Assisting the Deployment of Security-Sensitive Workflows by Finding Execution Scenarios

Daniel R. dos Santos[1,2,3], Silvio Ranise[1],
Luca Compagna[2], Serena E. Ponta[2]

[1]Fondazione Bruno Kessler, [2]SAP Labs France,
[3]University of Trento

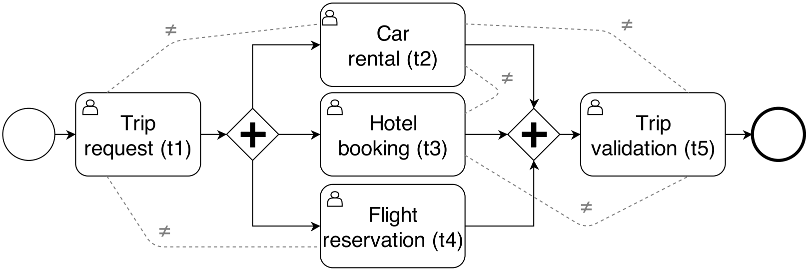
SECENTIS
A European Industrial Doctorate on Security and Trust

Outline

Outline

Context

- A workflow specifies a collection of tasks and the causal relationships between them
- Authorization policies specify which users can execute which tasks
- Additional constraints, such as Separation/Binding of Duty, further restrict the execution of tasks by users
  - Important to avoid errors and fraud, limiting the opportunities for abuse
- We call those workflows security-sensitive



task	roles
t1	r3
t2	r2
t3	r2
t4	r1
t5	r2

roles	users
r1	a
r2	a, b, c
r3	a, b

Workflow Satisfiability Problem

WSP

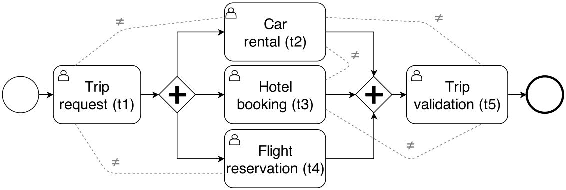
Is there an assignment of users to tasks such that a workflow terminates while satisfying all authorization constraints?

Run-time WSP

Answering sequences of user requests at execution time ensuring termination with the satisfaction of authorization constraints

WSP is NP-hard already in the presence of one SoD constraint

Problem

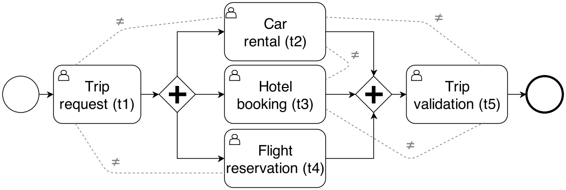


task	roles
t1	r3
t2	r2
t3	r2
t4	r1
t5	r2

roles	users
r1	a
r2	a, b, c
r3	a, b

Scenario = t1(a), t2(b), t3(c), X

WSP solution



task	roles
t1	r3
t2	r2
t3	r2
t4	r1
t5	r2

roles	users
r1	a
r2	a, b, c
r3	a, b

Scenario = t1(b), t2(a), t3(c), t4(a), t5(b)

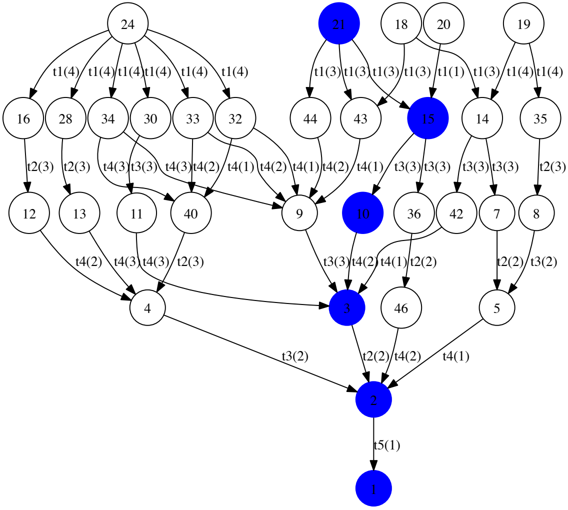
Solving the WSP - previous work

- Technique for the synthesis of run-time monitors for the WSP
  - C. Bertolissi, D. R. dos Santos, and S. Ranise. “Automated synthesis of run-time monitors to enforce authorization policies in business processes” in ASIACCS 2015
- It takes as input the specification of a security-sensitive workflow and an authorization policy and consists of two steps (off-line and on-line)
- Authorization policy is only considered in the on-line phase, the result of the off-line phase is based on a set of symbolic users and an unconstrained policy

Off-line

- A symbolic transition system S is derived from the execution constraints and the authorization constraints
- S is used to compute a symbolic reachability graph RG
  - Edges labeled by task-user pairs in which users are symbolically represented by variables
  - Nodes labeled by a symbolic representation of the set of states from which it is possible to reach a state in which the workflow successfully terminates
- Any path from a leaf node to the root corresponds to a symbolic eligible scenario

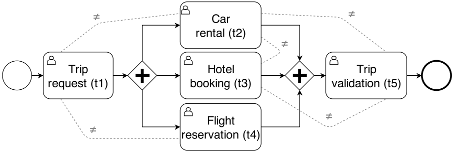
Off-line - Symbolic Reachability Graph



On-line

- A Datalog program is derived from *RG* by building clauses from the formulae labeling the nodes
- Each clause invokes predicates *auth*, which is an interface to the authorization policy, and *h*, which keeps track of which user has executed which task
- If the monitor grants a request (clause is satisfied), then the user can execute the task and the workflow can terminate while satisfying the authorization policy and the authorization constraints

Run-time WSP solution



task	roles
t1	r3
t2	r2
t3	r2
t4	r1
t5	r2

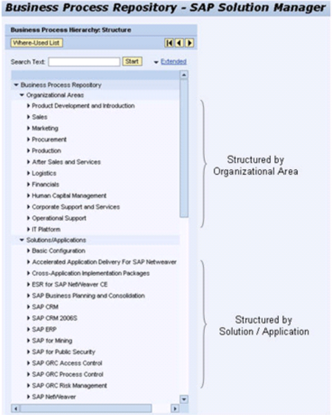
roles	users
r1	a
r2	a, b, c
r3	a, b

#	History	Query	Answer
0	$\emptyset$	$can\_do(a, t1)$	deny
1	-	$can\_do(b, t1)$	grant
2	$h(t1, b)$	$can\_do(c, t3)$	grant
3	$h(t3, c)$	$can\_do(a, t4)$	grant
4	$h(t4, a)$	$can\_do(b, t2)$	deny
5	-	$can\_do(a, t2)$	grant
6	$h(t2, a)$	$can\_do(b, t5)$	grant
7	$h(t5, b)$	-	-

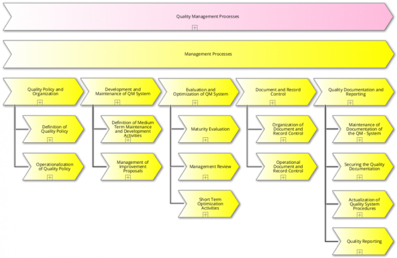
New problem

- Customers re-use business process models with authorization constraints from a repository
  - Usually, BPMN and constraints are available at design-time
  - Authorization policy is decided during deployment
- They must know, at deployment time, whether a process can be successfully instantiated with their authorization policy

Business Process Repositories



SAP BPR



Signavio Process Library

Business Process Repositories

- Many solutions to the WSP are less interesting in this scenario because they do not exploit re-use
- Design-time/run-time separation and re-use allow us to pre-compute and store part of the solution
- Later, we find concrete execution scenarios showing termination

Contributions

- Statements of two Scenario Finding Problems (SFPs) and discussion of their relationships with the WSP
- Solution based on a technique for the synthesis of run-time monitors for the WSP
- Validation of the solution on real-world examples from a repository of re-usable business process models

Outline

Scenario Finding Problems (SFP)

- An execution scenario is a sequence of user-task pairs
  - A scenario is eligible if all the constraints are satisfied
  - A scenario is authorized if every user is authorized to perform the assigned task
- It is possible to compute—once and for all—the set of eligible scenarios associated to a workflow in a repository
- The problem is then to look for those scenarios with some properties when an authorization policy becomes available

Scenario Finding Problems

Basic (B-SFP)

Given the set of eligible scenarios according to a set of authorization constraints, return a scenario which is authorized according to a given authorization relation

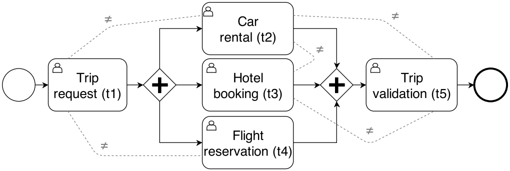
Minimal User Base SFP (MUB-SFP)

Given the set of eligible scenarios according to a set of authorization constraints, return a scenario which is authorized according to a given authorization relation and such that the set of users occurring in it is a minimal user base

SFP vs. WSP

- WSP:
  - Input: Workflow with constraints and authorization policy
  - Output: Eligible and authorized scenario
- B-SFP:
  - Input: Workflow with constraints and authorization policy + Pre-computed set of eligible scenarios
  - Output: Eligible and authorized scenario
- MUB-SFP:
  - Input: Workflow with constraints and authorization policy + Pre-computed set of eligible scenarios
  - Output: Eligible and authorized minimal scenario

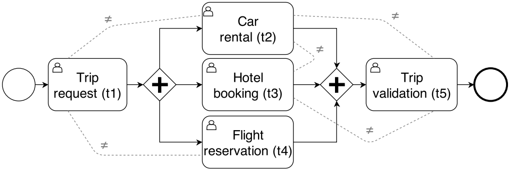
Example - eligible scenarios



$$U = \{Alice, Bob, Charlie, Dave, Erin, Frank\}$$

- $\eta_1 = t_1(Alice), t_2(Bob), t_3(Charlie), t_4(Dave), t_5(Erin)$
- $\eta_2 = t_1(Bob), t_2(Alice), t_3(Charlie), t_4(Alice), t_5(Bob)$
- $\eta_3 = t_1(Bob), t_4(Charlie), t_2(Alice), t_3(Dave), t_5(Bob)$
- ... (total = 19,080 scenarios)

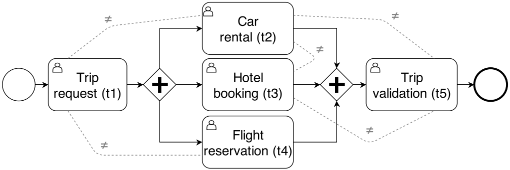
Example - authorized scenarios



$$TA = \{(Alice, t_1), (Bob, t_1), (Alice, t_2), (Bob, t_2),$$
$$(Charlie, t_3), (Alice, t_4), (Dave, t_4), (Bob, t_5), (Erin, t_5)\}$$

- $\eta_1 = t_1(Alice), t_2(Bob), t_3(Charlie), t_4(Dave), t_5(Erin)$
- $\eta_2 = t_1(Bob), t_2(Alice), t_3(Charlie), t_4(Alice), t_5(Bob)$
- $\eta_3 = t_1(Bob), t_4(Charlie), t_2(Alice), t_3(Dave), t_5(Bob)$
- ...

Example - minimal scenarios



$$TA = \{(Alice, t_1), (Bob, t_1), (Alice, t_2), (Bob, t_2), (Charlie, t_3), (Alice, t_4), (Dave, t_4), (Bob, t_5), (Erin, t_5)\}$$

- $\eta_1 = t_1(Alice), t_2(Bob), t_3(Charlie), t_4(Dave), t_5(Erin)$
- $\eta_2 = t_1(Bob), t_2(Alice), t_3(Charlie), t_4(Alice), t_5(Bob)$
- $\eta_3 = t_1(Bob), t_4(Charlie), t_2(Alice), t_3(Dave), t_5(Bob)$
- ...

Sketch of solution - B-SFP

- A scenario solving the B-SFP is also a solution of the WSP and vice versa, so we could re-use an algorithm $\mathcal{A}$ returning answers to the WSP to solve the B-SFP
- We would have to invoke $\mathcal{A}$ for every task-user pair in a scenario, because it does not exploit the fact that the scenarios in E are eligible
- A better approach is to consider each eligible scenario and check if all task-user pairs are authorized

Sketch of solution - MUB-SFP

- Maintain a variable storing an eligible and authorized scenario which is a candidate minimal user base
- When all eligible scenarios have been considered, we find a minimal user base

Outline

Solving the SFPs

- We reuse the monitor synthesis technique from ASIACCS2015
- It provides us with a compact data structure to represent the set of all eligible scenarios in a workflow
  - Reachability graph with symbolic users

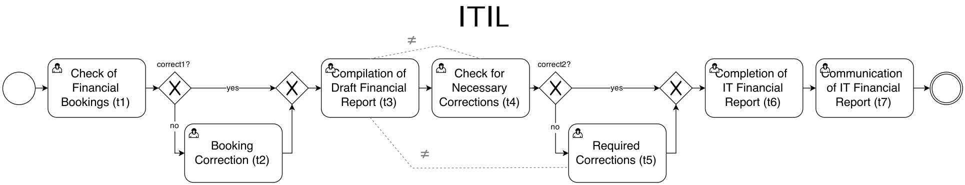
Solving the SFPs

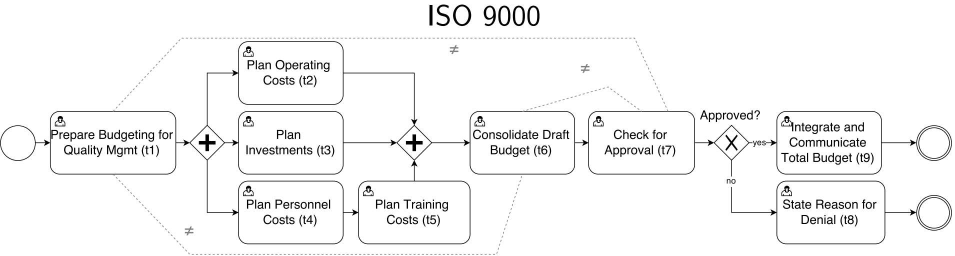
- Algorithm based on DFS to explore all paths in *RG* and check that the scenario is authorized by using the run-time monitor
- Instead of enumerating all users, we exploit a Datalog engine to find the right user
- Basic solution is extended with a set of facts which drives the search for a scenario with particular characteristics (conditionals, user-task pairs, etc.)

for all $v \in \text{Nodes}(RG)$ **do** $visited[v] \leftarrow \texttt{false}$;
end for
$\eta \leftarrow \epsilon$; $NI \leftarrow \text{NoIncoming}(RG)$;
while $(v \in NI$ **and** $\eta = \epsilon)$ **do**
    $\eta \leftarrow \mathrm{DFS}(v, \epsilon, \Gamma)$; $NI \leftarrow NI \setminus \{v\}$;
end while
return η
function $\mathrm{DFS}(v, \eta, H)$
    $visited[v] \leftarrow \texttt{true}$; $OE \leftarrow \text{OutGoing}(v)$;
    if $OE = \emptyset$ **then return** η
    else
        for all $v \xrightarrow{t(\upsilon)} w \in OE$ **do**
            if (**not** $visited[w]$ **and** $M, P, H \vdash^{\upsilon \mapsto u} can\_do(t, \upsilon))$ **then**
                return $\mathrm{DFS}(w, \text{append}(\eta, t(u)), H \cup \{h(t, u)\})$
            end if
        end for
    end if
    return ϵ
end function

Outline

Workflows


ITIL
Check of Financial Bookings (t1)
correct1?
yes
no
Booking Correction (t2)
Compilation of Draft Financial Report (t3)
Check for Necessary Corrections (t4)
correct2?
yes
no
Required Corrections (t5)
Completion of IT Financial Report (t6)
Communication of IT Financial Report (t7)
≠
≠


ISO 9000
Prepare Budgeting for Quality Mgmt (t1)
Plan Operating Costs (t2)
Plan Investments (t3)
Plan Personnel Costs (t4)
Plan Training Costs (t5)
Consolidate Draft Budget (t6)
Check for Approval (t7)
Approved?
yes
no
Integrate and Communicate Total Budget (t9)
State Reason for Denial (t8)
≠
≠
≠

Experimental setting

- Policies with and without possible scenarios
- Facts about user assignments and choice of branches
- Algorithm implemented in Python and pyDatalog and experiments run on a MacBook Air 2014

Experiments - B-SFP

#	Instance	Facts	Solution Scenario	Time (s)
0	TRW+P_0	$\emptyset$	$t1(b), t2(a), t4(a), t3(c), t5(b)$	0.288
1	ITIL+P_2	$\{c1, c2\}$	$t1(u3), t3(u9), t4(u8), t6(u9), t7(u9)$	4.267
2	ITIL+P_2	$\{c1, \textbf{not } c2\}$	$t1(u3), t3(u3), t4(u7), t5(u8), t6(u3), t7(u7)$	4.454
3	ITIL+P_2	$\{\textbf{not } c1, c2\}$	$t1(u3), t2(u1), t3(u9), t4(u8), t6(u9), t7(u9)$	4.374
4	ITIL+P_2	$\{\textbf{not } c1, \textbf{not } c2\}$	$t1(u3), t2(u1), t3(u3), t4(u7),$ $t5(u8), t6(u3), t7(u7)$	4.561
5	ISO+P_4	$\{appr\}$	$t1(u3), t4(u7), t5(u8), t2(u3), t3(u7),$ $t6(u9), t7(u7), t9(u8)$	6.581
6	ISO+P_4	$\{\textbf{not } appr\}$	$t1(u3), t4(u7), t5(u8), t2(u3), t3(u7),$ $t6(u7), t7(u8), t8(u6)$	6.637
7	TRW+P_1	$\emptyset$	ϵ	0.407
8	TRW+P_0	$\{t2(b)\}$	ϵ	1.554
9	ITIL+P_3	$\emptyset$	ϵ	9.562
10	ISO+P_5	$\emptyset$	ϵ	44.076

Experiments - MUB-SFP

#	Instance	Facts	Solution Scenario	Time (s)
11	TRW+P_0	$\emptyset$	$t1(b), t2(c), t3(b), t4(a), t5(a)$	2.385
12	ITIL+P_2	$\{c1, c2\}$	$t1(u1), t3(u1), t4(u7), t6(u1), t7(u1)$	108.819
13	ITIL+P_2	$\{c1, \textbf{not } c2\}$	$t1(u3), t3(u3), t4(u7), t5(u7), t6(u3), t7(u3)$	116.525
14	ITIL+P_2	$\{\textbf{not } c1, c2\}$	$t1(u1), t2(u1), t3(u1), t4(u7), t6(u1), t7(u1)$	108.827
15	ITIL+P_2	$\{\textbf{not } c1, \textbf{not } c2\}$	$t1(u3), t2(u3), t3(u3), t4(u7), t5(u7), t6(u3), t7(u3)$	116.533
16	ISO+P_4	$\{appr\}$	$t1(u5), t3(u5), t2(u5), t4(u5), t5(u5), t6(u3), t7(u7), t9(u7)$	166.632
17	ISO+P_4	$\{\textbf{not } appr\}$	$t1(u5), t3(u5), t2(u5), t4(u5), t5(u5), t6(u9), t7(u6), t8(u9)$	166.644

Outline

Conclusions

- We have introduced two SFPs and algorithms to solve them
- The approach exploits the fact that the eligible scenarios can be computed once and re-used with every authorization policy
- Experimental evaluation shows that it can be used at deployment time since it performs the computationally heaviest part when the workflow is added to a repository
- We intend to study
  - resiliency in SFPs
  - automatically suggest changes to authorization policies so that solutions of an SFP are optimal with respect to some criteria, e.g., least privilege.

Thank you!

`www.secentis.eu`

bertolissi@fbk.eu

dossantos@fbk.eu

ranise@fbk.eu