Security Annex Update

October 31, 2019

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.

DM19-1128

Some Suggestions

Jerome suggested splits the document into

- AADL security annex (the property sets and patterns)- standard
- its usage combining OSATE/ALISA and this annex - tech report

Charlie Payne/Steve Vestal: split in some way as well

Option(s) for discussion

Security Annex Document (specification)

- Specify security specific aspects of an architecture
- Include example analysis of the architecture

Security Annex Document (system analysis)

- Security analysis of the architecture
  - are there constructs specified to meet security policies/requirements
  - are the polies/constructs correct and appropriate for the system

Security Annex Document (threat analysis)

- Analysis of Vulnerabilities/Threats/Attacks

Noteworthy Changes from Previous Version

No inheritance of Encryption, Data_Authentication, and Subject_Authentication properties

Information Security level and Security Level caveats are now enumerations

Removed key_length field in Encryption property

- key length is declared only in a key classifier.
- eliminates issue of key length declarable in two locations.

Key classifiers are data components only

Added end-to-end secure path property.

Eliminated Security_Domain and Need_to_Know_Domains

Overview

Security Annex Standard

- presents the OSATE/ALISA environment as an exemplar for comprehensive policy/requirements documentation analysis and verification.
- includes security properties for classification and enforcement
  - property sets are user modifiable
  - example security components (e.g. key classifier)
- includes exemplar analysis methods and claims
  - Resolute and JAVA

Security Policies and Requirements

A security policy and requirements documentation and analysis approach that includes tool support

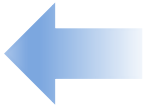
Architecture-Led Incremental System Assurance (ALISA) workbench is an exemplar of an approach

- Systematic documentation
- Assurance Cases
- Resolute and JAVA verification methods

Security Annex Properties

Property sets that can be edited by a user

- Security Classification Property Set
  - Security Clearances (subjects)
  - Information Security Levels (objects)
  - Security Levels (subjects and objects)
  - Trusted Classification
- Security Enforcement Property Set
  - Encryption properties
  - Data Authentication
  - Subject Authentication



Security Clearances

Principal security clearance and a supplemental statement

```
Security_Clearance: inherit enumeration (TopSecret, Secret, Confidential,
No_Clearance)
applies to (system, device, processor, virtual processor, thread, thread group,
subprogram, subprogram group, process, abstract);
--
Security_Clearance_Supplement: inherit aadlstring
applies to (system, device, processor, virtual processor, thread, thread group,
subprogram, subprogram group, process, abstract);
```

--Secondary security clearance and a supplemental statement

```
Secondary_Security_Clearance: inherit  enumeration (TopSecret, Secret, Confidential,
No_Clearance)
applies to (system, device, processor, virtual processor, thread, thread group,
subprogram, subprogram group, process, abstract);

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

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

Enumerations are modifiable by users.

Information Security Levels

```
Information_Security_Level: inherit enumeration (TopSecret, Secret,
Confidential, Unclassified)
 applies to (data, port, system, process, device, abstract);

Information_Security_Caveats: inherit list of enumeration (FOUO, NOFORN,
NOCONTRACTOR, PROPIN, IMCON, ORCON)
applies to (data, port, system, process, device, abstract);
```

Enumerations are modifiable by users.

Generalized Security Levels and Trusted Components

When no differentiation between subject and object is needed.

```
Security_Level: inherit enumeration (TopSecret, Secret, Confidential,
Unclassified) applies to ( system, processor, virtual processor, thread,
thread group, subprogram, subprogram group,  data, port, process, device,
abstract);

Security_Level_Caveats: inherit list of enumeration (FOUO, NOFORN,
NOCONTRACTOR, PROPIN, IMCON, ORCON)
applies to (system, processor, virtual processor, thread, thread group,
subprogram, subprogram group,  data, port, process, device, abstract);
```

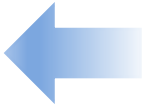
Declare a trusted component

```
Trusted : aadlboolean applies to (system, process, thread, thread group,
subprogram, subprogram group, processor, virtual processor, bus, virtual
bus, abstract);
```

Security Annex Properties

Property sets that can be edited by a user

- Security Classification Property Set
  - Security Clearances (subjects)
  - Information Security Levels (objects)
  - Security Levels (subjects and objects)
  - Trusted Classification
- Security Enforcement Property Set
  - Encryption properties
  - Data Authentication
  - Subject Authentication



Encryption Property

```
Encryption: record (
description: aadlstring;
-- an informal description of the encryption
encryption_form: enumeration (no_encryption, symmetric, asymmetric, hybrid,
authenticated_encryption, authentication_only, to_be_specified);
-- if the encryption form is hybrid both symmetric and asymmetric are used.
encryption_algorithm: list of enumeration (no_encryption, OTP, DES, TripleDES, AES,
RSA, ECC, to_be_specified);
-- the mode and algorithm listings must correlate
encryption_mode: list of enumeration (no_encryption, ECB, CBC, CFB, CTR, GCM,
CBC_MAC, to_be_specified);
padding: enumeration (no_padding, OAEP, to_be_specified);
authenticated_encryption_type: enumeration (no_authenticated_encryption, GCM,
CBC_MAC, Encrypt_then_MAC, MAC_then_Encrypt, Encrypt_and_MAC, AEAD, signcryption,
double_RSA);
key_type: list of SecurityEnforcementProperties::key_classifier;
private_key: SecurityEnforcementProperties::key_instance;  -- references an instance
public_key: SecurityEnforcementProperties::key_instance; -- references an instance
single_key: SecurityEnforcementProperties::key_instance;  -- references an instance
) applies to (data, port, abstract, system, bus, memory, device, processor,
      virtual processor, virtual bus, connection, process, thread, flow);
```

Encryption Key Properties

```
data key
end key;
-- extend abstract key to data classifiers
data symmetricKey extends key
properties
SecurityEnforcementProperties::keyLength => 2048 bits;
end symmetricKey;

data publicKey extends key
end publicKey;

data privateKey extends key
end privateKey;
```

Key Related Properties

```
Key_Length: Size applies to (data);
Crypto_Period: Time applies to (data);
Key_Classifier: type classifier (data);
Key_Instance: type reference (data);
Text_Type: enumeration (plainText, cipherText) applies to (data);
Key_Distribition_Method: enumeration (public_broadcast_channel,
public_one_to_one_channel, encrypted_channel, QKD, direct_physical_exchange,
courier) applies to (all);
```

Data Authentication

Used with Authenticated Encryption

```
Data_Authentication:  record
(
  description: aadlstring;
  authentication_form : enumeration (no_authentication, MAC, MIC, signature,
signcryption, double_RSA, to_be_specified);
authentication_algorithm: enumeration (no_authentication, RSA, ElGamal, DSA,
CBC_MAC, GCM, HMAC, UMAC, to_be_specified);
padding: enumeration (no_padding, OAEP, to_be_specified);
-- key_Length is declared in the authentication key type classifier or
-- in the classifier for the authentication key instance or for the key instance.
hash_Length: Size; -- optional, if the message is hashed before authentication. Does
not apply to authenticated encryption.
hash_algorithm : enumeration (no_hash, MD5, SHA1, SHA256, SHA512, SHA3,
to_be_specified);
authentication_key_type: list of SecurityEnforcementProperties::key_classifier;
authentication_key: SecurityEnforcementProperties::key_Instance;
)
applies to (data, port, abstract, system, bus, memory, device, processor,
virtual processor, virtual bus, connection, process, thread, flow);
```

Subject Authentication Property

```
Subject_Authentication: record
    (
    description: aadlstring;
    authentication_access_type: enumeration (no_authentication, single_password,
smart_card,  ip_addr, two_factor, multi_layered, bio_metric, to_be_specified);
    two_and_multi_layered_factors: list of enumeration (no_multifactor, smart_card,
token, PIV, OTP, biometric, multi_layered, to_be_specified);
    -- the listing is such that the initial factor required for authentication is
listed first, the second factor is listed second, etc.
    authentication_protocol: enumeration (no_authentication, cert_services, EAP,
PAP, SPAP, CHAP, MS_CHAP, Radius, IAS, Kerberos, SSL, TLS, NTLM, to_be_specified) ;
    authentication_role: enumeration (no_authentication, authenticator, accessor,
provider, requirer, mutual);
    )
    applies to (abstract, system, process, thread,device, processor, virtual
processor, connection, bus, virtual bus, flow);
```

Declares that a subject (component instance) can participate or participates in authentication as specified, including authentication negotiations employing the specified authentication protocol, or that the component (e.g. a bus or virtual bus) supports the authentication specified.

Cross Domain Solution Example

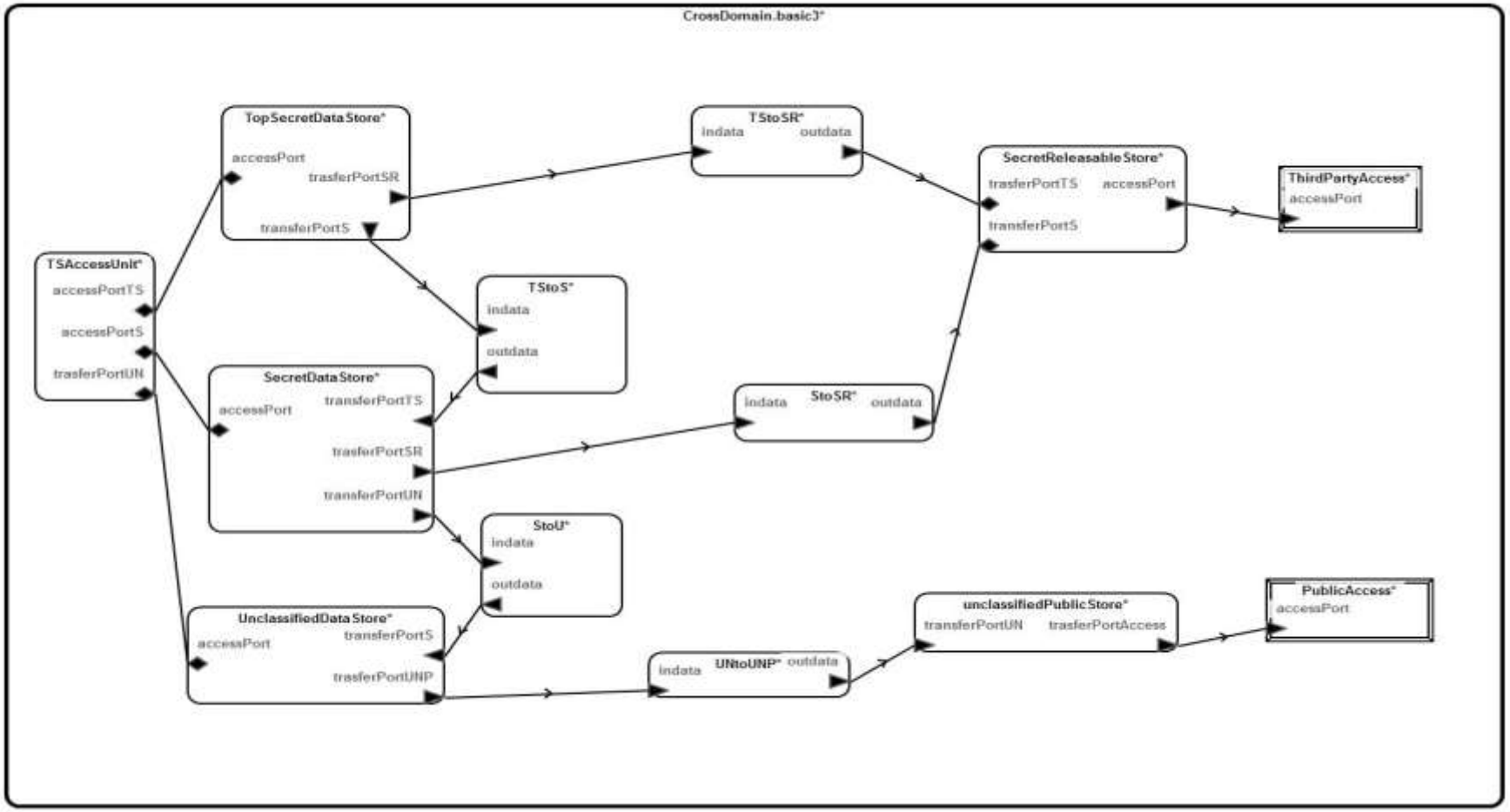
Cross Domain Solution

- three primary data stores (top secret, secret, and unclassified)
- two data stores for data that can be released (secret releasable and unclassified for public release).
- downgrading filters that downgrade top secret to secret, secret to unclassified, top secret to secret releasable, secret to secret releasable, and unclassified to unclassified public release.
- a super controller (subject) who can access and modify all three data stores

Examples are available at
https://github.com/osate/examples.git

Supporting files are available at
https://github.com/reteprelief/isse

Cross Domain AADL Model

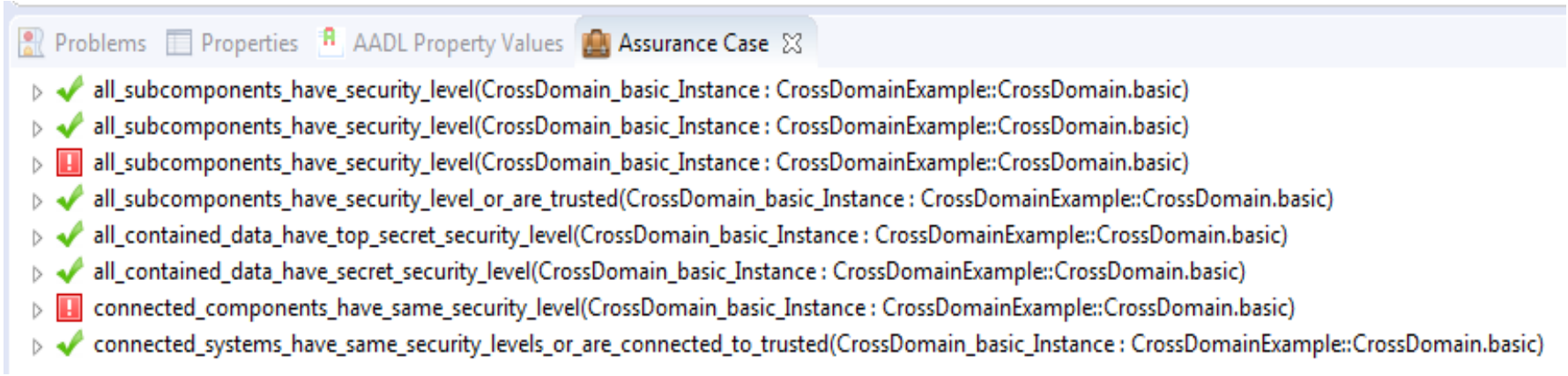


Security Level Property Associations

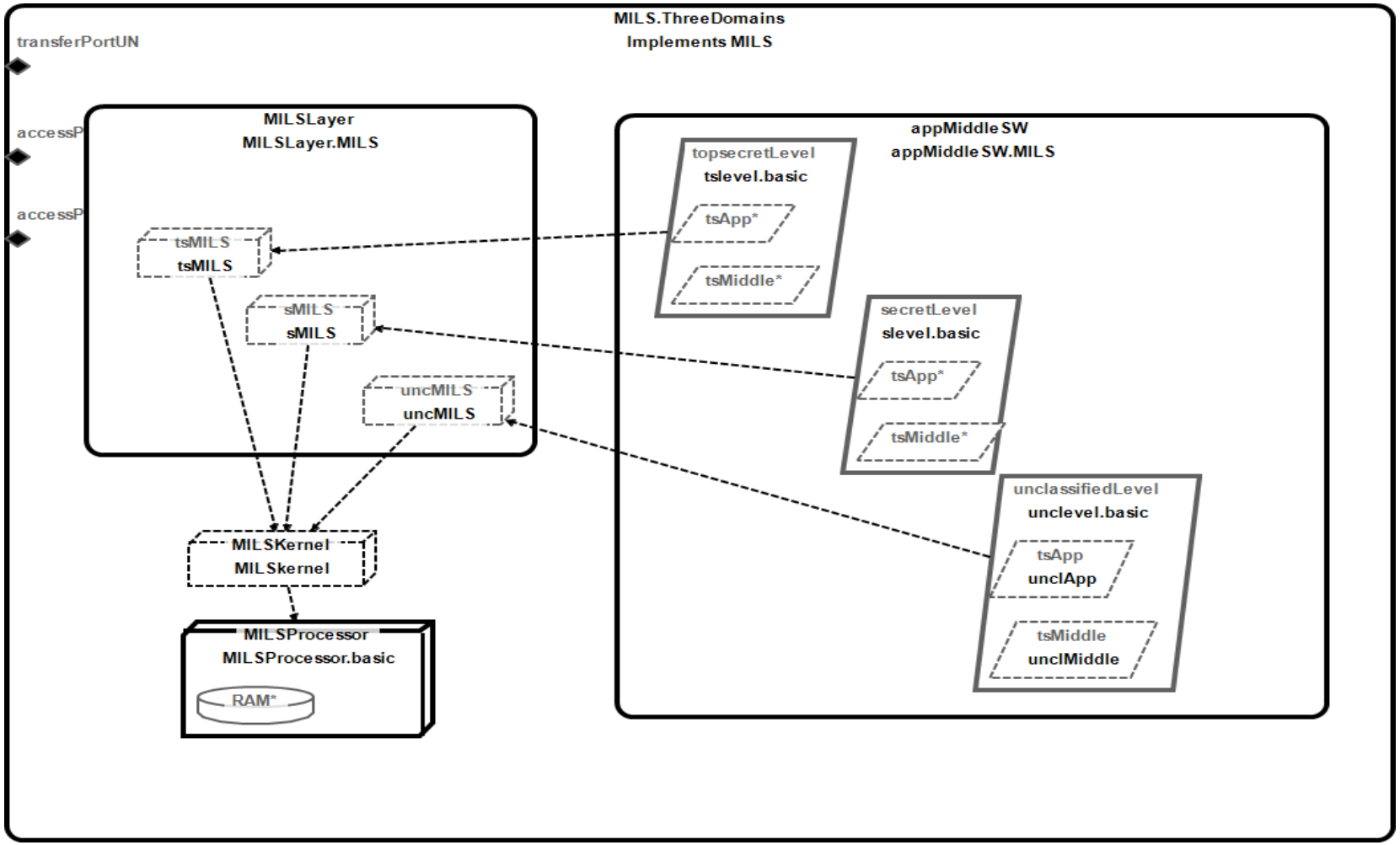
```
SecurityClassificationProperties::Security_Level => TopSecret applies to
TopSecretDataStore;
SecurityClassificationProperties::Security_Level => TopSecret applies to
TopSecretDataStore.topsecretdata;
SecurityClassificationProperties::Security_Level => Secret  applies to
SecretDataStore;
SecurityClassificationProperties::Security_Level => Unclassified  applies to
UnclassifiedDataStore;
SecurityClassificationProperties::Security_Level => Unclassified applies to
unclassifiedPublicStore;
SecurityClassificationProperties::Security_Level => Secret applies to
SecretReleasableStore; SecurityClassificationProperties::Security_Level =>
Unclassified  applies to PublicAccess;
SecurityClassificationProperties::Security_Level => Secret  applies to
ThirdPartyAccess;
                              -- trusted components (filters)
SecurityClassificationProperties::Trusted => true applies to TStoS;
SecurityClassificationProperties::Trusted => true applies to StoU;
SecurityClassificationProperties::Trusted => true applies to UNtoUNP;
SecurityClassificationProperties::Trusted => true applies to StoSR;
SecurityClassificationProperties::Trusted => true applies to TStoSR;
SecurityClassificationProperties::Trusted => true applies to TSAccessUnit;
```

Resolute Claims and Results

```
                    -- security level checks
prove all_subcomponents_have_security_level(this.TopSecretDataStore) -- should be true
prove all_subcomponents_have_security_level(this.SecretDataStore) -- should be true
prove all_subcomponents_have_security_level (this)  -- not true because some are trusted
prove all_subcomponents_have_security_level_or_are_trusted (this)  -- should be true
prove all_contained_data_have_top_secret_security_level(this.TopSecretDataStore) -- should
be true
prove all_contained_data_have_secret_security_level(this.SecretDataStore) -- should be true
                                        -- security connection checks
prove connected_components_have_same_security_level (this) -- should be false (some trusted)
prove connected_systems_have_same_security_levels_or_are_connected_to_trusted(this) --
should be true
```



MILS Architecture of the TSAccessUnit



MILS Three Domain Implementation

```
system implementation MILS.ThreeDomains
          subcomponents
                    appMiddleSW: system appMiddleSW.MILS;
                    MILSLayer: system MILSLayer.MILS;
                    MILSKernel: virtual processor MILSKernel;
                    MILSProcessor: processor MILSProcessor.basic;
properties                          --
          -- Schedule the partitions on a fixed timeline
Scheduling_Protocol => (FixedTimeline) applies to MILSKernel;
--        Bind the applications to the virtual processors
Actual_Processor_Binding => (reference (MILSLayer.tsMILS)) applies to appMiddleSW.topsecretLevel;
Actual_Processor_Binding => (reference (MILSLayer.sMILS)) applies to appMiddleSW.secretLevel;
Actual_Processor_Binding => (reference (MILSLayer.uncMILS)) applies to
appMiddleSW.unclassifiedLevel;
-- Bind the virtual processors to the separation kernel
          Actual_Processor_Binding => (reference (MILSKernel)) applies to MILSLayer.tsMILS;
          Actual_Processor_Binding => (reference (MILSKernel)) applies to MILSLayer.sMILS;
          Actual_Processor_Binding => (reference (MILSKernel)) applies to MILSLayer.uncMILS;
-- Bind MILS separation kernel to the hardware processor
          Actual_Processor_Binding => (reference (MILSProcessor)) applies to MILSKernel;

          end MILS.ThreeDomains;
```