On the Relationship between Finite Domain ABAM and PreUCONA

Asma Alshehri and Ravi Sandhu
Department of Computer Science

10th International Conference on Network and System Security (NSS)
September 28-30, 2016
Outline

- Introduction
- ABAM model
- PreUCON_A model
- Expressing PreUCONA IN ABAM
- Reducing ABAM to PreUCON_A
- Right-Less ABAM with Two Parameters (RL-ABAM2)
- Expressing RL-ABAM2 in PreUCON_A
- Conclusion
Introduction

➢ **ABAM:**
  - HRU + Attributes
  - Test for and modify attribute values
  - Set of attributes is finite

➢ **PreUCON**
  - Sub-model of UCON
  - Test for and modify attribute values prior access
  - Set of attributes is finite
Components:

- Subjects ($S$) and Objects ($O$)
- Attributes and Attribute Tuples
- Rights ($R$) and Access Matrix
- Attribute Predicates ($P$)
- Primitive Operations
- Commands

ABAM access matrix [*]

**Attribute Based Access Matrix (ABAM)**

- **ABAM Commands:**
  - Parameters (entities with possibly new attribute values)
  - Conditions
  - A sequence of primitive operations.

**Command**

\[ \alpha_i (X_1 : ATT(X_1), X_2 : ATT(X_2), \ldots, X_k : ATT(X_k)) \]

If \( r_1 \in [X_{s_1}, X_{o_1}] \land r_2 \in [X_{s_2}, X_{o_2}] \land \ldots \land r_m \in [X_{s_m}, X_{o_m}] \land P_1 \land P_2 \land \ldots \land P_n \)

then \( op_1, op_2, \ldots, op_l \)

end

<table>
<thead>
<tr>
<th>( s_1 : ATT(s_1) )</th>
<th>( s_2 : ATT(s_2) )</th>
<th>( o_1 : ATT(o_1) )</th>
<th>( o_2 : ATT(o_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( {\text{parent}} )</td>
<td>( {\text{read, write}} )</td>
<td>( {\text{read}} )</td>
<td></td>
</tr>
</tbody>
</table>

ABAM access matrix [*]

Three components:

- An object schema \( OS_\Lambda \)
  \[ OS_\Lambda = (a_1 : \Omega_1, a_2 : \Omega_2, \ldots, a_n : \Omega_n) \]

- A set of finite usage rights \( UR = \{r_1, r_2, \ldots, r_m\} \)

- A set of usage control commands \( \{UC_1, UC_2, \ldots, UC_m\} \)
### PreUCON\(A\) Commands:

<table>
<thead>
<tr>
<th>Non-Creating Commands</th>
<th>Creating Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command-Name(_r) (s, o)</strong></td>
<td><strong>Command-Name(_r) (s)</strong></td>
</tr>
<tr>
<td><strong>PreCondition:</strong> ( f_b(s, o) \rightarrow {yes, no} )</td>
<td><strong>PreCondition:</strong> ( f_b(s) \rightarrow {yes, no} )</td>
</tr>
<tr>
<td><strong>PreUpdate:</strong></td>
<td><strong>PreUpdate:</strong></td>
</tr>
<tr>
<td>( s.a_{i_1} := f_1, a_{i_1}(s, o) )</td>
<td>( s.a_{i_1} := f_1, a_{i_1}(s) )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( s.a_{i_p} := f_1, a_{i_p}(s, o) )</td>
<td>( s.a_{i_p} := f_1, a_{i_p}(s) )</td>
</tr>
<tr>
<td>( o.a_{j_1} := f_2, a_{j_1}(s, o) )</td>
<td>( o.a_1 := f_2, a_1(s) )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( o.a_{j_q} := f_2, a_{j_q}(s, o) )</td>
<td>( o.a_n := f_2, a_n(s) )</td>
</tr>
</tbody>
</table>

**Table 1.** PreUCON\(A\) commands
Expressing PreUCONA IN ABAM

Challenges:
- Condition Part in PreUCONA permits arbitrary computable Boolean functions
- Condition Part in ABAM only permits propositional logic formulas

Finite domain results in:
- PreUCONA can be computed for all possible attribute values of $s$ and $o$
- The results can be “compiled” into multiple ABAM commands.
Expressing PreUCON\textsubscript{A} IN ABAM

**Example:** Let the object schema $OS_A = \{a_1 : \{1,2\}, a_2 : \{2,3\}, a_3 : \{1,2,3\}\}$ and usage rights $UR = \{\text{update}\}$.

The initial values for $s$ and $o$ attributes are $[1,2,3]$ and $[2,3,1]$ respectively for $[a_1, a_2, a_3]$.

The PreUCON\textsubscript{A} update command is in the left side, and the table shows the possible ABAM commands:

<table>
<thead>
<tr>
<th>Updating to value 1</th>
<th>Updating to value 2</th>
<th>Updating to value 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command</strong> $update(s : ATT(s), o : ATT(o))$</td>
<td><strong>Command</strong> $update(s : ATT(s), o : ATT(o))$</td>
<td><strong>Command</strong> $update(s : ATT(s), o : ATT(o))$</td>
</tr>
<tr>
<td>if $s.a_1 \leq 2 \lor o.a_2 \leq 3 \land (s.a_3 = 1 \lor o.a_3 = 1)$</td>
<td>if $s.a_1 \leq 2 \lor o.a_2 \leq 3 \land ((s.a_3 = 1 \land o.a_3 = 2) \lor (s.a_3 = 2 \land o.a_3 = 1))$</td>
<td>if $s.a_1 \leq 2 \lor o.a_2 \leq 3 \land ((s.a_3 = 1 \land o.a_3 = 3) \lor (s.a_3 = 2 \land o.a_3 = 3) \lor (s.a_3 = 3 \land o.a_3 = 3))$</td>
</tr>
<tr>
<td>update attribute $o.a_3 = 1$</td>
<td>update attribute $o.a_3 = 2$</td>
<td>update attribute $o.a_3 = 3$</td>
</tr>
<tr>
<td>enter $update$ into $[s,o]$; delete $update$ from $[s,o]$; end</td>
<td>enter $update$ into $[s,o]$; delete $update$ from $[s,o]$; end</td>
<td>enter $update$ into $[s,o]$; delete $update$ from $[s,o]$; end</td>
</tr>
</tbody>
</table>

Table 2. Possible ABAM commands
Reducing ABAM to PreUCONA

Challenges of Reducing ABAM to PreUCONA

- PreUCONA command has the ability to grant a non-persistent right in each command.
- ABAM command has the power of granting one or more rights to the actor, maintaining the given rights in the corresponding cell of the actor, and permitting two or more parameters (more targets) in each command.
- Unrestricted use of rights in ABAM will result in undecidable safety whereas PreUCONA has decidable safety.

Reducing ABAM to PreUCONA is not possible.

Right-Less ABAM with Two Parameters (RL-ABAM2) is a restricted form of ABAM that can be reduced to PreUCONA.
RL-ABAM2 command is limited in terms:
- Number of parameters
- The if statement section
- The existence of rights

RL-ABAM2 command is defined as follows:

\[ \text{Command } \alpha_i(X_1 : ATT(X_1), X_2 : ATT(X_2)) \]

if \( p_1 \wedge p_2 \wedge \ldots p_n \)
then
\( op_1; op_2; \ldots; op_l \)
enter \( r_1 \) into \([X_1, X_2]\);
delete \( r_1 \) from \([X_1, X_2]\);
...
enter \( r_k \) into \([X_1, X_2]\);
delete \( r_k \) from \([X_1, X_2]\)
end
Expressing RL-ABAM2 in PreUCON\textsubscript{A}

Given an RL-ABAM2 schema with the following components: objects $O_{RL-ABAM2}$, subjects $S_{RL-ABAM2}$, access rights $R_{RL-ABAM2} = \{r_1, \ldots, r_k\}$, attributes tuple $ATT(o_i) = < a_1 = v_1, \ldots, a_n = v_n >$, where $o_i \in O_{RL-ABAM2}$, and a list of all attributes which are linked with their domains $G - V_{RL-ABAM2} = [a_1:V(a_1), \ldots, a_i:V(a_i), \ldots, a_n:V(a_n)]$, each RL-ABAM2 commands will have the following structure:

**Command** $\alpha_i (s_i : ATT(s_i), o_j : ATT(o_j))$

if $p_1 \land p_2 \land \ldots p_n$
then
create object $X2 : ATT(X2)$;
update attribute $s_i.a_k = v'_i$;
update attribute $o_i.a_s = v'_j$;
enter $r_i$ into $[s_i, o_j]$;
delete $r_i$ from $[s_i, o_j]$;
end
Expressing RL-ABAM2 in PreUCON\(_A\)

The corresponding PreUCON\(_A\) components of the RL-ABAM2 schema are extended as follows:

- Entity in PreUCON\(_A\) are objects \(O_{Pre\_UCON\(_A\)}\)
- \(O_{Pre\_UCON\(_A\)} = O_{RL\_ABAM2} \cup O_{lock}\)
- \(S_{Pre\_UCON\(_A\)} = S_{RL\_ABAM2}\)
- \(UR_{Pre\_UCON\(_A\)} = Command - R_{RL\_ABAM2}\)
- Auxiliary - \(OS_\Delta = [\text{lock}:V(\text{lock}), \text{type}:V(\text{type}), \text{R_to_select}:V(\text{R_to_select}), \text{position}:V(\text{position})]\)
- \(OS_\Delta = G - V_{RL\_ABAM2} \cup \text{Auxiliary} - OS_\Delta\)

The domain for each of these additional attributes is as follows: \(V(\text{lock}) = \{0, 1\}\), \(V(\text{type}) = \{\text{ordinary, lock}\}\), \(V(\text{R_to_select}) = UR_{Pre\_UCON\(_A\)}\), and \(V(\text{position}) = \{1,2\}\). The initial values for the proposed attributes are set as follows: For all \(o \in O_{RL\_ABAM2}\): \(o.\text{type} = \text{ordinary}\), \(o.\text{lock} = 0\), \(o.\text{position} = \phi\), and \(o.\text{R_to_select} = \phi\). For \(O_{lock}\): \(O_{lock}.\text{type} = \text{lock}\), \(O_{lock}.\text{lock} = 1\), \(O_{lock}.\text{position} = \phi\), \(O_{lock}.\text{R_to_select} = \phi\).
Expressing RL-ABAM2 in PreUCON_A

To apply a RL-ABAM2 command in PreUCON_A commands, a sequence of steps is introduced as follows:

1- Give a lock to the first parameter of the RL-ABAM2 command

```
Command get_lock (s_i : ATT(s_i), O_lock : ATT(O_lock))
if s_i.type = ordinary ∧ O_lock.type = lock ∧ s_i.lock = 0 ∧ O_lock.lock = 1
then
  update attribute s_i.lock = 1;
  update attribute O_lock.lock = 0;
  update attribute s_i.position = 1;
  update attribute s_i.R_to_select = UR_{Pre_UCON_A}
end
```

2- Decide the second parameter of the RL-ABAM2 command

```
Command pick_target(s_i : ATT(s_i), o_j : ATT(o_j))
if s_i.type = ordinary ∧ s_i.lock = 1 ∧ o_j.lock = ordinary ∧ s_i.position = 1 ∧ o_j.position = φ
then
  update attribute o_j.position = 2;
end
```
3- Implement a sequence of PreUCON\(\alpha\) commands (depend on the number of the operation over rights in the body of an RL-ABAM2 command)

**Command—\(r_1(s_i,o_j)\)**

**PreCondition:**

\[
\begin{align*}
&f_k(s_i,o_j) \land \\
&s_i.R\_to\_select = UR_{Pre\_UCON\alpha} \land s_i.lock = 1 \\
&\land s_i.position = 1 \land o_j.position = 2;
\end{align*}
\]

**PreUpdate:**

\[
\begin{align*}
&create o; \\
&s_i.a_k = v_{ij}^s; \\
&o_j.a_s = v_{ij}^o; \\
&s_i.R\_to\_select = UR_{Pre\_UCON\alpha} - \{r_1\}
\end{align*}
\]

**Command—\(r_2(s_i,o_j)\)**

**PreCondition:**

\[
\begin{align*}
&s_i.R\_to\_select = UR_{Pre\_UCON\alpha} - \{r_1\} \land s_i.lock = 1 \\
&\land s_i.position = 1 \land o_j.position = 2;
\end{align*}
\]

**PreUpdate:**

\[
\begin{align*}
&s_i.R\_to\_select = UR_{Pre\_UCON\alpha} - \{r_1,r_2\} \\
&\ldots \\
&\ldots
\end{align*}
\]

**Command—\(r_k(s_i,o_j)\)**

**PreCondition:**

\[
\begin{align*}
&s_i.R\_to\_select = \{r_k\} \land s_i.lock = 1 \land s_i.position = 1 \\
&\land o_j.position = 2;
\end{align*}
\]

**PreUpdate:**

\[
\begin{align*}
&o_j.position = \phi \\
&s_i.R\_to\_select = \phi
\end{align*}
\]
4- Release the lock from the first parameter (actor) of the RL-ABAM2 command.

**Command** `release_lock (s_i : ATT(s_i), O_lock : ATT(O_lock))`

if $s_i.type = \text{ordinary} \land O\_lock.type = \text{lock} \land s_i.lock = 1 \land O\_lock.lock = 0 \land s_i.R\_to\_select = \phi$

then

update attribute $s_i.lock = 0$;
update attribute $O\_lock.lock = 1$;
update attribute $s_i.position = \phi$; \textbf{end}
The study of ABAM indicates that a safe application of access rights could be based on the following principles:

1- Do not use rights in the if part of commands
2- Some rights could be left behind by commands so their next use is more efficient
3- There is a meaningful place for access matrix rights
Thank you!!
Any Questions??