

d=maximum depth of child nodes from the root, where the root is level 0, i.e., d is equivalent to the maximum level of the leaf nodes; and

if the tree is relatively balanced, then $\lfloor C/d \rfloor$ compartments (that is, the integral value of C/d) are available at each level for representing privilege sets.

To distinguish between privilege sets, combinations of compartments are used. At each level in the tree, where n is the number of compartments available for representing roles at that level, the number of privilege sets that can be distinguished is

$$\binom{n}{\lfloor n/2 \rfloor}$$

Privilege sets may be associated with compartments as follows:

1. A role at the root of the tree, with privileges available to all users, is associated with a randomly selected compartment. This compartment is removed from the set of compartments available to designate roles.

2. Roles at level 1 of the tree, where n_i indicates the number of nodes at level 1, are associated with unique sets of compartments drawn from the set of remaining compartments. The number of compartments needed for level 1 is the smallest number c such that

$$n_i \leq \binom{c}{\lfloor c/2 \rfloor}$$

Choose c compartments from the remaining set of compartments. Remove these c compartments from the set of compartments available to designate roles.

3. From the set of compartments chosen in step 2, assign a unique set of compartments to each privilege set at level 1. Step 2 ensures that there are enough compartments to make all the sets different.

One way of implementing this step is to generate a list L_1 of numbers from 1 to 2^c-1 , then extract from this list a second list L_2 containing all numbers whose binary representation contains $\lfloor C/2 \rfloor$ bits. Each bit is associated with a compartment. Assign to each privilege set at level 1 a different number from L_2 . Then label each privilege in a privilege set with compartment i if and only if bit i in the binary representation is a 1. For example, the mapping from bits to compartments in Table 1 shows how the procedure works for $c=3$ compartments. Extracting all sets of two compartments from the list (for example) gives $\{c_2, c_1\}$, $\{c_3, c_1\}$, $\{c_3, c_2\}$. Because all of the numbers associated with privilege sets have $\lfloor C/2 \rfloor$ bits, each privilege set will be labeled with a different set of compartments.

4. Label each privilege in each privilege set with the compartments assigned to the privilege sets above it in the tree, i.e., the inherited privileges.

5. Repeat steps 2 through 4 until all privilege sets have been assigned a set of compartments.

Each role must be able to access all privileges associated with its privilege set and all privilege sets associated with roles that it inherits, i.e., roles that are represented by ancestor nodes in the role hierarchy. Compartments may be assigned to roles as follows:

1. Assign to role R_i the set of compartments assigned to its privilege set.

2. For each ancestor role R_j from which role R_i inherits privileges, assign the compartments associated with the privilege set for R_j .

MLS systems typically provide 64 to 128 compartments for labeling privileges. The implementation of RBAC on an MLS system described by the non-prior art Meyers paper referred to above assigns one bit to each role, providing a capability for controlling access for 128 roles. This is inadequate for any real application. By comparison, the construction described in the previous section will provide the capability for

$$\binom{\lfloor C/d \rfloor}{\lfloor C/2d \rfloor}$$

roles. Tables 2 and 3 show the number of roles that can be controlled for various combinations of depth and breadth (branching factor) of role hierarchies.

TABLE 2

Number of Roles Supported with 64 Compartments		
Depth	Max. Branching Factor	Max. Roles
5	924	6×10^{14}
10	20	1×10^{13}
15	6	4.7×10^{10}
20	3	3.4×10^9

TABLE 3

Number of Roles Supported with 128 Compartments		
Depth	Max. Branching Factor	Max. Roles
5	5,200,300	3.8×10^{33}
10	924	4.5×10^{29}
15	70	4.7×10^{27}
20	20	1.0×10^{26}
25	10	1.0×10^{25}
30	6	2.2×10^{23}
40	3	1.2×10^{19}

FIGS. 5 and 6 provide an example of compartment labeling for a hierarchical privilege set defining 36 roles. The tree shown in FIG. 5 has a depth of 2 and a maximum branching factor of 6. A total of 9 compartments a,b, . . . i are needed. The privilege sets assigned to a role are those labeling the role's node in the tree, plus the labels of any ancestor nodes. For example, role R_{33} has compartments a, b, d, g, and i.

Consider roles R_0 , R_1 , and R_{20} . Privileges authorized for role R_0 are assigned compartment a. Privileges authorized for role R_1 are assigned compartments a, b, and c (a from role R_0 and b and c from role R_1). Privileges authorized for role R_{20} are assigned compartments a, b, c, g, and h (a from role R_0 ; b and c from role R_1 ; and g and h from role R_{20}). A user who establishes a session at role R_1 will be assigned compartments a, b, and c. Note that this user can access the privileges assigned to role R_0 because the user has compartment a. A user who establishes a session at role R_{20} will be assigned compartments a, b, c, g, and h. This user can access all inherited privileges, but not any other privilege sets, because all others have at least one compartment not assigned to role R_{20} .

FIG. 6 shows a portion of FIG. 5, with exemplary privilege sets associated with various roles. Each of the privileges P_1 and P_2 associated with role R_0 is labeled with compartment a. Therefore any user authorized for role R_0 , or any role that inherits privileges from R_0 (e.g. R_1 , R_7 , etc.), can access privileges P_1 and P_2 . Note that a user authorized