Electronics and
Software Systems Group (ESS)

UNIVERSITY OF Southampton

VAC – Verifier of Access Control

Anna Lisa Ferrara[1], **P. Madhusudan**[2], **Truc L. Nguyen**[3], **and Gennaro Parlato**[3]
[1]University of Bristol, UK – [2]University of Illinois, USA – [3]University of Southampton, UK

Aims

Maintain desirable security properties of access control policies while delegating administrative privileges.

Access Control Policies are designed to support authorized access to protect resources. When administrative permissions are delegated, security properties could be violated.

Examples of security properties:

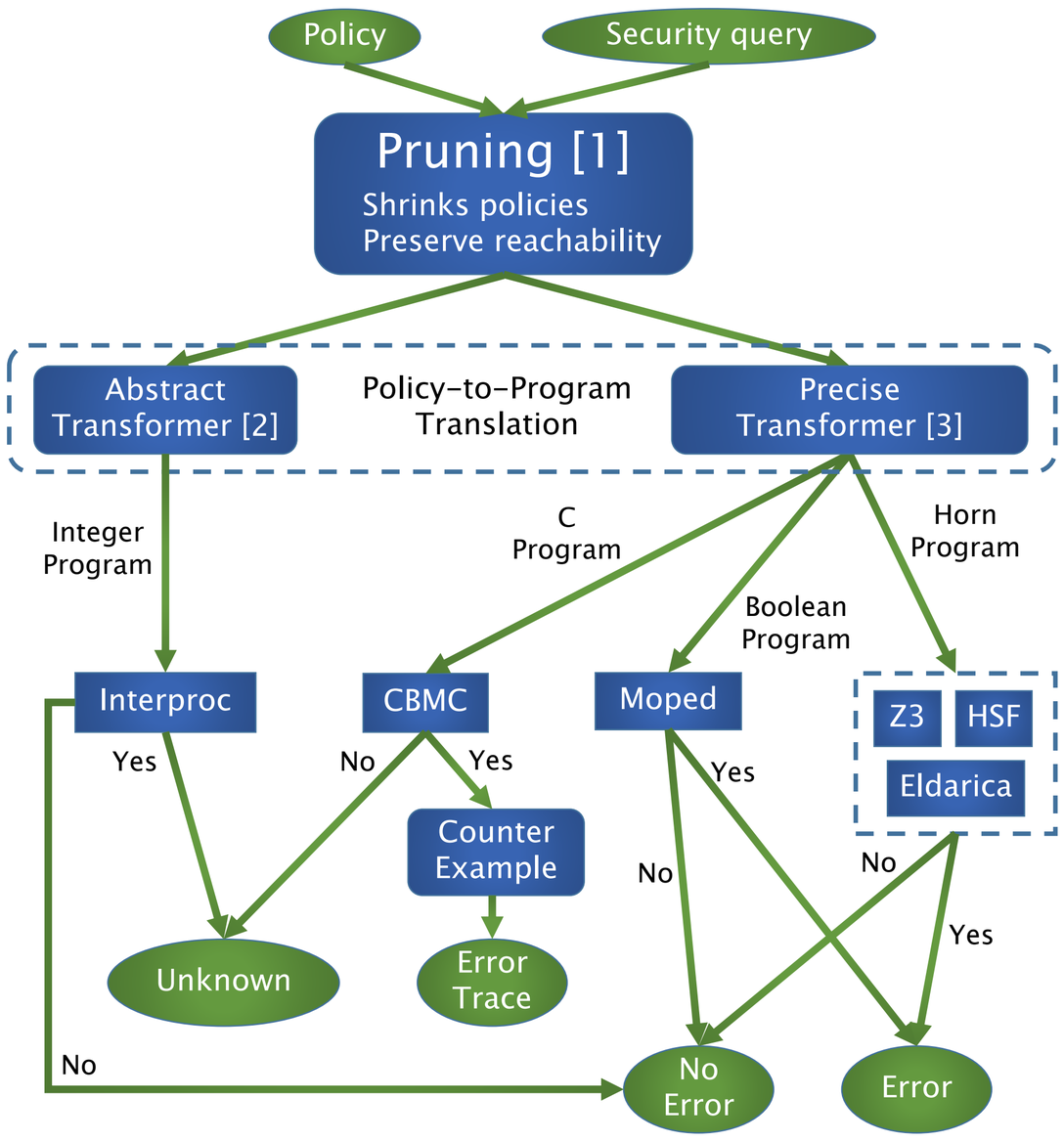
Availability properties:
– A doctor must always be able to access patients' record

Escalation of privileges:
– A receptionist cannot access patients' records

Conflict of interests:
– A doctor cannot be also a receptionist

Tool: Verifier of Access Control

- ✓ VAC is an automatic tool for the analysis of ARBAC policies.
- ✓ VAC uses a combination of reduction and transformation to program verification in order to perform the security analysis.
- ✓ VAC supports various back-ends, such as: Interproc, CBMC, HSF, Z3, Eldarica, Moped

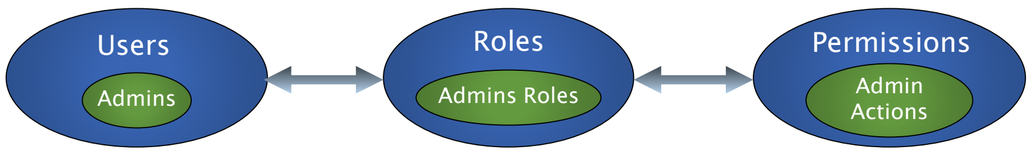


VAC architecture

Role-based Access Control

Role-based access control (**RBAC**) has emerged as a simple and effective access control mechanism for large organizations

- ✓ standardized by the NIST
- ✓ implement a variety of MAC and DAC policies
- ✓ is supported in several systems including: Oracle DBMS, Microsoft SQL Servers, Microsoft Active Directory, SELinux, NHS, …



RBAC simplifies policy specification and the management of user rights using a two tier management, it groups users into roles and assigns permissions to each role.
Administrative role-based access control (**ARBAC**) is a policy mechanism for controlling how changes can be made to the RBAC policy by various administrators.

Automated Analysis

Access control systems can been seen as state-transition systems. In an ARBAC system, state changes occur via administrative operations.
Security analysis techniques answer questions such as *whether an undesirable state is reachable and whether every reachable state satisfies some safety or availability properties.*

☹ Policies are difficult to inspect by hand (state space explosion)

☹ Monitoring is not acceptable (denial-of-service)

✓ **Static policy analysis is essential**

Experimental Results

VAC has been used to verify real policies and case studies

- ✓ Hospital Policies
- ✓ University Policies
- ✓ Bank Policies (case studies)
- ✓ Three suites of complex policies

			AFTER PRUNING				
#roles	#actions	#users	#roles	#actions	#users	Time	Violation
13	37	1092	4	5	6	0.029s	No
32	449	943	6	7	13	0.034s	No
531	5126	2000	2	0	2	0.253s	No
531	5126	2000	3	2	2	0.396s	Yes
4000	20000	10000	2	1	3	0.239s	Yes
20000	100000	50000	2	1	3	0.844s	Yes
30000	120000	70000	2	1	3	1.288s	Yes
40000	200000	10000	2	1	3	1.586s	Yes

VAC reduces the size of policies up to 90% on average.

Key results / Future work

- ➢ VAC effectively verifies all ARBAC policies from literature.
- ➢ The effectiveness of VAC mainly relies on Pruning module.
- ➢ VAC is easily extensible to interface with other back-ends.

We plan to enrich VAC with:
- ❖ Integration of VAC in policy designing phase.
- ❖ Analysis more ARBAC benchmarks from real-world policies.

Contact Information

VAC is publicly available on this site:
http://users.ecs.soton.ac.uk/gp4/VAC.html

For further information, please email to tnl2g10@soton.ac.uk

References:
[1] Policy Analysis for Self-Administrated Role-based Access Control (**TACAS'13**)
Ferrara, Madhusudan, Parlato
[2] Security Analysis of Role-based Access Control through Program Verification (**CSF'12**)
Ferrara, Madhusudan, Parlato
[3] VAC – Verifier of Administrative Role-base Access Control Policies (**CAV'14**)
Ferrara, Madhusudan, Nguyen, Parlato