Также производим декомпозицию отношения «Блюда» в два отношения «Вид блюда» и «Блюда».

Приложение базы данных было выполнено в среде Access 2007. Определив все взаимосвязи между отношениями, получим окончательную схему данных (рис. 2).

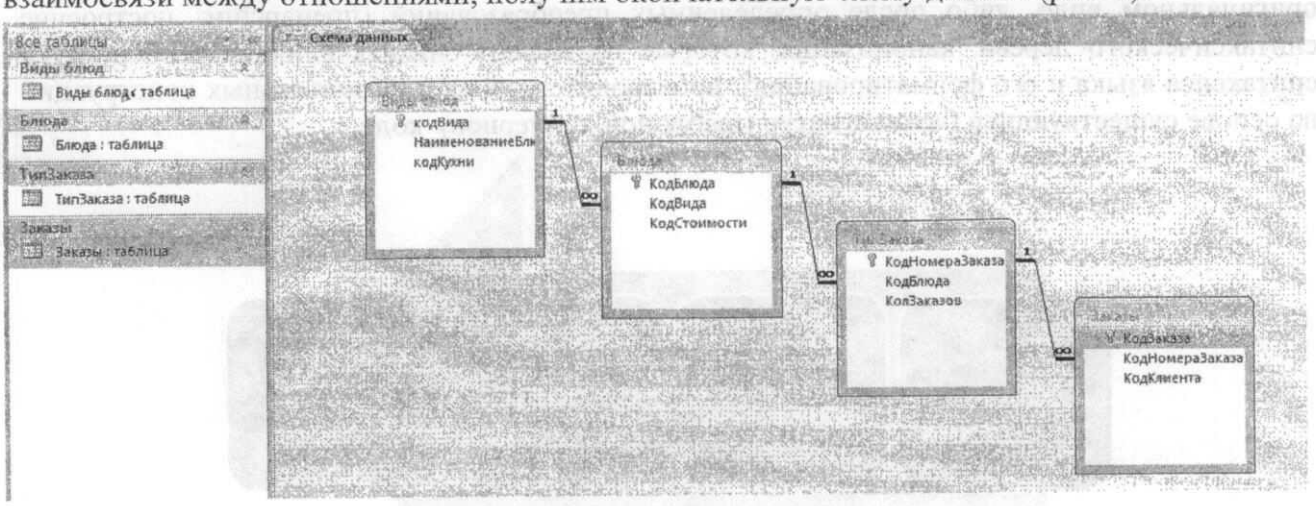

Рисунок 2 – Схема данных БД

Библиография

1. Клименко И.В., Лозовский А.В. Метод формальной нормализации отношений реляционной модели // Науч. мысль Кавказа. Прил. № 5. – 2004. – С. 115–119.
2. Мейер М. Теория реляционных баз данных. – М.: Мир, 1987. – 608 с.
3. Кузнецов С.Д. Основы баз данных. – М.: БИНОМ, 2007. – 484 с.
4. Фаворская М.Н. Распределенные базы данных: учеб. пос. – Красноярск: СибГАУ, 2003. – 116 с.

ВОПРОСЫ ВЫЯВЛЕНИЯ ДЕФЕКТОВ БЕЗОПАСНОСТИ КОДА МЕТОДОМ СТАТИЧЕСКОГО СИГНАТУРНОГО АНАЛИЗА

Марков А.С., Фадин А.А., Рауткин Ю.В.
НПО «Эшелон»
Москва, mail@cnpo.ru

Методы статического анализа исходных текстов программ в отличие от динамического тестирования позволяют избежать проблему проклятья размерности области тестовых данных и добиться большей степени автоматизации проверок на наличие дефектов безопасности, исходя из их конструктивных признаков. В настоящее время известен ряд техник статического синтаксического анализа, позволяющих эффективно определять некорректности кодирования (нефункциональные ошибки) [1]. Однако некоторые дефекты безопасности ПО могут иметь семантически или синтаксически корректную структуру, например, логические бомбы, ошибки конфигурирования, генераторы парольной информации и др. Для поиска подобных ошибок требуется привлечение эксперта, который исследует фрагменты кода повышенного риска, определяемые по некоторому шаблону [2]. В работе будут рассмотрена возможность использования сигнатурного анализа для выявления дефектов безопасности.

Работа была проведена при финансовой поддержке Минобрнауки.

Фактически под понятие сигнатурного анализа подпадает общий подход поиска тех или иных признаков объекта на основе сопоставления его содержимого с образцами из базы уже имеющихся признаков. Перспективным представляется использование данного подхода к

выявлению ошибок в конфигурации приложений, а также потенциально опасных конструкций в коде приложений, в том числе программных закладок и дефектов кода.

Рассмотрим общую схему проведения сигнатурного анализа, представленную на рисунке ниже. В полученном на входе анализатора исходном тексте ПО, либо в оригинальном виде, либо после определенных преобразований (препарсинг, построение синтаксического дерева, канонизация, которые позволяют снизить влияние особенностей синтаксиса языка и его форматирования), производится поиск подозрительных конструкций на основе существующей базы сигнатур (шаблонов, паттернов) кода.

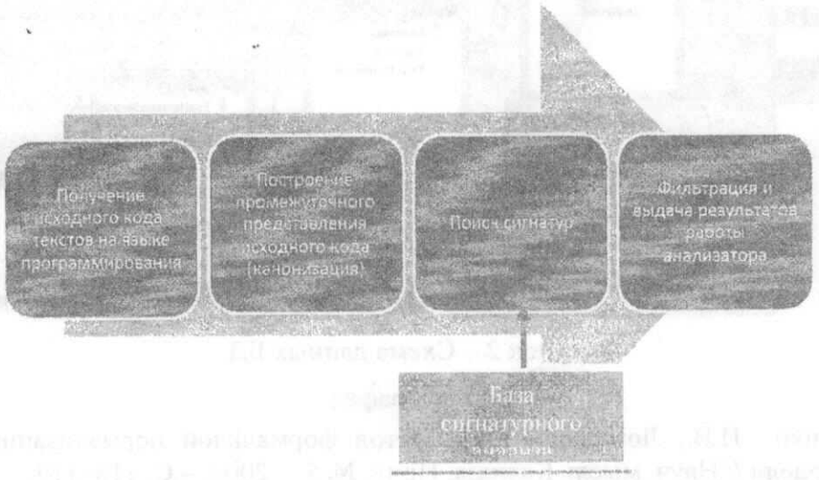

Рисунок – Общая схема выполнения сигнатурного анализа

В процессе работы анализатора выполняются сопоставления участков кода всем находящимся в базе паттернам (сигнатурам) потенциально опасных конструкций.

Результатом работы сигнатурного анализатора как правило является набор найденных подозрительных участков кода с указанием типа вероятного дефекта или программной закладки).

К примеру, сигнатурный анализатора безопасности кода можно представить кортежем:

$$CSA = < SDB, HDB, MLA, MSA, MSY, MLO, MRP >,$$

где *SDB* – база данных сигнатур (паттернов) ПОК, *HDB* – база данных структурной информации о коде (иерархического представления программы), *MLA* – программный модуль лексического анализа, *MSA* – программный модуль синтаксического анализа, *MSY* – программный модуль сигнатурного анализа, *MLO* – программный модуль логического вывода, *MRP* – программный модуль построения отчетов.

Сигнатурный анализатор осуществляет сквозной поиск в исходных текстах программы паттернов ПОК, используя базу сигнатур, которая представляет собой набор следующих кортежей:

$$SDB = <PE, DL>,$$

где *PE* – описание фрагмента ПОК (например, используя регулярные выражения), *DL* – оценка степени их опасности (например, числовая шкала 1–10).

В статье для демонстрации возможности выявления дефектов выбрана система классификации дефектов CWE (Common Weakness Enumeration). На сегодняшний день последняя версия стандарта 2.1 доступна по адресу cwe.mitre.org. Примеры дефектов, их признаки и способы выявления представлены в таблице.

Таблица – Примеры способов выявления дефектов безопасности

Элемент CWE верхнего уровня	Элемент CWE нижнего уровня	Признаки наличия	Способ обнаружения
Проблемы поведения (Behavioral Problems) (438)	Изменение поведения в новой версии окружения (Behavioral Change in New Version or Environment) (438)	<совпадение вызова со списком проблемных для зависимости, используемой в проекте>	Определить версию среды функционирования Сравнить вызов со списком вредоносных функций
	Неверный контроль частоты взаимодействия (Improper Control of Interaction Frequency) (799)	<нет таймаута или запрета попыток> в интерактивной функции (взаимодействующей с пользователем или внешней стороной).	Определить интерактивность функционального объекта (по наличию API заданного типа) Определить внутри интерактивного объекта наличие API по обработке задержки
Ошибки каналов и путей (Channel and path errors) (417)	Неверная защита альтернативного пути Improper Protection of Alternate Path - (424)	<отсутствие проверки при показе закрытой страницы>	Определить вызов функционала, отображающего содержимое страницы Удостовериться в наличии функционала контроля сессии доступа пользователя
	Скрытый канал памяти (Covert storage channel) (515)	<обращение к нестандартным информационным объектам (файл подкачки, реестр, прямой доступ к диску, операции с кучей)>	Поиск вызовов функционала из списка объектов потенциально скрытых каналов
	Скрытый канал времени (Covert timing channel) (385)	<обращение к нестандартным информационным объектам (бесконечные циклы, работа с очередью сообщений, таймером, высокопроизводительным таймером)>	Поиск вызовов функционала из списка объектов потенциальных скрытых каналов
Обработка данных (Data Handling) (19)		<поиск включения введенных пользователем данных при формировании строк>	Поиск вызовов получения пользовательских данных Поиск вызовов потенциально опасных функций с использованием пользовательских данных
Обработка ошибок (Error Handling) (388)		наличие в блоке catch-операции вывода в журнал	Поиск обработчиков исключений и определение внутри вызовов вывода в журнал
Ошибки обработчика (Handler Errors) (429)		<выброс исключения, у которого нет обработчика>	Создание списка всех выбросов исключений Создание списка всех обработчиков Сравнение двух списков
Злоупотребление API (API Abuse) (227)		<использование внешних компонент (библиотек) с некорректными версиями>	Составление списка всех внешних зависимостей с версиями Поиск потенциально опасных вызов из этих библиотек
Индикатор плохого качества кода (Indicator of Poor Code Quality) (398)	Присваивание вместо (Assigning instead of Comparing) (481)	<наличие «=» вместо «==» в блоке if, где сравниваются лишь переменные>	Проверка заданного списка регулярных выражений для содержимого заданных типов блоков
Ошибки очистки и инициализации(Initialization and Cleanup Errors) (452)	Неверная инициализация (Improper Initialization) (665)	Отсутствие инициализации значений объектов (например, строк) перед их использованием	Поиск объявлений объектов, которые требуют инициализации перед своим использованием (например, статический массив строки) Поиск первого использования объекта в вызовах и операциях (например, правая часть в присваивании; функции, использующие их в качестве входных значений), при отсутствии их

Элемент CWE верхнего уровня	Элемент CWE нижнего уровня	Признаки наличия	Способ обнаружения
			инициализации (левая часть в присваивании, функции использующие их в виде выходных значений)
Недостаточная инкапсуляция (Insufficient Encapsulation) (485)	Остаточный отладочный код (Leftover Debug Code) (489)	<наличие отладочного кода>	Поиск присутствия вызовов функций из списка отладочных Поиск подозрительных правил в названия отладочных функций (ключевые слова по debug)
Проблемы указателей (Pointer Issues) (465)	Разыменование нулевых указателей (NULL Pointer Dereference) (476)	<вызов операций по освобождению памяти для указателей, которые равны NULL>	Поиск вызовов функций, возвращающих указатель (на которую может повлиять пользователь), при отсутствии проверки ее значения следом за этим
Механизмы безопасности (Security Features) (254)	Использование жестко прописанных паролей (Use of hard-coded password) (259)	<наличие паролей и других данных авторизации непосредственно в коде приложения>	Поиск вызовов функций с параметрами авторизации, использующих аргументы со строковыми константами Поиск функций, требующих авторизации вместе с переменными, которым были присвоены строковые константы
Время и состояние (Time and State) (361)	Внешнее влияние на сферу определения (External Influence of Sphere Definition) (673)	<проверка наличия функций, использующих значения переменных окружения>	Поиск вызовов функций, читающих содержимое переменных окружения
Ошибки интерфейса пользователя (User Interface Errors) (445)	Нереализованная или неподдержимая функция в GUI (Unimplemented of unsupported feature in UI) (447)	<наличие скрытых элементов GUI, а также некорректностей в обработчиках событий>	Поиск наличия disabled=true hidden=true и других подобных свойств в файле GUI> Поиск отсутствия кода в функции-обработчике (Qt, Builder)

Попробуем обобщить перечисленные выше подходы к выявлению дефектов безопасности.

Для выявления потенциально опасных конструкций необходимо решать задачи следующего рода:

1) определение имени информационного объекта (как правило, переменной), которому происходит передача значения (как правило, присваивания) заданного потенциально опасного вызова (вызов функции, обращение к ассоциативному массиву значений);

2) определение внешнего вызова (прежде всего функции) из списка потенциально опасных для данного языка, платформы или при условии подключения/не подключения заданного модуля/библиотеки и другого подобного контекста;

3) определение внешнего вызова (прежде всего функции) из списка потенциально опасных, при условии использования в нем информационного объекта с известным именем;

4) определение внешнего вызова (прежде всего функции) из списка потенциально опасных, при условии использования в нем аргумента (строкового параметра или информационного объекта), содержащего заданное строковое регулярное выражение.

5) определение внешнего вызова из списка потенциально опасных, который использует тот же информационных объект, что уже был использован потенциально опасным вызовом;

6) поиск всех определений блочных конструкций языка (например, перехват исключений или определение класса) использующих объект заданного типа (класс-родитель, аргумент перехвата исключения);

7) определение для каждой блочной конструкции (например, класса) наличия заданного вложенного объекта (метода, другого класса) при условии выполнения условия (5).

Одним из решающих преимуществ сигнатурного анализа является его быстрота работы и легкость конструирования сигнатур (по сути, их можно оперативно дорабатывать и

настраивать для решения любой узкой задачи распознавания наборов). В дальнейшем на основе распознанных конструкций можно построить любую, самую сложную логику работы анализатора.

Чтобы решить задачи, описанные выше и построить базис для подобной логики, для каждого из поддерживаемых анализатором языков программирования требуется решить следующие задачи:

1) выполнить разбиение исходных текстов на строки операций с удалением комментариев, избыточных символов пробелов, переносов строк и другой незначимой для данного вида анализа информации;

2) выделить в строке операции вызова функционального объекта и извлечь литерала его названия (для повышения точности операции имеет смысл добавить небольшой список ключевых слов-исключений, представляющий зарезервированные слова этого языка, например: for, while, if);

3) выделить в строке операции передачу значения (в первую очередь, присваивание);

4) выделить литералы имен объектов, предположительно получающих значения, и литералов имен объектов, служащих источником данных значений.

Данный вид анализа в литературе по созданию компиляторов получил название «анализ потоков данных» (data-flow analysis). Рассмотрим схему работу анализатора, использующего сигнатурный анализ и анализ потоков данных для выявления дефектов программного кода.

Предложим следующее представление кода. По сути, каждая значимая строка конструкций исходного кода (за исключением специфических макросов или других директив компилятора) может быть представлена в виде следующего кортежа:

$$CD=<C, I, O>,$$

где C – это вызываемый элемент, I – это перечень элементов, чьё значение считывается, O – это перечень элементов, чье значение изменяется на основе считанных элементов.

Каждое поле типа «элемент» может быть одного из трех типов:

– объект (в данном элементе объявляется или создается объект, использующий память);

– ссылка (в данном выражении находится лишь имя или косвенная ссылка на имя существующего объекта);

– литерал (в данном выражении находится непосредственное значение элемента – строка, число и т.п.).

Исследование существующих баз дефектов кода ПО и изучение реальных примеров кода показали, что достаточно широкий спектр задач, связанных с выявлением потенциально опасных конструкций, можно решить с помощью сигнатурного анализа с внутрипроцедурным анализом потока данных (intro-procedural data-flow).

Среди достоинств сигнатурного метода статического анализа можно назвать относительную простоту реализации, очень высокую скорость работы, а также легкость портирования сигнатур на различные платформы и языки программирования.

Недостатком сигнатурного подхода является необходимость учитывать различные «вариации» конструкций и блоков кода, которые допускает синтаксис языка программирования и логика выполнения программы.

В связи с этим наиболее эффективной роль сигнатурного анализа видится в исследовании зависимостей модулей программ и внешних компонент, поиске вызовов функциональных объектов, а также проверке содержимого информационных объектов программы. Для класса ошибок, связанных с синтаксисом программы и достижимостью кода, его следует применять совместно с другими методами.

Библиография

1. Методы оценки несоответствия средств защиты информации / А.С.Марков, В.Л.Цирлов, А.В.Барабанов; Под ред. А.С.Маркова. М.: Радио и связь, 2012. 192 с.

2. Марков А.С., Найханова И.В., Цирлов В.Л. Сертификация программных систем по требованиям безопасности информации // Материалы X Всероссийской научно-технической конференции «Теоретические и прикладные вопросы современных информационных технологий» ТиПВСИТ'2009. Улан-Удэ: Изд-во ВСГТУ, 2009. С.244-249.

ПРИМЕР РАСЧЕТА ВЕЛИЧИНЫ РИСКА С ПРИМЕНЕНИЕМ АЛГОРИТМА НЕЧЕТКОГО ВЫВОДА МАМДАНИ

Найханова И.В.

Московский государственный технический университет им. Н.Э.Баумана
Москва, irishka_ever@mail.ru

Введение. Проблемы обеспечения безопасности различных систем в современном мире усиливается отсутствием единой развитой методической базы, позволяющей проводить адекватную оценку угроз ресурсам, а также степень защищенности данных систем. Пока наиболее популярным направлением является подход на основе оценки и управления рисками [1].

Известны [2,3,4] различные подходы к оценке и управлению рисками, такие как, статистический метод, подход на основе экспертных оценок и оценки субъективной вероятности, вероятностно-статистический подход, теоретико-вероятностный метод, метод расчета и управления экономическими рисками с использованием теории полезности. Принципиальная сложность проведения такого анализа рисков в области информационной безопасности для современных систем заключается в том, что для достижения адекватных оценок необходимо учитывать достаточно большое количество факторов, которые находятся в сложной зависимости друг от друга. Причем, зачастую достаточно трудно оценить степень достоверности полученного результата, поскольку при проведении анализа невозможно учесть все факторы. Ключевым недостатком большинства разработанных методов управления рисками является наличие у них одного из следующих конфликтующих по своей сути свойств [5,6]:

1) ярко выраженный качественный и обладающий значительным субъективизмом подход к анализу рисков. Наличие этого свойства обуславливает слабую формализуемость, и, следовательно, невозможность создать автоматизированные инструментальные средства анализа и управления рисками на основе разработанной методики;

2) жестко формализованный подход к анализу рисков. Это означает, что обычно методика недостаточно универсальна, то есть неполна. Кроме того, сложные формальные методы зачастую трудно применять на практике и, следовательно, эффект от их использования незначительный. При этом формальные методы достаточно сложно применить при отсутствии математических моделей процессов, происходящих в информационных системах, а также при недостаточном объеме статистических данных.

В этом противоречии известных методов и стандартов, чрезвычайно актуальным является построение методического обеспечения, которое бы отвечало следующим требованиям:

- достаточная формализуемость для реализации в виде инструментальных средств;
- необходимая простота для инженерного применения методик;
- адекватность и универсальность управления рисками, то есть применимость методики к большинству систем;
- учет динамики развития системы;

– комплексность, позволяющая охватить все существенные аспекты функционирования системы.

В данной работе попробуем показать, как можно использовать математический аппарат нечетких множеств и нечеткой логики для формализации задачи оценки рисков.

Основные понятия задачи. С точки зрения информационной безопасности к свойствам ресурса можно отнести перечень угроз, воздействующих на него, и критичность ресурса. Используем определения, принятые в работе [7]. Свойством угрозы является перечень уязвимостей, при помощи которых может быть реализована угроза.

Уязвимость – это слабое место в информационной системе, которое может привести к нарушению безопасности путем реализации некоторой угрозы. Свойствами уязвимости являются: вероятность (простота) реализации угрозы через данную уязвимость и критичность реализации угрозы через данную уязвимость.

Уязвимости, как и угрозы, могут быть оценены по трехуровневой качественной шкале. Значение уровня уязвимости показывает, насколько вероятно успешное осуществление угрозы с использованием данной уязвимости в случае, если эта угроза будет реализовываться. Соответствующие качественные уровни уязвимости могут быть определены, например, следующим образом:

В – вероятно. Уязвимость легко использовать, и существует слабая защита или защита вообще отсутствует. Вероятность успешной реализации угрозы лежит в интервале $0.9 \div 1$;

С – возможно. Уязвимость может быть использована, но существует определенная защита. Вероятность успешной реализации угрозы ~ 0.5;

Н – маловероятно. Уязвимость сложно использовать, и существует хорошая защита. Вероятность успешной реализации угрозы лежит в интервале $0 \div 0.1$.

Критичность ресурса – степень значимости ресурса для информационной системы, т.е., как сильно реализация угроз информационной безопасности на ресурс повлияет на работу информационной системы. Как правило, задается в уровнях (количество уровней может быть в диапазоне от 2 до 100) или в денежных единицах. В зависимости от выбранного режима работы может состоять из критичности ресурса по конфиденциальности, целостности и доступности.

В работе [8] предложена матрица (табл.1), с помощью которой можно определить уровень угрозы и рассчитать величину риска.

Таблица 1 – Определение величины риска

Критичность ресурса	Уровень угрозы								
	Низкий			Средний			Высокий		
	Уровень уязвимости								
	Н	С	В	Н	С	В	Н	С	В
0	0	1	2	1	2	3	2	3	4
1	1	2	3	2	3	4	3	4	5
2	2	3	4	3	4	5	4	5	6
3	3	4	5	4	5	6	5	6	7
4	4	5	6	5	6	7	6	7	8

Таким образом, имеем показатель критичности ресурса и уровень уязвимости ресурса. Необходимо вычислить уровень угрозы и величину риска.

Нечеткий логический вывод при расчете величины риска. В работе предлагается использовать нечеткий логический вывод для подсчета величины риска. Выполним моделирование нечеткого логического вывода в среде MatLab. Данную задачу решим в два этапа. Вначале определим уровень угрозы, а затем величину риска.

Введем входные лингвистические переменные:

1) *KR* – критичность ресурса;
2) *LI* – уровень входного показателя.

Выходная лингвистическая переменная *LO* – уровень выходного показателя.

Для фаззификации лингвистических переменных *KR*, *LI* (перевода четких значений в нечеткие – элементы терм-множества {*low*, *middle*, *high*}) применим функцию принадлежности *gauss2mf*, заданную формулами (1) и (2), значения параметров приведены в таблице 2.

$$\text{если } c_1 < c_2, \quad \text{то} \quad \mu(x) = \begin{cases} \exp\left((x-c_1)^2/(-2a_1^2)\right), & x < c_1 \\ 1, & c_1 \le x \le c_2 \\ \exp\left((x-c_2)^2/(-2a_2^2)\right), & x > c_2 \end{cases}; \tag{1}$$

$$\text{если } c_1 > c_2, \quad \text{то} \quad \mu(x) = \begin{cases} \exp\left((x-c_1)^2/(-2a_1^2)\right), & x < c_2 \\ \exp\left((x-c_1)^2/(-2a_1^2)\right) \cdot \exp\left((x-c_2)^2/(-2a_2^2)\right), & c_2 \le x \le c_1 \\ \exp\left((x-c_2)^2/(-2a_2^2)\right), & x > c_1 \end{cases}. \tag{2}$$

Если c_1<c_2, то параметры функции принадлежности интерпретируются следующим образом:

$c_1(c_2)$ – минимальное (максимальное) значение ядра нечеткого множества;

$a_1(a_2)$ – коэффициент концентрации левой (правой) части функции принадлежности.

Интервалы нечетких переменных располагаются на универсальной шкале *U*=[0,4] для *KR*, *U*=[0,8] для LI, *U*=[0,1] для *LO*.

Таблица 2 – Примеры значений параметров *gauss2mf*

Терм-множество	Вектор *Param*=[a_1, c_1, a_2, c_2)] переменной *KR*	Вектор *Param* входных лингвистической переменной *LI*	Вектор *Param* выходной лингвистической переменной *LO*
Low	[0.54 -0.16 0.54 0.16]	[1.09 -0.32 1.09 0.32]	[0.14 -0.04 0.14 0.04]
Middle	[0.54 1.84 0.54 2.16]	[1.09 3.68 1.09 4.32]	[0.14 0.46 0.14 0.54]
High	[0.54 3.84 0.54 4.16]	[1.09 7.68 1.09 8.32]	[0.14 0.96 0.14 1.04]

На рисунке 1 приведен график функции принадлежности для трех термов *Low*, *Middle*, *High* на *U*=[0,10]. На графике функции для терма *Middle* отмечены точки, заданные параметрами a_1, c_1, a_2, c_2. Параметры c_1 и c_2 задают ширину "раскрыва" функции и равны значению 0.54, параметры a_1, a_2 определяют ядро нечеткого множества для *Middle*, оно лежит в интервале [3.68, 4.32]. Аналогично задаются ядра нечетких множеств *Low* и *High*.

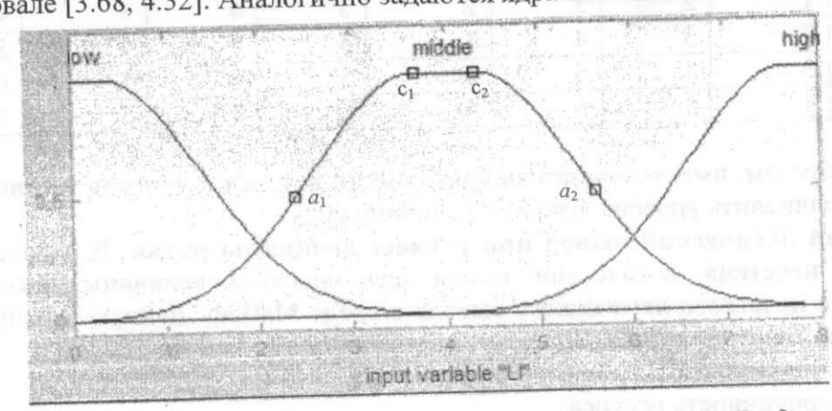

Рисунок 1 – График функции принадлежности gauss2mf

Значения входных лингвистических *KR*, *LI* вводятся в нечеткий регулятор, и осуществляется их фаззификация. Значение, рассчитанное по функции принадлежности, означает степень истинности быть некоторым термом, принадлежащим терм-множеству нечетких переменных, т.е. они примут значения *Middle* или *Low*, или *High*.

Для примера рассмотрим фаззификацию переменной *LI* значение, которой равно 7.

Для x=7 рассчитываем три варианта:

1) при a_1=1.09 и c_2=0,32 (так как $x>c_2$) получаем $\mu(x)\approx 0$;

2) при a_1=1.09 и c_2=4,32 (так как $x>c_2$) получаем $\mu(x)$=0.04;

3) при a_1=1.09 и c_1=7,68 (так как $x<c_1$), получаем $\mu(x)$=0,82.

Нулевые значения не принимаем во внимание. Откладываем на графике по вертикали (ось ординат) '0.82', а по горизонтали (ось абсцисс) – значение '7'. Точка пересечения принадлежит графику интервала нечетких значений «High». Аналогичным образом переводится значение '3.5' переменной *KR*.

Мягкие вычисления выполним с помощью системы нечетких правил, приведенной на рисунке 2.

Данная база является полной. Из нее удалены некорректные правила. Нечеткий регулятор просматривает последовательно все правила, так как каждое из них имеет единичный приоритет. В конце каждого правила указан его приоритет – (1). Рассмотрим правило N12:

If (KR is high) and (LI is high) then (LO is high).

Каждое правило имеет два подусловия, следующие в скобках после ключевых слов "If" и "and". Определяется степень истинности каждого подусловия: b_1 = (KR is high) и b_2 = (LI is high). Количественное значение степени истинности подусловий правила осуществляется по формулам (1) и (2), аргументом функции является b_i: $\mu_i'(b) = \min\{b_1, b_2\}$.

1. If (KR is low) and (LI is low) then (LO is low) (1)
2. If (KR is low) and (LI is middle) then (LO is low) (1)
3. If (KR is low) and (LI is high) then (LO is middle) (1)
4. If (KR is middle) and (LI is low) then (LO is low) (1)
5. If (KR is middle) and (LI is middle) then (LO is middle) (1)
6. If (KR is middle) and (LI is high) then (LO is middle) (1)
7. If (KR is middle) and (LI is high) then (LO is high) (1)
8. If (KR is high) and (LI is low) then (LO is middle) (1)
9. If (KR is high) and (LI is middle) then (LO is middle) (1)
10. If (KR is high) and (LI is low) then (LO is high) (1)
11. If (KR is high) and (LI is middle) then (LO is high) (1)
12. If (KR is high) and (LI is high) then (LO is high) (1)

Рисунок 2 – Система нечетких продукционных правил

Так как выходная переменная *LO* одна, и каждое правило состоит из одного подзаключения, например (LO is high), то процедуры активации и аккумуляции не выполняются.

На рисунке 3 представлена графическая интерпретация алгоритма нечеткого вывода Мамдани для рассматриваемого примера.

Деффаззификация осуществляется центроидным методом (методом центра тяжести).

Как видно из рисунка, при *KR*=3.5 и *LI*=7, *LO*=0.84 (high), т.е. уровень угрозы – высокий.

Для вычисления величины риска роль входной лингвистической переменной *LI* играет выходная переменная LO (уровень угрозы). На втором этапе выходная лингвистическая переменная *LO* является величиной риска. Нечеткий логический вывод осуществляется на той же базе правил. Фрагмент вычислений показан в таблице 3.

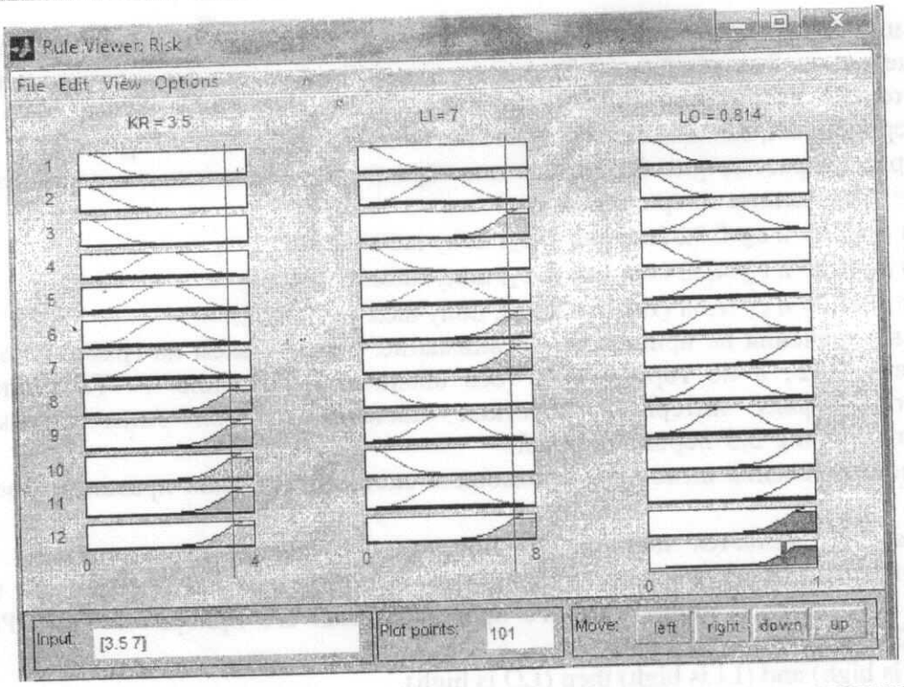

Рисунок 3 – Графическая интерпретация результатов нечеткого логического вывода

Таблица 3 – Фрагмент вычислений величины риска LO

N	Входные лингвистические переменные		Выходная лингвистическая переменная, LO U=[0..1]
	KR, U=[0..4]	LI, U=[0..1]	
1	2	0.814	0.608
2	3	0.3	0.547
3	3.8	0.9	0.838
	...	...	...

Пакет Fuzzy Logic Toolbox позволяет получить трехмерный график зависимости величины риска информационной безопасности от критичности ресурса и уровня угрозы (рис. 4).

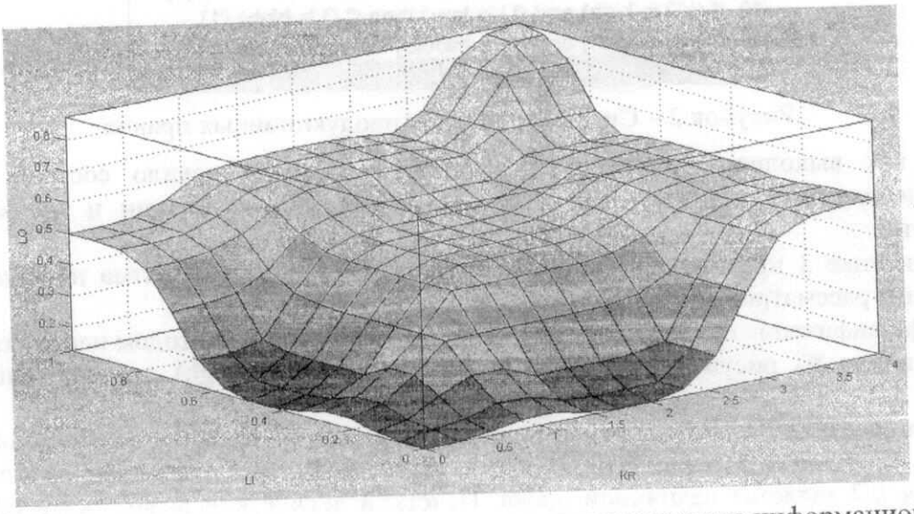

Рисунок 4 – Трехмерный график зависимости величины риска информационной безопасности от критичности ресурса и уровня угрозы

Заключение. Описанный в работе пример показывает, что нечеткий логический вывод можно применять для расчета многих показателей информационной безопасности, так, например, для расчета критичности ресурса, вероятности реализации угрозы и так далее. Многие показатели информационной безопасности определяются экспертным методом, а нечеткий логический вывод осуществляется на базе нечетких продукционных правил, которые должны отражать рассуждения экспертов.

Библиография

1. Петренко С.А. Управление информационными рисками. Экономически оправданная безопасность. – М.: Компания АйТи; ДМК Пресс, 2004. – 384 с.
2. Вишняков Я.Д., Радаев Н.Н. Общая теория рисков: учеб. пособие для студ. высш. учеб. заведений. – М.: Издательский центр «Академия», 2007. – 368 с.
3. Гранатуров В.М. Экономический риск: сущность, методы измерения, пути снижения : Учеб. пособие. – 2-е изд., перераб. и доп. – М.: Дело и Сервис, 2002. – 159 с.
4. Марков А.С. Управление рисками — нормативный вакуум информационной безопасности / А.С.Марков и В.Л.Цирлов // Открытые системы. СУБД. 2007. №8. С. 63-67.
5. Управление рисками / URL: http://ib-061.ucoz.ru/_ld/2/259_4Xe.docx.
6. Остапенко О.А., Карпеев Д.О., Асеев В.Н., Морев Д.Е., Щербаков В.Б. Риски систем: оценка и управление / под. ред. Ю. Н. Лаврухина, А. Г. Остапенко. – Воронеж: Междунар. ин-т компьют. технологий, 2007. – 261 с.
7. Алгоритм: модель анализа угроз и уязвимостей / URL: http://dsec.ru/download/threats_vuln.pdf.
8. Астахов А.М. Искусство управления информационными рисками. – М: ДМК Пресс, 2010. – 312 с.

ПРИМЕНЕНИЕ БИОИНСПИРИРОВАННЫХ МЕТОДОВ ОПТИМИЗАЦИИ ДЛЯ РЕАЛИЗАЦИИ КРИПТОАНАЛИЗА КЛАССИЧЕСКИХ И БЛОЧНЫХ КРИПТОСИСТЕМ

Чернышев Ю.О., Сергеев А.С., Дубров Е.О.

Донской государственный технический университет
Ростов-на-Дону, sergeev00765@mail.ru

Надежность защиты информации криптографическими методами определяется стойкостью к криптоанализу используемой системы шифрования. Как отмечено в [1], основными задачами в криптографии являются разработка новых способов шифрования, сложных для вскрытия, и нахождение новых способов вскрытия существующих шифров (задача крптоанализа). Задача вскрытия шифра состоит в нахождении секретного ключа среди множества всех возможных ключей, т.е. является задачей поиска.

Следует заметить, что в настоящее время в науке и технике находят широкое применение алгоритмы, основанные на природных системах. Это новое научное направление под названием «природные вычисления» объединяет математические методы, в которых заложен принцип природных механизмов принятия решений. К ним относятся методы моделирования отжига, генетические, эволюционные методы, алгоритмы роевого интеллекта и др. [2]. В настоящее время известны применения алгоритмов роевого интеллекта для оптимизации широкого круга задач, в том числе задач криптоанализа. В [3-6] рассматривались методы организации криптографических атак на традиционные симметричные криптосистемы, использующие шифры перестановок и замены, а также на блочные криптосистемы с использованием методов эволюционной оптимизации и генетического поиска, а в [7] – также с использованием квантового поиска.

Отметим, что квантовый поиск анализирует неструктурированные проблемы, которые в общем случае формулируются следующим образом [8,9]. Задана функция f(x), аргументы x

– целые числа, причем f(x) принимает значение 0 во всех случаях, кроме x=t. Необходимо найти значение t, используя наименьшее число запросов к f(x). В [9] приводится описание модифицированного алгоритма квантового поиска и его применение для решения NP-полных проблем нахождения инвариантов графа (определение клик, решение задачи раскраски), для чего производится анализ структуры графа, чтобы «выращивать» полные решения, рекурсивно расширяя последовательные частичные решения. В [7] приводится описание применения данного подхода для реализации процесса криптоанализа шифров перестановок, а также демонстрационный пример реализации квантового поиска для получения частичных решений, их расширения до получения исходного текста при исключении тупиковых решений. Опишем алгоритм квантового поиска, приведенный в [7].

1. Ввод исходного шифртекста.
2. Разделение шифртекста на фрагменты.
3. Для каждого фрагмента построение множества частичных решений.
4. Выполнение суперпозиции частичных решений.
5. Если полученные после суперпозиции частичные решения содержат количество букв k<n, где n – число букв шифртекста, то проверить, являются ли полученные частичные решения допустимыми. Если нет, исключить недопустимые и перейти к 4.
6. Если полученные после суперпозиции частичные решения содержат количество букв k>n , то произвести лексикографический перебор и выбрать те, которые после отсечения лишних букв дают ключевой текст.

Следует заметить, что, как отмечено в [10,11], современная криптография включает в себя 4 основных типа криптосистем: симметричные криптосистемы, криптосистемы с открытым ключом, системы электронной подписи, управление ключами. Среди классических методов с симметричной функцией шифрования отметим описанные в [10,12] шифры перестановки, простой и сложной замены и их комбинации. В ряде классических шифров перестановки применяются шифрующие таблицы, которые в общем случае задают правила перестановки букв в тексте. В [12] описано применение шифрующих таблиц для реализации одинарной и двойной перестановок, а также применение магических квадратов для шифрования перестановками. Разработка генетического алгоритма (ГА) криптоанализа требует решения следующих задач [1]:

- определение способа представления допустимого решения в виде хромосомы в ГА;
- выбор основных операторов и способов их реализации в ГА;
- исследование скорости сходимости и оценивание процента оптимальных решений в ГА.

Вначале рассмотрим методы криптоанализа симметричных шифров перестановок, описанных в [10,12]. Как и в методе криптоанализа блочного перестановочного шифра, описанного в [1], основной задачей является представление особи. Поскольку, как отмечено в [12], при использовании шифрующих таблиц ключом является перестановка $(p_1, p_2, \ldots p_n)$, то хромосома в ГА должна также задавать перестановку. Основной вопрос при этом – как осуществить представление отдельных генов особи. В простейшем случае кодирование осуществляется путем присвоения отдельным генам соответствующих элементов ключа, т.е. i –м геном хромосомы P считать элемент p_i. Как отмечено в [1], основным недостатком такого подхода является то, что гены получаются зависимыми друг от друга, что приводит к возможности получения нелегальных решений. Тем не менее такое определение генов интуитивно понятно и не требует дополнительных затрат на их формирование (вычисление).

Альтернативным подходом в данном случае является использование некоторого промежуточного представления, при котором набор генов задает некоторое правило или объект, из которого формируется ключ. При этом основной задачей является нахождение промежуточного решения, задаваемого в виде битовой строки для применения стандартных генетических операторов [1].

При реализации алгоритма криптоанализа использовался первый подход, т.е. в качестве генов особи рассматриваются элементы ключа. Для предотвращения получения нелегальных