Security Annex Update

January 28, 2020

Dave Gluch

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Document Markings

Copyright 2020 Carnegie Mellon University.

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.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM20-0057

Noteworthy Changes from Previous Version

Focused the content of standard to custom property sets and packages.

Restructured the Security Classification Properties as record types and eliminated inheritance.

Revised naming and structure of Security Enforcement Properties (e.g. Data_Security as a record type)

© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Overview

Security Annex Standard

- security properties for classification and enforcement
  - basic classification, encryption, authentication specifications
  - custom security components (e.g. keys, certificates)
- security specification examples
- a basic explanation of the analyses that are possible with the security properties (possibly examples??)

Technical Report or White Paper

- example using OSATE/ALISA and the security annex
- example security architecture models and analyses

© 2020 Carnegie Mellon University
[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

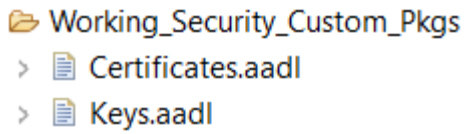
Security Property Sets and Custom Packages

Property Sets

- Security_Classification_Properties
  - Classification_Specifications (set of constants)
- Security_Enforcement_Properties
  - Encryption_Specifications (set of constants)
  - Authentication_Specifications (set of constants)

Custom Packages

- Keys (key classifiers)
- Certificates (certificate classifiers)



© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Security Annex Properties

Enumerations in property sets can be edited by a user

- Security Classification Property Set
  - Restructured Formatting
  - Security Clearances (subjects)
  - Information Security Levels (objects)
  - Security Levels (subjects and objects)
  - Trusted Classification
- Security Enforcement Property Set
  - Restructured Naming and Formatting
  - Data Security
  - Data Security Specification
  - Subject Authentication
  - Secure Flows

© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Security Clearances Changes

Principal Security Clearance

```
Security_Clearance: Security_Classification_Properties::level_type
applies to (system, device, processor, virtual processor, thread, thread group,
subprogram, subprogram group, process, abstract)

Secondary_Security_Clearance:Security_Classification_Properties::level_type
applies to (system, device, processor, virtual processor, thread, thread group,
subprogram, subprogram group, process, abstract);
```

Security Clearance Type Declaration (enumerations modifiable by users)

```
level_type : type record
         (
            description: aadlstring;
            level: enumeration (TopSecret, Secret, Confidential, Unclassified) ;
         ) ;
```

Supplemental Security Clearances

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


© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Classification Specifications

Excerpt from the property set of classification specifications, which are declared as constants of level_type

```
property set Classification_Specifications is
         with Security_ Classification_Properties;

TopSecret: constant Security_Classification_Properties::level_type =>
          [
             level=> TopSecret;
             description => “The highest level of security clearance that provides
                     access to Top Secret, Secret, and Confidential Information.";
          ];
Secret: constant Security_Classification_Properties::level_type =>
          [
             level=>Secret;
             description => "This level of security classification provides access
to Secret Information and, as needed, to Confidential Information."
          ];
```


© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Information Security Levels

```
Information_Security_Level: Security_Enforcement_Properties::level_type
 applies to (data, port, system, process, device, abstract);

Information_Security_Caveats: list of
                              Security_Properties_Revised::caveat_type
applies to (data, port, system, process, device, abstract);
```

Type declarations for level and caveat

```
level_type: type record
(
    description: aadlstring;
    level: enumeration (TopSecret, Secret, Confidential, Unclassified);
);
caveat_type: type record
(
    description: aadlstring;
    caveat: enumeration (FOUO, NOFORN, NOCONTRACTOR, PROPIN, IMCON, ORCON,
                         NORELEASE, RELIDO, REL_TO_PUBLIC, REL_TO_CONTRACTOR);
) ;
```

Same level_type as Security Clearance

Enumerations modifiable by users. Levels run left to right: highest to lowest.

© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Security Classification Specifications

Excerpt from the property set of classification specifications, which are declared as constants of level_type

```
property set Classification_Specifications is
         with Security_ Classification_Properties;


 TopSecretInformation: constant Security_Classification_Properties::level_type =>
          [
             level=> TopSecret;
             description => "This is the is the highest level of classified
information where such material would cause exceptionally grave damage to national
security if made publicly available.";
          ];
 SecretInformation: constant Security_Classification_Properties::level_type =>
          [
             level=>Secret;
             description => "Would cause serious damage to national security if it
were publicly available.";
          ];
```


© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Generalized Security Levels and Trusted Components

When no differentiation between subject and object is needed.

```
Security_Level:  Security_Classification_Properties::level_type
 applies to ( system, processor, virtual processor, thread, thread group,
subprogram, subprogram group,  data, port, process, device, abstract);
--
Security_Level_Caveats:list of
                        Security_Classification_Properties::caveat_type
applies to (system, processor, virtual processor, thread, thread group,
subprogram, subprogram group,  data, port, process, device, abstract);
```

Trusted Component

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


© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Security Annex Properties

Enumerations in property sets can be edited by a user

- Security Classification Property Set
  - Restructured Formatting
  - Security Clearances (subjects)
  - Information Security Levels (objects)
  - Security Levels (subjects and objects)
  - Trusted Classification
- Security Enforcement Property Set
  - Restructured Naming and Formatting
  - Data Security
  - Data Security Specification
  - Subject Authentication
  - Secure Flows

© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Data Security Properties

```
-- The Data Security property specifies the basic type of data protection.
Data_Security: record
(
description: aadlstring;
data_security_type: enumeration (no_protection, encryption, authentication,
authenticated_encryption, TLS);
) applies to (data, port, abstract, system, bus, memory, device, processor, virtual
processor, virtual bus, connection, process, thread, flow);

-- The Data_Security_Specification property specifies the details.

Data_Security_Specification:  list of
Security_Enforcement_Properties::Security_Specification_Type
 applies to (data, port, abstract, system, bus, memory, device, processor,
virtual processor, virtual bus, connection, process, thread, flow);
--
Security_Specification_Type: type record (
   description: aadlstring;
   encryption: Security_Enforcement_Properties::Encryption_Specification_Type;
   authentication: Security_Enforcement_Properties::Data_Authentication_Type;
   authenticated_encryption_type: enumeration (GCM, CBC_MAC, encrypt_then_MAC,
   MAC_then_encrypt, encrypt_and_MAC, AEAD, signcryption);
   );
```


© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Security Specification Type Declaration

```
Security_Specification_Type: type record (
   description: aadlstring;
   encryption: Security_Enforcement_Properties::Encryption_Specification_Type;
   authentication: Security_Enforcement_Properties::Data_Authentication_Type;
   authenticated_encryption_form: enumeration (GCM, CBC_MAC, encrypt_then_MAC,
MAC_then_encrypt, encrypt_and_MAC, AEAD, signcryption););

Encryption_Specification_Type: type record (
description: aadlstring;
algorithm_name: enumeration (OTP, DES, TripleDES, AES, RSA, ECC);
encryption_mode: enumeration (ECB, CBC, CFB, CTR, GCM, Blowfish, TwoFish);
padding: enumeration (no_padding, block_cipher, OAEP);
key_classifier: Security_Enforcement_Properties::Key_Classifier;);

Data_Authentication_Type: type record (
description: aadlstring;
authentication_algorithm: enumeration (RSA, ElGamal, DSA, ECC, ECDSA, CBC_MAC, GCM,
HMAC, CMAC, OMAC, UMAC, Poly1305_AES);
authentication_key_type: list of Security_Enforcement_Properties::Key_Classifier;
hash_algorithm: enumeration (MD5, SHA1, SHA2, SHA3, RIPEMD, Whirlpool, ChaCha20,
BLAKE);
hash_length: Size;
);
```


© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Encryption and Authentication Specifications - excerpts

```
property set Encryption_Specifications is
with Security_Enforcement_Properties;
with Keys;

AES256CBC: constant Security_Enforcement_Properties::Encryption_Specification_Type
=> [
algorithm_name => AES;
encryption_mode => CBC;
key_classifier => classifier (Keys::Key256);
];

RSA2048: constant Security_Enforcement_Properties::Encryption_Specification_Type =>[
algorithm_name => RSA;
key_classifier => classifier (Keys::Key2048);
];
```

```
property set Authentication_Specifications is
with Security_Enforcement_Properties;

HMAC512: constant Security_Enforcement_Properties::Data_Authentication_Type =>[
description =>" Defines a message authentication code (MAC) using a cryptographic
hash function and a secret cryptographic key.";
        authentication_algorithm => HMAC;
        hash_algorithm => SHA3;
        hash_length => 512 bits;
];
```


© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Encryption Keys Package and Properties

```
package Keys
public
with Security_Enforcement_Properties;
data Key128
properties
Security_Enforcement_Properties::Key_Length => 128 bits;
end Key128;
data Key256
properties
Security_Enforcement_Properties::Key_Length => 256 bits;
end Key256;
data Key1024
properties
Security_Enforcement_Properties::Key_Length => 1024 bits;
end Key1024;
data Key2048
properties
Security_Enforcement_Properties::Key_Length => 2048 bits;
end Key2048;
end Keys;
```

Key Related Properties

```
Key_Classifier: type classifier (data);
Key_Instance: type reference (data);
Key_Length: Size applies to (data);
Crypto_Period: Time applies to (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);
```


© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

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);
   two_and_multi_layered_factors: list of enumeration (no_multifactor, smart_card,
                                        token, PIV, OTP, biometric, multi_layered);
-- 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);
   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.

© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

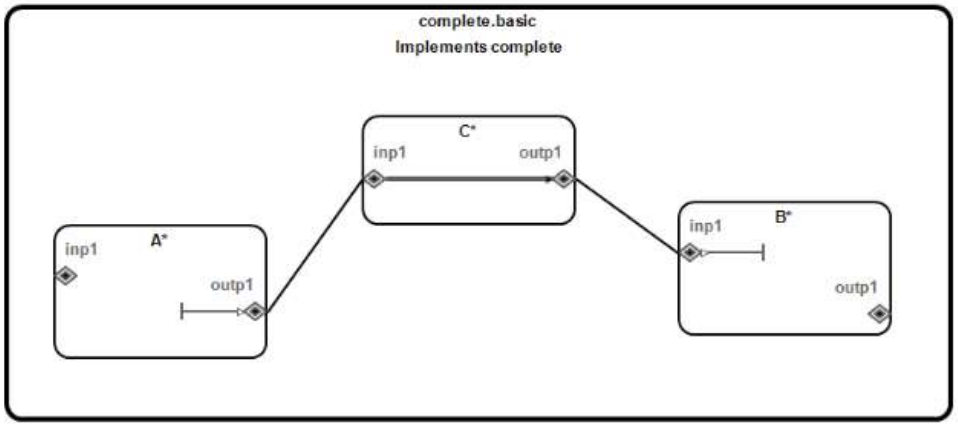
Secure Flows

The Secure_Flow property specifies that the data in an end-to-end flow is not altered by any element along the flow.

```
Secure_Flow: aadlboolean applies to (flow);
```

```
system implementation complete.basic
subcomponents
A: system A;
B: system B;
C: system C;  -- a pass through of data
connections
conn1: port A.outp1 <-> C.inp1;
conn2: port C.outp1 <-> B.inp1;
--
flows
secure_path: end to end flow A.secure_sourceA -> conn1 -> C.secure_pathC ->
conn2 -> B.secure_sinkB;

properties
-- declare secure flow from A to B
Security_Enforcement_Properties::Secure_Flow => true applies to secure_path;
```



© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

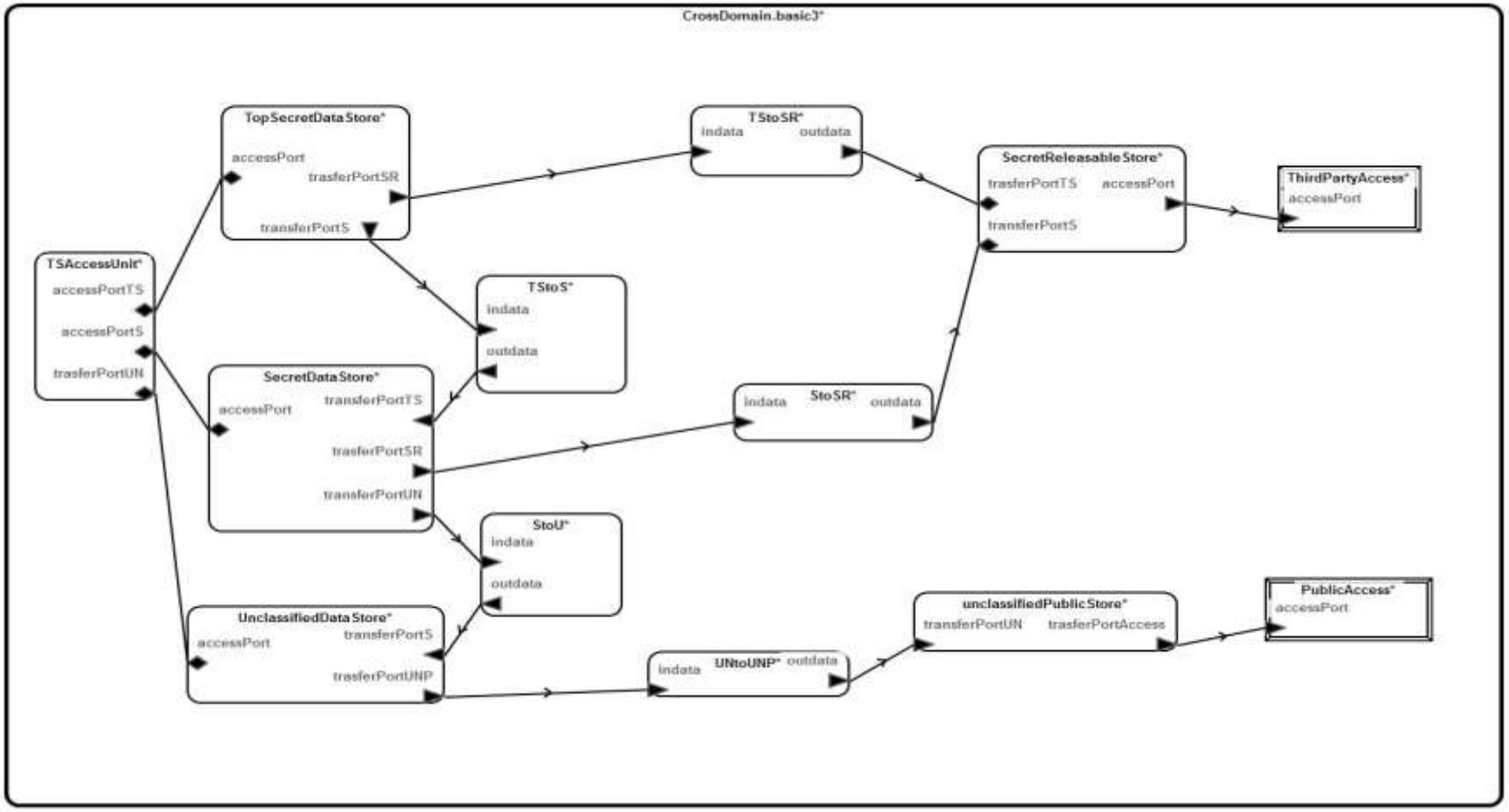
Cross Domain Solution Example

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

© 2020 Carnegie Mellon University
[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Cross Domain AADL Model

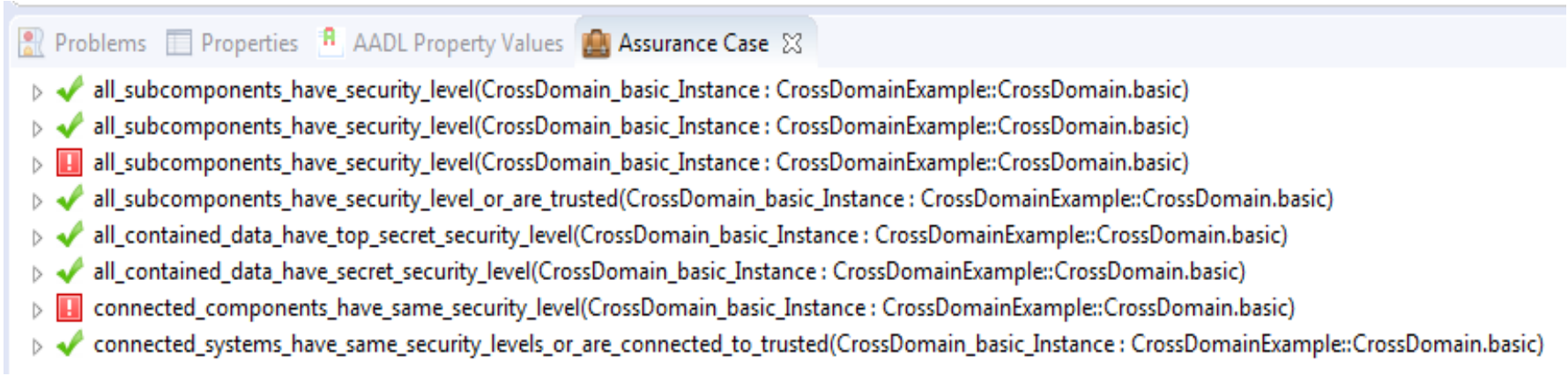


© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

Resolute Claims and Results

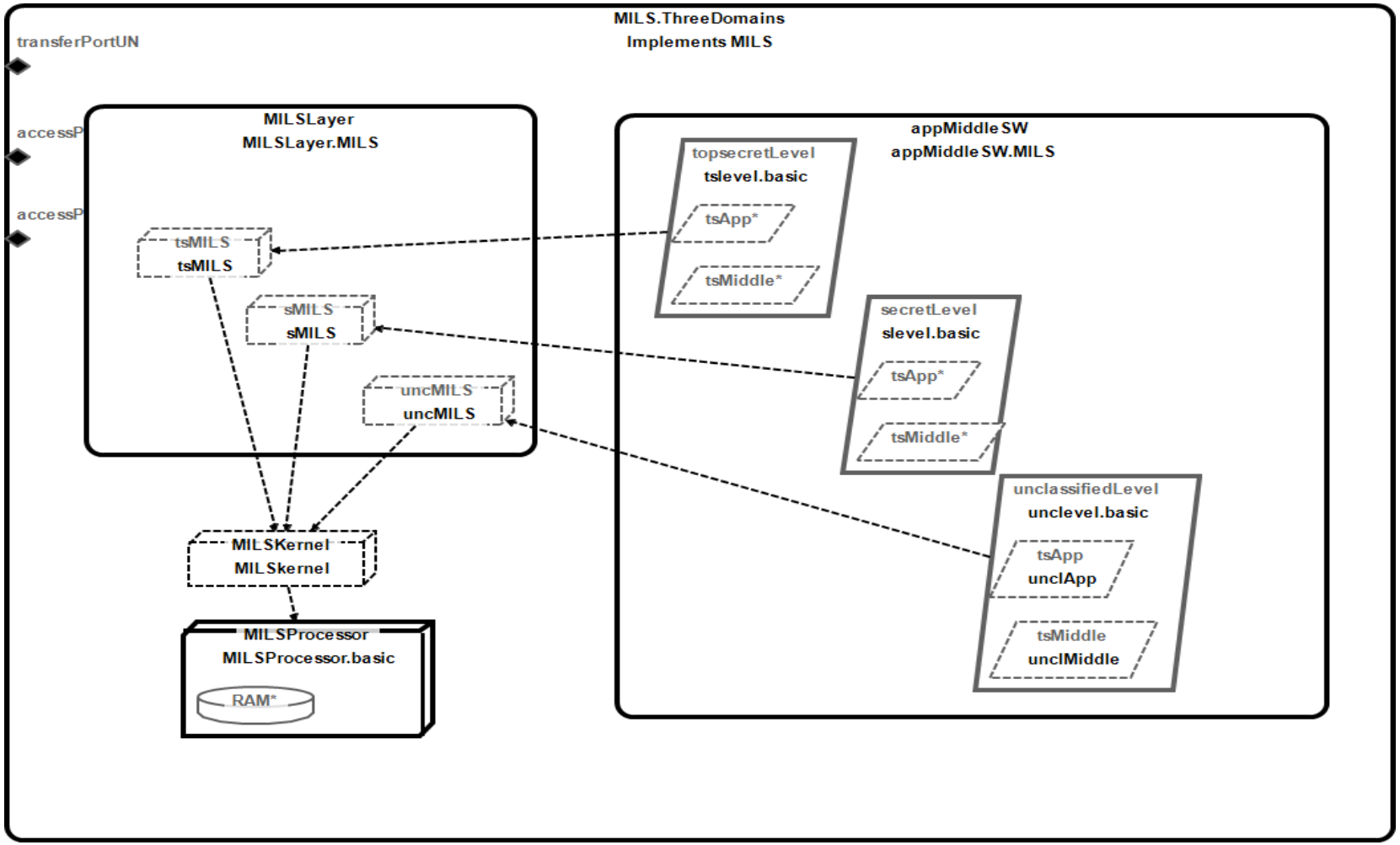
```
                    -- security level checks
prove all_subcomponents_have_security_level(this.TopSecretDataStore) -- should be true
prove all_subcomponents_have_security_level(this.SecretDataStore) -- should be true
prove all_subcomponents_have_security_level (this)  -- not true because some are 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
```



© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

MILS Architecture of the TSAccessUnit



© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.

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;
```


© 2020 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.