Formal Development of Critical Systems with UML: Methods and Tools

Jan Jürjens
Department of Computer Science
TU Munich
juerjens@in.tum.de
http://www.umlsec.org

Critical Software Engineering

“Penetrate-and-patch” (aka “banana strategy”):
• unreliable
• disruptive
Formal methods viewed in industry as expensive.
• training people
• constructing formal specifications.

Dependability by Design

Increase dependability with bounded investment in time, costs:
• Weave in dependability aspects/concerns into artefacts arising in industrial development and use of dependable systems (UML models, source code, configuration data).
• Tool-supported, theoretically sound, efficient automated synthesis and analysis.

Model-based Dependability

• Weave in dependability aspects into UML models.
• Generate code (or tests) from models.
• Generate models from evolving or legacy code. Here: take security.

UMLsec: Goals

Extension for secure systems development.
• evaluate UML specifications for weaknesses in design
• encapsulate established rules of prudent secure engineering as checklist
• make available to developers not specialized in secure systems
• consider security requirements from early design phases, in system context
• make certification cost-effective

UMLsec: How

Recurring security requirements, adversary scenarios, concepts offered as stereotypes with tags on component-level.
Use associated constraints to verify specifications using automated theorem provers and indicate possible weaknesses.
Ensures that UML specification provides desired level of security requirements.
Link to code via round-trip engineering etc.
Kinds of communication links resp. system nodes.
For adversary type A, stereotype s, have set Threats_A(s) ∈ \{delete, read, insert, access\} of actions that adversaries are capable of. Default attacker:

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Threats_A(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet</td>
<td>(delete, read, insert)</td>
</tr>
<tr>
<td>LAN</td>
<td>=</td>
</tr>
<tr>
<td>Smart card</td>
<td>=</td>
</tr>
</tbody>
</table>

Example **secure links**
Given default adversary type, constraint for stereotype **secure links** violated: According to the Threats_{default}(Internet) scenario, **Internet** link does not provide secrecy against default adversary.

Example **secure dependency**
Violates **secure dependency**: Random generator and **call** dependency do not give security level for random() to key generator.

**data security**
Security requirements of data marked **critical** enforced against threat scenario from deployment diagram.

Constraints: Data marked **secrecy**, **integrity**, **authenticity**, **fresh** fulfills respective formalized security requirements.
Secure Use of Cryptography

Variant of TLS (INFOCOM’99). Cryptoprotocol secure against default adversary?

What Does UMLsec Cover?

Security requirements: ≪secrery≫,…
Threat scenarios: Use Threats=⇒(ater).
Security concepts: For example ≪smart card≫.
Security mechanisms: E.g. ≪guarded access≫.

Model-based Security Aspects

• Define abstract security aspect.
• Define concretization (e.g. protocol).
• If possible, give conditions under which it is secure to weave in aspect using concretization, e.g. by simulation argument.

Secure Channel Aspect

Primary model with directives for security aspects (cf. join points in AspectJ).

To keep data secret, must be sent encrypted.

Secure Channel Aspect: Weaving

Exchange certificate and send encrypted data over Internet.

Aspect Validation

Need to prove concretization securely refines abstract aspect. Challenging problem in security.
For secure channel, have generic result. Often not possible.
⇒ Use translation validation on the weaving transformation, before or after code generation.
Translation Validation: Model or Code?

Model:
+ earlier (less work may have to be redone)
+ more abstract → more efficient
- more abstract may miss attacks
- code construction not completely automatic
- code generators not formally verified
Code:
+ “the real thing” (which is executed)
→ Do both! (as far as feasible; e.g. where largely automatic). Here: look at models.

Model-based Security Analysis

Logic-based security verification of crypto-based software models in UML which is as
• automatic and
• complete
as possible.
Note: can’t be both perfectly automated and complete: Security in general undecidable. Abstract and approximate safely.

Security Analysis

Following Dolev, Yao (1982): To analyze system, verify against attacker model from threat scenarios in deployment diagrams who
• may participate in some protocol runs,
• knows some data in advance,
• may intercept messages on some links,
• injects messages that it can produce in some links
• may access certain nodes.

Adversary: Simulation

A
m(x)
return({y:x})

Adversary
knowledge:
k = 1, y, x

\arg_{k=1} = x

B
m(x)
return({z})

Expressions

Exp: term algebra generated by Var U Keys U Data and
• _ :: _ (concatenation) and empty expression ε
• { _ } _ (encryption)
• Dec ( ) (decryption)
• Sign ( ) (signing)
• Ext ( ) (extracting from signature)
• Hash( _ ) (hashing)
by factoring out the equations Dec_{K^{-1}}({E})_k = E and Ext_K(Sign_{K^{-1}}(E)) = E (for K ∈ Keys).

Security Analysis in First-order Logic

Approximate set of possible data values flowing through system from above. Predicate knows(E) meaning that the adversary may get to know E during the execution of the protocol.
E.g. secrecy: For any secret s, check whether can derive knows(s) using automated theorem prover.
First-order Logic: Basic Rules

For initial adversary knowledge: Define $\text{knows}(E)$ for any $E$ initially known to the adversary (protocol-specific, e.g. $K_A, K_A^{-1}$).

Define above equations.

For evolving knowledge ($K^n$) define

$$\forall E. (\text{knows}(E) \Rightarrow \text{knows}(\text{head}(E)) \land \text{knows}(\text{tail}(E)))$$

Given Sequence Diagram …

… Translate to 1st Order Logic

Connection (or statechart transition) $TR1=(in(msg_in),\text{cond}(msg_in),out(msg_out))$ followed by $TR2$ gives predicate $PRED(TR1)=\forall msg_in. \left[\text{knows}(msg_in) \land \text{cond}(msg_in) \Rightarrow \text{knows}(msg_out) \land PRED(TR2)\right]$ (Assume: order enforced (!).)

Can include senders, receivers in messages.

Abstraction: find all attacks, may have false positives.

Example: Translation to Logic

$$\text{knows}(N) \land \text{knows}(K_2) \land \text{knows}(\text{Sign}_{K_2}(\ldots))$$

$\land \forall init_1,init_2, init_3 \left[\text{knows}(init_1) \land \text{knows}(init_2) \land \text{knows}(init_3) \land \text{and}(\text{Ext}(\ldots)) = \text{init_2} \Rightarrow \text{knows}(\ldots)\right]$

Example: Proposed Variant of TLS (SSL)

Presented at IEEE Infocom 1999.

Goal: send secret protected by session key using fewer server resources.
Check whether can derive knows(s).

Surprise: Yes!

That means: Protocol does not preserve secrecy of s ⇒ not secure.

The Fix

e-Setheo: knows(s) not derivable. Thus s remains secret.

Tool-support: Concepts

Meaning of diagrams informal in [OMG 2005].

Ambiguities problem for
• tool support
• establishing behavioral properties (e.g. safety, security)

Need precise semantics for used part of UML, especially to ensure critical requirements.

In development for UML 2.0 in joint project with IBM Rational (Bran Selic).

Execution Semantics

Behavioral interpretation of a UML subsystem:
1. Takes input events.
2. Events distributed from input and link queues between subcomponents to intended recipients where they are processed.
3. Output distributed to link or output queues.
4. Apply adversary model.
Industrial Applications: Biometry

Biometric Authentication System in development by a company in joint project.
Use our design and validation methods to develop and analyze system.
Discovered three significant security weaknesses against subsequently improved versions.

Related Approaches

UML + security:
• Model-based Risk Assessment (CORAS project, Stoelen et al.)
• Aspect-Oriented Modeling (France et al.)
• Model-driven Security (Basin et al.)
• Fernandez et al.
• Breu et al.
• ...

Conclusions

Formal development of critical systems with UML:
• formally based approach
• automated tool support
• industrially used notation
• integrated approach (specification, source-code, configuration data).

Resources

Jan Jürjens, Secure Systems Development with UML, Springer 2004
Jan Jürjens, IT-Security, Springer 2006 (estim.)

More information (papers, slides, tool etc.):
http://www.umlsec.org
(user: Participant, password: Iwasthere)