МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ, МОЛОДЕЖИ И СПОРТА УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ РАДИОЭЛЕКТРОНИКИ

ISSN 0135-1710

АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ И ПРИБОРЫ АВТОМАТИКИ

Всеукраинский межведомственный научно-технический сборник

Основан в 1965 г.

Выпуск 161

Харьков
2012

СОДЕРЖАНИЕ

УДК 62-50

В сборнике представлены результаты исследований, касающихся компьютерной инженерии, управления, технической диагностики, автоматизации проектирования, оптимизированного использования компьютерных сетей и создания интеллектуальных экспертных систем. Предложены новые подходы, алгоритмы и их программная реализация в области автоматического управления сложными системами, оригинальные информационные технологии в науке, образовании, медицине.

Для преподавателей университетов, научных работников, специалистов, аспирантов.

У збірнику наведено результати досліджень, що стосуються комп'ютерної інженерії, управління, технічної діагностики, автоматизації проектування, оптимізованого використання комп'ютерних мереж і створення інтелектуальних експертних систем. Запропоновано нові підходи, алгоритми та їх програмна реалізація в області автоматичного управління складними системами, оригінальні інформаційні технології в науці, освіті, медицині.

Для викладачів університетів, науковців, фахівців, аспірантів.

Свидетельство о государственной регистрации
печатного средства массовой информации КВ № 12073-944ПР от 07.12.2006 г.

Адрес редакционной коллегии: Украина, 61166, Харьков, просп. Ленина, 14, Харьковский национальный университет радиоэлектроники, комн. 321, тел. 70-21-326

УДК 658.512.011:681.326:519.713

В.И. ХАХАНОВ, ANDERS CARLSSON, С.В. ЧУМАЧЕНКО, БУТЕНКО С.А.

МОДЕЛИ УПРАВЛЕНИЯ УЯЗВИМОСТЬЮ

Предлагается математический аппарат создания инфраструктуры программно-аппаратных телекоммуникационных информационных кибернетических систем (КС), ориентированной на защиту от несанкционированного доступа к сервисам, определенным в спецификации системы, путем проникновения через легальные интерфейсы взаимодействия компонентов, имеющие уязвимости. Инфраструктура защитных сервисов создается вместе с киберсистемой и сопровождает последнюю в течение всего жизненного цикла, обслуживая все последующие модификации КС, и сама постоянно повышает свой интеллект путем пополнения истории и библиотек конструктивных и деструктивных компонентов.

1. Модель процессов тестирования уязвимостей и проникновений

Предлагаются технологичные и эффективные процесс-модели и методы диагностирования уязвимостей или функциональных нарушений в программных и/или аппаратных компонентах киберсистемы. Используются регистровые (векторные) или матричные (табличные) структуры данных, которые ориентированы на параллельное выполнение логических операций при поиске уязвимых компонентов КС.

Проблема синтеза или анализа компонентов произвольной системы и инфраструктуры сервисов может быть сформулирована в виде взаимодействия (симметрической разности – аналог xor-операции на булеане) в кибернетическом пространстве ее модели F с входными воздействиями T и реакциями L: $f(F,T,L)=\varnothing \rightarrow F\Delta T\Delta L=\varnothing$.

Киберпространство – совокупность взаимодействующих по метрике информационных процессов и явлений, использующих в качестве носителя компьютерные системы и сети. В частности, компонент пространства представлен k-мерным (кортежем) вектором $a=(a_1,a_2,...,a_j,...,a_k)$, $a_j=\{0,1\}$ в двоичном алфавите. Нуль-вектор есть k-мерный кортеж, все координаты которого равны нулю: $a_j=0,\ j=\overline{1,k}$.

Метрика β кибернетического (двоичного) пространства определяется единственным равенством, которое формирует нуль-вектор для xor-суммы расстояний d_i между ненулевым и конечным числом точек (объектов), замкнутых в цикл: $\beta=\bigoplus_{i=1}^{n} d_i = 0.$

Иначе: метрика β векторного логического двоичного пространства есть равная нуль-вектору xor-сумма расстояний между конечным числом точек (вершин) графа, образующих цикл. Сумма n-мерных двоичных векторов, задающих координаты точек цикла, равна нуль-вектору. Данное определение метрики оперирует отношениями, что позволяет сократить систему аксиом с трех до одной и распространить ее действие на любые конструкции n-мерного киберпространства.

Метрика β кибернетического многозначного пространства, где каждая координата вектора (объекта) определена в алфавите, составляющем булеан на универсуме примитивов мощностью p: $a_j=\{\alpha_1,\alpha_2,...,\alpha_r,...,\alpha_m\}$, $m=2^p$, есть равная $\varnothing$-вектору (по всем координатам) симметрическая разность расстояний между конечным числом точек, образующих цикл: $\beta=\Delta_{i=1}^{n} d_i=\varnothing.$

Равенство пустому вектору симметрической разности покоординатного теоретико-множественного взаимодействия подчеркивает равнозначность компонентов (расстояний), формирующих уравнения.

Введенная метрика представляет не только теоретический интерес, но имеет и практическую направленность на обобщение и классификацию задач тестирования путем создания модели xor-отношений на множестве из четырех основных компонентов. Процедуры синтеза тестов, моделирования уязвимостей и их диагностирования можно свести к xor-отношениям на графе (рис. 1) полного взаимодействия четырех вершин (функциональность, киберсистема (Unit Under Test), тест, уязвимости) $G = \{F, U, T, L\}$.

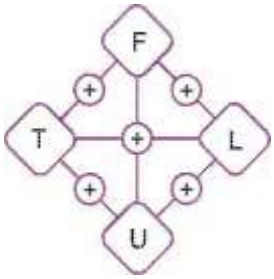


Рис. 1. Граф взаимодействия компонентов тестирования

Такой граф порождает четыре базовых треугольника, которые формируют 12 практически полезных триад отношений, формулирующих задачи тестирования:

$T \oplus F \oplus L = 0$	$T \oplus L \oplus U = 0$	$T \oplus F \oplus U = 0$	$F \oplus L \oplus U = 0$
1) $T = F \oplus L$	4) $T = L \oplus U$	7) $T = F \oplus U$	10) $F = L \oplus U$
2) $F = T \oplus L$	5) $L = T \oplus U$	8) $F = T \oplus U$	11) $L = F \oplus U$
3) $L = T \oplus F$	6) $U = T \oplus L$	9) $U = T \oplus F$	12) $U = F \oplus L$

Введение вершины U в граф взаимодействия компонентов тестирования расширяет функциональные возможности модели, появляются новые свойства полученной системы. Введение в структуру новой вершины должно иметь весомые аргументы в пользу ее целесообразности. Что касается представленного на рис. 1 графа, содержательно все формульно описанные задачи можно классифицировать в группы следующим образом.

Группа 1 – теоретические эксперименты (на модели функциональности), без киберсистемы: 1) Синтез теста по модели функциональности для заданного списка уязвимостей. 2) Построение модели функциональности на основе заданного теста и списка уязвимостей. 3) Моделирование уязвимостей функциональности на заданном тесте.

Группа 2 – реальные эксперименты (на киберсистеме), без модели функциональности: 4) Синтез теста путем физической эмуляции уязвимостей в КС. 5) Определение списка уязвимостей устройства при выполнении диагностического эксперимента. 6) Верификация теста и уязвимостей в эксперименте на реальной КС.

Группа 3 – тестовые эксперименты (верификация), без уязвимостей: 7) Синтез теста путем сравнения результатов моделирования модели и реальной КС. 8) Синтез функциональности по реальной КС и заданному тесту. 9) Верификация теста и модели функциональности относительно реальной КС с существующими уязвимостями.

Группа 4 – эксперименты в процессе функционирования, на рабочих воздействиях: 10) Проверка правильности поведения реальной КС на существующих или заданных уязвимостях. 11) Проверка работоспособности устройства относительно существующей модели в процессе функционирования. 12) Верификация функциональности и списка уязвимостей относительно поведения реального КС.

Наиболее популярными задачами из перечисленного выше списка являются: 1, 3, 5, 8, 9. Можно ввести и другую классификацию типов задач, которая дает возможность определить на графе $G = (F, U, T, L)$ все концептуальные пути решения целевых проблем: синтеза тестов, определения модели функциональности, генерирования модели уязвимостей и проектирования КС:

1) $T = F \oplus L$; 4) $F = T \oplus L$; 7) $L = T \oplus F$; 10) $U = T \oplus L$;

2) $T = U \oplus L$; 5) $F = U \oplus L$; 8) $L = T \oplus U$; 11) $U = T \oplus F$;

3) $T = F \oplus U$; 6) $F = T \oplus U$; 9) $L = F \oplus U$; 12) $U = F \oplus L$.

Все конструкции, представленные в отношениях, обладают замечательным свойством обратимости. Компонент, вычисляемый с помощью двух других, может быть использован в качестве аргумента для определения любого из двух исходных. Потому здесь можно говорить о транзитивной обратимости каждой триады отношений на полном графе, когда по двум любым компонентам всегда и однозначно можно восстановить или определить третий. При этом формат представления каждого компонента должен быть одинаковым по форме и размерности (векторы, матрицы). На основе предложенной метрики и моделей тестирования далее рассмотрим более подробно методы диагностирования уязвимостей или функциональных нарушений.

2. Граф-метод поиска функциональных нарушений в КС

Используется уравнение пространства

$$f(F,T,L,U) = 0 \rightarrow F \oplus T \oplus L \oplus U = 0,$$

которое трансформируется к виду $L = (T \oplus F) \oplus (T \oplus U)$. Диагностирование уязвимостей (функциональных нарушений) сводится к сравнению результатов модельного $(T \oplus F)$ и натурного $(T \oplus U)$ экспериментов, которое формирует список функциональных нарушений L, присутствующих в объекте диагностирования. Формула-модель процесса поиска блока F_i с функциональными нарушениями сводится к выбору решения посредством определения xor-взаимодействия между тремя компонентами:

$$L = F_i \leftarrow [(T \oplus F_i) \overset{p}{\underset{i=1}{\oplus}} (T \oplus U_i)] = 0.$$

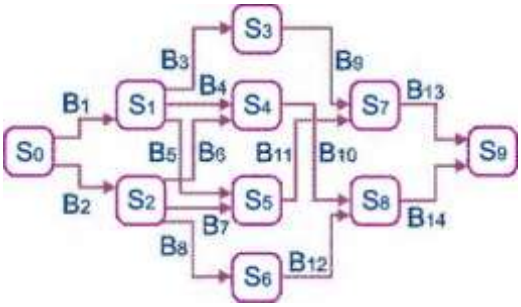
Аналитическая модель верификации HDL-кода с использованием механизма ассерций (дополнительных линий наблюдения) ориентирована на достижение заданной глубины диагностирования и представлена в следующем виде:

$$M = f(F, A, B, S, T, L), \qquad F = (A^* B) \times S;\ S = f(T, B);$$
$$A = \{A_1, A_2, ..., A_i, ..., A_n\};\ B = \{B_1, B_2, ..., B_i, ..., B_n\};$$
$$S = \{S_1, S_2, ..., S_i, ..., S_m\};\quad S_i = \{S_{i1}, S_{i2}, ..., S_{ij}, ..., S_{ip}\};$$
$$T = \{T_1, T_2, ..., T_i, ..., T_k\};\quad L = \{L_1, L_2, ..., L_i, ..., L_n\}.$$

Здесь $F = (A^* B) \times S$ – функциональность, представленная графом (рис. 2) транзакций программных блоков (Code-Flow Transaction Graph – CFTG), где $S = \{S_1, S_2, ..., S_i, ..., S_m\}$ – вершины или состояния программного продукта при моделировании тестовых сегментов. Иначе граф можно идентифицировать как ABC-граф – Assertion Based Coverage Graph. Каждое состояние $S_i = \{S_{i1}, S_{i2}, ..., S_{ij}, ..., S_{ip}\}$ определяется значениями существенных переменных КС (переменные, память, терминалы, компьютеры). Ориентированные дуги графа представлены совокупностью функциональных блоков:

$$B = (B_1, B_2, ..., B_i, ..., B_n),\ \overset{n}{\underset{i=1}{\cup}} B_i = B;\ \overset{n}{\underset{i=1}{\cap}} B_i = \varnothing,$$

где каждому из них может быть поставлена в соответствие ассерция $A_i \in A = \{A_1, A_2, ..., A_i, ..., A_n\}$. Каждая дуга B_i – группа операторов кода – формирует состояние вершины $S_i = f(T, B_i)$ в зависимости от теста $T = \{T_1, T_2, ..., T_i, ..., T_k\}$. Каждой вершине может быть поставлен ассерционный монитор, объединяющий ассерции входящих в вершину дуг $A(S_i) = A_{i1} \vee A_{i2} \vee ... \vee A_{ij} \vee ... \vee A_{in}$. Вершина может иметь более одной входящей (исходящей) дуги. Множество блоков с функциональными нарушениями представлено списком $L = \{L_1, L_2, ..., L_i, ..., L_n\}$.



$$B = (B_1B_3B_9 \vee (B_2B_7 \vee B_1B_5)B_{11})B_{13} \vee$$
$$\vee ((B_1B_4 \vee B_2B_6)B_{10} \vee B_2B_8B_{12})B_{14} =$$
$$= B_1B_3B_9B_{13} \vee B_2B_7B_{11}B_{13} \vee B_1B_5B_{11}B_{13} \vee$$
$$\vee B_1B_4B_{10}B_{14} \vee B_2B_6B_{10}B_{14} \vee B_2B_8B_{12}B_{14}.$$

Рис. 2. Пример АВС-графа для HDL-кода

Модель HDL-кода, представленная в форме АВС-графа, отображает не только структуру программного кода, но и тестовые срезы функциональных покрытий, формируемые с помощью программных блоков, входящих в рассматриваемую вершину. Последняя определяет отношение между достигнутым на тесте пространством переменных и потенциально возможным, которое формирует функциональное покрытие как мощность состояния i-вершины графа $Q = \mathrm{card}C_i^r / \mathrm{card}C_i^p$. В совокупности все вершины графа должны составлять полное покрытие пространства состояний переменных КС, которое формирует качество теста, равное 1 (100%): $Q = \mathrm{card}\bigcup_{i=1}^{m} C_i^r / \mathrm{card}\bigcup_{i=1}^{m} C_i^p = 1$. Кроме того, механизм ассерций $< A, C >$, существующий на графе, позволяет выполнять мониторинг дуг (function-coverage) $A = \{A_1, A_2, ..., A_i, ..., A_n\}$ и вершин (state- coverage) $C = \{C_1, C_2, ..., C_i, ..., C_m\}$. Ассерции на дугах отвечают за диагностирование функциональных нарушений в КС блоках. Ассерции на вершинах графа дают дополнительную информацию о качестве тестов (ассерций) в целях их улучшения или дополнения. Транзакционный граф КС блоков дает возможность: 1) использовать аппарат тестопригодного проектирования для оценки качества КС; 2) оценить затраты на создание тестов, диагностирование и исправление функциональных нарушений; 3) оптимизировать синтез теста путем решения задачи покрытия минимальным множеством активизированных путей всех дуг (вершин). Например, минимальный тест для приведенного АВС-графа имеет шесть сегментов, которые активизируют все существующие пути:

$$T = S_0S_1S_3S_7S_9 \vee S_0S_1S_4S_8S_9 \vee S_0S_1S_5S_7S_9 \vee$$
$$\vee S_0S_2S_4S_8S_9 \vee S_0S_2S_5S_7S_9 \vee S_0S_2S_6S_8S_9.$$

Тесту можно поставить в соответствие следующую матрицу активизации программных блоков:

B_{ij}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}
T_1	1	.	1	.	.	.	.	.	1	.	.	.	1	.
T_2	1	.	.	1	.	.	.	.	.	1	.	.	.	1
T_3	1	.	.	.	1	.	.	.	.	.	1	.	1	.
T_4	.	1	.	.	.	1	.	.	.	1	.	.	.	1
T_5	.	1	.	.	.	.	1	.	.	.	1	.	1	.
T_6	.	1	.	.	.	.	.	1	.	.	.	1	.	1

Матрица активизации иллюстрирует факт неразличимости на тесте функциональных нарушений в блоках 3 и 9, 8 и 12, которые составляют два класса эквивалентностей при

наличии одной ассерции (монитора) в вершине 9. Для устранения такой неразличимости необходимо создать два дополнительных монитора в вершинах 3 и 6. В результате 3 ассерции на вершинах $A = (A_3, A_6, A_9)$ дают возможность различить между собой все блоки программного кода. Таким образом, граф позволяет не только синтезировать оптимальный тест, но и определять минимальное число ассерционных мониторов здесь и далее в вершинах для поиска неисправных блоков с заданной глубиной диагностирования.

Увеличение числа ассерционных мониторов приводит к модификации таблицы активизации. Иначе, на заданном тесте и механизме ассерций необходимо однозначно решать задачу диагностирования функциональных нарушений КС с глубиной до функционального блока. При этом число ассерций и тестовых сегментов должно быть минимально допустимым для кодовой идентификации всех блоков:

$$|T| + |A| \geq \log_2 |B| = \mathrm{card}T + \mathrm{card}A \geq \log_2 \mathrm{card}B.$$

Первоначально количество ассерционных мониторов равно числу тестовых сегментов. Таблица активизации функциональных модулей позволяет выполнять идентификацию блоков кода с функциональными нарушениями на обобщенном векторе экспериментальной проверки (ассерционного мониторинга):

$$V = (V_1, V_2, ..., V_i, ..., V_n), V_i = \{0,1\}, V_i = T_i \oplus B_j, \forall j(B_{ij} = 1).$$

Координата вектора $V_i = T_i \oplus B_j = 1$ идентифицирует факт «падения» тест-сегмента на подмножестве активизированных им блоков. В соответствии с вектором V, заданным на таблице активизации с учетом приведенного выше правила вычисления его координат:

B_{ij}	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}	V
T_1	1	.	1	.	.	.	.	.	1	.	.	.	1	.	0
T_2	1	.	.	1	.	.	.	.	.	1	.	.	.	1	1
T_3	1	.	.	.	1	.	.	.	.	.	1	.	1	.	0
T_4	.	1	.	.	.	1	.	.	.	1	.	.	.	1	1
T_5	.	1	.	.	.	.	1	.	.	.	1	.	1	.	0
T_6	.	1	.	.	.	.	.	1	.	.	.	1	.	1	1

строится логическая функция функциональных нарушений КС, которая упрощается на основе использования координат вектора экспериментальной проверки V:

$$B = (\overline{T}_1 \vee B_1 \vee B_3 \vee B_9 \vee B_{13}) \wedge (\overline{T}_2 \vee B_1 \vee B_4 \vee B_{10} \vee B_{14}) \wedge$$
$$\wedge (\overline{T}_3 \vee B_1 \vee B_5 \vee B_{11} \vee B_{13}) \wedge (\overline{T}_4 \vee B_2 \vee B_6 \vee B_{10} \vee B_{14}) \wedge$$
$$\wedge (\overline{T}_5 \vee B_2 \vee B_7 \vee B_{11} \vee B_{13}) \wedge (\overline{T}_6 \vee B_2 \vee B_8 \vee B_{12} \vee B_{14});$$
$$\{V,T\} = (010101) \rightarrow$$
$$B = (0 \vee B_1 \vee B_4 \vee B_{10} \vee B_{14}) \wedge (0 \vee B_2 \vee B_6 \vee B_{10} \vee B_{14}) \wedge$$
$$\vee (0 \vee B_2 \vee B_8 \vee B_{12} \vee B_{14}) =$$
$$= (B_1 \vee B_4 \vee B_{10} \vee B_{14}) \wedge (B_2 \vee B_6 \vee B_{10} \vee B_{14}) \wedge$$
$$\vee (B_2 \vee B_8 \vee B_{12} \vee B_{14}) =$$
$$= B_1B_2 \vee B_4B_2 \vee ... \vee B_3B_6B_{12} \vee ... \vee B_{14}.$$

После преобразования конъюнктивной нормальной формы (КНФ) к дизъюнктивной нормальной форме полученные термы содержат все возможные решения в виде покрытия единичных координат вектора экспериментальной проверки одиночными или кратными функциональными нарушениями КС блоков. Выбор лучшего решения связан с определением терма ДНФ минимальной длины. В данном примере оптимальным решением является терм, содержащий один блок $B = B_{14}$, который своим функциональным нарушением покрывает три единицы в векторе экспериментальной проверки $V = (010101)$. Данный факт также очевиден из сравнения двух последних столбцов матрицы активизации B.

Другое аппаратно-ориентированное аналитическое решение связано с синтезом многовыходовой комбинационной схемы – дешифратора по матрице активизации программных блоков:

$$B_1 = T_1T_2T_3\overline{T}_4\overline{T}_5\overline{T}_6;\ B_2 = \overline{T}_1\overline{T}_2\overline{T}_3T_4T_5T_6;$$

$$B_3 = \overline{T}_1T_2T_3T_4T_5T_6;\ \ldots\ B_{14} = \overline{T}_1T_2\overline{T}_3T_4\overline{T}_5T_6.$$

Такое устройство имеет число входов, равное количеству тестовых сегментов, а выходных линий – равное числу КС блоков. При подаче на входы дешифратора вектора экспериментальной проверки формируется единичное значение на одном из его выходов. При этом номер выхода соответствует блоку, имеющему функциональные нарушения. Такая схема представляет интерес для создания замкнутой в цикл инфраструктуры тестирования и исправления функциональных нарушений, где адрес уязвимого блока может быть использован для его автоматической замены на альтернативный модуль из существующей библиотеки диверсных решений.

3. Векторно-логические методы диагностирования уязвимостей

Основная цель методов – определение места причины и вида уязвимости на структуре КС путем анализа таблиц уязвимостей (функциональных нарушений – ФН), построенных в процессе моделирования всех возможных деструктивов. Аналитическая модель проверки КС с использованием механизма ассерций ориентирована на достижение заданной глубины диагностирования уязвимостей и представлена в следующем виде:

$$M = f(F, L, T, C, A, t);\ C = \{C_1, C_2, ..., C_i, ..., C_m\};\ L = \{L_1, L_2, ..., L_i, ..., L_n\};$$

$$A(t) = \{A_1, A_2, ..., A_i, ..., A_k\};\ A \subseteq L;\ F = L \times C;\ k \le n;\ T = \{T_1, T_2, ..., T_i, ..., T_p\}.$$

Здесь C_i – группа операторов кода, нагруженная на вершину (компонент КС) L_i и формирующая ее состояние; F – функциональность, представленная транзакционным графом $F = L \times C$ в виде декартова произведения множества вершин и дуг; A – совокупность ассерций, как подмножество вершин транзакционного графа $A \subseteq L$. Метод поиска функциональных нарушений блока киберсистемы использует предварительно построенную таблицу ФН $B = [B_{ij}]$, где строка есть отношение между тестовым сегментом и подмножеством блоков $T_i \approx (B_{i1}, B_{i2}, ..., B_{ij}, ..., B_{in})$ с возможными ФН. Столбец таблицы формирует отношение между блоком и тестовыми сегментами $B_j \approx (T_{1j}, T_{2j}, ..., T_{ij}, ..., T_{pj})$, которые могут проверять блок с ФН. На стадии моделирования определяется обобщенная реакция $m = \{m_1, m_2, ..., m_i, ..., m_p\}$ механизма ассерций F на тест, путем формирования $m_i = (A_1 \vee A_2 \vee ... \vee A_i \vee ... \vee A_k)$, $A_i = \{0,1\}$ как реакции ассерций на тест-сегмент T_i. Поиск ФН основан на определении хог-операции между вектором состояния ассерций и столбцов таблицы ФН $m \oplus (B_1 \vee B_2 \vee ... \vee B_j \vee ... \vee B_n)$. Выбор решения определяется совокупностью векторов B_j с минимальным числом единичных координат

$$B = \min_{j=1,n}[B_j = \sum_{i=1}^{p}(B_{ij} \overset{p}{\oplus}_{i=1} m_i)],$$

формирующих программные блоки с ФН, проверяемые на тестовых сегментах.

Повышение адекватности модели диагностирования ФН блоков КС связано с расширением пространства реакции проекта в процессе его тестовой верификации. В этом случае следует расширить пространство существования механизма ассерций до двумерного путем модификации векторов m и A:

$$A_i = (A_{i1}, A_{i2}, ..., A_{ij}, ..., A_{ik});\ m_i = (m_{i1}, m_{i2}, ..., m_{ij}, ..., m_{ik});$$

$$B_{ij} = (B_{ij1}, B_{ij2}, ..., B_{ijr}, ..., B_{ijk});\ \{A_{ij}, m_{ij}, B_{ijr}\} = \{0,1\};\ i = \overline{1,p};\ \ j = \overline{1,n};\ \ r = \overline{1,k}.$$

При этом таблица ФН становится трехмерной по параметрам: $i = \overline{1,p}$; $j = \overline{1,n}$; $r = \overline{1,k}$ – числа строк или тест-сегментов, столбцов или функциональных блоков, ассерций или точек наблюдения для блоков соответственно:

B	A_1 B_1	A_2 B_2	A_3 B_3	A_4 B_4	A_5 B_5	A_6 B_6	A_7 B_7	A_8 B_8	m
T_1	1001	1110	0001	0110	0011	1110	0001	1110	1001
T_2	1010	1000	0001	0110	0001	0001	1110	1110	1000
T_3	1110	0001	1111	0111	1011	1110	1001	1101	0011
T_4	1000	0111	0110	1001	1000	0001	1011	1110	1011
T_5	0001	0110	1100	0111	0010	1110	0001	1000	1001
T_6	1110	1011	0110	0001	1110	1100	1001	0111	1110
T_7	0110	1101	0110	1000	0111	1011	0101	1110	0001
T_8	1000	1110	1001	0100	0101	1001	0111	0001	1110

Потенциально трехмерность таблицы ФН увеличивает объем диагностической информации, что дает возможность повысить глубину поиска уязвимости, в пределе до неделимого примитива КС. Однако для анализа трехмерной таблицы ФН необходимо модифицировать предложенные методы диагностирования.

Поиск уязвимостей по таблице функциональных нарушений на основе вектора экспериментальной проверки – реакции (m) киберсистемы на тест $m = (m_1, m_2, ..., m_i, ..., m_n), m_i = \{0,1\}$ сводится к методам анализа строк или столбцов. Первый метод основан на применении векторной xor-операции между реакцией m КС на тест, формально рассматриваемый в качестве входного вектор-столбца или маски m, и столбцов таблицы ФН $m \oplus (B_1 \vee B_2 \vee ... \vee B_j \vee ... \vee B_m)$. Для подсчета качества взаимодействия векторов $Q_j(m \oplus B_j)$ в целях выбора лучшего решения определяются столбцы с минимальным числом единиц результирующего вектора:

$$B = \min_{j=1,k} [B_j = \sum_{i=1}^{n} (B_{ij} \overset{n}{\underset{i=1}{\oplus}} m_i)].$$

Они идентифицируют и формируют примитивы с функциональными нарушениями, проверяемые на тестовых наборах. Аналитическая модель процесса получения решения в виде списка блоков с уязвимостями, присутствующих в КС, представлена в следующем виде:

$$B = \min_{j=1,k} [B_j = \sum_{i=1}^{n} (B_{ij} \overset{n}{\underset{i=1}{\oplus}} m_i)]; \; B^0 = \forall_{j=1,k} [B_j = \sum_{i=1}^{n} (B_{ij} \overset{n}{\underset{i=1}{\oplus}} m_i) = 0].$$

Здесь фигурирует вектор экспериментальной проверки, который является входным для последующего анализа таблицы ФН $m = f(A,B) \oplus f^*(A,B,L)$ есть результат проведения тестового эксперимента – сравнение функционалов (состояний выходов) эталонной $f(A,B)$ и реальной $f^*(A,B,L)$ КС с ФН L на тестовых наборах A. Во втором случае, если множество $B^0 > 1$, это означает наличие эквивалентных, не различимых на данном тесте и механизме ассерций, функциональных нарушений. Выбор лучшего решения $Y = \vee[(Q_1 \wedge Q_2) \oplus Q_1]$ с минимальным числом единичных координат из двух альтернатив, представленных векторными оценками $Q_1 \vee Q_2$, осуществляется с помощью структуры, изображенной на рис. 3.

$$Y = \vee[(Q_1 \wedge Q_2) \oplus Q_1]$$

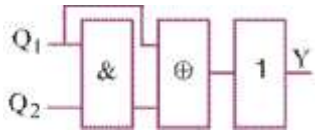


Рис. 3. Схема выбора лучшего решения

Достоинство метода – выбор лучшего решения из всех возможных одиночных и кратных ФН. По существу, в список ФН включаются такие одиночные ФН, которые при логическом умножении на вектор экспериментальной проверки дают результат в виде соответствующего вектор-столбца. Дизъюнкция всех столбцов, составляющих решение, равна вектору экспериментальной проверки $\overset{r}{\underset{j=1}{\vee}}(B_j \in B) = m$.

Вычислительная сложность метода анализа столбцов определяется следующей зависимостью: $Z^c = 3n^2 + n^2 = 4n^2$; $Z^r = 3n + n = 4n$. Здесь первая оценка учитывает выполнение координатных операций над матрицей, размерностью $n \times n$. Вторая оценка определяет вычислительную сложность регистровых параллельных операций для подсчета критериев качества и обработки матрицы соответственно.

Логический метод анализа строк таблицы ФН. Стратегия определения ошибок программного кода по таблице ФН связана с анализом ее строк, состоящим из двух процедур: 1) вычисление логического произведения конъюнкции строк, отмеченных единичными значениями вектора $T_i(m_i = 1)$, на отрицание дизъюнкции нулевых строк $T_i(m_i = 0)$ для одиночных дефектных блоков; 2) вычисление логического произведения дизъюнкции единичных строк на отрицание дизъюнкции нулевых строк для кратных дефектных блоков:

$$B^s = (\underset{\forall m_i=1}{\wedge} T_i) \wedge (\overline{\underset{\forall m_i=0}{\vee} T_i}); \quad B^m = (\underset{\forall m_i=1}{\vee} T_i) \wedge (\overline{\underset{\forall m_i=0}{\vee} T_i}); \qquad (1)$$

Формулы интересны тем, что они не привязаны к критериям качества диагностирования, а оперируют лишь двумя компонентами, таблицей ФН и вектором экспериментальной проверки. Выполнение процедуры диагностирования по формулам (1) для вектора экспериментальной проверки $m_1 = (0101010010010)$, заданного в последней таблице ФН, дает результат: $B^s(m_1, T) = D_2$, который не хуже, чем ранее полученный методом анализа столбцов. Для вектора экспериментальной проверки $m_2 = (1110011100000)$ результат диагностирования имеет вид: $B^m(m_2, T) = L_1 \vee L_2$. Вычислительная сложность метода анализа строк определяется следующей зависимостью: $Z^c = n^2$; $Z^r = n$. Первая оценка предназначена для подсчета числа координатных операций, вторая определяет вычислительную сложность процесса обработки на основе регистровых параллельных операций. Предложенные методы диагностирования уязвимостей для КС есть один из наиболее существенных компонентов инфраструктуры сервисного обслуживания проектируемых и функционирующих КС.

4. 3D-метод диагностирования уязвимостей

Мотивация определяется рыночной привлекательностью матричного метода поиска ФН в компонентах КС как самого технологичного, который ориентирован на параллельную обработку данных, что дает возможность существенно уменьшить время диагностического обслуживания при возникновении уязвимостей (проникновений).

Цель исследования – создание модели, метода и их программно-аппаратной реализации, ориентированных на существенное уменьшения времени тестирования и затрат памяти для хранения матрицы диагностирования путем формирования тернарных отношений (тест-монитор-функциональный компонент).

Задачи: 1) Разработка модели КС в виде транзакционного графа, а также матрицы активизации функциональных компонентов на тестах относительно выбранного множества

мониторов. 2) Разработка метода анализа матрицы активизации для поиска уязвимостей с заданной глубиной. 3) Синтез логических функций для встроенного диагностирования ФН.

Модель тестирования киберсистемы представлена в виде следующего преобразования начального уравнения диагноза, определенного хог-отношением параметров <тест – функциональность – уязвимые блоки>:

$$T \oplus F \oplus B = 0 \to B = T \oplus F \to B = \{T \times A\} \oplus F \to B = \{T \times A\} \oplus \{F \times m\},$$

которое оформлено в тернарное матричное отношение компонентов:

$$M = \{\{T \times A\} \times \{B\}\} \leftarrow M_{ij} = (T \times A)_i \oplus B_j .$$

Здесь координата матрицы (таблицы) равна 1, если пара тест-монитор $(T \times A)_i$ проверяет (активизирует) ФН функционального блока $B_j \in B$.

Модель киберсистемы представлена в виде транзакционного графа $G =< B, A >, B = \{B_1, B_2, ..., B_i, ..., B_n\}, A = \{A_1, A_2, ..., A_j, ..., A_m\}$, где определены множество дуг – функциональных блоков и вершин – мониторов для наблюдения совокупности примитивов КС. Для целей диагностирования на графовую модель накладывается совокупность тестовых сегментов $T = \{T_1, T_2, ..., T_r, ..., T_k\}$, которая активизирует транзакционные пути в графе. В общем случае модель тестирования представлена декартовым произведением $M =< B \times A \times T >$, которая имеет размерность $Q = n \times m \times k$. Чтобы уменьшить объем диагностической информации каждому тесту предлагается поставить в соответствие монитор, который отвечает за визуализацию пути активизации функциональных блоков, что дает возможность уменьшить размерность модели (матрицы) до $Q = n \times k$ при сохранении всех возможностей отношения триады $M =< B \times A \times T >$. Для пары тест-монитор возможны взаимно-однозначные соответствия $< T_i \to A_j >$, функциональные $< \{T_i, T_r\} \to A_j >$ и инъективные $< T_i \to \{A_j, A_s\} >$. Такое многообразие соответствий дает возможность дублировать один тестовый сегмент для различных мониторов, равно как и нагружать несколько тестов на один и тот же монитор. При этом ячейка матрицы $M_{ij} = \{0,1\}$ всегда сохраняет свою размерность, равную одному биту.

Аналитическая обобщенная модель матричного диагностирования с использованием механизма мониторов ориентирована на достижение заданной глубины поиска дефектов и представлена в следующем виде:

$$M = f(G, L, T, B, A, t);\ B = \{B_1, B_2, ..., B_i, ..., B_m\};\ L = \{L_1, L_2, ..., L_i, ..., L_n\};$$

$$A(t) = \{A_1, A_2, ..., A_i, ..., A_k\};\ A \subseteq L;\ G = L \times B;\ k \le n;\ T = \{T_1, T_2, ..., T_i, ..., T_p\}.$$

Здесь B_i – группа операторов кода, нагруженная на вершину L_i (блок КС) и формирующая ее состояние; G – функциональность, представленная транзакционным графом $G = (L, A) \times B$ в виде декартова произведения множества вершин и дуг; A – совокупность мониторов, как подмножество вершин транзакционного графа $A \subseteq L$. Метод поиска уязвимостей функциональных блоков (ФБ) использует предварительно построенную таблицу (матрицу) активизации ТАФБ $M = [M_{ij}]$, где строка есть отношение между тестовым сегментом и подмножеством активизированных блоков $T_i \to A_j \approx (M_{i1}, M_{i2}, ..., M_{ij}, ..., M_{in}), M_{ij} = \{0,1\}$, наблюдаемых на мониторе A_j. Столбец таблицы формирует отношение между функциональным блоком, тестовыми сегментами и мониторами $M_j = B_j \approx f(T, A)$. В механизм мониторов может быть введен параметр модельного времени, который частично усложняет матрицу активизации, указывая временной или модельный такт, на котором выполняется мониторинг состояния вершины или функционального блока на тест-сегменте $A_j = f(T_i, B_j, t_j)$.

Для диагностирования неисправностей на стадии моделирования определяется обобщенная реакция (вектор-столбец) $m = \{m_1, m_2, ..., m_i, ..., m_p\}$ механизма мониторов A на тест-сегменты T путем формирования $m_i = f(T_i, A_i)$. Поиск уязвимости ФБ основан на определении xor-операции между вектором состояния ассерций и столбцов таблицы ФН $m \oplus (M_1 \vee M_2 \vee ... \vee M_j \vee ... \vee M_n)$. Решение выбирается методом xor-анализа столбцов, путем выбора совокупности векторов B_j с минимальным числом единичных координат $B = \min\limits_{j=1,n}[B_j = \sum\limits_{i=1}^{p}(B_{ij} \oplus \overset{p}{\underset{i=1}{}} m_i)]$, формирующих функциональные блоки с уязвимостями, проверяемыми на тестовых сегментах. В дополнение к модели матричного диагностирования необходимо описать следующие важные свойства матрицы:

$$1)\ M_i = (T_i - A_j); \qquad 2)\ \bigvee_{i=1}^{m} M_{ij} \to \forall M_j = 1;$$

$$3)\ M_{ij} \oplus \underset{j=1}{\overset{n}{}} M_{rj} \neq M_{ij}; \qquad 4)\ M_{ij} \oplus \underset{i=1}{\overset{k}{}} M_{ir} \neq M_{ij};$$

$$5)\ \log_2 n \le k \leftrightarrow \log_2 |B| \le |T|;\ 6)\ B_j = f(T, A) \to B \oplus T \oplus A = 0.$$

Свойства означают: 1) Каждая строка матрицы есть соответствие или подмножество декартова произведения (тест-монитор). 2) Дизъюнкция всех строк матрицы дает вектор, равный единицам по всем координатам. 3) Все строки матрицы различны, что исключает тестовую избыточность. 4) Все столбцы матрицы различны, что исключает существование эквивалентных неисправностей. 5) Число строк матрицы должно быть больше двоичного логарифма от числа столбцов, что определяет потенциальную диагностируемость всех ФН блоков. 6) Функция диагностирования блока с ФН зависит от совокупного теста и мониторов, которые должны быть минимизированы без нарушения диагнозопригодности.

Для пояснения работоспособности модели и метода рассмотрим функциональности трех модулей, входящих в примера КС. Первым является компонент Row_buffer, для которого создан транзакционный граф (рис. 4). Вершины представлены состояниями переменных и мониторов, отвечающих за входящие в вершину транзакции или дуги, которым соответствуют функциональные блоки.

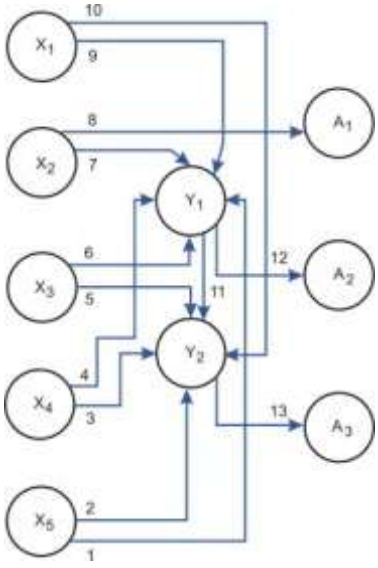


Рис. 4. Компонент Row_buffer транзакционного графа

На основе графа, полученного в процессе анализа КС, строится таблица активизации функциональных блоков, строки которой представляют пути активизации блоков к заказанной вершине-монитору. Таблица представляет собой покрытие строками-путями всех столбцов или функциональных блоков. При этом в ней не должно быть хотя бы двух одинаковых столбцов. Отличие таблицы заключается в формировании пары <тест-наблю-

даемая вершина>, что дает возможность существенно сократить ее размерность при 100% распознавании всех дефектных блоков. Здесь самое главное отличие предложенной модели заключается в возможности описания с помощью таблицы следующих отношений: различные тесты – одна вершина; один тест – различные вершины:

A_{ij}	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}
$t_1 \to D_3$	.	.	.	.	.	.	.	1	.	.	.	.	.
$t_2 \to D_1$	1	.	.	.	.	.	.	.	.	.	.	.	1
$t_3 \to D_1$	.	.	1	.	.	.	.	.	.	.	.	.	1
$t_4 \to D_1$	.	.	.	.	1	.	.	.	.	.	.	.	1
$t_5 \to D_1$	.	.	.	.	.	.	.	.	.	.	1	.	1
$t_6 \to D_1$	.	.	.	.	.	.	.	.	1	.	.	.	1
$t_7 \to D_2$	.	1	.	.	.	.	.	.	.	.	.	1	.
$t_8 \to D_2$	.	.	.	1	.	.	.	.	.	.	.	1	.
$t_9 \to D_2$	.	.	.	.	.	1	.	.	.	.	.	1	.
$t_{10} \to D_2$	.	.	.	.	.	.	1	.	.	.	.	1	.
$t_{11} \to D_2$	.	.	.	.	.	.	.	.	.	1	.	1	.

С помощью матрицы активизации функциональных блоков (транзакционного графа) и xor-метода поиска дефектов достаточно просто синтезировать логические функции для формирования комбинационной схемы, определяющей в процессе и по результатам моделирования номер функционального блока, который имеет семантические ошибки:

$$D_3 = T_8^1;$$

$$D_1 = T_{13}^1 T_1^1 \vee T_{13}^1 T_3^1 \vee T_{13}^1 T_5^1 \vee T_{13}^1 T_{11}^1 \vee T_{13}^1 T_9^1;$$

$$D_2 = T_{12}T_2 \vee T_{12}T_4 \vee T_{12}T_6 \vee T_{12}T_7 \vee T_{12}T_{10}.$$

Такое свойство становится возможным благодаря отсутствию эквивалентных уязвимостей или одинаковых столбцов в матрице активизации. Поэтому фиксация фактического состояния всех мониторов в вершинах D_1, D_2, D_3 на 11 тестовых наборах дает возможность однозначно идентифицировать некорректный функциональный модуль путем выполнения xor-операции между вектором ассерций и столбцами матрицы активизации. Нулевое значение всех координат результата xor-операции определяет номер столбца, соответствующего уязвимости модуля. Имплементация модели и метода в логическую функцию позволяет возможность определять уязвимый блок еще до завершения диагностического эксперимента, если это возможно. Это означает существенную экономию времени диагностирования отдельных видов уязвимостей. Например, тест-монитор $t_1 \to D_3$ дает возможность идентифицировать уже на первом тесте уязвимость блока B_8.

В качестве второго тестового примера для практического использования разработанной модели активизации и xor-метода поиска дефектов далее предлагается синтез матрицы диагностирования для модуля, представленного на рис. 5.

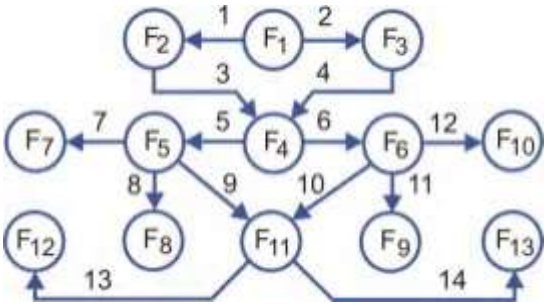


Рис. 5. Транзакционный граф main-TL

Данному графу ставится в соответствие следующая матрица диагностирования:

A_{ij}	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}
$P_1 \to F_7$	1	.	1	.	1	.	1	.	.	.	.	.	.	.
$P_2 \to F_8$	.	1	.	1	1	.	.	1	.	.	.	.	.	.
$P_3 \to F_9$	1	.	1	.	.	1	.	.	.	.	1	.	.	.
$P_4 \to F_{10}$	.	1	.	1	.	1	.	.	.	.	.	1	.	.
$P_5 \to F_{12}$	1	.	1	.	1	.	.	.	1	.	.	.	1	.
$P_6 \to F_{13}$	.	1	.	1	.	1	.	.	.	1	.	.	.	1
$P_1 \to F_2$	1	.	.	.	.	.	.	.	.	.	.	.	.	.
$P_2 \to F_3$	.	1	.	.	.	.	.	.	.	.	.	.	.	.

Система функций диагностирования имеет такой вид:

$$F_7 = T_1^1 T_3^1 T_5^1 T_7^1;\ F_8 = T_2^1 T_4^1 T_5^1 T_8^1;\ F_9 = T_{11}^1 T_6^1 T_1^1 T_3^1;$$

$$F_{10} = T_4^1 T_5^1 T_6^1 T_{12}^1;\ F_{12} = T_1^1 T_3^1 T_5^1 T_9^1 T_{13}^1;\ F_{13} = T_2^1 T_4^1 T_6^1 T_{10}^1 T_{14}^1;$$

$$F_2 = T_1^1;\ F_3 = T_2^1.$$

5. Мультиуровневая модель и метод диагностирования ФН

Представлена мультидеревом B, где каждая вершина есть трехмерная таблица активизации функциональных модулей, а дуги, исходящие из нее, есть переходы на нижний уровень детализации при диагностировании, когда замена рассматриваемого уязвимого функционального блока слишком дорога:

$$B = [B_{ij}^{rs}],\ \text{card}B = \sum_{r=1}^{n} \sum_{s=1}^{m_r} \sum_{j=1}^{k_{rs}} B_{ij}^{rs},$$

n – число уровней мультидерева диагностирования; m_r – количество функциональных блоков или компонентов на уровне r; k_{rs} – число компонентов в таблице B^{rs}; $B_{ij}^{rs} = \{0,1\}$ – компонент таблицы активизации, определенный сигналами проверки (непроверки) функционального модуля тест-сегментом $T_{i\text{-}A_i}$ относительно наблюдаемого монитора A_i. Каждая вершина-таблица имеет число исходящих вниз дуг, равное количеству функциональных блоков, диагностируемых (представленных) в таблице активизации. Структура мультидерева, соответствующая модели многоуровневого диагностирования, представлена на рис. 6.

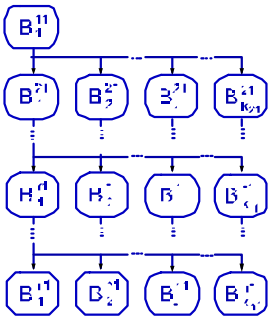


Рис. 6. Фрагмент мультидерева диагностирования

Процесс-модель или метод поиска дефектов по мультидереву диагностирования сводится к созданию движка (рис. 7) для спуска по одной из ветвей дерева на такую глубину, которая удовлетворяет пользователя по степени детализации:

$$B_j^{rs} \oplus A^{rs} = \begin{cases} 0 \to \{B_j^{r+1,s}, R\}; \\ 1 \to \{B_{j+1,}^{rs}, T\}. \end{cases}$$

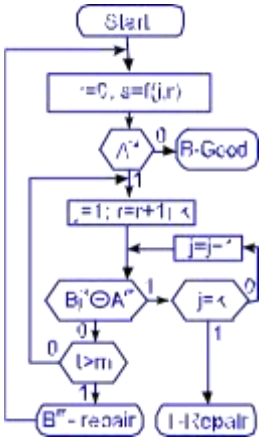


Рис. 7. Движок обхода мультидерева диагностирования

Здесь выполняется векторная xor-операция между столбцами матрицы и вектором экспериментальной проверки A^{rs}, который определяется реакцией функциональности, снятой с мониторов (ассерций) при подаче всех тест-сегментов. Если хотя бы одна координата полученной векторной xor-суммы равна нулю $B_j^{rs} \oplus A^{rs} = 0$, то выполняется одно из действий: переход к матрице активизации нижнего уровня $B_j^{r+1,s}$ или восстановление работоспособности (устранение уязвимости) функционального блока B_j^{rs}. При этом анализируется, что важнее: 1) время – тогда устранение уязвимости рассматриваемого блока; 2) деньги – тогда осуществляется переход вниз, для уточнения места уязвимости, поскольку замена более мелкого блока существенно уменьшает стоимость восстановления неуязвимости. Если хотя бы одна координата полученного вектора xor-суммы равна единице $B_j^{rs} \oplus A^{rs} = 1$, то выполняется переход к анализу следующего столбца матрицы. При нулевых значениях всех координат вектора (ассерционных) мониторов $A^{rs} = 0$ фиксируется неуязвимое состояние всего изделия. Если в рассматриваемой таблице зафиксированы все векторные xor-суммы, не равные нулю $B_j^{rs} \oplus A^{rs} = 1$, то коррекции подлежит тест, построенный для проверки данной функциональности.

Технологическая модель инфраструктуры встроенного тестирования, диагностирования и восстановления неуязвимости (рис. 8) имеет три компонента: 1. Тестирование КС (Unit Under Test (UUT)) с использованием эталонной модели (Model Under Test (MUT)) для формирования вектора экспериментальной проверки m_a, размерность которого соответствует числу тестовых наборов. 2. Поиск дефектов на основе анализа таблицы уязвимостей A. 3. Восстановление неуязвимости КС посредством замены уязвимых блоков на компоненты из Spare Good Primitives.

Процесс-модель встроенного сервисного обслуживания работает в реальном масштабе времени и позволяет поддерживать в работоспособном состоянии, без вмешательства человека, КС, что является целесообразным решением в случае применения технологий,

связанных с дистанционной эксплуатацией изделия. Предложенные процесс-модели анализа ассоциативных таблиц, а также введенные критерии качества логических решений позволяют решать задачи квазиоптимального покрытия, диагностирования уязвимостей программных и (или) аппаратных блоков. Модель векторных вычислений стала основой для разработки специализированной мультипроцессорной архитектуры, ориентированной на поиск, распознавание и принятие решений на основе использования структур ассоциативных таблиц.

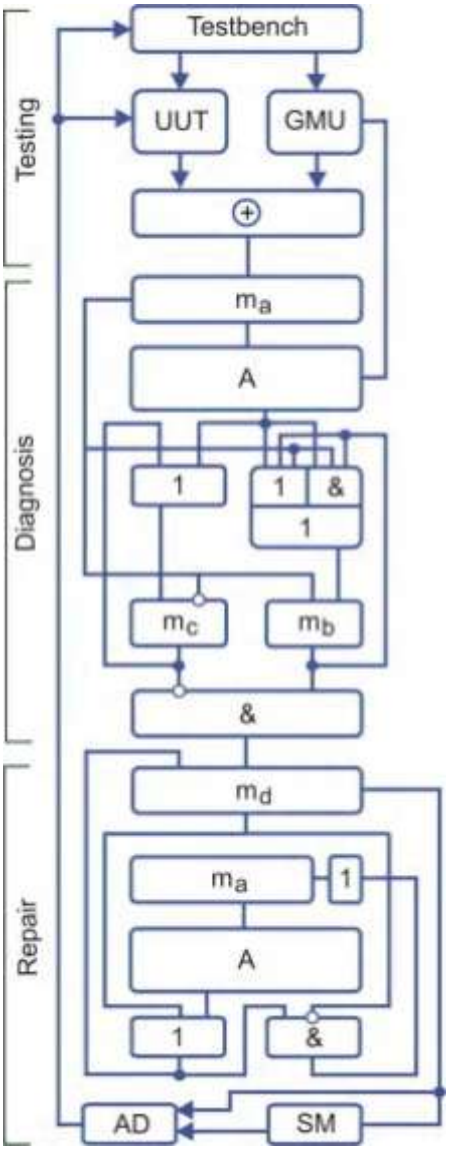


Рис. 8. Модель встроенного тестирования компонентов КС

Таким образом, представленная на рисунке графсхема дает возможность эффективно осуществлять сервисное обслуживание сколь угодно сложной киберсистемы. Преимущества такого движка, инвариантного к уровням иерархии, заключаются в простоте подготовки и представления диагностической информации в виде минимизированной таблицы активизации функциональных блоков или деструктивных компонентов (уязвимостей) на тестовых сегментах.

В последнем случае эффект (уменьшение времени) получен благодаря введению дополнительной инфраструктуры (рис. 9) к функциональности проекта, позволяющей избирательно осуществлять тестирование и диагностирование, а также перепрограммировать отдельные модули в случае фиксации уязвимостей.

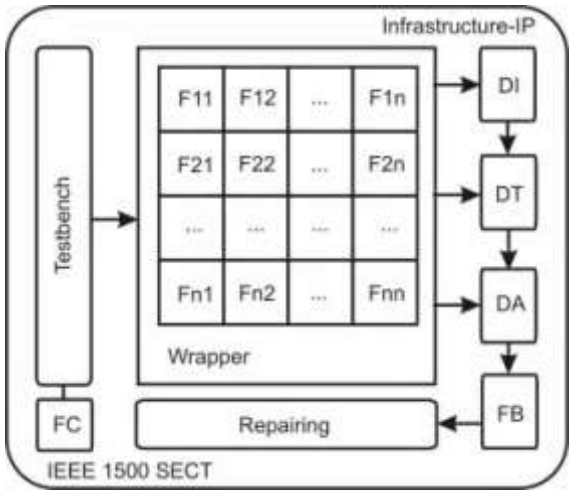


Рис. 9. Инфраструктура тестирования КС

Здесь представлены блоки: Testbench – тесты проверки функциональных блоков, FC – функциональное покрытие теста, F – функциональные блоки КС, DI – диагностическая информация, DT – методы и средства диагностирования, DA – результаты анализа процесса диагностирования, FB – уязвимые функциональные модули, Repairing – восстановление работоспособности функциональных модулей. Сервисное обслуживание отдельной ячейки функциональности осуществляется с помощью ячейки граничного сканирования, представленной на рис. 10.

6. Выводы

Предложена усовершенствованная процесс-модель определения уязвимостей в КС, которая отличается использованием xor-операции, что дает возможность повысить быстродействие диагностирования одиночных или кратных ФН на основе параллельного анализа таблицы ФН с помощью логических векторных операций and, or, xor.

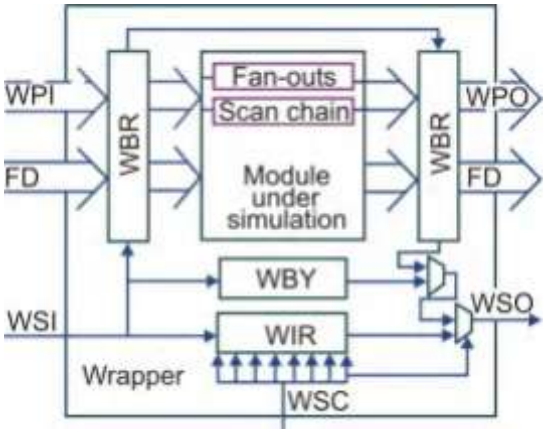


Рис. 10. Ячейка граничного сканирования

Разработаны структурная модель в виде мультидерева и метод (движок) его быстрого обхода, которые отличаются инвариантностью к уровням иерархии КС для диагностирования уязвимостей на тестовых сегментах, что дает возможность эффективно, в реальном масштабе времени, осуществлять сервисное обслуживание сколь угодно сложной киберсистемы.

Список литературы: 1. *Бондаренко М.Ф., Хаханов В.И., Литвинова Е.И.* Структура логического ассоциативного мультипроцессора // Автоматика и телемеханика. 2012. № 10. С. 71-92. **2.** *Hahanov V., Wajeb Gharibi, Litvinova E., Chumachenko S.* Information analysis infrastructure for diagnosis // Information an international interdisciplinary journal. 2011. Japan. Vol.14, № 7. P. 2419-2433. **3.** *Bishop M.* About Penetration Testing // IEEE Security & Privacy. 2007. Vol. 5, Iss. 6. P. 84 – 87. **4.** *Mainka C., Somorovsky J., Schwenk J.* Penetration Testing Tool for Web Services Security // 2012 IEEE Eighth World Congress on Services (SERVICES). 2012. P. 163 – 170. **5.** *Salas P.A.P., Padmanabhan Krishnan, Ross K.J.* Model-based Security Vulnerability Testing // 18th Australian Software Engineering Conference. 2007. P. 284 – 296. **6.** *Bau Jason, Bursztein Elie, Gupta Divij, Mitchell John.* State of the Art: Automated Black-Box Web Application Vulnerability Testing // 2010 IEEE Symposium on Securityand Privacy. 2010. P. 332 – 345. **7.** *Shahriar H., Zulkernine M.* Automatic Testing of Program Security Vulnerabilities // 33rd Annual IEEE International Computer Software and Applications Conference. 2009.Vol. 2. P. 550 – 555. **8.** *Sedaghat S., Adibniya F., Sarram M.-A.* The investigation of vulnerability test in application software // International Conference on the Current Trends in Information Technology (CTIT). 2009. P.1 – 5. **9.** *Wilhelm T.* Professional Penetration Testing. Syngress. 2009. 524 p. **10.** *Shakeel A., Heriyanto T.* BackTrack 4: Assuring Securityby Penetration Testing. Packt Publishing. 2011. 392 p. **11.** *Хаханов В.И., Anders Carlsson, Чумаченко С.В.* Инфраструктура PenTestING и управления уязвимостью // АСУ и приборы автоматики. 2012. Вып. 160. С. 36-54.

Поступила в редколлегию 17.10.2012

Хаханов Владимир Иванович, д-р техн. наук, декан факультета КИУ, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanov@kture.kharkov.ua.

Anders Carlsson, COM School of Computing, Blekinge Institute of Technology (BTH). Adress: Room H454D, Blekinge Institute of Technology, SE-37179 Karlskrona, Sweden.

Чумаченко Светлана Викторовна, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: математическое моделирование, теория рядов, методы дискретной оптимизации. Увлечения: путешествия, любительское фото. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: ri@kture.kharkov.ua.

Бутенко Сергей Александрович, студент группы КИ-09-4 факультета КИУ.Научные интересы:диагностика цифровых систем и сетей. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326.

АННОТАЦИИ

УДК 658.512.011:681.326:519.713

В.И. ХАХАНОВ, ANDERS CARLSSON, С.В. ЧУМАЧЕНКО, БУТЕНКО С.А.

МОДЕЛИ УПРАВЛЕНИЯ УЯЗВИМОСТЬЮ

Предлагается математический аппарат создания инфраструктуры программно-аппаратных телекоммуникационных информационных кибернетических систем (КС), ориентированной на защиту от несанкционированного доступа к сервисам, определенным в спецификации системы, путем проникновения через легальные интерфейсы взаимодействия компонентов, имеющие уязвимости. Инфраструктура защитных сервисов создается вместе с киберсистемой и сопровождает последнюю в течение всего жизненного цикла, обслуживая все последующие модификации КС, и сама постоянно повышает свой интеллект путем пополнения истории и библиотек конструктивных и деструктивных компонентов.

ABSTRACTS

UDC 658.512.011:681.326:519.713

Models of vulnerability management / V.I. Hahanov, Anders Carlsson, S.V. Chumachenko, S.A. Butenko // Management Information System and Devices. 2012. N 161. P.10-24.

The mathematical apparatus of the infrastructure of telecommunications hardware and software information of cybernetic systems (CS), oriented to protect against unauthorized access to the services defined in the system specification, by penetrating through legal interfaces of components that have vulnerabilities has been proposed. Infrastructure protection services created with kibersistemoy and accompanies the last for the entire life cycle, serving all subsequent modifications of the CS, and she is constantly improving our intelligence by enlarging the history of libraries and the constructive and destructive components.

Fig. 10. Ref.: 10 items.

РЕФЕРАТИ

УДК 658.512.011:681.326:519.713

Моделі керування вразливістю / В.І. Хаханов, Anders Carlsson, С.В. Чумаченко, С.А. Бутенко // АСУ та прилади автоматики. 2012. Вип. 161. С. 10-24.

Запропоновано математичний апарат створення інфраструктури програмно-апаратних телекомунікаційних інформаційних кібернетичних систем (КС), орієнтованої на захист від несанкціонованого доступу до сервісів шляхом проникнення через легальні інтерфейси взаємодії компонентів, що володіють вразливостями. Інфраструктура захисних сервісів створюється разом з кіберсистемою і супроводжує останню протягом усього життєвого циклу, обслуговуючи всі наступні модифікації КС, і сама постійно підвищує свій інтелект шляхом поповнення історії й бібліотек конструктивних і деструктивних компонентів.

Іл. 10. Бібліогр.: 11 назв.