Secure DDS

A Security Model suitable for Net-Centric, Publish-Subscribe, and Data Distribution Systems

RTESS, Washington DC, July 2007

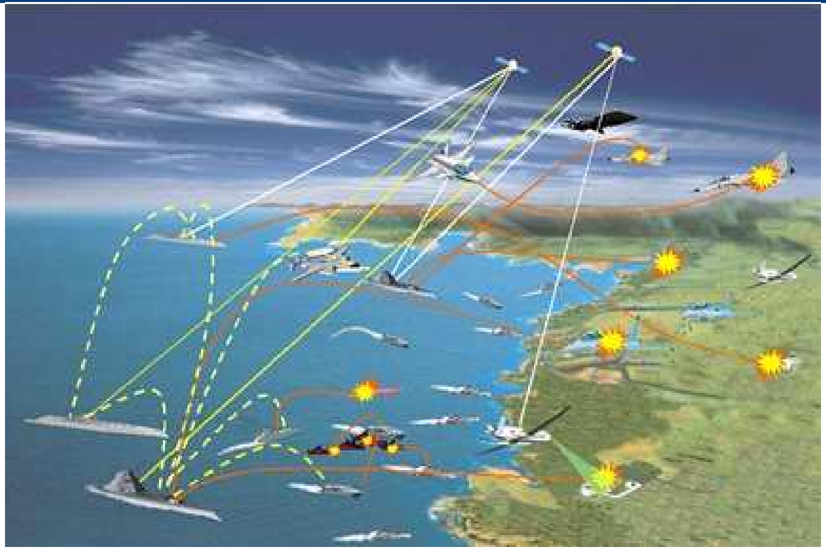
Gerardo Pardo-Castellote, Ph.D.
Chief Technology Officer
Real-Time Innovations, Inc.
gerardo@rti.com

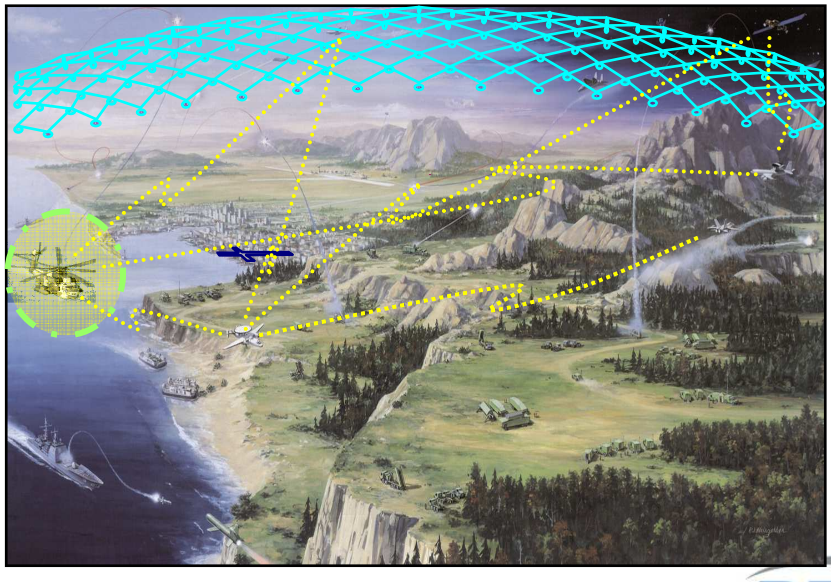
Agenda

- **Motivation & Requirements**
- DDS Concepts
- Security Concepts
- Is there a Conflict?
- Net-centric Security requirements
- Net-centric approach to Security
- Required DDS Extensions
- Conclusion

Motivation

- Net-centric systems rely on information sharing…
  - They are built on top a Shared Information Space or Common-Operational Picture
  - They assume information availability where and when it is needed
- As the system scope expends Security/IA becomes a concern
  - Information flow must be restricted to the intended/authorized recipients
  - Can no longer assume a single protection domain.





Application Requirements

- High-level goal:

  "*control and restrict access to information such that it is only accessible to the intended/authorized recipients.* "

- Secondary goals include:
  - **guaranteed access to the information** by the authorized users
    - **prevent denial of service attacks**
  - **audit trails**,
  - **ensure non-repudiation** (guarantee identity of the source of the information).
  - **deployable** within the exiting **Internet Environment**

- This breaks down into
  - Functional requirements:
    - Authentication, Confidentiality, Integrity
    - Availability
  - Non-functional requirements:
    - Manageability, Accountability, Assurance
  - Deployment requirements:
    - Work in WAN environment, operate across NATs

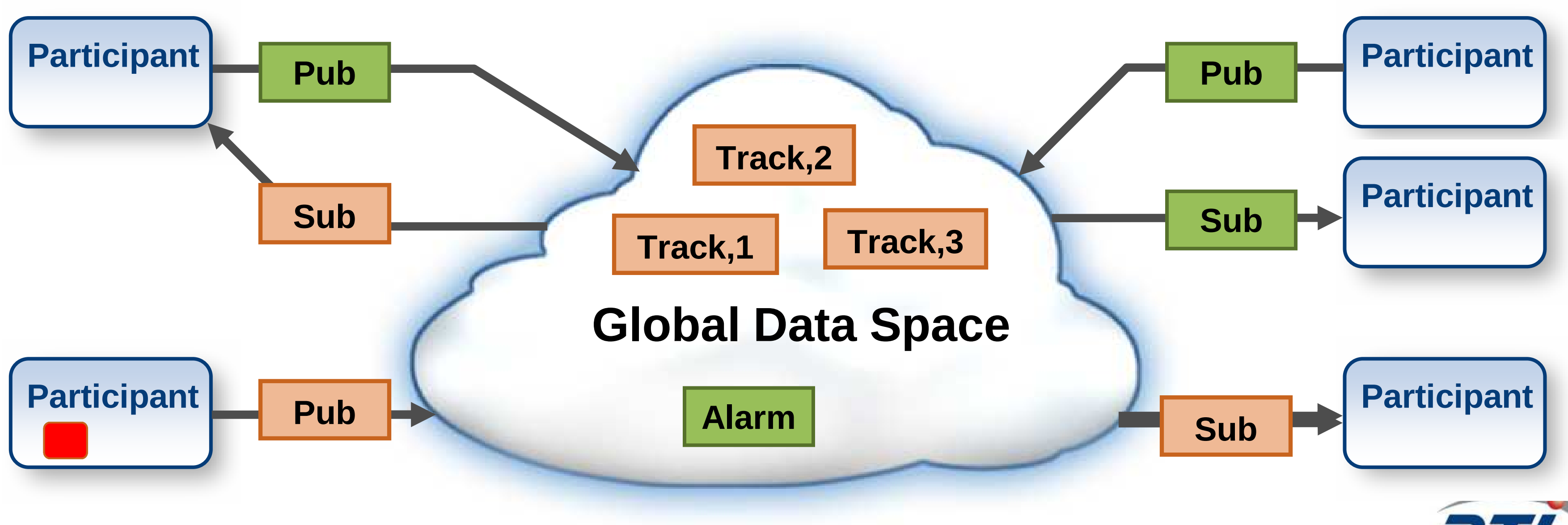
Agenda

- Motivation & Requirements
- **DDS Concepts**
- Security Concepts
- Is there a Conflict?
- Net-centric Security requirements
- Net-centric approach to Security
- Required DDS Extensions
- Conclusion

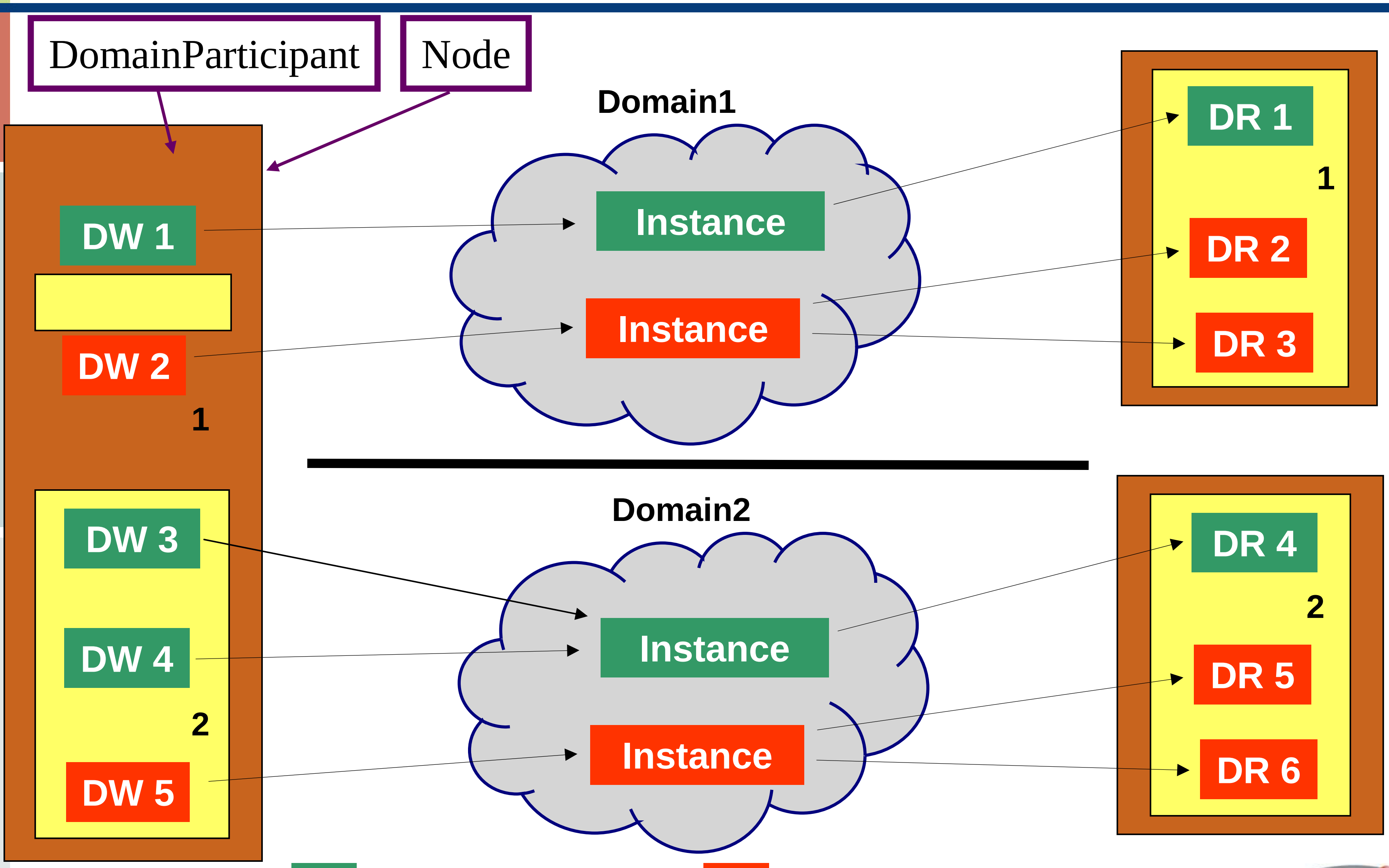
DDS Communications model

Provides a "**Global Data Space**" that is accessible to all interested applications.

- Data objects addressed by **Domain, Topic** and **Key**
- Subscriptions are **decoupled** from Publications
- Contracts established by means of **QoS**
- Automatic **discovery** and **configuration**

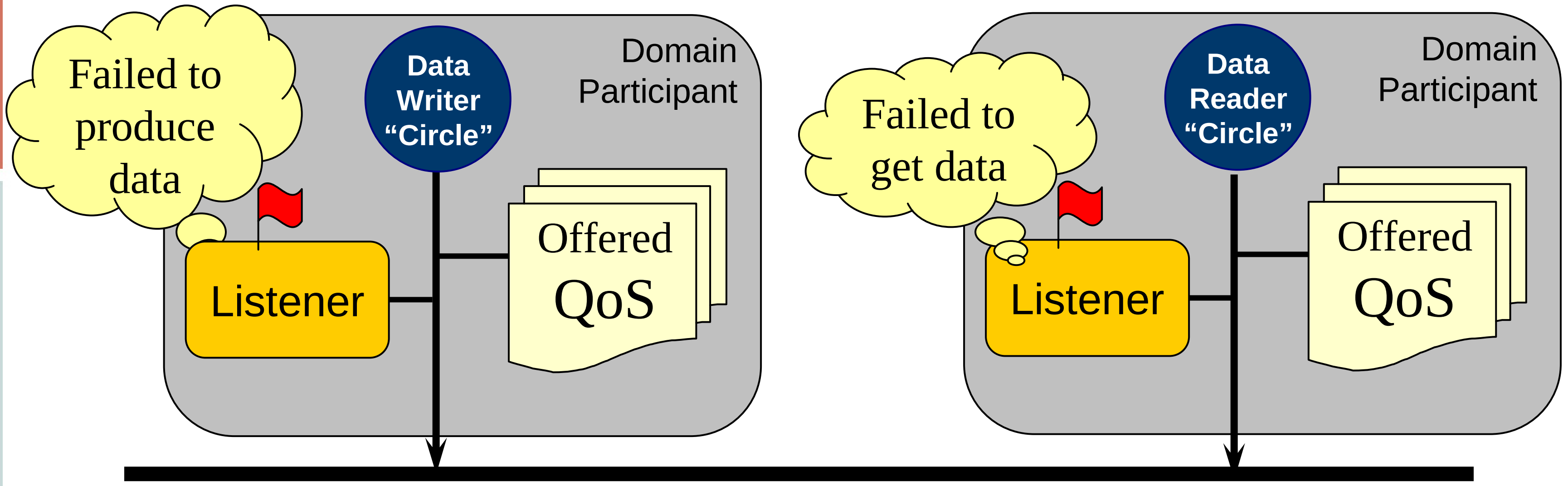


DDS Domain Separation



🟩 Topic “green” 🟥 Topic “orange”

DDS communications model



- Publisher declares information it has and specifies the Topic
  - and the offered QoS contract
  - and an associated listener to be alerted of any significant status changes
- Subscriber declares information it wants and specifies the Topic
  - and the requested QoS contract
  - and an associated listener to be alerted of any significant status changes
- DDS automatically discovers publishers and subscribers
  - DDS ensures QoS matching and alerts of inconsistencies

Agenda

- Motivation & Requirements
- DDS Concepts
- **Security Concepts**
- Is there a Conflict?
- Net-centric Security requirements
- Net-centric approach to Security
- Required DDS Extensions
- Conclusion

Security Terminology

- **Principals and Subjects**
  - The active entities
    - Principals are the entities that can be granted access control (e.g. the human users) .
    - Subjects are active computer-system entities that bound to principals and operate on their behalf. Subjects perform the actions on the objects.
- **Objects**
  - The passive entities
    - can be accessed or manipulated
      - e.g. memory, files, sockets, semaphores, etc.
- **Access operations**
  - the actions that can be performed on the objects
    - e.g. read memory, write a message on a socket.
- **Security model**
  - The architecture, concepts, and algorithms used to define the access control restrictions on the objects and the access rights given to principals.
  - Examples are Mandatory Access Control (MAC), Role-based Access Control (RBAC), Discretionary Access Control (DAC), etc.
- **Security policy**
  - a set of goals, rules, and regulations regarding the access rights to information and objects.
  - defines what is allowed and what is not allowed within a particular security model.
  - Examples are the Bell LaPadula model, the Biba model, Chinese Wall, etc.

Example: Unix File System

- "**principals**" → the registered users of the system (user accounts)
- "**subjects**" → the processes that user starts to perform actions
- "**objects**" → the files, devices, and other operating system resources whose access is restricted
- "**access operations**" → Read, Modify, Execute
- "**security model**" → Unix security model:
  - Each security target (file, device) has an owner, and a group, as well as a set of permissions (read, write, execute) for the owner, group and "other users".
  - Each user is assigned a set of groups to which they belong.
  - A subject (process) executing on behalf of a principal (user) is given permission to read, write, or modify a file based on the following rules:
    - (a) If he user is the one that owns the file, then access is granted according to the type of access (read/write/execute) and the corresponding permission for the owner of the file.
    - (b) Otherwise if the user belongs to the group to which the file belongs, then access is granted according to corresponding group permissions for the file.
    - (c) Otherwise access is granted according to corresponding "other users" permissions for the file.
- "**security policies**" → Defined by the organization by means of creating groups (e.g. 'admin', 'finance') and assigning users to those.

Security Concepts

- **Authentication**
  - Verify the identity of the principals accessing the information
  - Restrict information access to the authorized principals
  - Presumes a security model where the authorization rules are expressed
    - E.g. Mandatory Access Control or Role-Based Access Control
  - **State of the practice**: PKI, “Certificates” signed by a “recognized authority”
- **Confidentiality**
- **Integrity**
- **Non-repudiation**
- **Availability**

Security Concepts

- **Authentication**
- **Confidentiality**
  - Protect information so that only the intended recipients can see it, ensure the information cannot be snooped or intercepted.
  - Hide the contents of messages from third-party observers.
  - **State of the practice**: Encryption, Key-exchange algorithms
- **Integrity**
- **Non-repudiation**
- **Availability**

Security Concepts

- **Authentication**
- **Confidentiality**
- **Integrity**
  - Prevent data from being tampered/modified by a third party
  - Prevent spoofing/masquerading and the so called “man in the middle” attacks
  - **State of the practice**: Integrity check functions or cryptographic hash functions
- **Non-repudiation**
- **Availability**

Security Concepts

- **Authentication**
- **Confidentiality**
- **Integrity**
- **Non-repudiation**
  - Prove in an irrefutable way who the originator of the message is.
    - Depends on having Integrity. Knowing the source of a message that could have been modified in transit offers little value.
  - **State of the practice**: digital-signatures.
- **Availability**

Security Concepts

- **Authentication**
- **Confidentiality**
- **Integrity**
- **Non-repudiation**
- **Availability**
  - Protect the system/infrastructure so that authorized principals are not prevented from accessing the information
  - **State of the practice**: Boundary protection, use of challenges to Limit Denial of Service Attacks

Background: Public Key Infrastructure

- Two components:
  - Technology base: Asymmetric Key Encryption
  - Infrastructure/Policy: Set of trusted Certificate Authorities (CAs)

- Asymmetric Key Encryption
  - 2 different keys are used: PublicKey and PrivateKey
  - Messages encrypted with one key can only be decrypted with the other.
  - PublicKey is made available to everybody

- Certificate Authorities
  - Agreed set of “authorities” whose PublicKey is known
  - CAs are trusted to “sign” certificates asserting facts about the holder (e.g. their PublicKey, the company name, address, etc.)
    - These certificates are tamper-proof
    - Formats are standard e.g. X.509
  - Cryptographic techniques are used to verify the certificates

Example: web-based e-commerce

Assume you enter https://www.rti.com as the URL:

- **Authentication**
  - Browsers (e.g. Firefox, InternetExplorer) are pre-configured with the public keys of “recognized” CAs (e.g. RSA, Thawte, Verisign)
  - Each web site (e.g. RTI.com) gets a Certificate signed with Public Key of the CA.
  - The HTTPS protocol (layered on TLS/SSL) attempts to connect to www.rti.com as part of this rti.com supplies the signed certificate.
  - The certificate contains the name of the CA (e.g. RSA) and information such as “domain name: rti.com” validity period “jan 2007 to jan 2008”, public key of RTI, etc.
  - Certificate is encrypted with CA private key. Browser use the CA’s public key to verify the certificate
- **Confidentiality**
  - Once authenticated the browser and the RTI web-server use their respective public keys and to “negotiate” a fresh **private key**. This “negotiation” includes some random elements introduced by each side.
  - From there on all messages are encrypted using the newly-created **private key**.
- **Integrity and Non-repudiation**
  - The sender computes a hash of the message and encrypts the hash with its private key.
  - Using the public key anybody can verify that:
    - The message has not been tampered with
    - The sender (only one that knows public key) is the origin of the message.
- **Availability**
  - Before connecting and consuming state/recourses issue a challenge that forces client to consume significant CPU

Agenda

- Motivation & Requirements
- DDS Concepts
- Security Concepts
- **Is there a Conflict?**
- Net-centric Security requirements
- Net-centric approach to Security
- Required DDS Extensions
- Conclusion

Is there a Conflict?

- Security…
  - desires to restrict communication to only happen between authorized subjects
  - wants to confidentiality so that only communicating subjects see the information
- PubSub/DDS
  - attempts to create a ‘global information space’ where anybody can access the information it needs
  - de-couples communications so publishers are unaware of subscribers and vice-versa

Security in the Global Information Space

- The conflict is only apparent
- Publishers are decoupled from subscribers via the Global Information Space
  - This does not mean loss of access control to the information
  - It means that the Information Space must have an associated security model
- DDS can use standard PKI and cryptographic techniques to enforce the security policies
- The situation is analogous to access control policies in a file system
- **The key is to use a net-centric security model!**

Agenda

- Motivation & Requirements
- DDS Concepts
- Security Concepts
- Is there a Conflict?
- **Net-centric Security requirements**
- Net-centric approach to Security
- Required DDS Extensions
- Conclusion

General Secure DDS requirements

- DDS needs a net-centric (pub-sub friendly) security model:
  - Security model defined in terms of DDS concepts (Domains, Topics)
  - Authorization should be separate from apps
  - De-coupled via the GDS
  - Support one-to-many & multicast
  - Allow persistence and other DDS services

- We have investigated two possible models:
  - Domain-based MAC (Basic)
  - Topic-based RBAC   (Advanced)

Mandatory Access Control (MAC)

- Typical in Military systems
- Each “security object” is assigned a “security level” (Unclassified, Classified, Secret, Top-Secret).
  - Order: Unclassified < Classified < Secret < Top-Secret.
- Each ***principal*** is also assigned a “clearance level”
- MAC policy made of two rules:

(1) A ***principal*** is only allowed to read information where

security level <= clearance level

(2) A ***principal*** is only allowed to **write** information where

security level >= clearance level

Notes:

(1) prevents access to restricted information by users that have not been granted the right level of clearance.

(2) prevents users that have access to more secure information from leaking it to users that should not have access to it.

SELinux (Security-Enhanced Linux) project added a Mandatory Access Control architecture to the Linux kernel.

Role Based Access Control (RBAC)

- Information is only accessible to the owners as well as the ***principals*** that have the right "role".
  - E.g. Medical records may only be accessible to the patient itself as well as somebody assigned the role of "medical doctor".
- E.g. an organization could create roles representing the various job functions.
  - The rights to perform certain access operations could be assigned to specific roles.
  - Each member of the staff is assigned particular roles, and thus become authorized to perform certain access operations.

Notes:

Role-based access control provides the capability of what in A&D is generally called "**need to know**".

Systems including Microsoft Active Directory, SELinux, Solaris, Oracle DBMS, PostgreSQL 8.1, SAP R/3 and many others effectively implement some form of RBAC

Specific Secure DDS requirements

- Basic (must have)
  - Authenticated DomainParticipants
  - Confidential Domain communications
  - MAC for Domains
  - Integrity of all data
- Advanced (nice to have)
  - “Need to know”. RBAC per Topic
  - Separate read and write access
  - Centralized security configuration & management
  - Audit tools

Agenda

- Motivation & Requirements
- DDS Concepts
- Security Concepts
- Is there a Conflict?
- Net-centric Security requirements
- **Net-centric approach to Security**
- Required DDS Extensions
- Conclusion

RTI

Proposed DDS Security Models

- The approach based in two ideas:
  - Secure Domains
  - Confidential Topics

- Secure Domains
  - Limit each Global Data Space (DDS ***Domain***) to contain information at a single level of security
  - Only authorized participants are allowed to join a Domain
  - All domain communications are confidential
  - All information is accessible to all members of the domain

- Confidential need-to-know Topics
  - Within a Global Data Space allow participants to read and write ***Topics*** on a “need to know basis”
  - Separate “right to read” from “right to write”
  - Each Topic is authorized and protected separately

Secure DDS Domains

- Each DDS ***domain*** implements MAC at a single security level.
- Each DDS domain is configured with a CA. The PK of the CA is compiled into each application
- The CA gives certificates to each Participant that is allowed to join the domain
- When participants discover each other they use the pre-configured CA PK to verify their peer’s certificate to join the domain

- Result:
  - Limits access to the domain only to the principals (the DDS ***Participants***) that have the proper security clearance.
  - Mandatory Access Control (MAC) is therefore applied to each DDS ***Domain*** separately.

Implementing Secure Domains

- Use DTLS – datagram flavor of TLS/SSL
  - TLS/SSL provides functions to issue certificates and verify them
- Once verified DTLS establishes secure communications between each pair of participants
- Each message is also signed providing integrity and non-repudiation

- Disadvantages:
  - Multicast is not supported
  - All-or-nothing security

Using Secure Domains

- Security Admin uses TLS tools to create a CA. Obtains PK of CA. The CA-PK is ‘compiled’ into every application
- Security Admin issues certificates identifying the computers that can join the domain and distributes them securely
- App developers or executors install the Certificate in their computer (file, environment) or build it into their application
- At run-time the DTLS library takes care of authentication, encryption, etc.

Confidential DDS Topics

- The RBAC model applied to DDS Topics

- Each ***Topic*** is assigned a list of "reader roles" and a list of "writer roles".
  - 'reader roles' are the roles of the principals that can read the Topic
  - 'writer roles' are the roles of principals that can write the Topic

- Each ***Participant*** attached to the ***Domain*** is assigned a set of roles.
  - Write access to the ***Topic*** given only to ***Participants*** that have a role that appears in the list of "writer roles" for the ***Topic***
  - Read access to the ***Topic*** given only to ***Participants*** that have a role that appears in the list of "reader roles" for the ***Topic***

- Result:
  - Limits access to the Topic only to the principals (the DDS ***Participants***) that have the "need to know" for that Topic
  - A single domain can carry traffic of multiple security levels

Implementing Confidential Topics

- Use DTLS – datagram flavor of TLS/SSL
- Similar to Secure Domain DTLS is used for:
  - Authentication of participant
  - Checking of right to Join Domain
  - Negotiating session key -> Confidentiality
  - Signing messages -> Integrity, non-repudiation
- Details:
  - Participant Certificate is propagated via discovery
  - Each time a remote Entity (Topic, DataReader, DataWriter) is discovered the Participant certificate is checked to establish right to read/write Topic
- Disadvantages:
  - Multicast is not supported
  - Apps within domain can discover all entities

Using Secure Topics

- Security Admin (SA) uses TLS tools to create a CA. Obtains PK of CA. The CA-PK is ‘compiled’ into every application
- SA issues certificates identifying the Topics in the domain and the corresponding read-access roles and write-access roles
- SA issues certificates to individual participants identifying the domain, and their roles.
- App developers or executors install the Certificate in their computer (file, environment) or build it into their application
- DTLS library used internally to check that the Participant has the role it needs to create DataWriter and DataReaders by checking the roles in the participant against those in the Topic.

Supporting Multicast

- MC support cannot use standard SSL session-based encryption… possibilities are:
  - Topic-based encryption
  - DataWriter-based encryption
  - Role-based encryption
- **Topic-based**
  - Each Topic is encrypted with a separate crypto-key
  - Topic key distributed securely to authorized participants
- **DataWriter-based**
  - Each DataWriter uses its own crypto-key
  - DataWriter shares its crypto-key with authorized readers
- **Role-based**
  - Each role has a crypto-key
  - Each Writer or Message has its own crypto-key
  - Either
    - DataWriter key is encrypted with *all* the read-role keys and shared with readers
    - Or Message key is encrypted with *all* the read-role keys

    Either way any reader that has one of the roles can get the key.

Recommendation: DataWriter key

- Each DataWriter uses its own crypto-key to encrypt all messages
  - Multicast messages are allowed

- DW key must be propagated to all authorized readers
  - This can be done via the standard discovery mechanism by encrypting the DW key with the public keys of the authorized readers
  - This approach allows the DW to explicitly control who it communicates with and alings with the DDS::ignore_subscription() api

RTI

Agenda

- Motivation & Requirements
- DDS Concepts
- Security Concepts
- Is there a Conflict?
- Net-centric Security requirements
- Net-centric approach to Security
- **Required DDS Extensions**
- Conclusion

Required DDS Extensions

- Secure Domains can be implemented at the transport level. It does not require any extensions to DDS.

- Confidential Topics would require some extension to DDS.
  – Extensions would be limited to some new QoS policies

Security Attributes for DDS Entities

DDS Concept/Entity	*Security Attributes*	*Policy Notes*
DDS Domain/ DDS domainId	security_level security_access_roles[*]	A DDS ***Domain*** represents a separate global data-space. All he information on that Global Data Space is only accessible at the single level of security indicated by the *security_level* attribute The information in the ***Domain*** is only accessible to applications that have "need to know". The *security_access_roles* attribute lists a set of tags that represents the roles of applications/processes that have a 'need to know' for the information in the ***domain***. The DDS domainId is the integer used in the DDS API to represent a ***Domain***.
Application/ DDS DomainParticipantFactory	security_level security_roles[*]	Each Application is assigned a security level and a set of roles. Applications will only be allowed to "attach" to a domain if their *security_level* matches and they share at least one common role. The DDS DomainParticipantFactory is the singleton that represents the Application in the DDS API.
DDS Topic/ topicName	security_read_roles [*] security_write_roles [*]	A ***Topic*** is assigned two attributes: the *security_read_roles* and *security_write_roles*. The *security_read_roles* attribute lists the roles of the Processes that have a "need to know" for the information available under that topic and hence have "read permission" to the ***Topic***. The *security_write_roles* attribute lists the roles of Applications that have a "need to modify" the Topic and hence have "write permission" to the ***Topic***. The DDS topicName is the string used in the DDS API to represent a ***Topic***.
DDS DomainParticipant	security_level security_roles[*]	These attributes are set by the system when the Participant s created. See the create_participant() operation on the DDS ***DomainParticipantFactory***.

Security extensions to the operations on DDS Entities

DDS Entity	***operation***	***Changes to support security***
DDS DomainParticipantFactory (DPF)	create participant(domainId)	**Check that:** (DPF.security_level == domainId.security_level) AND (DPF.security_access_roles[] ∩ domainId. DPF.security_access_roles[] ≠ Ø) If either check fails, the ***participant*** creation fails If they both succeed the created DomainParticipant will have its security attributes set to: DP. security_level := DPF.security_level DP. security_roles[] := DPF.security_access_roles[] ∩ domainId. DPF.security_roles[]

DDS DomainParticipant (DP)	create_topic(topicName)
	
	lookup_topic(topicName)
		Check that:
		(DP. security_roles[]
		∩ (topicName.security_read_roles[]) ≠ Ø)
		OR
		(DP. security_roles[]
		∩ (topicName.security_write_roles[]) ≠ Ø)
		
		If the check fails, the operation fails, otherwise it proceeds as stated by the DDS specification.

DDS Entity	***operation***	***Changes to support security***
DDS Subscriber (S)	create_datareader(Topic)	Check that: (S.participant().security_roles[] $\cap$ Topic.security_read_roles[] $\neq \emptyset$) If the check fails, the operation fails, otherwise it proceeds as stated by the DDS specification.
DDS Publisher (P)	create_datawriter(Topic)	Check that: (S.participant().security_roles[] $\cap$ Topic.security_write_roles[] $\neq \emptyset$) If the check fails, the operation fails, otherwise it proceeds as stated by the DDS specification.

Agenda

- Motivation & Requirements
- DDS Concepts
- Security Concepts
- Is there a Conflict?
- Net-centric Security requirements
- Net-centric approach to Security
- Required DDS Extensions
- **Conclusion**

Conclusion

- Security requirements do not intrinsically conflict with the use of publish-subscribe or a net-centric global information space
- To be effective Security Models such as MAC and RBAC need to be mapped to global-information space concepts
- COTS technologies such as PKI and TLS can be used to provide Secure DDS domains and Confidential DDS Topics.
- Supporting fine-grained security may require some extensions to DDS