The Journal of Systems and Software 83 (2010) 2607–2617

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Fault coverage of Constrained Random Test Selection for access control: A formal analysis

Ammar Masood [a,*], Arif Ghafoor [b], Aditya P. Mathur [c]

[a] Avionics Engg Department, Institute of Avionics and Aeronautics, Air University, Islamabad, Pakistan
[b] School of ECE, Purdue University, West Lafayette, IN 47906, United States
[c] Department of Computer Science, Purdue University, West Lafayette, IN 47906, United States

ARTICLE INFO

Article history:
Received 19 September 2008
Received in revised form 25 July 2010
Accepted 16 August 2010
Available online 19 September 2010

Keywords:
Role Based Access Control (RBAC)
Finite state models
W-method
Fault coverage

ABSTRACT

A probabilistic model of fault coverage is presented. This model is used to analyze the variation in the fault detection effectiveness associated with the use of two test selection strategies: heuristics-based and Constrained Random Test Selection (CRTS). These strategies arise in the context of conformance test suite generation for Role Based Access Control (RBAC) systems. The proposed model utilizes coverage matrix based approach for fault coverage analysis. First, two boundary instances of fault distribution are considered and then generalized. The fault coverages of the test suites generated using the heuristics-based and the CRTS strategies, applied to a sample RBAC policy, are then compared through simulation. Finally the simulation results are correlated with a case study.

1. Introduction

Access control is the key security service used for information and system security. Access control mechanisms can be used to enforce various security policies, but the desired access control objectives can only be achieved if the underlying software implementation is correct. It therefore becomes essential to verify that the implementation conforms to the given policy. Earlier a model-based approach for conformance testing of a Role Based Access Control (RBAC) (Ferraiolo et al., 2001; Sandhu, 1998; Sandhu and Samarati, 1994) system has been proposed (Masood et al., 2009). We refer to such a system as an Access Control system Under Test, or simply as ACUT. The proposed test generation approach is based on representation of RBAC policies as finite state models and generating tests from these models using three conformance testing procedures.

The first procedure is based on a "complete Finite State Machine (FSM)" based conformance testing strategy. The strategy was to construct an FSM model of the RBAC policy and then generate tests from the model using the testing tree portion of the W-method (Chow, 1978). We examined the fault coverage of the complete FSM based conformance testing technique with respect to an RBAC fault model that consists of faults derived from first order mutations (DeMillo et al., 1978). The proposed technique provides complete fault detection with respect to the RBAC fault model that can be mapped to Chow's fault model (Chow, 1978; Fujiwara et al., 1991; Petrenko et al., 1996).

While complete FSM based conformance testing turns out to be highly effective in detecting RBAC faults, the size of the finite state model and that of the generated test suite is astronomical thereby rendering it unsuitable for practical use. Thus two strategies, namely, heuristics-based and Constrained Random Test Selection (CRTS) strategies have been further investigated in Masood et al. (2009). Heuristics-based strategy is centered on six heuristics that reduce the size of the original model and hence that of the generated test suite. CRTS strategy directly generates a significantly smaller test suite from the finite state model using random selection of paths and alleviates the need for an in-memory representation of the model.

In this paper, by contributing a probabilistic model of fault coverage, we formally analyze the variation in fault detection effectiveness associated with the use of heuristics-based and CRTS strategies in conformance test suite generation for RBAC systems. The proposed probabilistic model utilizes coverage matrix based approach for fault coverage analysis of the heuristics-based and the CRTS strategies. First two boundary instances of fault distribution are considered which are then generalized. The fault coverage of the test suites generated corresponding to the application of both heuristics-based and CRTS strategies on an example RBAC policy are then compared through a simulation for various cases of fault distribution. Finally the simulation results are compared with a case

* Corresponding author.
E-mail addresses: ammar.masood@mail.au.edu.pk (A. Masood), ghafoor@ecn.purdue.edu (A. Ghafoor), apm@cs.purdue.edu (A.P. Mathur).

0164-1212/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2010.08.022

study to establish correspondence with the fault distribution for real faults.

Organization: RBAC and Chow's FSM based test generation method are briefly reviewed in Section 2. Section 3 reviews the complete FSM-based, heuristics-based and CRTS strategies. In Section 4 we propose a framework for formally analyzing the fault coverage of the test generation techniques proposed in Masood et al. (2009). The two boundary instances of fault distribution considered in Section 4 are generalized in Section 5. In Section 6, through simulation, we compare the fault coverage of the test suites generated corresponding to the application of heuristics and CRTS strategy on an example RBAC policy. In Section 7 the simulation results are further compared with a case study. Section 8 reviews the related work in fault coverage analysis and Section 9 concludes the paper.

2. Background

2.1. Role Based Access Control

RBAC is often used to protect resources from unauthorized access. For example, a bank has a set of roles, e.g. Teller and Customer, users, e.g. John and Mary, and resources, e.g. accounts. Authorized personnel in this bank assign users to roles. In turn each role has associated permissions that allow a user to perform tasks. For example, a Teller is allowed to deposit a check into a customer's account and a customer may not have the permission to transfer money from her account to another customer's account.

An RBAC policy P is a 11-tuple (U, R, Pr, UR, $\leq_A$, $\leq_I$, I, S_u, D_u, S_r, D_r), where

- U and R are, respectively, finite sets of users and roles,
- Pr is a set of permissions,
- $UR \subseteq U \times R$ is a set of allowable user-role assignments,
- $PR \subseteq 2^{Pr} \times R$ is a set of allowable permission-role assignment,
- $\leq_A \subseteq R \times R$ and $\leq_I \subseteq R \times R$ are, respectively, activation and inheritance hierarchy relations on roles (Sandhu, 1998),
- $I = \{AS, DS, AC, DC, AP, DP\}$ is a finite set of allowable input requests for the ACUT, where *AS*, *DS*, *AC*, *DC*, *AP*, *DP* are, respectively *Assign*, *Deassign*, *Activate*, and *Deactivate* requests for user-role assignment and activation and *Assign* and *Deassign* for permissions-role assignments.
- S_u, $D_u : U \rightarrow Z^+$ are, respectively, static and dynamic cardinality constraints on U, where Z^+ denotes the set of non-negative integers.
- S_r, $D_r : R \rightarrow Z^+$ are, respectively, static and dynamic cardinality constraints on R.

The static (dynamic) cardinality of a user S_u (D_u) specifies the maximum number of roles it can be assigned to (can activate). Similarly, the static (dynamic) cardinality of each role S_r (D_r) specifies the maximum number of users who can be assigned to (can activate) this role. Note that while $(u, r) \in UR$ implies that assignment of u to r is allowable, u is authorized for assignment only when (i) an input request $AS(u, r)$ is received and (ii) the static user and role cardinality constraints are satisfied at the time the assignment request is received. For user u to be authorized to activate role r, (i) input request $AC(u, r)$ must be received, (ii) u must be assigned to r or permitted via $\leq_A$, and (iii) dynamic user and role cardinality constraints must be satisfied. We illustrate a sample policy using a simple example.

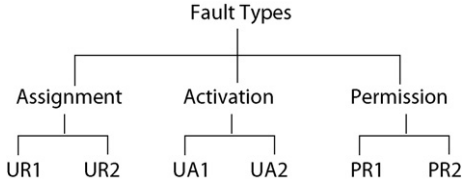

Fig. 1. A fault model for evaluating the effectiveness of tests of RBAC implementations.

Example 1. Consider the following policy P with two users, one role, and two permissions.

$$U = \{u_1, u_2\}, R = \{r_1\}, Pr = \{p_1, p_2\},$$
$$UR = \{(u_1, r_1), (u_2, r_1)\}, PR = \{(p_1, r_1), (p_2, r_1)\},$$
$$S_u(u_1) = S_u(u_2) = D_u(u_1) = D_u(u_2) = 1,$$
$$S_r(r_1) = 2, D_r(r_1) = 1, \leq_A = \leq_I = \{\}$$

Each permission p_1 and p_2 in Pr is associated with functions and resources related to the application under consideration.

2.2. RBAC fault model

The RBAC fault model (Masood et al., 2009) consists of three types of faults: user-role assignment, user-role activation, and permission-role assignment. As shown in Fig. 1, each fault is further categorized into two subcategories. Fault type UR1 restricts an authorized user from being assigned to a role or leads to an unauthorized deassignment. Fault type UR2 may lead to unauthorized role assignments. PR1 fault restricts a permission being assigned to an authorized role or cause an unauthorized deassignment. PR2 fault assigns a permission to an unauthorized role. UA1 and UA2 faults are similar to UR1 and UR2 and impact role activation.

2.3. Test generation from FSM

A Mealy Finite State Machine (FSM) is defined as $M = (Q, q_0, X, Y, \delta, O)$, where Q is a finite set of states, $q_0 \in Q$ a unique initial state, X and Y, respectively, the input and output alphabets, $\delta : Q \times X \rightarrow Q$ the state transition function, and $O : Q \times X \rightarrow Y$ the output function. We assume M to be complete, minimal, and connected (Chow, 1978).

Let $s = x_1 x_2 \ldots x_{k-1} x_k$, $x_i \in X$, $1 \leq i \leq k$, be a string of length $k > 0$ over the input alphabet X, also written as $s \in X^+$. We write $O^+(q_m, s) = r$ for string r of length k over the output alphabet Y, when $O(q_{l_j}, s_j) = r_j$, $1 \leq j \leq k$, $\delta(q_{l_j}, s_j) = q_{l_{j+1}}$, $q_m = q_{l_1}$, where all q's are states in Q. Similarly, we write $\delta^+(q_m, s) = q_k$, when $\delta(q_{l_j}, s_j) = q_{l_{j+1}}$, $1 \leq j \leq k$, $q_m = q_{l_1}$, $q_k = q_{l_k}$. States $q_i, q_j \in Q$, $i \neq j$, are considered distinguishable by string $s \in X^+$ if $O^+(q_i, s) \neq O^+(q_j, s)$.

Let $W = \{w_1, w_2, \ldots, w_m\}$ be a finite set of non-empty strings. W is a characterization set for M if for any two states $q_i, q_j \in Q$, $i \neq j$, there exists a $w \in W$ such that $O^+(q_i, w) \neq O^+(q_j, w)$. W is considered minimal if for all sets W', $|W'| < |W|$, W' is not a characterization set for M.

Let M model the expected behavior of the ACUT. Let $n = |Q|$. Let m be an estimate of the number of states in FSM M' that models the actual behavior of the ACUT. Note that the testing methodology proposed here does not require an explicit formulation of M'. Concatenation of sets R and S is written as $R.S = \{uv | u \in R, v \in S\}$. The W-method for generating test suite T from M proceeds in the following steps (Chow, 1978; Gill, 1968).

(1) Construct the testing tree Tr.
(2) Construct the transition cover set P_t from Tr. P_t contains all paths in Tr from its root to all internal nodes and leaves.
(3) Find the state characterization set W.
(4) Test set T is constructed as follows.

$$T = \begin{cases} \cup_{i=0}^{i=m-n} P_t X^i W & \text{if } m - n > 0 \\ P_t W & \text{otherwise} \end{cases}$$

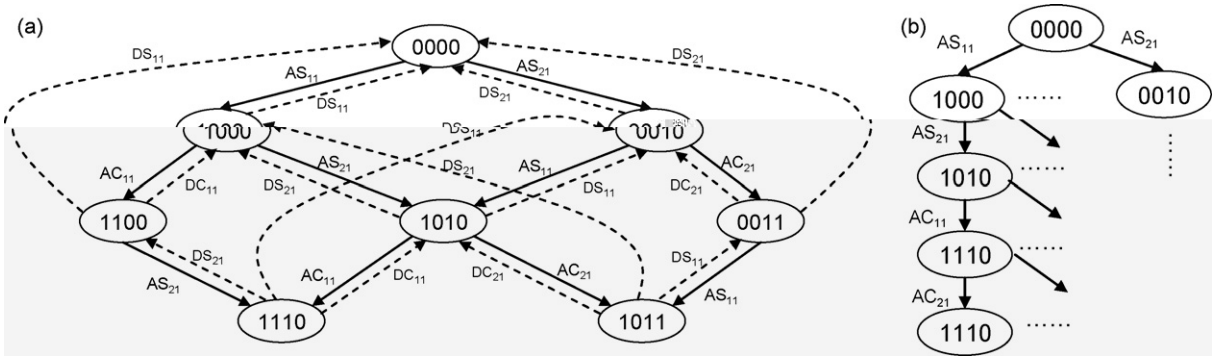

Fig. 2. (a) A complete finite state behavioral model derived from the RBAC policy in Example 1 and (b) its partial testing tree. Expected response is not shown. For each transition between two states, the response is *granted*. Self-loops corresponding to *denied* response are not shown to keep the figure uncluttered.

Chow's fault model for finite state machines, consists of four faults: operation, transfer, extra state, and missing state. An operation fault occurs when M' has a transition with an incorrect input or output label. A transfer fault occurs when M' has a transition that terminates at an incorrect state. Chow has shown that given m and a correct estimate of the number of states in M', T is guaranteed to detect all faults in the fault model.

State observability: Step 3 in the test generation procedure above is not needed when the states in the ACUT are observable. In this case there is a significant reduction in the size of the test suite as the equation in Step 4 reduces to $T = P_t$. It is to be noted that state observability assume availability of reliable state reporting methods in the ACUT, thus in cases where such assumption cannot be justified the tests have to be generated as per formula in Step 4.

A further reduction in the size of T becomes possible when the ACUT states are observable. In such cases there is no need to record paths from the root of Tr to each internal node as state identification is not required. Thus P_t now consists only of paths from the root of Tr to each leaf.

2.4. Modeling the expected behavior of the ACUT

The first step in the test generation strategy is to model the expected behavior of the ACUT as an FSM M. A state in M is a sequence of pairs of bits, one pair for each user-role combination as in the table below. For example, given two users u_1 and u_2, and one role r_1, a state is represented as a pattern of two consecutive pairs of bits. In this case, 1011 indicates that u_1 is assigned to role r_1 but has not activated r_1 and u_2 is assigned to r_1 and has activated it. The permission-role assignments are ignored to simplify the presentation.

Pattern	Role	
	Assigned	Activated
00	No	No
10	Yes	No
11	Yes	Yes
01	Not used	Not used

3. Generation of conformance test suite

In this section, we briefly review the three procedures proposed in Masood et al. (2009) for generating the test suite for conformance testing of the ACUT with respect to a specific RBAC policy. The three procedures provide varying fault detection effectiveness with respect to the RBAC fault model discussed in Section 2.2.

3.1. Procedure A: complete FSM-based

In this procedure tests are generated from the complete FSM (M = FSM(P)), derived from the policy P, as per the steps outlined in Section 2.3. The complexity of this procedure not only depends on the size of M but also on the observability of states in the ACUT. For the FSM(P) of Example 1 given in Fig. 2(a), the test set is partially shown in Fig. 2(b). For each input request subscripts i and j are used to denote, respectively, user u_i and role r_j. For example AS_{ij} is an abbreviation for "Assign u_i to role r_j. The testing tree is derived using the algorithm given in Chow (1978). It includes at least one path from the initial state to all other states in the FSM(P).

3.2. Procedure B: heuristics-based

In this procedure six heuristics, labeled H1–H6, are used to reduce the size of the model and of the test set. These heuristics are similar to the concept of state abstractions as used in various verification techniques. Each heuristic is explained with respect to Example 1 in Table 1 to illustrate the reduction in number of tests achieved through the application of a heuristic. Test generation is considered for three cases: All the inputs are considered in the complete FSM, *AS* and *AC* are ignored in assigned and active states, and additionally *DS* and *DC* are also ignored in unassigned and inactive states.

H1: Separating assignment and activation: Construct M_{AS} and $M_{AC_1}, M_{AC_2}, \ldots, M_{AC_k}$, $k > 0$. Here M_{AS} is an FSM that models all assignment requests. For each state $q_i \in M_{AS}$, there is an activation FSM M_{AC_i} that models the expected activation behavior under the assumption that the assignment state remains q_i. Fig. 3(a) shows M_{AS} and $M_{AC_{11}}$ for the policy in Example 1.

Note that a state in M_{AS} corresponds to assignments whereas a state in M_{AC} corresponds to activations. For the model in Fig. 2, application of this heuristic leads to an increase in the total number

Table 1
Size of test set for Example 1 obtained by applying various test generation techniques.

Procedure	Complete FSM	Ignore AS, AC in assigned and active states	Ignore DS, DC in unassigned and inactive states
A – H0	92	64	40
B – H1	44	32	16
B – H2	11	9	5
B – H3	1	1	1
B – H4	20	14	8
B – H5	92	64	40
B – H6	10	7	4
C – RT*k*	*i*	*i*	*i*

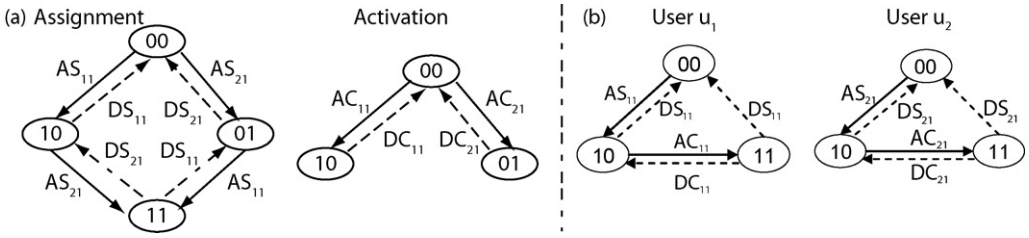

Fig. 3. Models constructed by applying (a) H1 and (b) H4. As in Fig. 2, self loops and outputs are not shown. Note that for H1 there are four activation FSMs though only $M_{AC_{11}}$ is shown here.

of states from 8 to 12. However, as in Table 1, the reduction in the number of tests is more than double. The reduction is due to the separation of the FSMs.

H2: FSM for activation and single test sequence for assignment: Construct model M_{AC} for activation requests with respect to a single state q_{max} that has the maximum number of assignments in M_{AS}. The single test sequence is the concatenation of requests along a path from the initial state of M_{AS} to q_{max}. For $q_{max} = 11$, $M_{AC} = M_{AC_{11}}$. Assignments and deassignments are covered using a sequence of AS and DS requests. The number of test cases now reduces from 92 to 11 in the best case.

H3: Single test sequence for assignment and activation: Use one test sequence that includes assignment, activation, de-activation, and deassignment requests for all the user-role pairs in any order. In this case the behavior of the ACUT is tested using a mix of all four types of requests. Doing so reduces the number of tests from 92 to 1 in the best case.

H4: FSM for each user: Construct M_u for each $u \in U$. Apply the test generation procedure separately to each model. Fig. 3(b) shows M_{u_1} for user u_1 and M_{u_2} for user u_2 for the policy in Example 1. The reduction in the number of states is from 8 to 6 and that in the size of the test set from 92 to 20.

H5: FSM for each role: Construct M_r for each $r \in R$. Apply the test generation procedure separately to each model. As the policy in Example 1 contains only one role r_1, M_{r_1} is the same as in Fig. 2. Hence there is no reduction in the size of the model or the test set. However, a reduction is expected when $|R| > 1$.

H6: Grouping users: Create user groups $UG = \{UG_1, UG_2, \ldots, UG_k\}$, $k > 0$, such that $\cup_{i=1}^{k} UG_i = U$ and $UG_i \cap UG_j = \varnothing$, $1 \le i, j \le k$, $i \ne j$. The groups can be created using one or more common attributes, e.g. all users that can be assigned to the same set of roles. The FSM is now constructed assuming that the user field in each input request corresponds to a user group and not to a user. For example, $AS(u_2, r_3)$ is a request to assign a user from UG_2 to role r_3.

3.3. Procedure C: CRTS

CRTS is aimed at reducing the number of test sequences without requiring reduction in the model size. We use the term "constrained" to describe CRTS to stress that though tests are generated randomly they are constrained by the model. The CRTS strategy is to select tests of length $k > 0$ randomly. A test case of length k is constructed by applying randomly generated sequence of requests $r = r_1, r_2, \ldots, r_{k-1}, r_k$ to FSM(P) and determining the corresponding state sequence $q = q_1 q_2, \ldots, q_{k-1}, q_k$. Request r_j, $1 \le j \le k$, is generated by selecting randomly user $u \in U$, role $r \in R$, and an input $I \in \{AS, DS, AC, DC\}$ from P. RTk represents test suite containing random tests of length k. The size of CRTS suite is selected to be comparable with the size of largest test suite obtained by the application of heuristics and it would be a fixed number, as an example some number i in Table 1. It is important to note that the CRTS is applied to the non-reduced FSM model as in Procedure A. While the original FSM is astronomically large, CRTS can be easily coded without the need to actually represent the model in internal memory.

4. Formal analysis of fault coverage

Fault coverage of the various test generation techniques proposed in Section 3 is analytically analyzed here with respect to the RBAC fault model given in Section 2.2. The fault coverage of a test suite used for conformance testing of systems modeled as FSM is evaluated as the ratio of faulty machines detected by the test suite and the total number of faulty machines (Petrenko et al., 1996). As discussed further in Section 8, the FSM based fault coverage is not suitable for analyzing the fault coverage with respect to RBAC faults, therefore we use a coverage matrix based approach for such analysis.

4.1. Fault coverage matrix

State observability is assumed in the FSM used to model the expected behavior of the ACUT. Under this assumption for all extra states in the M' the condition $\exists x \in X$, such that $_1(q, x) = q' \notin Q$ is satisfied. Thus, as already mentioned in Section 2.3, no concatenation of X^i is required with the transition cover set P_t for test generation because of state observability.

Based on the above assumption it can be observed that all faults in an M' could be associated with at least one transition in the FSM(P) where P is the policy implemented by the correct ACUT. The output, transfer and missing state faults are always associated with some transition and by virtue of assumption on the connectivity of extra states – a consequence of state observability – extra state faults would also be exhibited along a single transition.

Consider $\mathcal{T}_{H0} = \{t_1, t_2, \ldots, t_j\}$ as the set of transitions corresponding to FSM(P), where H0 represents the complete FSM based test generation technique (Procedure A in Section 3.1), and $F = \{f_1, f_2, \ldots, f_k\}$ is the set of RBAC faults related to faults in M'. Each transition $t \in \mathcal{T}_{H0}$ is of the form $(q_s, x/y, q_t)$, specifying that on application of input x, FSM(P) moves from state q_s to q_t while generating output y. Consider that $C(t, f) : \mathcal{T}_{H0} \times F \to \{0, 1\}$ specifies the coverage relation between the transitions and the faults such that $C(t_1, f_1) = 1$ implies that fault f_1 in M' is exhibited across transition t_1. Thus a test case executing t_1 is able to detect f_1. Note that a many-to-many relation can exist between the transitions and the faults. We define the coverage matrix $\mathbf{C}_{H0}$ as a matrix of j rows, one row corresponding to each transition in FSM(P), and of k columns where there is one column for each

$$\mathbf{C}_{H0} = \overset{\text{Faults}}{\overrightarrow{\begin{bmatrix} C(t_1, f_1) & C(t_1, f_2) & \ldots & C(t_1, f_k) \\ C(t_2, f_1) & C(t_2, f_2) & \ldots & C(t_2, f_k) \\ \vdots & \vdots & \vdots & \vdots \\ C(t_j, f_1) & C(t_j, f_2) & \ldots & C(t_j, f_k) \end{bmatrix}}}$$

$\mathbf{C}_{H0}(r, s) = 1$ thus signifies that t_r covers f_s. Note that $\mathbf{C}_{H0}$ considers the coverage of transitions with respect to FSM(P), whereas heuristics use smaller machines for test generation. By design the transitions in the test suites corresponding to all the smaller machines considered in the heuristics would be disjoint, i.e. con-

sidering $t = (q_s, x/y, q_t) \in \mathcal{T}_{M1}$ and $t' = (q'_s, x'/y', q'_t) \in \mathcal{T}_{M_2}$, such that $q_s = q'_s$ and $q_t = q'_t$ then either $x \neq x'$ or $y \neq y'$. Here $\mathcal{T}_M$ represents the set of transitions in the test suite generated from machine M. As an example considering the two machines M_{AC} and M_{AS} in H2, $\mathcal{T}_{M_{AS}} \cap \mathcal{T}_{M_{AC}} = \varnothing$.

The coverage matrix for heuristics, labeled as $\mathbf{C}_H$ where $H \in \{H1, H2, \ldots, H6\}$, includes the transitions of the test suites generated from the machines considered by the heuristic H. Similarly the coverage matrix corresponding to a test suite RTi of CRTS strategy, labeled as $\mathbf{C}_{RTi}$, would only include the distinct transitions in RTi. It is to be noted that $\mathcal{T}_x \subseteq \mathcal{T}_{H0}, x \in \{H1,H2, \ldots ,H6,RTi\}$.

Given a coverage matrix $\mathbf{C}_x$, $x \in \{H0, H1, H2, \ldots, H6, RTi\}$, consider two identity vectors $\mathbf{a}$ and $\mathbf{b}$ where $\mathbf{a}$ is a column vector of length k and $\mathbf{b}$ is a row vector of length equal to the number of rows in $\mathbf{C}_x$. We define $\mathbf{x} = \mathbf{C}_x \cdot \mathbf{a}$ and $\mathbf{x} = b \cdot \mathbf{C}_x$. Note that $\mathbf{x}$ signifies the power of transitions in exhibiting faults and $\mathbf{x}$ represents the visibility of a fault among all the transitions. In the remainder of this section the domain of x is always considered to be $\{H0, H1, H2, \ldots, H6, RTi\}$, and hence it is not explicitly mentioned. Corresponding to the complete FSM based test generation technique $\mathbf{H0} = \mathbf{C}_{H0} \cdot \mathbf{a}$ and $\mathbf{H0} = \mathbf{b} \cdot \mathbf{C}_{H0}$ are illustrated below.

$$\mathbf{H0} = \begin{bmatrix} C(t_1,f_1) & C(t_1,f_2) & \ldots & C(t_1,f_k) \\ C(t_2,f_1) & C(t_2,f_2) & \ldots & C(t_2,f_k) \\ \vdots & \vdots & \vdots & \vdots \\ C(t_j,f_1) & C(t_j,f_2) & \ldots & C(t_j,f_k) \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{k} C(t_1,f_i) \\ \sum_{i=1}^{k} C(t_2,f_i) \\ \vdots \\ \sum_{i=1}^{k} C(t_j,f_i) \end{bmatrix}$$

$$\mathbf{H0} = \begin{bmatrix} 1 & 1 & \ldots & 1 \end{bmatrix} \begin{bmatrix} C(t_1,f_1) & C(t_1,f_2) & \ldots & C(t_1,f_k) \\ C(t_2,f_1) & C(t_2,f_2) & \ldots & C(t_2,f_k) \\ \vdots & \vdots & \vdots & \vdots \\ C(t_j,f_1) & C(t_j,f_2) & \ldots & C(t_j,f_k) \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{j} C(t_i,f_1) & \sum_{i=1}^{j} C(t_i,f_2) & \ldots \sum_{i=1}^{j} C(t_i,f_k) \end{bmatrix}$$

4.2. Fault coverage analysis

An element $\mathbf{H0}(i, 1), 1 \leq i \leq j$ of column vector $\mathbf{H0}$ indicates the fault coverage of transition t_i, i.e. the total number of faults that are exhibited across t_i which can thus be detected by a test suite executing this transition. An element $\mathbf{H0}(1, i), 1 \leq i \leq k$ signifies the visibility of the fault f_i, where higher value indicates that f_i is exhibited across large number of transitions.

Note that fault coverage of a test generation technique is the ratio of faults detected by that technique with respect to the total number of faults. Thus given a coverage matrix, which could be the one corresponding to complete FSM, heuristics or CRTS strategy, one can use the vector $\mathbf{x}$ to determine the total number of faults that can be detected by the test generation technique x. The fault coverage FC_x of test generation technique x is computed by using

Table 2
Upper bound for various test generation techniques. $T = |U||R|$, $T_g = |UG||R|$, and $|\mathcal{T}|$ indicates the total number of transitions in M.

Procedure	Upper bound on	
	$\lvert Q\rvert$	$\lvert\mathcal{T}\rvert$
A – H0	3^T	$4T\lvert Q\rvert$
B – H1	$2^T + \sum_{i=0}^{T} C_i^T 2^i$	$2T\lvert Q\rvert$
B – H2	2^T	$2T(\lvert Q\rvert + 1)$
B – H3	No FSM	$4T$
B – H4	$\lvert U\rvert 3^{\lvert R\rvert}$	$4T\lvert Q\rvert$
B – H5	$\lvert R\rvert 3^{\lvert U\rvert}$	$4T\lvert Q\rvert$
B – H6	3^{T_g}	$4T\lvert Q\rvert$
C – RTk	3^T	ik

$C_r^n = \frac{n!}{r!C}$

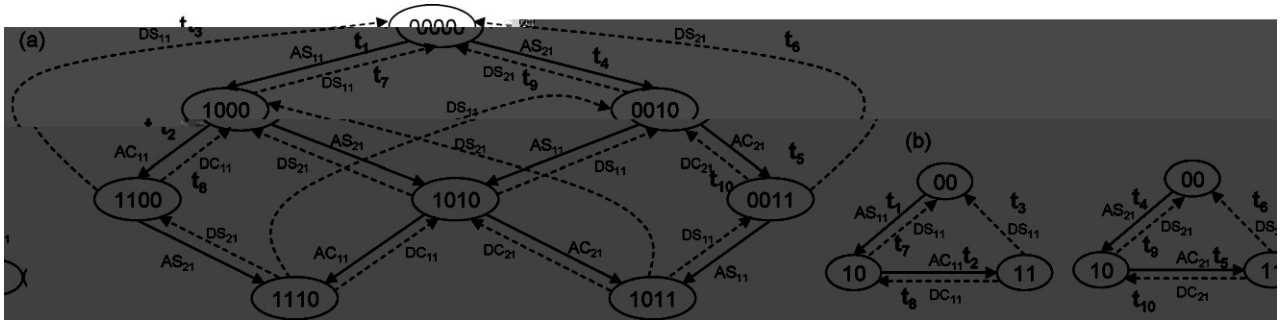

Fig. 4. Comparison of number of distinct transitions in complete FSM and machines considered in H4 (a) FSM(P), (b) M_{u_1} and M_{u_2}.

fault coverage of a test generation technique implies ability of the corresponding test set to detect f. The upper bound on the probability of detection of f by the technique $x \in \{H1, H2, \ldots, H6, RTi\}$ is given as:

$$p(f|x) = \sum_{i=1}^{|\mathcal{T}_x|} \frac{1}{|\mathcal{T}_{H0}|} = \frac{|\mathcal{T}_x|}{|\mathcal{T}_{H0}|} \tag{1}$$

The probability of detecting f using test generation technique x depends upon the total number of transitions considered in x. It would be equal to the probability of the union of events representing that f is detected by any $t' \in \mathcal{T}_x$. As these events are disjoint therefore the total probability is simply equal to the summation of probabilities of detecting f corresponding to all $t' \in \mathcal{T}_x$.

The boundary instances have been considered because of the perceived extremeness of these cases. $Instance_A$ considers one extreme of one-to-one correspondence between faults and transitions in the complete FSM such that number of transitions is equal to the number of faults; whereas, in the other extreme $Instance_B$ considers a single fault with uniform probability in the complete FSM. In the next section we consider more general instances of fault distribution.

5. General instances of fault distribution

We consider two general instances of fault distribution. $Instance_C$: It represents the more general form of $Instance_A$ considered in Section 4. In this case the total number of faults may not be equal to the total number of transitions. It is considered that given a fault $f \in F$, the total number of transitions in $\mathcal{T}_{H0}$ across which f is exhibited is uniformly distributed. We refer to this as distribution of f. Moreover we consider that the distribution of any two different faults $f_1, f_2 \in F$ is independent.

Considering a fault $f \in F$ and a transition $t \in \mathcal{T}_{H0}$. The probability of f being exhibited across t can be calculated as:

$$p(f \perp t) = p(f \perp t|f \sim 1)p(f \sim 1) + p(f \perp t|f \sim 2)p(f \sim 2) + \cdots + p(f \perp t|f \sim |\mathcal{T}_{H0}|)p(f \sim |\mathcal{T}_{H0}|)$$

where $p(f \perp t)$ is the probability of f being exhibited across t. The terms $p(f \sim i)$, $1 \le i \le |\mathcal{T}_{H0}|$ and $p(f \perp t \,|\, f \sim i)$ respectively denote the probability that f is exhibited across exactly i transitions and the probability of f being exhibited across t given that f is exhibited across exactly i transitions.

As f is uniformly exhibited across all transitions therefore $p(f \sim 1) = p(f \sim 2) = \cdots = p(f \sim |\mathcal{T}_{H0}|) = 1/|\mathcal{T}_{H0}|$. Note that $p(f \perp t|f \sim i) = i/|\mathcal{T}_{H0}|$, $1 \le i \le |\mathcal{T}_{H0}|$. Thus the above equation can be written as:

$$p(f \perp t) = \frac{1}{|\mathcal{T}_{H0}|} \sum_{i=1}^{|\mathcal{T}_{H0}|} \frac{i}{|\mathcal{T}_{H0}|} \tag{2}$$

In this case each element $C(t,f)$ of the coverage matrix $\mathbf{C}_{H0}$ would be equal to the value given by Eq. (2). The probability of detecting f by a test generation technique x is thus the probability of the disjunction of events corresponding to exhibition of f across all $t \in \mathcal{T}_x$. This is given as:

$$p(f|x) = p(f \perp t_1 \vee f \perp t_2 \vee \ldots \vee f \perp t_{|\mathcal{T}_x|}) \tag{3}$$

where $p(f|x)$ in Eq. (3) is evaluated using the well known result from probability theory (Leon-Garcia, 1994), that

$$P\left[\bigcup_{k=1}^{n} A_k\right] = \sum_{j=1}^{n} P[A_j] - \sum_{j<k} P[A_j \cap A_k] + \cdots + (-1)^{n+1} P[A_1 \cap \ldots \cap A_n]$$

A term with two entities such as $p(f \perp t \wedge f \perp t')$, $t \neq t'$, required for evaluating Eq. (3) is computed as,

$$\begin{aligned} p(f \perp t \wedge f \perp t') &= p(f \perp t \wedge f \perp t'|f \sim 1)p(f \sim 1) + \cdots \\ &\quad + p(f \perp t \wedge f \perp t'|f \sim |\mathcal{T}_{H0}|)p(f \sim |\mathcal{T}_{H0}|) \\ &= \frac{1}{|\mathcal{T}_{H0}|^2(|\mathcal{T}_{H0}|-1)} \sum_{i=1}^{|\mathcal{T}_{H0}|} i(i-1) \end{aligned}$$

where in general $p(f \perp t_1 \wedge \ldots \wedge f \perp t_i)$, $1 \le i \le |\mathcal{T}_x|$ is evaluated as,

$$\begin{aligned} p(f \perp t_1 \wedge \ldots \wedge f \perp t_i) &= \frac{1}{|\mathcal{T}_{H0}|^2(|\mathcal{T}_{H0}|-1)\ldots(|\mathcal{T}_{H0}|-|\mathcal{T}_x|-1)} \\ &\quad \times \sum_{i=1}^{|\mathcal{T}_{H0}|} i(i-1)\ldots(i-|\mathcal{T}_x|-1) \end{aligned}$$

which can be used for evaluating the terms corresponding to the conjunction of events in the expansion of Eq. (3).

By virtue of independence of fault distribution of all faults $f \in F$, the probability of detecting all faults $f \in F$ using testing technique x is simply:

$$p(F|x) = \prod_{m=1}^{|F|} p(f_m|x) \tag{4}$$

$Instance_D$: It is the more general form of $Instance_B$. In this case it is considered that the number of faults is more than one, i.e. $|F| = k > 1$. However, the fault distribution is the same as in the $Instance_B$ considered in Section 4, i.e. each $f \in F$ has equal probability of being exhibited across any transition $t \in \mathcal{T}_{H0}$. In this case the probability of detecting all faults $f \in F$ using the test generation technique x is:

$$p(F|x) = \prod_{m=1}^{|F|} \left(\sum_{i=1}^{|\mathcal{T}_x|} \frac{1}{|\mathcal{T}_{H0}|} \right) = \left(\frac{|\mathcal{T}_x|}{|\mathcal{T}_{H0}|} \right)^{|F|} \tag{5}$$

Eq. (5) illustrates that in case of fault distribution as per $Instance_D$, the probability of detection of all faults for any test generation technique $x \neq$ H0, decreases exponentially with the increase in the number of faults in the M'. It is to be noted that for both the $Instance_C$ and $Instance_D$ the computation of $p(F|x)$ has been possible because of the specific types of fault distributions considered and the independence of fault distributions; which may not be the case for general fault distributions.

6. Simulation

We simulated five different fault distributions to evaluate the fault coverage of test suites generated by the application of heuristics-based and CRTS strategies on the RBAC policy of Example 1. The other variable in this simulation is fault density set to $|F|/|\mathcal{T}_{H0}|$.

6.1. Setup

Application of heuristic H5 was not considered in the simulation as M_{r_1} is the same as FSM(P) for the policy in Example 1. Moreover, grouping of users was not done because of only two users in P, hence H6 was not applied. The CRTS strategy considered four test suites each consisting of 10 tests of lengths 2, 4, 6 and 8, labeled respectively as RT2, RT4, RT6 and RT8. These lengths were chosen to be comparable with the lengths of tests in the test suite generated from complete FSM. The size of CRTS suites (10 tests in each suite) was selected to be comparable with the size of largest test suite obtained by the application of heuristics.

The fault coverage data was averaged over a total of 10,000 iterations, where in each iteration coverage data was collected for all the test generation techniques corresponding to the five cases of fault distribution. The cases of fault distribution are labeled sequentially as Case 0, Case 1, . . ., Case 4. Case 0 and 4 respectively correspond to $Instance_C$ and $Instance_D$ in Section 5. Cases 1, 2 and 3 respectively correspond to the fault distribution in which 75%, 50% and 25% of faults are uniformly exhibited as in Case 0 and rest are exhibited as per Case 4. Cases 1, 2 and 3 achieve the required fault distribution in the limiting sense as during each iteration of the simulation each fault is exhibited along the transitions as per Case 0 with probability 0.75, 0.50 and 0.25 respectively.

6.2. Results and discussion

Fig. 5(a)–(e) illustrates the average fraction of faults detected by all the test suites corresponding to the five different values of fault density. It can be observed that fault coverage of all the testing techniques follows nearly the same trend for all the values of fault density. In other words there is not much variation in fault coverage results as fault density changes, which is expected as all faults are independently and identically distributed. However, as discussed later the variation in probability of detection of all faults ($p(F)$) with change in fault density is much more noticeable as exhibited in Fig. 6.

The common trend demonstrated in the average fault detection of all testing techniques, corresponding to all the cases of fault density, highlights that as expected the average number of faults detected by each test generation technique decrease as fault distribution limits to Case 4. This drop is more dramatic for H3, H2, RT2 and RT4. This is reasonable as the corresponding test suites include much lesser number of transitions as compared to others. What is worth noticing is that for Case 0 of fault distribution all the test generation techniques perform quite well. Fig. 6(a)–(e) illustrates the probability $p(F)$ of detecting all faults for all the test suites corresponding to the five values of fault density. As shown by Figs. 6 and 5, the variation in $p(F)$ with change in fault density is much more noticeable as compared to the common trend of average fraction of fault detection for all the values of fault density. For Case 0 and 4 the simulation results are the same as would be obtained by Eqs. (4) and (5) respectively. The drop in $p(F)$ with increase in fault density is by virtue of the product of probabilities corresponding to the detection of individual faults. This result is logical and it illustrates the inability of heuristics-based and CRTS strategies to detect all the faults on the average.

The $p(F)$ results for Case 1, 2 and 3 illustrate that as more and more faults are exhibited as per Case 4, the exponential term in Eq. (5) causes the impact of faults exhibited as per Case 4 to dominate the impact of faults exhibited as per Case 0 in the overall probability $p(F)$.

Overall, the results indicate that for Case 0 and low values of fault density, both the heuristics and CRTS test suites have high average fraction of faults detected as well as the probability of detecting all faults. For RT8 the probability of detecting all faults is quite good even for high values of fault density, thus indicating that CRTS suites with longer tests have higher fault detection effectiveness which is comparable to the effectiveness of even the best heuristic.

Table 3
Comparison of fault detection effectiveness (in %) obtained in the simulation (Section 6) and the case study.

Technique	Case study		Fault distribution case				
	Simple	Malicious	0	1	2	3	4
H3	98.70	0	90.00	68.80	50.95	31.34	12.01
H4	87.80	75.00	97.80	81.52	67.41	52.73	37.79
$RT_2(RT_4)$	90.00	47.50	93.52	72.93	54.06	34.73	16.50
$RT_4(RT_6)$	100	70.00	95.43	76.20	58.82	40.48	22.70
$RT_6(RT_{10})$	100	82.50	98.27	82.76	69.25	55.74	41.45

7. Comparison with a case study

Here we compare the simulation results with the fault coverage results obtained in an empirical study conducted to assess the fault detection effectiveness associated with the use of the three procedures described in Section 3. The details of the case study are given in Masood et al. (2009) and omitted here due to space limitations. As H1 has not used in the case study and H2 corresponds to complete FSM based test generation strategy, therefore the results of these two techniques are not comparable. Moreover H5 is not compared because it has not been considered in the simulation.

Table 3 illustrates the comparison between the fault coverage results for the simple and malicious faults obtained in the case study, and the fault coverage results for the simulation for the comparable test generation techniques. Simple faults were automatically injected through first order mutations of the application source code; whereas, malicious faults were manually injected and indicated intent of a malevolent programmer. The fault coverage obtained in simulation corresponds to fault density 0.01 which is close to the approximate fault density 0.015 in the case study.

By virtue of the comparison of the length of the longest test case included in the test suites, corresponding to complete FSM considered in the case study and in simulation (maximum length 4 in simulation and 8 in case study), the average fault detection effectiveness for RT2, RT4 and RT6 in the simulation is compared with the effectiveness result for RT4, RT6 and RT10 in the case study. This is indicated by showing the comparable RT technique used in the case study in parenthesis in the column labeled “Technique” in Table 3.

Table 3 indicates that the only reasonable comparison of fault coverage for simple faults in the case study and the coverage obtained in the simulation is for Case 0. In the case study, the

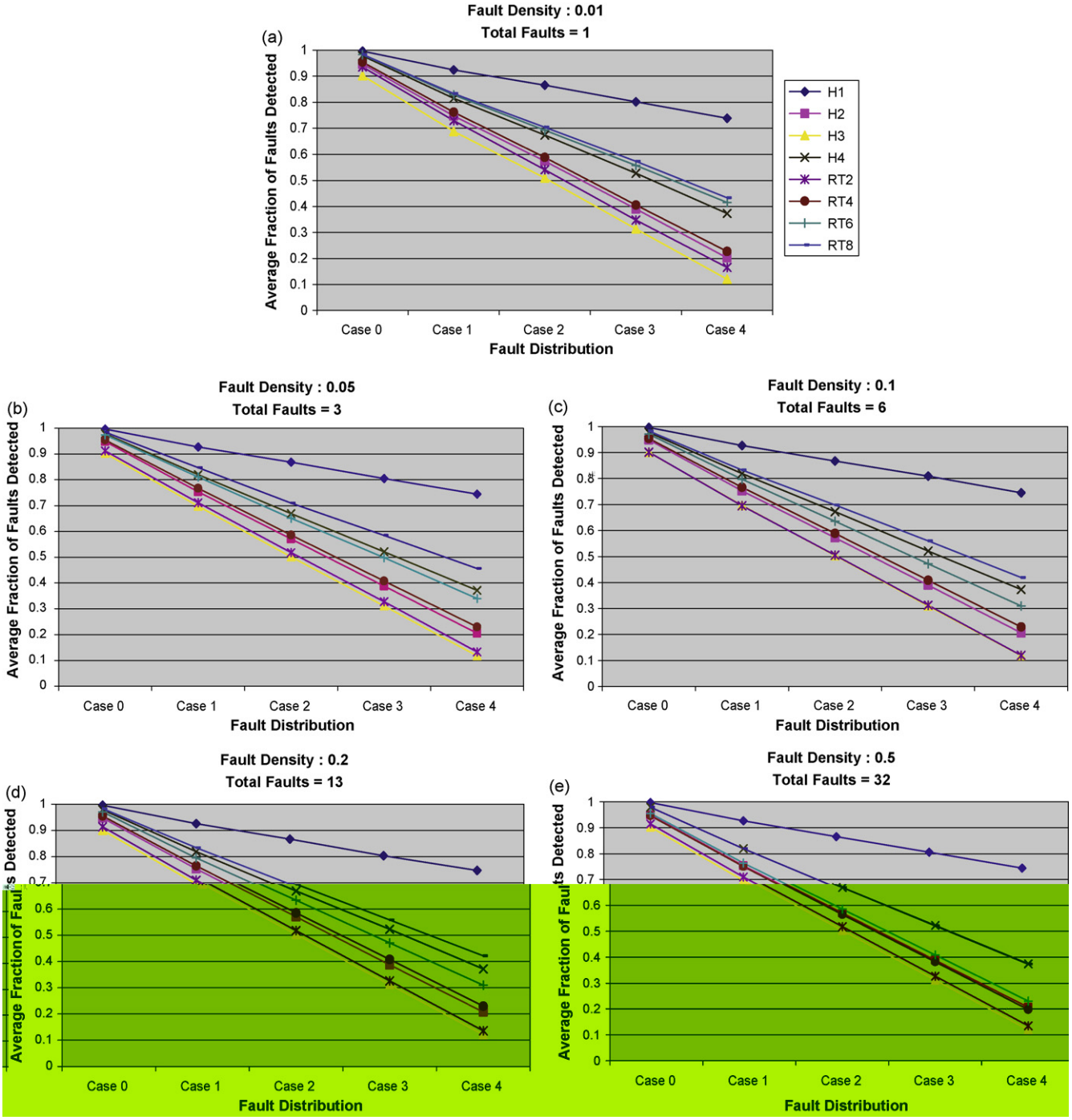

Fig. 5. Relation of average fraction of faults detected by all test generation techniques and fault density.

test suite corresponding to H3 detected 98.70% of the total faults, whereas in the simulation it is able to detect 90% of faults on the average. Test generated using H4 detected 87.80% of faults in the case study, whereas in the simulation its effectiveness is 97.80% on the average. This variation in results obtained from the case study and observed in the simulation highlights that the fault distribution in actual systems may not be uniform as considered in the Case 0 in the simulation. However, these results do illustrate that the actual fault distribution in the case study is close to the distribution considered in Case 0.

The higher fault detection effectiveness of H3 in the case study as compared to the one observed in simulation and the converse effect observed in the fault detection effectiveness values for H4 suggest that fault distribution in the case study is uniform for most of the faults and biased toward specific transitions for the remainder. The bias appears to be toward transitions out of or reaching to those specific states not modeled in the smaller machines generated using H4.

Table 3 further illustrates the wide variation between the fault coverage results obtained in the case study for the malicious faults and the results of simulation. For example, H3 did not detect any malicious fault but detected 90% faults in simulation. These results indicate that the fault distribution of malicious faults is not close to the uniform distribution considered in Case 0. We consider this observation to be logical as malicious faults are injected by a malicious programmer whose intent is to exploit some particular

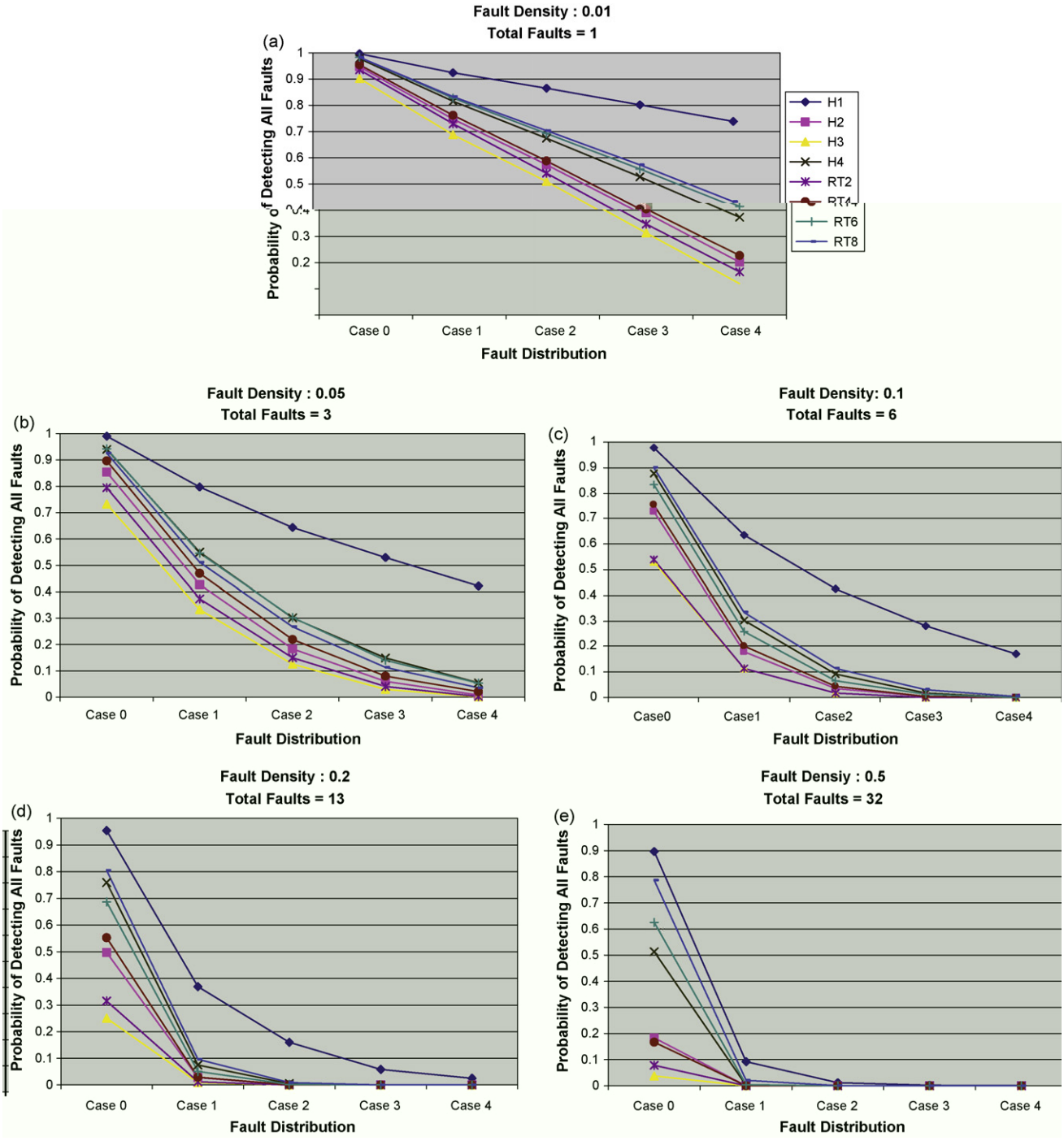

Fig. 6. Relation of probability of detection of all faults for all test generation techniques and fault density.

aspects of the system. A malicious programmer is thus not expected to inject faults uniformly. Therefore, it is possible that a test generation technique lagging another in simple fault detection, may out perform the latter in detecting malicious faults, such as the performance of H4 and H3 in the case study. H4 performed better than H3 in detecting malicious faults because malicious faults in the case study were specifically exhibited across transitions not included in the test suite corresponding to H3 but included in the test suite corresponding to H4. RT4, RT6 and RT10 respectively detected 90%, 100% and 100% of simple faults in the case study. The average fault detection for the comparable RT2, RT4 and RT6 respectively was 93.52%, 95.43% and 98.27% in the simulation. In addition to the difference in the simulated and the actual fault distribution, another reason for this variation in fault detection effectiveness, can be the limited number of random test suites considered (only 5) in the case study. Each iteration of simulation constructed a new random test suite for each of RT2, RT4, RT6 and RT8.

The variation between coverage results for malicious faults in the case study and the results obtained in the simulation is less for random test suites as compared to the test suites generated using heuristics. Due to the randomness in selecting transitions in the randomly generated tests, the transitions included in the test suites are not constrained to a specific set as is the case in the test suites generated using heuristics. Thus random test suites have bet-

ter coverage of malicious faults as compared to those generated from machines obtained by applying heuristics.

From the comparison between fault coverage results of simulation and the case study, it can be observed that the coverage results corresponding to Case 0 can be used as a reasonably good predictor for fault coverage of simple faults corresponding to the application of heuristics and the CRTS strategy. This is an encouraging result as it allows one to utilize analytical results for comparison of fault coverage of various test generation techniques. However, a word of caution is due: this observation is based on only one case study and may not hold across all access control implementations. Additional case studies might help in the refinement of the analytical framework proposed in Section 4. The results also illustrate that by virtue of the malicious intent involved in injecting malicious faults, it may not be possible to relate the distribution of malicious faults with any such distribution that considers uniform distribution of faults.

8. Related work

Fault coverage analysis in the context of protocol conformance testing based on finite state models has been discussed in Petrenko et al. (1996). The fault coverage of a test suite is measured as the ratio of the number of machines which fail on the test suite and the total number of faulty machines. The faulty machines can be generated either by exhaustive mutation analysis, random construction or structural analysis. Both the exhaustive mutation analysis and random construction approaches require part of the implementation machines to be generated and executed against the given test suite (Petrenko et al., 1996), thus making these approaches difficult to apply in practice.

The FSM based technique for fault coverage analysis is not suitable for fault coverage analysis with respect to RBAC fault model, as in contrast to one-to-one mapping between FSM faults and structural mutations of the model, a many-to-many mapping may exist between RBAC faults and the corresponding structural mutations of the FSM. As considered in Petrenko et al. (1996), the structural mutants correspond uniquely to the output, transfer, or missing/extra state FSM faults; whereas, in case of RBAC faults (Section 2.2) the mappings may not be unique. As an example consider a UA2 fault in the ACUT corresponding to Example 1 which permits both the users u_1 and u_2 to simultaneously activate r_1, thus causing violation of dynamic role cardinality constraint. This single fault in ACUT would correspond to two faults, incorrect transfer and extra state faults, in the FSM fault model.

Therefore, instead of the faulty machines based approach considered in Petrenko et al. (1996), we considered the coverage matrix based approach, introduced in (Rosenblum and Weyuker, 1997) and later extended in Sedigh-Ali et al. (2002) for temporal modeling of fault coverage, because the coverage matrix can be used to model the many-to-many relation between the RBAC faults and the FSM model. Moreover, coverage matrix is more suitable for studying the effect of varying the fault distributions on fault coverage. As the fault coverage analysis targeted by us over here has not been the focus of the studies in Rosenblum and Weyuker (1997) and Sedigh-Ali et al. (2002), therefore they did not consider random distribution of faults.

Although a number of test generation techniques for finite-state specifications have been proposed in literature (Lai, 2002); yet, not much research is reported in regards to test generation for RBAC systems. As the RBAC test generation techniques considered by Masood et al. (2009) are based on test generation from Finite State Machines (FSM), thus it is possible that more test generation procedures can be formulated by using the other FSM based test generation techniques. This subject has been already attempted in Masood et al. (2009), thus has not been the scope of this paper.

Test generation techniques for access control policies have also been proposed by Martin and Xie (2007) and Pretschner et al. (2008). The proposed fault coverage model is not applicable for the test generation approaches considered in Martin and Xie (2007) and Pretschner et al. (2008) as FSM modeling of RBAC policy (a pre-requisite for our work) has not been considered in both the works. Moreover the fault model considered in Martin and Xie (2007) is quite generic and no specific RBAC fault model (Section 2.2) has been considered in Pretschner et al. (2008). Additionally Martin and Xie (2007) considers the test generation with the perspective of verifying the correctness of access control policy; more of a policy validation technique in contrast to the access control implementation (ACUT) verification considered in the test generation approaches deliberated by us. Similarly the work in Pretschner et al. (2008) also focuses on policy validation using combinatorial test generation techniques, which are then compared based on mutation scores.

The relationship between partition testing and random testing has been studied by various researchers including Gutjahr (1999) and Ntafos (2001). They considered random distribution of failure rates for each program sub-domain. Such analysis cannot be used for comparison of heuristics and CRTS based test generation techniques as the transitions considered in the test suites of heuristics do not form a partition of $\mathcal{T}_{H0}$. Moreover, even under the case where transitions can be partitioned, the fundamental assumption of uniform input domain distribution (for comparison transitions in our framework can be considered as inputs in Gutjahr (1999) and Ntafos (2001)) does not hold.

9. Summary

In this paper a probabilistic model of fault coverage is proposed to formally study the fault coverage of heuristics and CRTS based test generation techniques in relation with conformance test suite generation for RBAC systems. After proposing a formal framework in Section 4 for analyzing the fault coverage of various test generation techniques, the detailed analysis of two general cases of fault distribution is presented in Section 5. The results of the simulation, reported in Section 6, indicate that as expected the fault coverage of all test generation techniques decrease as the fault distribution limits to the non-uniform case. Moreover, fault coverage of a test generation technique with lesser number of transitions in the corresponding test suite, as compared to another technique, is lower.

The simulation results indicate that, as expected, the probability of fault detection decreases drastically with an increase in fault density, because it is computed as the product of probabilities corresponding to the detection of individual faults. Moreover the results indicate that CRTS suites with longer tests always perform better than the suites with smaller tests, thus advocating the use of longer length test suites in practice while considering the trade off between higher effectiveness and higher cost. Finally the comparison of the simulation results with a case study does indicate usefulness of the proposed formal framework in estimating fault coverage of a test generation scheme for real RBAC implementations.

Acknowledgements

This research has been supported by the National Science Foundation under NSF Grant IIS-0964639 and a grant from the Software Engineering Research Center.

References

Chow, T.S., 1978. Testing software design modelled by finite state machines. IEEE Transactions on Software Engineering SE-4 (May (3)), 178–187.

DeMillo, R.A., Lipton, R.J., Sayward, F.G., 1978. Hints on test data selection. IEEE Computer 11 (April (4)), 34–41.

Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R., 2001. Proposed NIST standard for role-based access control. ACM Transactions on Information and Systems Security 4 (3), 224–274.

Fujiwara, S., Bochmann, G.v., Khendek, F., Amalou, M., Ghedamsi, A., 1991. Test selection based on finite state models. IEEE Transactions on Software Engineering 17 (June (6)), 591–603.

Gill, A., 1968. Finite-State Models for Logical Machines. John Wiley and Sons, Inc., New York.

Gutjahr, W.J., 1999. Partition testing vs. random testing: the influence of uncertainty. IEEE Transactions on Software Engineering 25 (November (5)), 661–674.

Lai, R., 2002. A survey of communication protocol testing. The Journal of Systems and Software 62, 21–46.

Leon-Garcia, A., 1994. Probability and Random Processes for Electrical Engineering. Addison-Wesley Publishing Company, Massachusetts.

Martin, E., Xie, T., 2007. Automated test generation for access control policies via change-impact analysis. In: Third International Workshop on Software Engineering for Secure Systems, pp. 5–12.

Masood, A., Bhatti, R., Ghafoor, A., Mathur, A., 2009. Scalable and effective test generation for role based access control systems. IEEE Transaction on Software Engineering 35 (5), 654–668.

Ntafos, S.C., 2001. On comparisons of random, partition, and proportional partition testing. IEEE Transaction on Software Engineering 27 (10), 949–960.

Petrenko, A., Bochmann, G.v., Yao, M., 1996. On fault coverage of tests for finite state specifications. Computer Networks and ISDN Systems 29 (1), 81–106.

Pretschner, A., Mouelhi, T., Traon, Y.L., 2008. Model-based tests for access control policies. In: IEEE International Conference on Software Testing, Verification and Validation, pp. 338–347.

Rosenblum, D.S., Weyuker, E.J., 1997. Using coverage information to predict the cost-effectiveness of regression testing strategies. IEEE Transactions on Software Engineering 23 (3), 146–156.

Sandhu, R., 1998. Role activation hierarchies. In: RBAC'98: Proceedings of the Third ACM Workshop on Role-based Access Control. ACM Press, New York, NY, USA, pp. 33–40.

Sandhu, R., Samarati, P., 1994. Access control: principles and practice. IEEE Communications 32 (9), 40–48.

Sedigh-Ali, S., Ghafoor, A., Paul, R.A., 2002. Temporal modeling of software test coverage. In: 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment, COMPSAC'02, pp. 823–828.

Ammar Masood recieved his PhD from the School of Electrical and Computer Engineering at Purdue University, West Lafayette, Indiana. He is currently working as Faculty Member in the Department of Avionics Engineering at the Institute of Avionics and Aeronautics, Air University, Islamabad, Pakistan. His research interests include information system security, software security testing and verification, access control systems, multimedia systems, and networking.

Arif Ghafoor is currently a Professor in the School of Electrical and Computer Engineering, at Purdue University, West Lafayette, IN, and is the Director of Distributed Multimedia Systems Laboratory. He has been actively engaged in research on multimedia information systems, database security, and parallel and distributed computing. Dr. Ghafoor has co-edited a book entitled "Multimedia Document Systems in Perspectives" and has co-authored a book entitled "Semantic Models for Multimedia Database Searching and Browsing". He has received the IEEE Computer Society 2000 Technical Achievement Award for his research contributions in the area of multimedia systems.

Aditya Mathur's research spans software testing, reliability, process control, and a study of the function of the human brain. His published work relates to investigations into the effectiveness of testing techniques and their applicability to the testing of sequential and distributed software, methods for the estimation of software reliability, and techniques and tools for managing a collection of Internet-enabled devices. He is the author of several textbooks including the recently published "Foundations of Software Testing".