

Sound Methods and Effective Tools for Model-based Security Engineering with UML

Jan Jürjens

Software & Systems Engineering
TU Munich, Germany



juerjens@in.tum.de
http://www.umlsec.org




Software Engineering & Security


„Penetrate-and-patch“ (aka „banana strategy“):

- insecure
- disruptive

Traditional formal methods: limited adoption in industry.

- training people
- constructing formal specifications.





 Jan Jürjens, TU Munich: Model-based Security Engineering with UML 2

Model-based Security

Increase security with bounded investment in **time**, **costs**:

- Extract models from **artefacts** arising in **industrial development** and **use** of security-critical systems (UML models, source code, configuration data).
- **Tool-supported**, **theoretically sound**, **efficient automated security analysis**.

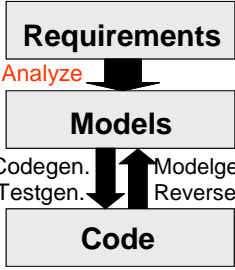


 Jan Jürjens, TU Munich: Model-based Security Engineering with UML 3

Model-based Security Engineering


- **Analyze** (UMLsec) models against security requirements.
- **Generate code** (or tests) from models.
- **Generate models** from evolving or legacy code.

Codegen.
Testgen.

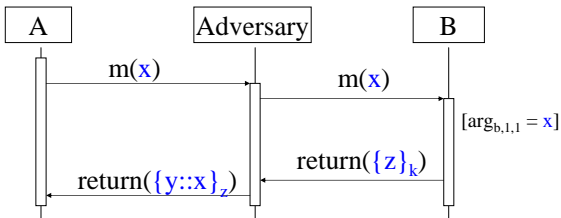


Modelgen./
Reverse E.

Goal: **model-based = source-code based**.


 Jan Jürjens, TU Munich: Model-based Security Engineering with UML 4

Adversary: Simulation



Adversary knowledge: $k^{-1}, y, x, \{z\}_k, z$

- $\forall e, k. Dec_{k-1}(\{e\}_k) = e$


 Jan Jürjens, TU Munich: Model-based Security Engineering with UML 5

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.

For any secret s, check whether can derive *knows*(s) using automated theorem prover.

 Jan Jürjens, TU Munich: Model-based Security Engineering with UML 6

First-order Logic: Basic Rules

Define $knows(E)$ for any E initially known to the adversary.

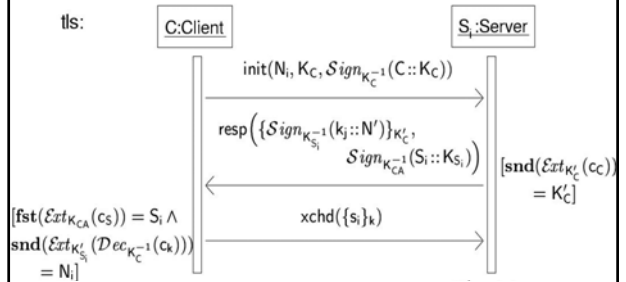
Define cryptosystem. E.g.: $Dec_{K^{-1}}(\{E\}_k) = E$

For evolving knowledge define

$$\forall E_1, E_2. (knows(E_1) \wedge knows(E_2) \Rightarrow knows(E_1 :: E_2) \wedge knows(\{E_1\}_{E_2}) \wedge knows(Dec_{E_2}(E_1)) \wedge knows(Sign_{E_2}(E_1)) \wedge knows(Ext_{E_2}(E_1)))$$

$$\forall E. (knows(E) \Rightarrow knows(head(E)) \wedge knows(tail(E)))$$

Given Sequence Diagram ...



... Translate to 1st Order Logic

Sequence diagram connection

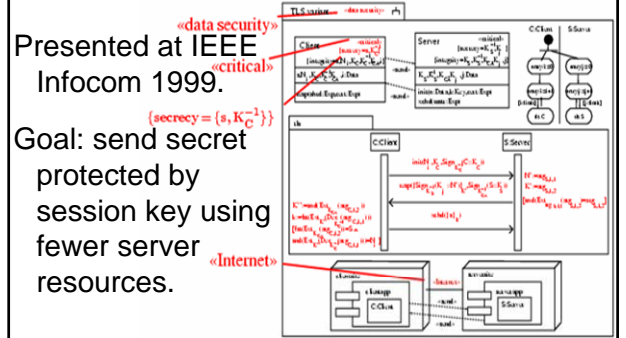
$$TR1 = (in(msg_in), cond(msg_in), out(msg_out))$$

followed by $TR2$ gives predicate $PRED(TR1) = 8 msg_in. [knows(msg_in) \wedge \neg cond(msg_in) \wedge knows(msg_out) \wedge PRED(TR2)]$

Abstraction (e.g. from senders, receivers): find all attacks, may have false positives.

Check whether can derive threat conjecture (e.g. $knows(s)$ for a secret s) from axioms.

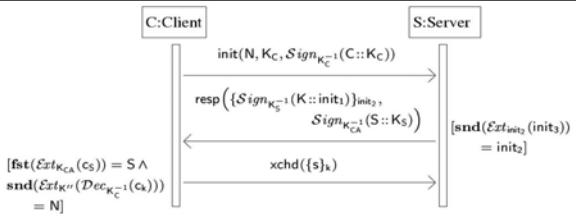
Example: Proposed Variant of TLS (SSL)



Presented at IEEE Infocom 1999.

Goal: send secret protected by session key using fewer server resources.

Example: Translation to Logic



$$knows(N) \wedge \neg knows(K_C) \wedge \neg knows(Sign_{K_C^{-1}}(C::K_C)) \wedge \exists 8 init_1, init_2, init_3. [knows(init_1) \wedge knows(init_2) \wedge knows(init_3) \wedge snd(Ext_{init_2}(init_3)) = init_2 \wedge knows(\{Sign_{K_S^{-1}}(\dots)\}) \wedge \dots]$$

Surprise ...

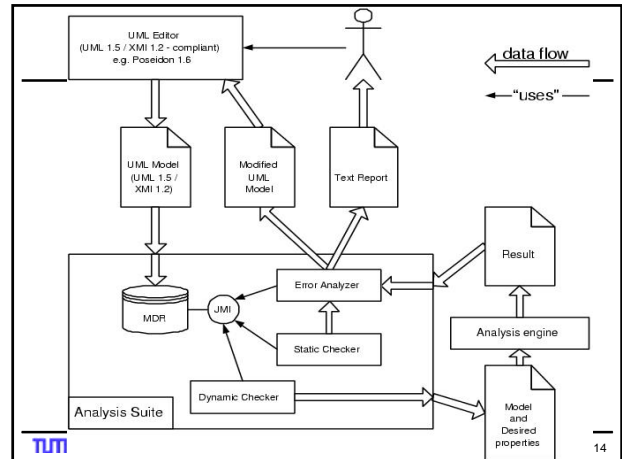
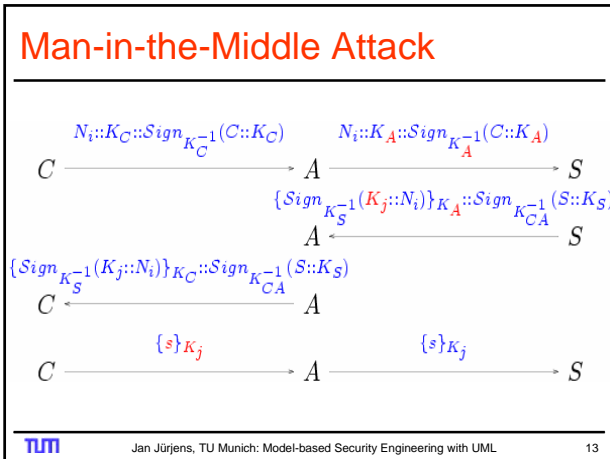
E-SETHEO csp03 single processor running on host .
(c) 2003 Max-Planck-Institut fuer Informatik and Technische Universitaet Muenchen

Can derive $knows(s)$. That is: Protocol does not preserve secrecy of s against adversaries.

Attack
analyzing results ...
proof found
time limit information: 298 total / 297 states
e-SETHEO done. exiting

→ Completely insecure wrt stated goals.

But why? Use prolog-based attack generator.



Biometric Authentication System

In development by large German telecommunication company.

In joint project, use presented security analysis tools at given UML specification.

So far, have discovered **three major attacks** against subsequently improved versions (misuse counter circumvented by dropping / replaying messages, smart-card insufficiently authenticated by recombining sessions).

TUM Jan Jürjens, TU Munich: Model-based Security Engineering with UML 15

Related Work

UML and security:

- C. Montangero et al.: Degas project.
- D. Basin et al.: Secure UML

UML verification:

- Lilius et al.: vUML
- ...

TUM Jan Jürjens, TU Munich: Model-based Security Engineering with UML 16

Resources

Jan Jürjens, Secure Systems Development with UML, Springer 2004

Workshop: CSDUML@SAFECOMP05 (Norway, Sept. 05)

Application to C source code: Memocode '05

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

Note: International Symposium on Secure Software Engineering (ISSSE 06 - IEEE) [S. Redwine, A. Hall, J. Wing]

TUM Jan Jürjens, TU Munich: Model-based Security Engineering with UML 17