Relationship-based Access Control for Online Social Networks: Beyond User-to-User Relationships

Sep. 3, 2012
PASSAT 2012, Amsterdam, The Netherlands

Yuan Cheng, Jaehong Park and Ravi Sandhu
Institute for Cyber Security
University of Texas at San Antonio

Outline

- Motivation
- Model Components
- Model
- Use Cases
- Conclusions

Relationship-based Access Control

- Users in Online Social Networks (OSNs) are connected with 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

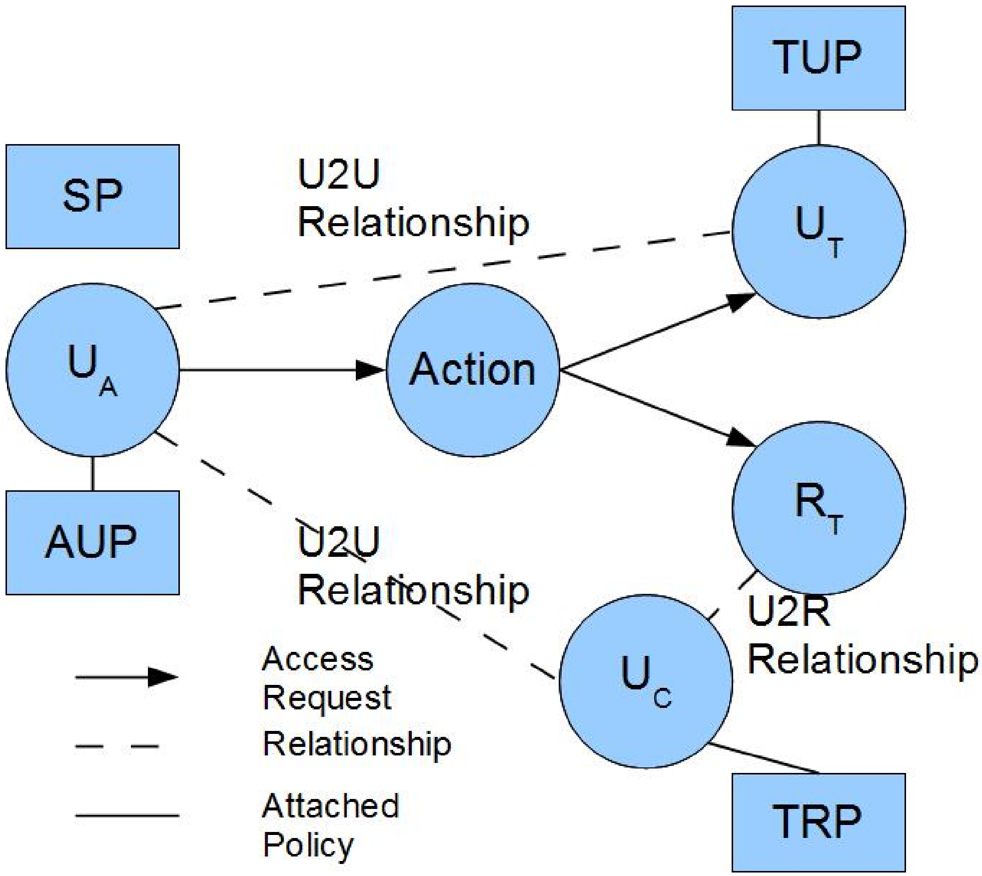
Sharings in Online Social Networks

- Online Social Networks provide services to promote information sharing by utilizing user activity information and shared contents
- Users share information with other users
  - A user creates information to share with other users.
  - A user sends information to other users. (e.g., poke, invite)
  - A user receives information from/about other users.
  - Information about a user's sharing activity is shared.
- Both resource and user as a target of sharing activity
  - Alice pokes bob

Controls in Online Social Networks

- 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 coworkers
- 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

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)>)

Limitation of U2U Relationships

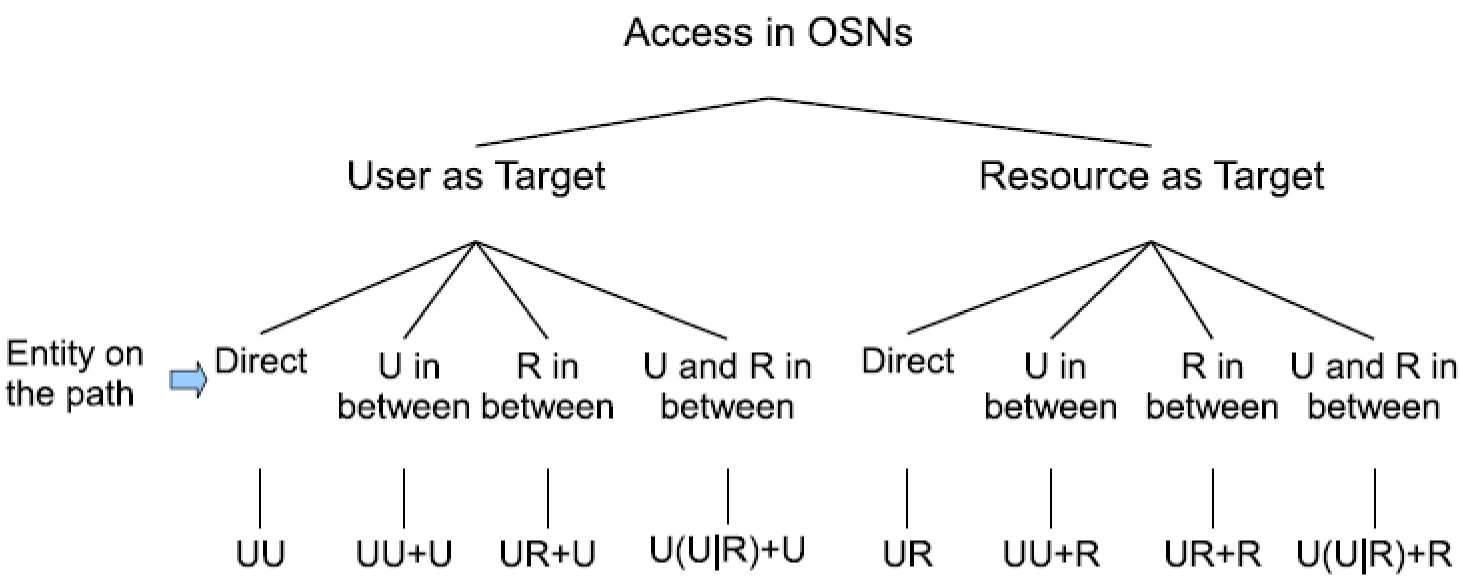
- We rely on the controlling user and ownership to regulate access to resources in UURAC (U2U Relationship-based AC)

- 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)

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

Access Scenarios



(b) Taxonomy based on Entity on the path

Related Works

- Access Control Models for OSNs

COMPARISON OF ACCESS CONTROL MODELS FOR OSNs

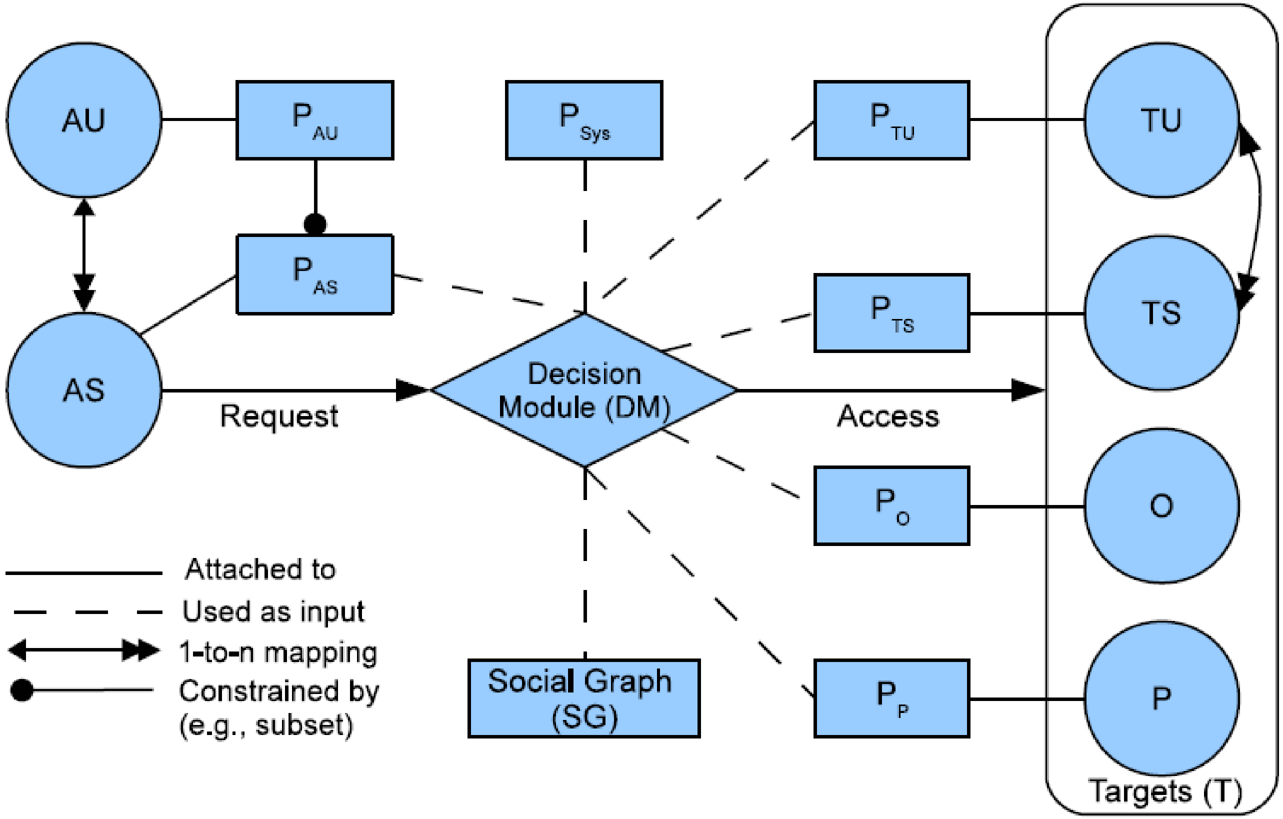
	Fong [14]	Fong [15], [16]	Carminati [10]	Carminati [6], [7]	UURAC	URRAC
Relationship Category						
Multiple Relationship Types		✓	✓	✓	✓	✓
Directional Relationship		✓	✓		✓	✓
U2U Relationship	✓	✓	✓	✓	✓	✓
U2R Relationship				✓		✓
Model Characteristics						
Policy Individualization	✓	✓	✓	✓	✓	✓
User & Resource 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	0 to n
Relationship Composition	f, f of f	exact type sequence	path of same type	exact type sequence	path pattern of different types	path pattern of different types, hopcount skipping

- The advantages of URRAC:
  - Path pattern of different relationship types and hopcount skipping make policy specification more expressive
  - System-level conflict resolution policy

Outline

- Motivation
- Model Components
- Model
- Use Cases
- Conclusions

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

Outline

- Motivation
- Model Components
- Model
- Use Cases
- Conclusions

Characteristics of URRAC 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
- 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
- Relationship-based Access Control

Social Networks

- Social graph is modeled as a directed labeled simple graph $G=<V, E, \Sigma>$
  - $V = U \cup R$, where U is users and R is resources
  - 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

URRAC Social Graph

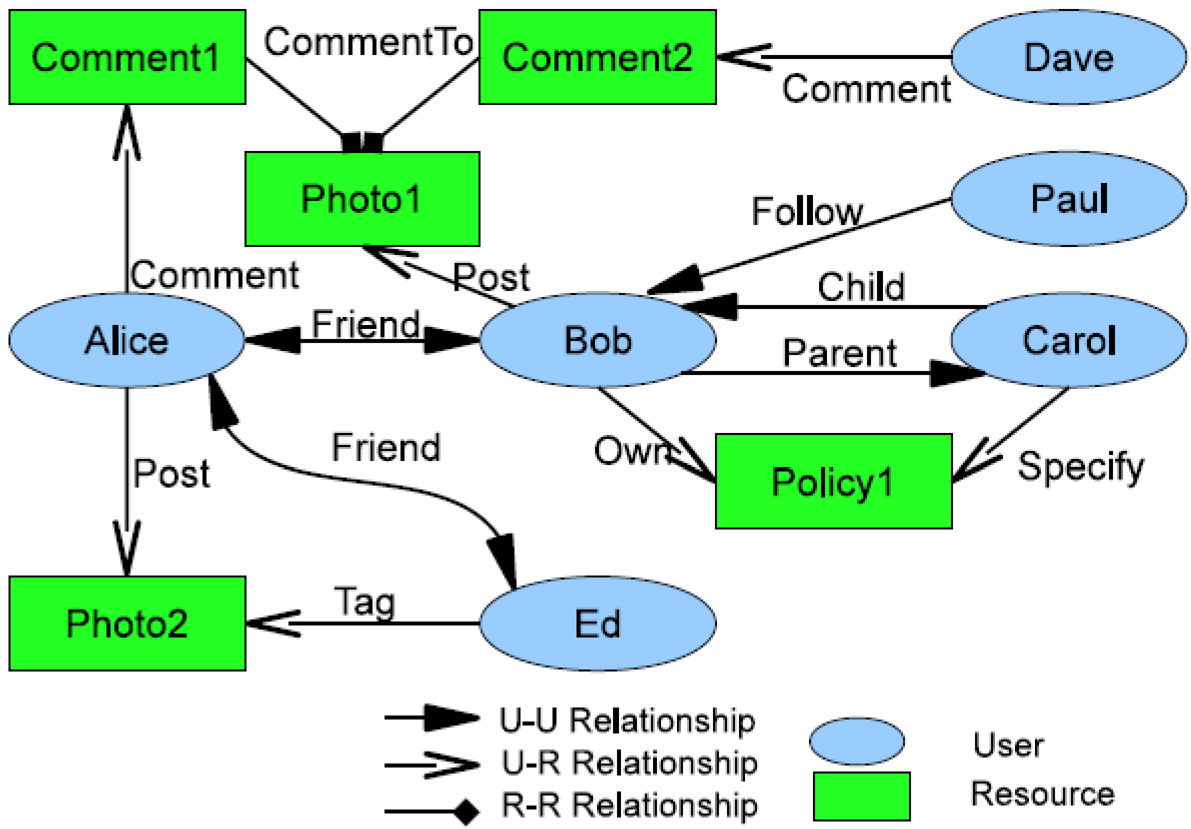


Fig. 3. A Sample Social Graph

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

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

Graph Rule Grammar

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

Hopcount Skipping

- Six degrees of separation
  - Any pair of persons are distanced by about 6 people on average. (4.74 shown by recent study)
  - Hopcount for U2U relationships is practically small
- 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

- System-defined conflict resolution for potential conflicts among user-specified policies
- Disjunctive, conjunctive and prioritized order between relationship types
  - ∧,∨, > represent disjunction, conjunction and precedence
  - @ is a special relationship “null’’ that denotes “self”

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.

Access Evaluation Procedure

- Policy Collecting
  - To authorize <*s, act, T*>, we need the following policies:
    - *s*'s session policy about *act*
    - a collection of act^{-1} policies from each target in *T*
    - system policies over *act* and *object type*, if target is an object

Policy Extraction

- Policy: <*ac* ... *e)*, *graph rule*>

It determines the starting node, where the evaluation starts

- Graph Rule: *start*, *path rule*
- Path Rule: *path spec* ∧|∨ *path spec*
- Path Spec: *path*, *hopcount*

If *s* is *start*, then every *t* in *T* (and u_c) becomes the evaluating node; otherwise, *s* is the evaluating node.

Path-check each path spec using Algorithm 2 in Cheng et al [11]

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 apply CRP_{Sys} to resolve conflicts.
- Compose the final result from the result of each policy set (P_{AS}, $P_{TU}/P_{TS}/P_{O}/P_{P}$, P_{Sys})

Outline

- Motivation
- Model Components
- Model
- Use Cases
- Conclusions

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 (cont.)

- **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_α,([own],1)$\vee$([child·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_α,([own],1)$\vee$([child·own],2))>*
  - CRP_{Sys}(*specify_policy*): *<specify_policy, (parent $\wedge$ @)>*

Outline

- Motivation
- Model Components
- Model
- Use Cases
- Conclusions

Summary

- 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

Future Work

- Incorporate attribute-based controls
- Extend DFS-based path checking algorithm to cover U2R and R2R relationships
- Undertake performance and scalability tests

Questions?