Security Annex Update

February 12, 2019

Dave Gluch

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Overview/Outline

Security Annex Standard

- provides guidance and support for using AADL for specifying, modeling, and analyzing secure system architectures

Information/Data Protection - Focus Discussions on properties relating to

- Security Information Levels
- Security Classification
- Encryption
- Encryption Keys
- Key Management
- Authorization and other topics as appropriate.

Example Models and Analyses

E_Enabled Aircraft Models

- Commercial Transport Aircraft
- Mission-Specific Aircraft (reconnaissance)

and ALISA Security Analysis Examples are available at

https://github.com/osate/examples.git

Supporting verification library files including property sets are available at

https://github.com/reteprelief/isse

Security Policies and Requirements

Security Annex

- A systematic security policy and requirements documentation and analysis approach that includes tool support
- Architecture-Led Incremental System Assurance (ALISA) workbench is an exemplar of an approach
  - Systematic documentation
  - Assurance Cases
  - Resolute and JAVA verification methods

Premise that system architecture security is a policy and requirements-driven endeavor

Issues of standards?

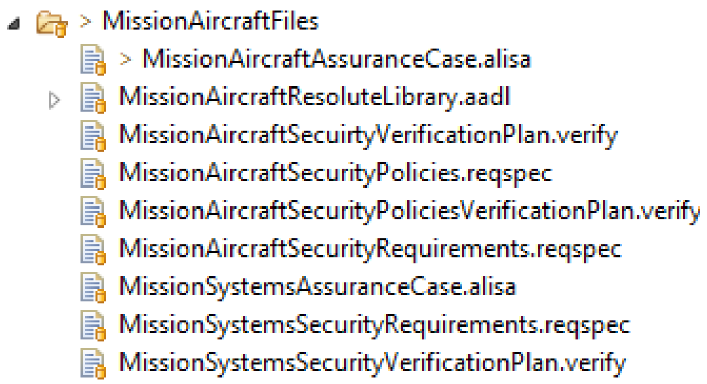
Security Policies and Requirements Documentation and Analysis

Documentation of Security Policies and Requirements

- ReqSpec of ALISA
- Naming convention to distinguish between policies and requirements

Policies and Requirements Verification – ALISA

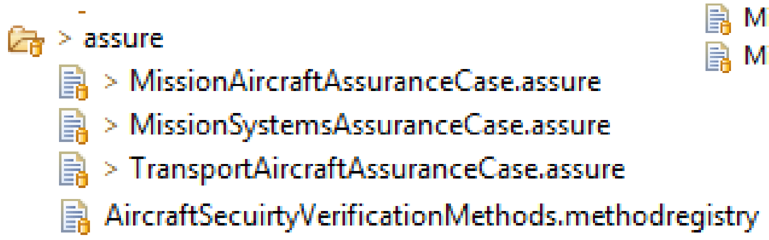
- Verification plans (.verify)
- Assurance cases (.alisa)
- Results (.assure)



```
> MissionAircraftFiles
    > MissionAircraftAssuranceCase.alisa
    MissionAircraftResoluteLibrary.aadl
    MissionAircraftSecuirtyVerificationPlan.verify
    MissionAircraftSecurityPolicies.reqspec
    MissionAircraftSecurityPoliciesVerificationPlan.verify
    MissionAircraftSecurityRequirements.reqspec
    MissionSystemsAssuranceCase.alisa
    MissionSystemsSecurityRequirements.reqspec
    MissionSystemsSecurityVerificationPlan.verify
```

Methods

- Resolute
- Java



```
> assure
    > MissionAircraftAssuranceCase.assure
    > MissionSystemsAssuranceCase.assure
    > TransportAircraftAssuranceCase.assure
    AircraftSecuirtyVerificationMethods.methodregistry
```

Overview/Outline

Security Annex Standard

- provides guidance and support for using AADL for specifying, modeling, and analyzing secure system architectures

Information/Data Protection - Focus Discussions on properties relating to

- Security Information Levels
- Security Classification
- Encryption
- Encryption Keys
- Key Management
- Authorization and other topics as appropriate.

Information Security Levels

Are properties applied to appropriate elements?

```
InformationSecurityLevel: aadlstring applies to (data, system, process, device,
abstract);
         --        --        --
     -- Information security caveats are treated as a single string.
          -- For example, a US information caveat value may be "//SI/TK//RELIDO"
          -- When the  security level and caveat classification are "Top Secret" and
          --  "//SI/TK//RELIDO" it specifies the concatenated classification
          --  "TOP SECRET//SI/TK//RELIDO"
          --
InformationSecurityCaveats: aadlstring applies to (data, system, process,
device,  abstract);
```

Information Security Levels apply to data classifiers and instances.

Other included categories are for conceptual/functional modeling.

Information Security Level Properties and AADL Component Categories

data: the data component, all its data subcomponents, and all its provides data access data classifiers have the specified information security level

system: all the system’s data subcomponents and all its data port, event data port and provides data access data classifiers have the specified information security level.

process: all the process’ data subcomponents and all its data port, event data port and provides data access data classifiers have the specified information security level.

device: all the devices’ data subcomponents all its data port and event data port data classifiers have the specified information security level

abstract: extensions have the specified security level consistent with the applicability and semantics of the information level security properties for the category of the extension

Security Classifications

Are properties applied to appropriate elements?

The security clearance property includes a principal security classification (e.g. Top Secret, Secret, Confidential) and a supplemental statement (e.g. specialized authorizations or restrictions).

```
SecurityClearance: inherit aadlstring applies to (system, device,
processor, virtual processor, thread, subprogram, process, abstract);
SecurityClearanceSupplement: inherit aadlstring applies to (system, device,
processor, virtual processor, thread, subprogram, process, abstract);
```

A secondary security clearance also includes a principal security classification and a supplemental statement.

```
SecondarySecurityClearance: inherit aadlstring applies to (system, device,
processor, virtual processor, thread, subprogram, process, abstract);
SecondarySecurityClearanceSupplement: inherit aadlstring applies to
(system, device, processor, virtual processor, thread, subprogram, process,
abstract);
```

No assumption is made about the relationship of the *Security_Clearance* property and the *Secondary_Security_Clearance* property.

Semantics of Security Clearances for AADL Categories - 1

system classifier: all instances have the specified clearance or clearances and all subcomponents have the specified clearance or clearances, consistent with the applicability and semantics of security clearance properties for the subcomponent category

device classifier: all instances have the specified clearance or clearances and all subcomponents have the specified clearance or clearances, consistent with the applicability and semantics of security clearance properties for the subcomponent category (i.e. abstract subcomponents have the specified security)

processor classifier: all instances have the specified clearance or clearances and all subcomponents have the specified clearance or clearances, consistent with the applicability and semantics of security clearance properties for the subcomponent category (i.e. virtual processor, abstract)

virtual processor classifier: all instances have the specified clearance or clearances and all subcomponents have the specified clearance or clearances, consistent with the applicability and semantics of security clearance properties for the subcomponent category (i.e. virtual processor, abstract)

```
SecurityClearance: inherit aadlstring applies to (system, device,
processor, virtual processor, thread, subprogram, process, abstract);
```

Semantics of Security Clearances for AADL Categories - 2

thread classifier: all instances have the specified clearance or clearances and all subcomponents have the specified clearance or clearances, consistent with the applicability and semantics of security clearance properties for the subcomponent category (i.e. subprogram, abstract)

subprogram classifier: all instances have the specified clearance or clearances and all subcomponents have the specified clearance or clearances, consistent with the applicability and semantics of security clearance properties for the subcomponent category (i.e. subprogram, abstract)

process classifier: all instances have the specified clearance or clearances and all subcomponents have the specified clearance or clearances, consistent with the applicability and semantics of security clearance properties for the subcomponent category (i.e. subprogram, thread, abstract)

abstract classifier: extensions have the specified clearance or clearances, consistent with the applicability and semantics of security clearance properties for the category of the extension

```
SecurityClearance: inherit aadlstring applies to (system, device,
processor, virtual processor, thread, subprogram, process, abstract);
```

Encryption Properties

The encryption property

- can be used as a higher level (redundant) indicator (e.g. in a system component to indicate that there are one or more contained elements that utilize encryption).
- Alternatively, it can be used early in the develop process specifying that encryption must be used for a model element. Once the details are defined, the encryption property can be replaced by the encryption_scheme property.

```
Encryption: inherit adlboolean applies to (data, abstract, system, bus,
memory, device, processor, virtual processor, virtual bus, connection);
  --  --
EncryptionScheme : inherit SecurityEnforcementProperties::EncryptionType
applies to (data, abstract, system, bus, memory, device, processor, virtual
processor, virtual bus, connection);
  --
```

Do we need both?
Perhaps rename the Boolean as *EncryptionRequired* and *EncryptionScheme* as *Encryption – see next slide*
Are properties applied to appropriate elements?

Encryption Properties - Alternative

```
EncryptionRequired: inherit aadlboolean applies to (data, abstract, system,
bus, memory, device, processor, virtual processor, virtual bus,
connection);
  --  --
Encryption : inherit SecurityEnforcementProperties::EncryptionType applies
to (data, abstract, system, bus, memory, device, processor, virtual
processor, virtual bus, connection);
```

Add the option "`to_be_specified`" to the encryption type enumeration field "`encryption_form.`"

```
encryption_form : enumeration (symmetric, asymmetric, hybrid,
authenticated_encryption, no_encryption, AEAD, to_be_specified);
```

Semantics of Encryption for AADL Categories - 1

When encryption is declared for a data classifier or data instance, the data is encrypted as specified and all subcomponents have the specified encryption, consistent with the applicability and semantics of encryption properties for the subcomponent category (i.e. data subcomponents and abstract).

When encryption is declared for an abstract classifier, its extensions have the specified encryption consistent with the applicability and semantics of encryption properties for the category of the extension.

When encryption is declared for a system classifier or instance all subcomponents have the specified encryption, consistent with the applicability and semantics of encryption properties for the subcomponent category.

When encryption is declared for a bus classifier or instance encryption is provided as specified for all data transmitted through it* and for all subcomponents, consistent with the applicability and semantics of encryption properties for the subcomponent category (i.e. virtual bus, abstract)

**it == an instance of the bus classifier or the bus instance*

Semantics of Encryption for AADL Categories - 2

When encryption is declared for a virtual bus classifier or instance encryption is provided as specified for all data transmitted through it* and for all subcomponents, consistent with the applicability and semantics of encryption properties for the subcomponent category (i.e. virtual bus, abstract).

When encryption is declared for a memory classifier or instance encryption is provided as specified for all data contained in it* and for all subcomponents, consistent with the applicability and semantics of encryption properties for the subcomponent category (i.e. memory, bus, abstract).

When encryption is declared for a device classifier or instance encryption is provided as specified for all data contained in it* and for all subcomponents, consistent with the applicability and semantics of encryption properties for the subcomponent category (i.e. bus, virtual bus, data, abstract).

When encryption is declared for a processor classifier or instance encryption is provided as specified for all data within it* and for all subcomponents, consistent with the applicability and semantics of encryption properties for the subcomponent category (i.e. memory, bus, virtual processor, virtual bus, abstract).

**it == an instance of the … classifier or the … instance*

Semantics of Encryption for AADL Categories - 3

When encryption is declared for a virtual processor classifier or instance is provided as specified for all data within it* and for all subcomponents, consistent with the applicability and semantics of encryption properties for the subcomponent category (i.e. virtual processor, virtual bus, abstract).

When encryption is declared for a connection instance the supporting transmission components must provide the specified encryption for all data transmitted through the connection. [Applying encryption to connections allows the specification of encryption at the functional level, independent of any execution platform declarations that support the connection. A check can be performed to verify that the encryption specified on the connection is provided by the underlying execution platform.]

Note that redundant encryption declarations can be specified for an architecture model. For example, an encryption scheme can be specified for a system, perhaps early in the development process. Later encryption may be specified for subcomponents of that system. Consequently, consistency should be established among encryption declarations for the system and any contained components.

Should data and event data ports have the encryption property? Since ports are logical connection points and data and event data ports can be typed with data classifiers, the encryption property should be associated with the data not the port. Data is encrypted independently of its role in specifying the data transmitted through the port.

Encryption Type

```
EncryptionType : type record (
encryption_form : enumeration (symmetric, asymmetric, hybrid,
authenticated_encryption, no_encryption, AEAD);
 algorithm : enumeration (tripledes, des, rsa, blowfish, twofish, aes, D_H,
 ECC, clear);
 key_type: SecurityEnforcementProperties::keyType;
 private_key : SecurityEnforcementProperties::keyInstance;
 public_key  : SecurityEnforcementProperties::keyInstance;
 single_key   : SecurityEnforcementProperties::keyInstance;
 MAC_key: SecurityEnforcementProperties::keyInstance
 authentication_type : enumeration (EtM, MtE, E_and_M, AEAD);
 );
```

Are the fields appropriate and complete?

Overview/Outline

Security Annex Standard

- provides guidance and support for using AADL for specifying, modeling, and analyzing secure system architectures

Information/Data Protection - Focus Discussions on properties relating to

- Security Information Levels
- Security Classification
- Encryption
- Encryption Keys
- Key Management
- Authorization and other topics as appropriate.

Encryption Keys

a key is declared as an AADL abstract component

```
abstract key
end key;
-- extend abstract key to data classifiers
data symmetricKey extends key
properties
SecurityEnforcementProperties::keyLength => 256 bits;
end symmetricKey;
data publicKey extends key
end publicKey;
data privateKey extends key
end privateKey;
```

Abstract key allows flexibility but?
Do we want a value property for a key?

Key Related Properties

```
keyLength: Size aadlinteger applies to (abstract, data);
cryptoPeriod: Time applies to (abstract, data);
keyType: type classifier (abstract, data);
keyInstance: type reference (data);
textType: enumeration (plainText, cipherText) applies to (abstract, data);
```

Key Management

Security Key Component

- Extensible to specific key components
- Key properties (e.g. length, cryptoPeriod)

Key/Certificate

- Generation
- Storage
- Distribution control
- Destruction
- Replacement

Should we include a certificate component?
For example:

```
abstract CertificateAbs
end CertificateAbs;
data Certificate extends CertificateAbs
end Certificate;
data implementation Certificate.SSL_TLS
subcomponents
Subject: data subject.certificate;
Issuer: data issuer.certificate;
PeriodOfValidity: data periodOfValidity.certificate;
AdminInformation: data adminInformation.certificate;
ExtendedInformatio: data extendedInformation.certificate;
end Certificate.SSL_TLS;
```

Authentication Property

```
Authentication: SecurityEnforcementProperties::authenticationType applies
to (abstract, system, bus, memory, device, processor, virtual processor,
virtual bus);

    authenticationType: type record

    (

    AuthenticationAccessType: enumeration (NoValue, single_password,
smart_card,  ip_addr, two_factor, multi_layered, bio_metric,
to_be_specified);

    AuthenticationProtocol: enumeration (NoValue, cert_services, EAP, PAP,
SPAP, CHAP, MS_CHAP, Radius, IAS, Kerberos, SSL, NTLM, to_be_specified) ;

    );
```

Summary

Security Annex Summary

- Guidance and support for using AADL for specifying, modeling, and analyzing secure system architectures
- Core AADL with security-specific properties
- ALISA for policies and requirements capture
- Verification of requirements and other analyses with
  - ALISA assurance cases (Resolute and Java based methods)
  - Libraries of claims and methods
- Representative Examples
- Custom plugins (modeling and analysis) - TBD

What analyses is supported?