Modelling and Analysing Resilience as a Security Issue within UML

Ricardo J. Rodríguez, José Merseguer and Simona Bernardi
{rjrodriguez, jmerse}@unizar.es, bernardi@di.unito.it

Universidad de Zaragoza
Zaragoza, Spain

Università di Torino
Torino, Italy

15th September 2010

SERENE’10: 2nd International Workshop on Software Engineering for Resilient Systems
Introduction

MARTE-DAM background

Security Analysis and Modeling profile (SecAM)

Advanced firewall running example

Obtaining a formal model

Experiments and discussion of the results

Related work and conclusions
Introduction (I)

- Security requirement specification as a part of system modeling
- **Broad and heterogeneous** field (hardware issues, coding bugs...)
- Security requirements as non-functional properties (NFPs)
- Exploit the UML profiling capabilities to support their specification

**UML**: well-known solution and comprehensive modelling language

- Tailored for **specific purposes**: profiling
- The OMG standard MARTE profile
  - Performance and schedulability analysis for RT and embedded systems
- The Dependability and Analysis Modelling (DAM) profile
  - MARTE specialization to support dependability NFPs

**MARTE + DAM**: performance and/on dependability requirements

→ missing security aspects
Introduction

Relation between **dependability-security**

- Security specification can be included in the MARTE-DAM framework
- MARTE-DAM: stereotypes and tagged to express NFPs
  - Attached to those UML model elements they affect
- **Security Analysis and Modelling (SecAM) profile**
  - Allow the expression of security NFPs
MARTE: Modelling and Analysis of RT Embedded systems

- **UML lightweight extension**
- Provides support for **schedulability and performance analysis**
- **NFPs with VSL (Value Specification Language) syntax**
- Design model element extending its semantic

---

**MARTE-DAM**

- DAM stereotypes specialise MARTE stereotypes
- MARTE NFP types
  - **value**
  - **expr (VSL expression)**
  - **source (req, est, assm)**
SecAM profile (I): Resilience package (1)

Domain model definition
- Comprehensive modelling of security issues
- Domain model for each relevant security aspects
  - e.g., confidentiality, resilience or integrity
- In this work: Resilience package

Threats
- From dependability:
  - Fault → Error → Failure
- From security:
  - Attack → Vulnerability → Intrusion
- AVI as a refinement of FEF
SecAM profile (1): Resilience package (2)

- **Fault** class from DAM::Threats:
  - extension with new attributes
- **OCL** to restrict source of errors

---

**DAM::Threats**

- Fault
  - occurrencePhase : PhaseOfCreation
  - boundary : Boundary
  - objective : Objective
  - causedBy : CausedBy
  - intent : Intent
  - capability : Capability

- Error
  - cause
  - effect

- Failure
  - cause
  - effect

**DAM::DAM_Library:Basic_DA_Types::Enumeration_Types**

- **<<enumeration>>**
  - PhaseOfCreation
    - development
    - operational
  - Boundary
    - internal
    - external
  - Objective
    - malicious
    - non-malicious
  - Capability
    - accidental
    - incompetence
  - Objective
    - deliberate
    - non-deliberate
  - CausedBy
    - natural
    - human-made
  - StepKind
    - error
    - failure
    - hazard
    - reallocation
    - replacement
    - vulnerable
    - intrusion
SecAM profile (II): building the profile (1)
SecAM profile (II): building the profile (2)

Figure: SecAM UML extensions
Example (I): system physical view and class diagram

Figure: System physical view

Figure: Class diagram

- How to use SecAM from a modeler viewpoint
- Advanced firewall
  - Exposed to attacks → vulnerable
  - Attend messages from WAN and forwarded them to LAN
  - Critical information systems (e.g. MAFTIA, CRUTIAL, OASIS)
- Includes a monitor
  - Tamper-proof embedded system → invulnerable
  - Its mission: to check firewall processes and to clean up those hung
Example (II): UML state-charts (1)

Figure: Monitor state-chart diagram.
Example (II): UML state-charts (2)

```
<gaAnalysisContext>
{contextParams={in$nProcesses, in$netLoad, in$success, in$attack, in$TProcess, out$crash}}
```

```
<gaWorkloadEvent>
{pattern=(open=(interArrivalTime=(exp($netLoad, ms))))}
```

```
<secaAttackGenerator>
{attack=$1}
```

```
<secaStep>
{hostDemand=
  (value=$TProcess, unit=ms, statQ=mean, source=asm);
  kind=vulnerable;
  vulnerability=$2}
```

```
<gaWorkloadGenerator>
{pop=$nProcesses}
```

```
create()
```

```
processMessage()
```

```
 attendsMessage()
```

```
$1=(occurrenceProb=
  (value=$attack, source=asm);
  type=active)

$2=(degree=high)
```

```
<gaStep>
{prob=(1 - $success*$attack)}
```

```
<secaStep>
{kind=intrusion;
  intrusion=
    (successProb=
      (value=$success, source=asm);
      cause=$1;
      origin=$2)}
```

```
 Crash
```

```
 processMessage()
```

```
destroy()
```

```
$1=(occurrenceProb=
  (value=$crash, unit=ms, statQ=mean, source=est)}
```

```
<gaStep>
{hostDemand=
  (value=$crash, unit=ms, statQ=mean, source=est)}
```

Figure: Process state-chart diagram.
Obtaining a formal model (I): Conversion of UML-SC

- Translation proposed by Merseguer et al. (*WODES’02*)
- Given for performance analysis purposes
- ArgoSPE tool: UML-SC annotated with SPT (precursor of MARTE)
- General ideas:
  - SC simple state $\rightarrow$ PN place
  - Entry and exit actions $\rightarrow$ immediate transitions
  - Do-activity actions $\rightarrow$ timed transitions
  - Stochastic resolution of transition conflicts
- Communication via events $\rightarrow$ PN places modelling event mailboxes

- Working out the PN to incorporate DAM and SecAM annotations
- Open workload: manually produced
- Simplified the subnets $\rightarrow$ gaining readability
Obtaining a formal model (II): Obtained DSPN

<table>
<thead>
<tr>
<th>Place</th>
<th>Initial marking</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P4</td>
<td>Idle</td>
<td>$n_{Processes}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transition</th>
<th>Parameter (type)</th>
<th>Value(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>NetworkLoad</td>
<td>$1 / \text{netload}$ (rate)</td>
</tr>
<tr>
<td>T3</td>
<td>processMessage</td>
<td>$1 / \text{Tprocess}$ (rate)</td>
</tr>
<tr>
<td>T8</td>
<td>TimeOut</td>
<td>$T_{Odelay}$ (delay)</td>
</tr>
<tr>
<td>t4</td>
<td>Intrusion</td>
<td>attack $\cdot$ success (weight)</td>
</tr>
<tr>
<td>t5</td>
<td>NonIntrusion</td>
<td>$1 - \text{attack} \cdot \text{success}$ (weight)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>attack</td>
<td>$[0.01 \ldots 0.5]$</td>
</tr>
<tr>
<td>success</td>
<td>$[0.01 \ldots 0.5]$</td>
</tr>
</tbody>
</table>
Description of the experiments

Availability

- At DSPN model level:

\[
\frac{MTTF}{MTTF + MTTDI} = 1 - \frac{E[P_5|Crash]}{N}
\]  

- **MTTF**: Mean Time To Failure
- **MTTDI**: Mean Time To Detect an Intrusion
- **E[P_i]**: mean number of tokens in place P_i
- **P_5|Crash**: unavailable state of the process

- Under **different assumptions**:
  - Three types of **network loads**: low, high, very high (0.01, 0.05, 0.1/ms)
  - Two types of **time-out durations**: short, long (1, 100 ms)
  - **Probabilities of attacks and successful attacks** from 1% up to 50%
Results (I): under low workload

(a) short time-out

(b) long time-out
Results (II): under high workload

(a) short time-out

(b) long time-out
Results (III): under very high workload

(a) short time-out

(b) long time-out
Discussion

Availability

- **Inverse proportion** to probability of attacks and of successful attacks
- Decreasing factor: sensitive to the network workload and monitor time-out assumptions
  - Higher for higher workloads and for longer time-out duration (e.g., 0.021% in case of low network workload and short time-out duration, 20.9% when very high network workload and long time-out duration)
- Incoming messages are potential attack carriers → frequency of attacks increases from low to very high network workload → higher availability decreasing factor
- Short time-out duration → promptly detection → higher availability
- Isolated hills close to 100% (low workload, short time-out)
  - Due to simulation accuracy (their height is lower than 0.01%)
- False alarms (i.e., time-out expires and no process is crashed)
  - Do not provoke side effects in the system
Related work and conclusions (I)

**Related work**

- **SecureUML** (*T. Lodderstedt et al.*)
  - Just focused on annotating static UML design models

- **UMLsec** (*J. Jürjens*)
  - Not worry on influence on the throughput of the system

Both approaches focus on the design phase and allow model-checking

- **Other work close** (*D. C. Petriu et al.*)
  - Not focussed on giving a unified framework

- **Dependability and SPNs**
  - **A. E. Rugina et al.**
    - Exclusively for the dependability field
    - Very bound to AADL (Architecture Analysis & Design Language)
  - Several works of **Bondavalli et al.**
    - Dependability attributes in early design phases of the system
    - Construct a Timed PN using graph transformation techniques in structural UML diagrams
Conclusions

- Proposal profile \( \subseteq \) MARTE-DAM profile
- Analysis of relevant dependability-security aspects
- Considering the system performance characteristics
  - e.g., to measure the real impact of introducing more security layers

Future work

- **Tools** supporting the SecAM approach
  - Reuse of existing tools for UML and MARTE
- Effort focused on the security analysis on top of existing tool sets
- Extend SecAM adding more security fields to its domain
  - **Easy fit**: SecAM-MARTE-DAM fit already done
- 

Modelling and Analysing Resilience as a Security Issue within UML

Ricardo J. Rodríguez, José Merseguer and Simona Bernardi
{rjrodriguez, jmerse}@unizar.es, bernardi@di.unito.it

Universidad de Zaragoza
Zaragoza, Spain

Università di Torino
Torino, Italy

15th September 2010

SERENE’10: 2nd International Workshop on Software Engineering for Resilient Systems