На правах рукописи

ПРОХОРОВ Роман Сергеевич

МОНИТОРИНГ И ПОЛИТИКА ОГРАНИЧЕНИЯ ИСПОЛЬЗОВАНИЯ ПРОЦЕССОВ НА ОСНОВЕ АНАЛИЗА ИХ ПОВЕДЕНИЯ

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

АВТОРЕФЕРАТ
диссертации на соискание ученой степени кандидата технических наук

Уфа – 2014

Работа выполнена на кафедре информационной безопасности ФГБОУ ВПО «Омский государственный университет им. Ф. М. Достоевского»

Научный руководитель: доктор физ.-мат. наук, профессор
БЕЛИМ Сергей Викторович

Официальные оппоненты: доктор техн. наук, профессор
ЗАХАРОВ Александр Анатольевич
ФГБОУ ВПО «Тюменский государственный университет»,
заведующий кафедрой «Информационной безопасности»

кандидат техн. наук, доцент
КЛАДОВ Виталий Евгеньевич
«Уфимский государственный авиационный технический университет»,
доцент кафедры «Вычислительной техники и защиты информации»

Ведущая организация: ФГАОУ ВПО «Санкт-Петербургский национальный университет информационных технологий, механики и оптики»,
г. Санкт-Петербург

Защита диссертации состоится 27 февраля 2015 г. в 10:00 часов на заседании диссертационного совета Д-212.288.07 на базе ФГБОУ ВПО «Уфимский государственный авиационный технический университет» по адресу:

450000, г. Уфа, ул. К. Маркса, 12.

С диссертацией можно ознакомиться в библиотеке ФГБОУ ВПО «Уфимский государственный авиационный технический университет» и на сайте www.ugatu.su

Автореферат разослан «___» _____ 2014 г.

Ученый секретарь
диссертационного совета
д.т.н., доцент  И. Л. Виноградова

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы исследования

Одной из главных задач при обеспечении информационной безопасности, является мониторинг процессов, который в свою очередь требует наличия надежной идентификации процессов. Прежде всего, идентификация процессов необходима при построении списка задач, выполняемых компьютерной системой. Анализ списка задач позволяет выявлять вредоносные и скрытые процессы. Также необходима идентификация процессов при реализации службы аудита, так как запись о каждом действии в системе должна содержать имя субъекта, выполнившего доступ.

Второй задачей, составляющей политику разграничения доступа в компьютерных системах, является ограничение использования программ Все современные операционные системы содержат механизмы ограничения возможности запуска тех или иных процессов для каждого из пользователей. В операционных системах семейства Linux данный механизм реализован с помощью запрета на исполнение файла для пользователя либо группы пользователей. В операционных системах семейства Windows реализована встроенная служба запрета запуска определенных процессов (Software restriction policies, SRP), позволяющая установить запрет на запуск по имени файла либо по хеш-значению исполняемого кода. Более совершенная реализация ограничения на запуск процессов присутствует в комплексных системах защиты информации в виде замкнутых программных сред.

На сегодняшний день в прикладных задачах применяется три основных подхода к идентификации исполняемого кода: по имени исполняемого файла-источника, по хэш-значению исполняемого файла-источника, по данным из заголовка исполняемого файла-источника. Во всех случаях идентификация процесса происходит по исполняемому файлу-источнику, а не по контексту самого процесса. Вследствие этого всем стандартным методам идентификации присущ общий набор недостатков. Во-первых, их достаточно легко обойти злоумышленнику с помощью модификации файла-источника. В первом случае достаточно изменить имя файла, а в двух других незначительные изменения содержимого исполняемого файла. Проблема может быть решена в рамках разделения доступа с помощью запрета на модификацию файла. Однако пользователь-злоумышленник может вносить в системы свои модифицированные файлы и запускать их на исполнение. Во-вторых, идентификация происходит исключительно до создания процесса. Все современные операционные системы позволяют заменить исполняемый код в рамках контекста одного процесса. Таким образом злоумышленник может запускать легитимный исполняемый код и подменять его нелигитимным в процессе выполнения. Это касается политики, организованной по принципу как «черного списка», так и «белого».

Более того, известно немало фактов эксплуатации уязвимостей операционных систем с целью обхода ограничений безопасности, реализуемых SRP, AppLocker, UAC и т.п.. Для Windows наибольшую известность получили следующие уязвимо-

сти: злоупотребление runas, iKat CreateProcess, DLL Injection, Process-In-Memory-Patching и др. Стоит также отметить, что в в сети Интернет можно легко найти программное обеспечение, эксплуатирующее соответствующие уязвимости (например, iKat).

Одним из способов устранения недостатков традиционных методов идентификации исполняемого кода является дополнительная динамическая идентификация процессов в период их исполнения по действиям, производимым в системе. Идентификация процессов по их поведению позволяет повысить эффективность следующих встроенных средств защиты информации: система ограниченного запуска программ Windows, служба обеспечения замкнутой программной среды в рамках комплексных систем защиты информации, монитор процессов для операционных систем Windows, служба аудита действий в системе.

На основе динамической идентификации процессов может быть реализован монитор процессов, позволяющий снизить риски следующих действий злоумышленника:

1. Запуск модифицированного исполняемого кода, запрещенного к выполнению в системе политикой ограниченного запуска программ Windows.
2. Нарушение замкнутой программной среды в рамках комплексной системы защиты информации путем запуска исполняемого кода с разрешенным именем и запрещенным кодом.
3. Попытка запуска запрещенного кода путем изменения имени и хэш-значения исполняемого файла. Задача предотвращения данной атаки решается путем сравнения списка процессов, формируемого стандартным диспетчером задач, со списком монитора процессов, формируемым на основе динамической идентификации процессов.
4. Попытка обхода системы аудита с помощью изменения имени процесса. Данная угроза может быть выявлена и предотвращена при сравнении стандартной службы аудита ОС Windows и записей динамического монитора процессов.

Таким образом, актуальной является задача разработки методики динамической идентификации процессов для создания дополнительных средств разграничения запуска программ и аудита действий процессов в системе. Также актуальным является реализация дополнительных средств защиты на базе динамической идентификации процессов.

Степень разработанности темы исследования

Области разработки методов и средств мониторинга процессов посвящены работы Ю.А. Брюхомицкого, Т.Р. Кашаева, В.В. Костюкова, Е.Н. Крючкова, А.Ю. Оладько, Г.А. Самигулиной и др. Стоит отметить, что методы распознавания аномального поведения, а также методы, применяемые в системах обнаружения вторжений, рассмотренные в работах С.В. Безобразова, С.С. Валеева, В.И. Васильева, М.Ю. Дьяконова, Д.М. Клионского, А.В. Корюкалова, В.Ф. Шаньгина и др., очень схожи с методами мониторинга процессов. Кроме того, имеются работы по системам мониторинга и обнаружения аномального поведения у зарубеж-

ных исследователей: J. Bhangoo, M. Damashek, E. Eskin, S. Forrest., A.K. Ghosh, P. Helman, S. Hofmeyr, W. Lee, M. Schatz, A. Schwartzbard, C. Warrender и др.

Предлагаемые в большинстве работ решения используются для детектирования вредоносных и аномальных активностей в системе, а также для осуществления мониторинга действий пользователей и процессов. На данный момент центральной проблемой остается задача формирования образа (профайла) нормального поведения или образа поведения целевого процесса. Существует множество подходов к формированию этого образа. Однако многие из них не позволяют осуществлять анализ в реальном времени из-за жестких критериев оценки «нормальности», которые могут быть получены только после завершения процесса или высокой стоимости необходимых вычислений. Также часто не учитывается тот факт, что большое число процессов в современных операционных системах являются многопоточными, что делает затруднительным анализ четких последовательностей событий или системных вызовов.

Таким образом, создание легковесной системы, позволяющей осуществлять мониторинг одно- и многопоточных процессов в реальном времени, остается актуальной задачей.

Цель работы

Основной целью диссертации является повышение безопасности операционных систем с помощью дополнительных средств защиты информации на основе идентификации процессов по их поведению.

Задачи исследования

1. Разработка алгоритмов фиксирования поведения процессов в системе в реальном времени.
2. Разработка методики идентификации процессов на основе анализа собранной информации об их поведении (динамического образа).
3. Разработка системы мониторинга и аудита процессов на основе их поведения в компьютерной системе.
4. Разработка системы ограничения использования программ по их поведению в компьютерной системе.
5. Разработка и реализация программного комплекса динамического мониторинга и аудита процессов, а также ограниченного использования программ и исследование эффективности работы программного комплекса.

Объект исследования

Объектом исследования являются процессы операционной системы семейства Windows и порождаемые ими события в компьютерной системе. Выбор операционной системы Windows обусловлен наличием в ней штатных механизмов генерации «событий» (events) при обращении процессов к системным службам.

Предмет исследования

Предметом исследования является эффективность систем защиты информации, базирующаяся на идентификации процессов в операционных системах семейства Windows.

Методы исследования

Для решения данного комплекса задач были использованы методы теории нейронных сетей, методы принятия решений, методы объектно-ориентированного программирования, методы формирования замкнутой программной среды, методы аудита действий в системе, а также методы ограниченного использования программ.

Основные положения, выносимые на защиту

1. Алгоритмы фиксирования поведения процесса в операционной системе и формирования динамического образа процесса, позволяющие получать информацию о его действиях в реальном времени (п. 11. паспорта специальности 05.13.19).
2. Методика идентификации процесса по его динамическому образу на основе системы нейронных сетей, устойчивая к нефункциональным вставкам и чувствительная к изменению поведения (п. 11).
3. Система мониторинга и аудита процессов на основе их поведения в операционной системе, позволяющая следить за процессами на протяжении всей активности и выявлять скрытые процессы (п. 13, 14).
4. Система ограничения использования программ по поведению процессов в операционной системе, контролирующая использование процесса не только в момент запуска, но и во время его работы (п. 13).
5. Программный комплекс, реализующий разработанные системы мониторинга процессов, аудита процессов и ограниченного использования программ и повышающий эффективность системы защиты компьютерных систем под управлением ОС Windows (п. 11, 14).

Научная новизна

1. Разработаны алгоритмы фиксирования поведения процесса в операционной системе. Новизна состоит в том, что образ процесса формируется и анализируется без остановки процесса, но при этом обладает всеми необходимыми признаками для идентификации процесса. Отличие предложенного алгоритма от традиционного состоит в том, что анализируется сам процесс, а не исполняемый файл-источник процесса.
2. Разработана методика идентификации процесса по его динамическому образу на основе системы нейронных сетей. Новизна состоит в использовании специализированного анализатора, включающего модуль предобработки последовательности событий, инициированных процессом, а также структуры нейронных сетей с модулем согласования выходов, ориентированного именно на рас-

познавание динамических образов процессов. Отличие предложенной методики состоит в анализе действий, выполняемых программой, а не ее исполняемого кода.

3. Разработана система мониторинга и аудита процессов на основе их поведения в операционной системе. Новизна состоит в том, что система способна отслеживать и идентифицировать процессы на всем протяжении их «жизни», в отличие от традиционного диспетчера процессов, который идентифицирует процессы только в момент их запуска, а также встроенных систем аудита, которые используют идентификатор процесса, присвоенный ему при запуске, во всех записях о процессе.
4. Разработана система ограничения использования программ по поведению процессов в операционной системе. Новизна состоит в том, что ограничение использования программы может быть реализовано не только в момент запуска процесса, но и на любом этапе его выполнения, если поведение процесса становится аномальным.
5. Разработан программный комплекс, реализующий разработанные системы мониторинга процессов, аудита процессов и ограниченного использования программ, который повышает эффективность системы защиты компьютерных систем под управлением ОС Windows; разработанный программный комплекс зарегистрирован в «Фонде Алгоритмов и Программ Российской Академии Наук» под серийным номером **PR13028** 11 июля 2013 года.

Практическая и научная значимость результатов

Практическая значимость результатов заключается в разработке метода сбора достаточной для анализа информации о поведении процессов в реальном времени, методов идентификации процессов по поведению, монитора процессов и системы аудита, позволяющих идентифицировать и отслеживать процессы не только в момент их порождения, но и на всем протяжении их работы. Также предложена система ограниченного использования программ на основе динамической идентификации процессов, позволяющая устранять ряд недостатков традиционных систем подобного назначения.

Научная значимость результатов состоит в разработке методики идентификации процессов по их динамическому образу, которая может быть применена как в задачах защиты информации, так и в системах диспетчеризации процессов в современных операционных системах.

Результаты внедрены на двух предприятиях.

Апробация работы

Основные результаты диссертации докладывались и обсуждались на следующих научных конференциях: XVI Международная научная конференция «Решетневские чтения» (Красноярск, 2012); Международная научно-практическая конференция «Актуальные вопросы в научной работе и образовательной деятельности» (Тамбов, 2013); V Всероссийская научно-практическая конференция «Информационные технологии и автоматизация управления» (Омск, 2013); V Межрегиональ-

ная научно-практическая конференция «Информационная безопасность и защита персональных данных: проблемы и пути их решения» (Брянск, 2013); I Международная научно-практическая конференция «Информационная безопасность в свете стратегии Казахстан-2050» (Астана, 2013), научная конференция «Математическое и компьютерное моделирование» (Омск, 2013).

Степень достоверности результатов работы

Достоверность основных результатов и положений диссертации подтверждается адекватностью применяемых методов и данными, полученными в процессе тестирования разработанных методов и алгоритмов.

Соответствие диссертации паспорту специальности

Диссертация соответствует следующим пунктам паспорта специальности 05.13.19:

11. Технологии идентификации и аутентификации пользователей и субъектов информационных процессов. Системы разграничения доступа.

В диссертации предложен метод и разработан алгоритм идентификации процессов системы. На основе разработанного алгоритма разработан программный комплекс, позволяющий реализовать разграничение прав на использование программ.

13. Принципы и решения (технические, математические, организационные и др.) по созданию новых и совершенствованию существующих средств защиты информации и обеспечения информационной безопасности.

Предложены системы мониторинга и аудита процессов, позволяющие повысить эффективность слежения за процессами в операционных системах, выявляющие скрытые процессы и совершенствующие существующие методы формирования замкнутой программной среды. Также предложена система ограничения использования программ по их поведению в операционной системе.

14. Модели, методы и средства обеспечения внутреннего аудита и мониторинга состояния объекта, находящегося под воздействием угроз нарушения его информационной безопасности.

Разработанный метод идентификации процессов повышает эффективность аудита действий в системе, так как позволяет отслеживать процессы, скрывающиеся от стандартного диспетчера процессов.

Публикации

Материалы диссертации опубликованы в 11 печатных изданиях, из них 3 статьи в журналах из списка, рекомендованного ВАК.

Структура и объем диссертации

Диссертация содержит: введение, 3 основные главы, заключение и библиографический список. Общий объем диссертации 113 страниц, в диссертации присутствует 17 рисунков и 15 таблиц. Библиографический список содержит 128 источников.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении рассматривается актуальность темы, ставятся цели и определяются объекты и методы исследования. Указывается новизна, практическая и научная значимость результатов, выносимых на защиту. Дается краткая характеристика содержания диссертации.

Первая глава является обзорной. В данной главе описываются основные методы и подходы к обнаружению вторжений по наличию аномального поведения в системе. Рассматриваются как детерминированные, так и интеллектуальные методы. В заключении главы формируются цель и задачи исследования.

Во второй главе описывается архитектура разработанного программного комплекса, позволяющего осуществлять динамическую идентификацию процессов в ОС на примере Windows. На рисунке 1 изображена блок-схема, отражающая эту архитектуру.

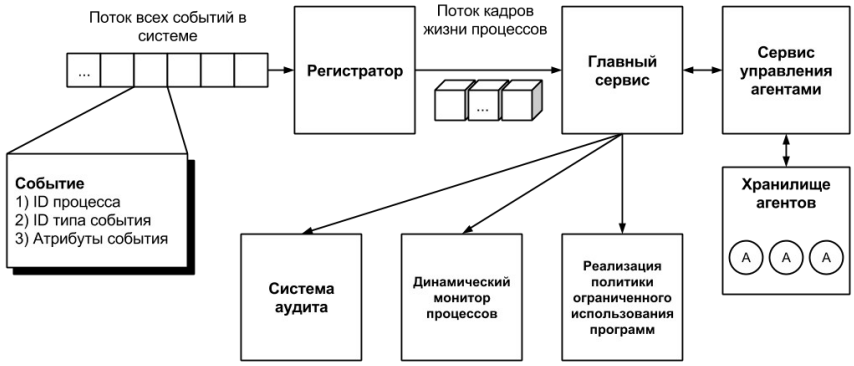

Рисунок 1 — Блок-схема программного комплекса

Модуль *регистратор* подключается к потоку событий уровня ядра. Каждое событие имеет статический идентификатор процесса, породившего его (идентификатор присваивается операционной системой; в Windows традиционно — PID), тип события и специфические для каждого типа атрибуты. События распределяются по контейнерам (кадрам «жизни») с помощью *алгоритма обработки входящего потока событий.* Поток событий в системе делится на множество подпотоков событий, каждый из которых соответствует отдельно взятому активному процессу. Как только в одном из таких подпотоков набирается *n* событий, подается сигнал «контейнер готов» и подпоток очищается. Таким образом, поток абсолютно всех событий в системе преобразуется во множество подпотоков контейнеров событий равной длины (рисунок 2).

Поток контейнеров с помощью *главного сервиса* передается на анализ *сервису управления агентами*, который отвечает за их создание, обучение, сохранение, получение и удаление. *Агент (анализатор)* — составной модуль, имеющий несколько небольших компонентов. Основной задачей данного модуля является распознавание известных типов процессов по принципу «свой/чужой».

Система аудита, *динамический монитор процессов*, а также *реализация политики ограниченного использования программ* представляют собой пользовательский интерфейс соответствующего функционала.

Сервис управления агентами обучает на очередном контейнере событий из потока все агенты, проходящие обучение на момент поступления данного контейнера, а также передает его для анализа всем обученным активным агентам.

Агент (и обучающийся, и активный) разбивает каждый контейнер на поток малых контейнеров с помощью *алгоритма обработки контейнеров событий.*

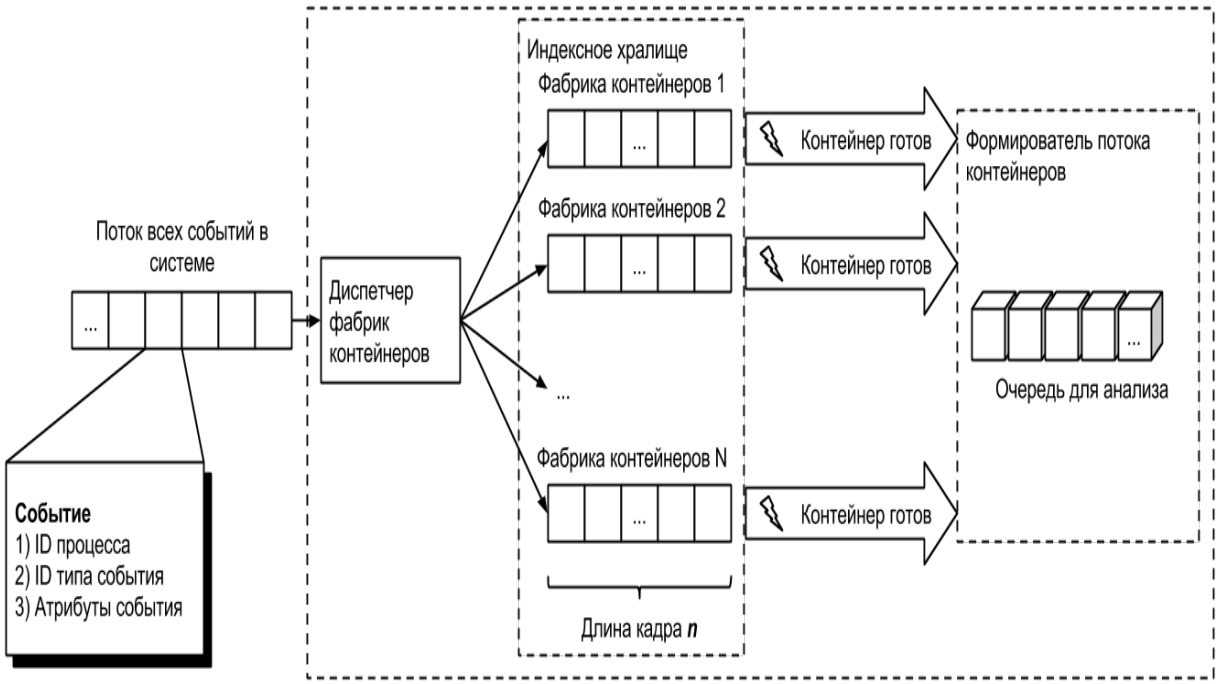

Рисунок 2 — Реализация алгоритма обработки потока событий

На рисунке 3 приведен пример преобразования согласно представленному алгоритму. Каждому типу события *C* присваивается группа *G*. Каждой группе соответствует один малый контейнер (число групп равно числу малых контейнеров в наборе, выделено четыре группы: дисковые, файловые операции, операции с реестром, а также сетевая активность). Иными словами, все события делятся по *m* группам, каждая из которых упаковывается в свой контейнер (группировка). Далее каждое событие каждого малого контейнера подвергается преобразованию в укрупненный класс события *S* и подсчитывается количество таких событий внутри укрупненного класса. Таким образом, каждый малый контейнер транслируется в вектор, компоненты которого отражают количество событий соответствующего укрупненного класса событий (трансляция), и входными данными для анализатора является *набор векторов*. Далее, векторы нормализуются (формула 1).

Данные преобразования осуществляются исходя из того, что для идентификации процессов интерес представляют не сами события, которые содержат большой объем частной информации, затрудняющей анализ, а классы событий. Также существенно снижается число значащих типов событий.

$$x_i' = \frac{x_i}{\sqrt{\sum_{i=1}^{n} x_i^2}}. \qquad (1)$$

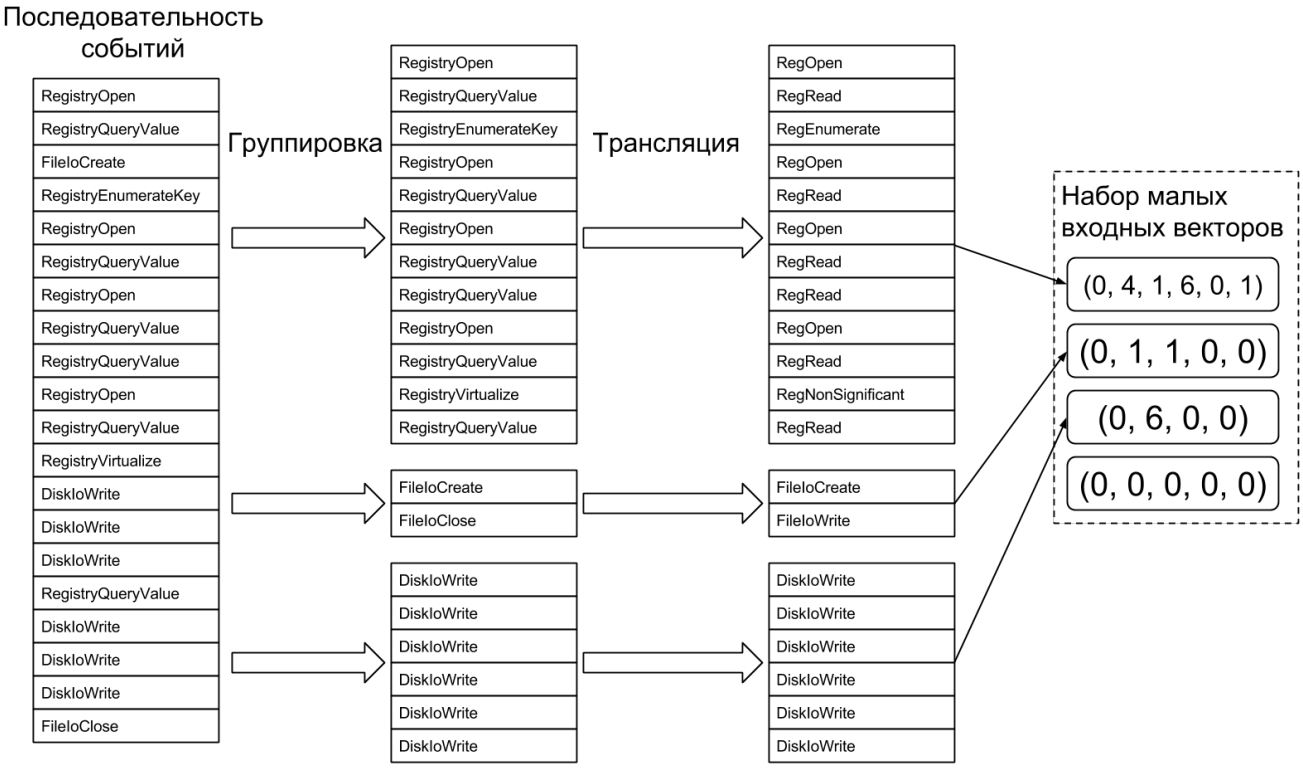

Рисунок 3 — Блок-схема алгоритмов преобразования (пример)

Далее дается описание низкоуровневых типов событий, которые предоставляет используемая библиотека Event Trace for Windows.

После сбора информации о событиях, инициированных процессом и формирования потока контейнеров, данные (набор входных векторов) поступают в анализатор. Каждый анализатор событий не принимает решение, а только предоставляет информацию другим модулям, которые и будут «выносить вердикт» анализируемому процессу (алгоритм использования множества анализаторов). При использовании такой архитектуры память (весь набор *динамических образов*) благополучно диссоциируется по компонентам анализатора следующего уровня. Данное свойство памяти называется *ассоциативностью*. Это позволяет легко заменять неэффективные анализаторы. Данный механизм очень схож по своей сути с механизмом иммунной памяти в естественных иммунных системах. Каждый анализатор имеет некую область стимуляции, в которой он способен распознавать множество слаборазличимых процессов и принимать их за «свои».

Поведенческим идентификатором (BID, Behavior ID) предлагается называть идентификатор известного динамического образа процесса.

Анализатором, связанным с BID, предлагается называть компонент системы анализа, который способен отвечать на вопрос «имеет ли процесс, кадр жизни которого анализируется, поведенческий идентификатор BID». Достаточными для анализа данными считается полный кадр времени жизни процесса — контейнер. Выходными данными является ответ на поставленный вопрос о соответствии связанному образу. Каждый агент — анализатор, связанный с BID.

Основной идеей является то, что каждый компонент этого набора анализаторов обучается распознавать конкретный класс процессов с одним BID. Данная идея базируется на принципе действия совокупности анализаторов, каждый из которых способен опознавать один процесс.

Структура модуля анализа приведена на рисунке 4. Входными данными являются четыре вектора размерностей 4, 5, 5 и 6. Используется вектор доверия $T=(t_1, t_2, \ldots, t_m)$, вычисляемый на втором шаге обучения (см. ниже).

Алгоритм обучения набора нейронных сетей. Для обучения системы из четырех нейронных сетей формируется обучающее множество наборов малых входных векторов. Каждый из таких наборов имеет ожидаемый результат: процесс «свой» (ожидаемый вектор $ex=(1,0)$) или процесс «чужой» (ожидаемый вектор $ex=(0,1)$). Все обучающее множество равномерно делится случайным образом на две примерно равные части: собственно, обучающее множество и тестирующее множество. Обучающее множество используется для настройки весов каждой нейронной сети внутри системы, при этом ожидаемый результат для набора входных векторов единый. Обучение проводится с помощью алгоритма обратного распространения ошибки. Тестирование системы нейронных сетей производится с целью установления уровня доверия каждой из них: формируется вектор доверия $T=(t_1, t_2, \ldots, t_m)$.

$$t_i=\begin{cases}0, \text{если } n_i=0,\\ \dfrac{s_i}{n_i}, \text{если } n_i>0,\end{cases} \qquad (2)$$

где s_i — число успешных идентификации *i*-й нейронной сети, n_i — общее число попыток идентификации.

Алгоритм использования структуры нейронных сетей. Обученной структуре нейронных сетей на вход подается набор входных векторов. Если каждая нейронная сеть считает свой результат в виде вектора (r_i, l_i), а финальный ответ вычисляется по формуле 3.

$$res=(r_1t_1+r_2t_2+\ldots+r_nt_n)-(l_1t_1+l_2t_2+\ldots+l_nt_n)=\sum_{i=1}^{n}((r_i-l_i)t_i). \qquad (3)$$

Если $res>0$, то считается, что процесс «свой» — в противном случае считается, что процесс «чужой». Однако, для более эффективной идентификации предлагается использование «желтой» и «красной» границ для каждой сессии работы агента: *P1* и *P2* > *P1*. В случае, когда $res<P1$, считается, что процесс «чужой».

Если $P1 \le res < P2$, то считается, что процесс скорее всего является своим. В таком случае необходимо подтверждение администратора. Такое событие считается достижением «желтой» границы. Если $res \ge P2$, то считается, что процесс определенно является «своим». В таком случае считается, что достигнута «красная» граница.

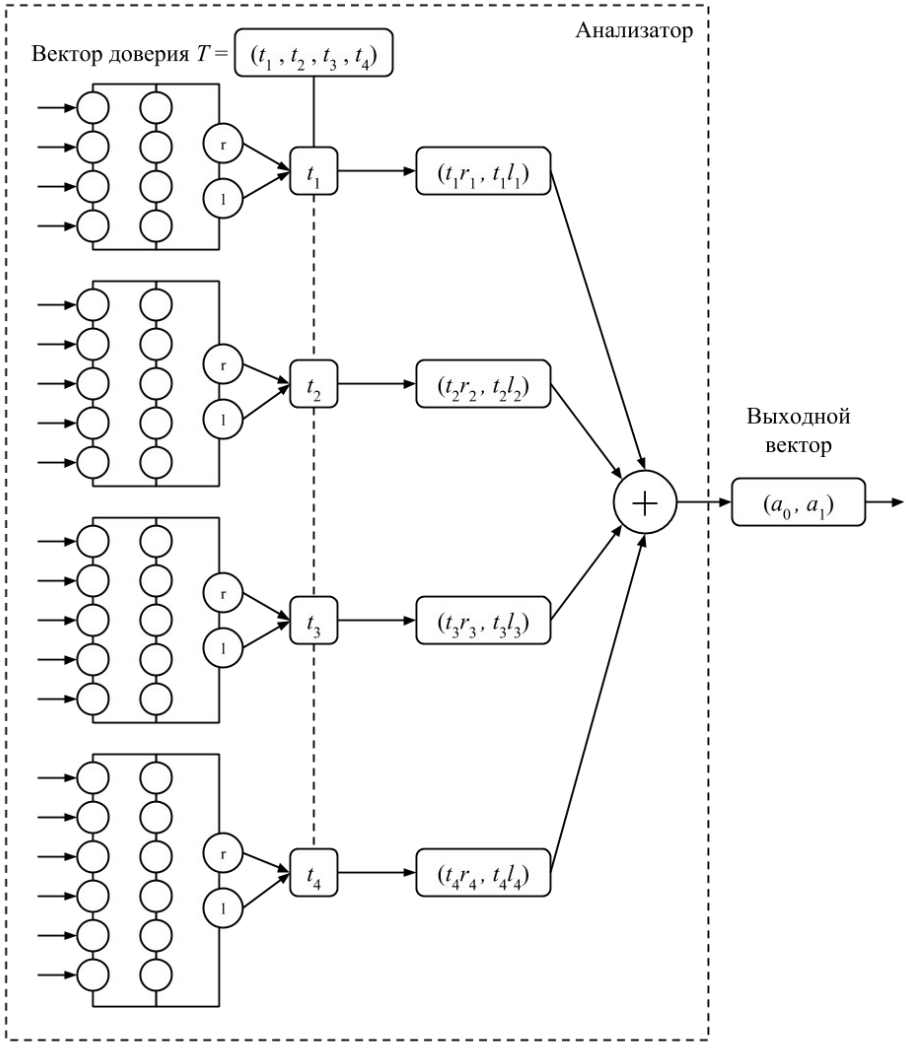

Рисунок 4 – Блок-схема анализирующего модуля

Алгоритм получения результата проверки набора векторов. С помощью набора обученных анализаторов возможно идентифицировать активные процессы. Из множества анализаторов $\{A\}$ выбирается анализатор $A_0 \in \{A\}$, который имеет

$$max_i\{res_i - P1_i\}, если\ max_i\{res_i - P1_i\} > 0 . \quad (4)$$

В противном случае $max_i\{res_i - P1_i\} \le 0$ считается, что ни один агент не среагировал, и процесс не опознан. В качестве P1 и P2 предлагается использовать значения, вычисляемые по формулам 5 и 6:

$$P1 = m\frac{\sum_{i=1}^{k} res_i^{(1)}}{k(m+k)} + k\frac{\sum_{i=1}^{m} res_i^{(0)}}{m(m+k)}, \quad (5)$$

$$P2 = \frac{\sum_{i=1}^{k} res_i^{(1)}}{k}, \quad (6)$$

где $res_i^{(1)}$ и $res_i^{(0)}$ — результаты вычисления анализирующей структуры для «своего» и «чужого» кадра на i-м этапе тестирования, а k и m – число «своих» и «чужих» кадров в обучающей выборке.

Далее устанавливается, что построенная система является мультиагентной.

В третьей главе дается описание динамического монитора процессов, системы аудита, а также реализации политики ограниченного использования программ, разработанных для повышения защищенности рабочей станции. Также приводятся результаты тестирования программного комплекса.

Комплекс содержит три приложения, демонстрирующих работу с основными компонентами разработанной методики: клиентское приложение с графическим интерфейсом, клиентское приложение с консольным интерфейсом и служба Microsoft Windows. Исходный код комплекса (реализован на ЯВУ C#) имеет полный свободный доступ.

IncinerateService – основное приложение, которое регистрируется как сервис (служба) Windows. Данная служба проектировалась таким образом, что администратор может управлять политиками и отслеживать активности процессов на компьютере пользователя со своего рабочего места, используя клиентское приложение, установленное на компьютере пользователя.

IncinerateCmd представляет собой консольное клиентское приложение, работающее с сервисом и подающее ему соответствующие команды. Взаимодействие осуществляется через pipe (WCF, SOAP). Приложение вызывается через командную строку Windows и завершается после ответа сервиса.

IncinerateUI представляет собой клиентское приложение с графическим интерфейсом (UI), позволяющее полностью управлять сервисом *Incinerate*.

Для запрещения использования какого-либо процесса с помощью клиента необходимо установить службу на рабочую станцию пользователя и с помощью клиентского приложения создать новое правило для определенного активного процесса. Как только наберется 500 образцов поведения целевых процессов, он будет сохранен в репозитории агентов. С помощью команды слежения, выбранный агент подключается к потоку кадров жизни процессов в операционной системе. Как только будет обнаружен процесс, имеющий схожее поведение, он будет завершен немедленно, через некоторое время или по ответу администратора — в зависимости от достигнутой границы и заданных стратегий.

Динамический монитор процессов предоставляет функционал по мониторингу активности всех процессов в системе. В отличие от стандартного диспетчера задач Windows, динамический монитор отображает процессы согласно их активности, что позволяет видеть процессы, способные «скрываться» от диспетчера задач: это клавиатурные шпионы, трояны, черви и т. п. Например, Ardamax Keylogger невидим таким диспетчерам задач, как Process Explorer и Windows Task Manager, однако при первой сетевой/реестровой/файловой активности в мониторе появится запись, сообщающая о наличии действий процесса с неустановленным именем. Динамический монитор также позволяет просматривать подробную активность каждого замеченного процесса: параметры чтения/модификации реестра Windows, номера портов и IP-адреса, имена файлов, с которыми работал процесс. Каждая запись, кроме данных об активности процесса, содержит имя процесса, идентификатор процесса в ОС (PID), а также поведенческий идентификатор (BID), характеризующий поведение процесса на основе данных, предоставленных анализатором.

Система аудита. С помощью подключения к регистратору событий была организована система сохранения логов активности процессов в формате XML.

Для оценки применимости методов и алгоритмов проводится ряд экспериментов. **Эффективность** (E) в данных экспериментах определяется как отношение числа успехов к общему числу анализируемых кадров, выраженная в процентах.

$$E = \frac{s}{n} \cdot 100\,\% \tag{7}$$

Исходные данные — кадры жизни всех процессов, собранных за несколько запусков операционной системы (около 7870000 событий).

Определение эффективной длины кадра жизни. Исследованы длины от 20 до 100 элементов (шаг 10). Было установлено, что эффективность анализирующего модуля **не меняется** при снижении размера контейнера. Так как необходимо обеспечить максимальную скорость сбора данных для анализа, то наиболее эффективное значение длины равно 20 — минимальному из возможных.

Определение эффективности одиночного анализатора. Использовались равные наборы кадров различных процессов. Из множества наборов выделялся один набор, принадлежащий процессу, который считался «своим» в одиночном эксперименте. Остальные наборы считались «чужими». Контейнеры равномерно делились случайным образом на 3 подмножества: обучающее, тестирующее и проверяющее. Анализируя полученные данные, можно выделить группу процессов, эффективность идентификации которых была наибольшей по сравнению с другими исследуемыми процессами (**80 — 99)%**.

Некоторые, участвовавшие в исследовании процессы, дали результат говорящий об их неразличимости: используемые кадры содержали похожие последовательности событий, чем было подтверждено предположение о слабой различимости процессов, порождающих одинаковые события. Более того, большинство процессов, для которых анализатор оказался слабоэффективным, являлись *интерактивными*. Поведение таких процессов сильно зависит от действий пользователя.

Таким образом, данная методика более эффективна для процессов, действующих без участия пользователя, что ориентирует ее на детектирование подозрительных активностей в замкнутых программных средах.

Тестирование устойчивости к нефункциональным вставкам. Был создан специальный процесс Z. (хэш H1). После добавления нефункциональной вставки в его код, хэш изменился (H2). Обучен новый агент на запущенном процессе. После установки агента в режим слежения, модифицированный процесс был обнаружен. Таким образом, показано, что методика устойчива к нефункциональным модификациям исходного кода программы в отличие от политики ограниченного использования программ Windows с идентификацией по хэш-значению.

Тестирование устойчивости при подмене легитимного исполняемого файла. В системе было разрешено использование процесса V. Исполняемый файл Z был запущен на исполнение с именем V. Активный агент, обученный на поведении процесса Z, успешно обнаружил схожее поведение и завершил процесс V, в отличие от политики ограниченного использования Windows, позволившей запустить процесс V. В диспетчере задач Windows процесс отображался под именем V.

Тестирование устойчивости при изменении имени запрещенного процесса. В системе было запрещено использование процесса Z. С помощью политики ограниченного использования программ Windows файл был запрещен к запуску по имени Z. После переименования файла Z в Y он был успешно запущен в обход политики. Активный агент, обученный на поведении процесса Z, обнаружил поведение, присущее Z, у процесса Y, и завершил его. Таким образом, показано, что методика устойчива к изменению имени запрещенного процесса.

Тестирование чувствительности к функциональным модификациям исходного кода. В системе было разрешено использование процесса V. В процесс V был внедрен изменяющий функциональность вредоносный код. Агент, обученный на поведении V, обнаружил изменение поведения. Послано сообщение администратору. Этим тестом доказывается чувствительность к функциональным модификациям.

Результаты тестирования программного комплекса показали, что динамическая идентификация может быть использована в качестве идентифицирующего подхода при организации политики ограниченного использования программ для большинства процессов.

Перспективы дальнейшей разработки темы.

В рамках дальнейших исследований планируется снижение количества ошибок первого и второго рода путем использования статистических классификаторов при анализе последовательностей нескольких кадров жизни процесса.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ И ВЫВОДЫ

1. Разработаны алгоритмы фиксирования поведения процессов в системе, которые в отличие от существующих аналогов позволяют собирать информацию о поведении процесса без его остановки в реальном времени: алгоритм сбора информации о поведении процесса в виде потока событий в операционной си-

стеме, алгоритм обработки потока событий и обработки потока контейнеров событий.

2. Разработана методика идентификации процессов на основе анализа собранной информации об их поведении (динамического образа) с помощью набора нейронных сетей, которая в отличие от стандартных способов идентификации использует динамический образ процесса, а не статические признаки исполняемого файла, что позволяет сделать идентификацию процесса устойчивой к нефункциональным изменениям файла-источника, а также обнаруживать аномальное поведение при сохранении статических признаков файла-источника.
3. Разработана система мониторинга и аудита процессов на основе их поведения в компьютерной системе, которая в отличие от стандартного диспетчера задач идентифицирует процесс на протяжении всей его активности, что позволяет фиксировать факт изменения поведения процесса, а также выявлять процессы, скрывающиеся от стандартного диспетчера задач Windows; предложенная система не является альтернативной и может использоваться вместе со стандартной.
4. Разработана мультиагентная система ограничения использования программ по их поведению в компьютерной системе, которая в отличие от стандартной системы контролирует использование не только в момент запуска процесса, но и на любом этапе его выполнения, если поведение процесса становится аномальным, при этом предложенная система позволяет идентифицировать процессы без их остановки и также не является альтернативной, поэтому может быть использована в комбинации со стандартной.
5. Разработан и реализован программный комплекс динамического мониторинга, аудита процессов и ограниченного использования программ. В рамках компьютерного эксперимента исследована эффективность работы программного комплекса, тестирование показало его высокую эффективность для неинтерактивных процессов.

ПУБЛИКАЦИИ АВТОРА ПО ТЕМЕ ДИССЕРТАЦИИ

Журналы из перечня ВАК

1. Прохоров Р. С. Выявление процессов с аномальным использованием ресурсов по общему состоянию системы / Р. С. Прохоров // Вестник Омского университета. 2012. №2 (64). С. 191 — 193.
2. Белим С. В. Идентификация процессов по их поведению в компьютерной системе / С. В. Белим, Р. С. Прохоров // Проблемы информационной безопасности. Компьютерные системы. 2013. № 2 (2). С. 7 — 12.
3. Прохоров Р. С. Методика бихевиористического распознавания процессов в операционной системе Windows 7 / Р. С. Прохоров // Вопросы защиты информации. 2013. №4. С. 54 — 57.

Другие издания

4. Прохоров Р. С. Бихевиористический анализ процессов на основе мультинейронной сети. / Р. С. Прохоров // Решетневские чтения: материалы XVI Международной научной конференции. 2012. С. 675.
5. Прохоров Р. С. Бихевиористическая идентификация процессов / Р. С. Прохоров // Математические структуры и моделирование. 2013. №1 (27). С. 103 —112.
6. Прохоров Р. С. Методика бихевиористической идентификации субъектов в функциональной среде. / Р. С. Прохоров // Актуальные вопросы в научной работе и образовательной деятельности: материалы конференции. 2013. Ч. 6. С. 116 — 117.
7. Прохоров Р. С. Методика бихевиористической классификации процессов в операционной системе Windows 7. / Р. С. Прохоров // Информационные технологии и автоматизация управления: материалы научно-практической межвузовской конференции. 2013. С. 163 — 165.
8. Прохоров Р. С. Идентификация образа процесса по его поведению с помощью нейронных сетей / Р. С. Прохоров // Информационная безопасность и защита персональных данных: проблемы и пути их решения: материалы всероссийской научно-практической конференции. 2013. С. 74 — 77.
9. Прохоров Р. С. Incinerate. Анализатор поведения процессов [Электронный ресурс] / Р. С. Прохоров // Фонд алгоритмов и программ. 2013. URL: http://fap.sbras.ru/node/3978
10. Прохоров Р. С. Использование методики бихевиористической классификации субъектов в функциональных средах для идентификации процессов в операционной системе Windows 7 / Р. С. Прохоров // Информационная безопасность в свете стратегии Казахстан-2050: 2013: сборник трудов международной научно-практической конференции. С. 478 — 483.
11. Прохоров Р. С. Идентификация процессов в операционной системе WINDOWS 7 по их поведению / Р. С. Прохоров // Математическое и компьютерное моделирование: сборник материалов научной конференции. 2013. С. 87 — 89.

Диссертант  Прохоров Р. С.

ПРОХОРОВ Роман Сергеевич

МОНИТОРИНГ И ПОЛИТИКА ОГРАНИЧЕНИЯ ИСПОЛЬЗОВАНИЯ ПРОЦЕССОВ НА ОСНОВЕ АНАЛИЗА ИХ ПОВЕДЕНИЯ

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

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук