Model-Driven Engineering for Safety, Security and Performance:
SysML-Sec

Ludovic APVRILLE
ludovic.apvrille@telecom-paristech.fr

Seminar - City University
Telecom ParisTech
Outline

Context: Security for Embedded Systems
Embedded systems

SysML-Sec
Method
SysML-Sec

Case study
Case Study

Demo
Demo

Conclusion
Conclusion, future work and references
Examples of Threats

Transport systems

- Use of exploits in Flight Management System (FMS) to control ADS-B/ACARS [Teso 2013]
- Remote control of a car through Wifi [Miller 2015] [Tecent 2017]

Medical appliances

- Infusion pump vulnerability, April 2015.
  http://www.scip.ch/en/?vuldb.75158
Examples of Threats (Cont.)

Internet of Things

- Proof of concept of attack on IZON camera [Stanislav 2013]

- Vulnerability on fitbit [Apvrille 2015]

- Hacking a professional drone [Rodday 2016]
Finding Vulnerabilities on IoTs

What’s inside? Let’s look together!
Inside a Fitbit

Don’t try this at home!
Inside a Fitbit (Cont.)

Again: don’t try this at home!
Inside a Fitbit (Cont.)
Fitbit: Hardware Components
Firmware Dumping
Then, How to Identify Vulnerabilities?

Investigations

- Testing ports (JTAG interface, UART, ...)
- Firmware analysis
- Memory dump
- Side-channel analysis (e.g. power consumption, electromagnetic waves)
- Fault injection
- ...

Secure your systems!

- Develop your system with security in mind from the very beginning
- Our solution: SysML-Sec, supported by TTool
Designing Safe and Secure Embedded Systems: SysML-Sec

Main idea

- **Holistic approach**: bring together experts in embedded systems, system architects, system designers and security experts (with SysML)

Common issues (addressed by SysML-Sec):

- Adverse effects of security over safety/real-time/performance properties
  - Commonly: only the design of security mechanisms
- Hardware/Software partitioning
  - Commonly: no support for this in tools/approaches in MDE and security approaches
SysML-Sec: Methodology

- Analysis
  - Requirements
    - Safety
    - Functional
    - Security
  - Attack Trees
- Fault Trees
- HW/SW Partitioning
  - Application
  - Architecture
  - Mapping
- Security Countermeasures
  - Redundancy, ...
- Safety Countermeasures
  - Failsafe Mode, Plausibility Check, ...
- Attack Scenarios
- Security Countermeasures
  - Firewall, Data Security, ...
- Software Design
- Security Countermeasures
  - Security Algorithms, ...

Legend
- Modeling
- Verification
- Security
- Safety
- User-defined
- Automatic
- Reconsideration

Code Generation

Fully supported by TTool
Partitioning

Before mapping
▶ Security mechanisms can be captured but not verified

After mapping
▶ Verify security (confidentiality, authenticity) according to attacker capabilities
   ▶ Whether different HW elements are or not on the same die
   ▶ Where are stored the cryptographic materials (keys)
   ▶ Where are performed encrypt/decrypt operations
▶ Impact of security mechanisms on performance and safety
   ▶ e.g. increased latency when inserting security mechanisms
Partitioning Verification

Modeling

Automatic Verification

Safety  Performance  Security
Safety and Security Mechanisms

Data Encryption/ Authentication

Safety

Security

Performance
Safety and Security Mechanisms (Cont.)

Data Security with Hardware Security Module

Safety

Security

Performance
Safety and Security Mechanisms (Cont.)

Redundancy/Coherence Check

Safety
Security
Performance

Add security
Add security
Safety and Security Mechanisms

Failsafe mode

systemCheck

[systemOk] [else]

defaultMode failsafeMode

Safety

Security

Performance
Safety/Security/Performance

Requirements

Security
   Automated generation
   Add/modify security mechanisms
   Modify architecture (private bus, etc.)
   Modify mapping
   Succeeds :-) Security leads to unsafe behaviour
   Reconsider security req.
   Succeeds :-) Security leads to degraded perf. (e.g., increased mean latency)

Safety
   Add/modify safety mech. (e.g. safe modes)
   Modify architecture (e.g. redundancy)
   Modify mapping
   Succeeds :-) Safety leads to unsafe behaviour
   Reconsider safety req.
   Succeeds :-) Safety leads to degraded perf.

Performance
   Reconsider algorithms
   Modify architecture (Nb of cores, etc.)
   Modify mapping
   Succeeds :-) Performance leads to unsafe behaviour
   Reconsider performance req.
   Fails
   Performance issue due to security mechanisms
   Performance issue due to safety mechanisms

System design

Verification of design w.r.t. requirements
Precise model of security mechanisms (security protocols)

Proof of security properties: confidentiality, authenticity

Channels between software blocks can be defined as private or public

- This should be defined according to the hardware support defined during the partitioning phase
Case Studies

Cyber security of connected vehicles
- Safety/Security/Performance
- EVITA FP7 Partners: Continental, BMW, Bosch, ...
- VEDECOM

H2020 AQUAS
- Automated train sub-systems (ClearSy): Safety/Security/Performance
- Industrial Drives (Siemens): Safety/Security/Performance

Nokia
- Digital architectures for 5G networks (Safety/Performance)
Case Study: VEDECOM Autonomous Vehicle

Model

Verification

Tests
Constraints

- Standard: ISO26262
  - SOTIF: Safety Of The Intended Function
- Security: impact of potential attacks on safety
Requirements
Attacks
Safety Verification (Before Mapping)

Reachability/Liveness

Queries

Safety Pragma
A[] Supervisor.running
Perception.distance<threshold -->
Supervisor.brakingOrder
Safety Verification (After Mapping)

Reachability Graph

Minimized RG
Security Verification

Dialog window

Backtracing

- PerceptionCalc1
  - calcMark = 500 : Natural;
  - calcObstacle = 500 : Natural;
  - calcTraj = 500 : Natural;
  - calcConfidenceLevel = 500 : Natural;
  - calcVariableCoherence = 500 : Natural;
  - calcRegulation = 500 : Natural;
  - calcInfrastructure = 500 : Natural;

- PerceptionCalc2
  - calcMark = 500 : Natural;
  - calcObstacle = 500 : Natural;
  - calcTraj = 500 : Natural;
  - calcRegulation = 500 : Natural;
  - calcConfidenceLevel = 500 : Natural;
  - calcVariableCoherence = 500 : Natural;
  - calcInfrastructure = 500 : Natural;
Performance Verification

Latency

Bus/CPU Load

Latency Bus/CPU Load

35/39  June. 2018  Institut Mines-Telecom  SysML-Sec
SW Design, Code generation, Test

- First SW model from mapping models
- SW model refinement
- SW model verification (safety, security)
- Code generation
  - (Virtual) Prototyping, test
Demo: SmartCard

- Main functions of the system
- Safety of the system (before mapping, after mapping)
- Performance
- Model enhanced with Security
- Impact on performance
Conclusion and Future Work

Achievements: SysML-Sec

- Methodology for designing safe and secure embedded systems
- Fully supported by TTool
- Applied to different domains, e.g., automotive systems, IoTs, malware

Future work

- Security risk assistance and backtracing
- Assistance to handle conflicts between security/safety/performance
  - Design space exploration
To Go Further ...

Web sites


References
