QFLan: A tool for the Quantitative Analysis of Highly Reconfigurable Systems

Andrea Vandin
Sant’Anna School of Advanced Studies Pisa, Italy
DTU Technical University of Denmark

Maurice H. ter Beek
ISTI CNR Pisa, Italy

Axel Legay
UCLouvain, Belgium

Alberto Lluch Lafuente
DTU, Denmark

Classes 21t-22t, Software Validation and Verification, Unipi, 04-05/12/2023
Class 21t 04/12/2023
[JSS22] Roberto Casaluce, Andrea Burattin, Francesca Chiaromonte, Alberto Lluch Lafuente, Andrea Vandin, White-box validation of quantitative product lines by statistical model checking and process mining [Minor revision]


Presented in [FM’18][TSE’18]
Prototypes in [FMSPLE’15][SPLC’15][ISOLA’16]

https://github.com/qflanTeam/QFLan/
Presented in [FM’18][TSE’18]
Prototypes in [FMSPLE’15][SPLC’15][ISOLA’16]

https://github.com/qflanTeam/QFLan/
QFLan

A framework for quantitative modeling and analysis of highly (re)configurable systems

Summary

QFLan is a software tool for the modeling and analysis of highly reconfigurable systems, including software product lines.

The tool offers an easy-to-use, rule-based probabilistic language to specify models with probabilistic behaviour. Quantitative constraints can be used to restrict the class of admissible configurations (or products), like (using a family of reconfigurable vending machines from here):

- machines can have a certain maximum cost,
- machines serving coffee-based beverages cannot sell tea,
- in order to serve cappuccino it is necessary to have the feature of serving also coffee,

Also it is possible to express conditions like:

- machines serving cappuccino provided with a coca dispenser can serve chocaccino.

QFLan has been combined with the distributed statistical model checker MultiVeStA to perform

https://github.com/qflanTeam/QFLan/
Feature Model
- Abstract and Concrete Features
- Cross-tree Constraints
- Quantitative Constraints

Behaviour
- Actions and Action Constraints
- Transitions
- Initial Configuration

MultiVeStA Analysis
- Analysis when a condition holds
- Analysis at varying of time

An Application to a Simple Security Scenario
- Schneier’s SafeLock Attack Tree
A simple vending machine product line

The feature model
A simple vending machine product line
The feature model: Abstract & Concrete Features

```
begin abstract features
  Machine Beverage CoffeeBased
end abstract features

begin concrete features
  Cocoa Tea Cappuccino Coffee
end concrete features

begin feature predicates
  price = \{ Cappuccino = 7, Coffee = 5, 
              Cocoa = 2, Tea = 5 \}
end feature predicates

begin feature diagram
  Machine -> {?Cocoa, Beverage}
  Beverage XOR -> {CoffeeBased, Tea}
  CoffeeBased OR -> {Cappuccino, Coffee}
end feature diagram
```
A simple vending machine product line
The feature model: Cross-tree constraints

begin abstract features
  Machine Beverage CoffeeBased
end abstract features

begin concrete features
  Cocoa Tea Cappuccino Coffee
end concrete features

begin feature diagram
  Machine -> {?Cocoa, Beverage}
  Beverage -XOR- {CoffeeBased, Tea}
  CoffeeBased -OR- {Cappuccino, Coffee}
end feature diagram

begin cross-tree constraints
  Cappuccino requires Coffee
  Tea excludes Cocoa
end cross-tree constraints

begin feature predicates
  price= { Cappuccino = 7, Coffee = 5,
           Cocoa = 2, Tea = 5 }
end feature predicates
A simple vending machine product line
The feature model: Cross-tree constraints

begin abstract features
  Machine Beverage CoffeeBased
end abstract features

begin concrete features
  Cocoa Tea Cappuccino Coffee
end concrete features

begin feature diagram
  Machine -> {?Cocoa, Beverage}
  Beverage XOR {CoffeeBased, Tea}
  CoffeeBased OR {Cappuccino, Coffee}
end feature diagram

begin cross-tree constraints
  Cappuccino requires Coffee
  Tea excludes Cocoa
end cross-tree constraints

begin feature predicates
  price= { Cappuccino = 7, Coffee = 5, 
           Cocoa = 2, Tea = 5 }
end feature predicates
A simple vending machine product line
The feature model: Cross-tree constraints

begin abstract features
  Machine Beverage CoffeeBased
end abstract features

begin concrete features
  Cocoa Tea Cappuccino Coffee
end concrete features

begin feature predicates
  price= { Cappuccino = 7, Coffee = 5, Cocoa = 2, Tea = 5 }
end feature predicates

begin feature diagram
  Machine -> {?Cocoa, Beverage}
  Beverage -XOR- {CoffeeBased,Tea}
  CoffeeBased -OR->{Cappuccino,Coffee}
end feature diagram

begin cross-tree constraints
  Cappuccino requires Coffee
  Tea excludes Cocoa
end cross-tree constraints
A simple vending machine product line
The feature model: Cross-tree constraints

begin abstract features
  Machine Beverage CoffeeBased
end abstract features

begin concrete features
  Cocoa Tea Cappuccino Coffee
end concrete features

begin feature predicates
  price= { Cappuccino = 7, Coffee = 5, Cocoa = 2, Tea = 5 }
end feature predicates

begin feature diagram
  Machine -> {?Cocoa, Beverage}
  Beverage -XOR- {CoffeeBased, Tea}
  CoffeeBased -OR- {Cappuccino, Coffee}
end feature diagram

begin cross-tree constraints
  Cappuccino requires Coffee
  Tea excludes Cocoa
end cross-tree constraints
A simple vending machine product line
The feature model: Quantitative constraints

begin abstract features
  Machine Beverage CoffeeBased
end abstract features

begin concrete features
  Cocoa Tea Cappuccino Coffee
end concrete features

begin feature diagram
  Machine -> {?Cocoa, Beverage}
  Beverage -XOR-> {CoffeeBased, Tea}
  CoffeeBased -OR-> {Cappuccino, Coffee}
end feature diagram

begin feature predicates
  price= { Cappuccino = 7, Coffee = 5, Cocoa = 2, Tea = 5 }
end feature predicates

begin quantitative constraints
  { price(Machine) <= 10 }
end quantitative constraints

begin cross-tree constraints
  Cappuccino requires Coffee
  Tea excludes Cocoa
end cross-tree constraints
Feature Model
- Abstract and Concrete Features
- Cross-tree Constraints
- Quantitative Constraints

Behaviour
- Actions and Action Constraints
- Transitions
- Initial Configuration

MultiVeStA Analysis
- Analysis when a condition holds
- Analysis at varying of time

An Application to a Simple Security Scenario
- Schneier’s SafeLock Attack Tree
A simple vending machine product line

Behaviour: actions and action constraints

begin actions
sell deploy reconfigure
chocaccino
serveCoffee serveCappuccino
serveChocaccino serveTea
end actions

begin action constraints
do(chocaccino) -> (has(Cappuccino)
    and has(Cocoa))
end action constraints
A simple vending machine product line

Behaviour: transitions

begin variables
sold = 0
deploys = 0
end variables

begin actions
sell deploy reconfigure
serveCoffee serveCappuccino
serveChocaccino serveTea
end actions

begin action constraints
do(chocaccino) -> (has(Cappuccino)
and has(Cocoa))
end action constraints

begin process dynamics
states = factory, deposit, operating, prepareCoffee, prepareCappuccino, prepareTea, prepareChocaccino

//Operating
///Coffee
operating -(Coffee,3) -> prepareCoffee,
prepareCoffee -(serveCoffee,1) -> operating,

//Cappuccino
operating -(Cappuccino,3) -> prepareCappuccino,
prepareCappuccino -(serveCappuccino,1) -> operating,

//Chocaccino
operating -(chocaccino,2) -> prepareChocaccino,
prepareChocaccino -(serveChocaccino,1) -> operating,

//Tea
operating -(Tea,3) -> prepareTea,
prepareCappuccino -(serveTea,1) -> operating,

operating -(reconfigure,1) -> deposit
end process
end processes diagram
A simple vending machine product line

Behaviour: initial configuration

begin variables
sold = 0
deploys = 0
end variables

begin actions
sell deploy reconfigure chocaccino serveCoffee serveCappuccino serveChocaccino serveTea
end actions

begin action constraints
do(chocaccino) -> (has(Cappuccino) and has(Cocoa))
end action constraints

begin init
installedFeatures = {Coffee}
initialProcesses = dynamics
end init

begin processes diagram
begin process dynamics
states = factory, deposit, operating, prepareCoffee, prepareCappuccino, prepareTea, prepareChocaccino

transitions =
//Factory
factory -(replace(Coffee,Tea,20))->factory,
factory -(install(Cocoa),10)->factory,
factory -(install(Cappuccino),10)->factory,
factory -(sell,1,{sold=1})-> deposit,

//Deposit
deposit -(install(Cappuccino),2.0)->deposit,
deposit -(uninstall(Cappuccino),2.0)->deposit,
deposit -(install(Cocoa),2.0)->deposit,
deposit -(uninstall(Cocoa),2.0)->deposit,
deposit -(deploy,2,{deploys=deploys+1})-> operating

//Coffee
operating -(Coffee,3)-> prepareCoffee,
prepareCoffee -(serveCoffee,1) -> operating,
//Cappuccino
operating -(Cappuccino,3)-> prepareCappuccino,
prepareCappuccino -(serveCappuccino,1) -> operating,
//Chocaccino
operating -(chocaccino,2)-> prepareChocaccino,
prepareChocaccino -(serveChocaccino,1) -> operating,
//Tea
operating -(Tea,3)-> prepareTea,
prepareCappuccino -(serveTea,1) -> operating,
operating -(reconfigure,1) -> deposit
end process
end processes diagram
Feature Model
- Abstract and Concrete Features
- Cross-tree Constraints
- Quantitative Constraints

Behaviour
- Actions and Action Constraints
- Transitions
- Initial Configuration

**MultiVeStA Analysis**
- Analysis when a condition holds
- Analysis at varying of time

An Application to a Simple Security Scenario
- Schneier’s SafeLock Attack Tree
A simple vending machine product line

MultiVeStA Analysis: analysis of sold machines

begin analysis
query = eval when {sold == 1.0}:
{ price(Machine) [delta=0.5],
  Coffee, Tea, Cappuccino, Cocoa }
default delta=0.05
alpha = 0.05
parallelism = 1
end analysis

begin processes diagram
begin process dynamics
states = factory, deposit, operating, prepareCoffee,
prepareCappuccino, prepareTea, prepareChocaccino
transitions =
//Factory
factory -(replace(Coffee, Tea), 20)-> factory,
factory -(install(Cocoa), 10)-> factory,
factory -(install(Cappuccino), 10)-> factory,
factory -(sell, 1, {sold=1}) -> deposit,

//Deposit
deposit -(install(Cappuccino), 2.0)-> deposit,
deposit -(uninstall(Cappuccino), 2.0)-> deposit,
deposit -(install(Cocoa), 2.0)-> deposit,
deposit -(uninstall(Cocoa), 2.0)-> deposit,
deposit -(deploy, 2, {deploys=deploys+1}) -> operating,
A simple vending machine product line
MultiVeStA Analysis: analysis of sold machines

begin analysis

query = eval when {sold == 1.0} :
{ price(Machine) [delta=0.5],
  Coffee, Tea, Cappuccino, Cocoa }

default delta=0.05
alpha = 0.05
parallelism = 1

end analysis

<table>
<thead>
<tr>
<th></th>
<th>Price</th>
<th>Coffee</th>
<th>Tea</th>
<th>Cappuccino</th>
<th>Cocoa</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.68</td>
<td>0.36</td>
<td>0.64</td>
<td>0.00</td>
<td>0.34</td>
</tr>
</tbody>
</table>
A simple vending machine product line

MultiVeStA Analysis: analysis at varying of time

begin quantitative constraints
   \{ \text{price(Machine)} \leq 15 \}
end quantitative constraints

begin init
   \text{installedFeatures} = \{ \text{Coffee} \}
   \text{initialProcesses} = \text{dynamics}
end init

begin variables
   \text{sold} = 0
   \text{deployed} = 0
end variables

begin action constraints
   \text{do(chocaccino) } \Rightarrow \text{has(Cappuccino)} \quad \text{and} \quad \text{has(Cocoa)}
end action constraints

begin analysis
query = \text{eval when \{sold = 1.0\} :}
   \{ \text{price(Machine)} [\delta = 0.5],
      \text{Coffee, Tea, Cappuccino, Cocoa} \}
default \delta = 0.05
\alpha = 0.05
\text{parallelism} = 1
end analysis

<table>
<thead>
<tr>
<th></th>
<th>Price</th>
<th>Coffee</th>
<th>Tea</th>
<th>Cappuccino</th>
<th>Cocoa</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{price(machine)} \leq 10</td>
<td>5.68</td>
<td>0.36</td>
<td>0.64</td>
<td>0.00</td>
<td>0.34</td>
</tr>
<tr>
<td>\text{price(machine)} \leq 15</td>
<td>9.07</td>
<td>0.49</td>
<td>0.51</td>
<td>0.45</td>
<td>0.44</td>
</tr>
</tbody>
</table>

MultiVeStA analysis of VendingMachine.gqian
SMC of queryVendingMachine.quatex. Cl=(0.05, 0.5, 0.05, 0.05, 0.05, 0.05)
Feature Model
- Abstract and Concrete Features
- Cross-tree Constraints
- Quantitative Constraints

Behaviour
- Actions and Action Constraints
- Transitions
- Initial Configuration

MultiVeStA Analysis
- Analysis when a condition holds
- Analysis at varying of time

An Application to a Simple Security Scenario
- Schneier’s SafeLock Attack Tree
Schneier’s SafeLock Attack Tree
An application of QFLan to security

Schneier’s simple attack tree
www.schneier.com/academic/archives/1999/12/attack_trees.html

A feature model version of the attack tree
[TSE’18]

begin quantitative constraints
//Restrict to attacks that cost less than 100
{ cost(Root) <= 100 }

end quantitative constraints
Schneier’s SafeLock Attack Tree
An application of QFLan to security

**Probabilities of successful attacks**

- Powerful attacker
- Failing attacker 20
- Failing attacker 10

**Costs of successful attacks**

- Cost powerful attacker
- Cost failing attacker 20
- Cost failing attacker 10
- Cumul_cost failing attacker 20
- Cumul_cost failing attacker 10

begin quantitative constraints
//Restrict to attacks that cost less than 100
{ cost(Root) <= 100 }
//Attacks can fail. Attacks attempts cost.
//Restrict to attackers with a maximum budget.
{ accumulated_cost <= 10}
//{ accumulated_cost <= 20}
end quantitative constraints
Extend semantics with notion of time
For the analysis of time-related properties

Continue investigating applicability to security domain

Adapt QFLan to attack trees domain

Synthesis of constraints
We had to relax the constraint “price(Machine) <= 10”
Can we synthesize the ‘right’ constraints automatically?
QUESTIONS?

https://github.com/qflanTeam/QFLan/
A Software Engineering Approach to Quantitative Security Risk Modeling and Analysis using QFLan

Andrea Vandin
Sant’Anna School of Advanced Studies Pisa, Italy
DTU Technical University of Denmark

Maurice H. ter Beek
ISTI CNR Pisa, Italy

Axel Legay
UCLouvain, Belgium

Alberto Lluch Lafuente
DTU, Denmark
From QFLan to RisQFLan
- QFLan’s Limitations for Risk Modeling and Analysis
- A Bank robbery scenario in RisQFLan
- How did we go from QFLan to RisQFLan?

Conclusions
From QFLan to RisQFLan
- QFLan’s Limitations for Risk Modeling and Analysis
- A Bank robbery scenario in RisQFLan
- How did we go from QFLan to RisQFLan?

Conclusions
Not entirely direct encoding of the scenario
- The extra root node, the extra states to model failures, etc

We need different types of nodes
- Attack, defense, countermeasure

We need richer constructs for building the tree diagram
- QFLan has: or, requires, excludes
- Missing common constructs: and, or, out, and, activates, inhibits

Attack attempts might fail
- The ‘install’ of an attack node might ‘fail’. Failures should be first-class citizens.

There is no ‘absolute security’
- Qualitative constraints like ‘excludes’ or ‘requires’ are too strong
- Often, failure probabilities are ‘scaled’ and not zeroed by defense mechanisms

Exact analysis might be necessary in some scenarios
- Complement MultiVeStA Statistical MC by PRISM/STORM exact Probabilistic MC
From QFLan to RisQFLan
- QFLan’s Limitations for Risk Modeling and Analysis
- A Bank robbery scenario in RisQFLan
- How did we go from QFLan to RisQFLan?

Conclusions
A Bank Robbery Scenario in RisQFLan

A screenshot of RisQFLan
A Bank Robbery Scenario in RisQFLan

Attack-defense tree

Desired scenario

Modeled Scenario

bit.ly/RisQFLan
A Bank Robbery Scenario in RisQFLan

Behaviour

Attack-defense tree

bit.ly/RisQFLan
A Bank Robbery Scenario in RisQFLan
Analysis: SMC with MultiVeStA

Behaviour

Statistical SMC Analysis

Attack-defense tree

bit.ly/RisQFLan
A Bank Robbery Scenario in RisQFLan
Analysis: PMC with PRISM/STORM

INFINITE STATE SPACE

begin exportDTMC
file = "RobBank.prism"
label with "Succeeded"
when has(RobBank)
end exportDTMC

BIT.LY/RISOQFLAN

bit.ly/RisQFLan

INFINITE STATE SPACE
From QFLan to RisQFLan
- QFLan’s Limitations for Risk Modeling and Analysis
- A Bank robbery scenario in RisQFLan
- How did we go from QFLan to RisQFLan?

Conclusions
From QFLan to RisQFLan
Generalizing the QFLan approach

QFLan Architecture
[FM’18][TSE’18]

GUI
- QFLan Editor
- MultiQuaTex Editor
- Views

CORE
- Built-in Constraint Solver
- Probabilistic Simulator
- SMC MultiVeStA

Generalized QFLan Architecture
[Draft’20]

GUI layer
- XTEXT grammar for DSL
  - DSL Editor
  - Views

CORE layer
- Interpreter
  - Constraint Solver
- Rated Transition System Generator
- DTMC Generator
  - Probabilistic Simulator
  - DTMC Exporter
- SMC MultiVeStA

Existing domain-independent components
Automatically generated domain-independent components
Domain-specific components necessary to instantiate the architecture in a new domain
From QFLan to RisQFLan
- QFLan’s Limitations for Risk Modeling and Analysis
- A Bank robbery scenario in RisQFLan
- How did we go from QFLan to RisQFLan?

Conclusions
RisQFLan: A Software Engineering Approach to Quantitative Security Risk Modeling and Analysis
- Obtained via a DSL-independent generalization of QFLan + its instantiation to security domain
- Both QFLan and RisQFLan are open-source projects

Main improvements
- Modeling: Richer constructs specific to the security domain
- Analysis: New support for exact PMC engines (PRISM, STORM) complementing existing SMC engine (MultiVeStA)

Related work
- Due to the generality and versatility of our framework, we succeeded in incorporating many features from proposals in the literature
  - E.g.: o-and, noticeability, countermeasures (see validation in [Draft20])
  - The explicit probabilistic attacker behaviour is somehow new, as
  - Specific dynamic threat profiles is a related feature. But it is often unsupported
    - Supported only recently by a few approaches in a limited way
- RisQFLan allows for nodes with multiple parents
  - This is convenient: allows to keep models small. But it is often unsupported
Attributes of leaf nodes are propagated up the tree via sum.
- Other approaches, e.g. SecurITree, allow for attribute-specific propagation functions (e.g., min, max, product)

Allow for non-deterministic (unspecified) aspects in RisQFLan
- Use external tools (Uppaal Stratego?) to synthetize the attacker with highest success probability/the defense with best impact

Even though the design of RisQFLan is inspired by the most common features from the literature, we want to:
- Better understand relation of RisQFLan with the huge related work

Validate RisQFLan scalability and expressiveness considering realistic scenarios
- E.g. the Attack Tree Benchmarks www7.in.tum.de/~kraemerj/upload/index.php
The great expressive power coming from the quantitative constraints, etc, might make it difficult to understand what a model does.

SMC and PMC give only limited information on what the model does.
- We get black-box numbers
- Are these numbers due to the nature of the studied system?
- Are these numbers due to bugs?

Can we exploit novel techniques to explain SMC?
STATISTICAL MODEL CHECKING MEETS PROCESS MINING
WHITE-BOX VALIDATION OF SIMULATION MODELS

Andrea Vandin

Sant’Anna
School of Advanced Studies – Pisa
Institute of Economics

Department of Excellence
2018 - 2022

EMbeDS
Economics and Management
in the era of Data Science

DTU
Danmarks Tekniske Universitet

Classes 21t-22t, Software Validation and Verification, Unipi, 04-05/12/2023
Class 21t 04/12/2023
WHITE-BOX VALIDATION OF PRODUCT LINES AND THREAT MODELS BY STATISTICAL MODEL CHECKING AND PROCESS MINING

Roberto Casaluce, Andrea Burattin, Francesca Chiaromonte, Alberto Lluch Lafuente, Andrea Vandin

Recently published at DEC2H
zenodo.org/record/6623377

Journal extension at JSS: 2nd round of review
zenodo.org/record/6623377

‘A SIMPLIFIED OVERVIEW’
WHAT IS PROCESS MINING?

• A family of techniques linking data science and process management to support the analysis of processes
• Aims at turning event logs into insights and actions
• Uses data to discover a process model
  • It observes events recorded by enterprise systems

WHAT IS PROCESS MINING?

The reference process...

The traces, or process logs

- The actual process...

Picture by Koen Olsthoorn
WHAT IS PROCESS MINING?

With Process Mining we can discover that
• The actual process is different from the expected one

Idea
Can PM explain the SMC results?

The reference process…

The traces, or process logs
  • The actual process…

Picture by Koen Olsthoorn
OUR METHODOLOGY 4 WHITE-BOX VALIDATION
OUR METHODOLOGY 4 WHITE-BOX VALIDATION
OUR METHODOLOGY 4 WHITE-BOX VALIDATION

Model creation
SMC analysis (MultiVeStA)

Numerical results and single counterexample

Black-box evaluation of numerical results

State-of-the-art life-cycle of SMC-analysed simulation

Informed guess driven by numerical results
OUR METHODOLOGY 4 WHITE-BOX VALIDATION

Model creation and update → SMC analysis (MultiVeStA) → Event logs → Numerical results and single counterexample → White-box process mining analysis → Behavioral evaluation of process mining results → Black-box evaluation of numerical results → Informed guess driven by numerical results

Unexpected behavior discovered with process mining and numerical results

Our novel SMC- and PM-guided methodology for white-box model validation

Model creation and update → SMC analysis (MultiVeStA) → Black-box evaluation of numerical results → Informed guess driven by numerical results

Usual life-cycle of SMC-analysed simulation models
1. Model creation
   - Design model

2. Logs generation
   - SMC analysis
   - Event logs for process mining

3. Logs pre-processing
   - Graph. rep. of proced. part of QFlan model

4. Process mining
   - White-box testing with process mining
   - Mined PM model
   - Conversion into procedural part of QFlan model

5. Automatic diff
   - Behaviroal evaluation of PM results
   - Diff model

Unexpected behavior discovered with process mining and numerical results
APPLICATION ON SIMPLE THREAT ANALYSIS EXAMPLE

Computers & Security
Volume 109, October 2021, 102381

Quantitative Security Risk Modeling and Analysis with RisQFLan

Maurice H. ter Beek, Axel Legay, Alberto Lluch Lafuente, Andrea Vandin

Andrea Vandin
APPLICATION ON SIMPLE THREAT ANALYSIS EXAMPLE

RobBank

OpenVault

BlowUp

Attributes
Cost = 90.0
Detection Rate = 0.0

Defense Effectiveness
ALL : RobBank = 0.3
APPLICATION ON SIMPLE THREAT ANALYSIS EXAMPLE

RobBank

OpenVault

BlowUp

TryOpenVault

chooseOV,2

add(OpenVault),2

fail(OpenVault),1

chooseBU,4

add(BlowUp),2

fail(BlowUp),10000

add(RobBank),3

fail(RobBank),1

TryBlowUp

Complete

Cost = 90.0

Detection Rate = 0.0

LockDown

Defense Effectiveness

ALL : RobBank = 0.3
APPLICATION ON SIMPLE THREAT ANALYSIS EXAMPLE

**Attributes**

- **Cost** = 90.0
- **Detection Rate** = 0.0

**Defense Effectiveness**

- **ALL**:
  - **RobBank** = 0.3

**TryOpenVault**

- **chooseOV,2**
- **add(OpenVault),2**
- **fail(OpenVault),1**

**TryBlowUp**

- **chooseBU,4**
- **add(BlowUp),2**
- **fail(BlowUp),10000**

**Complete**

- **add(RobBank),3**
- **fail(RobBank),1**
APPLICATION ON SIMPLE THREAT ANALYSIS EXAMPLE

RobBank

OpenVault

BlowUp

Attributes

Cost = 90.0

Detection Rate = 0.0

LockDown

Defense Effectiveness

ALL : RobBank = 0.3

TryOpenVault

chooseOV,2

add(OpenVault),2

fail(OpenVault),1

TryBlowUp

chooseBU,4

add(BlowUp),2

fail(BlowUp),10000

Complete

add(RobBank),3

fail(RobBank),1

github.com/RisQFLan/RisQFLan/wiki
APPLICATION ON SIMPLE THREAT ANALYSIS EXAMPLE

RobBank

OpenVault

 BlowUp

Attributes

Cost = 90.0

Detection Rate = 0.0

Defense Effectiveness

ALL : RobBank = 0.3

TryOpenVault

chooseOV,2 add(OpenVault),2 fail(OpenVault),1

TryBlowUp

chooseBU,4 add(BlowUp),2 fail(BlowUp),10000

Complete

add(RobBank),3 fail(RobBank),1
ANALYSIS OF ORIGINAL MODEL

TryOpenVault
chooseOV,2 
add(OpenVault),2
fail(OpenVault),1

ChooseBlowUp
chooseBU,4
add(BlowUp),2
fail(BlowUp),10000

Complete
add(RobBank),3
fail(RobBank),1

RobBank

OpenVault

BlowUp
Attributes
Cost = 90.0
Detection Rate = 0.0

Cost = 90.0
ANALYSIS OF ORIGINAL MODEL

Probability of successful bank robbery!? 0.17

Why?
1) My defences are good
2) The attacker is bad
3) Or my model is bad!?
ANALYSIS OF ORIGINAL MODEL

Probability of successful bank robbery!? 0.17

Why?
1) My defences are good
2) The attacker is bad
3) Or my model is bad!?

We set alpha=0.1, delta=0.1
MultiVeStA performs 240 simulations
• We generate logs for each simulation
• We ask Fluxicon Disco mine these logs
• Can we spot any issue in the model?
ANALYSIS OF ORIGINAL MODEL

Probability of successful bank robbery!?

0.17

Why?

1) My defences are good
2) The attacker is bad
3) Or my model is bad!?

We set alpha=0.1, delta=0.1

MultiVeStA performs **240 simulations**

- We generate logs for each simulation
- We ask Fluxicon Disco mine these logs
- Can we spot any issue in the model?

---

Probability of successful bank robbery!?

0.17

Why?

1) My defences are **good**
2) The attacker is **bad**
3) **Or my model is bad**!?

We set alpha=0.1, delta=0.1

MultiVeStA performs **240 simulations**

- We generate logs for each simulation
- We ask Fluxicon Disco mine these logs
- Can we spot any issue in the model?
ANALYSIS OF ORIGINAL MODEL

Probability of successful bank robbery!? 0.17

Why?
1) My defences are good
2) The attacker is bad
3) Or my model is bad!?

We set alpha=0.1, delta=0.1
MultiVeStA performs 240 simulations
- We generate logs for each simulation
- We ask Fluxicon Disco mine these logs
- Can we spot any issue in the model?

RobBank

BlowUp
Attributes
Cost = 90.0
Detection Rate = 0.0

OpenVault

Start-reset

TryOpenVault-chooseOV

TryBlowUp-chooseBU

Start-fail(BlowUp)

Start-succ(OpenVault)

Complete-fail(RobBank)

Complete-succ(RobBank)

ChooseOV, 2
Add(OpenVault), 2
Fail(OpenVault), 1
ChooseBU, 4
Add(BlowUp), 2
Fail(BlowUp), 10000
Add(RobBank), 3
Fail(RobBank), 1
FIRST REFINEMENT: PARSIMONIOUS ATTACKER

1 // We add the !allowed(RobBank)
2 Start - (chooseBU, 4, !allowed(RobBank)) \rightarrow TryBlowUp, TryBlowUp - (\text{succ}(\text{BlowUp}), 2) \rightarrow \text{Start},
3 TryBlowUp - (\text{fail}(\text{BlowUp}), 10000) \rightarrow \text{Start}

TryOpenVault
chooseOV, 2 add(OpenVault), 2 fail(OpenVault), 1

TryBlowUp
chooseBU, 4 add(BlowUp), 2 fail(BlowUp), 10000

Complete
add(RobBank), 3 fail(RobBank), 1

RobBank

Open Vault
BlowUp
Attributes
Cost = 90.0
Detection Rate = 0.0

LockDown
Defense Effectiveness
ALL : RobBank = 0.3
Probability of successful bank robbery!? 0.17 0.31

Why?
1) My defences are good
2) The attacker is bad
3) Or my model is bad!?

We set alpha=0.1, delta=0.1
MultiVeStA performs 240 simulations
• We generate logs for each simulation
• We ask Fluxicon Disco mine these logs
• Can we spot any issue in the model?
Probability of successful bank robbery!? 0.17

Why?
1) My defences are good
2) The attacker is bad
3) Or my model is bad!? 

We set alpha=0.1, delta=0.1

MultiVeStA performs 240 simulations
- We generate logs for each simulation
- We ask Fluxicon Disco mine these logs
- Can we spot any issue in the model?
Probability of successful bank robbery!? \(0.17\) Why?

1) My defences are *good*
2) The attacker is *bad*
3) **Or my model is bad**!? 

We set \(\alpha = 0.1\), \(\delta = 0.1\)

MultiVeStA performs **240 simulations**

- We generate logs for each simulation
- We ask **Fluxicon Disco** mine these logs
- Can we spot any issue in the model?

**FIRST REFINEMENT: PARSIMONIOUS ATTACKER – ANALYSIS**

![Diagram of a three-state model with transitions and labels.](image-url)
Probability of successful bank robbery!?

Why?

1) My defences are good
2) The attacker is bad
3) Or my model is bad!

FIRST REFINEMENT: PARSIMONIOUS ATTACKER - ANALYSIS

We set \( \alpha = 0.1 \), \( \delta = 0.1 \)

MultiVeStA performs 240 simulations

‣ We generate logs for each simulation
‣ We ask Fluxicon Disco mine these logs
‣ Can we spot any issue in the model?

TryOpenVault
chooseOV, 2
add(OpenVault), 2
fail(OpenVault), 1
chooseBU, 4
add(BlowUp), 2
fail(BlowUp), 10000
add(RobBank), 3
fail(RobBank), 1

TryBlowUp
chooseBU, 3

Start-reset
240

TryOpenVault
chooseOV, 144

Start-fail(OpenVault)
42

Start-succ(OpenVault)
102

TryBlowUp
chooseBU, 318

Start-fail(BlowUp)
180

Complete-succ(RobBank)
102

Complete-fail(RobBank)
29

Start-fail(RobVault)
73

Complete-fail(RobBank)
29

Complete-succ(RobBank)
73

Start-fail(RobVault)
42

TryOpenVault
161

TryBlowUp
180

Start-reset
240

ATTRIBUTES
RobBank
Cost = 90.0
Detection Rate = 0.0

Open Vault
Cost = 90.0

LockDown
Defense Effectiveness
ALL : RobBank = 0.3
begin attributes
  Cost = {BlowUp = 90, OpenVault = 0}
end attributes

begin quantitative constraints
  { value(Cost) <= 100 } 
end quantitative constraints

begin actions
  chooseOV
  chooseBU
  goBack
end actions

// Strategy where the attacker tries to blow up the vault
Start - (chooseBU, 4, !allowed(RobBank)) -> TryBlowUp,
TryBlowUp - (succe(BlowUp), 2) -> Start,
TryBlowUp - (fail(BlowUp), 10000) -> Start,
TryBlowUp - (goBack, 0.00001) -> Start,
Probability of successful bank robbery!?

We set alpha=0.1, delta=0.1

MultiVeStA performs **240 simulations**

- We generate logs for each simulation
- We ask **Fluxicon Disco** mine these logs
- Can we spot **any issue in the model?**

![Diagram showing the refinement process](image-url)
PM MEETS SMC: CONCLUSIONS & FUTURE WORKS

- We proposed a novel methodology for validating and enhancing simulation models to make them more reliable
  - We obtained: **SMC- and PM-guided white-box behavioral model validation and enhancement**

- **Future works**
  - More realistic models, from more domains (e.g., ABM from social sciences)
  - Conformance checking might help our white-box analysis
  - Currently, we use **PM after SMC**:
    - **Using PM during SMC**: streaming PM might help improving SMC analysis
    - **Using PM before SMC**: discovery algorithms might be applied to real data to
      - synthesize attack-defense trees and/or attacker behaviors
      - or parts of simulation models in general
THANK YOU FOR YOUR ATTENTION!

QUESTIONS?
FEEDBACK?

Andrea Vandin
andrea.vandin@santannapisa.it
www.santannapisa.it/en/andrea-vandin