Automated Analysis of Access Control Policies

Alessandro Armando

joint work with Silvio Ranise

Artificial Intelligence Laboratory (AI-Lab)
DIST, University of Genova
Genova

Security & Trust Research Unit
FBK-IRST
Trento

Access Control

- The process of
  - mediating requests to resources maintained by a system and
  - determining whether a request should be granted or denied
- Crucial role in system security
- Usually separation between
  - policies specified by a language with an underlying model
  - mechanisms enforcing policies
- Separation implies
  - protection requirements are independent of their implementation
  - analysis of policies can be done abstractly

Role-based Access Control

User	Permission
Alice	GrantTenure
Alice	AssignGrades
Alice	ReceiveHBenefits
Alice	UseGym
Bob	GrantTenure
Bob	AssignGrades
Bob	UseGym
Charlie	GrantTenure
Charlie	AssignGrades
Charlie	UseGym
David	AssignHWScores
David	Register4Courses
David	UseGym
Eve	ReceiveHBenefits
Eve	UseGym
Fred	Register4Courses
Fred	UseGym
Greg	UseGym

Role-based Access Control

User Assignment (UA)

User	Role
Alice	PCMember
Bob	Faculty
Charlie	Faculty
David	TA
David	Student
Eve	UEmployee
Fred	Student
Greg	UMember

Permission Assignment (PA)

Role	Permission
PCMember	GrantTenure
PCMember	AssignGrades
PCMember	ReceiveHBenefits
PCMember	UseGym
Faculty	AssignGrades
Faculty	ReceiveHBenefits
Faculty	UseGym
TA	AssignHWScores
TA	Register4Courses
TA	UseGym
UEmployee	ReceiveHBenefits
UEmployee	UseGym
Student	Register4Courses
Student	UseGym
UMember	UseGym

Role-based Access Control

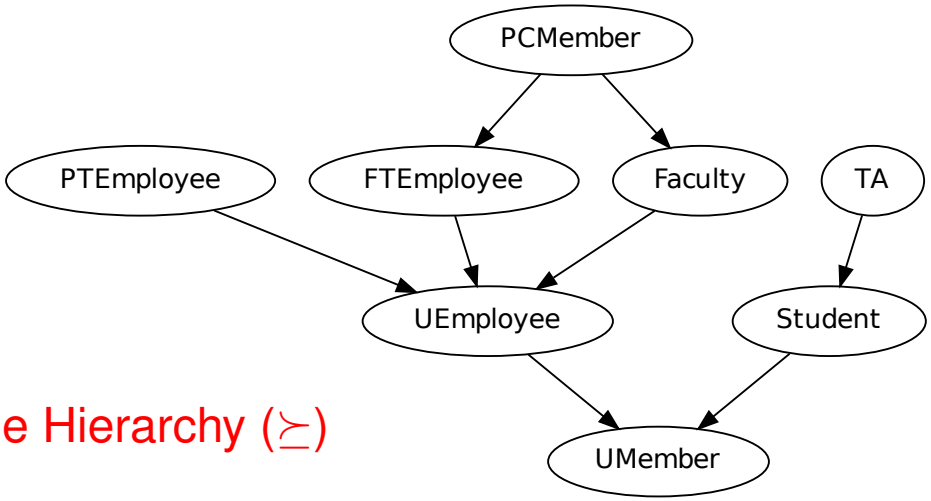
Permission Assignment (PA)

Role	Permission
PCMember	GrantTenure
Faculty	AssignGrades
TA	AssignHWScores
UEmployee	ReceiveHBenefits
Student	Register4Courses
UMember	UseGym

User Assignment (UA)

User	Role
Alice	PCMember
Bob	Faculty
Charlie	Faculty
David	TA
David	Student
Eve	UEmployee
Fred	Student
Greg	UMember

Role Hierarchy ($\succeq$)



Administrative RBAC

- Changes to RBAC policies subject to administrative policy.
- Several administrative models for RBAC: ARBAC97, SARBAC, Oracle DBMS, UARBAC, ...
- Key issue: definition of administrative domains, e.g.
  - ARBAC: admin. domain = role-based
  - UARBAC: admin. domain = attribute-based

ARBAC97: URA97 sub-model

In URA97, administrative actions can only modify the User Assignment (UA) relation.

- can_assign:
  $UEmployee : \{Student, \overline{TA}\} \Longrightarrow \oplus PTEmployee$
- can_revoke:
  $UEmployee : \{Student\} \Longrightarrow \ominus Student$

User	Role
Alice	PCMember
Bob	Faculty
Charlie	Faculty
David	TA
David	Student
Eve	UEmployee
Fred	Student
Greg	UMember

ARBAC97: URA97 sub-model

In URA97, administrative actions can only modify the User Assignment (UA) relation.

- can_assign:
  $UEmployee : \{Student, \overline{TA}\} \Longrightarrow \oplus PTEmployee$
- can_revoke:
  $UEmployee : \{Student\} \Longrightarrow \ominus Student$

User	Role
Alice	PCMember
Bob	Faculty
Charlie	Faculty
David	TA
David	Student
Eve	UEmployee
Fred	Student
Greg	UMember

ARBAC97: URA97 sub-model

In URA97, administrative actions can only modify the User Assignment (UA) relation.

- can_assign:
  $UEmployee \ : \ \{Student, \overline{TA}\} \Longrightarrow \oplus PTEmployee$
- can_revoke:
  $UEmployee \ : \ \{Student\} \Longrightarrow \ominus Student$

User	Role
Alice	PCMember
Bob	Faculty
Charlie	Faculty
David	TA
David	Student
Eve	UEmployee
Fred	Student
Fred	PTEmployee
Greg	UMember

ARBAC97: URA97 sub-model

In URA97, administrative actions can only modify the User Assignment (UA) relation.

- can_assign:
  $UEmployee \ : \ \{Student, \overline{TA}\} \Longrightarrow \oplus PTEmployee$
- can_revoke:
  $UEmployee \ : \ \{Student\} \Longrightarrow \ominus Student$

User	Role
Alice	PCMember
Bob	Faculty
Charlie	Faculty
David	TA
David	Student
Eve	UEmployee
Fred	Student
Fred	PTEmployee
Greg	UMember

ARBAC97: URA97 sub-model

In URA97, administrative actions can only modify the User Assignment (UA) relation.

- can_assign:
  $UEmployee : \{Student, \overline{TA}\} \Longrightarrow \oplus PTEmployee$
- can_revoke:
  $UEmployee : \{Student\} \Longrightarrow \ominus Student$

User	Role
Alice	PCMember
Bob	Faculty
Charlie	Faculty
David	TA
David	Student
Eve	UEmployee
Fred	Student
Fred	PTEmployee
Greg	UMember

ARBAC97: URA97 sub-model

In URA97, administrative actions can only modify the User Assignment (UA) relation.

- can_assign:
  $UEmployee : \{Student, \overline{TA}\} \Longrightarrow \oplus PTEmployee$
- can_revoke:
  $UEmployee : \{Student\} \Longrightarrow \ominus Student$

User	Role
Alice	PCMember
Bob	Faculty
Charlie	Faculty
David	TA
David	Student
Eve	UEmployee
Fred	PTEmployee
Greg	UMember

Administering Access Control Policies

- (A)RBAC model simplifies specification and administration of access control policies.
- Yet, in large systems (e.g., Dresdner bank: 40,000 users and 1,400 permissions), administration of RBAC policies can be very difficult.
- **Question:** Starting fron an initial RBAC policy and using the administrative actions in the ARBAC policy, is there a way to grant Alice access to `salaries.xls`?
- To predict the effects of changes on policies of real-world complexity by manual inspection is unfeasible: automated support needed!

URA97: security analysis problems

Let ψ be an administrative policy.

1. (Bounded) user-role reachability problem: Given (an integer $k \geq 0$, resp.) an initial RBAC policy, and a role r, does there exist a sequence of administrative actions in ψ (of length k, resp) assigning a user u to role r?
2. Role containment: Given an initial RBAC policy and two roles r_1 and r_2, does every member of role r_1 also belong to role r_2 in all reachable policies by applying finite sequences of administrative actions in ψ?
3. Weakest precondition: Given a user u and a role r, compute the minimal set of RBAC policies from which a sequence of administrative actions in ψ can make u a member of role r.
4. Inductive policy invariant: Check if a property remain unaffected under any (finite) sequence of administrative actions in ψ.

URA97: symbolic representation (I)

- Symbolic representation of RBAC policies and properties
- Fragment of many-sorted first-order logic
  - Sorts: *User*, *Role*
  - Predicate symbols: $ua : User \times Role$ (flexible)

$$\forall u, r.(ua(u, r) \Leftrightarrow \left(\begin{array}{ll} (u = u_1 \wedge r = Role\ 1) & \vee \\ (u = u_2 \wedge r = Role\ 2) & \vee \\ (u = u_3 \wedge r = Role\ 3) & \vee \\ \vdots & \end{array} \right))$$

- There exists a user who is member of a certain role

$$\exists u, r.(ua(u, r) \wedge r \succeq Student)$$

- No user can be assigned both *TA* and *PTEmployee*

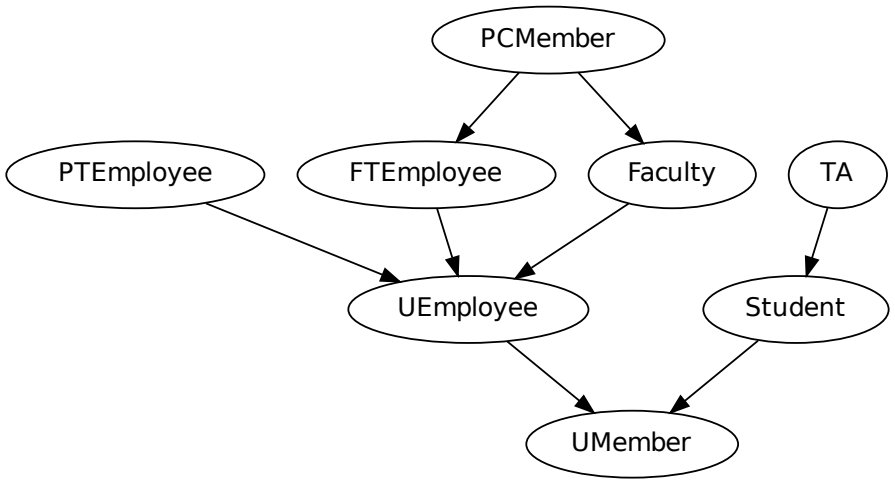
$$\forall u.\neg(ua(u, TA) \wedge ua(u, PTEmployee))$$

URA97: symbolic representation (II)

- Symbolic representation of structure underlying RBAC
- Fragment of many-sorted first-order logic
  - Sorts: *User*, *Role*
  - Predicate symbols: $\succeq$: $Role \times Role$ (rigid *partial order*)

$TA \succeq Student$,
$PTEmployee \succeq UEmployee$,
$UEmployee \succeq UMember$, ...,

$\forall r.(r \succeq r)$
$\forall r_1, r_2, r_3.((r_1 \succeq r_2 \wedge r_2 \succeq r_3) \Rightarrow r_1 \succeq r_3)$
$\forall r_1, r_2.((r_1 \succeq r_2 \wedge r_2 \succeq r_1) \Rightarrow r_1 = r_2)$



URA97: symbolic representation (III)

- Symbolic representation of structure underlying RBAC
- Fragment of many-sorted first-order logic
  - Predicate symbol: $ua : User \times Role$ (flexible)
  - ua and ua': before and after execution of action

- $UEmployee : \{Student, \overline{TA}\} \Longrightarrow \oplus PTEmployee$

$$\exists u_a, r_a.(ua(u_a, r_a) \wedge r_a \succeq UEmployee) \wedge$$
$$\exists u. \left(\begin{array}{l} ua(u, Student) \wedge \forall r_2.(r_2 \succeq TA \Rightarrow \neg ua(u, r_2)) \wedge \\ \forall x, y.(ua'(x, y) \Leftrightarrow ((x = u \wedge y = PTEmployee) \vee ua(x, y))) \end{array} \right)$$

- $UEmployee : \{Student\} \Longrightarrow \ominus Student$

$$\exists u_a, r_a.(ua(u_a, r_a) \wedge r_a \succeq UEmployee) \wedge$$
$$\exists u. \left(\begin{array}{l} \exists r_1.(ua(u, r_1) \wedge r_1 \succeq Student) \wedge \\ \forall x, y.(ua'(x, y) \Leftrightarrow (\neg(x = u \wedge y = Student) \wedge ua(x, y))) \end{array} \right)$$

Security analysis: bounded user-role reachability

Given an integer $k \geq 0$ and symbolic representation of

- T_{RBAC} = structure underlying RBAC policies
- $I(ua)$ = initial RBAC policy
- $G(ua)$ = user u is a member of role r
- $\tau(ua, ua')$ = administrative actions in ψ

Check the satisfiability of

$$T_{\text{RBAC}} \wedge I(ua_0) \wedge \tau(ua_0, ua_1) \wedge \cdots \wedge \tau(ua_{k-1}, ua_k) \wedge G(ua_k)$$

Can be reduced to the satisfiability of **B**ernays-**S**hönfinkel-**R**amsey formulae
$\Longrightarrow$ Decidable*!*

Security analysis: unbounded user-role reachability (I)

Given symbolic representation of

- T_{RBAC} = structure underlying RBAC policies
- $I(ua)$ = initial RBAC policy
- $G(ua)$ = user u is a member of role r
- $\tau(ua, ua')$ = administrative actions in ψ

Run a symbolic backward reachability procedure

- $R_0(ua) := G(ua)$ (goal)
- $R_{i+1}(ua) := \exists ua'.(R_i(ua') \wedge \tau(ua, ua'))$ (pre-image) for $i \geq 0$

Three requirements

1. Effective computation of BSR formulae for pre-images
2. Decidability of satisfiability of $(R_i \wedge I)$ (safety) and validity of $(R_{i+1} \Rightarrow R_i)$ (fix-point), both modulo T_{RBAC}
3. Termination of backward reachability

Security analysis: unbounded user-role reachability (II)

Effective computation of pre-images

> if pre-processing of negation in pre-conditions of administraitve actions to eliminate $\forall$

Satisfiability of $(R_i \wedge I)$ and validity of $(R_{i+1} \Rightarrow R_i)$ modulo T_{RBAC}

> can be reduced to satisfiability of BSR formulae
> $\Longrightarrow$ Decidable*!*

Termination of backward reachability

> by model-theoretic methods in combination with results on well-quasi-order

Security analysis: theoretical results, overview

Decidability of parameterized user-role reachability with respect to the number of users

- Role containment and weakest precondition can be reduced to unbounded user-role reachability
- Inductive policy invariant can be reduced to bounded user-role reachability

Extensions

- Parametric roles (limited use of negation in pre-conditions of administrative actions)
- Attributes (crucial for distributed and open environments)

Security analysis: practical results, overview (I)

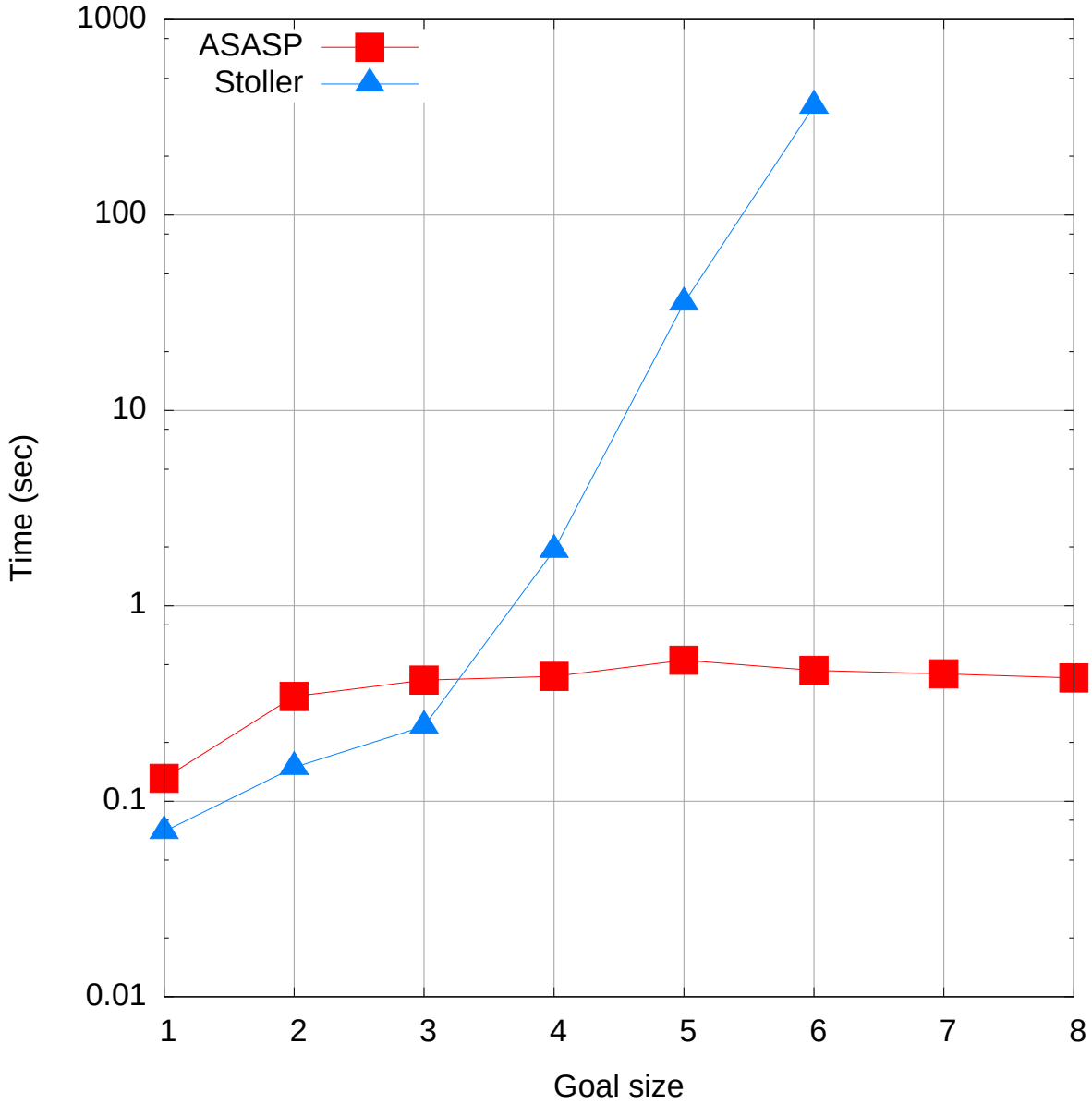
joint work with Francesco Alberti

Tool ASASP: Automated Symbolic Analysis of Administrative Policies

- architecture: client-server
- client = pre-image computation + generation of logical problems
- server = state-of-the-art SMT solvers and theorem provers on satisfiability problems
  - Z3, incomplete over BSR but incremental
  - SPASS (refutation) complete but not incremental
  - hierarchical combination of Z3 and SPASS
- Benchmarks for unbounded user-role reachability problem by Stoller *et al*
  - Parameter: goal size
  - Better scalability wrt. tool by Stoller *et al*

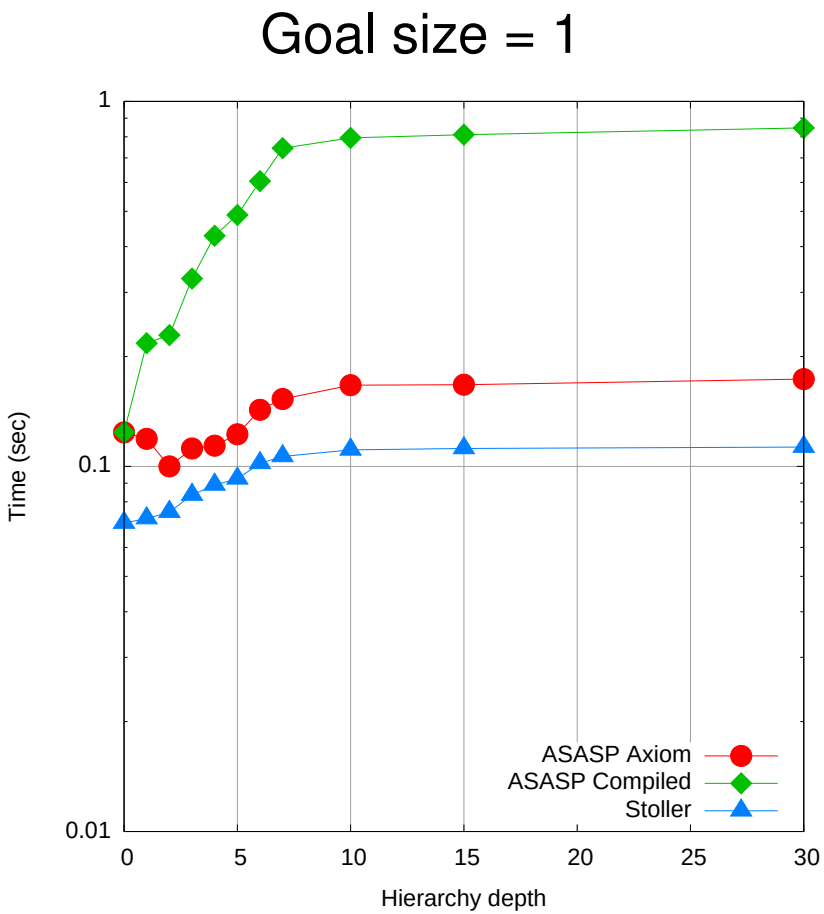
Security analysis: practical results, overview (II)

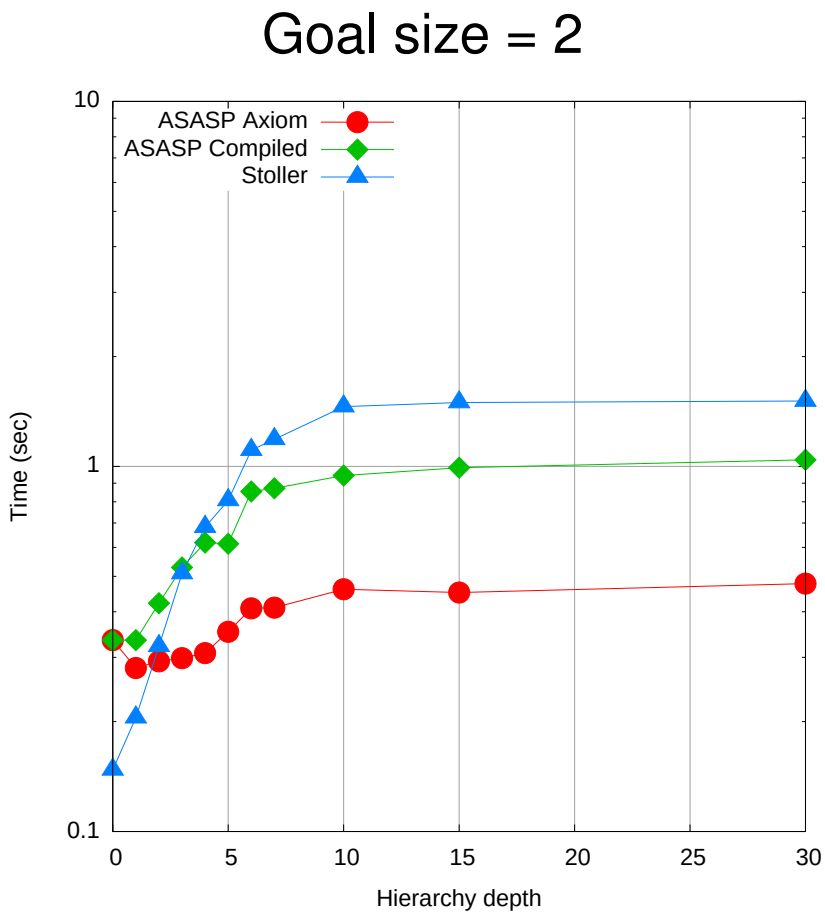
No role hierarchy



Security analysis: practical results, overview (III)

With role hierarchy

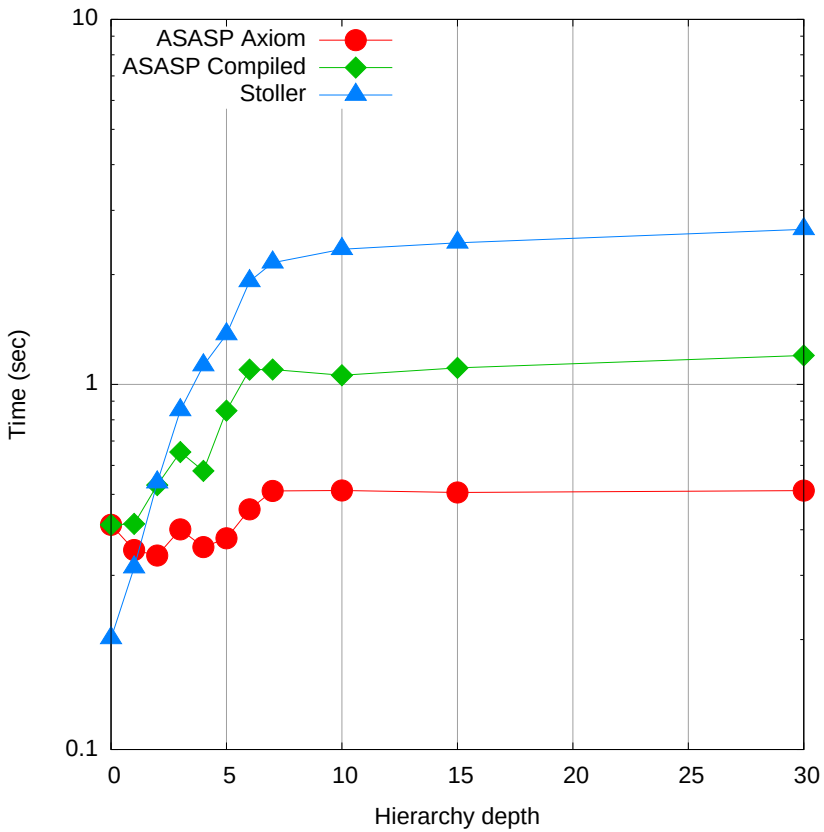




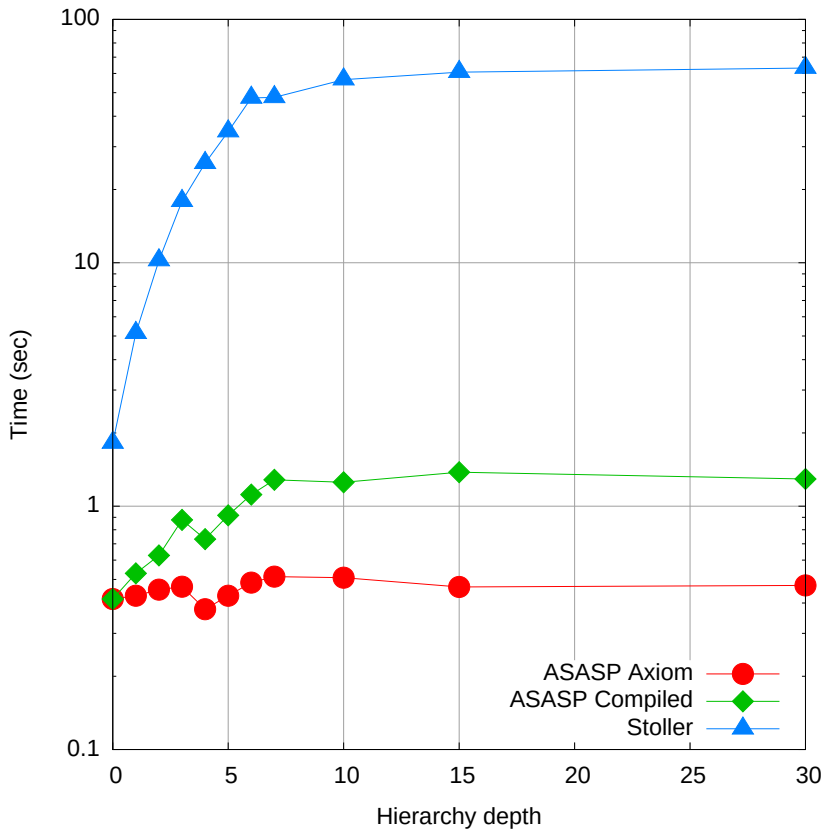
Security analysis: practical results, overview (IV)

With role hierarchy

Goal size = 3



Goal size = 4



Discussion

- Contributions
  -  Uniform and declarative specification/verification framework
  -  Parameterized security analysis of ARBAC
  -  Guaranteed termination of analysis
  -  Better scalability than state-of-the-art tools
- Future work
  -  Workflow systems
  -  Beyound BSR: policies with temporal/spatial constraints

References

- A. Armando and S. Ranise. **Automated Symbolic Analysis of ARBAC Policies**. In Proc. of 6th Int. Workshop on Security and Trust Management (STM'10), Athens, September 23-24, 2010.

- F. Alberti, A. Armando, and S. Ranise. **Efficient Symbolic Automated Analysis of Administrative Attribute-based RBAC-Policies**. In Proc. of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2011), March 22-24, 2011 (Hong Kong).

- Tool and benchmark problems publicly available at `http://st.fbk.eu`