_i \cdot \frac{1}{2}^i$$

Example, Redux





Using ω_{len}:

- score(Alice, Acme.Access, v_1) = 0.365
- score(Chuck, Acme.Access, v_2) = 0.328

Using ω_{card}:

- score(Alice, Acme.Access, v_1) = 0.365
- score(Chuck, Acme.Access, v_2) = 0.215

Note that ω_{ind} is irrelevant in this case…

This proof scoring function satisfies our desiderata

Theorem: Provided that the function ω used to parameterize osets is deterministic, the authorization scoring function

$$\text{score}(p, r, v) = \sum_{(C_i, w_i) \in \text{osets}_\omega(v.C, r)} w_i \cdot \frac{1}{2}^i$$

satisfies the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties.

The above scoring function

- is certainly not the only such authorization scoring function
- may not be the best scoring function for all situations
- may only be sensible to use on certain parts of a proof

However, it is an interesting building block…

In many situations, defining *the* proof scoring function to use could be a difficult task

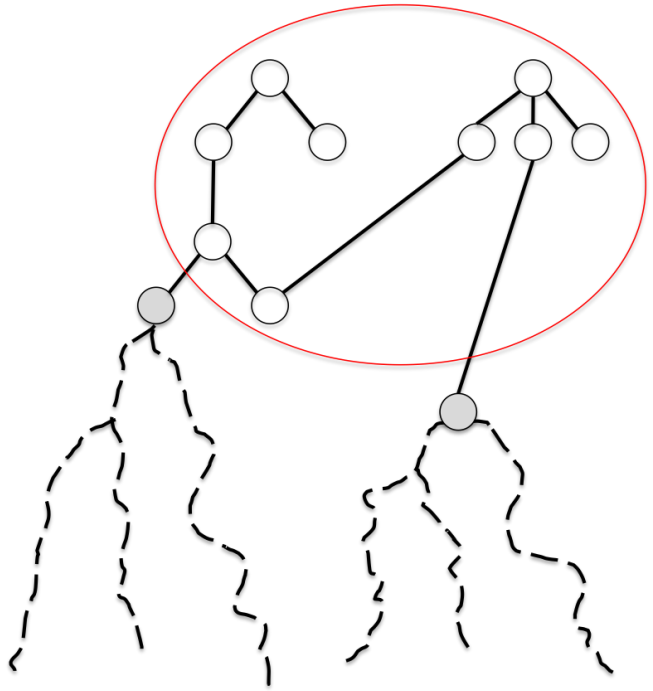
Example: Security administrators within an organization

Perfect information within domain

- Exact knowledge of resource/role semantics
- Very precise weighting and analysis
- Hand-tuned scoring is possible

On-demand information outside of domain

- Known semantics for horizon resources
- Full semantic knowledge of proof is unlikely
- Structure is discovered at runtime



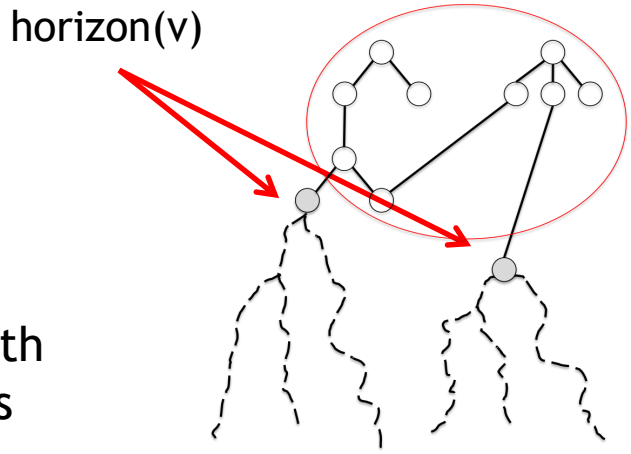
Under what circumstances can good "building block" functions be composed to construct proof scoring functions while still preserving the properties of each building block?

Fortunately, reasonable proof scoring functions maintain their properties under sequential composition

Definition: Assume that we have

- Principals p and p'
- Resources r and r'
- Views v and v'
- Functions score and score'

We say that score is sequentially composed with score' if r' ∈ horizon(v) and score'(p', r', v') is calculated when calculating score(p, r, v).



Theorem*: Let $\text{score}_1 : \mathcal{P} \times S \times \mathcal{V} \rightarrow \mathcal{T}$ and $\text{score}_2 : \mathcal{P} \times S \times \mathcal{V} \rightarrow \mathcal{T}$ be two authorization scoring functions that satisfy the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties. The sequential composition of these functions also satisfies the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties.

So far, we have focused on scoring *complete* proofs of authorization

If a policy is out of date or incomplete, users who should be able to do something might not be able to

Risk-based access control is one approach to limiting inflexibility

- Place a (typically monetary) cap on the amount of risk/damage permissible
- Tokenize this risk/damage and distributed it to users
- Compute "risk prices" for every resource in the system
- If users can pay the access price, they are permitted access

While this would be significantly more flexible than policy-based approaches, pricing access to individual resources is non-trivial

Alternate approach: Rather than pricing resources per user for *every* user, price deviations from expected policies

To price deviations from an expected policy, we first need to be able to quantify the degree of these deviations

A natural generalization of our framework provides one approach for doing exactly this

Step 1: Find the canonical proofs of authorization for the resource

- All minimal sets of credentials C such that F(p, r, C) = True
  - Note: These credentials may not all be materialized in the system
- Call the result csets(p, r)
- Note: The *RT* credential chain discovery process does this for us

Step 2: Find partial matches between v.C and csets(p, r)

- psets(p, r, v) = $\{(C_p, C_c) \mid C_c \in csets(p, r) \wedge C_p = v.C \cap C_c \wedge C_p \neq C_c\}$

Step 3: Evaluate the quality of each partial match

- leaves(C) = { c ∈ C | c of the form r ← p }
- $\psi(C_p, C_c) = |leaves(C_p \cap C_c)| \ / \ |leaves(C_c)|$
- opsets(p, r, v) = $\{ (w, C_p, C_c) \mid (C_p, C_c) \in psets(p, r, v) \wedge w = \psi(C_p, C_c) \}$

Step 4: Tying it all together

$$\phi(x) = \begin{cases} 1 & \text{if } x \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{score}(p, r, v) = \phi(|\text{sets}(v.C)|)$$

Ensures non-member scores always below 1

Score complete proofs…

$$+\alpha \sum_{(w_i, C_i) \in \text{osets}_\omega(v.C, r)} w_i \cdot \frac{1}{2}^i$$

…and partial proofs

$$+\beta \sum_{(w, C_p, C_c)_i \in \text{opsets}(p, r, v)} w \cdot \frac{1}{2}^i$$

Note: This function satisfies the *deterministic*, *simple ordering*, *authorization relevant*, *bounded*, and *monotonic* properties

Due to our composition theorem, this function can act as a template function that can be sequentially composed with other reasonable authorization scoring functions

This work is just a first step…

Question 1: These types of scoring functions *seem* sensible, but do they make sense in the context of real policies?

Medicine

Academic Departments

Defense

Question 2: RT_0 is a very simple language. What would scoring constructions for more feature-rich languages look like?

- Credentials with internal structure (e.g., RT_1)
- Flexible rule structure (e.g., SecPAL, Grey)
- Reasoning over aggregates like reputation (e.g., CTM, WBSNs)
- …

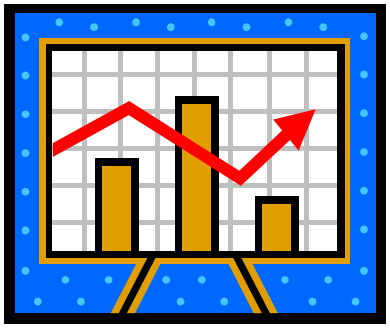
Efficiency and functional extensions…

Question 3: How can we efficiently construct cost-minimizing approximate proofs of authorization?

- Can we prune the state-space as we search?
- Applications to risk-based access control

Question 4: How can we efficiently execute top-*k* queries over (distributed) authorization datasets?

Group formation



Evaluating Policy Utilization

Conclusions

Interesting applications of reasoning about proofs of authorization

- User-to-user ranking of proofs
- User-to-ideal assessment of proof quality/robustness/etc.
- Understanding the changing needs of an organization
- Risk-aware authorization reasoning
- …

Our goals for this initial work

- Develop a formal model for proof scoring
- Identify necessary and desirable criteria for scoring functions
- Demonstrate that these criteria are attainable in practice
- Understand the situations in which scoring functions can be composed

There is still much to be done…

Thank you!

Towards Quantitative Analysis of Proofs of Authorization: Applications, Framework, and Techniques

Adam J. Lee
adamlee@cs.pitt.edu
Department of Computer Science
University of Pittsburgh

Ting Yu
yu@csc.ncsu.edu
Department of Computer Science
North Carolina State University

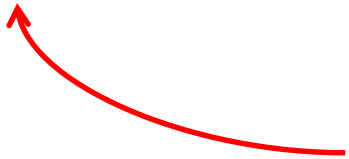
Questions?

Discuss: Why was this talk (hopefully) better than the first run through?

General tips and tricks…

Practice makes better

- *Alone*: Work on your "script," smooth out transitions
- *Research group*: Get used to other people being around
- *Broader population*: Assess comprehensibility to outsiders



e.g., other grad student friends, department seminars, etc…

Do you *really* want that laser pointer?

"Flash" is good, but too much flash is distracting

- *Good*: Animations to progressively build large diagrams or equations
- *Bad*: Animating every slide transition and every line of text…

Get out of your head and into your talk ☺