На правах рукописи

Туманов Юрий Михайлович

ЗАЩИТА СРЕД ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ ПУТЁМ ВЕРИФИКАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА НАЛИЧИЕ ДЕСТРУКТИВНЫХ СВОЙСТВ

Специальность: 05.13.19 – Методы и системы защиты информации, информационная безопасность

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Автор: Туманов

Москва – 2012

Работа выполнена в Национальном исследовательском ядерном университете «МИФИ» (НИЯУ МИФИ)

Научный руководитель: кандидат физико-математических наук, доцент кафедры №42 «Криптология и дискретная математика», Велигура Александр Николаевич

Официальные оппоненты: Доктор технических наук, ведущий научный сотрудник ИПИ РАН, Королёв Вадим Иванович

Кандидат технических наук, заместитель начальника отдела разработок ООО «КРИПТО-ПРО», Смирнов Павел Владимирович

Ведущая организация: ФГУП «Всероссийский научно-исследовательский институт проблем вычислительной техники и информатизации»

Защита состоится «28» мая 2012 г. в 16 часов 00 минут на заседании диссертационного совета ДМ 212.130.08 при Национальном исследовательском ядерном университете «МИФИ»: 115409, г. Москва, Каширское ш., д.31. Тел. для справок: +7 (495) 323-95-26, 324-73-34.

С диссертацией можно ознакомиться в библиотеке Национального исследовательского ядерного университета «МИФИ».

Отзывы в двух экземплярах, заверенные печатью, просьба направлять по адресу: 115409, г. Москва, Каширское ш., д.31, диссертационные советы НИЯУ МИФИ, тел.: +7 (495) 323-95-26.

Автореферат разослан «25» апреля 2012 г.

Ученый секретарь диссертационного совета,
кандидат технических наук, доцент Горбатов В.С.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. По мере развития информационных технологий стала возрастать потребность частных и юридических лиц удалённо взаимодействовать с большими объёмами данных и как следствие - обеспечивать безопасность такого взаимодействия. Раньше обеспечение этой потребности можно было осуществить посредством использования технологии выделенных серверов, однако, выделенные сервера обладают рядом недостатков: отсутствие отказоустойчивости, невозможность динамического выделения необходимых вычислительных ресурсов при изменении нагрузки.

Среды облачных вычислений (ОВ) являются очередным звеном в эволюционной цепочке подходов к предоставлению удалённого доступа к данным после выделенных серверов. Впервые идея среды ОВ была выдвинута в 1961 году Дж. Маккарти. Среда облачных вычислений - это модель программно-аппаратных средств вычислительной техники, позволяющая получать удалённый доступ к вычислительным ресурсам в любой момент времени. Среда ОВ позволяет динамически выделять требуемое программному обеспечению (ПО) процессорное время и память в зависимости от текущей нагрузки на это ПО. Впервые услуга предоставления доступа к данным на основе среды ОВ была предложена компанией Salesforce в 1999 году. Впоследствии, услуги предоставления доступа к данным с использованием сред ОВ стали предлагать на рынке Amazon (2002), Google (2005), Microsoft(2008) и множество других компаний.

Однако применение сред облачных вычислений ведет к появлению новых проблем информационной безопасности, таких как:

– проблема распространения вредоносного программного обеспечения (ВПО) посредством сред ОВ;
– проблема доверия поставщику услуг среды ОВ;
– проблема выявления ВПО, ориентированного на среды ОВ;
– проблема выявления ПО, не являющегося вредоносным, но содержащим в себе ошибки разработчика, которые могут привести к деструктивному воздействию ПО на среды облачных вычислений.

Для решения задачи противодействия распространению вредоносного программного обеспечения посредством сред ОВ обычно используются существующие решения – антивирусное ПО, системы обнаружения вторжений, системы предотвращения вторжений.

Задача обеспечения доверия поставщику услуг среды ОВ решается посредством административно-правовых и технических мер.

На данный момент не существует решений, позволяющих обеспечить защищённость по ряду параметров, которые определяет поставщик услуг сред облачных вычислений. В частности, не существует решений таких задач как: задачи выявления ВПО, ориентированного на среды ОВ, и задачи выявления программного обеспечения, не являющегося вредоносным, но содержащим в себе ошибки разработчика.

Таким образом, к настоящему времени являются актуальными и требуют решения следующие задачи:

– задача выявления ВПО, ориентированного на среды облачных вычислений;
– задача выявления ПО, не являющегося вредоносным, но содержащего в себе ошибки разработчика, которые могут привести к деструктивному воздействию программного обеспечения на среды облачных вычислений.

В дальнейшем перечисленные виды программного обеспечения будут обозна-

чаться как ПО, обладающее деструктивными свойствами для сред облачных вычислений.

Исследование области обеспечения безопасности сред облачных вычислений проводились как российскими, так и зарубежными учеными, среди которых следует отметить:

– Danish Jamil – провёл типизацию угроз для сред облачных вычислений и предложил ряд решений, позволяющих противодействовать рассмотренным угрозам;

– Michael Miller – провёл анализ механизмов безопасности сред облачных вычислений и выделил общие неустранимые недостатки;

– Станкевичус А.А. – провёл анализ возможности создания безопасной ячейки памяти и предложил подход к её реализации, позволяющий предотвращать несанкционированный доступ;

– Subashini S. – рассмотрел проблемы применения верификации в средах облачных вычислений и сделал вывод о сложности применения существующих методик верификации для ПО, передаваемого в среды ОВ, для выполнения.

В работе предлагается методика, позволяющая выделять набор деструктивных свойств ПО, верификацию на отсутствие которых требуется проводить. С использованием предложенной методики возможно выделить наборы программных инструкций, выявление которых позволит назвать программное обеспечение некорректным для конкретной среды ОВ. Например, не проводя проверки ПО на ошибки утечки памяти, некорректную работу с вычислительными или емкостными ресурсами, возможно назвать ПО некорректным, если в анализируемом программном обеспечении присутствуют вызовы определённых программных инструкций.

Подобный подход позволит поставщику услуг среды облачных вычислений самостоятельно выбирать набор деструктивных свойств ПО, анализ на наличие которых требуется произвести перед его выполнением в среде ОВ. Также предложенный подход позволит не проводить полную формальную верификацию корректности работы ПО, что занимает длительное время, требует привлечения отдельных специалистов и невозможно в автоматизированном режиме.

Возможность задания количества редакционных операций над словом в ходе проведения верификации ПО на отсутствие определенного деструктивного свойства позволяет искать либо строго заданное свойство в анализируемом ПО, либо подобные деструктивные свойства ПО.

Объект исследования. Среды облачных вычислений.

Предмет исследования. Деструктивные свойства программного обеспечения сред облачных вычислений.

Цель диссертационной работы. Повышение защищённости сред облачных вычислений путём выявления деструктивных свойств программного обеспечения.

Научная задача заключается в синтезе методики верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений.

Для решения поставленной задачи необходимо:

– провести анализ: требований, предъявляемым к методикам обеспечения защиты сред ОВ; методик верификации кода; методик выявления ПО, обладающего деструктивными свойствами;

– создать модель нарушителя среды ОВ для выявления возможных угроз и на её основе сформулировать ограничения при применении предлагаемой методики;

– сформулировать требования к разрабатываемой методике, удовлетворяющие специфике применения сред ОВ;

– синтезировать модель представления ПО, которая позволит анализировать все возможные пути выполнения ПО, а также программные инструкции, и на её основе создать формальное описание классифицирующего признака ПО;

– создать алгоритм классификации программного обеспечения;

– синтезировать методику верификации ПО для сред ОВ;

– разработать архитектуру программного комплекса и реализовать его.

Основными **методами исследований**, используемыми в работе, являются методы теории графов, теории множеств.

Научная новизна работы состоит в следующем:

– синтезирована новая математическая модель представления ПО, представленная в терминах теории графов и теории множеств, позволяющая анализировать процесс выполнения ПО и его инструкции;

– предложен способ формального описания классифицирующего признака ПО, основанный на синтезированной модели представления ПО;

– предложен новый алгоритм классификации программного обеспечения на ПО, обладающее заданным признаком, и ПО, не обладающее им;

– предложен подход к оценке подобия различных экземпляров программного обеспечения, основанный на мере Дамерау – Левенштейна;

– синтезирована методика верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений, использующая предложенный подход к оценке подобия различных экземпляров ПО.

Практическая значимость результатов заключается в следующем:

– реализован алгоритм, использующий меру Дамерау – Левенштейна, позволяющий в зависимости от заданного количества редакционных преобразований выявлять подобные экземпляры ПО;

– реализован программный комплекс, позволяющий осуществлять верификацию программного обеспечения на наличие деструктивных свойств для сред облачных вычислений;

– сформулированы рекомендации по использованию практических и теоретических результатов работы для верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений.

Результаты работы представляют практическую ценность для создания программных комплексов верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений.

Внедрение результатов исследований.

Предложенная методика верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений реализована в Центре вирусных исследований и аналитики «Eset» для осуществления формального вывода при анализе ПО о наличии в нём деструктивных свойств ПО.

Предложенная методика верификации ПО на наличие деструктивных свойств для сред облачных вычислений, реализована в компании ООО «Связьмонтажкомплектация» для повышения качеств предоставляемых компанией услуг по тестированию ПО для сред ОВ.

Предложенные в работе модели представления ПО и описаний деструктивных

свойств ПО реализованы в компании ООО «ТСС» для проведения оценки рисков информационной безопасности и создания моделей информационных систем.

Теоретические и прикладные результаты, полученные в ходе выполнения диссертационной работы, использованы в учебном курсе «Языки программирования» кафедры «Криптология и дискретная математика» НИЯУ МИФИ при создании лабораторных работ по курсу «Языки программирования 2».

Публикация и апробация работы. Результаты диссертации изложены в 11 публикациях, 7 из которых опубликованы в журналах рецензируемых ВАК РФ. Результаты работы докладывались на конференциях и семинарах различного уровня:

– 7-я Курчатовская молодёжная научная школа – 10-12.11.2009г, г. Москва;

– «Умник сколковец» – 15.03.2011г., г. Москва, МТЦ. «Победитель конкурса «Кадровый резерв молодых ученых и специалистов «Сколково» 15.03.2011г.;

– «Microsoft Research Summer School 2011» – 27.06-01.07.2011г, Великобритания, Кембридж;

– 11-й Национальный форум информационной безопасности «Информационная безопасность России в условиях глобального информационного общества» 29-30 января 2009 г.;

– Конкурс молодёжных предпринимательских проектов «Своё дело» 18 мая 2011г, г. Москва;

– X конкурс молодёжных инновационных проектов технопарка МИФИ. 2011г. г. Москва;

– XIV международная телекоммуникационная конференция молодых учёных и студентов «Молодежь и наука». 2011г. г. Москва.

Основные положения выносимые на защиту:

– математическая модель представления ПО, позволяющая получать формальный вывод о наличии или отсутствии деструктивных свойств ПО, анализировать процесс выполнения ПО и его инструкции;

– формальное описание классифицирующего признака ПО, использование которого обеспечивает отсутствие возможности пропуска ПО, обладающего известным деструктивным свойством;

– алгоритм классификации ПО, использующий подход к оценке подобия различных экземпляров ПО, основанный на мере Дамерау-Левенштейна, позволяющий автоматизировать разработанную методику;

– методика верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений, не требующая постоянного использования ресурсов среды ОВ, позволяющая выявлять новые и модифицированные экземпляры ПО, обладающего деструктивными свойствами;

– программный комплекс, повышающий защищённость сред ОВ, с использованием предложенной в работе методики.

Структура работы. Работа состоит из введения, четырех глав, заключения, списка литературы, включающего 106 наименований. Текст диссертации изложен на 135 страницах, включая 15 рисунков и 3 таблицы.

СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обоснована актуальность темы диссертации, выделены и сформулированы цель и задачи исследования, описана логика и структура диссертационной работы, определена новизна и практическая значимость диссертационной работы.

В **первой главе** представлены результаты анализа различных реализаций сред облачных вычислений. Рассмотрены требования, предъявленные к программному обеспечению, создаваемому для выполнения в средах облачных вычислений. На основании проведённого анализа создана схема сред облачных вычислений, представленная на рисунке 1. Также проведён сравнительный анализ существующих методик выявления деструктивных свойств программного обеспечения. В результате проведённого анализа сделан вывод о неэффективности существующих методик выявления деструктивных свойств ПО при их использовании в средах облачных вычислений и принято решение о создании методики, лишённой выявленных недостатков.

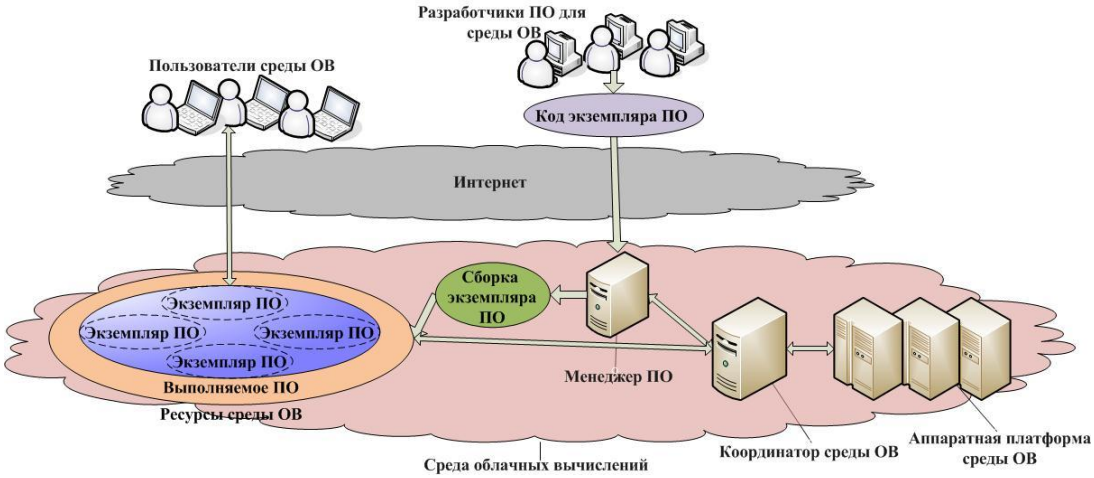

Рисунок 1 – Схема сред облачных вычислений

Рассмотрены виды атак на среды, в которых происходит выполнение кода программного обеспечения, возможные в результате некорректной реализации исходного кода ПО. Проведён анализ возможности реализаций атак на среды ОВ, аналогичных атакам в иных вычислительных средах. Результатом анализа является вывод о том, что в общем случае среды облачных вычислений являются уязвимыми к атакам, возможным в иных вычислительных средах. Также за счёт появления новых программных и аппаратных решений, при создании сред облачных вычислений, появляются новые виды атак, ориентированных на среды облачных вычислений.

На данный момент не существует единой модели нарушителя или модели угроз, рассматриваемых поставщиками услуг сред ОВ. Поставщики услуг сред облачных вычислений предпринимают меры, направленные на противодействие передаче в среду ОВ программного обеспечения, реализующего алгоритмы, деструктивно воздействующие на среды облачных вычислений. Предпринимаемые меры различаются в зависимости от политик безопасности, установленных в организации, являющейся поставщиком услуг сред ОВ. Основными применяемыми мерами являются следующие:

– административные;
– нормативно-правовые;
– технические меры, в частности, формальная верификация кода программного обеспечения.

Принимаемые поставщиками меры, направленные на противодействие передаче ПО, деструктивно воздействующего на среды облачных вычислений, обладают следующими недостатками:

– административные и нормативно-правовые меры не защищают от передачи

ПО, обладающего деструктивными свойствами для сред облачных вычислений, а лишь сообщают нарушителю реакцию организации на факт передачи подобного ПО;

– формальная верификация программного кода занимает большое количество человеческих ресурсов и затрагивает свойства ПО, которые могут быть несущественны для поставщика услуг среды облачных вычислений.

Решение задач, актуальных на данный момент для сред облачных вычислений возможно осуществлять при помощи программных реализаций существующих сигнатурных и проактивных методик, а также при помощи методик верификации кода.

Однако существующие программные реализации методик выявления деструктивных свойств ПО изначально созданы не для обеспечения безопасности облачных вычислений. Также реализованные в них методики выявления деструктивных свойств обладают описанными ниже недостатками.

В случае использования сигнатурных методик требуется, чтобы в анализируемом ПО присутствовала строго описанная сигнатура ПО, обладающего деструктивными свойствами - последовательность инструкций, свёртка или иное строго заданное свойство. Сигнатурные методики затруднительно применять для сред облачных вычислений, в общем случае в связи с тем, что среды облачных вычислений создаются с использованием различных программно-аппаратных платформ, и, даже если вредоносные алгоритмы будут совпадать, их реализация будет отличаться. Таким образом, для каждой среды ОВ потребуется создавать новый набор сигнатур с учётом программных интерфейсов, реализованных в ней.

Методика статистического анализа требует информации обо всех программных интерфейсах, предоставленных разработчику программного обеспечения, и для каждой среды облачных вычислений требуется создавать новый набор статистических данных, характеризующих деструктивные свойства.

Методика эвристического анализа с эмуляцией работы ПО для выявления деструктивных свойств требует от разработчика средств противодействия ПО, обладающего деструктивными свойствами, создания программно-аппаратной или программной среды, эмулирующей рассматриваемую среду облачных вычислений. Реализация данной методики связана с большими затратами человеческого ресурса и повышенной нагрузкой на среду ОВ.

Методика поведенческого анализа ПО на наличие деструктивных свойств для сред облачных вычислений требует полной переработки базы поведенческих свойств деструктивного ПО и создаёт существенную нагрузку на среду ОВ. Таким образом, применение методики поведенческого анализа при выявлении деструктивных свойств ПО экономически не выгодно для поставщика услуг сред облачных вычислений.

Методика виртуализации рабочего окружения для деструктивных свойств требует значительной нагрузки на среду облачных вычислений и, как и в случае поведенческого анализа, экономически не выгодна поставщику услуг среды ОВ.

Методики верификации в общем случае не автоматизируемы и требуют значительных временных затрат при создании программного кода, отвечающего заданным в методиках верификации критериям.

Исходя из вышеописанных недостатков методик, актуальной является разработка методики выявления деструктивных свойств ПО, ориентированной на среды облачных вычислений.

Также в первой главе проводится сравнительный анализ существующих методик

выявления деструктивных свойств, реализованных в антивирусном ПО, системах обнаружения, системах предотвращения вторжений, методиках проведения верификации ПО. Результаты анализа методик представлены в таблице 1.

Таблица 1 – Сравнительная характеристика методик.

Анализируемая методика	Критерий сравнения				
	Отсутствие возможности пропуска ПО, обладающего известным деструктивным свойством	Возможность автоматизации методики при выявлении деструктивных свойств	Возможность выявления модифицированных и новых деструктивных свойств	Отсутствие необходимости постоянного использования ресурсов среды ОВ в ходе исполнения методики	Возможность формального вывода о наличии или отсутствии деструктивных свойств
Методики сигнатурного анализа	+	+	–	+	–
Методики статистического анализа	–	+	+	–	–
Методики эвристического анализа	–	+	+	–	–
Методики поведенческого анализа	–	+	+	–	–
Методики виртуализации рабочего окружения	–	+	+	–	–
Методики автоматизируемой верификации, проводимой с участием человека	+	–	+	+	+
Методики автоматической верификации исходного кода	+	–	+	+	+
Методики автоматической верификации исполнимого кода	+	–	+	+	+
Авторская методика	+	+	+	+	+

Предложенная автором в работе методика верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений лишена недостатков приведённых выше методик при использовании в среде ОВ и обладает следующими преимуществами:

– предложенная методика разработана с учётом возможности использования вычислительных и емкостных ресурсов, предоставляемых средами облачных вычислений;

– предложенная методика позволяет получить формальный вывод о наличии деструктивных свойств в верифицируемом экземпляре ПО автоматизировано;

– предложенная методика позволяет выявлять новые и модифицированные существующие экземпляры ПО, обладающие деструктивными свойствами;

– в предложенной методике отсутствует возможность пропуска известного образца ПО, обладающего деструктивными свойствами.

Во **второй главе** приводятся результаты исследований различных подходов к выявлению деструктивных свойств программного обеспечения. Проводится построение модели нарушителя для представленной общей модели сред облачных вычислений. Для построенной модели нарушителя выдвигаются ограничения, в рамках которых применима предложенная методика. Далее производится построение модели верифицируемого ПО, формальное описание классифицирующего признака ПО, ставится задача классификации ПО, приводится алгоритм её решения. Приводится синтезированная методика верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений.

После наложения ограничений на общую модель нарушителя для сред облачных вычислений предполагается, что нарушитель обладает следующими возможностями:

– нарушитель обладает полными знаниями о программно-аппаратной среде ОВ;

– нарушитель обладает всеми необходимыми ресурсами для создания ПО для рассматриваемой среды ОВ;

– нарушитель имеет возможность передачи ПО в среду ОВ только штатным путем.

В рамках предложенной работы потенциальным нарушителем являются лицо, осуществляющее передачу программного кода ПО в среду облачных вычислений по штатному каналу.

Математическая модель представления ПО, разработанная в ходе выполнения работы, выглядит следующим образом.

Пусть существует алфавит $\boldsymbol{L}$, описывающий реализованный в среде ОВ язык программирования, и граф потока управления ПО $\boldsymbol{G}$, представляющий все возможные пути исполнения ПО.

Пусть алфавит $\boldsymbol{L}$ является конечным множеством мощности $|\boldsymbol{L}|$ и состоит из всех программных инструкций языка программирования, на котором разработан экземпляр верифицируемого ПО, тогда словом $\boldsymbol{w}$, принадлежащем множеству слов $\boldsymbol{W}$, называется конечная последовательность элементов $\boldsymbol{w} = \bigcup_i l_i$, где символ $\boldsymbol{l} \in \boldsymbol{L}$, $\boldsymbol{i} = \overline{1,k}$, $\boldsymbol{k} \in \boldsymbol{N_0}$.

Пусть граф потока управления ПО $\boldsymbol{G} = \{\boldsymbol{U},\boldsymbol{V}\}$ задается набором множеств: $\boldsymbol{U}$ – множество вершин рассматриваемого графа, $\boldsymbol{V}$ – множество дуг рассматриваемого графа. В графе потока управления каждая вершина $\boldsymbol{u} \in \boldsymbol{U}$ соответствует линейному

блоку ПО – участку кода, не содержащему в себе ветвлений и циклов. Линейные блоки ПО (в дальнейшем базовые блоки ПО) – участки программного кода приложения, не содержащие ветвлений, переходов и инструкций передачи управления. Множество дуг $\boldsymbol{V}$ включает в себя пары элементов из множества $\boldsymbol{U}$, то есть $\boldsymbol{V} \subset \{\boldsymbol{U} \text{x} \boldsymbol{U}\}$, таким образом, что дуга между двумя вершинами существует тогда и только тогда, когда переход из одной вершины в другую осуществим. Таким образом, если набор инструкций ПО, представленных вершиной графа $\boldsymbol{u_k}$, будет вызван сразу после обращения к инструкциям, представленных вершиной графа $\boldsymbol{u_i}$, то вершины $\boldsymbol{u_k}$ и $\boldsymbol{u_i}$ будут связаны между собой дугой $\{\boldsymbol{u_k},\boldsymbol{u_i}\}$. Существенно, что рассматриваемый граф потока управления ПО является ориентированным в связи с тем, что при исполнении кода приложения важен порядок вызова базовых блоков ПО.

В данной работе предлагается представлять программное обеспечение в виде следующей математической модели, описанной в терминах теории множеств и теории графов:

$\boldsymbol{R} = \{\boldsymbol{U},\boldsymbol{W},\boldsymbol{h},\boldsymbol{U_c},\boldsymbol{V},\boldsymbol{f},\boldsymbol{V_c}\}$, где $\boldsymbol{U}$ – множество вершин графа потока управления ПО, $\boldsymbol{W}$ – множество слов на рассматриваемом языке программирования, $\boldsymbol{h}$ – отображение разметки вершин графа потока управления $\boldsymbol{U}$, $\boldsymbol{h}$: $\boldsymbol{W} \times \boldsymbol{U} \rightarrow \boldsymbol{U_c}$, где $\boldsymbol{U_c}$ – множество размеченных вершин графа потока управления (далее вершин), $\boldsymbol{V}$ – множество дуг графа потока управления ПО, $\boldsymbol{f}$ – отображение $\boldsymbol{f}$:$\boldsymbol{V} \rightarrow \boldsymbol{V_c}$, где множество дуг $\boldsymbol{V_c} \subset \{\boldsymbol{U_c} \text{x} \boldsymbol{U_c}\}$ включает в себя пары элементов из множества $\boldsymbol{U_c}$, таким образом, что дуга между двумя вершинами существует тогда и только тогда, если она существовала во множестве дуг $\boldsymbol{V}$. Тогда под $\boldsymbol{G_c} = \{\boldsymbol{U_c},\boldsymbol{V_c}\}$ понимается размеченный граф потока управления $\boldsymbol{G} = \{\boldsymbol{U},\boldsymbol{V}\}$. В дальнейшем под путём $\boldsymbol{g_c} \in \boldsymbol{G_c}$ будет пониматься последовательность связанных переходами базовых блоков программного обеспечения, от входного блока, с которого начинается выполнение ПО, до конечного блока, в котором завершается выполнение ПО. Внутреннее представление программного обеспечения, используемое непосредственно в реализации методики, схематично изображено на рисунке 2.

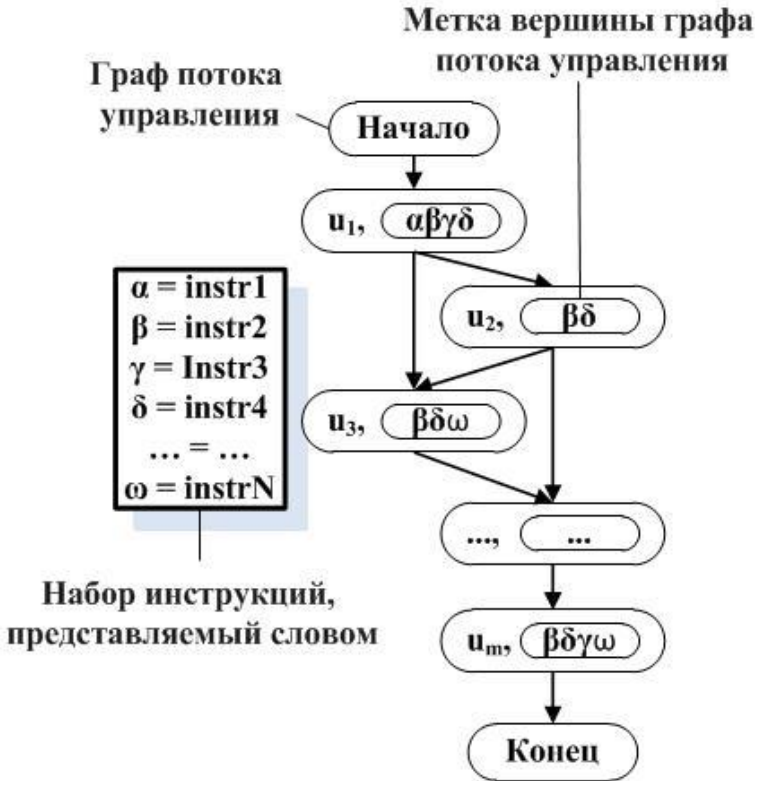

Рисунок 2 – Размеченный граф потока управления программного обеспечения

В ходе выполнения работы было синтезировано формальное описание классифицирующего признака ПО, характеризующее деструктивное свойство ПО, используемое в предложенной методике. Формальное описание классифицирующего признака представлено в терминах теории множеств и теории графов:

$\boldsymbol{CR} = \{\boldsymbol{W},\boldsymbol{D},\boldsymbol{PD},\boldsymbol{DPD},\boldsymbol{P},\boldsymbol{\psi}\}$, где $\boldsymbol{W}$ – множество слов на рассматриваемом языке программирования, $\boldsymbol{D}$ – множество размеченных доминаторов (далее доминаторов), $\boldsymbol{PD}$ – множество размеченных постдоминаторов (далее постдоминаторов), $\boldsymbol{DPD}$ – множество размеченных вершин, являющихся доминаторами и постдоминаторами (далее доминаторов-постдоминаторов), $\boldsymbol{P}$ – множество всех путей из множества доминаторов в множество постдоминаторов, проходящих через множество вершин, являющихся доминаторами и постдоминаторами, $\boldsymbol{\psi}$ – отображение, $\boldsymbol{\psi}:\boldsymbol{D}\times\boldsymbol{DPD}\times\boldsymbol{PD}\rightarrow\boldsymbol{P}$. В дальнейшем под вершиной $\boldsymbol{p}\in\boldsymbol{P}$ будет пониматься доминатор, постдоминатор или вершина, являющаяся доминатором и постдоминатором. Под путём $\boldsymbol{t}\in\boldsymbol{P}$ будет пониматься заданное подмножество вершин $\boldsymbol{p}\in\boldsymbol{P}$, которые связаны отношениями доминирования и постдоминирования, описывающих определенное деструктивное свойство ПО. Иными словами, в соответствии с предложенным формальным описанием классифицирующего признака ПО для каждого нового образца ПО, обладающего деструктивными свойствами, в базу данных добавляются информация о: размеченных вершинах графа потока управления и в какой последовательности они должны вызываться в ходе выполнения ПО.

Формальное описание классифицирующего признака, характеризующее деструктивное свойство ПО, используемое непосредственно в реализации методики, схематично представлено на рисунке 3.

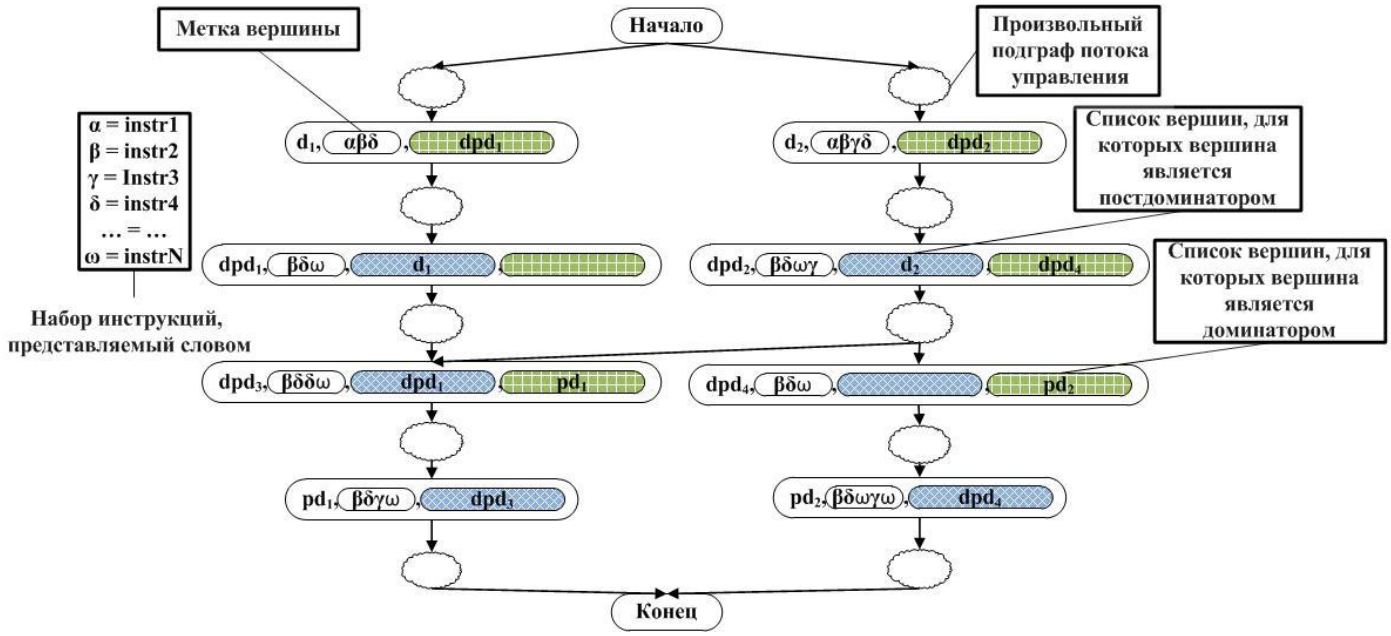

Рисунок 3 – Схематичное изображение формального описания классифицирующего признака ПО

Вершины $\boldsymbol{p}\in\boldsymbol{P}$ и $\boldsymbol{u_c}\in\boldsymbol{G_c}$ рассматриваются как подобные, если $\boldsymbol{DL}(\boldsymbol{p},\boldsymbol{u_c})\leq\boldsymbol{\theta}$, где $\boldsymbol{DL}(\boldsymbol{p},\boldsymbol{u_c})$ – преобразование Дамерау – Левенштейна, позволяющее получить минимальное количество операций вставки, удаления, замены и перестановки одного символа алфавита $\boldsymbol{l}\in\boldsymbol{L}$, необходимых для преобразования инструкций вершины $\boldsymbol{p}\in\boldsymbol{P}$ в

вершину $\boldsymbol{u_c}\in\boldsymbol{G_c}$, $\boldsymbol{\theta}$ – заданное количество редакционных преобразований над помеченной вершиной, в случае превышения которого вершины $\boldsymbol{p}\in\boldsymbol{P}$ и $\boldsymbol{u_c}\in\boldsymbol{U_c}$ не рассматриваются как подобные. Если $\boldsymbol{DL}(\boldsymbol{p},\boldsymbol{u_c})=0$, то вершины $\boldsymbol{p}\in\boldsymbol{P}$ и $\boldsymbol{u_c}\in\boldsymbol{G_c}$ рассматриваются как эквивалентные.

Пути $\boldsymbol{g_c}\in\boldsymbol{G_c}$ и $\boldsymbol{t}\in\boldsymbol{P}$ рассматриваются как подобные, если для каждой вершины $\boldsymbol{p}$, входящей в путь $\boldsymbol{t}$, существует вершина $\boldsymbol{u_c}$, входящая в путь $\boldsymbol{g_c}$, такая, что $\boldsymbol{DL}(\boldsymbol{p},\boldsymbol{u_c})\leq\boldsymbol{\theta}$. Пути $\boldsymbol{g_c}\in\boldsymbol{G_c}$ и $\boldsymbol{t}\in\boldsymbol{P}$ рассматриваются как эквивалентные, если для каждой вершины $\boldsymbol{p}$, входящей в путь $\boldsymbol{t}$, существует вершина $\boldsymbol{u_c}$, входящая в путь $\boldsymbol{g_c}$, такая, что $\boldsymbol{DL}(\boldsymbol{p},\boldsymbol{u_c})=\boldsymbol{\theta}$ $\forall$ $\boldsymbol{p},\boldsymbol{u_c}$, и в пути $\boldsymbol{g_c}$ отсутствуют вершины, кроме эквивалентных вершинам, входящим в $\boldsymbol{t}$.

Постановка задачи. Необходимо провести классификацию множества элементов, каждый элемент которого описывается кортежем $\boldsymbol{R}$, для выделения двух классов эквивалентности: ПО, обладающее классифицирующим признаком – $\boldsymbol{C}(\boldsymbol{M})$ и ПО, не обладающее классифицирующим признаком – $\boldsymbol{C}(\overline{M})$.

Критерием классификации является наличие подобных или эквивалентных путей $\boldsymbol{t}\in\boldsymbol{P}$ и $\boldsymbol{g_c}\in\boldsymbol{G_c}$. Таким образом, $\boldsymbol{R}\subset\boldsymbol{C}(\overline{M})$, если существует хотя бы один путь $\boldsymbol{g_c}\in\boldsymbol{G_c}$, такой, что для него существует подобный путь $\boldsymbol{t}\in\boldsymbol{P}$, для всех описаний классифицирующих признаков, представленных кортежами $\boldsymbol{CR}$. $\boldsymbol{R}\subset\boldsymbol{C}(\boldsymbol{M})$, если для любого пути $\boldsymbol{g_c}\in\boldsymbol{G_c}$, не существует подобного пути $\boldsymbol{t}\in\boldsymbol{P}$, для всех классифицирующих признаков, представленных кортежами $\boldsymbol{CR}$.

Иными словами, в верифицируемом образце ПО необходимо найти все пути, описывающие деструктивные свойства ПО. В случае, если в верифицируемом образце ПО найдены вершины подобные вершинам, описывающим деструктивное свойство ПО, и найденные подобные вершины доминируют и постдоминируют друг друга так же, как в описании деструктивного свойства ПО, то программное обеспечение классифицируется как обладающее деструктивными свойствами. В дальнейшем под описанием деструктивного свойства ПО будем понимать классифицирующий признак ПО.

Методика верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений состоит из следующих этапов:

– получение исходного кода программного обеспечения;
– построение графа потока управления ПО;
– создание размеченного графа потока управления ПО;
– поиск подобных вершин в размеченном графе управления ПО и описаниях деструктивных свойств ПО;
– поиск подобных путей в размеченном графе потока управления ПО и описаниях деструктивных свойств ПО;
– классификация ПО, как обладающего деструктивными свойствами или не обладающего деструктивными свойствами;
– передача ПО в среду облачных вычислений для выполнения или блокирование передачи.

Далее приведена теоретико-вероятностная оценка вероятности ложного срабатывания при выявлении деструктивных свойств ПО, в зависимости от заданного количества редакционных преобразований над вершиной графа $\boldsymbol{G_c}$.

В **третьей главе** приведено описание архитектуры программного комплекса, реализующего предложенную в работе методику.

Для реализации предлагаемой в работе методики верификации программного

обеспечения на наличие деструктивных свойств для сред облачных вычислений предлагается следующий набор функциональных модулей:

– модуль построения графа потока управления верифицируемого ПО;
– модуль разметки вершин графа потока управления верифицируемого ПО;
– модуль хранения внутреннего представления верифицируемого ПО;
– модуль хранения описаний деструктивных свойств ПО;
– модуль сравнения помеченных вершин верифицируемого ПО и вершин деструктивных свойств ПО;
– модуль поиска пути в размеченном графе потока управления верифицируемого ПО;
– модуль классификации верифицируемого ПО;
– модуль передачи ПО в среду ОВ.

Модуль построения графа потока управления верифицируемого ПО. Производит построение графа потока управления верифицируемого ПО или позволит сделать вывод о том, что построение графа потока управления невозможно.

Модуль разметки вершин графа потока управления верифицируемого ПО. В данном модуле происходит трансляция исходного кода верифицируемого ПО и его графа потока управления во внутреннее представление, используемое в предложенной методике. После создания внутреннего представления верифицируемого ПО полученное внутреннее представление передаётся в модуль хранения внутреннего представления верифицируемого ПО.

Модуль хранения внутреннего представления верифицируемого ПО. Осуществляет предоставление необходимых данных об анализируемом ПО модулю сравнения помеченных вершин и модулю поиска пути в размеченном графе потока управления, а также предоставляет им вычислительные и емкостные ресурсы среды ОВ для выполнения возложенных на них функций.

Модуль хранения описаний деструктивных свойств ПО. Производит предоставление описаний деструктивных свойств модулю сравнения помеченных вершин и модулю поиска пути в размеченном графе потока управления.

Модуль сравнения помеченных вершин верифицируемого ПО и вершин деструктивных свойств ПО. Реализует алгоритм сравнения слов, составленных из символов рассматриваемого языка программирования, являющихся метками вершин графа потока управления верифицируемого ПО, с описаниями деструктивных свойств ПО.

Модуль поиска пути в размеченном графе потока управления верифицируемого ПО. Отвечает за поиск пути между выявленными подобными вершинами в размеченном графе потока управления верифицируемого ПО и описаниях деструктивных свойств ПО.

Модуль классификации верифицируемого ПО. На основании данных, полученных из модуля поиска пути в размеченном графе потока управления, осуществляет классификацию верифицируемого ПО как обладающего деструктивными свойствами или не обладающего деструктивными свойствами. Результат классификации передаётся в модуль передачи ПО в среду ОВ.

Модуль передачи ПО в среду ОВ. Производит передачу ПО, классифицированного как не обладающего деструктивными свойствами, в среду ОВ или блокирует передачу, и производит оповещение администратора, если ПО классифицировано как

обладающее деструктивными свойствами.

Архитектура программного комплекса представлена на рисунке 4.

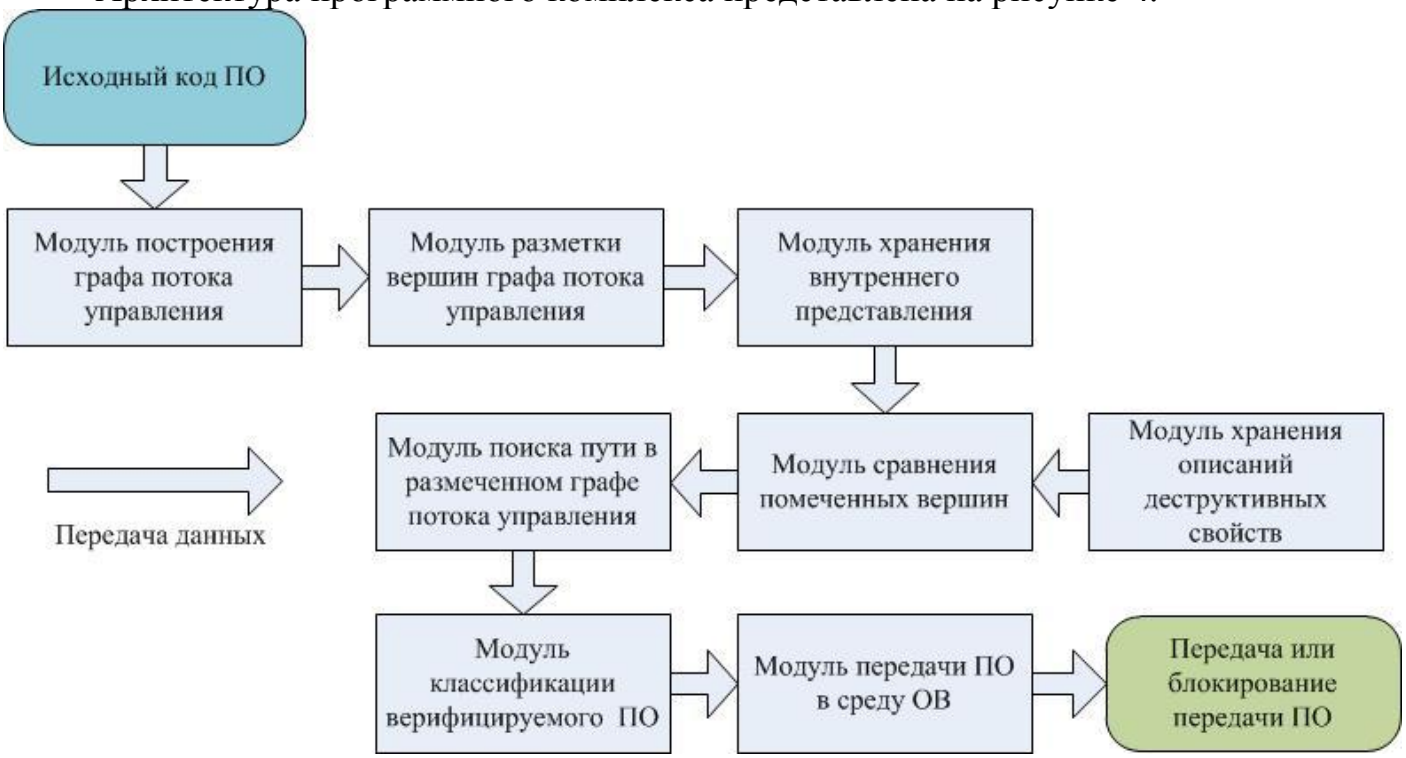

Рисунок 4 – Архитектура программного комплекса

На рисунке 5 представлена построенная автором модель работы программного комплекса, реализующего предложенную методику.

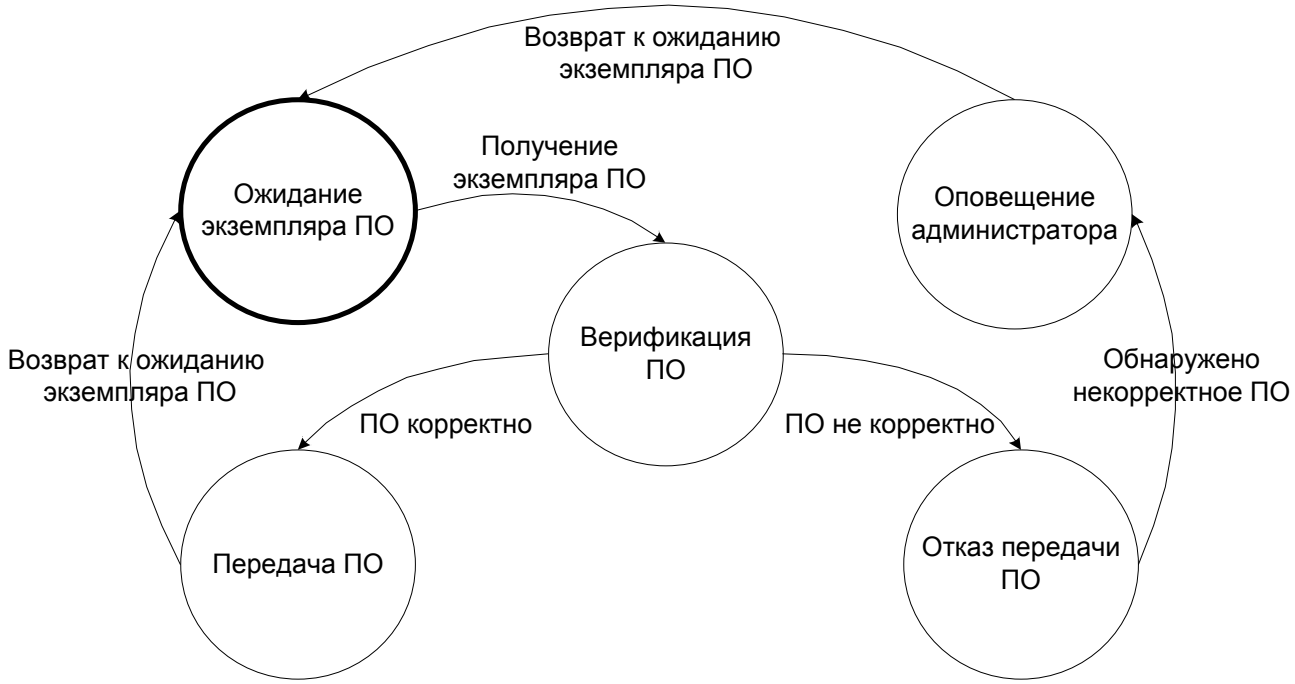

Рисунок 5 – Модель работы программного комплекса, реализующего предложенную в работе методику

Программный комплекс может находиться в следующих состояниях: **Ожидание экземпляра ПО** – начальное состояние системы; **Верификация ПО** – состояние, в которое переходит система при получении нового экземпляра ПО; **Передача ПО** – состояние, в которое переходит программный комплекс, в случае, если при анализе ПО не выявлено деструктивных свойств; **Отказ передачи ПО** – состояние, в которое пе-

реходит программный комплекс, если в результате анализа подтверждено присутствие деструктивных свойств; **Оповещение администратора** – состояние, в которое переходит программный комплекс, если образцу ПО отказано в передаче в среду ОВ. Переходы в модели помечены как: **ПО корректно** – переход, в случае если деструктивные свойства не выявлены; **ПО не корректно** – переход, приводящий к предотвращению передачи ПО в среду ОВ; **Обнаружено некорректное ПО** – переход, в случае если образцу ПО отказано в передаче в среду ОВ; **Возврат к ожиданию экземпляра ПО** – переход, возвращающий программный комплекс в начальное состояние.

В **четвертой главе** производится выбор алгоритмов и языка программирования для реализации методики верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений. Приводится описание разработанных структур данных, рассматривается реализация разработанной методики, приводится оценка производительности реализованной методики.

Разработка программного комплекса, реализующего методику, осуществлялась на языке программирования C# платформы .NET в связи с тем, что для работы с внутренним представлением .NET приложений наиболее полно подходят существующие в данном языке классы и пространства имен. Также для разработки использована СУБД Oracle 11g, используемая для проведения поиска по табличным описаниям деструктивных свойств ПО.

В ходе реализации алгоритмов, входящих в методику, произведены модификации использованных алгоритмов для снижения вероятности ложных срабатываний при использовании меры Дамерау – Левенштейна для выявления подобных вершин размеченного графа потока управления. Также в главе приводится обоснование выбора определённых алгоритмов на основании оценок их временных и емкостных сложностей.

Приводятся результаты тестирования разработанного программного комплекса верификации ПО на наличие деструктивных свойств для сред облачных вычислений.

В таблице 2, приведена доля ложных срабатываний реализованного программного комплекса, полученная в ходе его работы, в зависимости от количества редакционных преобразований над словом.

Таблица 2– Доля ложных срабатываний в зависимости от количества редакционных преобразований над словом.

Количество редакционных преобразований над словом, Θ	0	1	2	3	4	5
Доля ложных срабатываний	0,05	0,14	0,23	0,36	0,49	0,65

Алгоритм работы программного комплекса, реализующего методику верификации, созданную в ходе выполнения работы, представлен на рисунке 6.

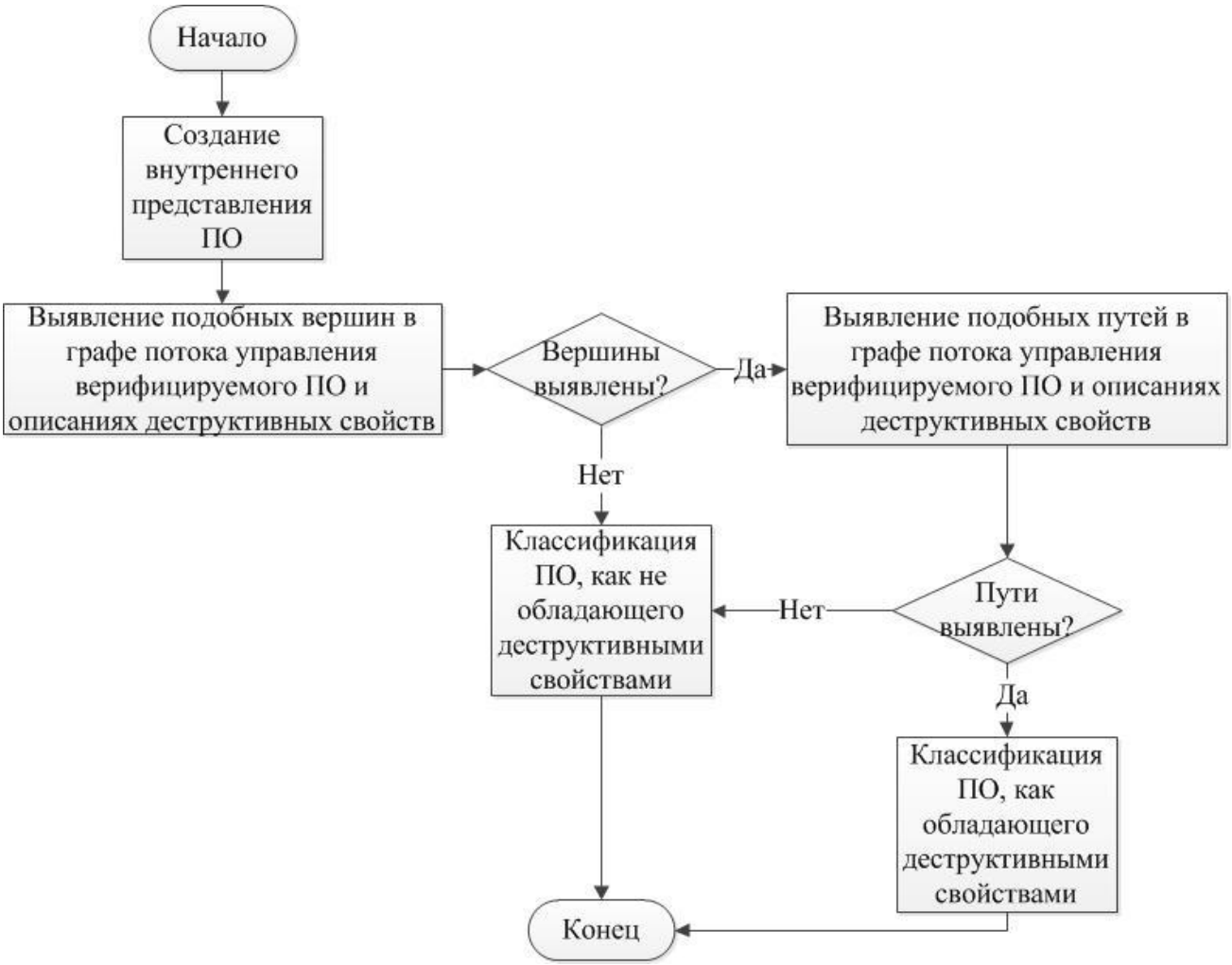

Рисунок 6 – Алгоритм программного комплекса

Предложенная методика верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений реализована в Центре вирусных исследований и аналитики «Eset». Реализованный программный комплекс используется для получения формального вывода о наличии или отсутствии деструктивных свойств ПО в анализируемых образцах ПО.

Программный комплекс, реализующий методику верификации ПО на наличие деструктивных свойств для сред ОВ, был внедрён в компании ООО «Связьмонтаж-комплектация» для тестирования ПО на наличие в нём деструктивных свойств для сред облачных вычислений.

Элементы предложенной методики внедрены в компанию ООО «ТСС». В компании ООО «ТСС» проведено внедрение следующих элементов методики: модель описания ПО, модель описания деструктивного свойства ПО, подход к оценке вероятности ошибок первого и второго рода, что позволило автоматизировать и формализовать процесс оценки рисков систем информационной безопасности.

Результаты диссертационной работы использованы на кафедре «Криптология и дискретная математика» НИЯУ МИФИ в учебном курсе «Языки программирования 2». В результате внедрения созданы две лабораторные работы, позволяющие слушателям получить знания о требованиях к ПО, создаваемому для выполнения в средах ОВ и о существующих методиках верификации ПО.

В **заключении** приведены основные результаты диссертационной работы, а также представлены выводы, полученные в ходе выполнения работы.

ОСНОВНЫЕ ВЫВОДЫ И РЕЗУЛЬТАТЫ РАБОТЫ

1. Синтезирована математическая модель представления программного обеспечения, позволяющая получать формальный вывод о наличии или отсутствии деструктивных свойств ПО, а также позволяющая анализировать процесс выполнения ПО и его инструкции, представленная в терминах теории графов и теории множеств.

2. Создано формальное описание классифицирующего признака программного обеспечения, основанное на синтезированной модели представления ПО, обеспечивающее отсутствие возможности пропуска ПО, обладающего известным деструктивным свойством, а также позволяющее формально характеризовать деструктивные свойства ПО.

3. Предложен алгоритм классификации ПО, позволяющий автоматизировать разработанную методику, использующий подход к оценке подобия различных экземпляров ПО, основанный на мере Дамерау-Левенштейна.

4. Синтезирована методика верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений, не требующая постоянного использования ресурсов среды ОВ, а также позволяющая выявлять новые и модифицированные экземпляры ПО, обладающего деструктивными свойствами.

5. Построена архитектура программного комплекса, реализующего методику верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений. На основании архитектуры разработан программный комплекс, повышающий защищённость сред ОВ с использованием предложенной в работе методики.

6. Предложенная методика верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений реализована в Центре вирусных исследований и аналитики «Eset» для осуществления формального вывода при анализе ПО о наличии в нём деструктивных свойств ПО.

7. Предложенная методика верификации ПО на наличие деструктивных свойств для сред облачных вычислений реализована в компании ООО «Связьмонтажкомплектация» для повышения качества предоставляемых компанией услуг по тестированию ПО для сред ОВ.

8. Практические и теоретические результаты реализованы в компании ООО «ТСС». Так, модель описания верифицируемого программного обеспечения и модель описания деструктивного свойства ПО получили применение при создании моделей информационных систем, что позволило автоматизировать и формализовать процесс оценки рисков систем информационной безопасности

9. Теоретические и прикладные результаты, полученные в ходе выполнения диссертационной работы, использованы в учебном курсе «Языки программирования» кафедры «Криптология и дискретная математика» НИЯУ МИФИ при создании лабораторных работ по курсу «Языки программирования 2» на базе алгоритмов и методов, использованных в методике верификации программного обеспечения на наличие деструктивных свойств для сред облачных вычислений. Разработанные лабораторные работы позволяют слушателям курса получить знания об особенностях программного обеспечения, создаваемого для выполнения в средах ОВ, и о существующих методиках верификации ПО.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

1. **Туманов Ю.М. Обнаружение вредоносных сценариев JavaScript на основе поведенческих сигнатур. //Безопасность информационных технологий 2009. №4. С.63-65**
2. **Филоненко А. В., Исаев И. К., Сидоров Д. В., Туманов Ю. М.. Поиск уязвимостей по бинарному коду с помощью проверки выполнимости ограничений. //Безопасность информационных технологий 2010. №2. С.83-86**
3. **Гаврилюк С. В., Туманов Ю. М. Разработка метода защиты вычислительных Грид-сетей от намеренного искажения результата вычислений //Безопасность информационных технологий 2010. №1. С.53-54**
4. **Варфоломеев А. А., Коренева А. М., Краснопевцев А. А., Туманов Ю. М., Фомичев В. М. «О реализации метода полного опробования ключей криптосистем в условиях различных математических распределенных вычислений» //Безопасность информационных технологий 2011. №1. С.82-83**
5. **Станкевичус А. А., Туманов Ю. М. –Разработка системы обнаружения вредоносных сценариев JavaScript на основе поведенческих сигнатур //Бизнес и Безопасность в России 2009 июнь С.130-131**
6. **Гаврилюк С. В., Туманов Ю. М. Использование поведенческого анализа с применением поведенческих сигнатур для выявления вредоносного кода на примере JavaScript сценариев //Безопасность информационных технологий 2010. №1. С.115-117**
7. **Моисеев А.В. Станкевичус А. А., Туманов Ю.М. «Защита среды распределённых вычислений при помощи искусственных иммунных систем» //Безопасность информационных технологий 2011. №4. С.103-105**
8. Туманов Ю. М. Выявление интерпретируемого вредоносного кода на основе поведенческих сигнатур. – Научная сессия НИЯУ МИФИ-2011. Аннотации докладов в 3 томах. Т. 3. М.:НИЯУ МИФИ, 2010. – с. 165.
9. Yury Tumanov «Behavioral detection of malicious interpreted code» [Электронный ресурс]: Microsoft Research Summer School 2011. Режим доступа к ресурсу: http://blogs.msdn.com/b/msr_er/archive/2011/07/12/top-students-descend-on-microsoft-research-cambridge.aspx
10. Туманов Ю.М. «Использование поведенческого анализа для выявления вредоносного кода, на примере JavaScript сценариев» – сборник аннотаций работ «7-я курчатовская молодежная школа» 2009г. С.157
11. Туманов Ю.М., «Разработка программы обнаружения вредоносных сценариев JavaScript на основе поведенческих сигнатур» [Электронный ресурс] «РусКрипто’2009». Режим доступа к ресурсу: http://www.ruscrypto.org/sources/conference/rc2009/