CONTROL
A New CORBA Security Level 2 Access Control Model

Polar Humenn

2-212 CST
Syracuse, NY 13244-4100
polar@adiron.com
CONTROL

- What is Security Level 2?
- What is the Current CORBA Access Control Model?
- What is CONTROL?
- How is it better than the current CORBA Security Model?
CORBA Security
What is Security Level 2?

- Attributes and Credentials
- AccessDecision Object
- RequiredRights Object
- Domain Manager Objects
- AccessPolicy Objects
Security Level 2 Credentials API

- Credentials Interface
  - Holder of Security Attributes

```java
local Credentials {
    AttributeList get_attributes(
        in AttributeTypeList types
    );
};
```
Security Level 2
Security Attributes

- Type Value Pairs

```c
struct ExtensibleFamily {
    unsigned short family_definer;
    unsigned short family;
};

struct AttributeType {
    ExtensibleFamily attribute_family;
    SecurityAttributeType attribute_type;
};

struct SecAttribute {
    AttributeType attribute_type;
    OID defining_authority;
    Opaque value;
};
typedef sequence<SecAttribute> AttributeList;
```
Security Level 2
Access Decision Model

- Request on Target is Intercepted
  - Get Client's Credentials
  - Get Attributes from Credentials
  - Hand Attributes to AccessDecision function
Security Level 2
Access Policy Model

● Required Rights
  ● Rights required to gain access to an operation of a specific interface.

● Access Policy
  ● Mapping of Security Attributes to Effective Rights

● Access Decision Function
  ● Combination of Required Rights and Effective Rights yields the access decision
Security Level 2 Access Decision Model

- Access Decision is a Pull Model

Diagram:
- Access Decision
  - AttributeList
  - Domain
  - InterfaceName
  - Operation
  - AccessPolicy
    - effective rights
    - required rights
  - Required Rights
  - Domain Manager
  - Access Policy
Security Level 2
Inherent Problems

- Scalability
  - Required Rights Object
  - Domain Manager Objects
  - Access Policy Objects
- Efficiency
  - Too many calls to outside the server
- Recursion
  - Protecting access to the RequiredRights, DomainManager, and AccessPolicy objects using the same model requires exceptions and confusion.
Security Level 2 Problems
The Interfaces Are Modeled After People

- Required Rights Administrator
  - Required Rights Object
- Domain Administrator
  - Domain Manager Objects
  - Access Policy Objects
- Effective Rights Administrator
  - Access Policy Object,
    - Attribute to Effective Rights Mapping
What's wrong that?

People are notoriously inefficient!
The Solution
Change the Model!

● Make the Policy Model a Push Model
  ● Handles Scalability
● Make the Access Decision in the Server
  ● Handles Efficiency
● Use a Domain Specific Language (DSL) for Access Control Policy
  ● Atomic Operation for Policy Update
  ● Comprehensive Examination of Policy
  ● Human Readability of Policy
CONTROL from Adiron

- Server Computed Access Decision
  - Efficient
- Every Server has an Access Manager Agent
  - Scalable
  - Atomic, Comprehensive Updates and Initialization
- Access Policy Language, SAL
  - Human Readable
  - Convertible to XML
CONTROL Toolkit ORB

CONTROL
- ORBAsec SL2 2.1 from Adiron
  - Kerberos
  - SSL
- ORBacus 3.3 for Java from Object Oriented Concepts

Features
- Automatic Authentication is based on initialization information.
- Access control is performed by interceptors.
- Application code remains security unaware.
- Application tools update access control policy.
CONTROL Toolkit ORB

- **Features**
  - Application code remains security unaware.
    - Creation of credentials is performed during initialization.
  - Access decisions are performed by interceptors.
  - Introduces concept of Domain, Owner, and Owner of Domain.
  - Access Policy is written in a Domain Specific Language.
    - You can look at it.
CONTROL Initialization

ORB.init()
Calls on Security Level 2 Principal Authenticator
Sets up Initial Services

ORB.BOA_init()
Reads Initial Access Control Policy
Sets up Access Manager Agent
Sets up Initial Services
Binds Access Manager Reference to Naming Service
Sets up Access Control Interceptor
CONTROL
Server Request
interface AccessManager {

   TaggedDescription get_access_description(
         in DescriptionType type
    ) raises ( AccessManagerError );

   void update_access_description(
         in TaggedDescription description
    ) raises ( AccessManagerError );

};
Access Policy Administration Push Model

Admin Client

SAL

Access Manager

Control Server

Access Manager

Control Server
Server Access Language
SAL

- Domain Specific Language (DSL)
  - Security Level 2 Credentials Attributes
  - CORBA Rights
  - Defines Access Control Policy based on
    - Credentials
    - Operations
    - Interfaces
    - Domains
    - Required Rights and Effective Rights Mapping
Why is a Domain Specific Language Better?

- Language is tailored for a specific purpose.
- Descriptions written in the language can be formally analyzed.
- SAL is much more expressive than the Security Level 2 IDL interfaces that are currently present (i.e. IDL is not a good security policy language).
Conclusion
CONTROL is great because ...

- It uses CORBA Security Access Control using Standard Security Level 2 Credentials and Attributes.
- It is scalable. Policy updating is a push model. Access Managers can be federated.
- It is efficient. The server calculates access decision without remote calls to policy, required rights, and domain manager objects.
- Access Policy Language, SAL, is Human Readable and Writable.
Server Access Language
SAL

- Basic Constructs
  - (tag name definition)

- Examples
  - (AttributeFamily Corba1 (0 1))
  - (AttributeFamily Corba2 (0 2))
  - (AttributeType AccessId (Corba2 2))
  - (RightFamily Corba0 (0 0))
  - (Right Get (Corba0 "get"))
  - (Right Set (Corba0 "set"))
  - (Right Manage (Corba0 "manage"))
Credentials Predicate

- A Credentials Predicate is a construct that compares attributes of the Credentials to explicit values
  - (CredentialsPred isBart
      (AccessId "bart@MYREALM.COM"))

- Can Use Regular Expressions
  - (CredentialsPred isAdmin
      (AccessId `.*/admin@MYREALM\..COM`))
Credentials Predicates Comparing to Other Attributes

- Credentials Predicates can compare values to attributes of another source.
  - (CredentialsPred isSameAsMe (AccessId (ValueOf Server AccessId)))
    - This construct compares the client's AccessId attribute with that of the server's credentials.
  - (CredentialsPred isOwner (AccessId (ValueOf Owner AccessId)))
    - This construct compares the client's AccessId attribute with the AccessId attribute of the Owner of the object if set, otherwise, the owner of the domain if set, otherwise the server.
Composite Predicates

- Credentials Predicates are expressive that they can be composed
- (CredentialsPred isBartOrAdmin (or isBart isAdmin))
- (CredentialsPred isBartOrLisa (or isBart (AccessId "lisa@MYREALM.COM")))))
Controls

- Controls Provide Access Decisions to Predicates
  - (CredentialsControl \textit{name}
    ((credentials-pred Allow|Disallow ) .... )
  
- (CredentialsControl AdminOnly
  ((isAdmin \textit{Allow})
   (true Disallow))


Controls for Operations, Interfaces, and Domains

- **OperationControl**
  - `(OperationControl name interface_name ((op-name credentials-control) ....) )`

- `(OperationControl AdminUpdate
  "IDL:adiron.com/AccessManager:1.0"
  ("update_description" AdminOnly)
  ("_is_a" Allow)
  ("*_" Disallow))`
Controls for Operations, Interfaces, and Domains

- InterfaceControl places OperationControls on Interfaces
  - (InterfaceControl name
    (interface-name op-control ... ))
- (InterfaceControl ServerPolicy
  ("IDL:adiron.com/AccessManager:1.0"
   AdminUpdate)
  ("MyObject" Allow)
  ("*" Disallow))
Controls for Operations, Interfaces, and Domains

- Domain Interface Control
  - (DomainInterfaceControl name
    ((domain-name interface-control) ....) )
  - (DomainInterfaceControl MyDomainPolicy
    ("A/B/C" ServerPolicy)
    ("A/B/D" ServerPolicy)
    ("A" AnotherPolicy))
Application of Policy

- (Apply
  - (CredentialsControl cred-control)
  - (OperationControl op-control)
  - (InterfaceControl interface-control)
  - (DomainControl domain-control)
  - (DomainOperationControl domain-op-control)
  - (DomainInterfaceControl domain-intf-control)
- default-Allow-Disallow)
CORBA Security Authorization Model

- Actually More Expressive than CORBA Security Authorization Model using Rights
  - Have "and", "or", "any" and "all".
- (RightsPred isManagable (any (Manage Set)))
- (RightsPred TotalControl (and (all Use Get) isManagable))
Operation and Interface Rights

- The Operation Rights constructs assigns Rights Predicates to operations of an interface.
  - (OperationRights AMRights
    "IDL:adiron.com/AccessManager:1.0"
    ("update_access_description" isManagable)
    ("*" (all Get)))

- The Interface Rights constructs assigns Operation Rights to Interfaces
  - (InterfaceRights ServerRights
    ("IDL:adiron.com/AccessManager:1.0" AMRights))
Filling out the Model
Assigning Effective Rights

- A Credentials Rights construct associates Credentials Predicates with effective rights.
- \( (\text{CredentialsRights MyAccessPolicy}) \)
  
  \[
  (\text{isAdmin} \quad (\text{Manage Set Get})) \\
  (\text{isBart} \quad (\text{Get Set})) \\
  (\text{true} \quad \text{none})
  \]
Marrying the Required Rights and Effective Rights

- A Credentials Rights Control construct marries a RightsPredicate and a CredentialsRights Construct to define the control
  - (CredentialsRightsControl MyPolicy
    (ServerRights MyAccessPolicy)
- Apply the Credentials Rights Control
  - (Apply (CredentialsRightsControl MyPolicy) Allow)
Conclusion
SAL is More Expressive than Security Level 2 IDL

- SAL can express
  - Predicates on Credentials
  - Controls based on Credentials Predicates
  - Mapping from Credentials to Rights
  - Predicates on Rights
  - Controls based on Rights Predicates and Mapping of Credentials to Rights
  - Rights Controls are reduced to Credentials based Controls
  - All Controls are can be based on any granularity of Credentials, Operation, Interface, and Domain