На правах рукописи

Станкевичус Алексей Алексеевич

МЕТОДИКА ЗАЩИТЫ ГЕТЕРОГЕННОЙ СРЕДЫ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ ОТ ВРЕДОНОСНОГО КОДА ВЫПОЛНЯЕМОГО ЗАДАНИЯ

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

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Автор: _____

Москва – 2009

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

Научный руководитель: кандидат технических наук, доцент
Петрова Тамара Васильевна

Официальные оппоненты: доктор технических наук, профессор
Оныкий Борис Николаевич

кандидат технических наук
Смирнов Павел Владимирович

Ведущая организация: Институт проблем информатики
Российской академии наук (ИПИ РАН)

Защита состоится «28» октября 2009 г. в 15 часов 00 минут на заседании диссертационного совета ДМ 212.130.08 в Центре информационных технологий и систем органов исполнительной власти по адресу: 123557, г. Москва, Пресненский Вал, д. 19. Тел. для справок: +7 (495) 323-95-26, 324-84-98.

С диссертацией можно ознакомиться в библиотеке Московского инженерно-физического института (государственного университета).

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

Автореферат разослан «___» сентября 2009 г.

Ученый секретарь
диссертационного совета Горбатов В.С.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. В настоящее время существует множество задач, для решения которых необходимы вычислительные мощности, недоступные в рамках отдельных компьютерных центров. Для решения таких задач были разработаны гетерогенные среды распределенных вычислений. Гетерогенная среда распределенных вычислений (далее СРВ) представляет собой географически распределенную инфраструктуру, в состав которой входит множество разнотипных ресурсов (таких как запоминающие устройства и процессоры, базы данных и сети) доступ к которым обеспечивается пользователю независимо от месторасположения пользователя и этих ресурсов. Для обозначения гетерогенной среды распределенных вычислений часто используется термин «Грид», образованный от английского Grid (решетка, сетка), при этом подчеркивается аналогия с сетью электропитания – подразумевается, что получать доступ к ресурсам гетерогенной среды распределенных вычислений почти так же легко и удобно, необходимо только «вставить штепсель в розетку».

Европейский Центр Ядерных Исследований (ЦЕРН) одним из первых начал использовать технологию Грид. Грид был необходим для обработки данных, поступающих от крупнейшего в мире ускорителя заряженных частиц – Большого Адронного Коллайдера (Large Hardon Collider, LHC). В настоящее время СРВ используются для решения множества задач по самым разным фундаментальным и прикладным направлениям – в физике высоких энергий и космофизике, микробиологии и медицине, метеорологии, самолетостроении и целом ряде других областей.

Пользователю СРВ может предоставляться доступ к большому количеству различных ресурсов, причем каждый из этих ресурсов может использоваться большим количеством пользователей, обрабатывающих различные данные. В случае, когда нарушитель получает доступ к СРВ, его действия могут повлиять на работу большого числа ресурсов и пользователей СРВ, при этом нарушитель может нанести колоссальный ущерб и ресурсам, и пользователям. Проблема безопасности гетерогенной среды распределенных вычислений очень остра. При распространении нарушителем вредоносного кода в составе вычислительного задания может быть нарушена работа отдельных вычислительных узлов и СРВ в целом. Кроме того, СРВ может быть использована как платформа для реализации распределенных атак.

На данный момент во многих гетерогенных средах распределенных вычислений используется инфраструктура открытых ключей для надежной аутентификации пользователей и производителей вычислительных заданий, что позволяет в случае распространения ими вредоносного кода использовать административные меры наказания. Однако такие методы не упреждают, а лишь констатируют факт реализации угрозы, что в условиях СРВ может быть сопряжено с колоссальным ущербом. Для предотвращения реализации угроз могут быть применены различные технические средства защиты от вредоносного кода, из которых наименьшее влияние на производительность кода оказывает его верификация.

Вычислительные задания для СРВ разрабатывают на различных языках программирования, таких как Си, C#, Java, Python, из которых язык Си является наиболее подходящим для применения в гетерогенной среде. Язык программирования Си является широко распространенным. В отличие от C#, для языка Си существуют компиляторы для большинства аппаратных платформ. Кроме того, производительность

программ на языке Си во многих случаях превышает производительность аналогичных программ на других языках программирования, в том числе на Java и Python, что также делает этот язык удобным для разработки вычислительных заданий СРВ.

В этих условиях актуальной является задача разработки методики защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания, основанной на автоматической верификации безопасности исходного кода. Создание такой методики позволит повысить уровень безопасности существующих гетерогенных сред распределенных вычислений, а также может обеспечить возможность применения СРВ для решения новых задач в ранее не допускавших применения СРВ условиях.

Объектом исследования диссертационной работы является гетерогенная среда распределенных вычислений.

Предметом исследования диссертационной работы является безопасность гетерогенной среды распределенных вычислений.

Целью диссертационной работы является разработка методики защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания и разработка на основе этой методики средства, обеспечивающего возможность безопасного применения в гетерогенной среде распределенных вычислений разработанных на языке Си вычислительных заданий из недоверенных источников.

В соответствии с поставленной целью в диссертационной работе **решаются следующие задачи**:

– анализ существующих методов защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания;

– построение модели нарушителя и анализ угроз;

– разработка методики защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания;

– разработка алгоритма проверки безопасности кода программ;

– разработка архитектуры системы защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания;

– разработка программного обеспечения системы защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания.

Основными **методами исследований**, используемыми в работе, являются методы структурного и функционального анализа, теории формальных грамматик и языков, теории доменов, теории автоматов, теории графов, теории алгоритмов.

Научная новизна работы характеризуется следующими результатами:

– построена модель нарушителя и сформулирован перечень угроз гетерогенной среде распределенных вычислений;

– разработана методика защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания;

– разработан алгоритм проверки безопасности кода программ;

– разработана архитектура системы защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания.

Практическая значимость результатов определяется следующим:

– подготовлены рекомендации по практическому применению разработанной методики защиты гетерогенной среды распределенных вычислений от вредоносного

кода выполняемого задания;

– разработано программное обеспечение системы защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания.

Результаты работы представляют практическую ценность для обеспечения безопасности информации, обрабатываемой на узлах гетерогенной среды распределенных вычислений, в первую очередь, целостности и конфиденциальности информации в ОЗУ ЭВМ, а также позволяют защитить диспетчер СРВ от атак типа «отказ в обслуживании» со стороны вредоносного вычислительного задания.

Внедрение результатов исследований. Основные результаты исследований используются в АОЗТ «Виланд» для обеспечения возможности применения в гетерогенной среде распределенных вычислений ЭВМ, предназначенных для обработки конфиденциальной информации, во время их простоя.

Результаты диссертационной работы внедрены в учебный процесс на факультете «Информационная безопасность» Московского инженерно-физического института (государственного университета).

Публикации и апробация работы. Результаты диссертации изложены в 8 публикациях и докладывались на конференциях и семинарах различного уровня:

– на Всероссийской конференции студентов, аспирантов и молодых ученых «Технологии Microsoft в теории и практике программирования» в 2005 г;

– на XII Всероссийской научной конференции «Проблемы информационной безопасности в системе высшей школы» в 2006 г;

– на XIV Всероссийской научной конференции «Проблемы информационной безопасности в системе высшей школы» в 2007 г;

– на IV Всероссийской конференции студентов, аспирантов и молодых учёных «Технологии Microsoft в теории и практике программирования» в 2007 г;

– на V Всероссийской конференции студентов, аспирантов и молодых ученых «Технологии Microsoft в теории и практике программирования» в 2008 г;

– на XVII Международном научно-техническом семинаре «Современные технологии в задачах управления, автоматики и обработки информации» в 2008 г;

– на XVI Всероссийской конференции «Проблемы информационной безопасности в системе высшей школы» в 2009 г.

Основные положения, выносимые на защиту:

– результаты анализа методов защиты гетерогенной среды распределенных вычислений от вредоносного кода;

– построенная модель нарушителя и сформулированный перечень угроз;

– алгоритм проверки безопасности исходного кода программ;

– методика защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания;

– архитектура системы защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания;

– программный комплекс, позволяющий обеспечить защиту гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания, написанного на языке Си.

Структура работы. Работа состоит из введения, пяти глав, заключения, списка литературы, включающего 131 наименование, и трех приложений. Текст диссертации

изложен на 140 страницах, включая 35 рисунков и 10 таблиц.

СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обосновывается актуальность темы диссертации, выделяются и формулируются цели и задачи исследования, описывается структурно-логическая схема диссертационной работы.

В **первой главе** проводится анализ особенностей СРВ с точки зрения защиты от вредоносного кода выполняемого задания, анализ известных методов защиты СРВ от вредоносного кода выполняемого задания, проводится анализ подходов к построению верификаторов исходного кода программ.

В данной работе рассматривается защита сред распределенных вычислений, обладающих целым рядом особенностей:

– в вычислениях участвует большое количество вычислительных узлов;

– используется медленный канал связи вычислительных узлов с диспетчером СРВ;

– СРВ предназначена для решения задач, требующих проведения большого количества вычислений при относительно небольшом размере входных данных и результата, передаваемого на диспетчер СРВ;

– СРВ предназначена для достижения как можно большей производительности вычислений;

– среда исполнения вычислительных заданий гетерогенна программно;

– среда исполнения вычислительных заданий гетерогенна аппаратно;

– вычислительные задания не взаимодействуют с пользователями ЭВМ, на которых они исполняются.

– вычислительные задания не взаимодействуют с внешними устройствами.

Предназначенное для выполнения в СРВ вычислительное задание не должно быть привязано к конкретному виду техники или программному интерфейсу операционной системы и должно быть переносимо с одной платформы на другие. Данные для обработки поставляются выполняемому заданию гетерогенной среды распределенных вычислений от диспетчера СРВ. Результаты вычислений выполняемое в СРВ задание передает диспетчеру СРВ. В процессе обработки данных выполняемое задание СРВ не требует взаимодействия с пользователем ЭВМ, на которой происходит его выполнение. Как правило, для обработки данных не требуется взаимодействия с какими-либо внешними устройствами.

Рассматриваются различные способы защиты СРВ от вредоносного кода вычислительного задания:

– виртуализация, при которой выполнение программного обеспечения (далее ПО) происходит в виртуальной среде, доступ из которой к реальному программному и аппаратному обеспечению ограничен;

– системы безопасного программирования, при применении которых разработка ПО ведется на специально разработанных языках, не позволяющих нарушить безопасность системы;

– верификация, суть которой заключается в том, что перед применением ПО производится проверка его безопасности;

– подпись ПО, при которой автор ПО подписывает его своей ЭЦП, после чего в случае выявления вредоносного характера ПО, к автору применяются нормативно-

правовые санкции;

– анализ сетевой активности, при котором ведется наблюдение за сетевой активностью участников СРВ, что позволяет предотвратить некоторые сетевые атаки.

Сделан вывод о том, что рассмотренные в работе способы защиты от вредоносного кода обладают целым рядом недостатков:

– падение производительности (полная эмуляция, системы безопасного программирования, трансляция в безопасное ПО, использование трассирующих мониторов);

– сложность применения (верификация, проводимая с участием человека, использование содержащего доказательство кода, использование содержащего модель кода, верификация исполнимого кода);

– повышение требований к аппаратному обеспечению и операционной системе (далее ОС) и, как следствие, сложность применения в гетерогенной среде (виртуализация уровня приложений, виртуализация уровня ОС, паравиртуализация, частичная виртуализация, частичная эмуляция);

– неспособность обеспечить всестороннюю защиту от вредоносного ПО (системы безопасного программирования, защита от переполнения буфера, защита от атак «отказ в обслуживании», подпись ПО);

– ограничение круга средств разработки (системы безопасного программирования, верификация исходного кода).

Кроме того, многие рассмотренные способы не учитывают специфики СРВ, что ведет к снижению уровня безопасности или применению излишне жесткой политики безопасности, а также к возникновению избыточной дополнительной нагрузки.

Перспективным подходом представляется применение автоматической статической верификации на уровне исходных кодов, так как верификация сама по себе не влияет на производительность ПО, и при этом сохраняется возможность применения ПО в гетерогенной среде. Кроме того, для защиты от атак «отказ в обслуживании» необходимо использовать дополнительные средства, обеспечивающие контроль интенсивности сетевого взаимодействия.

Вторая глава посвящена построению методики защиты СРВ от вредоносного кода выполняемого задания. Проводится построение модели нарушителя, анализ угроз, формулируются требования к безопасному ПО, производится построение модели памяти и основных алгоритмов защиты от вредоносного ПО.

Методика защиты СРВ от вредоносного кода выполняемого задания состоит в определении модели нарушителя и перечня угроз, получении перечня мер по предотвращению угроз, формулировании требований к безопасности кода, автоматизации построения формальной модели кода и верификации на соответствие определенным ранее требованиям и проведении верификации безопасности выполняемого кода на вычислительных узлах.

В данной работе предполагается, что владельцы вычислительных узлов, а также пользователи и производители вычислительных заданий доверяют производителю системного ПО СРВ и владельцу диспетчера СРВ, и существуют взаимные отношения доверия между производителем системного программного обеспечения СРВ и владельцем диспетчера СРВ.

В таблице 1 приведено описание возможностей рассматриваемых в данной рабо-

те нарушителей безопасности СРВ.

Таблица 1 – Описание возможностей нарушителей

Идентификатор нарушителя	Наименование Нарушителя	Описание возможностей нарушителя
А	Злонамеренный пользователь вычислительного задания	Может предоставлять данные для обработки в СРВ при помощи существующего вычислительного задания
Б	Злонамеренный разработчик вычислительного задания	Может разрабатывать вычислительные задания, используемые другими участниками СРВ
В	Злонамеренный разработчик вычислительного задания в сговоре с злонамеренным пользователем вычислительного задания	Может разрабатывать вычислительное задание, используемое другими участниками СРВ. Может использовать вычислительные задания, создаваемые различными участниками СРВ для обработки своих данных.

В таблице 2 приведены угрозы, исходящие от нарушителей.

Таблица 2 – Угрозы, исходящие от нарушителей

Наименование угрозы	Описание угрозы	Идентификатор нарушителя
1	2	3
Угроза целостности узла СРВ	Осуществление программного воздействия, влекущего выход компонентов ЭВМ из строя.	А, Б, В
Угроза целостности обрабатываемой на узле СРВ информации	Модификация или удаление информации в ОЗУ ЭВМ, на жестком диске или на ином носителе	А, Б, В
Угроза целостности вычислительного задания	Нарушение целостности вычислительного задания в ОЗУ ЭВМ путем предоставления реализующих эксплуатацию уязвимости данных для обработки	А, В
Угроза доступности диспетчера СРВ	Реализация атаки «отказ в обслуживании» или «распределенный отказ в обслуживании» на диспетчер СРВ	А, Б, В
Угроза доступности узла СРВ	Реализация атаки «отказ в обслуживании» на узел СРВ или перевод узла СРВ в нештатное состояние	А, Б, В
Угроза доступности обрабатываемой на узле СРВ информации	Программное воздействие, влекущее зашифрование или сокрытие информации, обрабатываемой на ЭВМ	А, Б, В

1	2	3
Угроза конфиденциальности обрабатываемой на узле СРВ информации	Программное воздействие, влекущее компрометацию конфиденциальной информации, обрабатываемой на узле СРВ путем передачи ее по сети	А, Б, В
Угроза конфиденциальности предоставляемой пользователем вычислительного задания информации	Разработка вычислительного задания, осуществляющего компрометацию предоставляемой пользователем вычислительного задания информации путем передачи ее по сети	Б, В

В работе предлагается набор мер по предотвращению выявленных угроз. Меры по предотвращению угроз приведены в таблице 3.

Таблица 3 – Меры по предотвращению угроз

Наименование угрозы	Меры по предотвращению
1	2
Угроза целостности узла СРВ	Запрет на использование в вычислительном задании инструкций и программных интерфейсов, способных нарушить целостность компонентов ЭВМ. Верификация ПО на отсутствие уязвимостей, способных привести к выполнению произвольного кода.
Угроза целостности обрабатываемой на узле СРВ информации	Запрет на работу с невыделенными областями ОЗУ. Запрет на непосредственное обращение к информации на жестком диске и иных носителях. Верификация ПО на отсутствие уязвимостей, способных привести к выполнению произвольного кода.
Угроза целостности вычислительного задания	Верификация ПО на отсутствие уязвимостей, способных привести к выполнению произвольного кода.
Угроза доступности диспетчера СРВ	Запрет на использование команд и программных интерфейсов, обеспечивающих неконтролируемое сетевое взаимодействие. Организация специального программного интерфейса, обеспечивающего возможность безопасного обмена данными с диспетчером СРВ. Верификация ПО на отсутствие уязвимостей, способных привести к выполнению произвольного кода.
Угроза доступности узла СРВ	Запрет на использование команд и программных интерфейсов, обеспечивающих неконтролируемое сетевое взаимодействие. Запрет на использование команд и программных интерфейсов, способных перевести ЭВМ в нештатное состояние. Верификация ПО на отсутствие уязвимостей, способных привести к выполнению произвольного кода.

1	2
Угроза доступности обрабатываемой на узле СРВ информации	Запрет на доступ к обрабатываемой на ЭВМ информации. Верификация ПО на отсутствие уязвимостей, способных привести к выполнению произвольного кода.
Угроза конфиденциальности обрабатываемой на узле СРВ информации	Запрет на чтение неинициализированных областей ОЗУ. Запрет на непосредственное обращение к информации на жестком диске и иных носителях. Верификация ПО на отсутствие уязвимостей, способных привести к выполнению произвольного кода.
Угроза конфиденциальности предоставляемой пользователем вычислительного задания информации	Запрет на использование команд и программных интерфейсов, обеспечивающих неконтролируемое сетевое взаимодействие. Организация специального программного интерфейса, обеспечивающего возможность безопасного обмена данными с диспетчером СРВ. Верификация ПО на отсутствие уязвимостей, способных привести к выполнению произвольного кода.

Безопасное программное обеспечение не должно нарушать политики безопасности системы, на которой производится его исполнение. В СРВ вычислительное задание должно производить преобразование входных данных и выработку выходных данных в соответствии с некоторым алгоритмом, реализуемым этим ПО. Это означает, что ПО должно использовать только определенные способы взаимодействия с аппаратным обеспечением и операционной системой, исключающие возможность получения доступа к данным, хранимым или обрабатываемым данной системой, а также возможность нарушения штатного режима функционирования системы или СРВ в целом.

Таким образом, при проверке безопасности ПО необходимо убедиться, что исходный код не содержит запрещенных политикой безопасности прямых обращений к операционной системе, не содержит запрещенных политикой безопасности обращений к областям памяти, которые не были ранее выделены этим ПО для работы, а также не содержит способного к самомодификации кода.

То есть, для защиты от вредоносного ПО необходимо обеспечить контроль диапазонов индексов при обращении к массивам, контроль диапазонов адресов при обращении к динамически выделяемой памяти, контроль значений указателей при обращении к данным и вызове функций через указатель, контроль преобразования типов указателей, убедиться в отсутствии вызовов функций операционной системы и сторонних библиотек, а также в отсутствии вставок на языке ассемблера, а кроме этого обеспечить контроль сетевого трафика.

Автором данной работы построена методика защиты СРВ от вредоносного кода выполняемого задания, суть которой заключается в проведении верификации соответствия выполняемого задания ряду требований, при удовлетворении которым выполняемое задание может считаться безопасным для СРВ.

В данной работе рассматривается именно гетерогенная СРВ, ее вычислительные узлы могут функционировать на различных аппаратных и программных платформах и под управлением различных ОС. Для различных вычислительных узлов владельцам

этих узлов необходимо выработать политику безопасности, учитывающую специфику применяемого аппаратного и программного обеспечения. Вредоносным программным обеспечением для данного вычислительного узла СРВ будем считать ПО, нарушающее выработанную для этого узла политику безопасности.

Для выявления вредоносного ПО автором данной работы разработан синтетический метод верификации, сочетающий несколько разнотипных техник: статический анализ формальных свойств и мониторинг формальных свойств.

Статический анализ формальных свойств заключается в проверке свойств автоматически вырабатываемой на основе исходного кода модели. Для построения модели предлагается использовать абстрактную интерпретацию исходного кода программы. Проверять необходимо невозможность нарушения верифицируемым ПО при произвольных входных данных требований безопасности, определенных для вычислительного задания СРВ. Вычислительное задание переводится в представление в виде направленного графа, узлы которого соответствуют линейным участкам программы, а дуги – ветвлениям потока управления. Переменные языка программирования представляются в виде описания диапазона вероятных значений, которые данная переменная может принимать при различных путях исполнения и различных входных данных. Все переменные, используемые в каждом из линейных участков программы, предполагаются способными иметь произвольные значения. Далее над диапазонами вероятных значений производятся операции, соответствующие операциям, выполняемым над переменными программы. В случае невыполнения требований безопасности на каком-либо участке, участок помечается как потенциально опасный. Далее, рассматриваются все дуги графа, ведущие в потенциально опасные участки, и производится уточнение диапазонов вероятных значений переменных на входе каждого из таких потенциально опасных участков. В случае, когда в результате такого уточнения потенциально опасный участок остается потенциально опасным, производится рассмотрение всех путей в графе длины 2, 3 и т.д., ведущих в данный узел и процесс повторяется. При этом завершение процесса производится в случае достижения заданной глубины анализа, либо при превышении заданного времени, отводимого на проверку.

Когда абстрактная интерпретация не позволяет достоверно установить безопасность или небезопасность участка ПО, предлагается применять мониторинг формальных свойств ПО. В основу мониторинга формальных свойств ПО положено выделение требующих верификации формальных свойств ПО и встраивание проверки этих свойств в систему мониторинга. При исполнении ПО следует проводить постоянный мониторинг, целью которого является контроль выделенных свойств. В случае выявления системой мониторинга попытки нарушения политики безопасности, система мониторинга останавливает исполнение ПО, предотвращая тем самым выявленную попытку нарушения политики безопасности. В результате вычислительное задание со встроенной в него системой мониторинга становится безопасным.

В работе вводится абстрактный домен двоичных n-мерных векторов. Этот домен позволяет проводить абстрактную интерпретацию программ, содержащих такие операции над целочисленными переменными как побитовый сдвиг, побитовое исключающее ИЛИ, побитовое НЕ, побитовое ИЛИ, побитовое И. За счет применения этого абстрактного домена становится возможной верификация ПО, содержащего побитовые операции. Предлагаемый домен построен как декартово произведение n абстракт-

ных доменов двоичных переменных, он обозначается X^{An} и имеет следующий вид:

$$X^{An} = \{x | x = \langle x_1, x_2, \dots, x_n \rangle, \text{где } x_i \in \{\bot, 0, 1, \top\}, i = 1,2, \dots, \mathrm{n}\}. \quad (1)$$

Под $\bot$ и $\top$ понимаются «дно» и «вершина» абстрактного домена двоичных переменных, то есть его наименьший и наибольший элементы, соответственно. Отношение порядка в X^{An} обозначим как $\leq$ и зададим следующим образом:

$$\forall A \in X^{An}, \forall B \in X^{An}, \mathrm{A} = \langle \mathrm{a}_1, \mathrm{a}_2, \dots, \mathrm{a}_\mathrm{n} \rangle, \mathrm{B} = \langle \mathrm{b}_1, \mathrm{b}_2, \dots, \mathrm{b}_\mathrm{n} \rangle \quad (2)$$

$$\mathrm{A} \leq \mathrm{B} \Leftrightarrow a_i \leq b_i \; \forall i \in \{1,2, \dots, n\}, \quad (3)$$

$$\text{где } a_i \leq b_i \text{ задается как}$$

$$\bot \leq 0 \leq \top, \bot \leq 1 \leq \top, \forall n \in \{\bot, 0, 1, \top\} \; n \leq n. \quad (4)$$

Операция абстракции $f_a: 2^{V_n} \to X^{An}$ задается следующим образом:

$$\forall C \in 2^{V_n}, A = f_a(C) \Leftrightarrow A = \langle a_1, a_2, \dots, a_n \rangle, \forall i \in \{1,2, \dots, n\}, \quad (5)$$

$$a_i = \begin{cases} 0, \text{если } \forall c \in C, c = \langle c_1, c_2, \dots, c_n \rangle \; c_i = 0, \\ 1, \text{если } \forall c \in C, c = \langle c_1, c_2, \dots, c_n \rangle \; c_i = 1, \\ \top, \text{если } \exists b, c \in C, b = \langle b_1, b_2, \dots, b_n \rangle, c = \langle c_1, c_2, \dots, c_n \rangle : b_i \neq c_i, \\ \bot, \text{если } C = \emptyset. \end{cases} \quad (6)$$

Операция конкретизации $f_c: X^{An} \to 2^{V_n}$ задается следующим образом:

$$\forall A \in X^{An}, A = \langle a_1, a_2, \dots, a_n \rangle, \forall c \in V_n, c = \langle c_1, c_2, \dots, c_n \rangle, \quad (7)$$

$$c \in f_c(A) \Leftrightarrow \forall i \in \{0,1, \dots, n\} \; c_i \in \begin{cases} \{0\}, \text{если } a_i = 0, \\ \{1\}, \text{если } a_i = 1, \\ \{0,1\}, \text{если } a_i \in \{\bot, \top\}. \end{cases} \quad (8)$$

Заметим, что $C \subseteq f_c(f_a(C))$ для любого $C \in 2^{V_n}$, то есть при использовании абстракции не происходит потери конкретных значений.

На рисунке 1 приведена построенная автором данной работы модель ячейки памяти, не требующей инициализации.

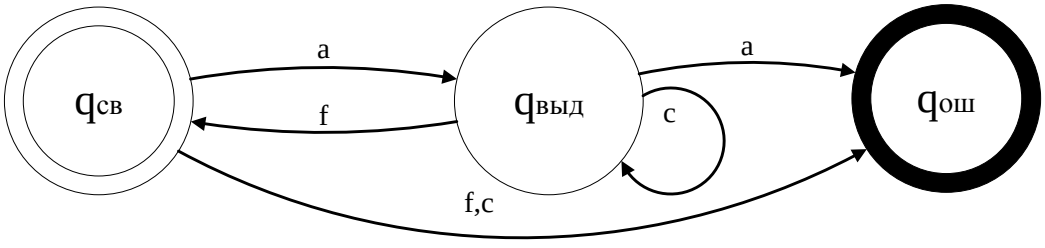

Рисунок 1 – Модель ячейки памяти, не требующей инициализации

Ячейка может находиться в одном из трех состояний: $q_{св}$ – ячейка свободна, $q_{выд}$ – ячейка выделена, $q_{ош}$ - ошибка. Начальным состоянием является $q_{св}$. При переходе в состояние $q_{ош}$ происходит остановка работы модели. Переходы в модели помечены как *a* - выделение, *f* - освобождение, *c* – обращение к данным в ячейке. В зависимости от того, требуется ли обеспечить защиту произвольных областей памяти от чтения, в данной модели под обращением понимается только запись или любой вид обращения (чтение или запись) соответственно для случаев, когда защита не требуется и требуется.

На рисунке 2 приведена построенная автором данной работы модель ячейки памяти, требующей инициализации. Ячейка может находиться в одном из четырех состояний: $q_{св}$ – ячейка свободна, $q_{выд}$ – ячейка выделена, но не инициализирована, $q_{ин}$ – ячейка выделена и инициализирована, $q_{ош}$ - ошибка. Начальным состоянием является $q_{св}$. При переходе в состояние $q_{ош}$ происходит остановка работы модели. Переходы в модели помечены как *a* - выделение, *f* - освобождение, *r* – чтение данных из ячейки, и *w* – запись данных в ячейку.

Утверждение. Модель ячейки памяти, требующей инициализации сводится к модели ячейки памяти, не требующей инициализации, если под операцией *a* первой модели понимать последовательность операций *a, w* второй модели, выполняемых подряд.

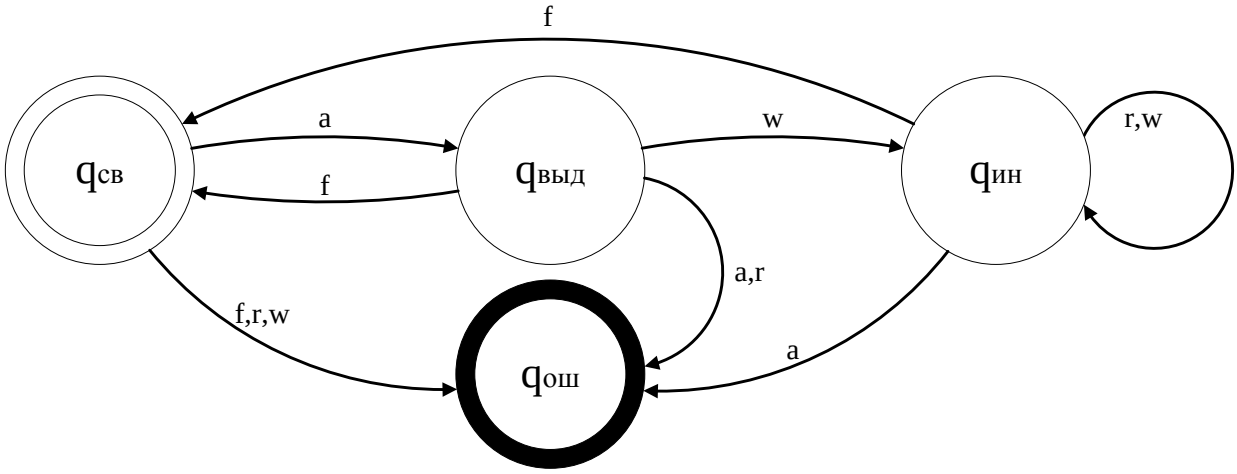

Рисунок 2 – Модель ячейки памяти, требующей инициализации

При статическом анализе формальных свойств и мониторинге формальных свойств вычислительного задания необходимо осуществлять контроль соответствия поведения вычислительного задания модели ячейки памяти, отражающей требования политики безопасности.

При взаимодействии вычислительного задания с диспетчером СРВ необходимо предотвращать реализацию атак «отказ в обслуживании» на диспетчер СРВ.

Введем следующие обозначения: V – пропускная способность канала или системы обработки данных диспетчера СРВ, U – скорость передачи данных вычислительным узлом, N – общее количество вычислительных узлов СРВ, W – общий объем передаваемых в единицу времени диспетчеру СРВ всеми вычислительными узлами данных.

Атаку «отказ в обслуживании» следует считать успешно реализованной, когда W значительно превышает V. Атака «отказ в обслуживании» не может быть реализована,

если для каждого вычислительного узла U<V/N, так как при этом W < V.

Проводимый и на диспетчере СРВ и на вычислительных узлах с целью предотвращения передачи данных со скоростью выше U мониторинг сетевого взаимодействия может позволить защитить диспетчер СРВ от атак «отказ в обслуживании» со стороны вредоносного вычислительного задания.

В **третьей главе** производится построение концептуальной и функциональной моделей системы, предлагается архитектура системы защиты СРВ от вредоносного ПО, проводится детальное проектирование составляющих системы.

С учетом особенностей защищаемой среды распределенных вычислений, приведем требования, которые предъявляются к системе защиты СРВ от вредоносного кода выполняемого задания. Система защиты должна:

– выявлять вредоносный код и не допускать его исполнение;

– выявлять попытки реализации атак типа «отказ в обслуживании» на диспетчер СРВ и предотвращать их;

– функционировать в гетерогенной аппаратной и программной среде;

– обеспечивать по возможности низкую дополнительную нагрузку на узлы СРВ в процессе вычислений;

– функционировать в автоматическом режиме.

Система защиты СРВ от вредоносного ПО состоит из безопасной библиотеки, модуля внедрения динамической верификации и верификатора исходного кода, кроме того, предусмотрен интерфейс разработчика программного обеспечения. На вход системы поступает исходный код вычислительного задания на языке Си. В автоматическом режиме производится проверка безопасности вычислительного задания, включающая при необходимости преобразование потенциально опасных участков кода в код, использующий безопасную библиотеку. Алгоритм проверки безопасности кода программ приведен на рисунке 3. Алгоритм подразумевает проведение абстрактной интерпретации описанным во второй главе способом.

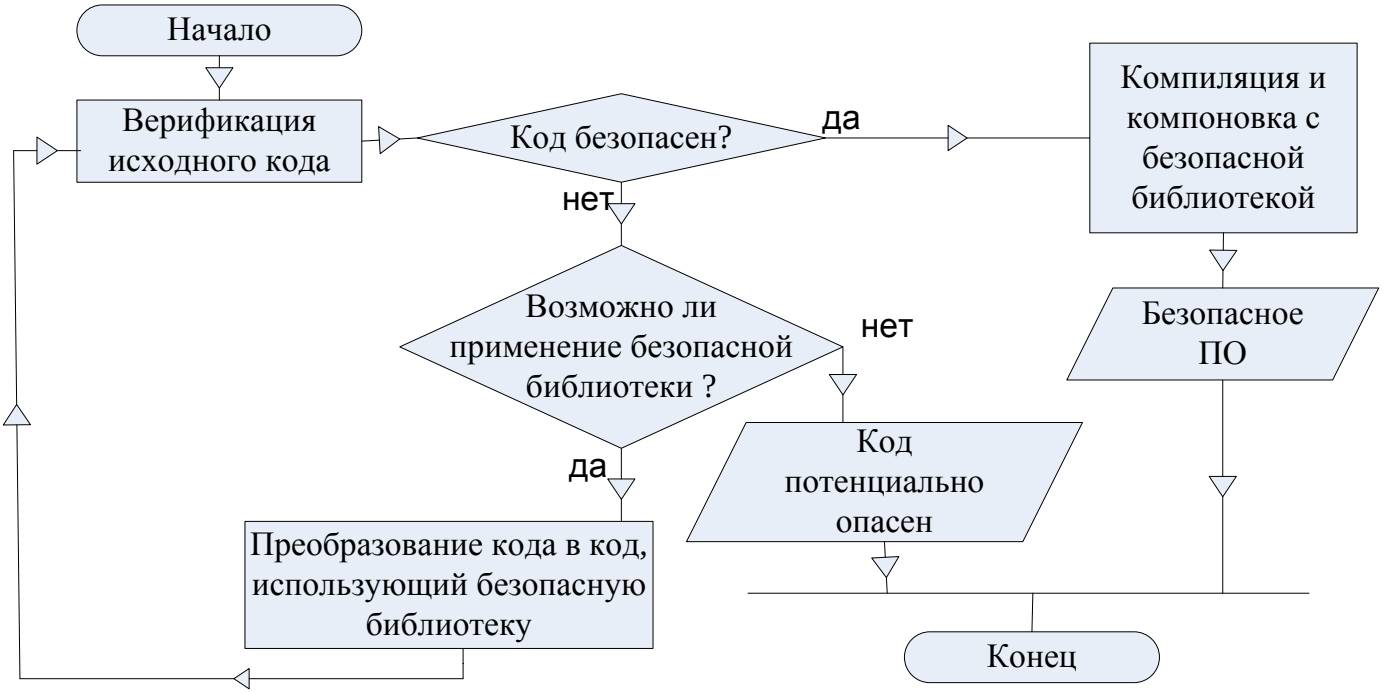

Рисунок 3 - Алгоритм проверки безопасности кода программ

Преобразование потенциально опасного кода в код, использующий безопасную библиотеку, производится в полностью автоматическом режиме путем замены потенциально опасных операций на вызов функций монитора обращений, обеспечивающего безопасность выполнения этих операций. На рисунке 4 приведена общая схема взаимодействия вычислительного задания с безопасной библиотекой.

Рисунок 4 – Общая схема взаимодействия с безопасной библиотекой

Безопасная библиотека содержит монитор обращений и набор функций для работы с оперативной памятью и осуществления ввода-вывода. Безопасность взаимодействия вычислительного задания с общим пулом памяти верифицируется статически. Безопасность взаимодействия с безопасным пулом памяти обеспечивается монитором обращений, проводится валидация всех используемых при взаимодействии с безопасным пулом параметров. Безопасность взаимодействия со службами СРВ и ОС обеспечивается монитором обращений аналогичным образом.

В состав политики безопасности входят следующие параметры:

– необходимость обнуления выделяемой памяти,
– необходимость защиты от чтения произвольных областей памяти,
– необходимость защиты от вызова произвольных функций ОС,
– необходимость защиты от вызова произвольных ассемблерных команд,
– разрешенные имена и режим доступа к файлам ОС,
– необходимость ведения аудита действий ПО,
– необходимость защиты от атак «отказ в обслуживании» на диспетчер СРВ.

Действующая политика безопасности определяется на основе политик безопасности, задаваемых владельцем диспетчера СРВ и владельцем вычислительного узла путем выбора более строгих ограничений из задаваемых в каждой политике.

При использовании различных платформ СРВ и ОС требуется реализация версий монитора обращений для обеспечения поддержки безопасного взаимодействия со службами каждой из платформ СРВ и ОС, при этом остальная часть системы остается неизменной.

В **четвертой главе** производится выбор технологий для построения системы защиты СРВ от вредоносного кода выполняемого задания, проводится разработка структур данных, рассматривается реализация системы защиты СРВ от вредоносного

кода выполняемого задания, проводится оценка производительности реализованной системы защиты СРВ от вредоносного кода выполняемого задания.

Для реализации системы защиты СРВ от вредоносного кода выполняемого задания был выбран язык программирования Си. При реализации верификатора использовался генератор лексических анализаторов Flex и генератор синтаксических анализаторов Bison. В качестве препроцессора языка Си принято решение использовать переносимый препроцессор MCPP (также написанный на языке Си и распространяемый в виде исходных кодов). Кроме того, использовался язык C#, на котором производилась реализация интерфейса разработчика вычислительного задания.

При реализации системы для представления информации о выделенной памяти, для представления формальных описаний функций безопасной библиотеки и для представления элементов абстрактных доменов были разработаны и реализованы специальные структуры данных.

Было проведено две фазы тестирования: модульное тестирование и системное тестирование. Тестирование позволило контролировать полноту функциональности системы, отсутствие дефектов.

Произведена оценка производительности безопасного кода, полученного после обработки потенциально вредоносного исходного кода разработанной системой защиты. Для оценки производительности использовались как специально разработанные автором тесты, так и существующие свободно распространяемые программы, которые могут быть условно разделены на 3 типа: тип А – интенсивно работающие с указателями, тип Б – интенсивно выделяющие и освобождающие оперативную память, тип В – производящие интенсивные математические вычисления. Производилось измерение производительности безопасных программ, безопасность которых обеспечена разработанным средством при использовании различных политик безопасности, а также программ, обработанных при помощи специального программного средства CCured, которое позволяет повысить уровень безопасности программ. Оценка производительности производилась на ЭВМ с процессором Intel Core 2 Duo с тактовой частотой 2 ГГц, снабженной 2 ГБ ОЗУ. Результаты оценки производительности приведены в таблице 4.

Таблица 4 – Результаты оценки производительности безопасного кода

Наименование измеряемой характеристики	Тип программ		
	А	Б	В
Падение производительности небезопасной версии (раз)	1.00	1.00	1.00
Падение производительности при выделении памяти в безопасном пуле без дополнительных проверок (раз)	1.07	0.97	1.03
Падение производительности при проверке записи (раз)	1.61	1.57	1.02
Падение производительности при проверке чтения и записи (раз)	2.36	1.72	1.03
Падение производительности при проверке чтения, записи и обнулении памяти (раз)	3.18	1.93	1.05
Падение производительности при защите с использованием только CCured (раз)	2.04	1.83	1.21

Производительность программ, безопасность которых обеспечена разработанным средством, в большинстве случаев превысила производительность программ, обработанных при помощи других средств, при том, что разработанное средство обеспечивает более высокий уровень безопасности.

В **пятой главе** приводятся примеры практического применения разработанной автором методики защиты гетерогенной среды распределенных вычислений от вредоносного кода вычислительного задания для решения конкретных прикладных задач в двух проектах.

Первый проект представляет собой использование предложенной системы для защиты данных, составляющих коммерческую тайну в АОЗТ «Виланд».

Компания «Виланд» занимается разработкой и продвижением на рынок энергосберегающих технологий. Для моделирования и численного решения некоторых задач компания использует специально разрабатываемое специалистами компании ПО, функционирующее на выделенной ЭВМ. Компания располагает парком ЭВМ различных конфигураций, используемых для хранения и обработки информации, составляющей коммерческую тайну. Значительную часть рабочего дня, а также в ночное время эти ЭВМ простаивают, при этом вычислительная мощность этих ЭВМ могла бы быть использована для решения вычислительных задач путем организации СРВ. Сотрудники, занимающиеся разработкой и сопровождением ПО, функционирующего на выделенной ЭВМ компании, не должны получить доступ к обрабатываемой на остальных ЭВМ конфиденциальной информации, составляющей коммерческую тайну компании. Кроме того, ошибки и сбои в работе вычислительного ПО не должны влиять на работоспособность ЭВМ, задействованных в обработке конфиденциальной информации.

В компании «Виланд» было произведено развертывание СРВ на платформе BOINC. Диспетчер СРВ был установлен на выделенной ЭВМ, которая в то же время функционирует и в роли вычислительного узла СРВ. При этом вычислительные задания для ЭВМ, задействованных в обработке конфиденциальной информации, распространяются в виде исходных кодов на языке Си, проверка безопасности которых производится при помощи разработанной системы защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания.

В результате применения для расчетов семи дополнительных ЭВМ, средняя за неделю производительность вычислений составила приблизительно 410% от производительности одной выделенной ЭВМ до развертывания СРВ. Теоретически, подобный рост производительности вычислений мог бы быть достигнут при закупке и введении в эксплуатацию в качестве выделенных ЭВМ как минимум 3 дополнительных ЭВМ, по параметрам аналогичных используемой на данный момент, кроме того, возникли бы дополнительные расходы, связанные с обслуживанием и амортизацией нового оборудования.

В задачу второго проекта входила разработка подсистемы защиты от вредоносного кода СРВ, объединяющей ЭВМ кафедры «Криптология и дискретная математика» НИЯУ МИФИ. В ходе выполнения курсовых, учебно-исследовательских, дипломных и прочих научных и практических работ на кафедре «Криптологии и дискретной математики» НИЯУ МИФИ (далее – Кафедра) часто возникает потребность в применении высокопроизводительных вычислительных систем, при этом Кафедра

располагает значительным количеством ЭВМ, ресурсы которых могут быть использованы при объединении в СРВ. Отсутствие надежной системы защиты СРВ от вредоносного кода выполняемого задания делало недопустимой организацию СРВ на вывычислительных мощностях кафедры для выполнения вычислительных заданий, разработанных студентами, аспирантами и сотрудниками Кафедры.

Применение разработанной методики защиты СРВ от вредоносного кода выполняемого задания позволило обеспечить надежную защиту ЭВМ Кафедры, делая возможным развертывание и применение СРВ в образовательной и научно-исследовательской деятельности.

В качестве платформы СРВ на Кафедре была выбрана свободно распространяемая платформа с открытым исходным кодом BOINC, разработанная в институте Беркли. На все вычислительные узлы СРВ Кафедры произведена установка разработанной системы защиты СРВ от вредоносного кода вычислительного задания. На ЭВМ, используемых в качестве учебных рабочих станций во время лабораторных работ, не содержится конфиденциальной информации и используется политика безопасности, допускающая чтение неинициализированных и невыделенных областей памяти. На остальных ЭВМ Кафедры используется более жесткая политика безопасности, запрещающая чтение неинициализированных и невыделенных областей памяти.

В **заключении** приведены основные результаты диссертационной работы.

ОСНОВНЫЕ ВЫВОДЫ И РЕЗУЛЬТАТЫ РАБОТЫ

1. Проведен анализ существующих методов защиты гетерогенной среды распределенных вычислений от вредоносного кода, представленных в публикациях. На основе анализа сделан вывод о недостаточности обеспечиваемого этими методами уровня защищенности. С использованием существующих методов и средств практически невозможно решить задачу надежной защиты гетерогенной среды распределенных вычислений без существенного падения производительности вычислений.
2. Построена модель нарушителя и сформулирован перечень угроз гетерогенной среде распределенных вычислений. Предложены меры по предотвращению выявленных угроз.
3. Предложена методика защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания, включающая проведение статической и динамической верификации соответствия выполняемого задания ряду требований, при удовлетворении которым выполняемое задание может считаться безопасным для гетерогенной среды распределенных вычислений.
4. Предложен алгоритм проверки безопасности кода программ.
5. Разработана архитектура системы защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания. Разработанная архитектура обеспечивает возможность применения системы для защиты гетерогенных сред распределенных вычислений, организованных с использованием различных платформ распределенных вычислений.
6. Разработано программное обеспечение на основе архитектуры системы защиты гетерогенной среды распределенных вычислений от вредоносного кода выполняемого задания. Реализованная система позволяет производить верификацию вычислительных заданий на языке Си для гетерогенной среды распределенных вычис-

лений.

7. Разработанное программное обеспечение применено для защиты данных, составляющих коммерческую тайну в АОЗТ «Виланд» и для защиты гетерогенной среды распределенных вычислений, объединяющей ЭВМ кафедры «Криптология и дискретная математика» НИЯУ МИФИ от вредоносного кода вычислительных заданий.

Основные публикации по теме диссертации:

Основные положения диссертационной работы опубликованы в 8 печатных трудах, в том числе 2 статьях в журналах, включенных ВАК РФ в перечень ведущих рецензируемых научных журналов и изданий:

1. Станкевичус А.А. Методика защиты клиентского программного обеспечения peer-to-peer сети от модификации и анализа / А.А. Станкевичус // Технологии Microsoft в теории и практике программирования: Труды Всероссийской конференции студентов, аспирантов и молодых ученых. Центральный регион. Москва, 17 – 18 февраля 2005 г. – М.: МГТУ им. Н.Э.Баумана, 2005. – С. 85.

2. Станкевичус А.А. Разработка комплексного подхода к инженерному анализу исполнимого кода / А.А. Станкевичус // Научная сессия МИФИ-2007. XIV Всероссийская научная конференция «Проблемы информационной безопасности в системе высшей школы». Сборник научных трудов. М.: МИФИ, 2007. – С. 129.

3. Павлова Е.А., Станкевичус А.А. Оптимизация процессов анализа исполнимого кода / Е.А. Павлова, А.А. Станкевичус // Технологии Microsoft в теории и практике программирования: Труды IV Всероссийской конференции студентов, аспирантов и молодых учёных. Центральный регион. Москва, 2 – 3 апреля 2007 г. – М.: Вузовская книга, 2007. – С. 188.

4. Станкевичус А.А. Архитектура системы защиты вычислительного узла Грид от вредоносного кода / А.А. Станкевичус // Технологии Microsoft в теории и практике программирования: Труды V Всероссийской конференции студентов, аспирантов и молодых ученых. Центральный регион. Москва, 1 – 2 апреля 2008 г. – М.: Вузовская книга, 2008. – С. 156 – 157.

5. Ежов Д.А., Станкевичус А.А. Концептуальная модель системы комплексной защиты Грид-сети / Д.А. Ежов, А.А. Станкевичус // Современные технологии в задачах управления, автоматики и обработки информации: Труды XVII Международного научно-технического семинара. Алушта, сентябрь 2008 г. – СПб.: ГУАП, 2008. – С. 229.

6. Мальчуков А.В., Станкевичус А.А. Анализ методов защиты Грид от вредоносного кода / А.В. Мальчуков, А.А. Станкевичус // Современные технологии в задачах управления, автоматики и обработки информации: Труды XVII Международного научно-технического семинара. Алушта, сентябрь 2008 г. – СПб.: ГУАП, 2008. – С. 233.

7. Станкевичус А.А. О некоторых методах защиты Грид от вредоносного кода / А.А. Станкевичус // Безопасность информационных технологий, №3, 2008. – С. 45-48. (Перечень ВАК)

8. Станкевичус А.А. Инструментальное средство проверки безопасности кода, предназначенного для исполнения в Грид-сетях / А.А. Станкевичус // Безопасность информационных технологий, №1, 2009. – С. 58-61. (Перечень ВАК)