Component Protection Policy for the
GetAccess, ValidateAccess, CollectRecords
Configuration Example

Matt Young, Brian Nienhaus, Rose Gamble
Department of Computer Science
University of Tulsa
July 2010

© R.F. Gamble, University of Tulsa, 2010
Profile Highlights

- A component, depicted in the CPP UML profile, is represented by a unique id (program_id when mapped to UNITY), a location, and any ports the component has.
- A CPP Port is represented by
  - an id,
  - an (a list) associated policy(s),
  - a list of interaction partners
  - a communication style
- Component ports are restricted to communicating with designated interaction partners
- CommunicationType, stereotyped as <<Communication>> states the style of communication at the port
- Policies are associated with represented by the <<Policy>> UML stereotype
  - Each policy has a set of Enforcements and Assertions, represented by stereotypes.
  - Enforcements are non-functional Booleans stating what functional requirements must be met by the Policy / Port
  - Assertions stereotype contains three attributes
    - The stated attribute 'states' if the policy is utilized. False means the policy is not in effect at the port.
    - The dataType assertion represents what kind if data or information the port is sending or receiving
    - the dataStructure assertion states what data structure the data type is stored in.
- Enforcements are defined as followed
  - logEvents -> Create a log if events occurred (not necessarily creating an audit record)
  - auditAuthentication -> User data (from an authentication attempt) must be audited and stored in a record stating that the access attempt took place and when
  - encryptAuditRecords -> Audit records, for storage and transmission, must be encrypted
  - validateEncryption -> When receiving encrypted data (in our example, Audit records), the component (receiving port) must validate that the file was sent encrypted
  - auditorComm -> The component (port) can communicate with the Central Auditor (not sure if it is needed since it would be handled using the interaction partners)
  - auditTailFreq -> A time correlated audit trail is made with received audit records
CPP Enumerations for the Audit Example

© R.F. Gamble, University of Tulsa, 2010
Example Instantiation

• Three distinct components
  – **GetAccess**
    • Gathers user data (username and password).
    • Sends to ValidateAccess
  – **ValidateAccess**
    • Receives user information from GetAccess
    • Checks if the user is valid and not currently logged in (CheckUser)
    • If access allowed, returns a verdict of *true* (SendVerdict) to GetAccess, otherwise returns a verdict of *false*
    • Generates an audit record in accordance with its specified Audit policy
    • Sends audit records (SendAuditData) to CollectRecords
  – **CollectRecords**
    • Creates a time correlated audit trail from audit records sent by one or more ValidateAccess components
Component Configuration for Audit Example

Legend:
C1: GetAccess
C2: ValidateAccess
C3: CollectRecords
Capable of having multiple instances of the same component
CollectRecords Instantiation
Note: The Policy information is not shown. The ‘stated’ Boolean would be set to false meaning the policy would not be enforced or a policy does not govern the GetAccess component, thus this information is not shown.