AADL Security Annex

Julien Delange, Peter Feiler, Will Klieber ,
Min-Young Nam, Joseph Seibel

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

DM-0003597

Agenda

Why a security annex?

Main Concepts

Analysis Support

Rationale for security annex

Increasing attention on security for cyber-physical systems

- **Automotive:** Jeep hack (1.4M recall), Ford (400K)
- **Avionics:** Chris Roberts story

Security issues are not only a matter of code

- **Configuration**: inappropriate encryption/isolation
- **Deployment**: collocated components

Add security characteristics in Architecture Models

- Extend **AADL** with security annotations

Design guidelines

	Pros	Cons
Property Set	• No learning curve, easy to adopt • No specific tool support (i.e. parser) • Extensible, easily modifiable • Tool compatibility • Quick to design/prototype	• Limited capability or expressiveness
Annex Language	• Clear, separate declarations • Ability to use with other languages	• Need to train users • Support of specific parser • Tool compatibility • Long design process (i.e. EMV2)

Security Annex document

Written using markdown

- Facilitate review and track changes
- Using pandoc to convert to SAE format

Exercised early to investigate potential uses

- Security analysis
- Code generation for seL4

Starting ballot in 2016

Concepts – security levels & domains

Security Levels

Distinguish levels (top-secret, secret)

Compliance with approach such as Bell-Lapadula

```
security_levels : list of aadlinteger => (100) applies to (all);
top_secret      : constant aadlinteger => 10;
secret          : constant aadlinteger => 40;
unclassified    : constant aadlinteger => 100;
```

Domains

Distinguish domains (i.e. entertainment, command & control)

Not a security hierarchy

Capture MILS requirements

```
domains : list of aadlstring applies to (all);
```

Concepts – Trust and Exposure

Trust

Can I trust this component?

Reflect efforts to prove component correctness

0 = no review/verification ; 100 = formally verified

```
trust : aadlinteger 0 .. 100 => 0 applies to (all);
```

Exposure

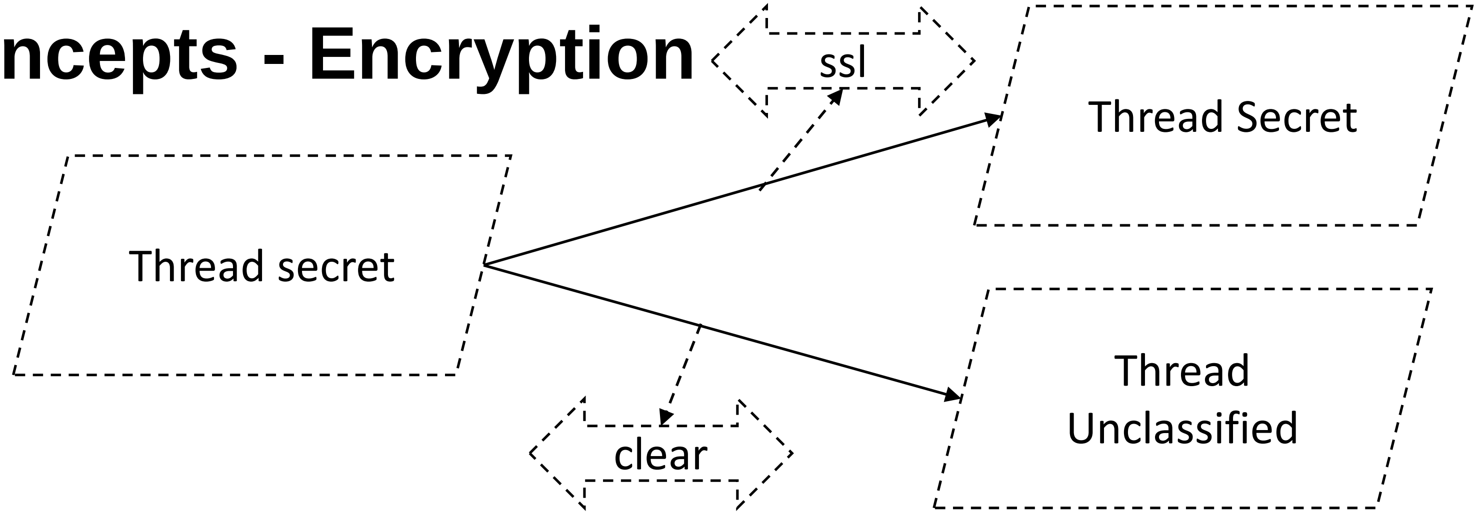
How exposed is my component?

Reflect the possibility of a physical attack

0 = protected (in a box) ; 100 = exposed to everybody

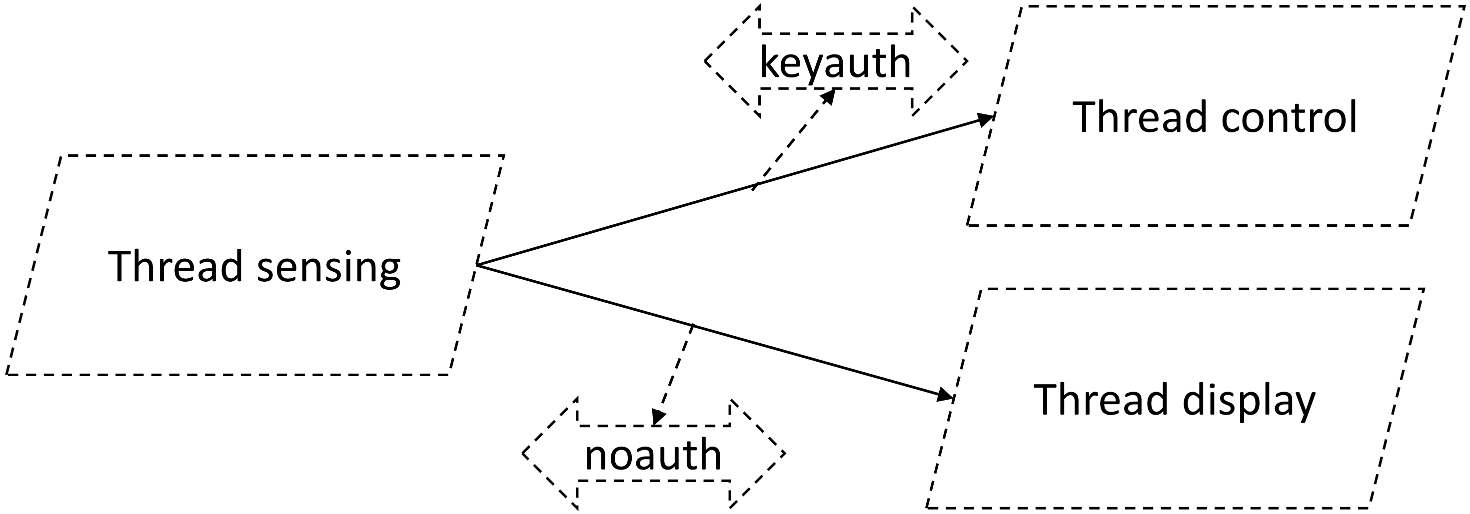
```
exposure : aadlinteger => 100 applies to
         (bus, virtual bus, processor, device, system, memory);
```

Concepts - Encryption




```
encryption : security_properties::encryption_type applies to
   (port, virtual bus, bus, memory, access);
encryption_type    : type record (
   method          : security_properties::supported_encryption_method;
   algorithm       : security_properties::supported_encryption_algorithm;
   public_key      : aadlstring;
   private_key     : aadlstring;
   key             : aadlstring;
   operation_mode  : security_properties::supported_operation_mode;
);
supported_encryption_method     :
              type enumeration (symetric, assymetric, clear);
supported_encryption_algorithm  :
              type enumeration (tripledes, des, rsa, blowfish, aes, clear);
supported_operation_mode : type enumeration (ecb, cbc, pcbc, cfb, ofb, ctr);
```

Concepts - Authentication




```
supported_authentication_methods:
        type enumeration (shared_password, user_password, key, ipaddr);
authentication_method :
        list of security_properties::supported_authentication_methods
        applies to (bus, virtual bus);
```

Analysis Tools

Attack Surface

Discovery of vulnerabilities in the architecture

Measure of how safe is your system

Attack Impact & Attack Tree

Graphical representation of vulnerabilities & their impact

Similar goal than FMEA & FTA for safety

Code Generation to seL4

First formally verified kernel with a focus on security

Leverage HACMS/SMACCM efforts

Julien Delange

CMU-SEI

4500 5th avenue

Pittsburgh, PA15213

+1-412-268-9652

jdelange@sei.cmu.edu