Security Annex Update

January 30, 2018

Dave Gluch

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. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM18-0071

Security Annex

The security annex provides guidance and support for modeling and analyzing, using the AADL, security

- Policies and Requirements
  - Documentation
  - Verification
- Protections
  - Access Control and Protection
  - Information/Data Protection
  - Action/Command Protection
- Architectures
  - Specialized architectures (e.g., MILS, D-MILS)
  - Secure kernels (e.g., seL4, MILS separation kernels)
- Vulnerabilities
- Threats/Attacks

Security Annex Documentation

Security is a property of the ‘composite’ that is broader than Error Model Annex, Behavior Annex, etc.

- No safety annex, performance annex

Security Annex establishes a framework for using AADL/OSATE for security engineering similar to the ARINC 653 annex

- Security Annex Standard Document
- Supplemental Elements
  - Security Annex: ALISA
  - Security Annex: Resolute
  - Security Annex: EMV2
  - Security Annex: Security Architecture Modeling Guidelines
  - Security Annex: Transport Aircraft Example System
  - Others (e.g. attack trees, MILS analysis)

Security Policies and Requirements

Documentation of policies, requirements, verification, and assurance information, etc. in AADL models.

Verification of security policies

- ensure that requirements and lower level security policies satisfy high-level policies
- ensure there are mechanisms that enforce security policies/requirements

Does not provide an assessment of the effectiveness, self-consistency, or validity of the policies.

Security Policies and Requirements – Implementation

High Level Security Policies (e.g. organizational)

- Captured as goals in ALISA/OSATE (.goals)
- Employ stakeholders and relevant organizations – optional
- Naming to identify (e.g. TransportAircraftSecuirtyPolicies.goals)

Lower Level Security Policies (e.g. operating system)

- Captured as requirements (.reqspec)
- Naming to identify (e.g. *OSSecurityPolicies.reqspec*)

Verification and Assurance

- Security requirements satisfy high-level security policies
- Lower-level security policies are requirements and satisfy high-level security policies
- Both are requirements in ALISA/OSATE

Examples - Policies

File: TransportAircraftSecuirtyPolicies.goals

```
stakeholder goals SecurityPolicies for TransportAircraft_pkg::transportAircraft
[
description "These are the high level (system) security policies for the Aircraft."
goal Security: "System Security"
[
description "The system must provide security protection."
stakeholder sei.dpg sei.phf SAE_AADL_Standards.bal
]
goal MasterSecurityPolicy: "A Master System Security Policy must be developed and
certified."
[
description "A master system security policy document must be developed and
certified by all of the agencies and organizations involved in flight certification
of the aircraft."
stakeholder sei.sam sei.dpg SAE_AADL_Standards.bal SAE_AADL_Standards.AADL_Committee
FAA_Certification.inspec01
]
]
```

File: TransportAircraftSecurityReqs.reqspec

```
system requirements TransportAircraftSecurityReqs for
                                        TransportAircraft_pkg::transportAircraft
```

Information/Data Protection

Model, assess, and assure data protection approaches and levels

- Security Classifications – model, assess, and assure security classification operational policies and implementations including
  - personnel
  - information
- Cryptography & Encryption – model, assess, and assure security encryption and supporting cryptographic methods and implementations
- Protected Containment – model, assess, and assure protected containment units such as protected address spaces, virtual machines, and partitions
  - Data confidentiality, Integrity (authenticity of data), availability

Information/Data Protection - Security Classifications

Model, assess, and assure security classification operational policies and implementations including

- personnel
- information

Implementation with AADL Property Sets and Resolute Annex

- Property Set (`Security_Properties.aadl`)
- Library (`Security_Resolute_Lib.aadl`)
  - claim and computational functions
- Prove Statements

Security Property Set

```
property set Security_Properties is

          -- We include US DOD and DOE
          Security_Clearance_Levels_US: type enumeration
          (SSBI, SCI, SAP, Top_Secret, Secret, Public_Trust, Confidential, Unclassified,
Q_Clearance, L_Clearance);
--
          Highest_Security_Clearance_Level_US: Security_Properties::Security_Clearance_Levels_US
          applies to (all);

-- This secondary clearance is provided in the event a person holds another clearance from a
different agency.
          Secondary_Security_Clearance_Level_US:Security_Properties::Security_Clearance_Levels_US
          applies to (all);
--
-- Information Security Levels
--
          Information_Security_Level_US: enumeration
          (SCI, SAP, Top_Secret, Secret, Confidential, Controlled_Unclassified, Unclassified)
          applies to (all);     -- may want to reduce the all to specific component categories

... additional  . . .


end Security_Properties;
```

Security Resolute Library

```
package Resolute_Security_Lib
public
            with Security_Properties;
             with Resolute_Stdlib;
annex Resolute {**
    secret : string ="Secret"
    topSecret : string = "Top Secret"

            has_secret_information_level_security (p: component) <=
            ** "componet " p " has secret information security" **
            has_property (p, Security_Properties::Information_Security_Level_US) and
            property (p, Security_Properties::Information_Security_Level_US) = secret
--
    has_secret_clearance (p: component) <=
            ** "the highest security clearance is secret of component " p " is Secret" **
            has_property (p, Security_Properties::Highest_Security_Clearance_Level_US)
            and
            property (p, Security_Properties::Highest_Security_Clearance_Level_US) = secret

            is_connected(a : component, conn : connection, b : component) <=
            ** "there is a connection between " a " and " b **
            connected (a, conn, b)
--
            secret_to_secret (s: component, c: connection, d:component) <=
            ** "the connection " c " is secret to secret between " s " the source and " d "the destination"**
     is_connected (s, c, d) and  has_secret_clearance (s) and has_secret_clearance (d)

... additional  . . .
     **};
end Resolute_Security_Lib;
```

Information/Data Protection - Cryptography and Containment

Cryptography – model, assess, and assure security cryptographic methods and implementations, including encryption, authentication, authenticated encryption, key management, etc.

Implementation

- AADL augmented with property sets

Protected Containment – model, assess, and assure protected containment units such as protected address spaces, virtual machines, and partitions, data confidentiality, integrity (authenticity of data), availability

Implementation

- AADL augmented with property sets

Cryptography Property

```
property set Security_Properties is
                                                ...
                                                ...
-- cryptography (authentication, encryption, etc.) work in progress
--
encryption_scheme : Security_Properties::encryption_type applies to (port, virtual
bus, bus, memory, access, data);
--
  encryption_type : type record (
  encryption_form : enumeration (symmetric, asymmetric, hybrid,
authenticated_encryption, no_encryption, AEAD);
  algorithm : enumeration (tripledes, des, rsa, blowfish, twofish, aes, clear);
  private_key : aadlstring;  -- maybe better as an integer?
  public_key  : aadlstring;
  single_key          : aadlstring;
  operation_mode : Security_Properties::supported_operation_mode_types;
  authenticated_type  : enumeration (EtM, MtE, E_and_M);
  MAC_key: aadlstring;
    -- EtM is encrpt-then-MAC -- MAC is message authentication code
    -- MtE: MAC-then-Encrypt
    -- E&M: Encrypt and MAC
                                  );
```

Action/Command Protection

Model, assess, and assure access control of execution of actions/commands including

- Security kernels (e.g. seL4)
- Operating system security controls
- Specialized operating systems
- Protected Containment
- Virtual Trusted Execution Environments
- External Actions

Implementation

- AADL augmented with property sets

Security Architectures (Modeling)

Model, and analyze security architectures such as

- Multiple Independent Levels of Security (MILS, D-MILS)
- Secure kernels (e.g. seL4, MILS separation kernels)

Implementation

- AADL augmented as needed
- Adventium MILS Analysis Tool

Vulnerability Modeling and Analysis

Identify, model, and analyze security vulnerabilities

- Architecture/code defects
- Malicious code
- Misuse/improper use

Implementation

- AADL augmented as needed
- EMV 2
- Security vulnerability ontology

Threat/Attack Modeling

Capture and analyze threat/attack models including:

- Threat models (e.g. STRIDE)
- Attack/attacker models
- Attack surface models
- Attack trees
- Chain of events models
- Attack patterns
- Denial of Service

Implementation

- AADL augmented as needed
- EMV 2

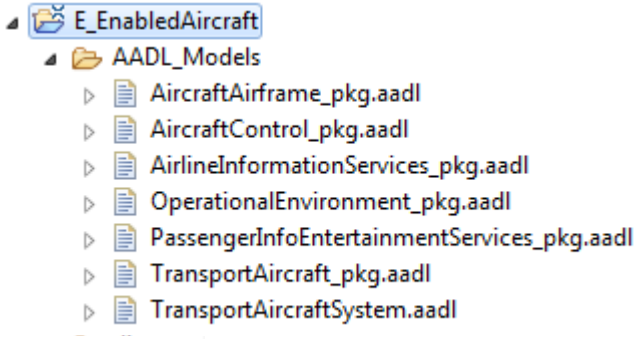
Summary and Status

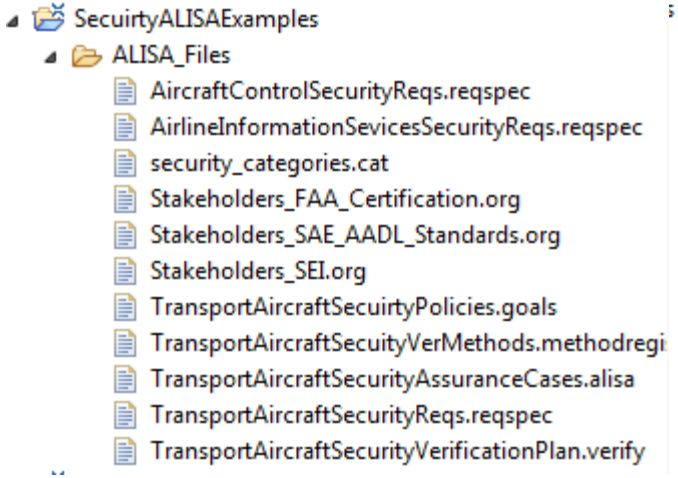
E-enabled aircraft model

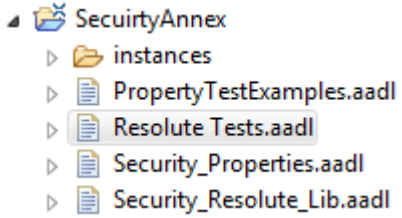
ALISA for Policy and Requirements

Protection and Encryption

- Property Sets
- Resolute Library







Comments/Questions