Automatic Security Analysis of Business Processes

Daniel Ricardo dos Santos[1,2]

Advisors:
Silvio Ranise[1]
Luca Compagna[2]
Serena Ponta[2]

[1] Security and Trust - FBK
[2] SAP Labs France

October 21st, 2014

Outline

1. Introduction
2. State of the art
3. Preliminary Results
4. Future Work

SECENTIS

- This work is part of the SECENTIS project and aims to apply the resulting tools on the SAP HANA database and cloud platform

SECENTIS

A European Industrial Doctorate on Security and Trust

Context

- Business processes and process-aware applications need to enforce security policies in the form of complex authorization constraints
- Separation/Binding of Duty and others related to the execution history or contextual information (e.g., location/time)
- Termination (WSP), authorization delegation, and resiliency

Problem

- Developers may directly implement a policy in the application or use run-time enforcement monitors provided by the execution platform
- We must verify that the policy enforced by the application and the intended policy, specified by the business rules, are compatible
- We work on methods for synthesizing run-time monitors and analyzing database-backed web applications that realize workflows

Research Goals

- Given a workflow specification and a set of authorization constraints (policy), generate a run-time monitor that enforces the policy

- Given a process-aware application implemented in JavaScript+SQL and a set of authorization constraints (policy), detect and correct vulnerabilities in policy enforcement

State of the art

- Workflow Satisfiability has been extensively studied, but not the synthesis of a full monitor for causality and authorization constraints [1, 4]
- Deutsch et al. [6, 7, 5] worked on the specification and verification of data-driven web applications and business processes with correctness properties specified in temporal logic, but no special attention to security
- Policy-weaving problem: taking as input a program, a high-level policy and a description of how system calls affect privilege; automatically rewrite the program in a way that it satisies the policy [9, 8, 10]

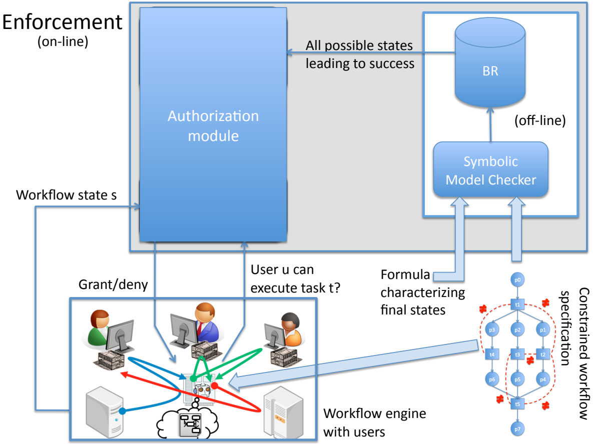
Automated Synthesis of Run-time Monitors

- New methodology to automatically synthesize run-time monitors capable of ensuring the successful termination of workflows while enforcing authorization policies and SoD constraints
- Divided in two parts: (i) specification and (ii) verification of security-aware workflows.
- Specification starts with Petri nets for the control-flow and security requirements, then derives a symbolic representation to be used by a model checker, considering a finite but unknown number of users.

Automated Synthesis of Run-time Monitors

- The verification part has an off-line and an on-line phase, in the off-line phase we compute all possible terminating executions of the workflow and in the on-line phase we use this information to synthesize a run-time monitor, that can be implemented in Datalog or SQL.
- Control-flow is DAG (no loops)
- Data-flow is completely abstracted

Architecture



Enforcement
(on-line)

Authorization module

All possible states leading to success

BR

(off-line)

Symbolic Model Checker

Workflow state s

Grant/deny

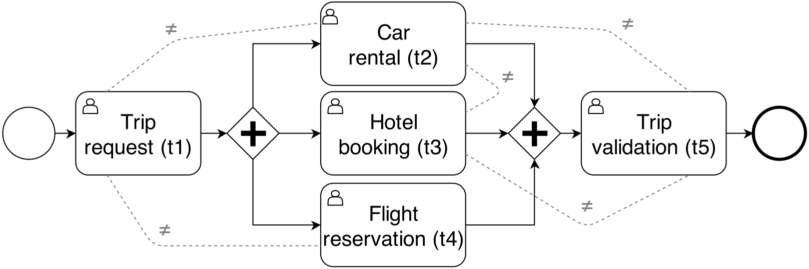
User u can execute task t?

Formula characterizing final states

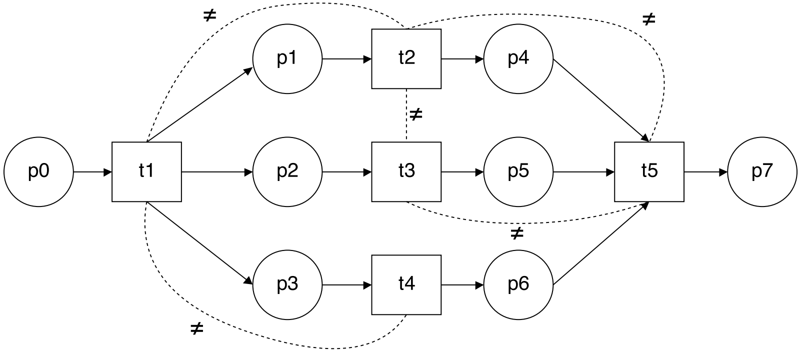
Constrained workflow specification

Workflow engine with users

Example - BPMN



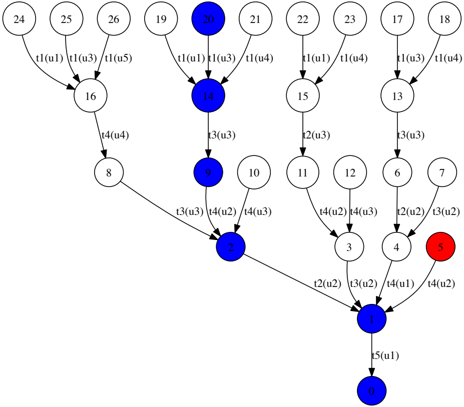
Example - Petri net



Example - Transition System

event	enabled		action	
	CF	Auth	CF	Auth
$t1(u)$	$p0\wedge\neg d_{t1}$	$a_{t1}(u)$	$p0, p1, p2, p3, d_{t1} := F, T, T, T, T$	$h_{t1}(u) := T$
$t2(u)$	$p1\wedge\neg d_{t2}$	$a_{t2}(u) \wedge \neg\, h_{t3}(u) \wedge \neg\, h_{t1}(u)$	$p1, p4, d_{t2} := F, T, T$	$h_{t2}(u) := T$
$t3(u)$	$p2\wedge\neg d_{t3}$	$a_{t3}(u) \wedge \neg h_{t2}(u)$	$p2, p5, d_{t3} := F, T, T$	$h_{t3}(u) := T$
$t4(u)$	$p3\wedge\neg d_{t4}$	$a_{t4}(u) \wedge \neg h_{t1}(u)$	$p3, p6, d_{t4} := F, T, T$	$h_{t4}(u) := T$
$t5(u)$	$p4\wedge p5\wedge p6\wedge\neg d_{t5}$	$a_{t5}(u) \wedge \neg\, h_{t3}(u) \wedge\ \neg\, h_{t2}(u)$	$p4, p5, p6, p7, d_{t5} := F, F, F, T, T$	$h_{t5}(u) := T$

Example - State Space

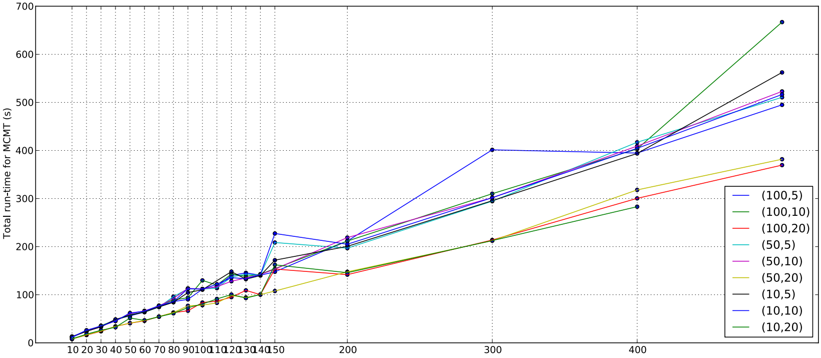


Example - Monitor

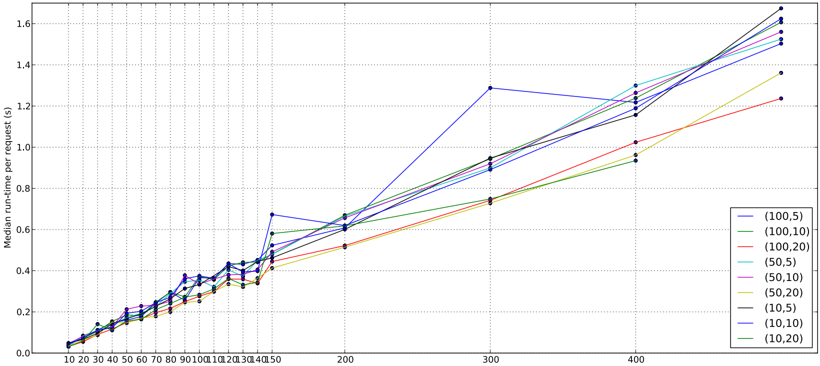
- $U = \{a, b, c\}, R = \{r_1, r_2, r_3\}$
- $UA = \{(a, r1), (a, r2), (a, r3), (b, r2), (b, r3), (c, r2)\}$
- $TA = \{(r_3, t1), (r_2, t2), (r_2, t3), (r_1, t4), (r_2, t5)\}$

	CF	Auth					*can_do*	
#	Token in	h_{t1}	h_{t2}	h_{t3}	h_{t4}	h_{t5}	(u, t)	Resp.
0	$p0$	-	-	-	-	-	$(a, t1)$	deny
1	$p0$	-	-	-	-	-	$(b, t1)$	grant
2	$p1, p2, p3$	b	-	-	-	-	$(b, t2)$	deny
3	$p1, p2, p3$	b	-	-	-	-	$(a, t2)$	grant
4	$p4, p2, p3$	b	a	-	-	-	$(c, t3)$	grant
5	$p4, p5, p3$	b	a	c	-	-	$(a, t4)$	grant
6	$p4, p5, p6$	b	a	c	a	-	$(b, t5)$	grant
7	$p7$	b	a	c	a	b	-	-

Results


Total run-time for MCMT (s)
0
100
200
300
400
500
600
700
10 20 30 40 50 60 70 80 90100110120130140150
200
300
400
(100,5)
(100,10)
(100,20)
(50,5)
(50,10)
(50,20)
(10,5)
(10,10)
(10,20)

Results


Median run-time per request (s)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
10 20 30 40 50 60 70 80 90 100 110 120 130 140 150
200
300
400
(100,5)
(100,10)
(100,20)
(50,5)
(50,10)
(50,20)
(10,5)
(10,10)
(10,20)

TestREx: a testbed for repeatable exploits

- A framework for packing and running applications with their environments; injecting exploits and monitoring their success; and generating security reports
- Provided with a corpus of example vulnerabilities
- Goal: A benchmark on which we can test the effectiveness of our techniques
- Developed in collaboration with Stanislav Dashevskyi

Future Work

- Overcome the limitations of our current monitor approach: control- and data-flow
- Test our results in SAP HANA, using workflows provided by them and their execution engine
- Work on policy analysis and policy-weaving for JavaScript
- Integrate TestREx with policy analysis and testing

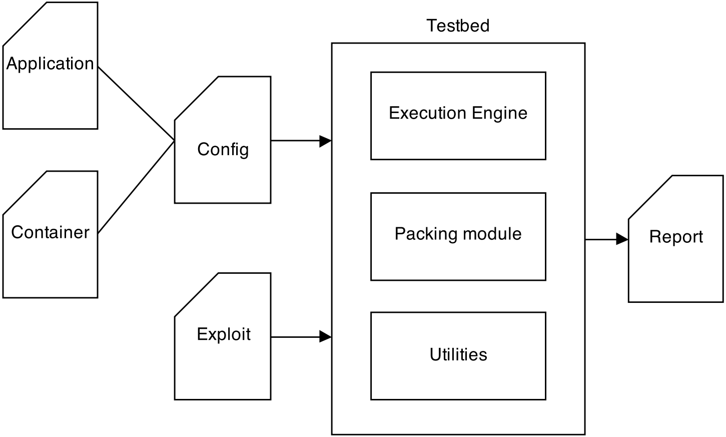
Future Work - other ideas to be considered

- User-role assignment ensuring least privilege in workflows
- Purpose-based access control for workflows

Thank you!

dossantos@fbk.eu

TestREx: a testbed for repeatable exploits



[1] David A. Basin, Samuel J. Burri, and Günter Karjoth. Dynamic enforcement of abstract separation of duty constraints. In *ESORICS*, pages 250–267, 2009.

[2] Paolina Centonze, Gleb Naumovich, Stephen J. Fink, and Marco Pistoia. Role-based access control consistency validation. In *Proceedings of the 2006 International Symposium on Software Testing and Analysis*, ISSTA '06, pages 121–132, New York, NY, USA, 2006. ACM.

[3] Adam Chlipala. Static checking of dynamically-varying security policies in database-backed applications. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 1–, Berkeley, CA, USA, 2010. USENIX Association.

[4] Jason Crampton, Michael Huth, and JimHuan-Pu Kuo. Authorized workflow schemas: deciding realizability through ltl(f) model checking. *International Journal on*

Software Tools for Technology Transfer, 16(1):31–48, 2014.

[5] Alin Deutsch, Richard Hull, Fabio Patrizi, and Victor Vianu. Automatic verification of data-centric business processes. In *Proceedings of the 12th International Conference on Database Theory*, ICDT '09, pages 252–267, New York, NY, USA, 2009. ACM.

[6] Alin Deutsch, Liying Sui, and Victor Vianu. Specification and verification of data-driven web services. In *Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '04, pages 71–82, New York, NY, USA, 2004. ACM.

[7] Alin Deutsch, Liying Sui, and Victor Vianu. Specification and verification of data-driven web applications. *Journal of*

Computer and System Sciences, 73(3):442 – 474, 2007. Special Issue: Database Theory 2004.

[8] Matthew Fredrikson, Richard Joiner, Somesh Jha, Thomas W. Reps, Phillip A. Porras, Hassen Saïdi, and Vinod Yegneswaran. Efficient runtime policy enforcement using counterexample-guided abstraction refinement. In P. Madhusudan and Sanjit A. Seshia, editors, *CAV*, volume 7358 of *Lecture Notes in Computer Science*, pages 548–563. Springer, 2012.

[9] WilliamR. Harris, Somesh Jha, and Thomas Reps. Secure programming via visibly pushdown safety games. In P. Madhusudan and SanjitA. Seshia, editors, *Computer Aided Verification*, volume 7358 of *Lecture Notes in Computer Science*, pages 581–598. Springer Berlin Heidelberg, 2012.

[10] Richard Joiner, Thomas Reps, Somesh Jha, Mohan

Dhawan, and Vinod Ganapathy. Efficient runtime enforcement techniques for policy weaving. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2014)*, 2014.