$PBAC_B$ Model

Jaehong Park

University of Texas at San Antonio (UTSA), San Antonio, Texas, USA, 78249
jae.park@utsa.edu
1. \( AU, A, AT, O \) and \( OR \) are acting users, action instances, action types, objects, and object roles respectively.

2. \( G, U, G^{-1} \) and \( U^{-1} \) are sets of role-specific variations of ‘wasGeneratedBy’ and ‘used’ dependencies and matching sets of inverse dependencies, respectively.

3. \{‘c’, ‘c^{-1}’\} is the set of ‘wasControlledBy’ dependency and its inverse dependency.

4. Base provenance data \( PD_B \) forms a directed graph and is formally denoted as a triple \( < V_B, E_B, D_B > \):
   - \( V_B = AU \cup A \cup O \), a finite set of acting users, action instances, and objects that have been involved in transactions in the system and are represented as vertices;
   - \( D_B = \{‘c’\} \cup U \cup G \cup \{‘c^{-1}’\} \cup U^{-1} \cup G^{-1} \), a finite set of base dependency types;
   - \( E_B \subseteq \{(A \times AU \times ‘c’) \cup (A \times O \times U) \cup (O \times A \times G) \cup (AU \times A \times ‘c^{-1}’) \cup (O \times A \times U^{-1}) \cup (A \times O \times G^{-1})\} \), denoting dependency edges, is the set of existing base dependencies in the provenance data.
PBAC\(_B\) Model Definition(2/3)

1. \(D_{NO}\), disjoint from \(D_B\), is a finite set of abstracted names for dependencies of objects.

2. Let \(\Sigma\) be an alphabet of terms in \(D_B \cup D_{NO}\). The set \(DPATH\) of regular expressions is inductively defined as follows:
   - \(\forall p \in \Sigma, p \in DPATH; \epsilon \in DPATH\);
   - \((P_1|P_2), (P_1.P_2), P_1*, P_1+, P_1? \in DPATH\), where \(P_1 \in DPATH\) and \(P_2 \in DPATH\).

3. \(DPATH_B \subseteq DPATH\), is the set of regular expression using only alphabet of terms in \(D_B\).

4. \(DL_O : D_{NO} \rightarrow DPATH\), defines each \(dn \in D_{NO}\) as a path expression. \(DL_O\) is also viewed as a list of pairs of object dependency names and corresponding dependency paths.

5. \(\lambda_O : D_{NO} \rightarrow DPATH_B\), maps each \(dn \in D_{NO}\) to a path expression using only base dependency types \(d_b \in D_B\) by repeatedly expanding the definitions of any \(dn_i \in D_{NO}\) that occurs in \(DL_O(dn)\).
PBAC_B Model Definition (3/3)

1. \( PE \) is a language specified in the policy expression grammar \( PG \).
2. \( P \subseteq PE \), is a finite set of policies.
3. \( \gamma : AT \rightarrow P \), a mapping of an action type to a policy.
4. \( \delta_O : O \times DPATH_B \rightarrow 2^{V_B} \), a function mapping an object and a base dependency path to vertices in \( PD_B \) such that \( o_2 \in \delta(o_1, dpath) \) iff there exists a path in \( PD_B \) from \( o_1 \) to \( o_2 \) whose edge labels form a string that satisfies the regular expression \( dpath \).
Algorithm 1 $AccessEvaluation(au, a, O)$

1. (Rule Collecting Phase)
2. $at \leftarrow a$’s action type
3. $p \leftarrow \gamma(at)$
4. $RULE_{UA} \leftarrow$ user authorization rules $UARule$ found in $p$
5. $RULE_{AV} \leftarrow$ action validation rules $AVRule$ found in $p$
6. (User Authorization Phase)
7. (Action Validation Phase)
8. Evaluate a final truth value of $UAuth$ and $AVal$ using conjunctive connective
PBAC_B Model Access Evaluation (2/3)

1: (User Authorization Phase)
2: for all rules in RULE_{UA} do
3: Extract the path rule (ObjRole, DName) from rules
4: Determine the object o ∈ O, whose role is ObjRole
5: Extract dependency path expression dpath_{b} in DPATH_{B} from DName using λ_{O} function
6: Determine vertices by tracing base provenance data PD_{B} through the paths expressed in dpath_{b} that start from the object o using δ_{O} function
7: Determine the truth value by evaluating the result against the rule
8: end for
9: UAuth ← a combined truth value based on conjunctive or disjunctive connectives between rules
1: (Action Validation Phase)
2: \textbf{for all rules in } \textit{RULE}_{AV} \textbf{ do}
3: \hspace{1em} Extract path rules \((\textit{ObjRole}, \textit{DName})\) from \textit{rules}
4: \hspace{1em} \textbf{for all path rules extracted do}
5: \hspace{2em} Determine the object \(o \in O\), whose role is \textit{ObjRole}
6: \hspace{2em} Extract dependency path expression \(dpath_b\) in \(\textit{DPATH}_B\) from \textit{DName} using \(\lambda_O\) function
7: \hspace{2em} Determine vertices by tracing base provenance data \(\textit{PD}_B\) through the paths expressed in \(dpath_b\) that start from the object \(o\) using \(\delta_O\) function
8: \hspace{1em} \textbf{end for}
9: \hspace{1em} Determine the truth value by evaluating the results of all the extracted path rules
10: \textbf{end for}
11: \(AVal \leftarrow \) a combined result based on conjunctive or disjunctive connectives between rules
PBAC\textsubscript{B} Model Policy Grammar

Policy ::= “allow” < Req > “⇒” < UARules > “∧” < AVRules > | “true”

Req ::= (“” < ActUser > “,” < ActType > “,” < ObjRoles > “)”

ObjRoles ::= < ObjRole > | < ObjRole > “,” < ObjRoles >

UARules ::= < UARule > | (“” < UARules > “)” |

AVRules ::= < AVRule > | (“” < AVRules > “)” |

Connect ::= ∨/∧

UARule ::= < ActUser > < oper1 > < PathRule >

AVRule ::= “|” < PathRule > “|” < oper2 > < Number > |

PathRule ::= (“” < ObjRole > “,” < DName > “)”

oper1 ::= “∈”|“∉”
oper2 ::= “=”|“≠”|“≥”|“≤”|“<”|“>”
oper3 ::= “=”|“≠”|“⊆”

DName ::= dn\textsubscript{1}|dn\textsubscript{2}|...|dn\textsubscript{n}

Number ::= [0 – 9]+

ActUser ::= au

ActType ::= at\textsubscript{1}|at\textsubscript{2}|...|at\textsubscript{m}

ObjRole ::= o\textsubscript{role\textsubscript{1}}|o\textsubscript{role\textsubscript{2}}|...|o\textsubscript{role\textsubscript{k}}