Presenting Your Research

Prof. Adriana Kovashka
University of Pittsburgh
October 13, 2021
Some Guidelines

1. Consider the audience & event
2. Structure based on audience and time
3. How you say it matters
4. Don’t bury the lead
5. Be concise
6. Give credit
7. Q&A: the unknown unknowns
8. Delivery tips
9. Clear slides
Consider Audience and Event

- Your group meeting?
- CVPR?
- Job talk to entire CS department?
- Interdisciplinary?
- K-12?
- Formal? Casual?

- Degree of detail
- Degree of jargon
- Depth vs. breadth
- Latest vs. arc of progress
Structure based on Audience and Time

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)
Structure based on Audience and Time

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
How You Say It Matters

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...
Don’t Bury the Lead

- Don’t leave contribution implicit
- Reiterate and rephrase message throughout
- Verbally: give salient markers; “Important… ”stress that”…
- “Punchlines” for results
Be Concise

“I didn't have time to write a short letter, so I wrote a long one instead.”
— Mark Twain

• Prep the “concept bullets”
• Breathe, and use fewer words
• Short text phrases (not sentences)
• Not every detail needs to surface

https://en.wikipedia.org/wiki/Mark_Twain
Give Credit

• Paint the big picture of literature for context
• (Clusters of) related work & key contrasts
• Give credit for borrowed slides, per slide

[Slide credit: Jane Smith]
Q&A: The Unknown Unknowns

- Guess likely questions & prepare
- Don’t skip to backup slides unless necessary
- Sometimes it’s better to defer a question
- Answer, then stop
- Share feedback with co-authors afterwards

https://duffylondon.com/product/tables/abyss-horizon/

Slide credit: Kristen Grauman
Delivery Tips

• Practice and get feedback; iterate
• The first slide - what will you say?
• Be loud enough
• Use pauses
• Flow: think through transition *in* and *out* of each slide
• Qualitative examples: say something about one or two
• Manage time: stopwatch, prevent derail
• If you’re skipping something, then skip it
• Think about where you want to stand / test the room
• Check the laptop, AV
• Nerves: “If you’re nervous, it means you care” ~Trevor Darrell

https://www.ted.com/playlists/497/practice_makes_perfect

Slide credit: Kristen Grauman
... reduce cognitive load:

• Animation – to focus attention
• Font size – 28+ for main text
• Simplest visual possible to make the point
• Consistency: font size, capitalization, alignment...
• Avoid jitter of text placement in consecutive slides
• One liners where possible
• Use color to link pieces of equations
• Delete “Hi my name is …” from notes of first slide!
• Avoid content-free “Thank you!” slide
More 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 😊
Example: A Mediocre Talk

• 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: $s \leftarrow q$, signed by $p$
    - $P$ says that anyone that satisfies $q$ can access $s$
    - $P$ must control $s$
- An inference scheme $F : P \times S \times 2^C \rightarrow \{\text{true, false}\}$
Views

• We assume principals have some view of the system.

\[
\text{res}(s \leftarrow q) \rightarrow s \\
\text{ac}(s) \rightarrow \{c \in C \mid \text{res}(c) = s\}
\]

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 \( v_p = \langle S \subseteq S, C \subseteq C, A \rangle \), where for each \( s \in S \), \( \text{ac}(s) \subseteq C \), and \( A \) is the abstraction of any auxiliary information that \( p \) has about the system.

• This allows us to define proof scoring functions, score: \( P \times S \times V \rightarrow T \)
Properties of Scoring Functions

Required Properties

1. Deterministic
2. Simple ordering
   • $F(A,s,C)=T \land F(B,x,C)=F \rightarrow \text{score}(A,s,v) > \text{score}(B,s,v)$
3. Authorization relevant

Optional Properties

4. Interpretable
5. Bounded
6. Monotonic
Overview of RTₐ

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 <- B₁.R₁ ∩ ... ∩ Bₙ.Rₙ

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.

1: Function score($p \in \mathcal{P}, A.R \in \mathcal{R}, v \subseteq \mathcal{V}$) : $\mathbb{R}$
2: // Filter credentials and initialize storage vector
3: $C = \{ c_i \mid c_i \in v, C \land \text{head}(c) = A.R \}$
4: Discard all $c_i \in C$ of the form $A.R \leftarrow P', P' \neq P$
5: $\bar{s} = [1, 0, \ldots, 0]$ // vector in $\mathbb{R}^{|C|+1}$
6: 
7: for all $c_i \in C$ do
8: $\bar{w}_i = v.A.\text{weight}(c_i)$ // weight vector for $c_i$
9: if $c_i.A.R \leftarrow P$ then
10: $\bar{t} = [1, 1]$
11: else if body$(c_i) = B_1.R_1 \cap \cdots \cap B_k.R_k$ then
12: $\bar{t} = [1, B_1.\text{score}(p, B_1.R_1), \ldots, B_k.\text{score}(p, B_k.R_k)]$
13: else if body$(c_i) = A.R_1.R_2$ then
14: Find $B \subseteq A.R_1$ such that $\forall B_j \in B : P \in B_j.R_2$
15: $\bar{t} = [1, \max_{B_j \in B}(B_j.\text{score}(p, B.R_2))]$
16: if $\bar{t}$ contains any 0 entries then
17: $\bar{s}[i] = 0$
18: else
19: $\bar{s}[i] = \bar{t} \cdot \bar{w}_i$
20: 
21: // Get master weight vector and combine all weights
22: $\bar{w} = v.A.\text{weight}(A.R)$
23: return $\bar{s} \cdot \bar{w}$
Scoring Functions: Take 1

Theorem 1: The function \( \text{score} : \mathcal{P} \times \mathcal{R} \times \mathcal{V} \rightarrow \mathbb{R} \) 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

Slide credit: Adam Lee
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 \( \omega : 2^C \times 2^{2^C} \rightarrow [0,1] \) weight the contribution of each proof

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

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

\[
\omega_{li}(C_s, C) = \alpha \cdot \omega_{\text{len}}(C_s, \_ ) + \beta \cdot \omega_{\text{ind}}(C_s, C)
\]
Scoring Functions: Take 2

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

- **Proof sketch**
  - Deterministic: \( \omega \) 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
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

Slide credit: Adam Lee
Definitions

Definition 7 (Horizon): The horizon of a view $v$ is defined as the set of resources $\text{horizon}(v) = \{ r \mid \exists c \in ac(v.S) : r \in \text{body}(c) \land r \notin v.S \}$. That is, $\text{horizon}(v)$ 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 $\text{score}_1$ and $\text{score}_2$. We say that $\text{score}_1$ is sequentially composed with $\text{score}_2$ if there exists a resource $r' \in \text{horizon}(v)$, a principal $p'$, and a view $v'$ such that $\text{score}_2(p', r', v')$ is calculated when calculating $\text{score}_1(p, r, v)$.

Definition 9 (Order-Preserving Homomorphism): Let $\text{score}_1 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_1$ and $\text{score}_2 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_2$ 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 $\text{score}_1(p, s, v) \leq t_1$ (resp. $\text{score}_2(p, s, v) \leq t_2$) then $p$ cannot access resource $s$. Similarly, if $\text{score}_1(p, s, v) > t_1$ (resp. $\text{score}_2(p, s, v) > t_2$) 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) $t > t_2 \rightarrow f(t) > t_1$. 

Slide credit: Adam Lee
Composition Theorem

Theorem 3: Let $\text{score}_1 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_1$ and $\text{score}_2 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_2$ 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 $\text{score}_1$ with $\text{score}_2$ 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 $f(x) \rightarrow x$ is an authorization scoring function that is also deterministic, simple ordering, authorization relevant, bounded, and monotonic.

Corollary 2: Let $\text{score} : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_1$ 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 $\text{score}$ with an arbitrary any number of other deterministic, simple ordering, authorization relevant, bounded, and monotonic authorization scoring functions along horizon($v$) is an authorization scoring function that is also deterministic, simple ordering, authorization relevant, bounded, and monotonic.

Corollary 3: Let $\text{score}_1 : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_1, \ldots, \text{score}_n : \mathcal{P} \times \mathcal{S} \times \mathcal{V} \rightarrow \mathcal{T}_n$ 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 $\text{score}_1, \ldots, \text{score}_n$ using $f_1, \ldots, f_{n-1}$ is also authorization scoring function that is also deterministic, simple ordering, authorization relevant, bounded, and monotonic.

Arbitrary composition along horizon

Arbitrary depth of composition

Slide credit: Adam Lee
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 $csets(p, A.R)$ the set of all canonical proofs for the principal $p$ and the role $A.R$.

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

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

$$\psi(C_p, C_c) = \frac{|leaves(C_p \cap C_c)|}{|leaves(C_c)|}$$
Scoring Functions: Take 3

Goal: Score role membership, as well as non-membership
- Membership: Obvious reasons
- Non-membership: Approximate pricing

\[
\phi(x) = \begin{cases} 
1 & \text{if } x \geq 1 \\
0 & \text{otherwise} 
\end{cases} \quad (20)
\]

\[
\text{score}(p, A.R, v) = \phi(|\text{sets}(v.C)|) + \alpha \sum_{(w_i,C_i) \in \text{osets}_\omega(v,C,A.R)} w_i \cdot \frac{1}{2^i} + \beta \sum_{(w,C_p,C_c) \in \text{opsets}(p,A.R,v)} w \cdot \frac{1}{2^i} \quad (21)
\]
Scoring Functions: Take 3

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

Slide credit: Adam Lee
Example: A Better Talk

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

**Example:** Access to a company database

- Acme.Access ← Acme.Mgr
- Acme.Mgr ← Alice

- Acme.Access ← Acme.PMgr.Asst
- Acme.PMgr ← Acme.POrg.Mgr
- Acme.POrg ← MegaCorp
- MegaCorp.Mgr ← Bob
- Bob.Asst ← Chuck

**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

**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

Slide credit: Adam Lee
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
- 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 $\leftarrow$ Bob
   - Bob is a member of Alice’s “Friend” role

2. **Simple containment:** Acme.Contractor $\leftarrow$ WidgetTech.Employee
   - WidgetTech employees are “Contractors” at Acme

3. **Intersection containment:** Tech.Disct $\leftarrow$ StateU.Student $\cap$ IEEE.member
   - Students at Univ who are IEEE members are eligible for a discount

4. **Linking containment:** Acme.PMgr $\leftarrow$ Acme.POrg.Mgr
   - Members of the “Mgr” role defined by any member of “Acme.POrg” are members of Acme’s “PMgr” role

Slide credit: Adam Lee
An $RT_0$ trust management system consists of:

- A set $P$ of principals
- A set $R$ of roles/resources
- A set $C$ of credentials
- An inference scheme $F : P \times R \times 2^C \rightarrow \{\text{True, False}\}$

Each principal has their own view of the system:

- A set $R \subseteq R$ for which they have complete knowledge
- A set $C \subseteq C$ of credentials
  - $ac(r) \equiv \{ c \in C \mid \text{head}(c) = r \}$
  - $\forall r \in R : ac(r) \subseteq C$
- A store of auxiliary information $A$
  - Ignored in this talk, see paper for details

Proofs are scored relative to some principal’s view:

- $score : P \times R \times V \rightarrow T$
What properties should a proof scoring function have?

Necessary properties ensure that proof scores “make sense”

- **Deterministic**
- **Simple ordering:**
  \[ \forall v \in V : F(p_1, r, v.C) \land \neg F(p_2, r, v.C) \rightarrow \text{score}(p_1, r, v) \geq \text{score}(p_2, r, v) \]
- **Authorization relevant:**
  - if \( F(p, r, C) = \text{True} \), then \( C \) is a proof for \( p \) to access \( r \)
  - if \( F(p, r, C') = \text{False} \) 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:** \( \exists \ b_1, b_2 : \forall p, r, v : b_1 \leq \text{score}(p, r, v) \leq b_2 \)
- **Monotonic:** \( v \subseteq v' \rightarrow \text{score}(p, r, v) \leq \text{score}(p, r, v') \)

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 \( \text{sets}(C, r) \) represent the minimal proofs for \( r \) contained in \( C \)

One simple scoring construction is the following:

\[
\text{score}(p, r, v) = \sum_{i=1}^{\mid\text{sets}(v, C, r)\mid} \frac{1}{2^i}
\]

This function:

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

Slide credit: Adam Lee
This simple notion of robustness is not very exciting, but can easily be tuned

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

Examples:

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

Our scoring construction can then be rewritten as:

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

Slide credit: Adam Lee
Example, Redux

Using $\omega_{\text{len}}$:
- $\text{score}(\text{Alice, Acme.Access, } v_1) = 0.365$
- $\text{score}(\text{Chuck, Acme.Access, } v_2) = 0.328$

Using $\omega_{\text{card}}$:
- $\text{score}(\text{Alice, Acme.Access, } v_1) = 0.365$
- $\text{score}(\text{Chuck, Acme.Access, } v_2) = 0.215$

Note that $\omega_{\text{ind}}$ is irrelevant in this case...

Slide credit: Adam Lee
This proof scoring function satisfies our desiderata

**Theorem:** Provided that the function $\omega$ used to parameterize osets is deterministic, the authorization scoring function

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

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?

Slide credit: Adam Lee
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 $\text{score}$ and $\text{score}'$

We say that $\text{score}$ is **sequentially composed** with $\text{score}'$ if $r' \in \text{horizon}(v)$ and $\text{score}'(p', r', v')$ is calculated when calculating $\text{score}(p, r, v)$.

**Theorem:** Let $\text{score}_1 : P \times S \times V \rightarrow T$ and $\text{score}_2 : P \times S \times V \rightarrow T$ 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.

Slide credit: Adam Lee
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 $F(p, r, C) = \text{True}$
  - 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 $csets(p, r)$
- $psets(p, r, v) = \{(C_p, C_c) \mid C_c \in csets(p, r) \land A \land C_p = v.C \cap C_c \land A \land C_p \neq C_c\}$

**Step 3:** Evaluate the quality of each partial match
- $leaves(C) = \{c \in C \mid c \text{ of the form } r \leftarrow p\}$
- $\psi(C_p, C_c) = \frac{|leaves(C_p \cap C_c)|}{|leaves(C_c)|}$
- $opsets(p, r, v) = \{(w, C_p, C_c) \mid (C_p, C_c) \in psets(p, r, v) \land w = \psi(C_p, C_c)\}$
Step 4: Tying it all together

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

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

Score complete proofs...

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

...and partial proofs

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

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?

**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

Slide credit: Adam Lee
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!


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?

Slide credit: Adam Lee
Why was this better?