Security Policy Manager Solutions Guide

The Trusted RUBIX Security Policy Manager (SPM) is a mechanism to enforce flexible and dynamic Attribute Based Access Control (ABAC) security policies during the operation of the Trusted RUBIX Relation Database Management System (RDBMS). Security policies are created using the XML Trusted RUBIX Security Markup Language (RXSML) which is based upon the policy language of the OASIS XACML 2.0 standard. The RXSML language allows policy creation and execution using a host of context attributes and functions to manipulate them. The RXSML language also allows actions to be executed based upon the outcome (e.g., Permit, Deny) of the security policy execution. Policies may be configured to release information across any domain defined by the operating system's Mandatory Access Control policies (OS-MAC), including the Multilevel Security (OS-MAC MLS) policy and the Type Enforcement (OS-MAC TE, SELinux platforms only) policy.

For details of the RXSML policy language used for the following solutions please see the SPM Tutorial document.

Solution 1: Cross Domain Releasability Policy

Requirements:

- Two domains: US and France
- Single table in US domain containing US government employees with three columns: *name*, *pay_grade*, *organization*
- Read-write access by US domain to all rows of table
- Read-only access by France to a subset of rows in the table where *pay_grade* is less than ten
- France must not see actual value of *organization* (e.g., NSA, CIA, etc) but shall see a "cover" of *US Government Employee*
- The rows France is allowed to see must be audited
- No access by France to remainder of rows in the table where pay_grade is greater than or equal to 10

Environment of Solution:

- US domain connects to Trusted RUBIX server platform on network interface fixed with session label *Confidential:US*
- France domain connects to Trusted RUBIX server platform on network interface fixed with session label *Confidential:FR*
- Table named *xtab* created with columns *name* (string), *pay_grade* (integer), and *organization* (string)
- Table *xtab* populated by US domain

Security Policy of Solution:

- DAC permission given to France on table *xtab* to allow *select*

Copyright © 2014
Infosystems Technology, Inc.

- DAC permission given to US on table *xtab* to allow all operations
- SPM security policy called *US-Access* created with target of *Confidential:US* session label which always evaluates to *Permit*
- SPM security policy called *FR-Access* created with target of *Confidential:FR* session label. Rules are evaluated in order and the policy evaluates to *Permit* immediately if a rule evaluates to *Permit*; otherwise, it evaluates to *Deny*
- The *FR-Access* policy contains the following rules:
    - Rule1: If the current operation is *select* **and** column *pay_grade* < 10 then *Permit*
    - Rule2: If the current operation is of type *open* then *Permit*
    - Rule3: *Deny*
- The *FR-Access* policy contains the following obligations:
    - Obligation 1: On *Permit* audit with optional audit data set to the value of the *name* column
    - Obligation 2: On *Permit* set the field *organization* to *US Government Employee*
- SPM security policy set is configured to override the OS-MAC MLS policy and assigned to table *xtab* containing policies *US-Access* and *FR-Access*. Policies are evaluated in order and the policy set evaluates to *Permit* immediately if a policy evaluates to *Permit*; otherwise, it evaluates to *Deny*

Behavior of Solution:

Any operation on table *xtab* submitted by US will succeed. Any operation submitted on table *xtab* by France other than *open* and *select* will fail. A *select* operation on table *xtab* by France will have all rows where *pay_grade* >= 10 filter out of the result set. If the column *organization* is selected by France its value will always be *US Government Employee*. All rows selected by France will be audited and the value of the *name* column will be included in the audit record.

An equivalent solution could be constructed using Type Enforcement (OS-MAC TE policy) of an SELinux enabled OS instead of the OS-MAC MLS policy.

Solution 2: Individual Tables with IP Address White Lists

Requirements:

- Multiple user tables in the database
- Users are able to access a specific user table only if the user is connecting from an IP address approved for that table
- The set of approved IP addresses for a specific table must be administrated through the SQL client interface by a specific administrative user connecting from a specific IP address (127.0.0.1)

Environment of Solution:

- Administrative table *iplists* created with columns *user-table-name* (string) and *ip-address* (string)

Copyright © 2014
Infosystems Technology, Inc.

- Each row in *iplists* specifies the name of a user table (*user-table-name* column) in the database and an IP address (*ip-address* column) that may be used to connect to that table
- A row in *iplists* indicates users may access the table specified by name in *user-table-name* from the IP address specified by *ip-address*
- The administrative user has a user name of *ipadmin*
- User tables created and populated as desired

Security Policy of Solution:

- DAC permissions given to user *ipadmin* on table *iplists* for all operations
- DAC permissions given to basic users on user tables as desired
- All objects and sessions are at label *Unclassified*
- SPM policy called *IPlists* targeting all operations and assigned to table *iplists*. Rules are evaluated in order and the policy evaluates to *Permit* immediately if a rule evaluates to *Permit*; otherwise, it evaluates to *Deny*
- The *IPlists* policy contains the following rules:
    - Rule1: If user name is *ipadmin* **and** subject's IP address context attribute is 127.0.0.1 then *Permit*
    - Rule2: *Deny*
- SPM policy called *UserTab* targeting all operations. Rules are evaluated in order and the policy evaluates to *Permit* immediately if a rule evaluates to *Permit*; otherwise, it evaluates to *Deny*
- The *UserTab* policy contains the following variable definition:
    - Extract all values of the *ip-address* column of table *iplists* where the *user-table-name* column value is equal to the resource attribute *table-name* (this will create a multi-valued container with all values of *ip-address* whose corresponding *user-table* column value is equal to the name of the table currently being operated upon)
- The *UserTab* policy contains the following rules:
    - Rule1: If the current operation is of type *open* then *Permit*
    - Rule2: If subject's IP address context attribute is within the multi-valued container specified by the variable definition then *Permit*
    - Rule3: *Deny*
- The *UserTab* policy is configured with a scope covering the subtree and assigned to the database (this will cover all existing objects in the database and new objects as they are created)

Behavior of Solution:

Any user who accesses a table from a client host with a corresponding *user-table-name* / *ip-address* entry in the *iplists* table will succeed. All users who accesses a table from a client host without a corresponding *user-table-name* / *ip-address* entry in the *iplists* table will fail. Access to table *iplists* from client IP addresses other than 127.0.0.1 or from users other than *ipadmin* will fail. User *ipadmin* will be able to perform any operation on table *iplists* from client host 127.0.0.1. As updates to the *iplists* table occur the effects upon the security policy behavior will be immediate and transactionally controlled.

Copyright © 2014
Infosystems Technology, Inc.

Solution 3: Row Access Restricted to Row Creator

Requirements:

- Single user table
- Table rows may be accessed only by the user that created the row

Environment of Solution:

- A table named *tab* created with desired columns and a column named *creator* (string)

Security Policy of Solution:

- DAC permissions given to users on table *tab* as desired
- All objects and sessions are at label *Unclassified*
- SPM policy called *UserTabInsert* targeting an operation of *insert* and always evaluates to *Permit*
- The *UserTabInsert* policy contains the following obligation:
    - Obligation 1: On *Permit* set the value of the *creator* column to the value of the *subject-name* context attribute
- SPM policy called *UserTabAllOthers* targeting all operations except *insert*. Rules are evaluated in order and the policy evaluates to *Permit* immediately if a rule evaluates to *Permit*; otherwise, it evaluates to *Deny*
- The *UserTabAllOthers* policy contains the following rules:
    - Rule1: If the current operation is of type *open* then *Permit*
    - Rule2: If operation is not *update* **and** the *subject-name* context attribute is equal to the *owner* column value then *Permit*
    - Rule3: If operation is *update* **and** the column *owner* is not being updated **and** the *subject-name* context attribute is equal to the *owner* column value then *Permit*
    - Rule4: *Deny*
- SPM security policy set is assigned to table *tab* containing policies *UserTabInsert* and *UserTabAllOthers*. Policies are evaluated in order and the policy set evaluates to *Permit* immediately if a policy evaluates to *Permit*; otherwise, it evaluates to *Deny*

Behavior of Solution:

All rows in the *tab* table will have the user name of the inserter as the value of the *owner* column. All attempts to update the *owner* column will fail. All row based operations, other than update of the *owner* column, will succeed only if the current user name is equal to the value of the *owner* column of the row being operated upon. All select operations on the *tab* table will have all rows with *owner* column values that are not equal to the current user name filtered out of the result set.

Copyright © 2014
Infosystems Technology, Inc.