Access Control for Online Social Networks using Relationship Type Patterns

Dissertation Defense

Yuan Cheng
Department of Computer Science
University of Texas at San Antonio
4/16/2014

Roadmap

2008	2009	2010	2011	2012	2013	2014
Entered the program	Joined the SNGuard project	Passed the qualifying exam		Passed the proposal		Expected to graduate
	Literature review	Identified the problems →	ACON[5], IEEE IC[6] ↘		UURAC Evaluation[1]	
			Delved into ReBAC	UURAC[4] ↗ → ↓		UURAC$_A$[2]
				URRAC[3]		
				3rd party applications →	PSOSM[7]	

Outline

- Introduction
- UURAC
- $UURAC_A$
- URRAC
- Conclusion

Background

- Security issues in OSNs can be organized into at least four categories
  - Privacy breaches (focus of this work)
  - Spam and phishing attacks
  - Sybil attacks
  - Malware attacks
- Privacy breaches
  - Easy to happen from OSN providers, other users, and 3rd party applications
  - OSN providers store user data
  - 3rd party applications provide extra functionalities
  - Major threats are from *peer users*
    - Not aware of who they share with and how much
    - Have difficulty in managing privacy controls

Why Privacy is Hard to Protect in OSNs

- Users tend to give out too much information
  - Unaware of privacy issues
  - Promote sharing vs. Protect privacy
- Users tend to be Reactive rather than Proactive
- Privacy policies
  - Changing over time
  - Confusing
  - Privacy thresholds vary by individuals

The Challenges of OSN Access Control

- Lack of a Central Administrator
  - Traditional access control mechanisms, such as RBAC, requires an administrator to manage access control
  - No such administrator exists in OSNs
- Dynamic Changing Environment
  - Frequent content updates and volatile nature of relationships
  - Identity and attribute-based access control are not effective for OSNs

Relationship-based Access Control

- Users in OSNs are connected by social relationships (user-to-user relationships)
- Owner of the resource can control its release based on such relationships between the access requester and the owner

Motivating Examples

- Related User's Control
  - There exist several different types of relationships in addition to *ownership*
  - e.g., Alice and Carol want to control the release of Bob's photo which contains Alice and Carol's image.
- Administrative Control
  - A change of relationship may result in a change of authorization
  - Treat *administrative activities* different from normal activities
    - Policy specifying, relationship invitation and relationship recommendation
  - e.g., Bob's mother Carol may not want Bob to become a friend with her colleagues, to access any violent content or to share personal information with others.
- Attribute-aware ReBAC
  - Exploit more complicated topological information
  - Utilize attributes of users and relationships
  - e.g., common friends, duration of friendship, minimum age, etc.

Problem Statement

- Traditional access control mechanisms are not suitable for OSNs
  - OSNs keep massive resources and change dynamically
- Existing relationship-based access control approaches are coarse-grained and limited
  - Commercial systems support either limited types or limited depth of U2U relationships
  - Academic works are also not flexible and expressive enough in relationship composition
- Policy administration and conflict resolution are missing
  - Multiple users can specify policies for the same resource
- Using relationships alone does not meet users' expectations

Thesis

- Users and resources are interconnected through U2U, U2R and R2R relationships, which form the basis of an OSN system, the *social graph*.
- By utilizing regular expression notation for policy specification, it is efficient and effective to regulate access in OSNs in terms of the *pattern of relationship path* on the social graph and the *hopcount limit* on the path.
- Integrating attribute-based policies further enables finer-grained controls that are not available in ReBAC alone.

Scope and Assumptions

- Assumptions
  - The threat model does *not* include OSN providers
  - Users' computers are *not* compromised by malicious intruders or malwares
  - *Do not* consider the case when a hacker gains unauthorized access to a site's code and logic
- Scope
  - Aim to improve the *access control* mechanism
    - ReBAC

Contributions

- Identified access control characteristics for OSNs based on relationships
  - Supporting essential characteristics that need to be addressed by OSN access control
- Further built two ReBAC models that utilize different kinds of relationships, using regular expression notation.
  - Greater generality and flexibility of path patterns in policy specifications
  - Addressed administrative control and policy conflict resolution
- Integrated attribute-based policies into ReBAC.
- Provided two effective path checking algorithms for access control policy evaluation.
  - With proof of correctness and complexity analysis
  - Enhanced the algorithms for attribute-aware ReBAC
- Implemented the algorithms and evaluated the performance.

Social Networks

- Social graph is modeled as a directed labeled simple graph $G=<U, E, \Sigma>$
  - Nodes U as users
  - Edges E as relationships
  - $\Sigma=\{\sigma_1, \sigma_2, \ldots, \sigma_n, \sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_n^{-1}\}$
  as relationship types supported

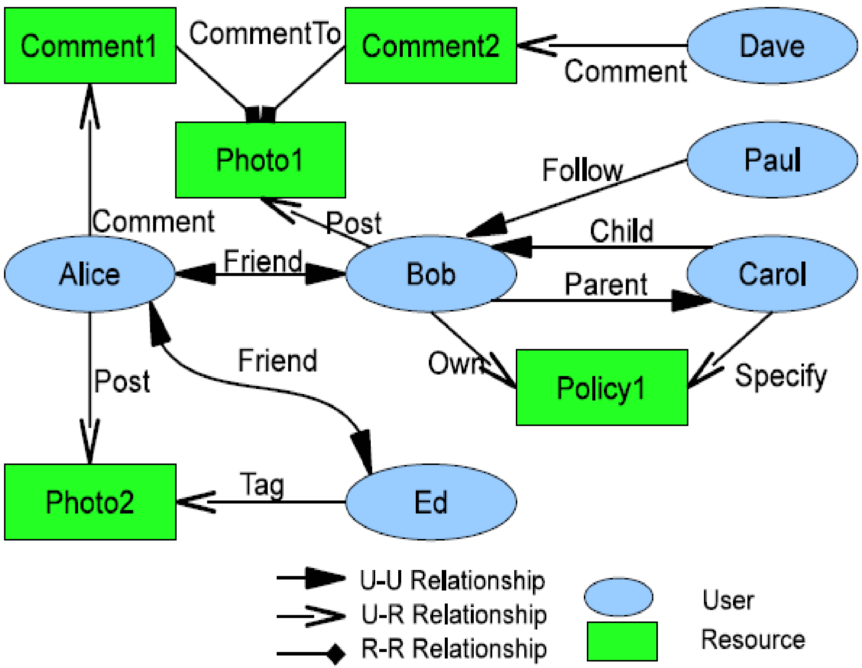


Fig. 3. A Sample Social Graph

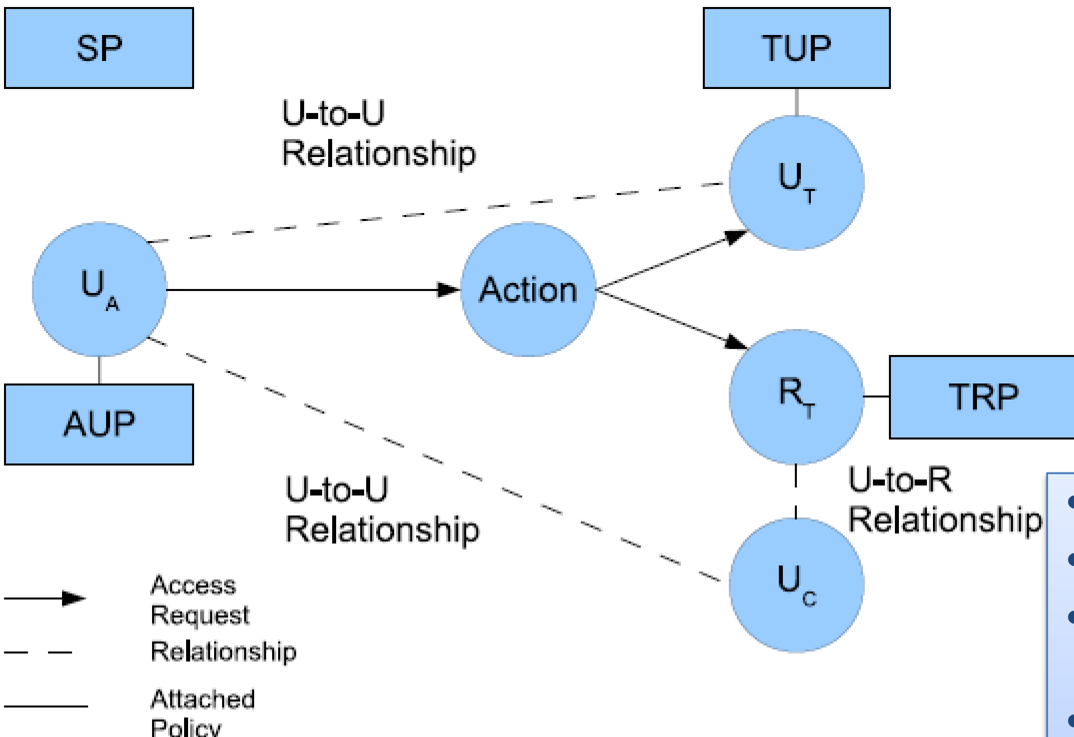
Characteristics of Access Control in OSNs

- Policy Individualization
  - Users define their own privacy and activity preferences
  - Related users can configure policies too
  - Collectively used by the system for control decision
- User and Resource as a Target
  - e.g., poke, messaging, friendship invitation, etc.
- User Policies for Outgoing and Incoming Actions
  - User can be either requester or target of activity
  - Allows control on 1) activities w/o knowing a particular resource and 2) activities against the user w/o knowing a particular access requestor
  - e.g., block notification of friend's activities; restrict from viewing violent contents

Outline

- Introduction
- UURAC
- $UURAC_A$
- URRAC
- Conclusion

U2U Relationship-based Access Control (UURAC) Model



U_A: Accessing User
U_T: Target User
U_C: Controlling User
R_T: Target Resource
AUP: Accessing User Policy
TUP: Target User Policy
TRP: Target Resource Policy
SP: System Policy

- Policy Individualization
- User and Resource as a Target
- Separation of user policies for incoming and outgoing actions
- Regular Expression based path pattern w/ max hopcounts (e.g., <u_a, (f*c,3)>)

Access Request and Evaluation

- Access Request <u_a, *action*, *target*>
  - u_a tries to perform *action* on *target*
  - Target can be either user u_t or resource r_t
- Policies and Relationships used for Access Evaluation
  - When u_a requests to access a user u_t
    - u_a's AUP, u_t's TUP, SP
    - U2U relationships between u_a and u_t
  - When u_a requests to access a resource r_t
    - u_a's AUP, r_t's TRP, SP
    - U2U relationships between u_a and u_c

Policy Representations

Accessing User Policy	$< action, (start, path\ rule) >$
Target User Policy	$< action^{-1}, (start, path\ rule) >$
Target Resource Policy	$< action^{-1}, u_c, (start, path\ rule) >$
System Policy for User	$< action, (start, path\ rule) >$
System Policy for Resource	$< action, (r.typename, r.typevalue), (start, path\ rule) >$

- *$action^{-1}$* in TUP and TRP is the passive form since it applies to the recipient of action
- TRP has an extra parameter *u_c* to specify the controlling user
  - U2U relationships between u_a and u_c
- SP does not differentiate the active and passive forms
- SP for resource needs *r.typename, r.typevalue* to refine the scope of the resource

Example

- Alice's policy P_{Alice}:
  - $< poke, \left(u_a, (f*, 3)\right) >, < poke^{-1}, \left(u_t, (f, 1)\right) >,$
  - $< read, \left(u_a, (\Sigma*, 5)\right) >$
- Harry's policy P_{Harry}:
  - $< poke, \left(u_a, (cf*, 5) \vee (f*, 5)\right) >, < poke^{-1}, \left(u_t, (f*, 2)\right) >$
- Policy of file2 P_{file2}:
  - $< read^{-1}, Harry, (uc, \neg(p+, 2) >$
- System's policy P_{Sys}:
  - $< poke, \left(u_a, (\Sigma*, 5)\right) >$
  - $< read, (filetype, photo), \left(u_a, (\Sigma*, 5)\right) >$

- "Only Me"
  - $< poke, \left(u_a, (\emptyset, 0)\right) >$ says that ua can only poke herself
  - $< poke^{-1}, \left(u_t, (\emptyset, 0)\right) >$ specifies that ut can only be poked by herself
- The Use of Negation Notation
  - $(fffc \wedge \neg fc)$ allows the coworkers of the user's distant friends to see, while keeping away the coworkers of the user's direct friends

Policy Extraction

It determines the starting node, where the evaluation starts

The other user involved in access becomes the evaluating node

- Policy: <*a* … *pe*, *graph rule*>
- Graph Rule: *start*, *path rule*
- Path Rule: *path spec* ∧|∨ *path spec*
- Path Spec: *path*, *hopcount*

Path-check each path spec using Algorithm 2 (introduced in detail later)

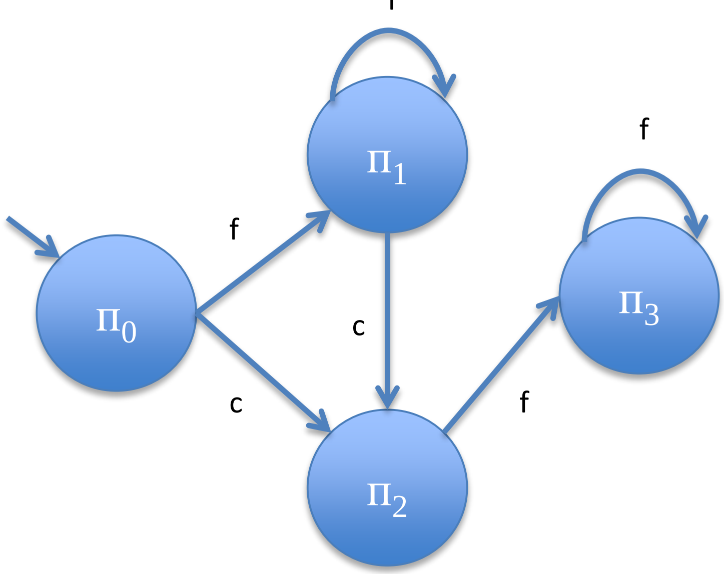
Path Checking Algorithms

- Two strategies: DFS and BFS
- Parameters: G, path, hopcount, s, t

Access Request: (Alice, read, r_t)

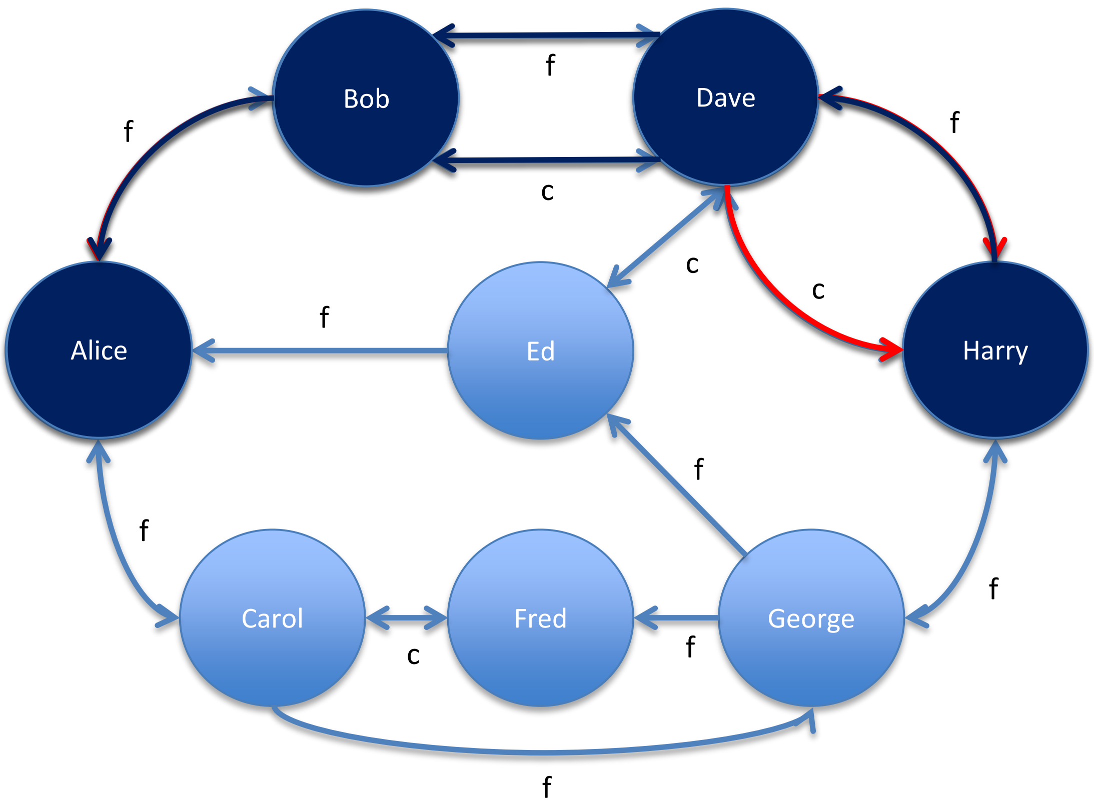
Policy: ($read^{-1}$, r_t, (f*cf*, 3))

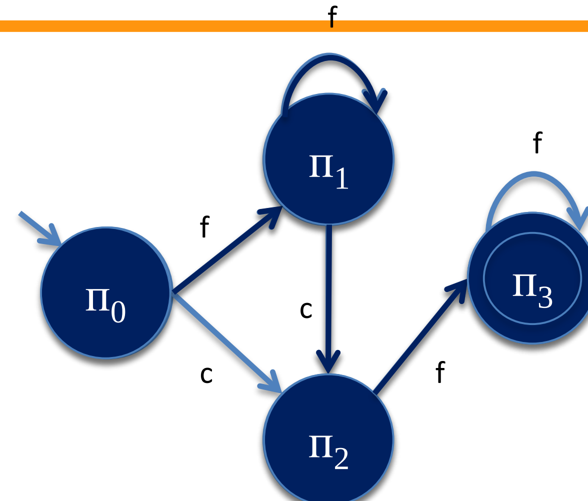
Path pattern: f*cf*
Hopcount: 3



DFA for f*cf*

Path pattern: f*cf*
Hopcount: 3





Case 3: [illegible] ching
[illegible] d an
[illegible] DFA not
at an accepting state

d: 3

currentPath: (H,D,f)(D,B,c)(B,A,f)

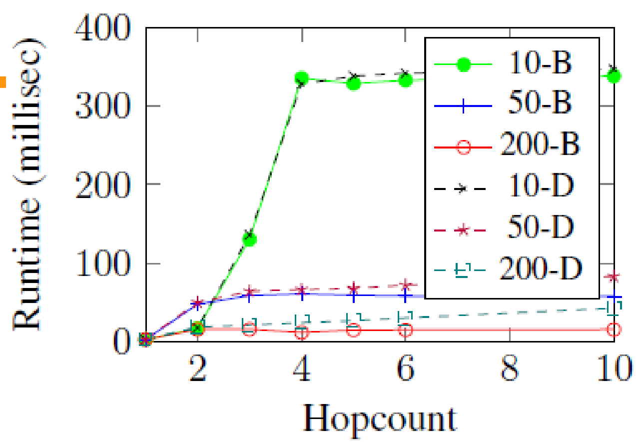
stateHistory: 0123

Complexity

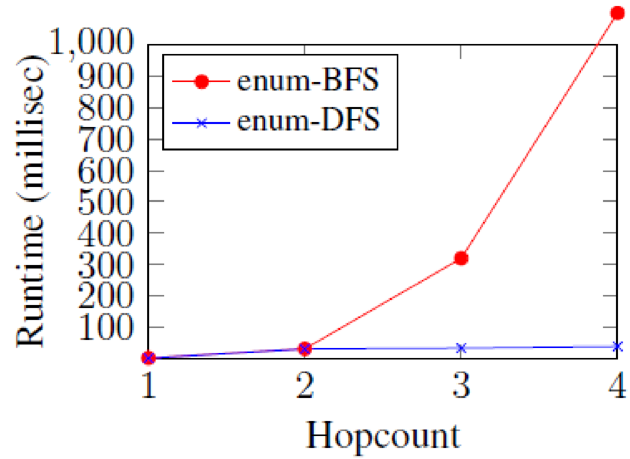
- Time complexity is bounded between $[O(dmin^{Hopcount}), O(dmax^{Hopcount})]$, where *dmax* and *dmin* are maximum and minimum out-degree of node
  – Users in OSNs usually connect with a small group of users directly, the social graph is very sparse
  – Given the constraints on the relationship types and hopcount limit, the size of the graph to be explored can be dramatically reduced

Evaluation

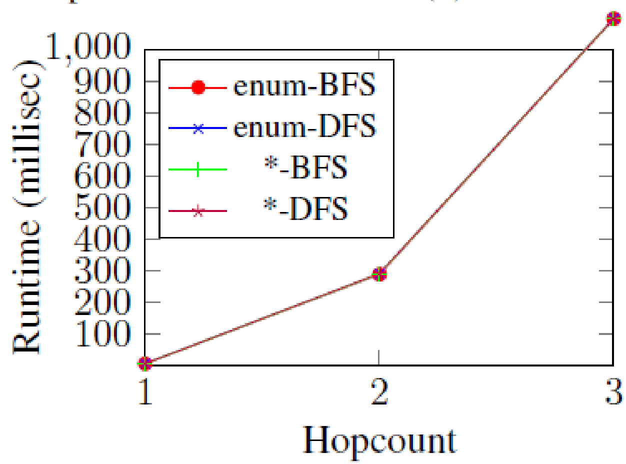
- Experiment 1 examines the performance w.r.t policies with different hopcount limit
  - 1000 users, single relationship type
  - *-pattern and enumeration path
- Experiment 2 studies the performance w.r.t different node degrees
  - 1000 users, two relationship types
  - Various density: 100, 200, 500 and 1000
  - Enumeration path



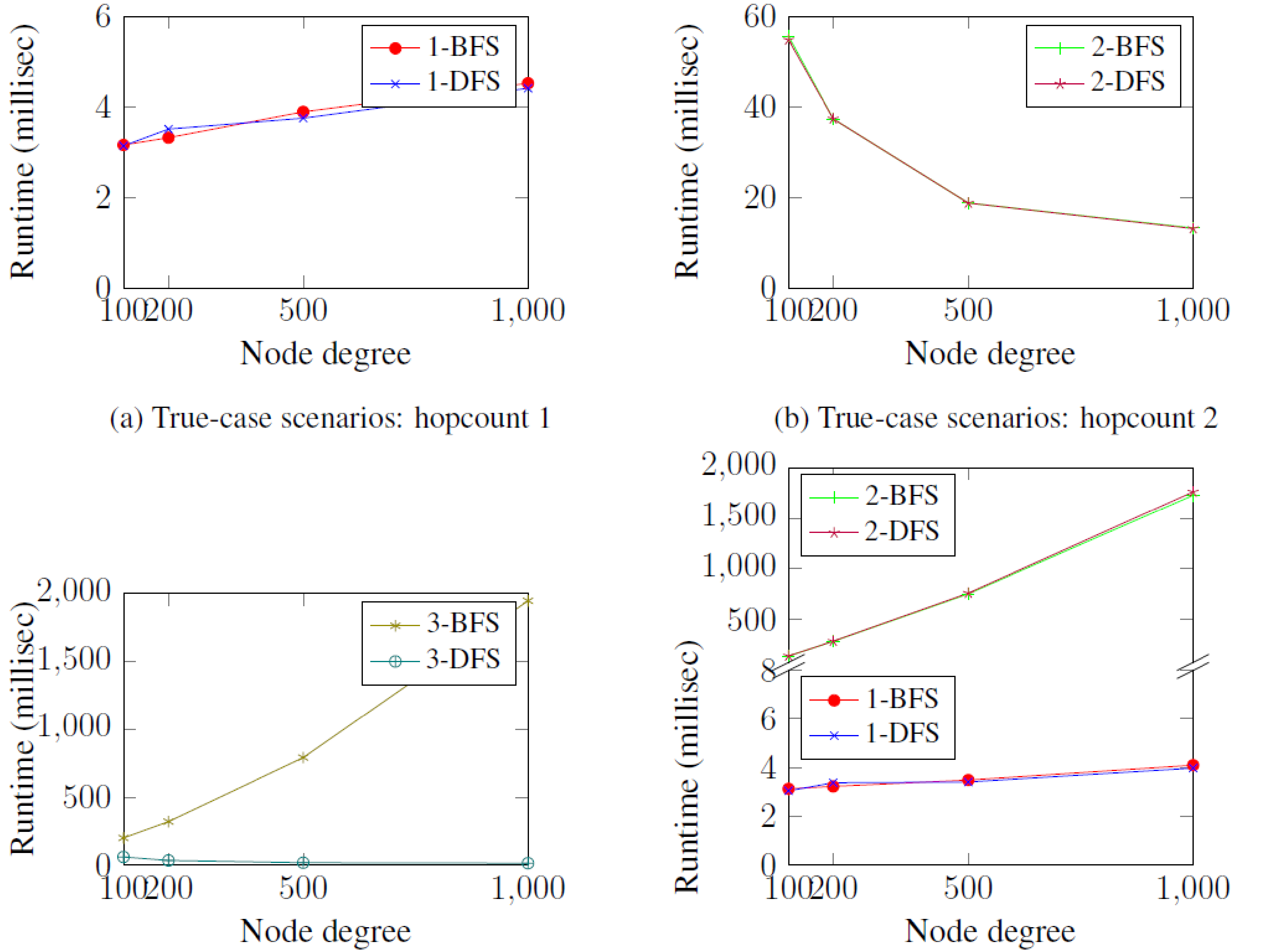
(a) True-case scenarios: *-patterns



(b) True-case scenarios: enum-patterns



(c) False-case scenarios



(a) True-case scenarios: hopcount 1

(b) True-case scenarios: hopcount 2

(c) True-case scenarios: hopcount 3

(d) False-case scenarios

Observations

- Exp. 1:
  - 1) For *-pattern, a qualified path can be always found within 4 hops; BFS outplays DFS for large hopcount in sparse graph
  - 2) For enum-path, time cost of BFS leaps
- Exp. 2:
  - 1) Hopcount increases, search space expands
  - 2) It's more likely to find a path at a shorter time in denser graphs when hopcount is 2
  - 3) BFS suffers from the increase of search space
- In false cases, both are exhaustive search. But large hopcount is barely seen in practical OSN scenarios.
- BFS vs DFS:
  - Similar for 1, 2-hop, but DFS in general better for intermediate hopcount values (3, 4, 5, etc.)

Outline

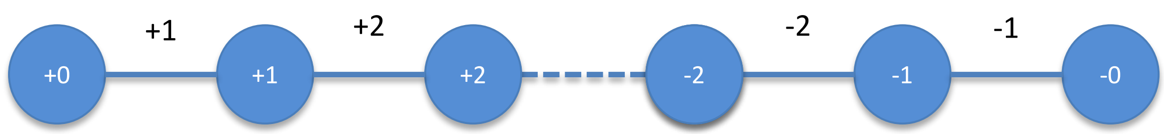
- Introduction
- UURAC
- UURAC_A
- URRAC
- Conclusion

Beyond Relationships

- ReBAC usually relies on type, depth, or strength of relationships, but cannot express more complicated topological information
- ReBAC lacks support for attributes of users, resources, and relationships
- Useful examples include common friends, duration of friendship, minimum age, etc.

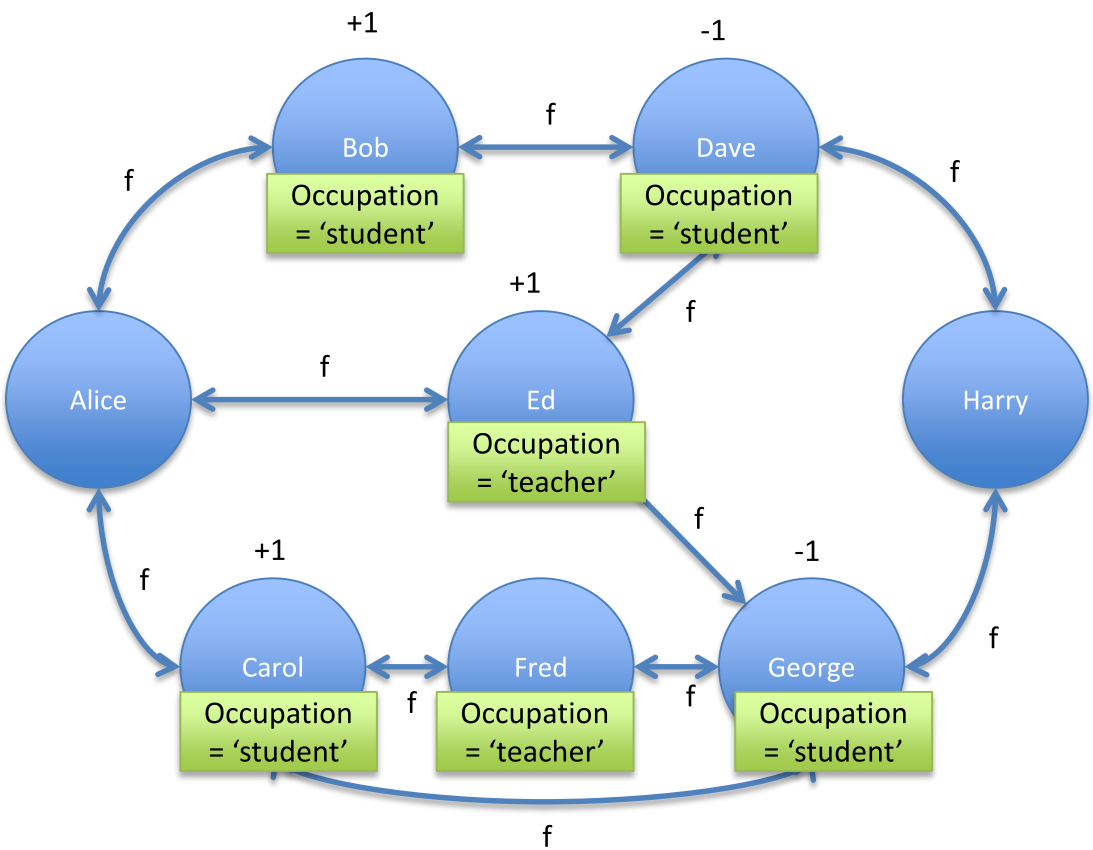
Attribute-based Policy

- $<quantifier, f(ATTR(N), ATTR(E)), count \geq i>$



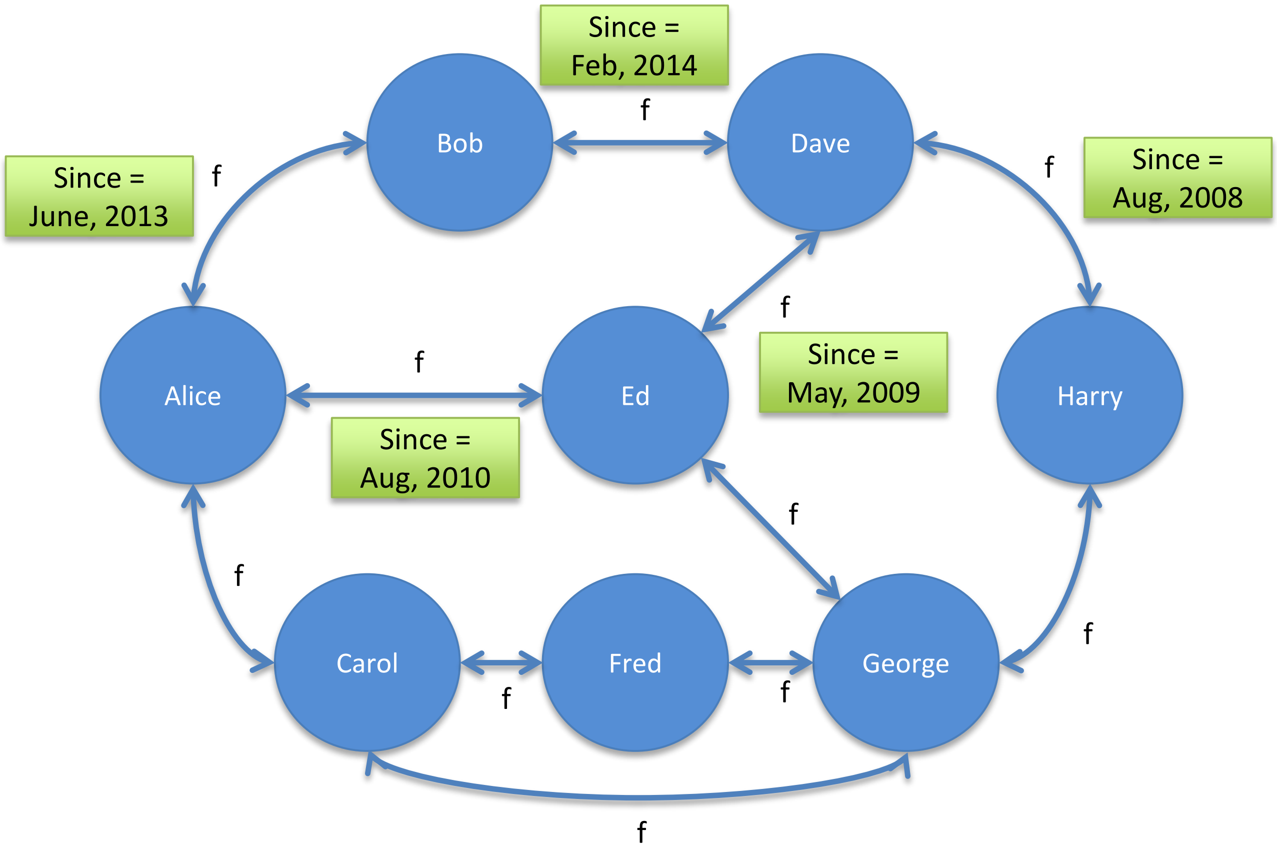
$\forall$[+1, -2], age(u) > 18
$\exists$[+1, -1], weight(e) > 0.5
$\exists${+1, +2, -1}, gender = "male"

Example: Node Attributes



<access, (u_a, ((f^*, 4): ∃[+1, -1], occupation = 'student', count ≥ 3)))>

Example: Edge Attributes



<read, Photo1, (u_a, ((f*, 3): ∀[+1, -1], duration ≥ 3 month, _)))>

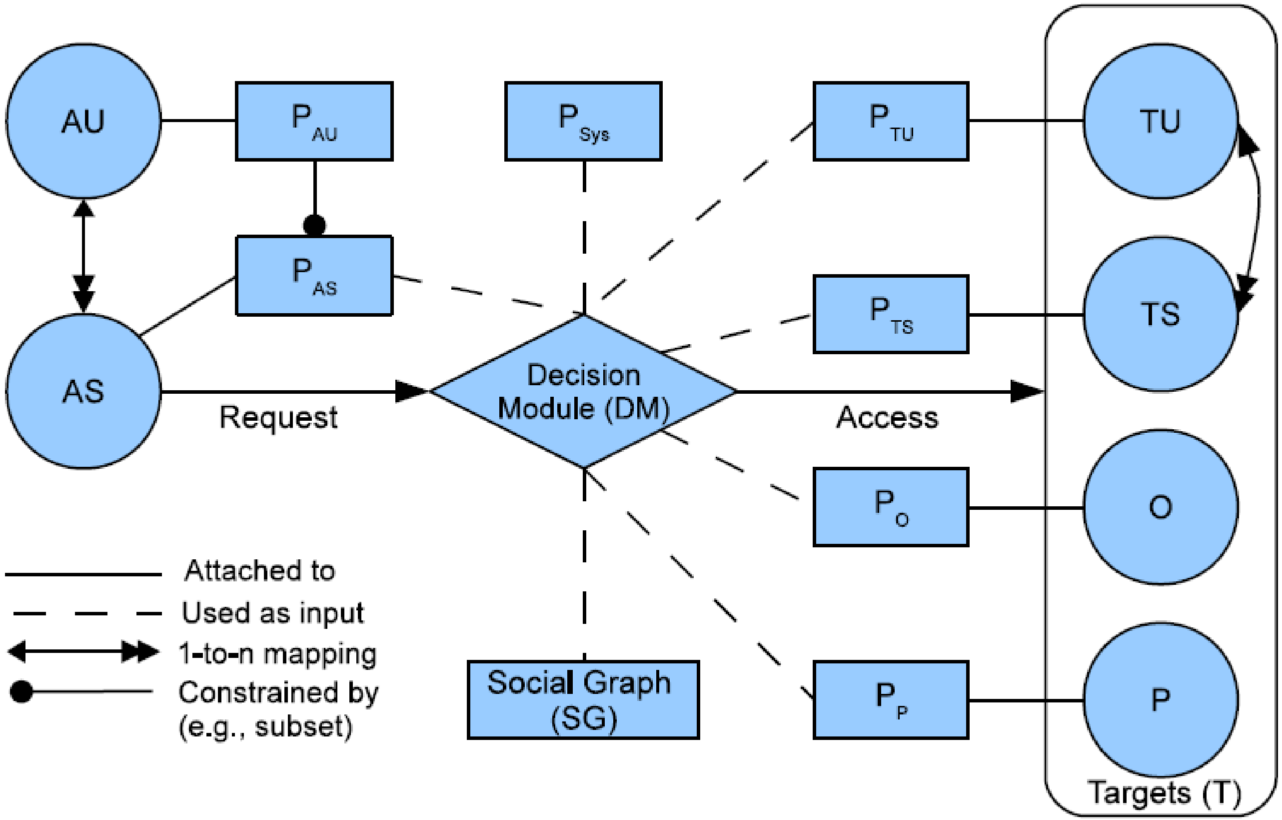
Outline

- Introduction
- UURAC
- $UURAC_A$
- URRAC
- Conclusion

Beyond U2U Relationships

- There are various types of relationships between users and resources in addition to U2U relationships and ownership
  - e.g., share, like, comment, tag, etc
- U2U, U2R and R2R
- U2R further enables relationship and policy administration

URRAC Model Components



AU: Accessing User
AS: Accessing Session
TU: Target User
TS: Target Session
O: Object
P: Policy
P_{AU}: Accessing User Policy
P_{AS}: Accessing Session Policy
P_{TU}: Target User Policy
P_{TS}: Target Session Policy
P_{O}: Object Policy
P_{P}: Policy for Policy
P_{Sys}: System Policy

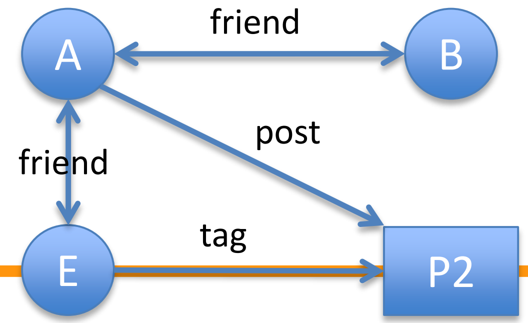
Differences with UURAC

- U2R Relationship-based Access Control
- Access Request
  - (s, act, T) where T may contain multiple objects
- Policy Administration
- User-session Distinction
- Hopcount Skipping
  - Local hopcount stated inside “[[]]” will not be counted in global hopcount.
  - E.g., “([f*,3][[c*, 2]],3)”, the local hopcount 2 for *c** does not apply to the global hopcount 3, thus allowing *f** to have up to 3 hops.

Policy Conflict Resolution

- System-defined conflict resolution for potential conflicts among user-specified policies
- Disjunctive, conjunctive and prioritized order between relationship types
  - $<share^{-1}, (own \vee tag \vee share)>$
  - $<read^{-1}, (own \wedge tag)>$
  - $<friend\_request, (parent > @)>$

Example

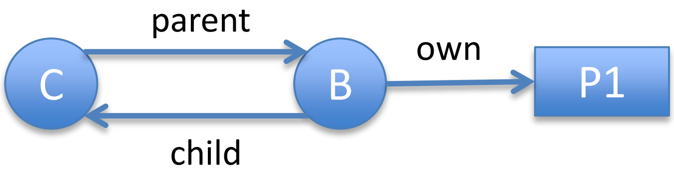


- **View a photo where a friend is tagged.** *Bob and Ed are friends of Alice, but not friends of each other. Alice posted a photo and tagged Ed on it. Later, Bob sees the activity from his news feed and decides to view the photo: (Bob, read, Photo2)*
  - *Bob's* $P_{AS}(read)$: $<read,(u_a,([\Sigma_{u\_u}*,2][[\Sigma_{u\_r},1]],2))>$
  - *Photo2's* $P_O(read^{-1})$ by *Alice*:

    $<read^{-1},(t,([post^{-1},1][friend*,3],4))>$
  - *Photo2's* $P_O(read^{-1})$ by *Ed*: $<read^{-1},(u_c,([friend],1)$

    In conflicts
  - $AP_{Sys}(read)$: $<read,(ua,([\Sigma_{u\_u}*,5][[\Sigma_{u\_r},1]],5))>$
  - $CRP_{Sys}(read)$: $<read^{-1},(own\wedge tag)>$

Example



- **Parental control of policies.** *The system features parental control such as allowing parents to configure their children's policies. The policies are used to control the incoming or outgoing activities of children, but are subject to the parents' will. For instance, Bob's mother Carol requests to set some policy, say Policy1 for Bob: (Carol, specify policy, Policy1)*
  - *Carol's* P_{AS}*(specify_policy)*: $<specify\_policy,(u_a,([own],1)\vee([child\cdot own],2))>$
  - *Policy1's* P_P*(specify_policy*$^{-1}$*)* by *Bob*: $<specify\_policy^{-1},(t,([own^{-1}],1))>$
  - P_{Sys}*(specify_policy)*: $<specify\_policy,(u_a,([own],1)\vee([child\cdot own],2))>$
  - CRP_{Sys}*(specify_policy)*: $<specify\_policy, (parent\wedge @)>$

Outline

- Introduction
- UURAC
- $UURAC_A$
- URRAC
- Conclusion

Comparison with Our Approach

	Fong [24]	Fong [23, 25]	Carminati [15]	Carminati [11, 12]	UURAC, URRAC & UURAC$_A$
Relationship Category					
Multiple Relationship Types		✓	✓	✓	✓
Directional Relationship		✓	✓		✓
U2U Relationship	✓	✓	✓	✓	✓
U2R Relationship				✓	✓(only URRAC)
Model Characteristics					
Policy Individualization	✓	✓	✓	✓	✓
User & Resource as a Target				(partial)	✓
Outgoing/Incoming Action Policy				(partial)	✓
Relationship Composition					
Relationship Depth	0 to 2	0 to n	1 to n	1 to n	0 to n
Relationship Composition	f, f of f	exact type sequence	path of same type	exact type sequence	path pattern of different types
Attribute-aware Access Control					
Common-friends$_k$	✓				✓(only UURAC$_A$)
User Attributes		(partial)			✓(only UURAC$_A$)
Relationship Attributes			(partial)		✓(only UURAC$_A$)

- Passive form of action allows outgoing and incoming action policy
- Path pattern of different relationship types and hopcount skipping make policy specification more expressive
- Attribute-aware access control based on attributes of users and relationships
- System-level conflict resolution policy

Publications

1. **Yuan Cheng**, Jaehong Park and Ravi Sandhu, An Access Control Model for Online Social Networks Using User-to-user Relationships. Submitted to IEEE TDSC.
2. **Yuan Cheng**, Jaehong Park and Ravi Sandhu, Attribute-aware Relationship-based Access Control for Online Social Networks. Submitted to DBSec 2014.
3. **Yuan Cheng**, Jaehong Park and Ravi Sandhu, Relationship-based Access Control for Online Social Networks: Beyond User-to-User Relationships. In Proceedings 4th IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT), Amsterdam, Netherlands, September 3-5, 2012. (Winner of Best Paper Award)
4. **Yuan Cheng**, Jaehong Park and Ravi Sandhu, A User-to-User Relationship-based Access Control Model for Online Social Networks. In Proceedings 26th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy (DBSec 2012), Paris, France, July 11-13, 2012.
5. Jaehong Park, Ravi Sandhu and **Yuan Cheng**, ACON: Activity-Centric Access Control for Social Computing. Proceedings 5th International Conference on Availability, Reliability and Security (ARES), Vienna, Austria, August 22-26, 2011.
6. Jaehong Park, Ravi Sandhu and **Yuan Cheng**, A User-Activity-Centric Framework for Access Control in Online Social Networks. IEEE Internet Computing, 15(5): 62-65, September 2011.
7. **Yuan Cheng**, Jaehong Park and Ravi Sandhu, Preserving User Privacy from Third-party Applications in Online Social Networks. In Proceedings of the 2nd International Workshop on Privacy and Security in Online Social Media (PSOSM), Rio de Janeiro, Brazil, May 14, 2013. (Runner-up of The Best Paper Award)
8. Jaehong Park, **Yuan Cheng** and Ravi Sandhu, Towards A Framework for Cyber Social Status Based Trusted Open Collaboration. In Proceedings of the 5th IEEE International Workshop on Trusted Collaboration (TrustCol 2010), Chicago, Illinois, October 9, 2010.
9. **Yuan Cheng**, Dang Nguyen, Khalid Bijon, Ram Krishnan, Jaehong Park and Ravi Sandhu, Towards Provenance and Risk-Awareness in Social Computing. First International Workshop on Secure and Resilient Architectures and Systems, Minneapolis, Minnesota, September 19, 2012.

Summary

- UURAC
  - Proposed a U2U relationship-based model and a regular expression-based policy specification language for OSNs
  - Provided a DFS-based path checking algorithm
- URRAC
  - Proposed a U2U, U2R and R2R relationship-based access control model for users' usage and administrative access in OSNs
    - Access control policies are based on regular expression-based path patterns
    - Hopcount skipping for more expressiveness
  - Provided a system-level conflict resolution policies based on relationship precedence
- $UURAC_A$
  - Incorporated attribute-awareness to UURAC model
  - Enhanced the path checking algorithm

Future Research

- Access control for 3rd party applications
  - Current strategy: all-or-nothing
  - Apps often gain much more rights than necessary
- User-specified conflict resolution policy
  - Specified by users
  - Applies to a smaller context
  - Raises ambiguity
- Unconventional relationships

Questions/Comments

Numbers and Facts

- **Survey Data from PEW Internet (2011)**
  - 47% of American adults use at least one OSN.
  - close to double the 26% of adults who used an OSN in 2008.
- **Statistics from Facebook**
  - One billion monthly active users as of Oct 2012.
  - 552 million daily active users on average in June 2012.
  - 600 million monthly active users who used Facebook mobile products in Sep 2012.

Control on Social Interactions

- A user wants to control other users' access to her own shared information
  - Only friends can read my post
- A user wants to control other users' activities who are related to the user
  - My children cannot be a friend of my co-workers
  - My activities should not be notified to my co-workers
- A user wants to control her outgoing/incoming activities
  - No accidental access to violent contents
  - Do not poke me
- A user's activity influences access control decisions
  - Once Alice sends a friend request to Bob, Bob can see Alice's profile

Privacy Breaches

- Easy to happen from OSN providers, other users, and 3rd party applications
- OSN providers store user data
  - Users have to trust OSNs to protect and not to misuse the data
  - OSNs can benefit from analyzing and sharing the data (e.g., targeted advertisement)
- 3rd party applications provide extra functionalities
  - Simply all-or-nothing control
  - Access to more information than actual need
  - Be able to post or access user data without user's knowledge
- Another major threats are from *peer users*
  - Not aware of who they share with and how much
  - Have difficulty in managing privacy controls

Limitation of U2U Relationships

- We rely on the controlling user and ownership to regulate access to resources in UURAC (U2U Relationship-based AC)

- Needs more flexible control
  - Parental control, related user's control (e.g., tagged user)
  - User relationships to resources (e.g., U-U-R)
  - User relationships via resources (e.g., U-R-U)

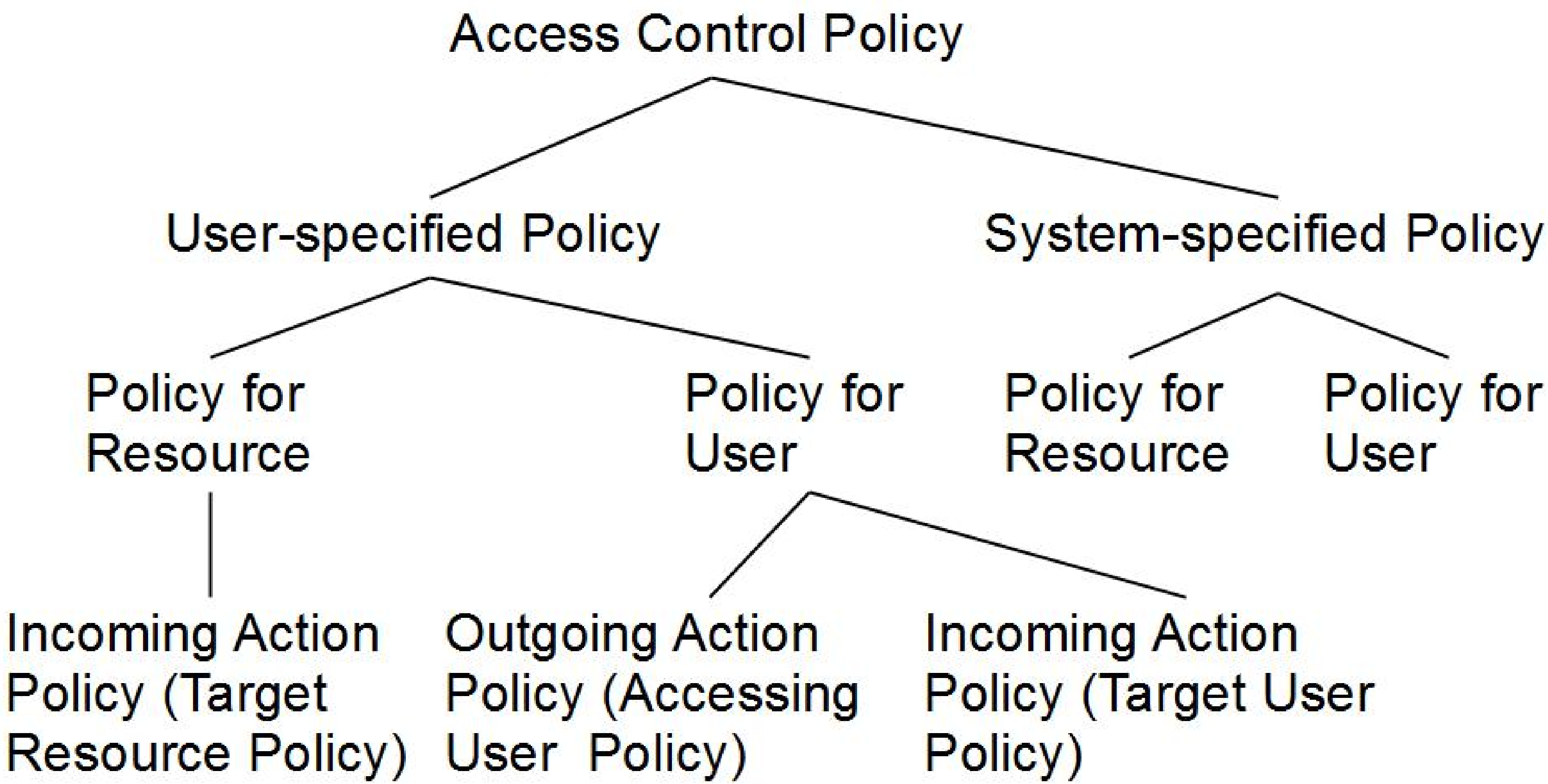
Motivating Examples

- Related User's Control
  - There exist several different types of relationships in addition to *ownership*
  - e.g., Alice and Carol want to control the release of Bob's photo which contains Alice and Carol's image.
  - e.g., Betty shares Ed's original post and acquires the ability to decide how the shared post can be available to others.

Motivating Examples (cont.)

- Administrational Control
  - Policy administration is important
  - A change of relationship may result in a change of authorization
  - Treat *administrative activities* different from normal activities
    - Policy specifying, relationship invitation and relationship recommendation
  - e.g., Bob's mother Carol may not want Bob to become a friend with her colleagues, to access any violent content or to share personal information with others.

Policy Taxonomy



UUURAC Graph Rule Grammar

$GraphRule ::= \text{"("} < StartingNode > \text{","} < PathRule > \text{")"}$
$PathRule ::= < PathSpecExp > | < PathSpecExp >< Connective >< PathRule >$
$Connective ::= \vee | \wedge$
$PathSpecExp ::= < PathSpec > | \neg < PathSpec >$
$PathSpec ::= \text{"("} < Path > \text{","} < HopCount > \text{")"} | \text{"("} < EmptySet > \text{","} < Hopcount > \text{")"}$
$HopCount ::= < Number >$
$Path ::= < TypeExp > | < TypeExp >< Path >$
$EmptySet ::= \emptyset$
$TypeExp ::= < TypeSpecifier > | < TypeSpecifier >< Wildcard >$
$StartingNode ::= u_a | u_t | u_c$
$TypeSpecifier ::= \sigma_1 | \sigma_2 | .. | \sigma_n | \sigma_1^{-1} | \sigma_2^{-1} | .. | \sigma_n^{-1} | \Sigma$ where $\Sigma = \{\sigma_1, \sigma_2, .., \sigma_n, \sigma_1^{-1}, \sigma_2^{-1}, .., \sigma_n^{-1}\}$
$Wildcard ::= \text{"} * \text{"} | \text{"?"} | \text{"} + \text{"}$
$Number ::= [0 - 9]+$

World-Leading Research with Real-World Impact!

Policy Evaluation

- Evaluate a combined result based on conjunctive or disjunctive connectives between path specs
- Make a collective result for multiple policies in each policy set.
  - Policy conflicts may arise. We assume system level conflict resolution strategy is available (e.g., disjunctive, conjunctive, prioritized).
- Compose the final result from the result of each policy set (AUP, TUP/TRP, SP)

Policy Collecting

- To authorize (u_a, action, target) if target = u_t
  - E.g., (Alice, poke, Harry)

AUP

TUP

SP

P_{Alice}

<poke, (ua, (f*,3))>

< poke^{-1}, (ua, (f*,3))>

P_{Harry}

<poke, (ua, (cf*,5)∨(f*,5))>

<poke^{-1}, (ut, (f*,2))>

P_{Sys}

<poke, (ua, (Σ*,5))>

Policy Collecting

- To authorize (u_a, action, target) if target = r_t
  - Determine the controlling user for r_t:
    - $u_c \leftarrow$ owner(r_t)
  - E.g., (Alice, read, file2)

AUP

TRP

SP

P_{Alice}

<read, (u_a, (Σ*, 5))>

<$read^{-1}$, file1, (u_c, (cf*, 4))>

P_{Harry}

<$read^{-1}$, file2, (u_c, ¬(p+, 2))>

P_{Sys}

<read, photo, (ua, (Σ*, 5))>

Additional Characteristics of URRAC

- Policy Administration
  - Policy and Relationship Management
  - Users specify policies for other users and resources
- User-session Distinction
  - A user can have multiple sessions with different sets of privileges
  - Especially useful in mobile and location-based applications

URRAC Action and Access Request

- $ACT = \{act_1, act_2, . . ., act_n\}$ is the set of OSN supported actions
- Access Request <*s*, *act*, *T*>
  - *s* tries to perform *act* on *T*
  - Target $T \subseteq (2^{TU \cup R} - \emptyset)$ is a non-empty set of users and resources
    - T may contain multiple targets

URRAC Authorization Policy

Accessing User Policy	$< act, graphrule >$
Accessing Session Policy	$< act, graphrule >$
Target User Policy	$< act^{-1}, graphrule >$
Target Session Policy	$< act^{-1}, graphrule >$
Object Policy	$< act^{-1}, graphrule >$
Policy for Policy	$< act^{-1}, graphrule >$
System Policy for User	$< act, graphrule >$
System Policy for Resource	$< act, o.type, graphrule >$ where $o.type$ is optional

- *action*$^{-1}$ in TUP, TSP, OP and PP is the passive form since it applies to the recipient of action
- SP does not differentiate the active and passive forms
- SP for resource needs *o.type* to refine the scope of the resource

URRAC Graph Rule Grammar

$GraphRule \rightarrow$ "(" $StartingNode$ "," $PathRule$ ")"
$PathRule \rightarrow PathSpecExp \,|\, PathSpecExp\ Connective\ PathRule$
$Connective \rightarrow \vee \,|\, \wedge$
$PathSpecExp \rightarrow PathSpec \,|$ "¬" $PathSpec$
$PathSpec \rightarrow$ "(" $Path$ "," $HopCount$ ")" | "(" $EmptySet$ "," $HopCount$ ")"
$HopCount \rightarrow Number$
$Path \rightarrow$ ["[" $TypeSeq$ "]" | "[" $TypeSeq$ "," $HopCount$ "]" | "[[" $TypeSeq$ "," $HopCount$ "]]"]+
$EmptySet \rightarrow \emptyset$
$TypeSeq \rightarrow TypeExp$ {"." $TypeExp$}
$TypeExp \rightarrow TypeSpecifier \,|\, TypeSpecifier\ Wildcard$
$StartingNode \rightarrow u_a | u_c | t$
$TypeSpecifier \rightarrow \sigma_1 | \sigma_2 | \ldots | \sigma_n | \sigma_1^{-1} | \sigma_2^{-1} | \ldots | \sigma_n^{-1} | \Sigma$ where $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n, \sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_n^{-1}\}$
$Wildcard \rightarrow$ "$*$" | "?" | "$+$"
$Number \rightarrow [0-9]+$

Hopcount Skipping

- U2R and R2R relationships may form a long sequence
  - Omit the distance created by resources
  - Local hopcount stated inside “[[]]” will not be counted in global hopcount.
  - E.g., “([*f**,3][[*c**, 2]],3)”, the local hopcount 2 for *c** does not apply to the global hopcount 3, thus allowing *f** to have up to 3 hops.

Policy Conflict Resolution (cont.)

- $< read^{-1}, (own \wedge tag) >$
  - The more rigid one between the owner's and the tagged users' "read-1" policies over the photo is honored.
- $< friend\_request, (parent > @) >$
  - When child attempts friendship request to someone, parents' policies get precedence over child's own will.
- $< share^{-1}, (own \vee tag \vee share) >$
  - A weblink is sharable if either the original owner, or any of the tagged users or shared users allows.

Attribute Policy Taxonomy

$\forall$ [+m, -n]	All entities from the mth to the nth last, $m+n \leq h$ where m and n are non-negative integers and h is a hopcount limit
$\forall$ [+m, +n]	All entities from the mth to the nth, $m \leq n \leq h$
$\forall$ [-m, -n]	All entities from the mth last to the nth last, $h \geq m \geq n$
$\exists$ [+m, -n]	One entity from the mth to the nth last, $m + n \leq h$
$\exists$ [+m, +n]	One entity from the mth to the nth, $m \leq n \leq h$
$\exists$ [-m, -n]	One entity from the mth last to the nth last, $h \geq m \geq n$
$\forall$ $\{2^{\{\pm N\}}\}$	All entities in this set
$\exists$ $\{2^{\{\pm N\}}\}$	One entity in this set

I·C·S
The Institute for Cyber Security

UTSA