

where `project1` and `mrec1` are objects. The set of objects may pertain to files, ports, clipboards, email messages, records and fields. The selection of entities included in this set is a matter of choice determined by the protection requirements of the system. The system **20** maintains the association between the logical object names and the corresponding resources (e.g., files or the system clipboard).

[0031] Operations are actions that can be performed on the protected resources. Subsets of these operations are environment specific. For example, in an operating system environment, operations may include read (r) and write (w) operations. The operations may also include a fixed set of administrative operations that create, modify and delete data and relations of the access control database.

[0032] The system **20** may also define user attributes (UA) and object attributes (OA). User and object attributes characterize users/objects and serve as user/object containers. Each object is also considered to be an attribute of itself, i.e., $O \subseteq OA$. User attributes, respectively object attributes included in FIG. 3 are Doctor and Secret, respectively `mrec1`, Med Records, and TS (for top secret). However, containers can satisfy a variety of uses. Containers may represent folders (e.g., “DHS Proposals”), application-specific work areas (e.g., “Smith Inbox”), or a column in a database table (e.g., “Diagnosis”).

[0033] The system **20** has the ability to configure instances of access control policies, as well as combinations of different access control policies, named policy classes as denoted by PC. FIG. 3 includes two policy classes, RBAC and MLS, which refer to the policy objectives rather than the way the objectives are enforced.

[0034] Also included in FIG. 2 is a set of processes (P). Each process is associated with a single user. A user may have multiple processes running at any one time. The function `process_user: P → U` associates a process with its user.

[0035] The following summarizes the definitions:

[0036] U denotes the set of users.

[0037] UA denotes the set of user attributes.

[0038] O denotes the set of PM objects.

[0039] OA denotes the set of object attributes, with $O \subseteq OA$ (each object is an object attribute).

[0040] OP is the set of system operations.

[0041] PC denotes the set of policy classes.

[0042] P denotes the set of processes.

[0043] The sets of event pattern/response relations (EP-R), user denies (UDENY) and process denies (PDENY) are described below with respect to the relevant relations.

Assignment Relations

[0044] Assignments are binary relations denoted by “→.” Users are assigned to one or more user attributes and objects are assigned to zero or more object attributes. For example, the user with the identifier “alice” of FIG. 3 is assigned to “Doctor” and “Top Secret” and the object “mrec1” is assigned to “Med Records” and “TS.”

[0045] A user attribute may be assigned to another user attribute and an object attribute may be assigned to another object attribute. The second object attribute cannot be an object. The only other restriction on assignments is that any chain of attribute assignments cannot be a cycle.

[0046] A user attribute may be assigned to one or more operation sets `ops`, with $ops \subseteq OP$. In turn, those operation sets may be assigned to object attributes. Abbreviated notations may be used to illustrate these assignments. For example, Doctor

Med Records may be used instead of `Doctor → {w} → Med Records`. As discussed below, these assignment relations indirectly derive the capabilities and permissions, which are fundamental in performing and managing access control.

[0047] Not all users (and their processes) are controlled under all policies, nor are all user attributes and object attributes relevant to all policies. To afford appropriate policy class mappings, users, user attributes, objects and object attributes are associated with relevant policy classes through assignment relations (a user can be assigned to a policy class only through one or more user attributes). A user or user attribute “belongs to” or “is contained in” a policy class if there exists a chain of one or more assignments that starts with that user or user attribute and ends with the policy class. Similarly, an object or object attribute “belongs to” or “is contained in” a policy class if there exists a chain of one or more assignments that starts with that object or object attribute and ends with the policy class. In FIG. 3, the object `project1` is protected (contained in) only under the RBAC policy class, while `mrec2` is protected under (contained in) both RBAC and MLS.

[0048] The following summarizes the definitions:

[0049] $UUA \subseteq U \times UA$ denotes the user-to-user-attribute assignment relation.

[0050] $UAUA \subseteq UA \times UA$ denotes the user-attribute-to-user-attribute assignment relation, which has no cycles.

[0051] $OAOA \subseteq OA \times (OA - O)$ denotes the object-attribute-to-object-attribute assignment relation, which has no cycles.

[0052] $UAOPS \subseteq U \times 2^{OP}$ denotes the user-attribute-to-operation-set assignment relation.

[0053] $OPSOA \subseteq 2^{OP} \times OA$ denotes the operation-set-to-object-attribute assignment relation.

[0054] $UAPC \subseteq U \times PC$ denotes the user-attribute-to-policy-class assignment relation.

[0055] $OAPC \subseteq OA \times PC$ denotes the object-attribute-to-policy-class assignment relation.

Conventions

[0056] The following conventions are used throughout the remainder of this description. The notations \rightarrow^* respectively \rightarrow^+ denote a chain of zero or more assignments, respectively a chain of one or more assignments. For example, consider the assignments illustrated in FIG. 3 where `bob → Doctor → Intern → RBAC`. In this case, one could write `bob →+ RBAC` (or `bob →* RBAC`). Also, one could write `bob →* bob` but not `bob →+ bob`. Also, user `u` is said to “have a user attribute” `ua` if `u →+ ua`, and object `o` is said to “have an object attribute” `oa` if `o →* oa`.

Permission Relations

[0057] Permissions are assertions regarding the capabilities of users and their processes in performing operations on objects, irrespective of any known exceptions. Permissions are triples of the form `(u, op, o)` indicating that a user `u ∈ U` is able to perform operation `op ∈ OP` on the contents of object `o ∈ O`.

[0058] Referring to FIG. 4, the set of capabilities of a user `u ∈ U` in a policy class `pc ∈ PC` is $caps_{pc}(u) = \{(op, o) \mid \exists ops \in 2^{OP}, \exists oa \in OA, \exists ua \in UA: u \rightarrow^+ ua \rightarrow^+ pc, o \rightarrow^* oa \rightarrow^+ pc, ua \rightarrow ops \rightarrow oa, op \in ops\}$.