

Economical Use of Formal Methods

Yi Mao

atsec information security corporation

yi@atsec.com

Agenda

- Formal method requirements from CC
- Use of formal methods in CC evaluations
- Reasons for economical use of formal methods
- Example: Prove the sufficiency and conditional efficiency of a Hierarchical Role-Based Access Control designed for a file system on some smart cards
- Conclusions

Formal Method Requirements

Common Criteria (CC) V3.1 requires using formal methods in the following families:

- ADV_SPM.1 (Security Policy Modelling)
- ADV_FSP.6 (Functional Specification)
- ADV_TDS.6 (TOE Design)

Formal Methods and CC evaluations

- Some CC evaluations used formal methods:
 - Tenix Interactive Link Data Diode Device
 - AAMP7G Microprocessor
 - Java Card Platform
- Some tools used by formal methods:
 - Isabelle
 - ACL2
 - Caduceus

Pros and Cons

- **Benefits**

- Correctness of product designs and implementations is mechanically checked
- High level of assurance is gained

- **Drawbacks**

- High cost and complexity of the evaluation effort
- Steep learning curve
- Often requires Ph.D. in the fields of theorem prover/model checking

- **Tendency**

- Formal methods = using theorem prover and/or model checking tools
- Too difficult and costly to use formal methods
- Avoid using formal methods whenever possible

A Proposal

What to change

- Formal methods \neq using theorem prover and/or model checking tools
- Constructing short human-readable proofs is also one way of using formal methods

Benefits

- Gain high confidence in correctness
- Avoid the overhead of mastering any theorem prover or model checking tools
- Reduce cost and effort
- Encourage using formal methods whenever possible

Implications of the Proposal

NOT intended to discourage

- use of proof assistance tools
- development of proof assistance tools

Intended to encourage

- use of formal methods, including tool-aided as well as non-tool-aided
- accurate characterization of product design through cost-controlled formalization

Rationale

- All tools theoretically rely on underlying logic for proof construction. Therefore, proofs can be constructed in proper logic without using tools.

Exciting Time

Let us see an example!

Need for Access Control on Smart Card

- Multiple applications on one card
 - Smart cards as computers, as well as Internet nodes
 - Communication, information retrieval, e-commerce, etc.
- Multiple users on one card
 - Card vendors
 - Card issuers
 - Service providers
 - Card holders
 - Administrators
- Privacy and security concerns
 - Personal data
 - Application-specific data
 - Secret keys

Main Questions for an Access Control Mechanism (ACM)

- Is it efficient?
 - Memory usage
 - Execution time
- Is it sufficient?
 - Can ANY security requirements be represented?

ACM on Unix/Linux

- Three classes of users
 - Owner, Group, Other
- Three types of permissions
 - Read (r), Write (w), Execute (x)
- An example

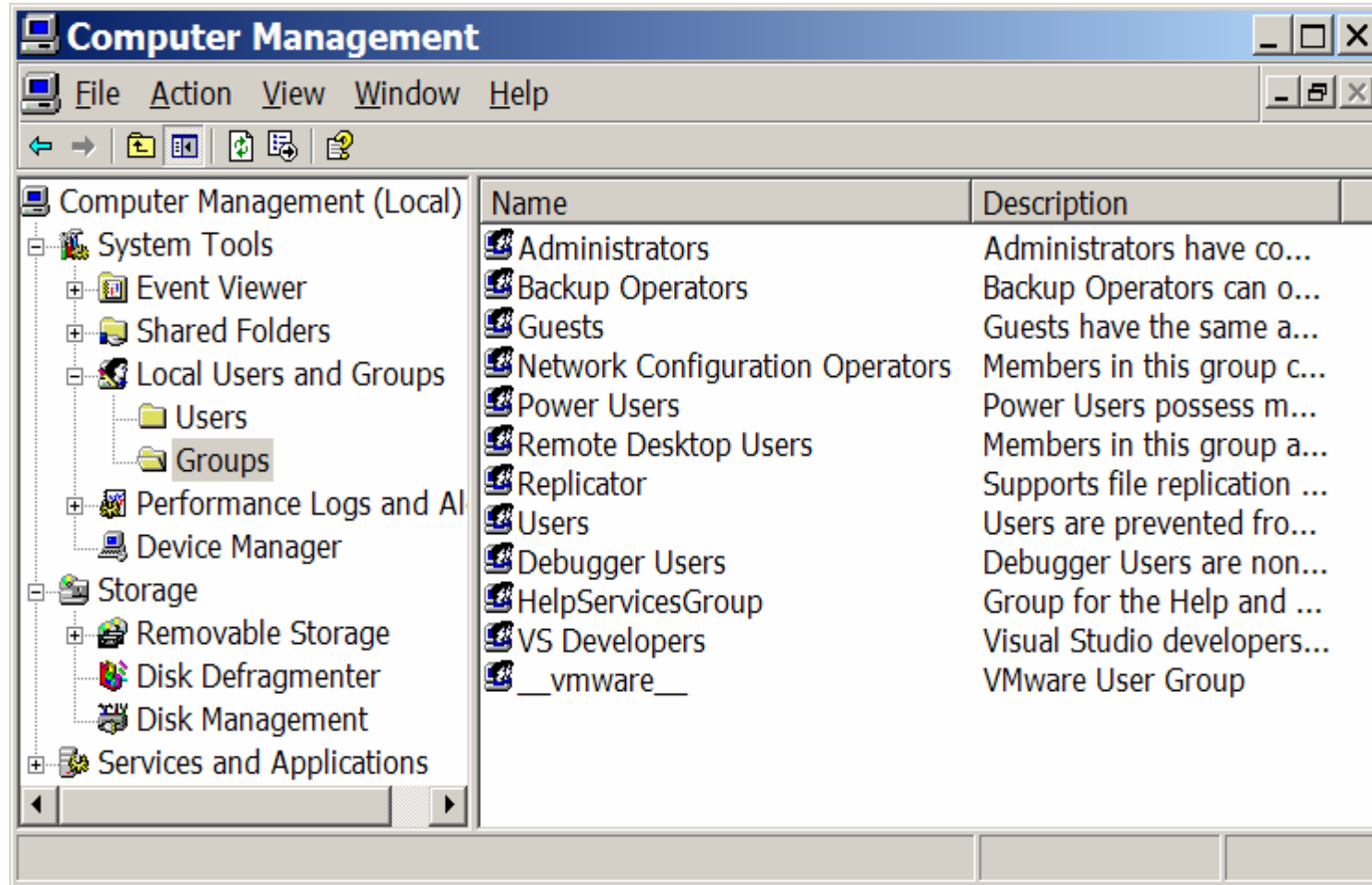
```
Jsmart% ls -l dir1  
  
drwxr-xr-x    9/21/2005 4:01pm      256    Dir1  
-rwxr-xr--    9/25/2005 10:22am       29     f1  
-rwxr-xr--    9/25/2005 10:23am       58     f2
```

- Limitations
 - Fixed number of groups in the access control list
 - A user can join only one group in the access control list
 - Fixed number of permission types

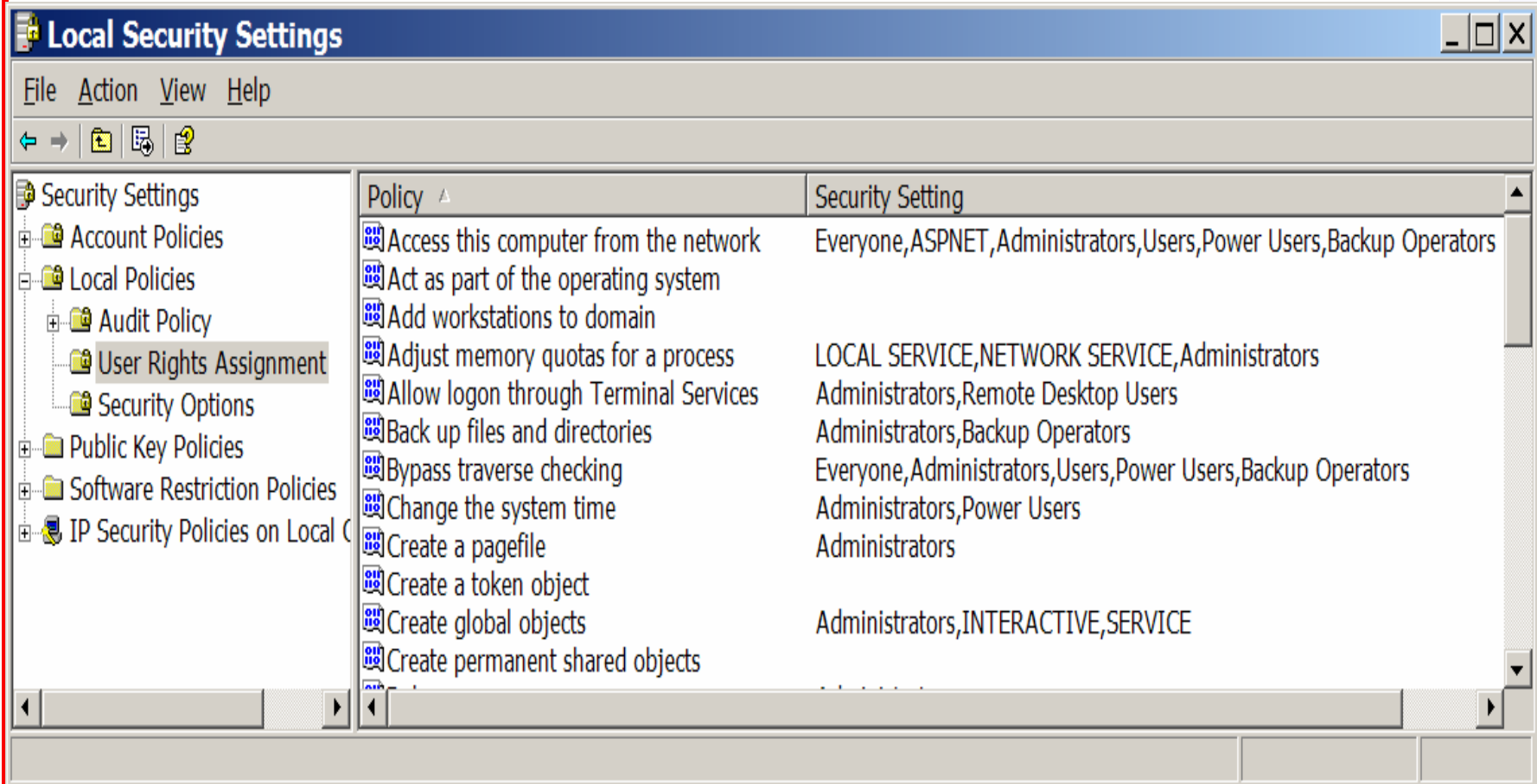
ACM on Windows (1)

- Three categories of entities
 - Users
 - Groups
 - Tasks (Applications)
- Two kinds of assignments
 - Assign users to groups
 - Assign groups to tasks
- Limitations
 - Flat organization of groups
 - Redundant assignment of one task to many groups

ACM on Windows (2): User Role Assignment



ACM on Windows (3): Role Task Assignment



Constraints of ACM on Smart Cards

Maximum flexibility for access control with
minimal resources

- Limited space to store AC policies
 - A flat organization of ACM as in Windows is a luxury for smart cards
- Flexible number of roles in AC policies
 - A fixed number of groups in an ACL as in Unix is too restrictive for smart cards

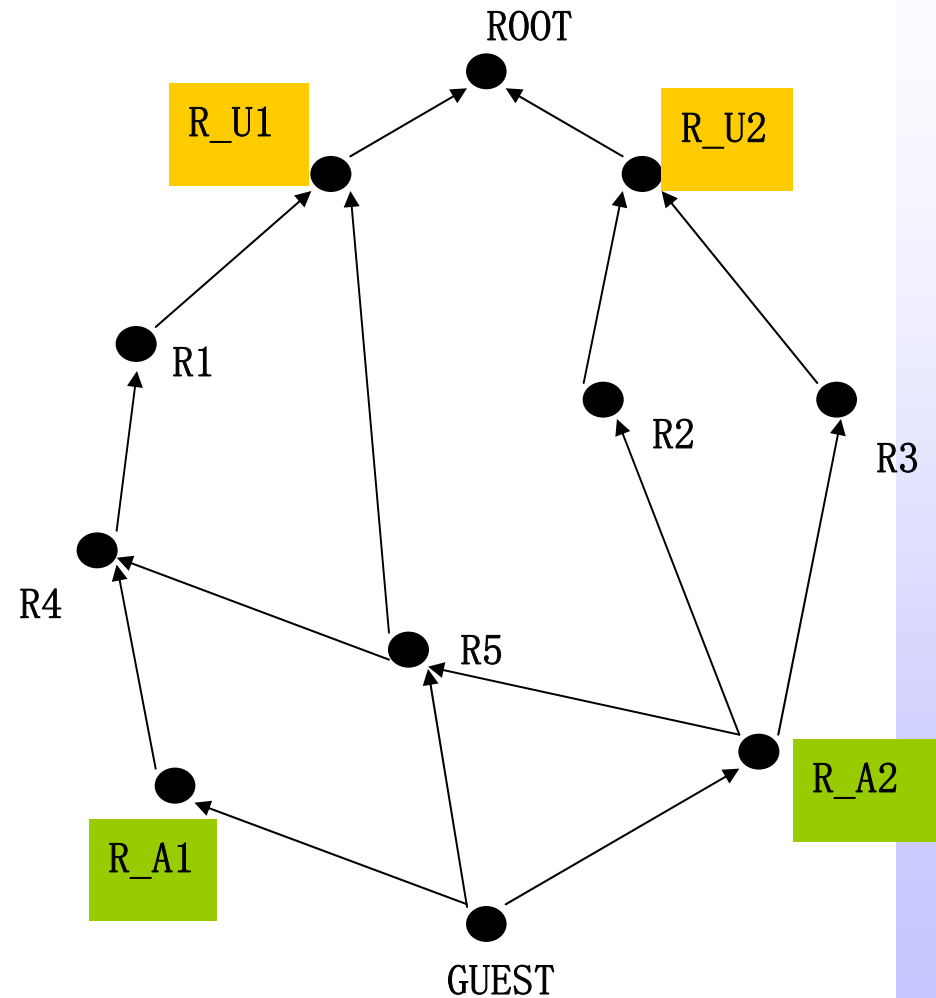
Gemalto designed a Hierarchical Role-Based Access Control (HRBAC) system

Key Concepts of HRBAC

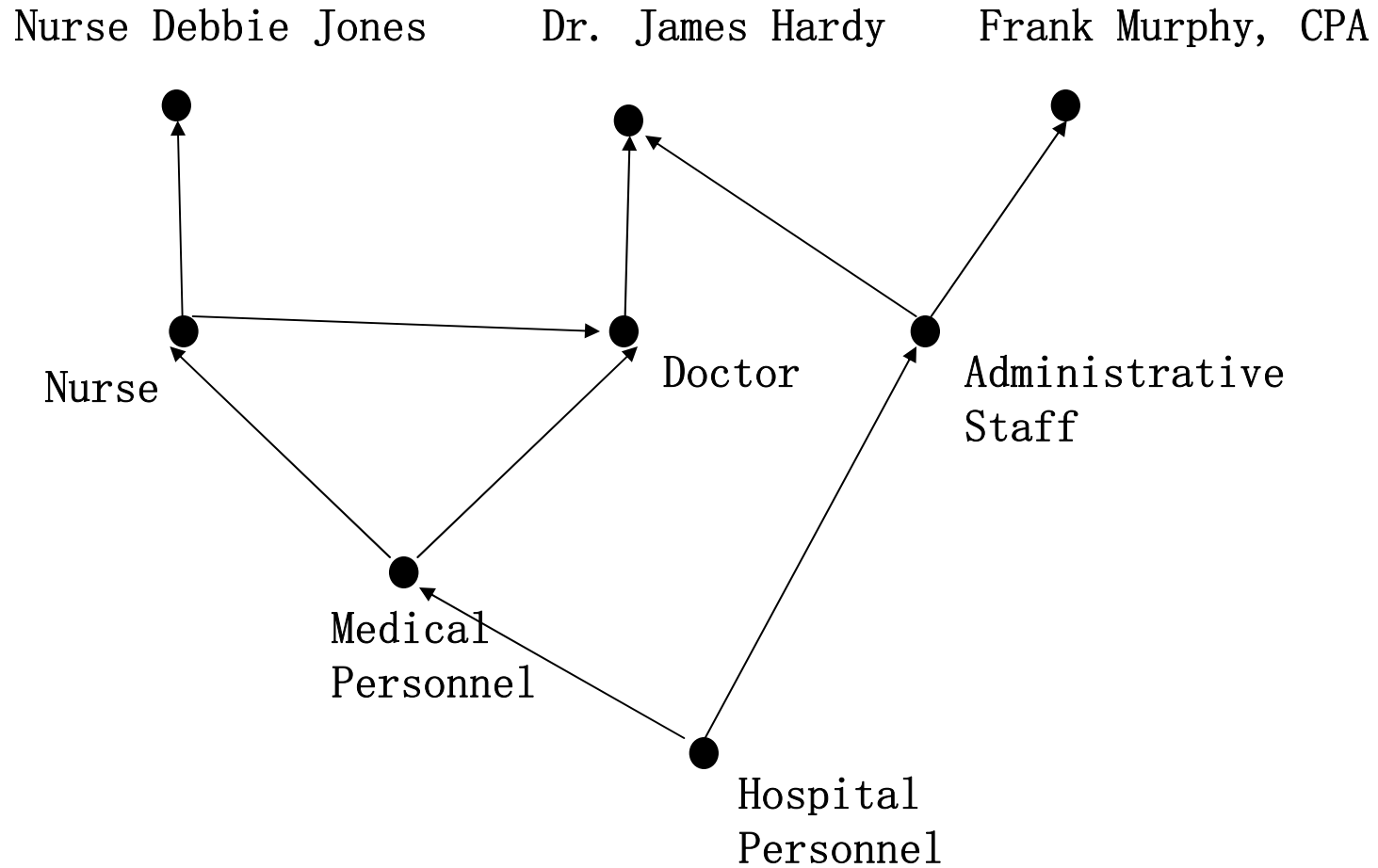
- Role: Unification of users, groups, tasks, applications in a single concept
- Hierarchy of roles with inheritance of permissions
 - Allows sparse specification of permissions; only need to specify incremental permissions with respect to hierarchy
- Resources protected by role-based access policies
 - Protect resources (e.g. files, devices, ports, etc.) with access control lists (ACLs)

A Hierarchy of Roles

- Acyclic graph structure
 - Nodes are roles
 - Arrows go from roles with lower ranks to higher ones
 - Access rights are inheritable along the arrows
 - ROOT – the most powerful role
 - GUEST – has the least privilege



An Example of Role Hierarchy



The Challenge

- Is HRBAC sufficient?
Can it represent ANY security requirements?
- How do you know it is sufficient?
Can you prove it?

A Formal Model of HRBAC

An HRBAC model can be formally written as a six-tuple $\langle R, >, OP, F, Perm, Forb \rangle$, where

- R is a set of roles
- $>$ is a partial ordering among roles
- OP is a set of operations
- F is a set of files (or other type of resources)
- $Perm$ is a set of all permitted accesses (e.g. $\langle r, o, f \rangle \in Perm$ means that role r has access to file f with operation o)
- $Forb$ is a set of all forbidden accesses (e.g. $\langle r, o, f \rangle \in Forb$ means that role r has no access to file f with operation o)

Granted Accesses

With an HRBAC model $\langle R, >, OP, F, Perm, Forb \rangle$

- *Perm* and *Forb* are explicitly given access policies
- Implicitly granted accesses due to role hierarchy
- Let *GA* be the set of all granted accesses by the HRBAC model
- Intuitively, *GA* should contain all triples that are either explicitly permitted (i.e. in *Perm*) or can be inherited from a triple in *Perm* but not explicitly forbidden (i.e. in *Forb*)

Modeling Access Requirements

- Intuitively, access requirements are a bunch of specifications defining who can do what on which resources (e.g. file).
- Normally, a hierarchy of roles is determined from the organizational chart. Assume the existence of such a hierarchy.
- Formally, any access requirements can be modeled as a quintuple $\langle R, >, OP, F, AP \rangle$, where AP stands for a set of triples $\langle r, o, f \rangle$ that represent expected access control policies.

A First Order Language

- Individual variables:
a, b, c, ... z
- Unary predicates: R, OP, F
(Their intended interpretations are the sets *R*, *OP* and *F*, respectively)
- Binary predicate: >
(Its intended interpretation is the hierarchical relationship >)
- Ternary predicates: Perm, Forb, GA, AP
(Their intended interpretations are the sets of triples *Perm*, *Forb*, *GA* and *AP*, respectively)
- Usual logical connectives and quantifiers

$\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists$

Axioms and Inference Rules

- Axiom for granted accesses of an HRBAC model

$$(A0) \quad \forall r \forall o \forall f (R(r) \wedge OP(o) \wedge F(f) \rightarrow (GA(r, o, f) \leftrightarrow Perm(r, o, f) \vee (\neg Forb(r, o, f) \wedge \exists r'(R(r') \wedge (r > r') \wedge Perm(r', o, f))))))$$

- Axioms for the transformation between AP and Perm and Forb

$$(A1) \quad \forall r \forall o \forall f (R(r) \wedge OP(o) \wedge F(f) \rightarrow Perm(r, o, f) \leftrightarrow (AP(r, o, f) \wedge \neg \exists r'(R(r') \wedge (r > r') \wedge AP(r', o, f))))$$

$$(A2) \quad \forall r \forall o \forall f (R(r) \wedge OP(o) \wedge F(f) \rightarrow Forb(r, o, f) \leftrightarrow (\neg AP(r, o, f) \wedge \exists r'(R(r') \wedge (r > r') \wedge AP(r', o, f))))$$

Axioms and Inference Rules (Cont.)

- Axiom for the existence of minimal role in any chain of access control policies

$$(A3) \quad \forall r \forall o \forall f (R(r) \wedge OP(o) \wedge F(f) \wedge AP(r, o, f) \rightarrow \exists r'' (R(r'') \wedge (r=r'' \vee r > r'') \wedge AP(r'', o, f) \wedge \neg \exists r' (R(r') \wedge (r'' > r') \wedge AP(r', o, f)))$$

- Axioms for “>” being irreflexive, asymmetric and transitive
- Usual First Order Logic (FOL) axioms and inference rules

The Theorem of Sufficiency of HRBAC

Theorem Given any access requirements, there is an HRBAC model whose granted accesses exactly matches the expected access policies.

Formally, for any access requirements given as $\langle R, >, OP, F, AP \rangle$, there is an HRBAC $\langle R, >, OP, F, Perm, Forb \rangle$ model such that the *GA* of HRBAC is the same as *AP*. That is to prove that

$$\forall r \forall o \forall f (R(r) \wedge OP(o) \wedge F(f) \rightarrow (GA(r, o, f) \leftrightarrow AP(r, o, f)))$$

A Proof of Theorem in the FOL

(1)	$\forall r \forall o \forall f (R(r) \wedge OP(o) \wedge F(f) \rightarrow ((\neg AP(r, o, f) \wedge \exists r' (R(r') \wedge (r > r') \wedge AP(r', a, f))) \leftrightarrow Forb(r, o, f)))$	(A2)
(2)	$R(r) \wedge OP(o) \wedge F(f) \rightarrow (Forb(r, o, f) \leftrightarrow (\neg AP(r, o, f) \wedge \exists r' (R(r') \wedge (r > r') \wedge AP(r', a, f))))$	(1), \forall -
(3)	$R(r) \wedge OP(o) \wedge F(f)$	Assumption
(4)	$Forb(r, o, f) \leftrightarrow (\neg AP(r, o, f) \wedge \exists r' (R(r') \wedge (r > r') \wedge AP(r', a, f)))$	(2), (3), \rightarrow -
(5)	$\neg Forb(r, o, f) \leftrightarrow (AP(r, o, f) \vee \neg \exists r' (R(r') \wedge (r > r') \wedge AP(r', o, f)))$	(4), FOL
(6)	$\neg Forb(r, o, f) \wedge \exists r' (R(r') \wedge (r > r') \wedge AP(r', o, f)) \leftrightarrow (AP(r, o, f) \vee \neg \exists r' (R(r') \wedge (r > r') \wedge AP(r', o, f))) \wedge \exists r' (R(r') \wedge (r > r') \wedge AP(r', o, f))$	(5), FOL
(7)	$\neg Forb(r, o, f) \wedge \exists r' (R(r') \wedge (r > r') \wedge AP(r', o, f)) \leftrightarrow (AP(r, o, f) \wedge \exists r' (R(r') \wedge (r > r') \wedge AP(r', o, f)))$	(6), FOL
(8)	$\forall r \forall o \forall f (R(r) \wedge OP(o) \wedge F(f) \rightarrow (\exists r' (R(r') \wedge (r > r') \wedge AP(r', o, f)) \leftrightarrow \exists r' (R(r') \wedge (r > r') \wedge Perm(r', o, f))))$	Lemma
(9)	$\exists r' (R(r') \wedge (r > r') \wedge AP(r', o, f)) \leftrightarrow \exists r' (R(r') \wedge (r > r') \wedge Perm(r', o, f))$	(8), (3), \rightarrow -

A Proof of Theorem in the FOL (Cont.)

(10)	$\neg \text{Forb}(r, o, f) \wedge \exists r' (R(r') \wedge (r > r') \wedge \text{Perm}(r', o, f)) \leftrightarrow (\text{AP}(r, o, f) \wedge \exists r' (R(r') \wedge (r > r') \wedge \text{AP}(r', o, f)))$	(7), (9), FOL
(11)	$\forall r \forall o \forall f (R(r) \wedge \text{OP}(o) \wedge F(f) \rightarrow (\text{Perm}(r, o, f) \leftrightarrow (\text{AP}(r, o, f) \wedge \neg \exists r' (R(r') \wedge (r > r') \wedge \text{AP}(r', o, f))))))$	(A1)
(12)	$R(r) \wedge \text{OP}(o) \wedge F(f) \rightarrow (\text{Perm}(r, o, f) \leftrightarrow (\text{AP}(r, o, f) \wedge \neg \exists r' (R(r') \wedge (r > r') \wedge \text{AP}(r', o, f))))$	(11), \forall -
(13)	$\text{Perm}(r, o, f) \leftrightarrow (\text{AP}(r, o, f) \wedge \neg \exists r' (R(r') \wedge (r > r') \wedge \text{AP}(r', o, f)))$	(3), (12), \rightarrow -
(14)	$\text{Perm}(r, o, f) \vee (\neg \text{Forb}(r, o, f) \wedge \exists r' (R(r') \wedge (r > r') \wedge \text{Perm}(r', o, f))) \leftrightarrow (\text{AP}(r, o, f) \wedge \exists r' (R(r') \wedge (r > r') \wedge \text{AP}(r', o, f))) \vee (\text{AP}(r, o, f) \wedge \neg \exists r' (R(r') \wedge (r > r') \wedge \text{AP}(r', o, f)))$	(10), (13), FOL
(15)	$\text{Perm}(r, o, f) \vee (\neg \text{Forb}(r, o, f) \wedge \exists r' (R(r') \wedge (r > r') \wedge \text{Perm}(r', o, f))) \leftrightarrow \text{AP}(r, o, f)$	(6), FOL
(16)	$\text{GA}(r, o, f) \leftrightarrow \text{Perm}(r, o, f) \vee (\neg \text{Forb}(r, o, f) \wedge \exists r' (R(r') \wedge (r > r') \wedge \text{Perm}(r', o, f)))$	(A0), FOL
(17)	$\text{GA}(r, o, f) \leftrightarrow \text{AP}(r, o, f)$	(15), (16), FOL
(18)	$\forall r \forall o \forall f (R(r) \wedge \text{OP}(o) \wedge F(f) \rightarrow (\text{GA}(r, o, f) \leftrightarrow \text{AP}(r, o, f)))$	(3),(8), FOL

A Proof of Lemma in the FOL

(1)	$R(r)$	Assumption
(2)	$\exists r' (R(r') \wedge (r > r') \wedge AP(r', o, f))$	Assumption
(3)	$R(r_0) \wedge (r > r_0) \wedge AP(r_0, o, f)$	(2), \exists -
(4)	$\forall r \forall o \forall f (R(r) \wedge OP(o) \wedge F(f) \wedge AP(r, o, f) \rightarrow \exists r'' (R(r'') \wedge (r = r'' \vee r > r'') \wedge AP(r'', o, f) \wedge \neg \exists r' (R(r') \wedge (r'' > r') \wedge AP(r', o, f)))$	(A3)
(5)	$R(r_0) \wedge OP(o) \wedge F(f) \wedge AP(r_0, o, f) \rightarrow \exists r'' (R(r'') \wedge (r_0 = r'' \vee r_0 > r'') \wedge AP(r'', o, f) \wedge \neg \exists r' (R(r') \wedge (r'' > r') \wedge AP(r', o, f)))$	(4), \forall -
(6)	$OP(o) \wedge F(f)$	Assumption
(7)	$R(r_0) \wedge OP(o) \wedge F(f) \wedge AP(r_0, o, f)$	(3), (6), \wedge -+
(8)	$\exists r'' (R(r'') \wedge (r_0 = r'' \vee r_0 > r'') \wedge AP(r'', o, f) \wedge \neg \exists r' (R(r') \wedge (r'' > r') \wedge AP(r', o, f)))$	(5), (7), \rightarrow -
(9)	$R(r_1) \wedge (r_0 = r_1 \vee r_0 > r_1) \wedge AP(r_1, o, f) \wedge \neg \exists r' (R(r') \wedge (r_1 > r') \wedge AP(r', o, f))$	(8), \exists -
(10)	$\forall r \forall o \forall f (R(r) \wedge OP(o) \wedge F(f) \rightarrow (\text{Perm}(r, o, f) \leftrightarrow (AP(r, o, f) \wedge \neg \exists r' (R(r') \wedge (r > r') \wedge AP(r', o, f))))))$	(A1)

A Proof of Lemma in the FOL (Cont.)

(11)	$R(r1) \wedge OP(o) \wedge F(f) \rightarrow (Perm(r1, o, f) \leftrightarrow (AP(r1, o, f) \wedge \neg \exists r' (R(r') \wedge (r1 > r') \wedge AP(r', o, f))))$	(10), \forall -
(12)	$R(r1) \wedge OP(o) \wedge F(f)$	(6), (9), \wedge -+
(13)	$Perm(r1, o, f) \leftrightarrow (AP(r1, o, f) \wedge \neg \exists r' (R(r') \wedge (r1 > r') \wedge AP(r', o, f)))$	(11), (12), \rightarrow -
(14)	$(AP(r1, o, f) \wedge \neg \exists r' (R(r') \wedge (r1 > r') \wedge AP(r', o, f))) \rightarrow Perm(r1, o, f)$	(13), \leftrightarrow -
(15)	$(r0=r1 \vee r0>r1) \wedge AP(r1, o, f) \wedge \neg \exists r' (R(r') \wedge (r1 > r') \wedge AP(r', o, f)) \rightarrow (r0=r1 \vee r0>r1) \wedge Perm(r1, o, f)$	(14), FOL
(16)	$(r0=r1 \vee r0>r1) \wedge Perm(r1, o, f)$	(9), (15), FOL
(17)	$R(r1) \wedge (r>r1) \wedge Perm(r1, o, f)$	(3), (9), (16), trans. FOL
(18)	$\exists r' (R(r') \wedge (r>r') \wedge Perm(r', o, f))$	(17), \exists +
(19)	$\exists r' (R(r') \wedge (r>r') \wedge AP(r', o, f)) \rightarrow \exists r' (R(r') \wedge (r>r') \wedge Perm(r', o, f))$	(2), (18), \rightarrow +
(20)	$\exists r' (R(r') \wedge (r>r') \wedge Perm(r', o, f))$	Assumption

A Proof of Lemma in the FOL (Cont.)

(21)	$R(r1) \wedge (r>r1) \wedge \text{Perm}(r1, o, f)$	(20), $\exists-$
(22)	$\text{Perm}(r1, o, f) \rightarrow (AP(r1, o, f) \wedge \neg \exists r' (R(r') \wedge (r1>r') \wedge AP(r', o, f)))$	(13), $\leftrightarrow -$
(23)	$\text{Perm}(r1, o, f) \rightarrow AP(r1, o, f)$	(22), FOL
(24)	$R(r1) \wedge (r>r1) \wedge \text{Perm}(r1, o, f) \rightarrow R(r1) \wedge (r>r1) \wedge AP(r1, o, f)$	(23), FOL
(25)	$R(r1) \wedge (r>r1) \wedge AP(r1, o, f)$	(21), (24), $\rightarrow -$
(26)	$\exists r' (R(r1) \wedge (r>r') \wedge AP(r', o, f))$	(25), $\exists+$
(27)	$\exists r' (R(r') \wedge (r>r') \wedge \text{Perm}(r', o, f)) \rightarrow \exists r' (R(r') \wedge (r>r') \wedge AP(r', o, f))$	(20), (26), $\rightarrow +$
(28)	$\exists r' (R(r') \wedge (r>r') \wedge AP(r', o, f)) \leftrightarrow \exists r' (R(r') \wedge (r>r') \wedge \text{Perm}(r', o, f))$	(18), (26), $\leftrightarrow +$
(29)	$R(r) \wedge OP(o) \wedge F(f) \rightarrow (\exists r' (R(r') \wedge (r>r') \wedge AP(r', o, f)) \leftrightarrow \exists r' (R(r') \wedge (r>r') \wedge \text{Perm}(r', o, f)))$	(1), (6), (28), $\rightarrow +$
(30)	$\forall r \forall o \forall f (R(r) \wedge OP(o) \wedge F(f) \rightarrow (\exists r' (R(r') \wedge (r>r') \wedge AP(r', o, f)) \leftrightarrow \exists r' (R(r') \wedge (r>r') \wedge \text{Perm}(r', o, f))))$	(29), $\forall+$

Results on Efficiency

- More theorems can be proved:
$$\forall r \forall o \forall f (R(r) \wedge OP(o) \wedge F(f) \rightarrow (Perm(r, o, f) \rightarrow AP(r, o, f)))$$
$$\forall r \forall o \forall f (R(r) \wedge OP(o) \wedge F(f) \rightarrow (Forb(r, o, f) \rightarrow \neg AP(r, o, f)))$$
- *Perm* is a subset of *AP*
- *Forb* is a subset of the complement set of *AP*
- The richer the role hierarchy, the smaller *Perm*
- The larger *AP*, the smaller *Forb*
- When $\text{cardinality_of}(Perm) + \text{cardinality_of}(Forb) < \text{cardinality_of}(AP)$, savings on memory space is gained
- Efficient to use HRBAC model when *AP* is a big set and roles can be fully organized into a rich hierarchy.

Conclusions

- Non-tool-aided formal proofs should be advocated. This largely increases the assurance level, but does not necessary substantially increase the cost.
- A mathematical proof not only verifies the correctness of a statement, but also provides a better understanding of what has been proven.
- Proofs done by hand with paper and pencil are as valuable as proofs constructed by automated tools.
- Not enough to say something is secure. Prove it!

Acknowledgement

I thank my atsec colleagues for the review and helpful feedback:

Brenda Grove

Helmut Kurth

Fiona Pattinson

David Ochel

References

- Yi Mao and Mike Montgomery, “Hierarchical Role-Based Access Control” e-Smart 2006, Sept. 20-22, Sophia Antipolis, France.
- Chris Walsh, “Developing a CC EAL7 Multi-Level Security Capability”.
- Raymond Richards, *et al.*, “The Common Criteria, Formal Methods and ACL2”.
- Stéphanie Motré and Corinne Téri, “Using B Method to Formalize the Java Card Runtime Security Policy for a Common Criteria Evaluation”, Proc. Of 23rd National Information Systems Security Conference (NISSC 2000), Baltimore, USA, October 16-19, 2000.
- David Hardin, *et al.*, “A Robust Machine Code Proof Framework for Highly Secure Applications”.

Thank you for your attention!