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

Introduction

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

- **PreUCON$_A$**
  - Sub-model of UCON
  - Test for and modify attribute values prior access
  - Set of attributes is finite

Attribute Based Access Matrix (ABAM)

➢ Components:

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

	s_1: $ATT(s_1)$	s_2: $ATT(s_2)$	o_1: $ATT(o_1)$	o_2: $ATT(o_2)$
s_1: $ATT(s_1)$		{parent}	{read, write}	
s_2: $ATT(s_2)$			{read}	

ABAM access matrix [*]

* Zhang, X., Li, Y., Nalla, D.: An attribute-based access matrix model. In: the 2005 ACM Symposium on Applied Computing, pp. 359-363 (2005).

Attribute Based Access Matrix (ABAM)

➢ ABAM Commands:

- ➢ Parameters (entities with possibly new attribute values)
- ➢ Conditions
- ➢ A sequence of primitive operations.

Command $\alpha_i(X_1 : ATT(X_1), X_2 : ATT(X_2), .., X_k : ATT(X_k))$
If $r_1 \in [X_{s1}, X_{o1}] \wedge r_2 \in [X_{s2}, X_{o2}] \wedge .... r_m \in [X_{sm}, X_{om}] \wedge P_1 \wedge P_2 \wedge ... P_n$
then $op_1; op_2; ....; op_l$
end

	$s_1: ATT(s_1)$	$s_2: ATT(s_2)$	$o_1: ATT(o_1)$	$o_2: ATT(o_2)$
$s_1: ATT(s_1)$		{parent}	{read, write}	
$s_2: ATT(s_2)$			{read}	

ABAM access matrix [*]

* Zhang, X., Li, Y., Nalla, D.: An attribute-based access matrix model. In: the 2005 ACM Symposium on Applied Computing, pp. 359-363 (2005).

PreUCONA

- **Three components:**
  - An object schema OS_Δ
    - $OS_\Delta = (a_1 : \Omega_1, a_2 : \Omega_2, \ldots, a_n : \Omega_n)$
  - A set of finite usage rights $UR = \{r_1, r_2, \ldots, r_m\}$
  - A set of usage control commands $\{UC_1, UC_2, \ldots, UC_m\}$

➢ PreUCONA Commands:

Non-Creating Commands	Creating Commands
Command-Name$_r$(s, o) **PreCondition:** $f_b(s,o) \rightarrow \{yes, no\}$; **PreUpdate:** $s.a_{i_1} := f_1, a_{i1}(s,o)$; ... $s.a_{i_p} := f_1, a_{ip}(s,o)$; $o.a_{j_1} := f_2, a_{j1}(s,o)$; ... $o.a_{j_q} := f_2, a_{jq}(s,o)$;	**Command-Name$_r$(s)** **PreCondition:** $f_b(s) \rightarrow \{yes, no\}$; **PreUpdate:** create o; $s.a_{i_1} := f_1, a_{i1}(s)$; ... $s.a_{i_p} := f_1, a_{ip}(s)$; $o.a_1 := f_2, a_1(s)$; ... $o.a_n := f_2, a_n(s)$;

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 PreUCONA IN ABAM

➢ **Example:** Let the object schema $OS_{\Delta} = [a_1 : \{1,2\}_1, a_2 : \{2,3\}, .a_3 : \{1,2,3\}]$ and usage rights UR = {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 PreUCONA update command is in the left side, and the table shows the possible ABAM commands:

$\text{Command}_{update}(s, o)$
PreCondition: $s.a_1 \leq 2 \vee o.a_2 \leq 3$
PreUpdate: $o.a_3 := max(s.a_3, o.a_3);$

Updating to value 1	Updating to value 2	Updating to value 3
Command *update*($s : ATT(s), o : ATT(o)$) **if** $s.a_1 \leq 2 \vee o.a_2 \leq 3 \wedge (s.a_3 = 1 \vee o.a_3 = 1)$ update attribute $o.a_3 = 1$ enter *update* into $[s, o]$; delete *update* from $[s, o]$; **end**	**Command** *update*($s : ATT(s), o : ATT(o)$) **if** $s.a_1 \leq 2 \vee o.a_2 \leq 3 \wedge ((s.a_3 = 1 \wedge o.a_3 = 2) \vee (s.a_3 = 2 \wedge o.a_3 = 1)($ update attribute $o.a_3 = 2$ enter *update* into $[s, o]$; delete *update* from $[s, o]$; **end**	**Command** *update*($s : ATT(s), o : ATT(o)$) **if** $s.a_1 \leq 2 \vee o.a_2 \leq 3 \wedge ((s.a_3 = 1 \wedge o.a_3 = 3) \vee (s.a_3 = 2 \wedge o.a_3 = 3) \vee (s.a_3 = 3 \wedge o.a_3 = 1) \vee (s.a_3 = 3 \wedge o.a_3 = 2) \vee (s.a_3 = 3 \wedge o.a_3 = 3))$ update attribute $o.a_3 = 3$ enter *update* into $[s, o]$; delete *update* from $[s, o]$; **end**

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**

Right-Less ABAM with Two Parameters (RL-ABAM2)

- RL-ABAM2 command is limited in terms:
  - Number of parameters
  - The if statement section
  - The existence of rights
- RL-ABAM2 command is is defined as follows:

Command $\alpha_i(X_1 : ATT(X_1), X_2 : ATT(X_2))$
if $p_1 \wedge p_2 \wedge ...p_n$
then
$op_1; op_2; ...; op_l$;
enter r_1 into $[X_1, X2]$;
delete r_1 from $[X_1, X2]$;
...
enter r_k into $[X_1, X2]$;
delete r_k from $[X_1, X2]$
end

Expressing RL-ABAM2 in PreUCONA

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

Command α_i $(s_i : ATT(s_i), o_j : ATT(o_j))$
if $p_1 \wedge p_2 \wedge ...p_n$
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 PreUCONA

- The corresponding PreUCONA components of the RL-ABAM2 schema are extended as follows:

  - Entity in PreUCON$_A$ are objects $O_{Pre\_UCON_A}$
  - $O_{Pre\_UCON_A} = O_{RL-ABAM2} \cup O_{lock}$
  - $S_{Pre\_UCON_A} = S_{RL-ABAM2}$
  - $UR_{Pre\_UCON_A} = Command - R_{RL-ABAM2}$
  - $Auxiliary-OS_\Delta = [lock{:}V(lock), type{:}V(type), R\_to\_select{:}V(R\_to\_select), position{:}V(position)]$
  - $OS_\Delta = G - V_{\{RL-ABAM2\}} \cup Auxiliary - OS_\Delta$

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

Expressing RL-ABAM2 in PreUCONA

To apply a RL-ABAM2 command in PreUCONA 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 $s_i.type = ordinary \wedge O\_lock.type = lock \wedge s_i.lock = 0 \wedge O\_lock.lock = 1$
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 $s_i.type = ordinary \wedge s_i.lock = 1 \wedge o_j.lock = ordinary \wedge s_i.position = 1 \wedge o_j.position = \phi$
then
update attribute $o_j.position = 2$;
end

Expressing RL-ABAM2 in PreUCONA

3- Implement a sequence of PreUCONA commands (depend on the number of the operation over rights in the body of an RL-ABAM2 command)

$\mathbf{Command} - r_1(s_i, o_j)$

PreCondition: $f_b(s_i, o_j) \wedge$
$s_i.R\_to\_select = UR_{Pre\_UCON_A} \wedge s_i.lock = 1$
$\wedge\ s_i.position = 1 \wedge o_j.position = 2;$

PreUpdate:
$create\ o;$
$s_i.a_k = v_i';$
$o_j.a_s = v_j';$
$s_i.R\_to\_select = UR_{Pre\_UCON_A} - \{r_1\}$

$\mathbf{Command} - r_2(s_i, o_j)$

PreCondition: $s_i.R\_to\_select = UR_{Pre\_UCON_A} - \{r_1\} \wedge s_i.lock = 1$
$\wedge\ s_i.position = 1 \wedge o_j.position = 2;$

PreUpdate:
$s_i.R\_to\_select = UR_{Pre\_UCON_A} - \{r_1, r_2\}$
....
....

$\mathbf{Command} - r_k(s_i, o_j)$

PreCondition: $s_i.R\_to\_select = \{r_k\} \wedge s_i.lock = 1 \wedge s_i.position = 1$
$\wedge\ o_j.position = 2;$

PreUpdate:
$o_j.position = \phi$
$s_i.R\_to\_select = \phi$

Expressing RL-ABAM2 in PreUCONA

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 $s_i.type = ordinary \wedge O\_lock.type = lock \wedge s_i.lock = 1 \wedge O\_lock.lock = 0 \wedge s_i.R\_to\_select = \phi$
then
update attribute $s_i.lock = 0$;
update attribute $O\_lock.lock = 1$;
update attribute $s_i.position = \phi$; **end**

Conclusion

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??