ОТЗЫВ ОФИЦИАЛЬНОГО ОППОНЕНТА

на диссертационную работу Асланяна Айка Кареновича

«Методы статического анализа для поиска дефектов в исполняемом коде программ»,

представленную к защите на соискание ученой степени кандидата физико-математических наук по специальности 05.13.11 – «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей»

Актуальность темы

Объем и сложность программного обеспечения постоянно растут: размер многих современных проектов составляет миллионы строк исходного кода, а также включает в себя бинарный код проприетарных компонентов. Для поиска ошибок в таких проектах требуется применять автоматические инструменты, использующие анализ программ. Большинство существующих инструментов статического анализа программ, предназначенных для поиска ошибок, работают с исходным кодом программы, но этого иногда оказывается недостаточно, т.к. в больших программных системах могут использоваться компоненты, представленные только в двоичных кодах. Ошибки также могут появиться из-за использования оптимизирующих компиляторов из-за неопределенностей в семантике языков. В таких случаях требуется инструмент анализа исполняемого кода. Именно такой *актуальной* задаче поиска дефектов в исполняемом коде программ на основе методов статического анализа посвящена диссертационная работа Асланяна А. К.. Разработанные в ней методы удовлетворяют следующим требованиям: архитектурная независимость, высокая точность истинных срабатываний и масштабируемость, необходимая для анализа исполняемых файлов размером десятки мегабайт (несколько миллионов строк кода).

Структура работы

Диссертация состоит из четырех глав, введения, заключения и приложения. Объем диссертации составляет 118 страниц, включая 22 таблиц и 11 рисунков. Список литературы содержит 89 наименований.

Во введении сформулирована цель диссертационной работы, дана постановка задач, обоснованы актуальность и научная новизна выбранного направления исследования, сформулированы основные положения, выносимые на защиту, а именно:

- Методы анализа значений и помеченных данных, позволяющие проводить межпроцедурный, чувствительный к контексту и чувствительный к потоку данных анализ исполняемых файлов.

- Методы поиска клонов исполняемого кода и сравнения двух версий исполняемых файлов для автоматического поиска и определения характера изменений в новых версиях программ.
- Методы поиска дефектов и неисправленных фрагментов в новой версии исполняемого файла.

В первой главе приводится обзор существующих методов статического анализа исполняемого кода программ, методов поиска клонов исполняемого кода и сравнения исполняемых файлов.

В обзоре показано, что современные методы статического анализа исполняемого кода восстанавливают ассемблер, графы потока управления, графы вызовов функций, косвенные переходы и косвенные вызовы. На основе восстановленных данных выполняется анализ интервалов значений, анализ потока данных, а также поиск некоторых типов дефектов. Рассматриваются четыре основные группы методов поиска клонов исполняемого кода: методы, которые сравнивают исполняемый код как текст, методы, основанные на токенах, методы, основанные на метриках графов потока управления и графов вызовов, а также методы, основанные на поведении программы. Рассматриваются также несколько подходов к сравнению исполняемых файлов: основанные на метриках, на хешировании, на сравнении графов потока управления и на символьном выполнении.

Отмечается, что существующие анализаторы исполняемого кода имеют ограниченную масштабируемость. В открытом доступе находятся результаты двух инструментов поиска дефектов, которые имеют низкий процент правильных срабатываний (меньше 10%). Рассмотренные методы поиска клонов исполняемого кода и сравнения исполняемых файлов не учитывают поток данных, что влияет на количество правильных срабатываний.

Во второй главе приводится разработанная диссертантом архитектура инструмента анализа исполняемых файлов, которая обеспечивает масштабируемость и независимость от целевой архитектуры, а также поддерживает возможность расширения набора используемых видов анализа. Для целей проведения анализа восстанавливается ассемблер, а затем формируется промежуточное представление с использованием специальных программ с открытым исходным кодом IDA Pro и Binnavi. В качестве промежуточного представления используется REIL (Reverse Engineering Intermediate Language), состоящее из 17 инструкций без побочных эффектов. Поддерживается трансляция ассемблера для архитектур x86, x64 и ARM в REIL. Анализ состоит из шести этапов: дизассемблирование, трансляция ассемблера в REIL, межпроцедурный анализ, внутрипроцедурный анализ, поиск неисправленных фрагментов в новых версиях исполняемых файлов и поиск дефектов.

Межпроцедурный анализ проводится на основе аннотаций функций (каждая аннотация представляет собой краткое описание интересующего поведения функции). Внутрипроцедурный анализ включает в себя разработанные методы

анализа значений, помеченных данных, достигающих определений и трансформации удаления мертвого кода. Для проведения анализа разработана модель памяти, которая моделирует память в стеке, куче и в статической памяти.

В третьей главе описываются методы поиска клонов исполняемого кода, сравнения исполняемых файлов и анализ характера изменений в их новых версиях программ.

Рассматриваются три типа клонов. Разработанный метод поиска клонов позволяет обнаруживать все три типа клонов исполняемого кода. Ключевая особенность метода – эвристический алгоритм поиска наибольшего общего подграфа для двух графов зависимостей программы, который работает за полиномиальное время от количества вершин графов зависимостей.

Алгоритм сравнения двух версий исполняемых файлов основан на методе поиска клонов кода и анализирует графы зависимостей программы, а также графы вызовов функций. Метод анализа характера изменений в новых версиях программ находит все различия между функциями в старых и новых версиях исполняемого файла, которые могут быть вызваны исправлением дефектов. При этом уровень истинных срабатываний составляет более 90%.

В четвертой главе описываются методы поиска дефектов в исполняемом коде. Разработаны методы поиска дефектов использования памяти после освобождения, двойного освобождения памяти, переполнения буферов, форматных строк, внедрения команд и поиска неисправленных фрагментов в новых версиях программ. В конце главы проводится оценка времени и качества анализа, которая показывает, что реализованный инструмент демонстрирует быстрое время работы и высокий процент правильных срабатываний.

В заключении приводятся основные результаты диссертационной работы, определяющие ее *научную новизну*:

- Разработана архитектура и методы статического анализа исполняемого кода, которые являются архитектурно независимыми, масштабируемыми и легко расширяемыми
- Разработаны методы поиска клонов исполняемого кода и сравнения двух версий исполняемых файлов для автоматического поиска и определения характера изменений в новых версиях программ
- Предложены и разработаны методы поиска различных дефектов, внедрения команд и поиска неисправленных фрагментов в новой версии исполняемого кода

В приложении приведен эвристический алгоритм нахождения наибольшего общего подграфа двух графов зависимостей и даны необходимые доказательства оценки его временной сложности.

Работа прошла научную и практическую апробацию, по ней опубликовано 6 научных статей, предложенные методы встроены в практические инструменты.

По диссертации можно отметить следующие **недостатки**:

- Т.к. в работе для дезассемлирования используются программы с открытым исходным кодом, отсутствует информация о полноте восстановления кода

- Не совсем понятно, как использовать информацию о найденных дефектах для программ, исходные коды которых недоступны
- Работа иллюстрируется значительным количеством примеров находимых ошибок, тем не менее, не описан способ вычисления количества истинных срабатываний
- В четвертой главе приводится метод поиска дефектов переполнения буфера, который ограничивается только теми дефектами, которые являются результатом неправильного использования помеченных данных.

Указанные недостатки не снижают общей научной и практической ценности результатов данной работы.

Автореферат полно и правильно отражает содержание диссертационной работы.

Диссертационная работа Асланяна А. К. является завершенным научным исследованием и имеет практическую значимость. Она отвечает всем требованиям, предъявляемым ВАК РФ к кандидатским диссертациям, а ее автор, Асланян Айк Каренович, заслуживает присуждения ему ученой степени кандидата физико-математических наук по специальности 05.13.11 – «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей».

Официальный оппонент
кандидат технических наук, начальник отделения «Системы программирования» Публичного акционерного общества «Институт электронных управляющих машин им. И. С. Брука», Российская федерация, 119334, Москва, ул. Вавилова, д. 24

тел. +7 (499) 135-89-49

адрес электронной почты: ineum@ineum.ru

В.Ю. Волконский

Подпись кандидата технических наук Волконско
В.Ю. заверяю, зам. генерального директора ПАО
«ИНЭУМ им. И.С. Брука» .М. Фельдман

я 2019 г.