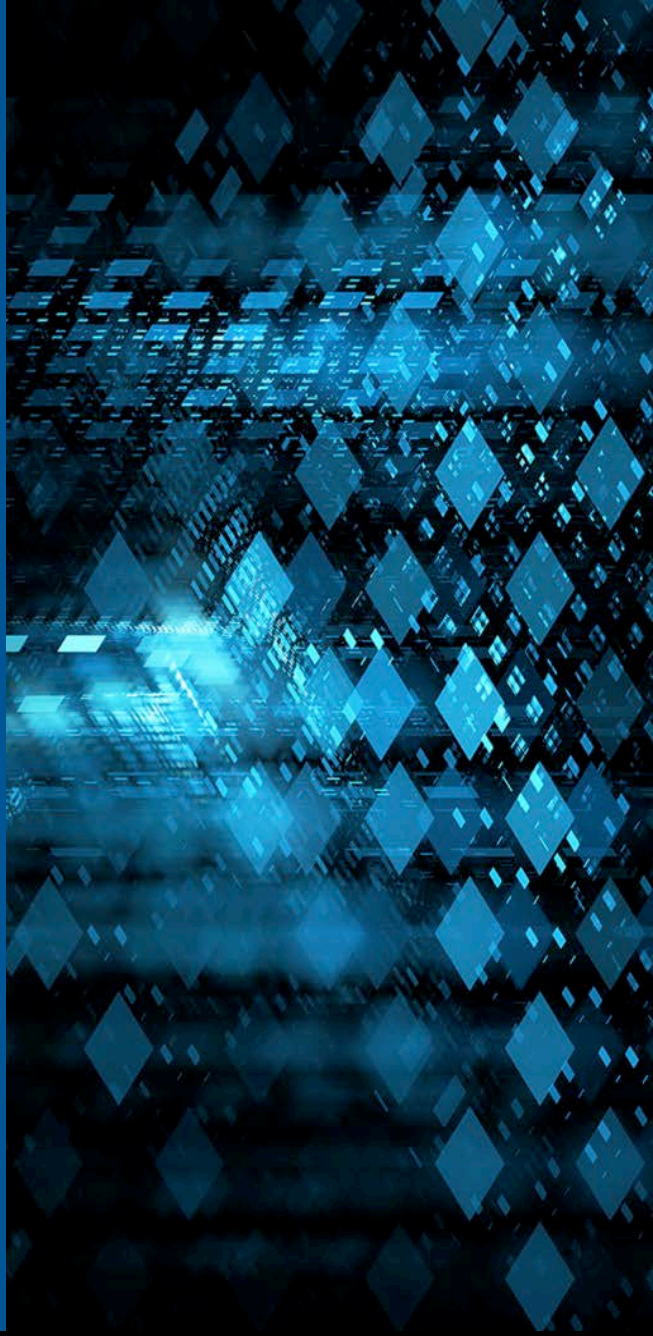
AADL for Secure & Safe Systems Design & Analysis

Part 5 - Security

Julien Delange

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0003990

Tutorial Agenda

Introduction: required background, role of MBE, tutorial overview

AADL Concepts: learn enough to use AADL and OSATE

Flow Latency: how to capture flow characteristics? How can I generate a flow analysis from my architecture model?

Safety Analysis: how to capture safety in an AADL model? What types of reports can I generate? How can I generate them?

Security Analysis: representation of security aspects. How to detect security issues? What type of reports can we generate?

Recent security reports

Jeep hack (July 2015)

- Remote control of a car, 1.4M recalls
- https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

Airplanes, ATC control system (2012)

- No encryption or authentication between planes and ground stations
- Sniffing packets or injecting forged data (what is the impact with the autopilot?)
- https://www.youtube.com/watch?v=CXv1j3GbgLk

Medical devices

- Remote access to infusion pumps without authentication
- Hundreds of devices are **insecure by design**
- https://securityintelligence.com/news/do-no-harm-medical-device-vulnerabilities-put-patients-at-risk/

Towards an AADL security modeling guide

Security is being a lot of attention recently

- **Fact**: our world is becoming software intensive
- **Reality**: it opens new propagation paths
- **Examples**: software security is now a life-threatening concern

Capture architecture security concerns

- Detect architecture issues leading to vulnerabilities (sharing security domains, etc.)
- Guideline to model secure architecture (e.g. MILS)

Analyze architecture from a security perspective

- Detect common vulnerabilities from top CVE (MITRE)
- Automatically generate reports: attack surface, attack impact, etc.
- Analyze their **impact & propagation** through the architecture

Ongoing security modeling guide

Modeling rules for secure systems

- How to use AADL for designing safe system?
- How to model software and hardware security mechanisms?
- Rely on popular vulnerability (CVE) and weaknesses (CWE)

Additional property set

- Specify implementation details (encryption, domains, etc.)
- Support analysis tools to detect vulnerabilities

Improving System and Software Security with AADL

https://insights.sei.cmu.edu/sei_blog/2016/02/improving-system-and-software-security-with-aadl.html

Security properties overview

- **security_levels** – associate a security level (top-secret, secret, unclassified) to a modeling element (A)
- **domains** – associate a domain with a modeling element (A)
- **trust** – specify component security assurance, how much this component is trustable (A)
- **exposure** – specify exposure to the outside world (A)
- **encryption** – specify how data is encrypted on components connection or memory storage (A, CG)
- **protocol** – specify protocols (http, ftp, tcp) used to realize a communication (A, CG)
- **authentication_method** – authentication methods implemented on a particular connection (A)

Used for:
A = Analysis
CG = Code Generation

Security properties: levels and domains

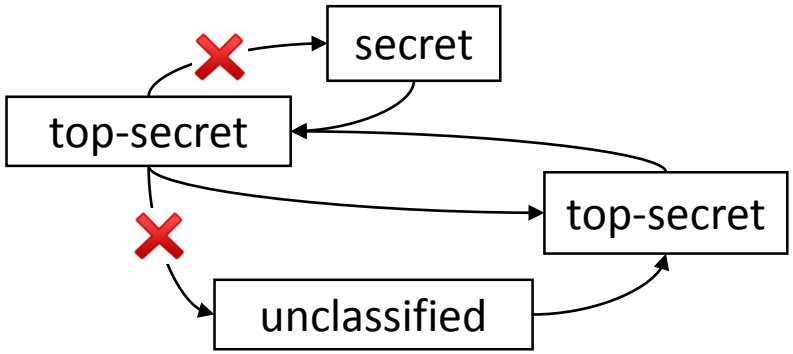
security_levels

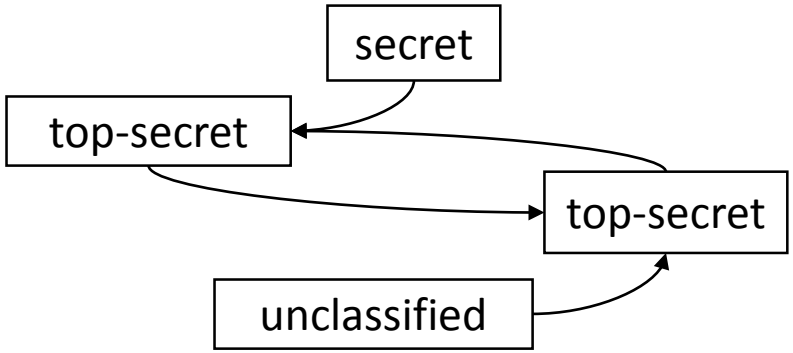
- associate a security level to a modeling element (component, feature, data, etc.)
- generic label (secret, top-level, unclassified)
- support for analysis methods using ordering (biba, bell-lapadula, etc.) – an unclassified component cannot communicate with a top-secret one

domains

- unordered classifier associated with each element
- project- or user-specific
- support for analysis method focused on isolation (e.g. MILS)

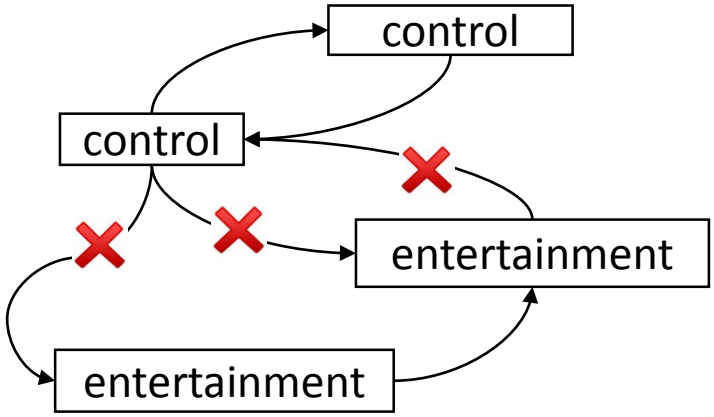
Security properties: levels and domains

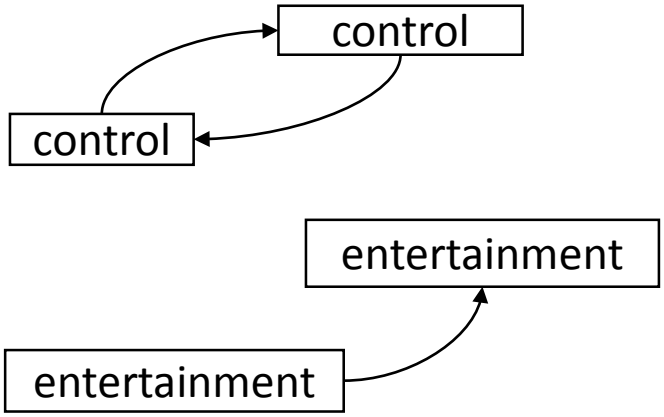




Goal: protect data from lower levels

Analysis using levels





Goal: isolate data across domains

Analysis using domains

Security properties: trust and exposure

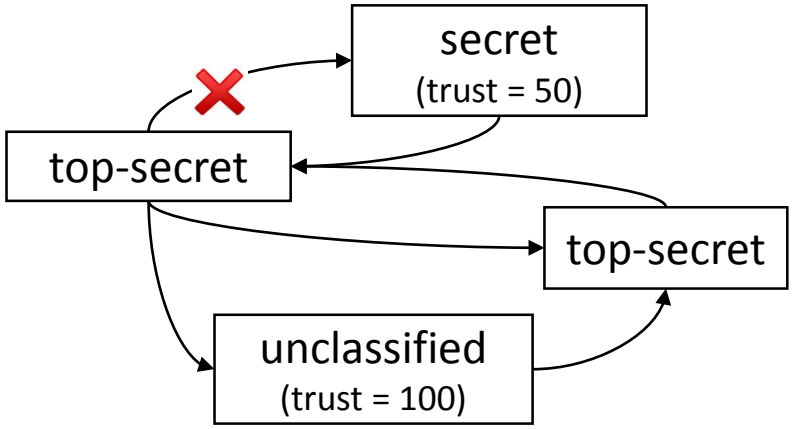
trust - gauge assurance in implementation correctness

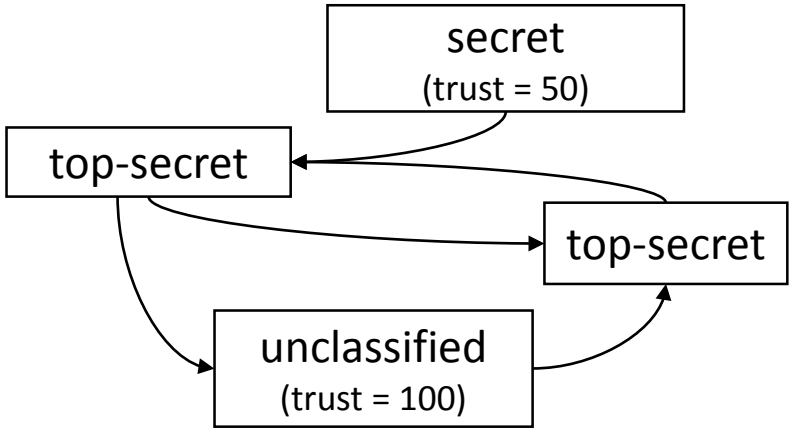
- range between 0 and 100, user- and project- dependent
- reflect validation, review or certification efforts
- used to bypass usual security rule

exposure - physical exposure of a component

- limited to AADL physical components: processor, memory, etc
- range between 0 and 100, user- and project- dependent

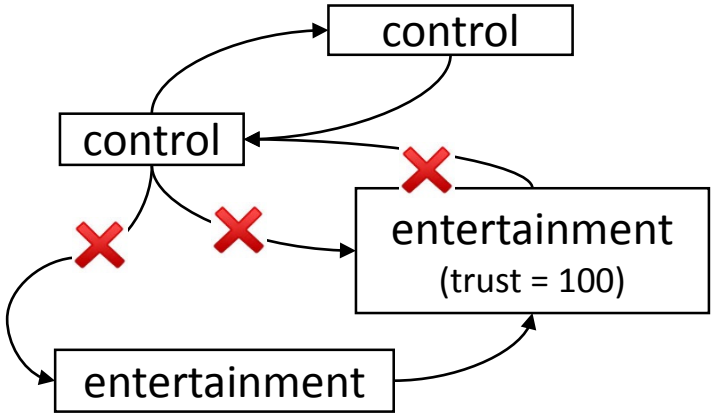
Security properties: trust

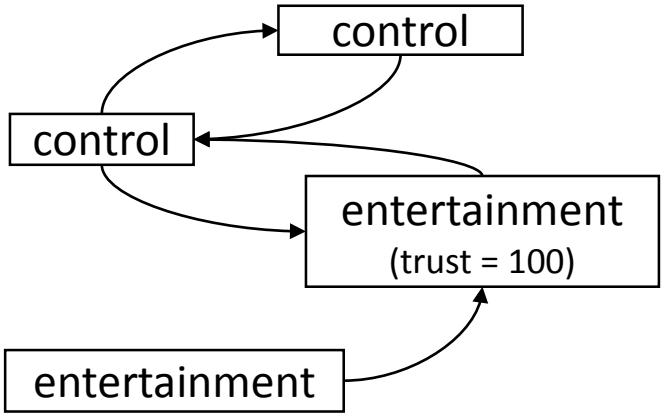




Goal: protect data from lower levels

Analysis using levels

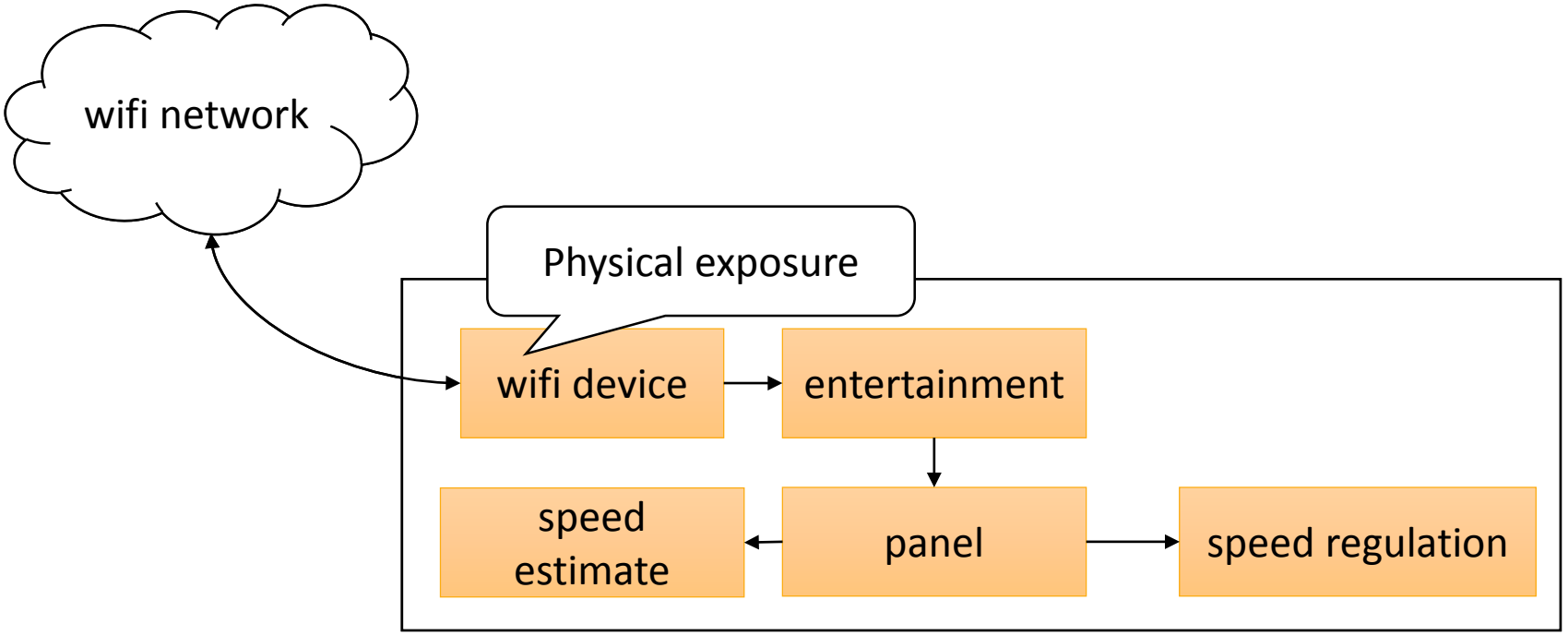




Goal: isolate data across domains

Analysis using domains

Security properties: exposure



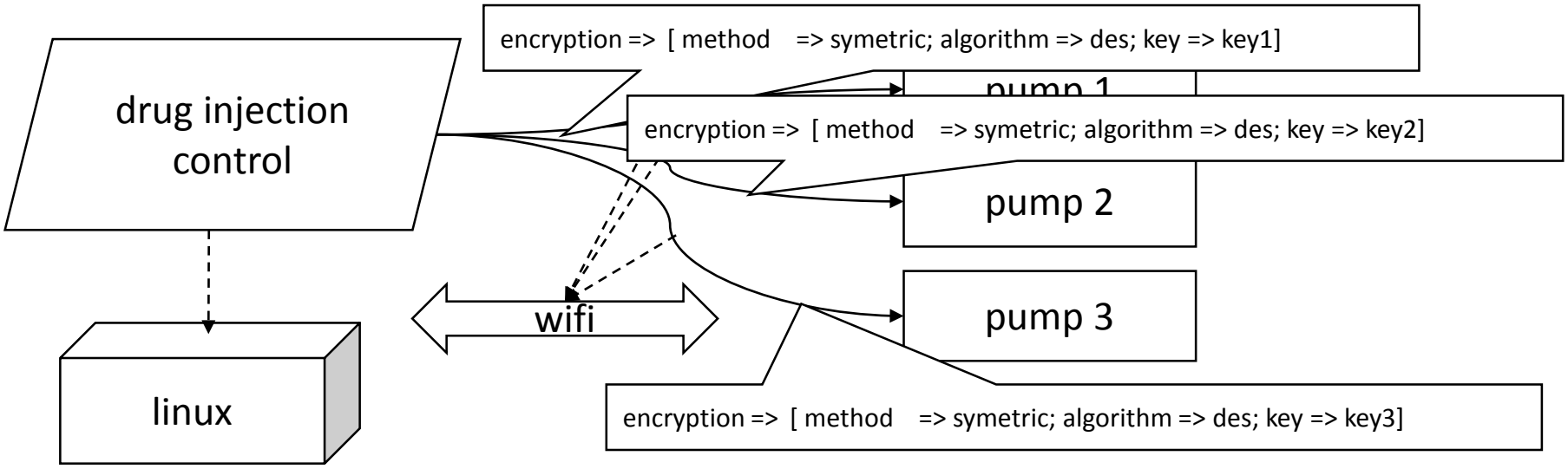
capture exposure of physical component

use architecture information to analyze impact

connection between component (features)

binding with platform and buses

Security properties: encryption

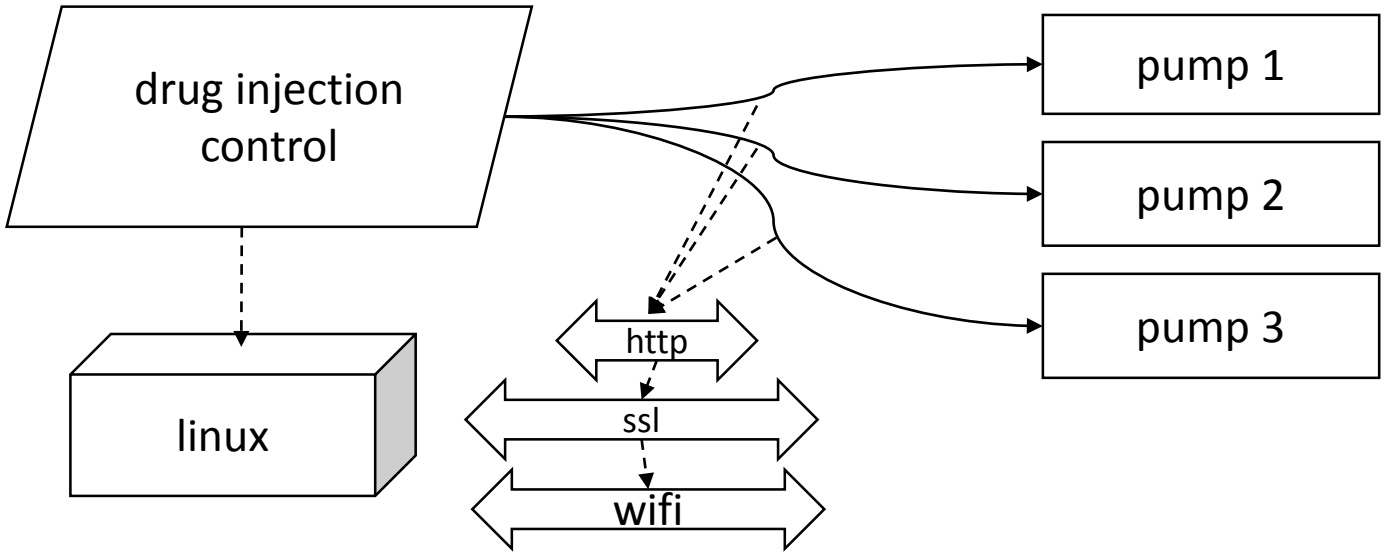


specify encryption on connection or virtual buses

detect inappropriate use of encryption

- reuse of keys
- inappropriate encryption algorithm or deployment (keys)

Security properties: protocol



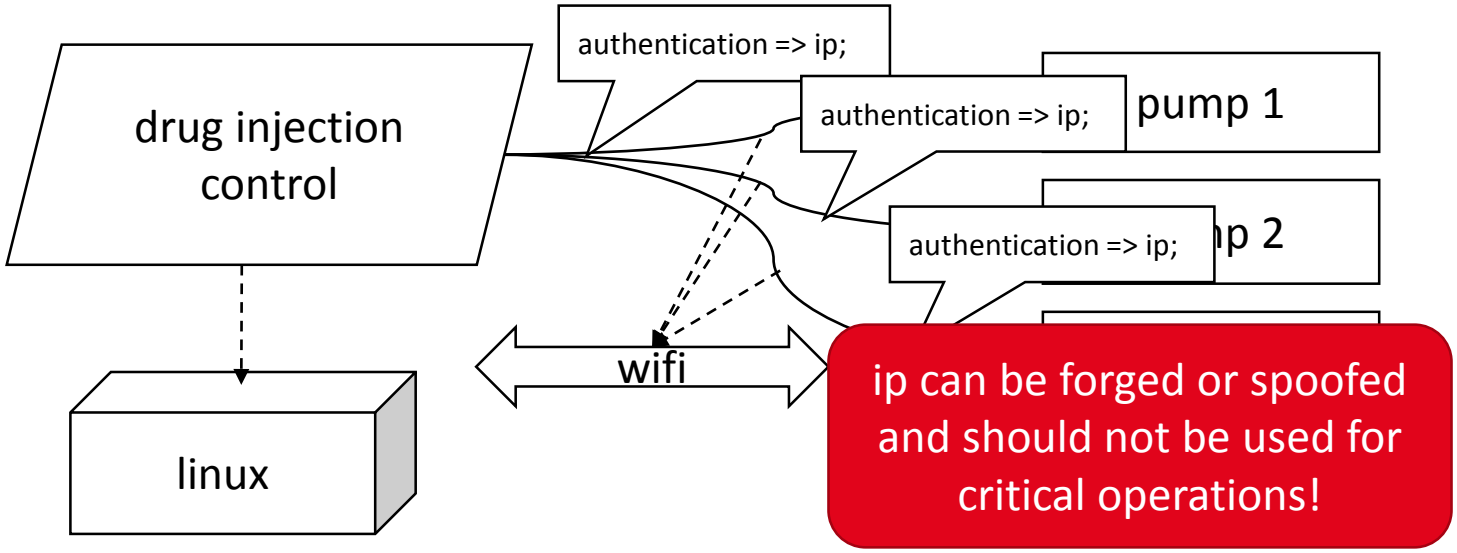
model the protocol stack

find inappropriate protocol

protocol configuration

invalid protocol use or prohibited for security reasons

Security properties: authentication_method



specify authentication method for logical connections

Support set of generic method (ip, userpass, shared key)

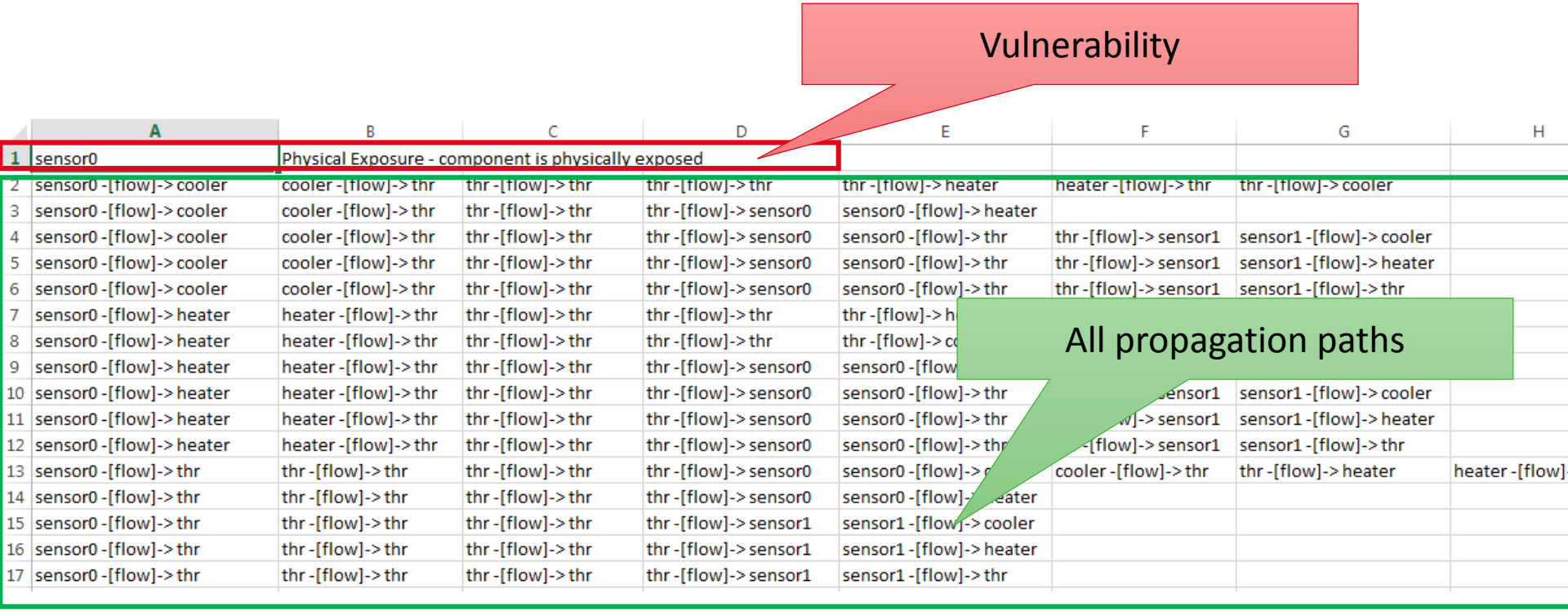
Extensible property, similar to `AADL_Project.aadl`

Security Analysis Tools – Attack Impact, textual

Similar to Fault Impact/FMEA, bottom-up approach

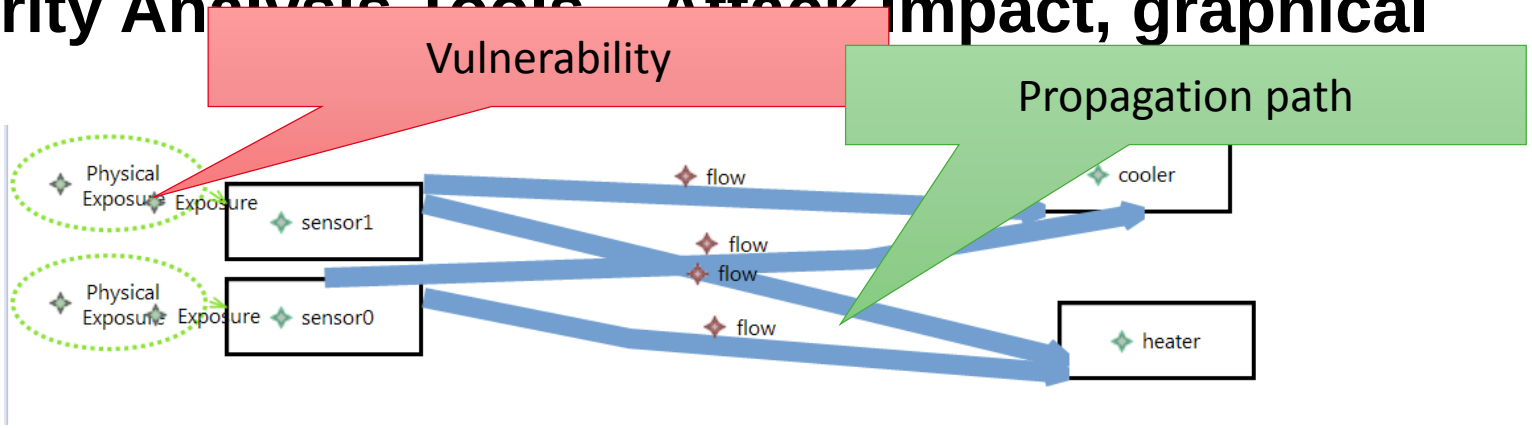
All paths from a vulnerability to impacted component

Vulnerability Propagations can be extended by user



	A	B	C	D	E	F	G	H
1	sensor0	Physical Exposure - component is physically exposed						
2	sensor0 -[flow]-> cooler	cooler -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> heater	heater -[flow]-> thr	thr -[flow]-> cooler	
3	sensor0 -[flow]-> cooler	cooler -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> sensor0	sensor0 -[flow]-> heater			
4	sensor0 -[flow]-> cooler	cooler -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> sensor0	sensor0 -[flow]-> thr	thr -[flow]-> sensor1	sensor1 -[flow]-> cooler	
5	sensor0 -[flow]-> cooler	cooler -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> sensor0	sensor0 -[flow]-> thr	thr -[flow]-> sensor1	sensor1 -[flow]-> heater	
6	sensor0 -[flow]-> cooler	cooler -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> sensor0	sensor0 -[flow]-> thr	thr -[flow]-> sensor1	sensor1 -[flow]-> thr	
7	sensor0 -[flow]-> heater	heater -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> h			
8	sensor0 -[flow]-> heater	heater -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> c			
9	sensor0 -[flow]-> heater	heater -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> sensor0	sensor0 -[flow			
10	sensor0 -[flow]-> heater	heater -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> sensor0	sensor0 -[flow]-> thr	ensor1	sensor1 -[flow]-> cooler	
11	sensor0 -[flow]-> heater	heater -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> sensor0	sensor0 -[flow]-> thr	]-> sensor1	sensor1 -[flow]-> heater	
12	sensor0 -[flow]-> heater	heater -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> sensor0	sensor0 -[flow]-> thr	[flow]-> sensor1	sensor1 -[flow]-> thr	
13	sensor0 -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> sensor0	sensor0 -[flow]-> c	cooler -[flow]-> thr	thr -[flow]-> heater	heater -[flow]
14	sensor0 -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> sensor0	sensor0 -[flow]- eater			
15	sensor0 -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> sensor1	sensor1 -[flow]-> cooler			
16	sensor0 -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> sensor1	sensor1 -[flow]-> heater			
17	sensor0 -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> thr	thr -[flow]-> sensor1	sensor1 -[flow]-> thr			

Security Analysis Tools – Attack Impact, graphical



User-friendly, graphical representation

Graphical representation of flow "importance"

Distinguish how vulnerabilities *flow* in the architecture

Graphical representation of impact

Show impact of a vulnerability within the architecture

Interface with Attack Tree

Auto-Generation of Attack Tree for component

Isolate vulnerabilities related to a component in a separate diagram

Security Analysis Tools – Attack Tree

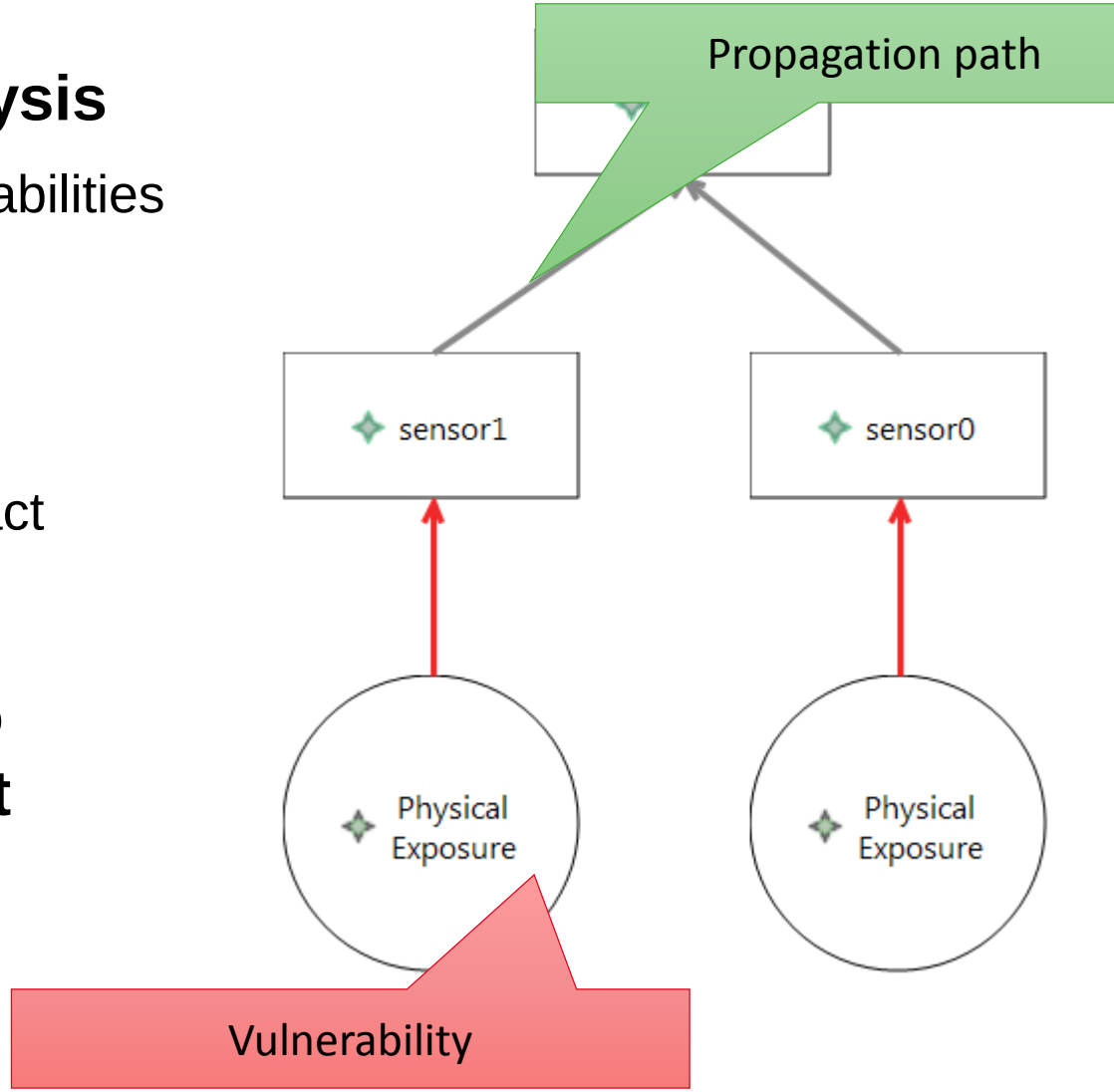
Similar to Fault-Tree Analysis

No more error but vulnerabilities

Top-down approach

Complement Attack Impact

Identify all contributors to compromise a component



Exercise 5 - Objectives

Specify some components as exposed

- Set exposure properties to 0 for all components
- Set the exposure to 50 for `sensor0` and `sensor1` (physically exposed to attackers)
- For the wireless variance, set exposure to 80 for `eth0` (the network) to model the exposure to external attackers. Keep 0 for the wired variance

Generate reports

- **Attack Impact** – system architecture with vulnerabilities and flows
- **Attack Tree** – how to compromise the heater or cooler?

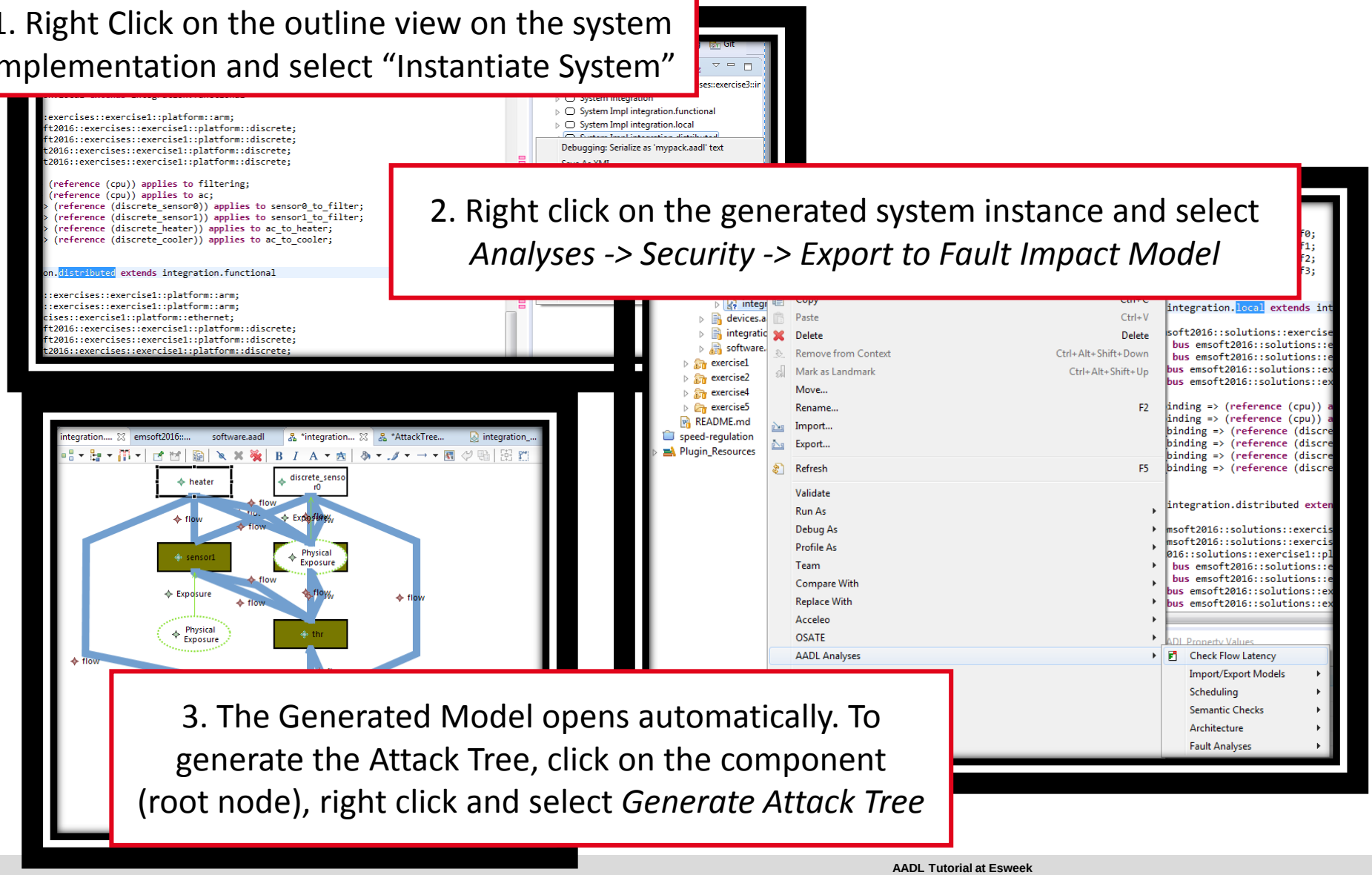
Observe difference between wired and wireless

- What is the difference? why?

Exercise 5 – Generating Attack Impact

1. Right Click on the outline view on the system implementation and select “Instantiate System”

2. Right click on the generated system instance and select *Analyses -> Security -> Export to Fault Impact Model*



3. The Generated Model opens automatically. To generate the Attack Tree, click on the component (root node), right click and select *Generate Attack Tree*