Aegis: Automatic Enforcement of Security Policies in Workflow-driven Web Applications

Luca Compagna[2], **Daniel Ricardo dos Santos**[1,2,3],
Serena Elisa Ponta[2], Silvio Ranise[1]

[1]Fondazione Bruno Kessler (FBK)
[2]SAP Labs France
[3]University of Trento

CODASPY 2017 - March 22-24, Scottsdale - USA

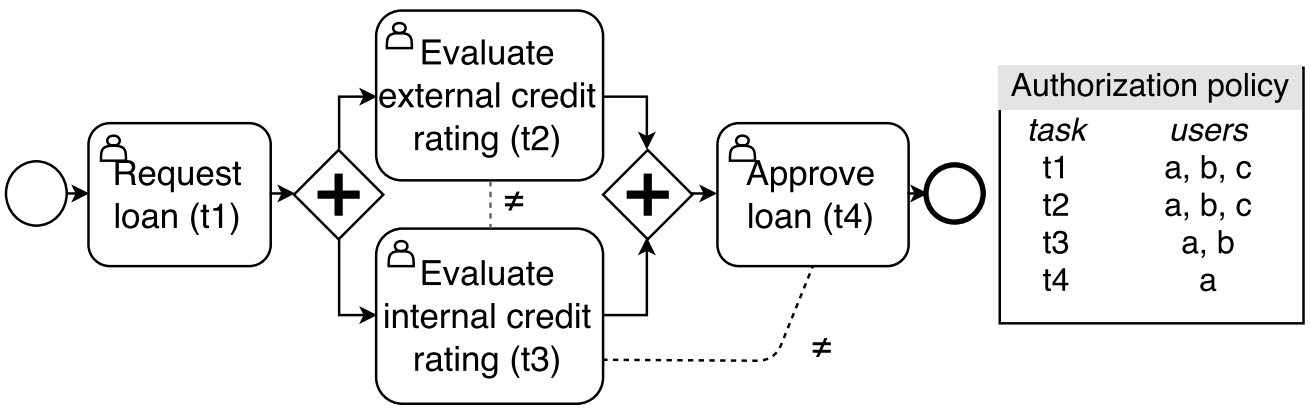
Motivation

- Many web applications are **workflow-driven**: a pre-defined sequence of tasks must be performed by users to reach a goal
  - Examples: e-commerce, healthcare, ERP (business processes)

- These applications must enforce:
  - **Control-flow integrity**: users must not skip tasks, repeat unique tasks, etc.
  - **Data-flow integrity**: values sent from the server and communicated back or relayed to third parties are not changed by the user
  - **Authorization policy**: users must only execute tasks that they are authorized to
  - **Authorization constraints**: e.g., Separation of Duties

Problems

- Policy **specification** and **enforcement** are hard
- Workflow **Satisfiability** Problem: security policies may lead to situations where a workflow cannot be completed because no user can execute a task without violating them
- Conflict between business **compliance** and business **continuity**



Satisfying trace: t1(a), t2(c), t3(b), t4(a)
Unsatisfying trace: t1(a), t2(b), t3(a), X

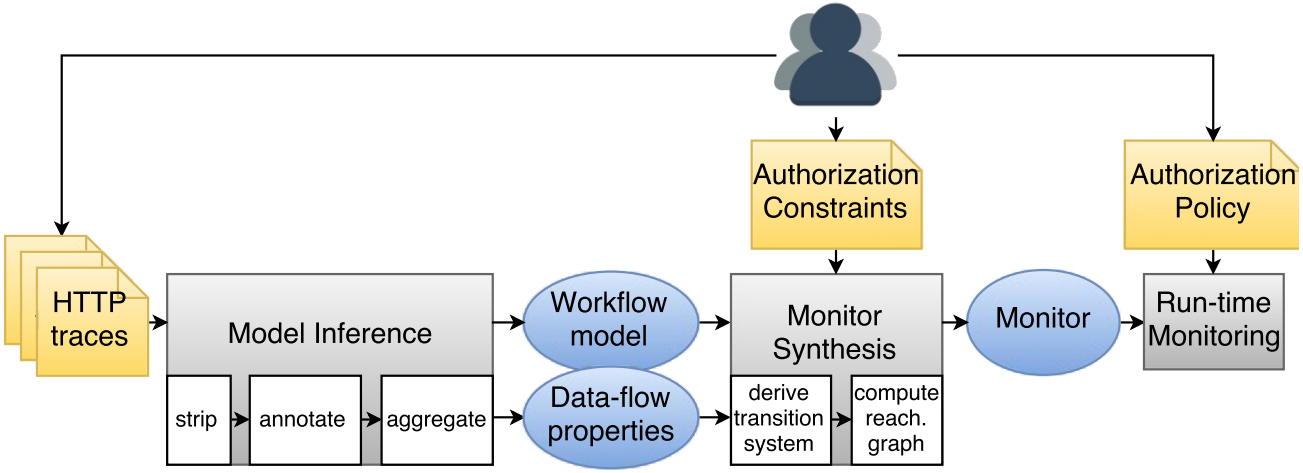
Aegis

- Aegis is a tool to **synthesize run-time monitors** that:
  - enforce security policies (combinations of control- and data-flow integrity, authorization policies and constraints)
  - grant or deny requests of users to execute tasks based on the policy and the possibility to terminate the current workflow instance
- Monitors are synthesized once per workflow (**off-line**) and support different authorization policies at **run-time** (multiple deployments of an application)

Aegis

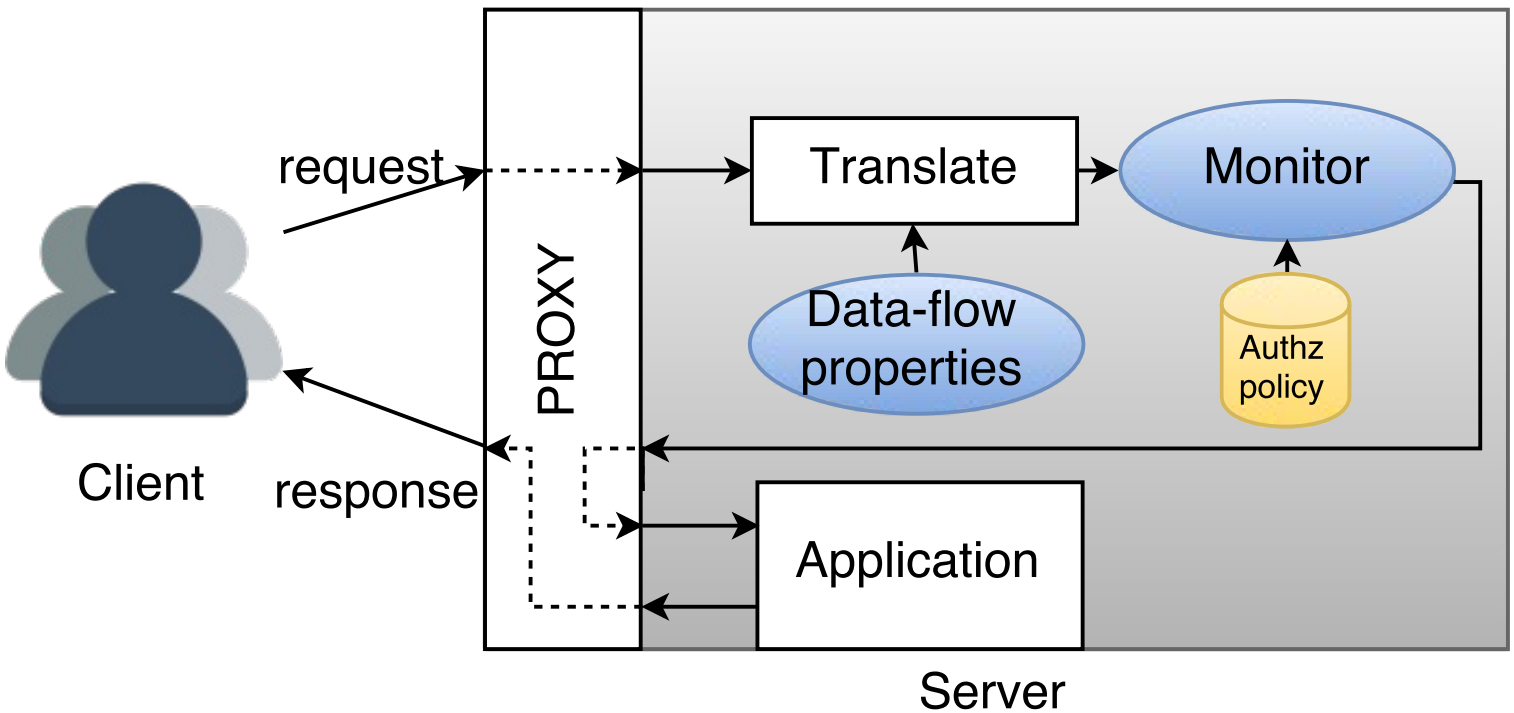
- Aegis is **black-box**
  - takes as input sets of **HTTP traces** representing user actions executed while interacting with a target web application
- Traces can be collected using test automation tools (e.g., Selenium) and must be manually edited to contain only critical tasks
- Synthesized monitor only controls those tasks given in input and ignores the rest of the application

Steps



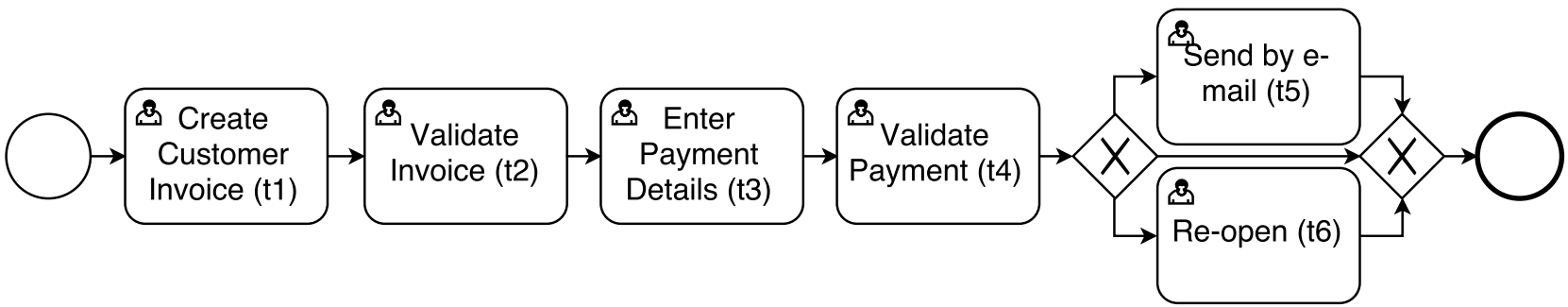
- **Model inference**: clean the traces and obtain a model with control-flow and data properties
  - **process mining** and **differential analysis**
- **Monitor synthesis**: synthesize a run-time monitor from the model (computes possible executions using **model checking**)
- **Run-time monitoring**: reverse proxy intercepts requests and translates to **SQL queries** for the monitor

Run-time


request
response
Client
PROXY
Translate
Monitor
Data-flow properties
Authz policy
Application
Server

Example 1 - Enforcing constraints

- ERP application with invoicing process (among others)
- User wants to enforce:
  - Control-flow/data-flow integrity
  - Authorization policy
  - Authorization constraints: SoD(t1,t2); SoD(t3,t4)

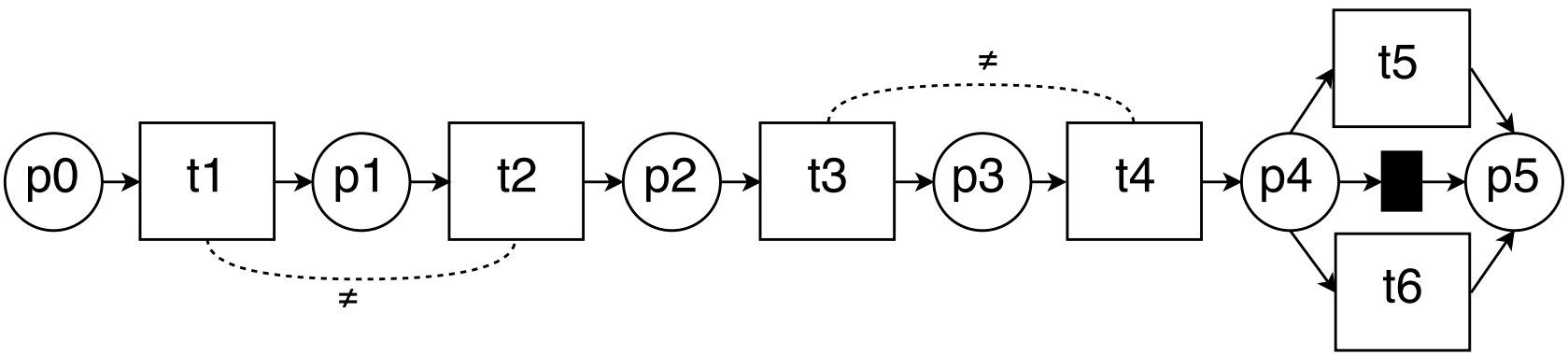


Model Inference

$\tau_1 =$ {/invoice?action=create&value=10&prod=abc, /invoice/validate?id=1,
/invoice/pay/create?id=1&value=10, /invoice/pay/validate?id=1},

$\tau_2 =$ {/invoice?action=create&value=20&prod=def, /invoice/validate?id=2,
/invoice/pay/create?id=2&value=20, /invoice/pay/validate?id=2, POST /invoice/send BODY id=2},

$\tau_3 =$ {/invoice?action=create&value=30&prod=ghi&prod2=jkl, /invoice/validate?id=3,
/invoice/pay/create?id=3&value=30, /invoice/pay/validate?id=3, /invoice/reopen?id=3}.



t_1 : /invoice?action=create&value=<<I>>&prod=<<DC>>,

t_2 : /invoice/validate?id=<<IID>>,

t_3 : /invoice/pay/create?id=<<IID>>&value=<<I>>,

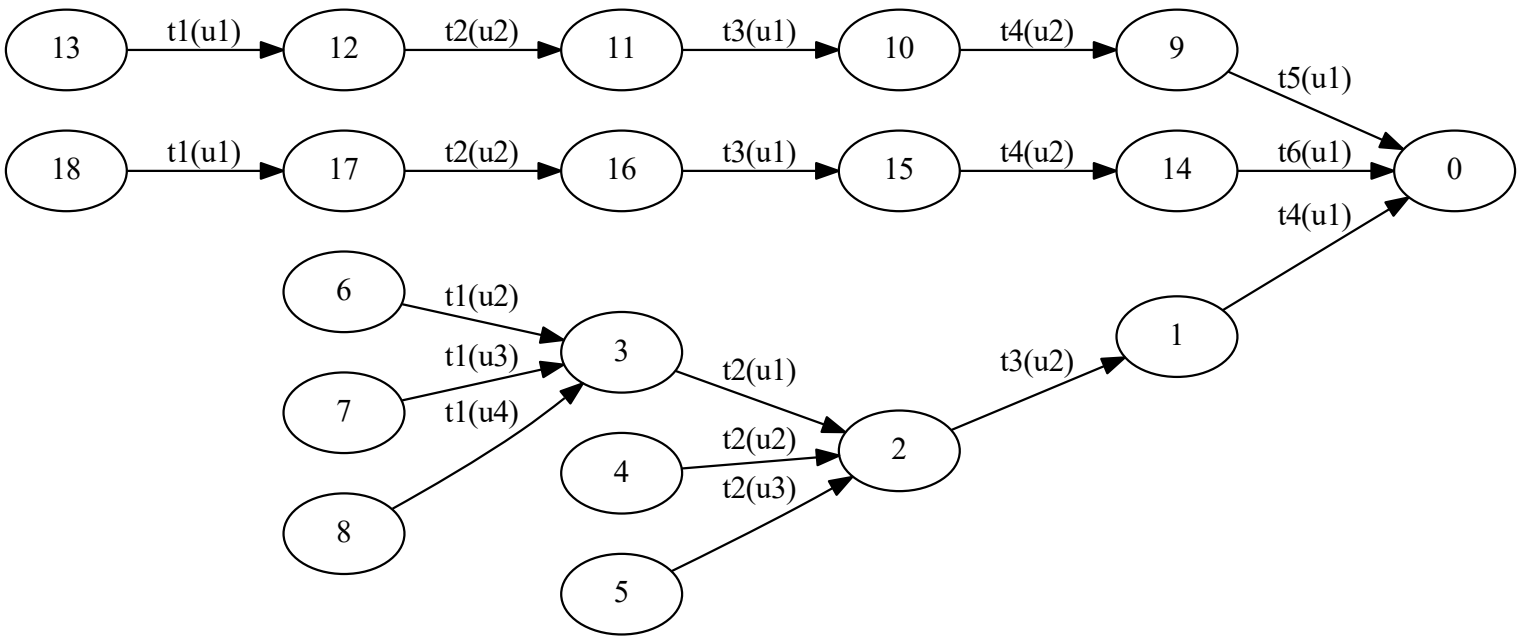
t_4 : /invoice/pay/validate?id=<<IID>>,

t_5 : POST /invoice/send BODY id=<<IID>>,

t_6 : /invoice/reopen?id=<<IID>>

Monitor synthesis

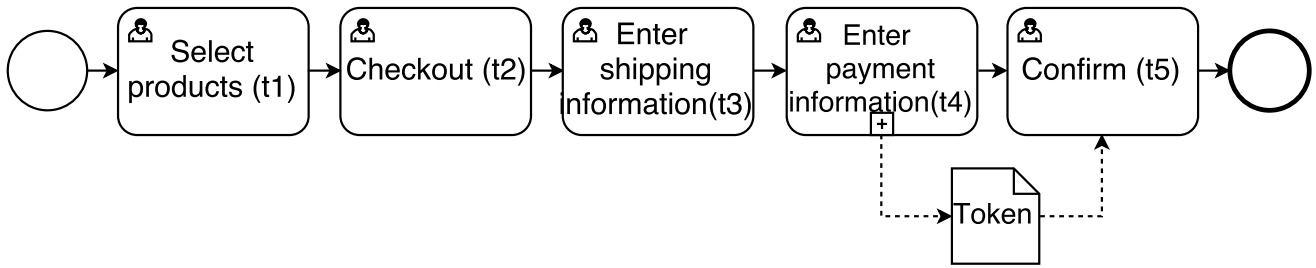
- **Transition system** derived from model
- Symbolic model checker computes **reachability graph** representing all possible executions that respect the policy
- **Formulae** labeling the nodes in the graph (states) are translated to **SQL queries**



Run-time monitoring

- Authorization policy is specified as task-user assignment
- Proxy receives a request
  `GET /invoice/validate?id=5 Cookie: sid=abcd1234`
- Identifies that it refers to task $t2$ of instance 5 of the invoice process performed by user with cookie `abcd1234` and queries the monitor
- Sends the request to the monitor and acts based on the monitor's decision
  - Request is granted if all properties are satisfied and the workflow can still terminate

Example 2 - Mitigating vulnerabilities



- No need to implement authorization policy and constraints
- Vulnerabilities: token replay, parameter tampering
- Mitigated because of the integrity enforcement

Experiments - applications

#	Application	Language	Parameters	Downloads
1	Odoo	Python	JSON	2M
2	Dolibarr	PHP	GET	850k
3	WebERP	PHP	GET	617k
4	ERPNext	Python	JSON	25k
5	OpenEMR	PHP	GET	382k
6	BikaLIMS	Python	REST	111k
7	OpenCart	PHP	GET	9M
8	TomatoCart	PHP	GET	119k
9	osCommerce	PHP	GET	80k
10	AbanteCart	PHP	GET	21k

- Automated attacks:
  - workflow and authorization violations (1-6)
  - token replay and parameter tampering (7-10)

Experiments - results

Application	**Synthesis**	Original	Query	Aegis	**Overhead**
Odoo	**21.3 s**	112 ms	6 ms	132 ms	**20 ms**
Dolibarr	**14.2 s**	93 ms	5 ms	103 ms	**10 ms**
WebERP	**20 s**	51 ms	6 ms	59 ms	**8 ms**
ERPNext	**13.3 s**	222 ms	7 ms	251 ms	**29 ms**
OpenEMR	**19.1 s**	95 ms	7 ms	112 ms	**17 ms**
BikaLIMS	**31.2 s**	306 ms	7 ms	326 ms	**20 ms**
OpenCart	**19.1 s**	65 ms	6 ms	77 ms	**12 ms**
TomatoCart	**15.8 s**	63 ms	4 ms	71 ms	**8 ms**
osCommerce	**22.2 s**	79 ms	7 ms	95 ms	**16 ms**
AbanteCart	**19.8 s**	117 ms	8 ms	127 ms	**10 ms**

Conclusion

- We have described and evaluated Aegis, a tool to enforce authorization policies and constraints, control- and data-flow integrity, and ensure the satisfiability of web applications
- Experiments show the practical viability in enforcing the desired properties and mitigating related vulnerabilities with a small performance overhead
- We intend to test Aegis in more real-world applications and to explore monitor inlining by embedding synthesized monitors into the applications

Thank you!

dossantos@fbk.eu

www.secentis.eu

UNIVERSITY
OF TRENTO