

## The Trusted RUBIX™ ABAC Security Policy Manager

- |   |  |
|---|--|
| <ul style="list-style-type: none"><li>• XML based Attribute Based Access Control policies</li><li>• 29 context attributes available for policy decision</li><li>• 44 functions available to manipulate attributes</li><li>• Modular policies using rules, policies, and policy sets</li><li>• Full integration with the underlying OS-MAC policy</li><li>• Construct releasability policies across any MLS domain</li><li>• Construct refining policies within any MLS domain</li></ul> | <ul style="list-style-type: none"><li>• Update policies while the database is on-line</li><li>• Customizable auditing based upon policy outcome</li><li>• Dynamically set row-fields based upon policy outcome</li><li>• Policy decision based upon any row-field value</li><li>• Hide database objects based upon policy outcome</li><li>• Perform set operations on groups of attributes</li><li>• Policy engine integrated with server for fast execution</li></ul> |
|---|--|

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 based Security Policy Markup Language (SPML). The SPML language allows policy creation and execution using a host of context attributes and functions to manipulate them. The SPML language also allows actions to be executed based upon the outcome of the security policy execution. Policies may be configured to release information across any domain defined by the underlying Multilevel Secure Mandatory Access Control policy (OS-MAC).

The SPML language is based upon the policy language of the [OASIS XACML 2.0](#) standard. The attributes used to write policy logic are typed (e.g., string, integer) and are categorized as subject attributes (e.g., subject name, subject IP address), resource attributes (e.g., object name, object label, row values), action attributes (e.g., operation, operation category), and environment attributes (e.g., system date and time). The functions used to manipulate attribute values are categorized as logic functions (e.g., and, or), comparative functions (e.g., equal, greater than), conversion functions (e.g., cast, convert to lower case), group-of-values functions (e.g., testing if a value is in a group of values), and set functions (e.g., intersection, union).

Access control logic code is organized into rules, policies, and sets of policies and algorithms may be specified to define how they interact with each other. Policies and policy sets may be referenced by name allowing for the elegant, modular design of complex policy logic and the reuse of policy logic without code duplication. Policies are assigned to RDBMS objects and may be specified to protect a single object or an entire subtree of objects. Policies may also be configured to automatically protect newly created objects.

Policies may be configured to override the underlying OS-MAC policy (i.e., a releasability policy) or to further restrict operations beyond the OS-MAC policy (i.e., a refining policy). Objects that have no ABAC policy associated with them are by default protected by the underlying OS-MAC security policy.

In addition to permitting or denying a database operation on an object, SPML may be used to define actions that are conditionally taken based upon policy decisions. Possible actions are to set the value of a specific row field during a row-based operation, to change the default behavior when an operation is denied, and to write a customizable audit record.

## Language Features

### Major Language Constructs:

- **attribute:** Value reflecting the context of the database system used to make policy decisions.
- **target:** Set of context attributes for which a rule, policy, or policy set applies.
- **rule:** Predicate over attributes used to reach an outcome of permit or deny.
- **policy:** Set of rules that combine to form a single policy outcome.
- **policy set:** Set of policies or other policy sets that combine to form a single policy outcome.
- **obligation:** Action performed conditionally based upon policy or policy set outcome.

### Subject Attributes:

- subject-id
- group-id
- session-start-time
- session-start-dateTime
- dns-name
- subject-name
- group-name
- session-start-date
- ip-address
- session-label

### Resource Attributes:

- resource-label
- row-label
- view-label
- catalog-label
- column-name
- view-name
- catalog-name
- any row-field value
- resource-name
- table-label
- schema-label
- database-label
- table-name
- schema-name
- database-name

### Action and Environment Attributes:

- action-id
- current-time
- current-dateTime
- action-type
- current-date

### Comparison Functions:

- equal
- greater-than
- less-than
- time-in-range
- ipAddress-match
- MAC-check
- not-equal
- greater-than-or-equal
- less-than-or-equal
- dnsName-match
- regexp-match

### Mathematical Functions:

- add
- multiply
- mod
- round
- floor
- sum
- subtract
- divide
- abs

### Conversion Functions:

- cast
- concatenate
- map
- string-normalize-to-lower-case
- string-normalize-space

### Boolean Functions:

- or
- n-of
- and
- not

### Set and Bag Functions:

- one-and-only
- is-in
- any-of
- any-of-any
- any-of-all
- map
- union
- subset
- bag-size
- bag
- all-of
- all-of-any
- all-of-all
- intersection
- at-least-one-member-of
- set-equals

### Obligations:

- **set-field:** Set a row-field with a value constructed from attributes (row select, insert, or update).
- **set-error-code:** Set the error code used after a policy denial thus changing the default database behavior.
- **audit:** Write an audit record containing values constructed from attributes.