XXXI Неделя науки СПбГПУ. Материалы межвузовской научной конференции. Ч. VI: С.166-167, 2003.

УДК 50.41.00

М.Ю. Маркевич, О.А.Сумкина, А.С. Зайцева (3 курс, каф. ИБКС),
В.В. Платонов, к.т.н., проф., А.С. Монин, ст. преп.

МЕТОДЫ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ ИСХОДНЫХ ТЕКСТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

При разработке программного обеспечения очень важен этап тестирования готового программного продукта (ПП) на предмет соответствия его характеристик заявленному техническому описанию. Одним из основных этапов такого тестирования является исследование исходного текста данного ПП. Особое внимание при этом уделяется поиску потенциальных уязвимостей, приводящих к сбоям в выполнении алгоритма программы, и обнаружению функциональных модулей, реализующих недокументированные возможности данного ПП. Такие уязвимости и программные «закладки» успешно используются для нарушения целостности информационной системы, в которую входит данный ПП, конфиденциальности хранящейся в ней информации и создания условий, приводящих к отказу в обслуживании.

Для автоматизации и упрощения процесса тестирования, что имеет особое значение для сложных ПП с большим объемом исходного текста, существует ряд вспомогательных программных средств (например, утилиты RATS или ITS-4), однако их универсальность и функциональность оставляют желать лучшего. Достаточно отметить, что ни одно из них не предлагает хотя бы базовых возможностей по поиску указанных выше типов уязвимостей, не говоря уже о реализации пользовательского интерфейса и представлении результатов тестирования в наглядном виде.

Учитывая актуальность проблемы и указанные выше недостатки существующих вспомогательных средств, авторы доклада разработали и реализовали в комплексе прикладных программ механизм тестирования исходных текстов ПП.

При синтаксическом и семантическом разборе исходного текста программы наиболее удобным на наш взгляд объектом анализа является функция (подпрограмма, процедура). Это справедливо как для «классического» процедурного, так и для объектно-ориентированного программирования. Правда, в последнем случае задача определения взаимосвязей между функциональными модулями усложняется и требует, в частности, дополнительного анализа объектов классов и механизма наследования.

Исследование собственно объекта (функции) проводится с различных позиций – логики, семантики, статистики, а также при помощи поиска по сигнатуре.

При логическом анализе вычленяются конструкции языка, соответствующие условным операторам, операторам цикла, мультиветвления и безусловного перехода. Это позволяет в дальнейшем восстановить алгоритм выполнения функции и построить блок-схему.

Основной задачей семантического анализа в текущей реализации программного комплекса является поиск явных рекурсивных функций (неявная рекурсия легко находится при помощи логического анализа и построенной на его основе блок-схемы), счетчиков циклов и определения с некоторой долей погрешности смысловой нагрузки тех или иных операторов.

Статистический анализ – вспомогательный, он дает лишь количественные характеристики функционального модуля. Например, функции с большим уровнем вложенности операторов (возможно, даже недопустимым по стандарту ANSI C) считаются потенциально небезопасными и требующими более тщательной проверки. То же в полной мере относится к множеству условий логического оператора – превышение некоторого усредненного количества условий может сигнализировать о необходимости дополнительного исследования данного оператора.

Поиск по сигнатуре – это не что иное, как простое сравнение операторов функционального модуля с базой данных потенциальных уязвимостей.

Разработанный авторами доклада комплекс программ обладает следующими основными возможностями:

- Распознавание исходного текста ПП, написанного на языке программирования ANSI C или C++ и приведение его к формализованному виду.
- Сканирование исходного текста ПП на предмет обнаружения уязвимостей, связанных с использованием разработчиками небезопасных (например, с точки зрения переполнения стека памяти) стандартных функций языка программирования согласно базе данных потенциально небезопасных функций.
- Редактирование и создание новых (пользовательских) баз данных уязвимостей при помощи специального редактора баз данных.
- Статистический анализ исходного текста ПП (исследование уровня вложенности операторов, глубины рекурсии, сложности логических конструкций условных операторов).
- Анализ взаимодействия между функциональными блоками программы (определения диапазонов параметров функций, возвращаемых значений, условий выхода и операторов безусловного перехода).
- Построение блок-схемы алгоритма программы или отдельных ее частей.
- Генерация отчета о результатах тестирования в формате HTML или формате простого текстового документа.

В основе комплекса находится мультиплатформенная библиотека с базовыми функциями, которую активно используют более платформозависимые прикладные программы, реализующие консольный и оконный интерфейс пользователя. Такой подход позволяет абстрагироваться от особенностей конкретной программной среды и разработать универсальное средство, предназначенное для работы с различными классами ОС. В настоящий момент комплексом поддерживаются операционные системы BeOS, BSD/UNIX, HP-UX, Linux, QNX и Windows.